

ChibiOS/HAL

9.0.0

Reference Manual

Sat Sep 13 2025 08:10:14

1 Introduction	1
2 Deprecated List	3
3 Topic Index	5
3.1 Topics	5
4 Hierarchical Index	7
4.1 Class Hierarchy	7
5 Data Structure Index	11
5.1 Data Structures	11
6 File Index	17
6.1 File List	17
7 Topic Documentation	23
7.1 Compiler portability.	23
7.1.1 Detailed Description	23
7.1.2 Macro Definition Documentation	24
7.1.2.1 CC_SECTION	24
7.1.2.2 CC_WEAK	24
7.1.2.3 CC_USED	24
7.1.2.4 CC_ALIGN_DATA	24
7.1.2.5 CC_ALIGN_CODE	25
7.1.2.6 CC_PACK	25
7.1.2.7 CC_NO_INLINE	25
7.1.2.8 CC_FORCE_INLINE	25
7.1.2.9 CC_NO_RETURN	26
7.1.2.10 CC_ROMCONST	26
7.1.2.11 CC_LIKELY	26
7.1.2.12 CC_UNLIKELY	26
7.2 HAL	27
7.2.1 Detailed Description	27
7.2.2 HAL Device Drivers Architecture	27
7.2.3 HAL Normal Device Drivers	27
7.2.3.1 Diagram	28
7.2.4 HAL Complex Device Drivers	28
7.2.5 HAL Interfaces	28
7.2.6 HAL Inner Code	28
7.2.7 Peripheral Interfaces	29
7.2.7.1 Detailed Description	29
7.2.7.2 Generic Flash Interface	29
7.2.8 Configuration	43

7.2.8.1 Detailed Description	43
7.2.8.2 Macro Definition Documentation	45
7.2.9 Normal Drivers	55
7.2.9.1 Detailed Description	55
7.2.9.2 HAL Driver	57
7.2.9.3 ADC Driver	68
7.2.9.4 CAN Driver	89
7.2.9.5 Crypto Driver	110
7.2.9.6 DAC Driver	182
7.2.9.7 EFL Driver	204
7.2.9.8 GPT Driver	217
7.2.9.9 I2C Driver	233
7.2.9.10 I2S Driver	253
7.2.9.11 ICU Driver	264
7.2.9.12 MAC Driver	284
7.2.9.13 PAL Driver	308
7.2.9.14 PWM Driver	358
7.2.9.15 RTC Driver	385
7.2.9.16 SDC Driver	399
7.2.9.17 Serial Driver	435
7.2.9.18 SIO Driver	463
7.2.9.19 SPI Driver Selector	511
7.2.9.20 SPIv1 Driver	511
7.2.9.21 SPIv2 Driver	543
7.2.9.22 ST Driver	580
7.2.9.23 TRNG Driver	592
7.2.9.24 UART Driver	601
7.2.9.25 USB Driver	632
7.2.9.26 WDG Driver	685
7.2.9.27 WSPI Driver	692
7.2.10 Complex Drivers	721
7.2.10.1 Detailed Description	721
7.2.10.2 Buffered SIO Driver	722
7.2.10.3 Managed Flash Storage Driver	737
7.2.10.4 MMC over SPI Driver	764
7.2.10.5 Serial NOR Flash Driver	792
7.2.10.6 Serial over USB Driver	813
7.2.10.7 Improved Serial NOR Driver	831
7.2.11 Interfaces and Classes	916
7.2.11.1 Detailed Description	916
7.2.11.2 Blocks	916
7.2.11.3 Streams	925

7.2.11.4 Base Object	978
7.2.12 Inner Code	980
7.2.12.1 Detailed Description	980
7.2.12.2 I/O Buffers Queues	980
7.2.12.3 MMC/SD Block Device	1005
7.2.12.4 I/O Bytes Queues	1030
7.2.13 Support Code	1053
7.2.13.1 Detailed Description	1053
7.2.13.2 MII/RMII Header	1053
7.2.13.3 USB CDC Header	1068
7.2.14 OSAL	1073
7.2.14.1 Detailed Description	1073
7.2.14.2 Macro Definition Documentation	1079
7.2.14.3 Typedef Documentation	1089
7.2.14.4 Function Documentation	1091
7.2.14.5 Variable Documentation	1104
8 Data Structure Documentation	1105
8.1 BaseAsynchronousChannel Struct Reference	1105
8.1.1 Detailed Description	1108
8.1.2 Field Documentation	1108
8.1.2.1 vmt	1108
8.2 BaseAsynchronousChannelVMT Struct Reference	1108
8.2.1 Detailed Description	1109
8.3 BaseBlockDevice Struct Reference	1109
8.3.1 Detailed Description	1111
8.3.2 Field Documentation	1111
8.3.2.1 vmt	1111
8.4 BaseBlockDeviceVMT Struct Reference	1112
8.4.1 Detailed Description	1112
8.5 BaseBufferedStream Struct Reference	1113
8.5.1 Detailed Description	1114
8.5.2 Field Documentation	1115
8.5.2.1 vmt	1115
8.6 BaseBufferedStreamVMT Struct Reference	1115
8.6.1 Detailed Description	1116
8.7 BaseChannel Struct Reference	1116
8.7.1 Detailed Description	1117
8.7.2 Field Documentation	1118
8.7.2.1 vmt	1118
8.8 BaseChannelVMT Struct Reference	1118
8.8.1 Detailed Description	1119

8.9 BaseFlash Struct Reference	1119
8.9.1 Detailed Description	1121
8.9.2 Field Documentation	1121
8.9.2.1 vmt	1121
8.10 BaseFlashVMT Struct Reference	1122
8.10.1 Detailed Description	1122
8.11 BaseObject Struct Reference	1122
8.11.1 Detailed Description	1123
8.11.2 Field Documentation	1124
8.11.2.1 vmt	1124
8.12 BaseObjectVMT Struct Reference	1124
8.12.1 Detailed Description	1124
8.13 BasePersistentStorage Struct Reference	1124
8.13.1 Detailed Description	1125
8.13.2 Field Documentation	1125
8.13.2.1 vmt	1125
8.14 BasePersistentStorageVMT Struct Reference	1126
8.14.1 Detailed Description	1126
8.15 BaseSequentialStream Struct Reference	1126
8.15.1 Detailed Description	1127
8.15.2 Field Documentation	1127
8.15.2.1 vmt	1127
8.16 BaseSequentialStreamVMT Struct Reference	1128
8.16.1 Detailed Description	1128
8.17 BlockDeviceInfo Struct Reference	1129
8.17.1 Detailed Description	1129
8.17.2 Field Documentation	1129
8.17.2.1 blk_size	1129
8.17.2.2 blk_num	1129
8.18 BufferedSerialVMT Struct Reference	1130
8.18.1 Detailed Description	1131
8.19 BufferedSIODriverVMT Struct Reference	1131
8.19.1 Detailed Description	1132
8.20 BufferedStreamAdapter Struct Reference	1133
8.20.1 Detailed Description	1133
8.20.2 Field Documentation	1133
8.20.2.1 vmt	1133
8.21 BufferedStreamAdapterVMT Struct Reference	1134
8.21.1 Detailed Description	1134
8.22 CANRxFrame Struct Reference	1134
8.22.1 Detailed Description	1135
8.22.2 Field Documentation	1135

8.22.2.1 FMI	1135
8.22.2.2 TIME	1135
8.22.2.3 DLC	1136
8.22.2.4 RTR	1136
8.22.2.5 IDE	1136
8.22.2.6 SID	1136
8.22.2.7 EID	1136
8.22.2.8 _align1	1136
8.22.2.9 [union]	1136
8.22.2.10 data8	1136
8.22.2.11 data16	1137
8.22.2.12 data32	1137
8.22.2.13 [union]	1137
8.23 CANTxFrame Struct Reference	1137
8.23.1 Detailed Description	1138
8.23.2 Field Documentation	1138
8.23.2.1 DLC	1138
8.23.2.2 RTR	1138
8.23.2.3 IDE	1139
8.23.2.4 SID	1139
8.23.2.5 EID	1139
8.23.2.6 _align1	1139
8.23.2.7 [union]	1139
8.23.2.8 data8	1139
8.23.2.9 data16	1139
8.23.2.10 data32	1139
8.23.2.11 [union]	1140
8.24 cdc_linecoding_t Struct Reference	1140
8.24.1 Detailed Description	1140
8.24.2 Field Documentation	1140
8.24.2.1 dwDTERate	1140
8.24.2.2 bCharFormat	1140
8.24.2.3 bParityType	1141
8.24.2.4 bDataBits	1141
8.25 CRYConfig Struct Reference	1141
8.25.1 Detailed Description	1141
8.25.2 Field Documentation	1141
8.25.2.1 dummy	1141
8.26 CRYDriver Struct Reference	1142
8.26.1 Detailed Description	1142
8.26.2 Field Documentation	1142
8.26.2.1 state	1142

8.26.2.2 config	1142
8.27 EFlashDriverVMT Struct Reference	1143
8.27.1 Detailed Description	1143
8.28 event_source Struct Reference	1144
8.28.1 Detailed Description	1144
8.28.2 Field Documentation	1144
8.28.2.1 flags	1144
8.28.2.2 cb	1145
8.28.2.3 param	1145
8.29 FileStream Struct Reference	1145
8.29.1 Detailed Description	1146
8.29.2 Field Documentation	1147
8.29.2.1 vmt	1147
8.30 FileStreamVMT Struct Reference	1147
8.30.1 Detailed Description	1148
8.31 flash_descriptor_t Struct Reference	1148
8.31.1 Detailed Description	1149
8.31.2 Field Documentation	1149
8.31.2.1 attributes	1149
8.31.2.2 page_size	1149
8.31.2.3 sectors_count	1149
8.31.2.4 sectors	1149
8.31.2.5 sectors_size	1149
8.31.2.6 address	1150
8.31.2.7 size	1150
8.32 flash_interface Struct Reference	1150
8.32.1 Detailed Description	1151
8.32.2 Field Documentation	1152
8.32.2.1 vmt	1152
8.33 flash_interface_i Interface Reference	1152
8.33.1 Detailed Description	1153
8.34 flash_interface_vmt Struct Reference	1153
8.34.1 Detailed Description	1154
8.34.2 Field Documentation	1154
8.34.2.1 instance_offset	1154
8.34.2.2 get_descriptor	1154
8.34.2.3 read	1154
8.34.2.4 program	1154
8.34.2.5 start_erase_all	1154
8.34.2.6 start_erase_sector	1154
8.34.2.7 query_erase	1155
8.34.2.8 verify_erase	1155

8.34.2.9 acquire_exclusive	1155
8.34.2.10 release_exclusive	1155
8.35 flash_sector_descriptor_t Struct Reference	1155
8.35.1 Detailed Description	1155
8.35.2 Field Documentation	1156
8.35.2.1 offset	1156
8.35.2.2 size	1156
8.36 GPTConfig Struct Reference	1156
8.36.1 Detailed Description	1157
8.36.2 Field Documentation	1157
8.36.2.1 frequency	1157
8.36.2.2 callback	1157
8.37 GPTDriver Struct Reference	1157
8.37.1 Detailed Description	1158
8.37.2 Field Documentation	1158
8.37.2.1 state	1158
8.37.2.2 config	1158
8.38 hal_adc_config Struct Reference	1158
8.38.1 Detailed Description	1158
8.39 hal_adc_configuration_group Struct Reference	1159
8.39.1 Detailed Description	1159
8.39.2 Field Documentation	1160
8.39.2.1 circular	1160
8.39.2.2 num_channels	1160
8.39.2.3 end_cb	1160
8.39.2.4 error_cb	1160
8.40 hal_adc_driver Struct Reference	1160
8.40.1 Detailed Description	1162
8.40.2 Field Documentation	1162
8.40.2.1 state	1162
8.40.2.2 config	1162
8.40.2.3 samples	1162
8.40.2.4 depth	1162
8.40.2.5 grpp	1162
8.40.2.6 thread	1162
8.40.2.7 mutex	1163
8.41 hal_buffered_serial Struct Reference	1163
8.41.1 Detailed Description	1165
8.41.2 Field Documentation	1165
8.41.2.1 vmt	1165
8.42 hal_buffered_siol_driver Struct Reference	1165
8.42.1 Detailed Description	1167

8.42.2 Field Documentation	1167
8.42.2.1 vmt	1167
8.43 hal_can_config Struct Reference	1167
8.43.1 Detailed Description	1168
8.43.2 Field Documentation	1168
8.43.2.1 dummy	1168
8.44 hal_can_driver Struct Reference	1168
8.44.1 Detailed Description	1169
8.44.2 Field Documentation	1169
8.44.2.1 state	1169
8.44.2.2 config	1169
8.44.2.3 txqueue	1169
8.44.2.4 rxqueue	1169
8.44.2.5 rxfull_event	1170
8.44.2.6 txempty_event	1170
8.44.2.7 error_event	1170
8.44.2.8 sleep_event	1170
8.44.2.9 wakeup_event	1170
8.45 hal_dac_config Struct Reference	1171
8.45.1 Detailed Description	1171
8.46 hal_dac_conversion_group Struct Reference	1171
8.46.1 Detailed Description	1172
8.46.2 Field Documentation	1172
8.46.2.1 num_channels	1172
8.46.2.2 end_cb	1172
8.46.2.3 error_cb	1172
8.47 hal_dac_driver Struct Reference	1173
8.47.1 Detailed Description	1174
8.47.2 Field Documentation	1174
8.47.2.1 state	1174
8.47.2.2 grpp	1174
8.47.2.3 samples	1174
8.47.2.4 depth	1174
8.47.2.5 config	1174
8.47.2.6 thread	1174
8.47.2.7 mutex	1175
8.48 hal_efl_config Struct Reference	1175
8.48.1 Detailed Description	1175
8.49 hal_efl_driver Struct Reference	1175
8.49.1 Detailed Description	1177
8.49.2 Field Documentation	1177
8.49.2.1 vmt	1177

8.49.2.2 mutex	1177
8.50 hal_i2c_config Struct Reference	1177
8.50.1 Detailed Description	1178
8.50.2 Field Documentation	1178
8.50.2.1 dummy	1178
8.51 hal_i2c_driver Struct Reference	1178
8.51.1 Detailed Description	1179
8.51.2 Field Documentation	1179
8.51.2.1 state	1179
8.51.2.2 config	1179
8.51.2.3 errors	1179
8.51.2.4 mutex	1179
8.52 hal_i2s_config Struct Reference	1180
8.52.1 Detailed Description	1180
8.52.2 Field Documentation	1180
8.52.2.1 tx_buffer	1180
8.52.2.2 rx_buffer	1181
8.52.2.3 size	1181
8.52.2.4 end_cb	1181
8.53 hal_i2s_driver Struct Reference	1181
8.53.1 Detailed Description	1182
8.53.2 Field Documentation	1182
8.53.2.1 state	1182
8.53.2.2 config	1182
8.54 hal_mac_config Struct Reference	1182
8.54.1 Detailed Description	1183
8.55 hal_mac_driver Struct Reference	1183
8.55.1 Detailed Description	1184
8.55.2 Field Documentation	1184
8.55.2.1 state	1184
8.55.2.2 config	1184
8.55.2.3 tdqueue	1184
8.55.2.4 rdqueue	1184
8.55.2.5 es	1184
8.55.2.6 flags	1184
8.55.2.7 cb	1185
8.55.2.8 arg	1185
8.56 hal_mac_receive_descriptor Struct Reference	1185
8.56.1 Detailed Description	1186
8.56.2 Field Documentation	1186
8.56.2.1 offset	1186
8.56.2.2 size	1186

8.57 hal_mac_transmit_descriptor Struct Reference	1186
8.57.1 Detailed Description	1187
8.57.2 Field Documentation	1187
8.57.2.1 offset	1187
8.57.2.2 size	1187
8.58 hal_serial_config Struct Reference	1187
8.58.1 Detailed Description	1188
8.58.2 Field Documentation	1188
8.58.2.1 speed	1188
8.59 hal_serial_driver Struct Reference	1188
8.59.1 Detailed Description	1190
8.59.2 Field Documentation	1190
8.59.2.1 vmt	1190
8.60 hal_sio_config Struct Reference	1190
8.60.1 Detailed Description	1190
8.61 hal_sio_driver Struct Reference	1191
8.61.1 Detailed Description	1192
8.61.2 Field Documentation	1192
8.61.2.1 vmt	1192
8.61.2.2 state	1192
8.61.2.3 config	1192
8.61.2.4 enabled	1192
8.61.2.5 cb	1193
8.61.2.6 arg	1193
8.61.2.7 sync_rx	1193
8.61.2.8 sync_rxidle	1193
8.61.2.9 sync_tx	1193
8.61.2.10 sync_txend	1193
8.62 hal_spi_config Struct Reference	1194
8.62.1 Detailed Description	1195
8.62.2 Field Documentation	1195
8.62.2.1 circular	1195
8.62.2.2 end_cb	1195
8.62.2.3 ssline	1195
8.62.2.4 ssport	1195
8.62.2.5 ssmask	1195
8.62.2.6 sspad	1196
8.62.2.7 slave	1196
8.62.2.8 data_cb	1196
8.62.2.9 error_cb	1196
8.63 hal_spi_driver Struct Reference	1196
8.63.1 Detailed Description	1197

8.63.2 Field Documentation	1198
8.63.2.1 state	1198
8.63.2.2 config	1198
8.63.2.3 thread	1198
8.63.2.4 mutex	1198
8.63.2.5 sync_transfer	1198
8.64 hal_trng_config Struct Reference	1198
8.64.1 Detailed Description	1199
8.65 hal_trng_driver Struct Reference	1199
8.65.1 Detailed Description	1199
8.65.2 Field Documentation	1199
8.65.2.1 state	1199
8.65.2.2 config	1200
8.66 hal_uart_config Struct Reference	1200
8.66.1 Detailed Description	1201
8.66.2 Field Documentation	1201
8.66.2.1 txend1_cb	1201
8.66.2.2 txend2_cb	1201
8.66.2.3 rxend_cb	1201
8.66.2.4 rxchar_cb	1201
8.66.2.5 rxerr_cb	1201
8.67 hal_uart_driver Struct Reference	1202
8.67.1 Detailed Description	1203
8.67.2 Field Documentation	1203
8.67.2.1 state	1203
8.67.2.2 txstate	1203
8.67.2.3 rxstate	1203
8.67.2.4 config	1203
8.67.2.5 early	1203
8.67.2.6 threadrx	1203
8.67.2.7 threadtx	1204
8.67.2.8 mutex	1204
8.68 hal_wspi_config Struct Reference	1204
8.68.1 Detailed Description	1205
8.68.2 Field Documentation	1205
8.68.2.1 end_cb	1205
8.68.2.2 error_cb	1205
8.69 hal_wspi_driver Struct Reference	1205
8.69.1 Detailed Description	1206
8.69.2 Field Documentation	1206
8.69.2.1 state	1206
8.69.2.2 config	1206

8.69.2.3 thread	1206
8.69.2.4 mutex	1206
8.70 hal_xsnor_base Struct Reference	1207
8.70.1 Detailed Description	1208
8.70.2 Field Documentation	1208
8.70.2.1 vmt	1208
8.70.2.2 fls	1208
8.70.2.3 state	1208
8.70.2.4 config	1208
8.70.2.5 commands	1209
8.70.2.6 mutex	1209
8.70.2.7 descriptor	1209
8.71 hal_xsnor_base_c Class Reference	1209
8.71.1 Detailed Description	1210
8.72 hal_xsnor_base_vmt Struct Reference	1210
8.72.1 Detailed Description	1211
8.72.2 Field Documentation	1211
8.72.2.1 dispose	1211
8.72.2.2 init	1211
8.72.2.3 read	1211
8.72.2.4 program	1211
8.72.2.5 start_erase_all	1211
8.72.2.6 start_erase_sector	1211
8.72.2.7 query_erase	1212
8.72.2.8 verify_erase	1212
8.72.2.9 mmap_on	1212
8.72.2.10 mmap_off	1212
8.73 hal_xsnor_macronix_mx25 Struct Reference	1212
8.73.1 Detailed Description	1214
8.73.2 Field Documentation	1214
8.73.2.1 vmt	1214
8.73.2.2 fls	1214
8.73.2.3 state	1214
8.73.2.4 config	1214
8.73.2.5 commands	1214
8.73.2.6 mutex	1215
8.73.2.7 descriptor	1215
8.74 hal_xsnor_macronix_mx25_c Class Reference	1215
8.75 hal_xsnor_macronix_mx25_vmt Struct Reference	1216
8.75.1 Detailed Description	1217
8.75.2 Field Documentation	1217
8.75.2.1 dispose	1217

8.75.2.2 init	1217
8.75.2.3 read	1217
8.75.2.4 program	1217
8.75.2.5 start_erase_all	1217
8.75.2.6 start_erase_sector	1217
8.75.2.7 query_erase	1218
8.75.2.8 verify_erase	1218
8.75.2.9 mmap_on	1218
8.75.2.10 mmap_off	1218
8.76 hal_xsnor_micron_n25q Struct Reference	1218
8.76.1 Detailed Description	1220
8.76.2 Field Documentation	1220
8.76.2.1 vmt	1220
8.76.2.2 fls	1220
8.76.2.3 state	1220
8.76.2.4 config	1220
8.76.2.5 commands	1220
8.76.2.6 mutex	1221
8.76.2.7 descriptor	1221
8.77 hal_xsnor_micron_n25q_c Class Reference	1221
8.78 hal_xsnor_micron_n25q_vmt Struct Reference	1222
8.78.1 Detailed Description	1223
8.78.2 Field Documentation	1223
8.78.2.1 dispose	1223
8.78.2.2 init	1223
8.78.2.3 read	1223
8.78.2.4 program	1223
8.78.2.5 start_erase_all	1223
8.78.2.6 start_erase_sector	1223
8.78.2.7 query_erase	1224
8.78.2.8 verify_erase	1224
8.78.2.9 mmap_on	1224
8.78.2.10 mmap_off	1224
8.79 halclkcfg_t Struct Reference	1224
8.79.1 Detailed Description	1224
8.79.2 Field Documentation	1225
8.79.2.1 dummy	1225
8.80 HMACSHA256Context Struct Reference	1225
8.80.1 Detailed Description	1225
8.80.2 Field Documentation	1225
8.80.2.1 dummy	1225
8.81 HMACSHA512Context Struct Reference	1226

8.81.1 Detailed Description	1226
8.81.2 Field Documentation	1226
8.81.2.1 dummy	1226
8.82 ICUConfig Struct Reference	1226
8.82.1 Detailed Description	1227
8.82.2 Field Documentation	1227
8.82.2.1 mode	1227
8.82.2.2 frequency	1228
8.82.2.3 width_cb	1228
8.82.2.4 period_cb	1228
8.82.2.5 overflow_cb	1228
8.83 ICUDriver Struct Reference	1228
8.83.1 Detailed Description	1229
8.83.2 Field Documentation	1229
8.83.2.1 state	1229
8.83.2.2 config	1229
8.84 io_buffers_queue Struct Reference	1229
8.84.1 Detailed Description	1231
8.84.2 Field Documentation	1231
8.84.2.1 waiting	1231
8.84.2.2 suspended	1231
8.84.2.3 bcounter	1231
8.84.2.4 bwrptr	1231
8.84.2.5 brdptr	1231
8.84.2.6 btop	1232
8.84.2.7 bsize	1232
8.84.2.8 bn	1232
8.84.2.9 buffers	1232
8.84.2.10 ptr	1232
8.84.2.11 top	1232
8.84.2.12 notify	1233
8.84.2.13 link	1233
8.85 io_queue Struct Reference	1233
8.85.1 Detailed Description	1234
8.85.2 Field Documentation	1234
8.85.2.1 q_waiting	1234
8.85.2.2 q_counter	1234
8.85.2.3 q_buffer	1234
8.85.2.4 q_top	1235
8.85.2.5 q_wrptr	1235
8.85.2.6 q_rdprr	1235
8.85.2.7 q_notify	1235

8.85.2.8 q_link	1235
8.86 IOBus Struct Reference	1235
8.86.1 Detailed Description	1236
8.86.2 Field Documentation	1236
8.86.2.1 portid	1236
8.86.2.2 mask	1236
8.86.2.3 offset	1236
8.87 MemoryStream Struct Reference	1237
8.87.1 Detailed Description	1238
8.87.2 Field Documentation	1239
8.87.2.1 vmt	1239
8.88 MemStreamVMT Struct Reference	1239
8.88.1 Detailed Description	1239
8.89 mfs_bank_header_t Union Reference	1239
8.89.1 Detailed Description	1240
8.89.2 Field Documentation	1240
8.89.2.1 magic1	1240
8.89.2.2 magic2	1241
8.89.2.3 counter	1241
8.89.2.4 reserved1	1241
8.89.2.5 crc	1241
8.89.2.6 [struct]	1241
8.89.2.7 hdr8	1241
8.89.2.8 hdr32	1241
8.90 mfs_data_header_t Union Reference	1242
8.90.1 Detailed Description	1242
8.90.2 Field Documentation	1243
8.90.2.1 magic1	1243
8.90.2.2 magic2	1243
8.90.2.3 id	1243
8.90.2.4 crc	1243
8.90.2.5 size	1243
8.90.2.6 [struct]	1243
8.90.2.7 hdr8	1243
8.90.2.8 hdr32	1244
8.91 mfs_nocache_buffer Union Reference	1244
8.91.1 Detailed Description	1244
8.91.2 Field Documentation	1245
8.91.2.1 dhdr	1245
8.91.2.2 bhdr	1245
8.91.2.3 data8	1245
8.91.2.4 data16	1245

8.91.2.5 data32	1245
8.92 mfs_record_descriptor_t Struct Reference	1245
8.92.1 Field Documentation	1246
8.92.1.1 offset	1246
8.92.1.2 size	1246
8.93 mfs_transaction_op_t Struct Reference	1246
8.93.1 Detailed Description	1246
8.93.2 Field Documentation	1247
8.93.2.1 offset	1247
8.93.2.2 size	1247
8.93.2.3 id	1247
8.94 MFSCfg Struct Reference	1247
8.94.1 Detailed Description	1248
8.94.2 Field Documentation	1249
8.94.2.1 flashp	1249
8.94.2.2 erased	1249
8.94.2.3 bank_size	1249
8.94.2.4 bank0_start	1249
8.94.2.5 bank0_sectors	1249
8.94.2.6 bank1_start	1249
8.94.2.7 bank1_sectors	1250
8.95 MFSDriver Struct Reference	1250
8.95.1 Detailed Description	1251
8.95.2 Field Documentation	1251
8.95.2.1 state	1251
8.95.2.2 config	1251
8.95.2.3 current_bank	1252
8.95.2.4 current_counter	1252
8.95.2.5 next_offset	1252
8.95.2.6 used_space	1252
8.95.2.7 descriptors	1252
8.95.2.8 tr_next_offset	1252
8.95.2.9 tr_limit_offset	1252
8.95.2.10 tr_nops	1253
8.95.2.11 tr_ops	1253
8.95.2.12 nbuf	1253
8.96 mmc_spi_config_t Struct Reference	1253
8.96.1 Detailed Description	1254
8.96.2 Field Documentation	1255
8.96.2.1 spip	1255
8.96.2.2 lscfg	1255
8.96.2.3 hscfg	1255

8.97 mmc_spi_driver_t Struct Reference	1255
8.97.1 Detailed Description	1257
8.97.2 Field Documentation	1257
8.97.2.1 vmt	1257
8.97.2.2 config	1257
8.97.2.3 block_addresses	1257
8.97.2.4 buffer	1257
8.98 mmc_spi_driver_vmt Struct Reference	1258
8.98.1 Detailed Description	1259
8.99 MMCSDBlockDevice Struct Reference	1259
8.99.1 Detailed Description	1261
8.99.2 Field Documentation	1262
8.99.2.1 vmt	1262
8.100 MMCSDBlockDeviceVMT Struct Reference	1262
8.100.1 Detailed Description	1263
8.101 NullStream Struct Reference	1263
8.101.1 Detailed Description	1265
8.101.2 Field Documentation	1265
8.101.2.1 vmt	1265
8.102 NullStreamVMT Struct Reference	1265
8.102.1 Detailed Description	1265
8.103 PALConfig Struct Reference	1265
8.103.1 Detailed Description	1266
8.104 palevent_t Struct Reference	1266
8.104.1 Detailed Description	1266
8.104.2 Field Documentation	1267
8.104.2.1 threads	1267
8.104.2.2 cb	1267
8.104.2.3 arg	1267
8.105 PWMChannelConfig Struct Reference	1267
8.105.1 Detailed Description	1268
8.105.2 Field Documentation	1268
8.105.2.1 mode	1268
8.105.2.2 callback	1268
8.106 PWMConfig Struct Reference	1268
8.106.1 Detailed Description	1269
8.106.2 Field Documentation	1270
8.106.2.1 frequency	1270
8.106.2.2 period	1270
8.106.2.3 callback	1270
8.106.2.4 channels	1270
8.107 PWMDriver Struct Reference	1271

8.107.1 Detailed Description	1272
8.107.2 Field Documentation	1272
8.107.2.1 state	1272
8.107.2.2 config	1272
8.107.2.3 period	1272
8.107.2.4 enabled	1272
8.107.2.5 channels	1272
8.108 RTCArm Struct Reference	1272
8.108.1 Detailed Description	1273
8.108.2 Field Documentation	1273
8.108.2.1 dummy	1273
8.109 RTCDatetime Struct Reference	1273
8.109.1 Detailed Description	1274
8.109.2 Field Documentation	1274
8.109.2.1 year	1274
8.109.2.2 month	1274
8.109.2.3 dstflag	1274
8.109.2.4 dayofweek	1274
8.109.2.5 day	1274
8.109.2.6 millisecond	1274
8.110 RTCDriver Struct Reference	1275
8.110.1 Detailed Description	1275
8.110.2 Field Documentation	1275
8.110.2.1 vmt	1275
8.111 RTCDriverVMT Struct Reference	1276
8.111.1 Detailed Description	1277
8.112 SDCCfg Struct Reference	1278
8.112.1 Detailed Description	1278
8.112.2 Field Documentation	1278
8.112.2.1 bus_width	1278
8.113 SDCDriver Struct Reference	1279
8.113.1 Detailed Description	1280
8.113.2 Field Documentation	1280
8.113.2.1 vmt	1280
8.113.2.2 config	1280
8.113.2.3 cardmode	1280
8.113.2.4 errors	1280
8.113.2.5 rca	1280
8.113.2.6 buf	1280
8.114 SDCDriverVMT Struct Reference	1281
8.114.1 Detailed Description	1282
8.115 SerialDriverVMT Struct Reference	1282

8.115.1 Detailed Description	1283
8.116 SerialUSBConfig Struct Reference	1284
8.116.1 Detailed Description	1284
8.116.2 Field Documentation	1285
8.116.2.1 usbp	1285
8.116.2.2 bulk_in	1285
8.116.2.3 bulk_out	1285
8.116.2.4 int_in	1285
8.117 SerialUSBDriver Struct Reference	1285
8.117.1 Detailed Description	1287
8.117.2 Field Documentation	1287
8.117.2.1 vmt	1287
8.118 SerialUSBDriverVMT Struct Reference	1288
8.118.1 Detailed Description	1289
8.119 SHA1Context Struct Reference	1289
8.119.1 Detailed Description	1290
8.119.2 Field Documentation	1290
8.119.2.1 dummy	1290
8.120 SHA256Context Struct Reference	1290
8.120.1 Detailed Description	1290
8.120.2 Field Documentation	1290
8.120.2.1 dummy	1290
8.121 SHA512Context Struct Reference	1291
8.121.1 Detailed Description	1291
8.121.2 Field Documentation	1291
8.121.2.1 dummy	1291
8.122 sio_driver_vmt Struct Reference	1291
8.122.1 Detailed Description	1292
8.123 snor_nocache_buffer Struct Reference	1293
8.123.1 Field Documentation	1293
8.123.1.1 buf	1293
8.123.1.2 cmd	1293
8.124 SNORConfig Struct Reference	1294
8.124.1 Detailed Description	1294
8.124.2 Field Documentation	1294
8.124.2.1 busp	1294
8.124.2.2 buscfg	1294
8.125 SNORDriver Struct Reference	1295
8.125.1 Detailed Description	1297
8.125.2 Field Documentation	1297
8.125.2.1 vmt	1297
8.125.2.2 config	1297

8.125.2.3 nocache	1297
8.125.2.4 mutex	1297
8.126 SNORDriverVMT Struct Reference	1297
8.126.1 Detailed Description	1298
8.127 threads_queue_t Struct Reference	1298
8.127.1 Detailed Description	1299
8.127.2 Field Documentation	1299
8.127.2.1 tr	1299
8.128 unpacked_mmc_cid_t Struct Reference	1299
8.128.1 Detailed Description	1300
8.128.2 Field Documentation	1300
8.128.2.1 mid	1300
8.128.2.2 oid	1300
8.128.2.3 pnm	1301
8.128.2.4 prv_n	1301
8.128.2.5 prv_m	1301
8.128.2.6 psn	1301
8.128.2.7 mdt_m	1301
8.128.2.8 mdt_y	1301
8.128.2.9 crc	1301
8.129 unpacked_mmc_csd_t Struct Reference	1301
8.129.1 Detailed Description	1303
8.129.2 Field Documentation	1303
8.129.2.1 csd_structure	1303
8.129.2.2 spec_vers	1303
8.129.2.3 taac	1303
8.129.2.4 nsac	1303
8.129.2.5 tran_speed	1303
8.129.2.6 ccc	1303
8.129.2.7 read_bl_len	1303
8.129.2.8 read_bl_partial	1304
8.129.2.9 write_blk_misalign	1304
8.129.2.10 read_blk_misalign	1304
8.129.2.11 dsr_imp	1304
8.129.2.12 c_size	1304
8.129.2.13 vdd_r_curr_min	1304
8.129.2.14 vdd_r_curr_max	1304
8.129.2.15 vdd_w_curr_min	1304
8.129.2.16 vdd_w_curr_max	1304
8.129.2.17 c_size_mult	1304
8.129.2.18 erase_grp_size	1305
8.129.2.19 erase_grp_mult	1305

8.129.2.20 wp_grp_size	1305
8.129.2.21 wp_grp_enable	1305
8.129.2.22 default_ecc	1305
8.129.2.23 r2w_factor	1305
8.129.2.24 write_bl_len	1305
8.129.2.25 write_bl_partial	1305
8.129.2.26 content_prot_app	1305
8.129.2.27 file_format_grp	1305
8.129.2.28 copy	1306
8.129.2.29 perm_write_protect	1306
8.129.2.30 tmp_write_protect	1306
8.129.2.31 file_format	1306
8.129.2.32 ecc	1306
8.129.2.33 crc	1306
8.130 unpacked_sdc_cid_t Struct Reference	1306
8.130.1 Detailed Description	1307
8.130.2 Field Documentation	1307
8.130.2.1 mid	1307
8.130.2.2 oid	1307
8.130.2.3 pnm	1307
8.130.2.4 prv_n	1307
8.130.2.5 prv_m	1307
8.130.2.6 psn	1307
8.130.2.7 mdt_m	1308
8.130.2.8 mdt_y	1308
8.130.2.9 crc	1308
8.131 unpacked_sdc_csd_10_t Struct Reference	1308
8.131.1 Detailed Description	1309
8.131.2 Field Documentation	1309
8.131.2.1 csd_structure	1309
8.131.2.2 taac	1309
8.131.2.3 nsac	1309
8.131.2.4 tran_speed	1310
8.131.2.5 ccc	1310
8.131.2.6 read_bl_len	1310
8.131.2.7 read_bl_partial	1310
8.131.2.8 write_blk_misalign	1310
8.131.2.9 read_blk_misalign	1310
8.131.2.10 dsr_imp	1310
8.131.2.11 c_size	1310
8.131.2.12 vdd_r_curr_min	1310
8.131.2.13 vdd_r_curr_max	1310

8.131.2.14 vdd_w_curr_min	1311
8.131.2.15 vdd_w_curr_max	1311
8.131.2.16 c_size_mult	1311
8.131.2.17 erase_blk_en	1311
8.131.2.18 erase_sector_size	1311
8.131.2.19 wp_grp_size	1311
8.131.2.20 wp_grp_enable	1311
8.131.2.21 r2w_factor	1311
8.131.2.22 write_bl_len	1311
8.131.2.23 write_bl_partial	1311
8.131.2.24 file_format_grp	1312
8.131.2.25 copy	1312
8.131.2.26 perm_write_protect	1312
8.131.2.27 tmp_write_protect	1312
8.131.2.28 file_format	1312
8.131.2.29 crc	1312
8.132 unpacked_sdc_csd_20_t Struct Reference	1312
8.132.1 Detailed Description	1313
8.132.2 Field Documentation	1313
8.132.2.1 csd_structure	1313
8.132.2.2 taac	1313
8.132.2.3 nsac	1313
8.132.2.4 tran_speed	1313
8.132.2.5 ccc	1314
8.132.2.6 read_bl_len	1314
8.132.2.7 read_bl_partial	1314
8.132.2.8 write_blk_misalign	1314
8.132.2.9 read_blk_misalign	1314
8.132.2.10 dsr_imp	1314
8.132.2.11 c_size	1314
8.132.2.12 erase_blk_en	1314
8.132.2.13 erase_sector_size	1314
8.132.2.14 wp_grp_size	1314
8.132.2.15 wp_grp_enable	1315
8.132.2.16 r2w_factor	1315
8.132.2.17 write_bl_len	1315
8.132.2.18 write_bl_partial	1315
8.132.2.19 file_format_grp	1315
8.132.2.20 copy	1315
8.132.2.21 perm_write_protect	1315
8.132.2.22 tmp_write_protect	1315
8.132.2.23 file_format	1315

8.132.2.24 crc	1315
8.133 USBConfig Struct Reference	1316
8.133.1 Detailed Description	1317
8.133.2 Field Documentation	1317
8.133.2.1 event_cb	1317
8.133.2.2 get_descriptor_cb	1317
8.133.2.3 requests_hook_cb	1317
8.133.2.4 sof_cb	1317
8.134 USBDescriptor Struct Reference	1318
8.134.1 Detailed Description	1318
8.134.2 Field Documentation	1318
8.134.2.1 ud_size	1318
8.134.2.2 ud_string	1318
8.135 USBDriver Struct Reference	1319
8.135.1 Detailed Description	1320
8.135.2 Field Documentation	1320
8.135.2.1 state	1320
8.135.2.2 config	1320
8.135.2.3 transmitting	1320
8.135.2.4 receiving	1320
8.135.2.5 epc	1320
8.135.2.6 in_params	1321
8.135.2.7 out_params	1321
8.135.2.8 ep0state	1321
8.135.2.9 ep0next	1321
8.135.2.10 ep0n	1321
8.135.2.11 ep0endcb	1321
8.135.2.12 setup	1322
8.135.2.13 status	1322
8.135.2.14 address	1322
8.135.2.15 configuration	1322
8.135.2.16 saved_state	1322
8.136 USBEndpointConfig Struct Reference	1322
8.136.1 Detailed Description	1324
8.136.2 Field Documentation	1324
8.136.2.1 ep_mode	1324
8.136.2.2 setup_cb	1324
8.136.2.3 in_cb	1325
8.136.2.4 out_cb	1325
8.136.2.5 in_maxsize	1325
8.136.2.6 out_maxsize	1325
8.136.2.7 in_state	1325

8.136.2.8 out_state	1325
8.137 USBInEndpointState Struct Reference	1326
8.137.1 Detailed Description	1326
8.137.2 Field Documentation	1326
8.137.2.1 txsize	1326
8.137.2.2 txcnt	1326
8.137.2.3 txbuf	1327
8.137.2.4 thread	1327
8.138 USBOutEndpointState Struct Reference	1327
8.138.1 Detailed Description	1327
8.138.2 Field Documentation	1328
8.138.2.1 rxsize	1328
8.138.2.2 rxcnt	1328
8.138.2.3 rxbuf	1328
8.138.2.4 thread	1328
8.139 WDGConfig Struct Reference	1328
8.139.1 Detailed Description	1329
8.140 WDGDriver Struct Reference	1329
8.140.1 Detailed Description	1329
8.140.2 Field Documentation	1330
8.140.2.1 state	1330
8.140.2.2 config	1330
8.141 wspi_command_t Struct Reference	1330
8.141.1 Detailed Description	1331
8.141.2 Field Documentation	1331
8.141.2.1 cfg	1331
8.141.2.2 cmd	1331
8.141.2.3 addr	1331
8.141.2.4 alt	1331
8.141.2.5 dummy	1331
8.142 xsnor_buffers Struct Reference	1332
8.142.1 Detailed Description	1332
8.142.2 Field Documentation	1332
8.142.2.1 databuf	1332
8.142.2.2 cmdbuf	1333
8.142.2.3 spibuf	1333
8.143 xsnor_bus_configs Union Reference	1333
8.143.1 Detailed Description	1334
8.143.2 Field Documentation	1335
8.143.2.1 wspi	1335
8.143.2.2 spi	1335
8.144 xsnor_bus_spi Struct Reference	1335

8.144.1 Detailed Description	1336
8.144.2 Field Documentation	1336
8.144.2.1 drv	1336
8.144.2.2 cfg	1336
8.145 xsnor_bus_wspi Struct Reference	1336
8.145.1 Detailed Description	1337
8.145.2 Field Documentation	1337
8.145.2.1 drv	1337
8.145.2.2 cfg	1338
8.146 xsnor_commands Struct Reference	1338
8.146.1 Detailed Description	1338
8.146.2 Field Documentation	1338
8.146.2.1 cmd	1338
8.146.2.2 cmd_addr	1339
8.146.2.3 cmd_data	1339
8.146.2.4 cmd_addr_data	1339
8.147 xsnor_config Struct Reference	1339
8.147.1 Detailed Description	1340
8.147.2 Field Documentation	1340
8.147.2.1 bus_type	1340
8.147.2.2 bus	1340
8.147.2.3 buffers	1340
8.147.2.4 options	1340
9 File Documentation	1341
9.1 main.dox File Reference	1341
9.2 main.dox File Reference	1341
9.3 ccportab.h File Reference	1341
9.3.1 Detailed Description	1342
9.4 hal.dox File Reference	1344
9.5 hal_adc.dox File Reference	1344
9.6 hal_buffererd_sio.dox File Reference	1344
9.7 hal_buffers.dox File Reference	1344
9.8 hal_can.dox File Reference	1344
9.9 hal_crypto.dox File Reference	1344
9.10 hal_dac.dox File Reference	1344
9.11 hal_efl.dox File Reference	1344
9.12 hal_flash.dox File Reference	1344
9.13 hal_gpt.dox File Reference	1344
9.14 hal_i2c.dox File Reference	1344
9.15 hal_i2s.dox File Reference	1344
9.16 hal_icu.dox File Reference	1344

9.17 hal_interfaces.dox File Reference	1344
9.18 hal_ioblock.dox File Reference	1344
9.19 hal_mac.dox File Reference	1344
9.20 hal_mfs.dox File Reference	1344
9.21 hal_mii.dox File Reference	1344
9.22 hal_mmc_spi.dox File Reference	1344
9.23 hal_mmcsd.dox File Reference	1344
9.24 hal_norflash.dox File Reference	1344
9.25 hal_objects.dox File Reference	1344
9.26 hal_pal.dox File Reference	1344
9.27 hal_peripherals.dox File Reference	1344
9.28 hal_pwm.dox File Reference	1344
9.29 hal_queues.dox File Reference	1344
9.30 hal_rtc.dox File Reference	1344
9.31 hal_sdc.dox File Reference	1344
9.32 hal_serial.dox File Reference	1344
9.33 hal_serial_usb.dox File Reference	1344
9.34 hal_sio.dox File Reference	1344
9.35 hal_spi.dox File Reference	1344
9.36 hal_spi_v1.dox File Reference	1344
9.37 hal_spi_v2.dox File Reference	1344
9.38 hal_st.dox File Reference	1344
9.39 hal_trng.dox File Reference	1344
9.40 hal_uart.dox File Reference	1344
9.41 hal_usb.dox File Reference	1344
9.42 hal_usb_cdc.dox File Reference	1344
9.43 hal_wdg.dox File Reference	1344
9.44 hal_wspi.dox File Reference	1344
9.45 hal.h File Reference	1344
9.45.1 Detailed Description	1347
9.46 hal_adc.h File Reference	1347
9.46.1 Detailed Description	1348
9.47 hal_buffered_serial.h File Reference	1349
9.47.1 Detailed Description	1350
9.48 hal_buffers.h File Reference	1350
9.48.1 Detailed Description	1352
9.49 hal_can.h File Reference	1352
9.49.1 Detailed Description	1353
9.50 hal_channels.h File Reference	1354
9.50.1 Detailed Description	1355
9.51 hal_crypto.h File Reference	1355
9.51.1 Detailed Description	1358

9.52 hal_dac.h File Reference	1359
9.52.1 Detailed Description	1360
9.53 hal_efl.h File Reference	1360
9.53.1 Detailed Description	1361
9.54 hal_files.h File Reference	1361
9.54.1 Detailed Description	1362
9.55 hal_flash.h File Reference	1362
9.55.1 Detailed Description	1364
9.56 hal_gpt.h File Reference	1364
9.56.1 Detailed Description	1366
9.57 hal_i2c.h File Reference	1366
9.57.1 Detailed Description	1367
9.58 hal_i2s.h File Reference	1367
9.58.1 Detailed Description	1369
9.59 hal_icu.h File Reference	1369
9.59.1 Detailed Description	1370
9.60 hal_jobblock.h File Reference	1370
9.60.1 Detailed Description	1371
9.61 hal_mac.h File Reference	1371
9.61.1 Detailed Description	1373
9.62 hal_mii.h File Reference	1373
9.62.1 Detailed Description	1376
9.63 hal_mmc_spi.h File Reference	1376
9.63.1 Detailed Description	1377
9.64 hal_mmcsd.h File Reference	1377
9.64.1 Detailed Description	1381
9.65 hal_objects.h File Reference	1382
9.65.1 Detailed Description	1382
9.66 hal_pal.h File Reference	1382
9.66.1 Detailed Description	1386
9.67 hal_persistent.h File Reference	1386
9.67.1 Detailed Description	1387
9.68 hal_pwm.h File Reference	1387
9.68.1 Detailed Description	1388
9.69 hal_queues.h File Reference	1388
9.69.1 Detailed Description	1390
9.70 hal_rtc.h File Reference	1390
9.70.1 Detailed Description	1392
9.71 hal_sdc.h File Reference	1392
9.71.1 Detailed Description	1393
9.72 hal_serial.h File Reference	1393
9.72.1 Detailed Description	1395

9.73 hal_serial_usb.h File Reference	1395
9.73.1 Detailed Description	1397
9.74 hal_sio.h File Reference	1397
9.74.1 Detailed Description	1400
9.75 hal_spi.h File Reference	1400
9.75.1 Detailed Description	1400
9.76 hal_spi_v1.h File Reference	1400
9.76.1 Detailed Description	1402
9.77 hal_spi_v2.h File Reference	1403
9.77.1 Detailed Description	1405
9.78 hal_st.h File Reference	1405
9.78.1 Detailed Description	1406
9.79 hal_streams.h File Reference	1406
9.79.1 Detailed Description	1407
9.80 hal_trng.h File Reference	1407
9.80.1 Detailed Description	1408
9.81 hal_uart.h File Reference	1408
9.81.1 Detailed Description	1410
9.82 hal_usb.h File Reference	1410
9.82.1 Detailed Description	1414
9.83 hal_usb_cdc.h File Reference	1414
9.83.1 Detailed Description	1415
9.84 hal_wdg.h File Reference	1416
9.84.1 Detailed Description	1416
9.85 hal_wspi.h File Reference	1416
9.85.1 Detailed Description	1419
9.86 hal_buffered_sio.c File Reference	1419
9.86.1 Detailed Description	1420
9.87 hal_buffered_sio.h File Reference	1420
9.87.1 Detailed Description	1421
9.88 hal_mfs.c File Reference	1421
9.88.1 Detailed Description	1423
9.89 hal_mfs.h File Reference	1423
9.89.1 Detailed Description	1425
9.90 hal_serial_nor.c File Reference	1425
9.90.1 Detailed Description	1426
9.91 hal_serial_nor.h File Reference	1426
9.91.1 Detailed Description	1428
9.92 hal_xsnor_macronix_mx25.c File Reference	1428
9.92.1 Detailed Description	1429
9.93 hal_xsnor_macronix_mx25.h File Reference	1429
9.93.1 Detailed Description	1430

9.94 hal_xsnor_macronix_mx25_impl.inc File Reference	1430
9.94.1 Detailed Description	1433
9.95 hal_xsnor_micron_n25q.c File Reference	1434
9.95.1 Detailed Description	1434
9.96 hal_xsnor_micron_n25q.h File Reference	1434
9.96.1 Detailed Description	1435
9.97 hal_xsnor_micron_n25q_impl.inc File Reference	1435
9.97.1 Detailed Description	1437
9.98 xsnor.dox File Reference	1438
9.99 xsnor_arch.dox File Reference	1438
9.100 xsnor_macronix_mx25.dox File Reference	1438
9.101 xsnor_micron_n25q.dox File Reference	1438
9.102 hal_flash_interface.h File Reference	1438
9.102.1 Detailed Description	1439
9.103 hal_xsnor_base.h File Reference	1439
9.103.1 Detailed Description	1441
9.104 hal_xsnor_base.c File Reference	1441
9.104.1 Detailed Description	1443
9.105 bufstreams.c File Reference	1443
9.105.1 Detailed Description	1443
9.106 bufstreams.h File Reference	1443
9.106.1 Detailed Description	1444
9.107 chprintf.c File Reference	1444
9.107.1 Detailed Description	1444
9.108 chprintf.h File Reference	1445
9.108.1 Detailed Description	1445
9.109 chscanf.c File Reference	1445
9.109.1 Detailed Description	1446
9.110 chscanf.h File Reference	1446
9.110.1 Detailed Description	1446
9.111 memstreams.c File Reference	1446
9.111.1 Detailed Description	1447
9.112 memstreams.h File Reference	1447
9.112.1 Detailed Description	1447
9.113 nullstreams.c File Reference	1448
9.113.1 Detailed Description	1448
9.114 nullstreams.h File Reference	1448
9.114.1 Detailed Description	1449
9.115 hal.c File Reference	1449
9.115.1 Detailed Description	1449
9.116 hal_adc.c File Reference	1449
9.116.1 Detailed Description	1450

9.117 hal_buffered_serial.c File Reference	1450
9.118 hal_buffers.c File Reference	1450
9.118.1 Detailed Description	1451
9.119 hal_can.c File Reference	1451
9.119.1 Detailed Description	1452
9.120 hal_crypto.c File Reference	1452
9.120.1 Detailed Description	1454
9.121 hal_dac.c File Reference	1454
9.121.1 Detailed Description	1455
9.122 hal_efl.c File Reference	1455
9.122.1 Detailed Description	1456
9.123 hal_flash.c File Reference	1456
9.123.1 Detailed Description	1456
9.124 hal_gpt.c File Reference	1456
9.124.1 Detailed Description	1457
9.125 hal_i2c.c File Reference	1457
9.125.1 Detailed Description	1458
9.126 hal_i2s.c File Reference	1458
9.126.1 Detailed Description	1458
9.127 hal_icu.c File Reference	1459
9.127.1 Detailed Description	1459
9.128 hal_mac.c File Reference	1459
9.128.1 Detailed Description	1460
9.129 hal_mmc_spi.c File Reference	1460
9.129.1 Detailed Description	1461
9.130 hal_mmc_sdc.c File Reference	1462
9.130.1 Detailed Description	1462
9.131 hal_pal.c File Reference	1462
9.131.1 Detailed Description	1463
9.132 hal_pwm.c File Reference	1463
9.132.1 Detailed Description	1464
9.133 hal_queues.c File Reference	1464
9.133.1 Detailed Description	1465
9.134 hal_rtc.c File Reference	1465
9.134.1 Detailed Description	1466
9.135 hal_sdc.c File Reference	1466
9.135.1 Detailed Description	1467
9.136 hal_serial.c File Reference	1467
9.136.1 Detailed Description	1468
9.137 hal_serial_usb.c File Reference	1468
9.137.1 Detailed Description	1469
9.138 hal_sio.c File Reference	1470

9.138.1 Detailed Description	1471
9.139 hal_spi.c File Reference	1471
9.139.1 Detailed Description	1471
9.140 hal_spi_v1.inc File Reference	1471
9.140.1 Detailed Description	1472
9.141 hal_spi_v2.inc File Reference	1472
9.141.1 Detailed Description	1473
9.142 hal_st.c File Reference	1473
9.142.1 Detailed Description	1474
9.143 hal_trng.c File Reference	1474
9.143.1 Detailed Description	1475
9.144 hal_uart.c File Reference	1475
9.144.1 Detailed Description	1476
9.145 hal_usb.c File Reference	1476
9.145.1 Detailed Description	1477
9.146 hal_wdg.c File Reference	1477
9.146.1 Detailed Description	1478
9.147 hal_wspi.c File Reference	1478
9.147.1 Detailed Description	1478
9.148 hal_adc_ild.c File Reference	1479
9.148.1 Detailed Description	1479
9.149 hal_adc_ild.h File Reference	1479
9.149.1 Detailed Description	1480
9.150 hal_can_ild.c File Reference	1480
9.150.1 Detailed Description	1481
9.151 hal_can_ild.h File Reference	1481
9.151.1 Detailed Description	1482
9.152 hal_crypto_ild.c File Reference	1482
9.152.1 Detailed Description	1485
9.153 hal_crypto_ild.h File Reference	1485
9.153.1 Detailed Description	1488
9.154 hal_dac_ild.c File Reference	1488
9.154.1 Detailed Description	1488
9.155 hal_dac_ild.h File Reference	1488
9.155.1 Detailed Description	1489
9.156 hal_efl_ild.c File Reference	1490
9.156.1 Detailed Description	1490
9.157 hal_efl_ild.h File Reference	1490
9.157.1 Detailed Description	1491
9.158 hal_gpt_ild.c File Reference	1491
9.158.1 Detailed Description	1492
9.159 hal_gpt_ild.h File Reference	1492

9.159.1 Detailed Description	1493
9.160 hal_i2c_1ld.c File Reference	1493
9.160.1 Detailed Description	1494
9.161 hal_i2c_1ld.h File Reference	1494
9.161.1 Detailed Description	1495
9.162 hal_i2s_1ld.c File Reference	1495
9.162.1 Detailed Description	1495
9.163 hal_i2s_1ld.h File Reference	1496
9.163.1 Detailed Description	1496
9.164 hal_icu_1ld.c File Reference	1496
9.164.1 Detailed Description	1497
9.165 hal_icu_1ld.h File Reference	1497
9.165.1 Detailed Description	1498
9.166 hal_1ld.c File Reference	1498
9.166.1 Detailed Description	1499
9.167 hal_1ld.h File Reference	1499
9.167.1 Detailed Description	1499
9.168 hal_mac_1ld.c File Reference	1499
9.168.1 Detailed Description	1500
9.169 hal_mac_1ld.h File Reference	1500
9.169.1 Detailed Description	1501
9.170 hal_pal_1ld.c File Reference	1502
9.170.1 Detailed Description	1502
9.171 hal_pal_1ld.h File Reference	1502
9.171.1 Detailed Description	1504
9.172 hal_pwm_1ld.c File Reference	1504
9.172.1 Detailed Description	1504
9.173 hal_pwm_1ld.h File Reference	1505
9.173.1 Detailed Description	1506
9.174 hal_rtc_1ld.c File Reference	1506
9.174.1 Detailed Description	1506
9.175 hal_rtc_1ld.h File Reference	1506
9.175.1 Detailed Description	1507
9.176 hal_sdc_1ld.c File Reference	1507
9.176.1 Detailed Description	1508
9.177 hal_sdc_1ld.h File Reference	1508
9.177.1 Detailed Description	1510
9.178 hal_serial_1ld.c File Reference	1510
9.178.1 Detailed Description	1510
9.179 hal_serial_1ld.h File Reference	1511
9.179.1 Detailed Description	1511
9.180 hal_sio_1ld.c File Reference	1511

9.180.1 Detailed Description	1512
9.181 hal_sio_ild.h File Reference	1512
9.181.1 Detailed Description	1514
9.182 hal_spi_ild.c File Reference	1514
9.182.1 Detailed Description	1514
9.183 hal_spi_ild.h File Reference	1515
9.183.1 Detailed Description	1515
9.184 hal_spi_v2_ild.c File Reference	1516
9.184.1 Detailed Description	1516
9.185 hal_spi_v2_ild.h File Reference	1516
9.185.1 Detailed Description	1517
9.186 hal_st_ild.c File Reference	1517
9.186.1 Detailed Description	1518
9.187 hal_st_ild.h File Reference	1518
9.187.1 Detailed Description	1518
9.188 hal_trng_ild.c File Reference	1518
9.188.1 Detailed Description	1519
9.189 hal_trng_ild.h File Reference	1519
9.189.1 Detailed Description	1520
9.190 hal_uart_ild.c File Reference	1520
9.190.1 Detailed Description	1520
9.191 hal_uart_ild.h File Reference	1520
9.191.1 Detailed Description	1521
9.192 hal_usb_ild.c File Reference	1522
9.192.1 Detailed Description	1523
9.192.2 Variable Documentation	1523
9.192.2.1 in	1523
9.192.2.2 out	1523
9.193 hal_usb_ild.h File Reference	1523
9.193.1 Detailed Description	1525
9.194 hal_wdg_ild.c File Reference	1525
9.194.1 Detailed Description	1525
9.195 hal_wdg_ild.h File Reference	1526
9.195.1 Detailed Description	1526
9.196 hal_wspi_ild.c File Reference	1526
9.196.1 Detailed Description	1527
9.197 hal_wspi_ild.h File Reference	1527
9.197.1 Detailed Description	1528
9.198 halconf.h File Reference	1528
9.198.1 Detailed Description	1531
9.199 mcuconf.h File Reference	1531
9.199.1 Macro Definition Documentation	1531

9.199.1.1 PLATFORM_MCUCONF	1531
9.200 osal.c File Reference	1531
9.200.1 Detailed Description	1532
9.201 osal.h File Reference	1532
9.201.1 Detailed Description	1536
Index	1537

Chapter 1

Introduction

Author

Giovanni Di Sirio (gdisirio@users.sourceforge.net).

Features

- Hardware Abstraction Layer (HAL) supporting numerous high level device drivers abstracting common features in modern MCUs.
- Driver models for: ADC, CAN, DAC, EXT, GPT, I2C, I2S, ICU, MAC, MMC, PAL, PWM, RTC, SDC, Serial, UART, USB, USB-CDC, SPI, ST, WDG.
- Abstraction of streams.
- Abstraction of block devices.
- The drivers can be ported to virtually any MCU architecture thanks to an high level API.
- The HAL can be used with or without an RTOS.
- The HAL can be adapted to any reasonable RTOS thanks to an Operating System Abstraction Layer (OSAL). ChibiOS/RT and ChibiOS/NIL are supported out of the box.

Chapter 2

Deprecated List

Global [canReceive](#) (canp, mailbox, crfp, timeout)

Global [canTransmit](#) (canp, mailbox, ctfp, timeout)

Global [flashGetAddressOffset](#) ([BaseFlash](#) *devp, void *addr)

Global [flashGetOffsetAddress](#) ([BaseFlash](#) *devp, [flash_offset_t](#) offset)

Global [HAL_FAILED](#)

Global [HAL_SUCCESS](#)

Global [MMCConfig](#)

Global [MMCDriver](#)

Global [sdGetWouldBlock](#) ([SerialDriver](#) *sdp)

Global [sdPutWouldBlock](#) ([SerialDriver](#) *sdp)

Chapter 3

Topic Index

3.1 Topics

Here is a list of all topics with brief descriptions:

Compiler portability.	23
HAL	27
Peripheral Interfaces	29
Generic Flash Interface	29
Configuration	43
Normal Drivers	55
HAL Driver	57
ADC Driver	68
CAN Driver	89
Crypto Driver	110
DAC Driver	182
EFL Driver	204
GPT Driver	217
I2C Driver	233
I2S Driver	253
ICU Driver	264
MAC Driver	284
PAL Driver	308
PWM Driver	358
RTC Driver	385
SDC Driver	399
Serial Driver	435
SIO Driver	463
SPI Driver Selector	511
SP1v1 Driver	511
SP1v2 Driver	543
ST Driver	580
TRNG Driver	592
UART Driver	601
USB Driver	632
WDG Driver	685
WSPI Driver	692
Complex Drivers	721
Buffered SIO Driver	722
Managed Flash Storage Driver	737

MMC over SPI Driver	764
Serial NOR Flash Driver	792
Serial over USB Driver	813
Improved Serial NOR Driver	831
Base Serial NOR Driver class	831
Base Flash Interface	857
Macronix MX25 SNOR Driver	863
Micron N25Q Driver	894
Interfaces and Classes	916
Blocks	916
Abstract I/O Block Device	917
Streams	925
Abstract Streams Interface	925
Abstract I/O Channel Interface	931
Abstract Files Interface	941
Abstract Persistent Storage Interface	949
Memory Streams Class	953
Null Streams Class	955
Buffered Serial Class	957
Buffered Streams Class	965
Output Formatter Utility	967
Input Formatter Utility	973
Base Object	978
Inner Code	980
I/O Buffers Queues	980
MMC/SD Block Device	1005
I/O Bytes Queues	1030
Support Code	1053
MII/RMII Header	1053
USB CDC Header	1068
OSAL	1073

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

base_object_c	
hal_xsnor_base_c	1209
hal_xsnor_macronix_mx25_c	1215
hal_xsnor_micron_n25q_c	1221
BaseBlockDeviceVMT	1112
MMCSDBlockDeviceVMT	1262
SDCDriverVMT	1281
mmc_spi_driver_vmt	1258
BaseFlashVMT	1122
EFlashDriverVMT	1143
SNORDriverVMT	1297
BaseObject	1122
BaseBlockDevice	1109
MMCSDBlockDevice	1259
mmc_spi_driver_t	1255
BaseFlash	1119
SNORDriver	1295
hal_efl_driver	1175
BasePersistentStorage	1124
BaseSequentialStream	1126
BaseBufferedStream	1113
NullStream	1263
BaseChannel	1116
BaseAsynchronousChannel	1105
SerialUSBDriver	1285
hal_buffered_serial	1163
hal_buffered_siol_driver	1165
hal_serial_driver	1188
FileStream	1145
RTCDriverVMT	1276
MemoryStream	1237
BaseObjectVMT	1124
BasePersistentStorageVMT	1126

BaseSequentialStreamVMT	1128
BaseBufferedStreamVMT	1115
BaseChannelVMT	1118
BaseAsynchronousChannelVMT	1108
BufferedSerialVMT	1130
BufferedSIODriverVMT	1131
SerialDriverVMT	1282
SerialUSBDriverVMT	1288
sio_driver_vmt	1291
FileStreamVMT	1147
BlockDeviceInfo	1129
BufferedStream	
BufferedStreamAdapter	1133
BufferedStreamAdapterVMT	1134
CANRxFrame	1134
CANTxFrame	1137
cdc_linecoding_t	1140
CRYConfig	1141
CRYDriver	1142
event_source	1144
flash_descriptor_t	1148
flash_interface	1150
flash_interface_i	1152
hal_xsnor_base_c	1209
flash_interface_vmt	1153
flash_sector_descriptor_t	1155
GPTConfig	1156
GPTDriver	1157
hal_adc_config	1158
hal_adc_configuration_group	1159
hal_adc_driver	1160
hal_can_config	1167
hal_can_driver	1168
hal_dac_config	1171
hal_dac_conversion_group	1171
hal_dac_driver	1173
hal_efl_config	1175
hal_i2c_config	1177
hal_i2c_driver	1178
hal_i2s_config	1180
hal_i2s_driver	1181
hal_mac_config	1182
hal_mac_driver	1183
hal_mac_receive_descriptor	1185
hal_mac_transmit_descriptor	1186
hal_serial_config	1187
hal_sio_config	1190
hal_sio_driver	1191
hal_spi_config	1194
hal_spi_driver	1196
hal_trng_config	1198
hal_trng_driver	1199
hal_uart_config	1200
hal_uart_driver	1202
hal_wspi_config	1204
hal_wspi_driver	1205
hal_xsnor_base	1207

hal_xsnor_base_vmt	1210
hal_xsnor_macronix_mx25	1212
hal_xsnor_macronix_mx25_vmt	1216
hal_xsnor_micron_n25q	1218
hal_xsnor_micron_n25q_vmt	1222
halclkcfg_t	1224
HMACSHA256Context	1225
HMACSHA512Context	1226
ICUConfig	1226
ICUDriver	1228
io_buffers_queue	1229
io_queue	1233
IOBus	1235
MemStreamVMT	1239
mfs_bank_header_t	1239
mfs_data_header_t	1242
mfs_nocache_buffer	1244
mfs_record_descriptor_t	1245
mfs_transaction_op_t	1246
MFSCConfig	1247
MFSDriver	1250
mmc_spi_config_t	1253
NullStreamVMT	1265
PALConfig	1265
palevent_t	1266
PWMChannelConfig	1267
PWMConfig	1268
PWMDriver	1271
RTCAlarm	1272
RTCDateTime	1273
RTCDriver	1275
SDCConfig	1278
SDCDriver	1279
SerialUSBConfig	1284
SHA1Context	1289
SHA256Context	1290
SHA512Context	1291
snor_nocache_buffer	1293
SNORConfig	1294
threads_queue_t	1298
unpacked_mmc_cid_t	1299
unpacked_mmc_csd_t	1301
unpacked_sdc_cid_t	1306
unpacked_sdc_csd_10_t	1308
unpacked_sdc_csd_20_t	1312
USBConfig	1316
USBDescriptor	1318
USBDriver	1319
USBEndpointConfig	1322
USBInEndpointState	1326
USBOutEndpointState	1327
WDGConfig	1328
WDGDriver	1329
wspi_command_t	1330
xsnor_buffers	1332
xsnor_bus_configs	1333
xsnor_bus_spi	1335
xsnor_bus_wspi	1336

xsnor_commands	1338
xsnor_config	1339

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

BaseAsynchronousChannel	
Base asynchronous channel class	1105
BaseAsynchronousChannelVMT	
BaseAsynchronousChannel virtual methods table	1108
BaseBlockDevice	
Base block device class	1109
BaseBlockDeviceVMT	
BaseBlockDevice virtual methods table	1112
BaseBufferedStream	
Buffered stream class	1113
BaseBufferedStreamVMT	
BaseBufferedStream virtual methods table	1115
BaseChannel	
Base channel class	1116
BaseChannelVMT	
BaseChannel virtual methods table	1118
BaseFlash	
Base flash class	1119
BaseFlashVMT	
BaseFlash virtual methods table	1122
BaseObject	
Base object class	1122
BaseObjectVMT	
BaseObject virtual methods table	1124
BasePersistentStorage	
Base persistent storage class	1124
BasePersistentStorageVMT	
BasePersistentStorage virtual methods table	1126
BaseSequentialStream	
Base stream class	1126
BaseSequentialStreamVMT	
BaseSequentialStream virtual methods table	1128
BlockDeviceInfo	
Block device info	1129
BufferedSerialVMT	
BufferedSerial virtual methods table	1130

BufferedSIODriverVMT	
BufferedSIODriver virtual methods table	1131
BufferedStreamAdapter	
Buffered stream adapter object	1133
BufferedStreamAdapterVMT	
BufferedStreamAdapter virtual methods table	1134
CANRxFrame	
CAN received frame	1134
CANTxFrame	
CAN transmission frame	1137
cdc_linecoding_t	
Type of Line Coding structure	1140
CRYConfig	
Driver configuration structure	1141
CRYDriver	
Structure representing an CRY driver	1142
EFlashDriverVMT	
EFlash virtual methods table	1143
event_source	
Events source object	1144
FileStream	
Base file stream class	1145
FileStreamVMT	
FileStream virtual methods table	1147
flash_descriptor_t	
Type of a flash device descriptor	1148
flash_interface	
Structure representing a flash interface interface	1150
flash_interface_i	
Generic flash interface	1152
flash_interface_vmt	
Interface flash_interface_i virtual methods table	1153
flash_sector_descriptor_t	
Flash sector descriptor	1155
GPTConfig	
Driver configuration structure	1156
GPTDriver	
Structure representing a GPT driver	1157
hal_adc_config	
Driver configuration structure	1158
hal_adc_configuration_group	
Conversion group configuration structure	1159
hal_adc_driver	
Structure representing an ADC driver	1160
hal_buffered_serial	
Buffered serial channel class	1163
hal_buffered_sio_driver	
Buffered SIO driver class	1165
hal_can_config	
Type of a CAN configuration structure	1167
hal_can_driver	
Structure representing an CAN driver	1168
hal_dac_config	
Driver configuration structure	1171
hal_dac_conversion_group	
DAC Conversion group structure	1171
hal_dac_driver	
Structure representing a DAC driver	1173

hal_efl_config	Type of a structure representing a flash driver configuration	1175
hal_efl_driver	Structure representing an embedded flash driver	1175
hal_i2c_config	I2C driver configuration structure	1177
hal_i2c_driver	Structure representing an I2C driver	1178
hal_i2s_config	Driver configuration structure	1180
hal_i2s_driver	Structure representing an I2S driver	1181
hal_mac_config	Driver configuration structure	1182
hal_mac_driver	Structure representing a MAC driver	1183
hal_mac_receive_descriptor	Structure representing a MAC receive descriptor	1185
hal_mac_transmit_descriptor	Structure representing a MAC transmit descriptor	1186
hal_serial_config	PLATFORM Serial Driver configuration structure	1187
hal_serial_driver	Full duplex serial driver class	1188
hal_sio_config	Driver configuration structure	1190
hal_sio_driver	Structure representing a SIO driver	1191
hal_spi_config	Driver configuration structure	1194
hal_spi_driver	Structure representing an SPI driver	1196
hal_trng_config	Driver configuration structure	1198
hal_trng_driver	Structure representing a TRNG driver	1199
hal_uart_config	Driver configuration structure	1200
hal_uart_driver	Structure representing an UART driver	1202
hal_wspi_config	Driver configuration structure	1204
hal_wspi_driver	Structure representing an WSPI driver	1205
hal_xsnor_base	Structure representing a XSNOR base driver class	1207
hal_xsnor_base_c	Base class of all SNOR drivers	1209
hal_xsnor_base_vmt	Class hal_xsnor_base_c virtual methods table	1210
hal_xsnor_macronix_mx25	Structure representing a SNOR Macronix MX25 driver class	1212
hal_xsnor_macronix_mx25_c		1215
hal_xsnor_macronix_mx25_vmt	Class hal_xsnor_macronix_mx25_c virtual methods table	1216
hal_xsnor_micron_n25q	Structure representing a SNOR Micron N25Q driver class	1218
hal_xsnor_micron_n25q_c		1221

hal_xsnor_micron_n25q_vmt	
Class hal_xsnor_micron_n25q_c virtual methods table	1222
halclkcfg_t	
Type of a clock configuration structure	1224
HMACSHA256Context	
Type of a HMAC_SHA256 context	1225
HMACSHA512Context	
Type of a HMAC_SHA512 context	1226
ICUConfig	
Driver configuration structure	1226
ICUDriver	
Structure representing an ICU driver	1228
io_buffers_queue	
Structure of a generic buffers queue	1229
io_queue	
Generic I/O queue structure	1233
IOBus	
I/O bus descriptor	1235
MemoryStream	
Memory stream object	1237
MemStreamVMT	
MemStream virtual methods table	1239
mfs_bank_header_t	
Type of a bank header	1239
mfs_data_header_t	
Type of a data block header	1242
mfs_nocache_buffer	
Type of an non-cacheable MFS buffer	1244
mfs_record_descriptor_t	1245
mfs_transaction_op_t	
Type of a buffered write/erase operation within a transaction	1246
MFSCConfig	
Type of a MFS configuration structure	1247
MFSDriver	
Type of an MFS instance	1250
mmc_spi_config_t	
Type of a MMC/SD over SPI driver configuration structure	1253
mmc_spi_driver_t	
Structure representing a MMC/SD over SPI driver	1255
mmc_spi_driver_vmt	
MMCSDriver virtual methods table	1258
MMCSDBlockDevice	
MCC/SD block device class	1259
MMCSDBlockDeviceVMT	
MMCSDBlockDevice virtual methods table	1262
NullStream	
Null stream object	1263
NullStreamVMT	
NullStream virtual methods table	1265
PALConfig	
Generic I/O ports static initializer	1265
palevent_t	
Type of a PAL event record	1266
PWMChannelConfig	
Type of a PWM driver channel configuration structure	1267
PWMConfig	
Type of a PWM driver configuration structure	1268

PWMDriver	
Structure representing a PWM driver	1271
RTCAAlarm	
Type of a structure representing an RTC alarm time stamp	1272
RTCDateTime	
Type of a structure representing an RTC date/time stamp	1273
RTCDriver	
Structure representing an RTC driver	1275
RTCDriverVMT	
RTCDriver virtual methods table	1276
SDCConfig	
Driver configuration structure	1278
SDCDriver	
Structure representing an SDC driver	1279
SDCDriverVMT	
SDCDriver virtual methods table	1281
SerialDriverVMT	
SerialDriver virtual methods table	1282
SerialUSBConfig	
Serial over USB Driver configuration structure	1284
SerialUSBDriver	
Full duplex serial driver class	1285
SerialUSBDriverVMT	
SerialDriver virtual methods table	1288
SHA1Context	
Type of a SHA1 context	1289
SHA256Context	
Type of a SHA256 context	1290
SHA512Context	
Type of a SHA512 context	1291
sio_driver_vmt	
SIODriver virtual methods table	1291
snor_nocache_buffer	1293
SNORConfig	
Type of a SNOR configuration structure	1294
SNORDriver	
Type of SNOR flash class	1295
SNORDriverVMT	
SNOR virtual methods table	1297
threads_queue_t	
Type of a thread queue	1298
unpacked_mmc_cid_t	
Unpacked CID register from MMC	1299
unpacked_mmc_csd_t	
Unpacked CSD register from MMC	1301
unpacked_sdc_cid_t	
Unpacked CID register from SDC	1306
unpacked_sdc_csd_10_t	
Unpacked CSD v1.0 register from SDC	1308
unpacked_sdc_csd_20_t	
Unpacked CSD v2.0 register from SDC	1312
USBConfig	
Type of an USB driver configuration structure	1316
USBDescriptor	
Type of an USB descriptor	1318
USBDriver	
Structure representing an USB driver	1319

USBEndpointConfig	Type of an USB endpoint configuration structure	1322
USBInEndpointState	Type of an IN endpoint state structure	1326
USBOutEndpointState	Type of an OUT endpoint state structure	1327
WDGConfig	Driver configuration structure	1328
WDGDriver	Structure representing an WDG driver	1329
wspi_command_t	Type of a WSPI command descriptor	1330
xsnor_buffers	SNOR driver configuration	1332
xsnor_bus_configs	Union of possible bus configurations	1333
xsnor_bus_spi	SPI-specific configuration fields	1335
xsnor_bus_wspi	WSPI-specific configuration fields	1336
xsnor_commands	SNOR command configuration structure	1338
xsnor_config	SNOR driver configuration	1339

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

ccportab.h	Compiler portability layer	1341
hal.h	HAL subsystem header	1344
hal_adc.h	ADC Driver macros and structures	1347
hal_buffered_serial.h	Buffered Serial Driver code	1349
hal_buffers.h	I/O Buffers macros and structures	1350
hal_can.h	CAN Driver macros and structures	1352
hal_channels.h	I/O channels access	1354
hal_crypto.h	Cryptographic Driver macros and structures	1355
hal_dac.h	DAC Driver macros and structures	1359
hal_efl.h	Embedded Flash Driver macros and structures	1360
hal_files.h	Data files	1361
hal_flash.h	Generic flash driver class header	1362
hal_gpt.h	GPT Driver macros and structures	1364
hal_i2c.h	I2C Driver macros and structures	1366
hal_i2s.h	I2S Driver macros and structures	1367
hal_icu.h	ICU Driver macros and structures	1369
hal_ioblock.h	I/O block devices access	1370
hal_mac.h	MAC Driver macros and structures	1371

hal_mii.h	MII macros and structures	1373
hal_mmc_spi.h	MMC over SPI driver header	1376
hal_mmc_sdc.h	MMC/SD cards common header	1377
hal_objects.h	Base object	1382
hal_pal.h	I/O Ports Abstraction Layer macros, types and structures	1382
hal_persistent.h	Generic persistent storage class header	1386
hal_pwm.h	PWM Driver macros and structures	1387
hal_queues.h	I/O Queues macros and structures	1388
hal_rtc.h	RTC Driver macros and structures	1390
hal_sdc.h	SDC Driver macros and structures	1392
hal_serial.h	Serial Driver macros and structures	1393
hal_serial_usb.h	Serial over USB Driver macros and structures	1395
hal_sio.h	SIO Driver macros and structures	1397
hal_spi.h	SPI Driver selector macros and structures	1400
hal_spi_v1.h	SPI (v1) Driver macros and structures	1400
hal_spi_v2.h	SPI (v2) Driver macros and structures	1403
hal_st.h	ST Driver macros and structures	1405
hal_streams.h	Data streams	1406
hal_trng.h	TRNG Driver macros and structures	1407
hal_uart.h	UART Driver macros and structures	1408
hal_usb.h	USB Driver macros and structures	1410
hal_usb_cdc.h	USB CDC macros and structures	1414
hal_wdg.h	WDG Driver macros and structures	1416
hal_wspi.h	WSPI Driver macros and structures	1416
hal_buffered_sio.c	Buffered SIO Driver code	1419
hal_buffered_sio.h	Buffered SIO Driver macros and structures	1420
hal_mfs.c	Managed Flash Storage module code	1421
hal_mfs.h	Managed Flash Storage module header	1423
hal_serial_nor.c	Serial NOR serial flash driver code	1425

hal_serial_nor.h	
Serial NOR driver header	1426
hal_xsnor_macronix_mx25.c	
Generated SNOR Macronix MX25 source	1428
hal_xsnor_macronix_mx25.h	
Generated SNOR Macronix MX25 header	1429
hal_xsnor_macronix_mx25_impl.inc	
Template of SNOR Macronix MX25 source	1430
hal_xsnor_micron_n25q.c	
Generated SNOR Micron N25Q source	1434
hal_xsnor_micron_n25q.h	
Generated SNOR Micron N25Q header	1434
hal_xsnor_micron_n25q_impl.inc	
Template of SNOR Micron N25Q source	1435
hal_flash_interface.h	
Generated Flash Interface header	1438
hal_xsnor_base.h	
Generated SNOR Base Driver header	1439
hal_xsnor_base.c	
Generated SNOR Base Driver source	1441
bufstreams.c	
Buffered streams code	1443
bufstreams.h	
Buffered streams structures and macros	1443
chprintf.c	
Mini printf-like functionality	1444
chprintf.h	
Mini printf-like functionality	1445
chscanf.c	
Mini scanf-like functionality	1445
chscanf.h	
Mini scanf-like functionality	1446
memstreams.c	
Memory streams code	1446
memstreams.h	
Memory streams structures and macros	1447
nullstreams.c	
Null streams code	1448
nullstreams.h	
Null streams structures and macros	1448
hal.c	
HAL subsystem code	1449
hal_adc.c	
ADC Driver code	1449
hal_buffered_serial.c	
I/O Buffers code	1450
hal_buffers.c	
I/O Buffers code	1450
hal_can.c	
CAN Driver code	1451
hal_crypto.c	
Cryptographic Driver code	1452
hal_dac.c	
DAC Driver code	1454
hal_efl.c	
Embedded Flash Driver code	1455
hal_flash.c	
Generic flash driver class code	1456

hal_gpt.c	
GPT Driver code	1456
hal_i2c.c	
I2C Driver code	1457
hal_i2s.c	
I2S Driver code	1458
hal_icu.c	
ICU Driver code	1459
hal_mac.c	
MAC Driver code	1459
hal_mmc_spi.c	
MMC over SPI driver code	1460
hal_mmc_sdc.c	
MMC/SD cards common code	1462
hal_pal.c	
I/O Ports Abstraction Layer code	1462
hal_pwm.c	
PWM Driver code	1463
hal_queues.c	
I/O Queues code	1464
hal_rtc.c	
RTC Driver code	1465
hal_sdc.c	
SDC Driver code	1466
hal_serial.c	
Serial Driver code	1467
hal_serial_usb.c	
Serial over USB Driver code	1468
hal_sio.c	
SIO Driver code	1470
hal_spi.c	
SPI Driver selector code	1471
hal_spi_v1.inc	
SPI (v1) Driver code	1471
hal_spi_v2.inc	
SPI (v2) Driver code	1472
hal_st.c	
ST Driver code	1473
hal_trng.c	
TRNG Driver code	1474
hal_uart.c	
UART Driver code	1475
hal_usb.c	
USB Driver code	1476
hal_wdg.c	
WDG Driver code	1477
hal_wspi.c	
WSPI Driver code	1478
hal_adc_ild.c	
PLATFORM ADC subsystem low level driver source	1479
hal_adc_ild.h	
PLATFORM ADC subsystem low level driver header	1479
hal_can_ild.c	
PLATFORM CAN subsystem low level driver source	1480
hal_can_ild.h	
PLATFORM CAN subsystem low level driver header	1481
hal_crypto_ild.c	
PLATFORM cryptographic subsystem low level driver source	1482

hal_crypto_lld.h	PLATFORM cryptographic subsystem low level driver header	1485
hal_dac_lld.c	PLATFORM DAC subsystem low level driver source	1488
hal_dac_lld.h	PLATFORM DAC subsystem low level driver header	1488
hal_efl_lld.c	PLATFORM Embedded Flash subsystem low level driver source	1490
hal_efl_lld.h	PLATFORM Embedded Flash subsystem low level driver header	1490
hal_gpt_lld.c	PLATFORM GPT subsystem low level driver source	1491
hal_gpt_lld.h	PLATFORM GPT subsystem low level driver header	1492
hal_i2c_lld.c	PLATFORM I2C subsystem low level driver source	1493
hal_i2c_lld.h	PLATFORM I2C subsystem low level driver header	1494
hal_i2s_lld.c	PLATFORM I2S subsystem low level driver source	1495
hal_i2s_lld.h	PLATFORM I2S subsystem low level driver header	1496
hal_icu_lld.c	PLATFORM ADC subsystem low level driver source	1496
hal_icu_lld.h	PLATFORM ICU subsystem low level driver header	1497
hal_lld.c	PLATFORM HAL subsystem low level driver source	1498
hal_lld.h	PLATFORM HAL subsystem low level driver header	1499
hal_mac_lld.c	PLATFORM MAC subsystem low level driver source	1499
hal_mac_lld.h	PLATFORM MAC subsystem low level driver header	1500
hal_pal_lld.c	PLATFORM PAL subsystem low level driver source	1502
hal_pal_lld.h	PLATFORM PAL subsystem low level driver header	1502
hal_pwm_lld.c	PLATFORM PWM subsystem low level driver source	1504
hal_pwm_lld.h	PLATFORM PWM subsystem low level driver header	1505
hal_rtc_lld.c	PLATFORM RTC subsystem low level driver source	1506
hal_rtc_lld.h	PLATFORM RTC subsystem low level driver header	1506
hal_sdc_lld.c	PLATFORM SDC subsystem low level driver source	1507
hal_sdc_lld.h	PLATFORM SDC subsystem low level driver header	1508
hal_serial_lld.c	PLATFORM serial subsystem low level driver source	1510
hal_serial_lld.h	PLATFORM serial subsystem low level driver header	1511
hal_sio_lld.c	PLATFORM SIO subsystem low level driver source	1511
hal_sio_lld.h	PLATFORM SIO subsystem low level driver header	1512

hal_spi_lld.c	PLATFORM SPI (v1) subsystem low level driver source	1514
hal_spi_lld.h	PLATFORM SPI (v1) subsystem low level driver header	1515
hal_spi_v2_lld.c	PLATFORM SPI (v2) subsystem low level driver source	1516
hal_spi_v2_lld.h	PLATFORM SPI (v2) subsystem low level driver header	1516
hal_st_lld.c	PLATFORM ST subsystem low level driver source	1517
hal_st_lld.h	PLATFORM ST subsystem low level driver header	1518
hal_trng_lld.c	PLATFORM TRNG subsystem low level driver source	1518
hal_trng_lld.h	PLATFORM TRNG subsystem low level driver header	1519
hal_uart_lld.c	PLATFORM UART subsystem low level driver source	1520
hal_uart_lld.h	PLATFORM UART subsystem low level driver header	1520
hal_usb_lld.c	PLATFORM USB subsystem low level driver source	1522
hal_usb_lld.h	PLATFORM USB subsystem low level driver header	1523
hal_wdg_lld.c	WDG Driver subsystem low level driver source template	1525
hal_wdg_lld.h	WDG Driver subsystem low level driver header template	1526
hal_wspi_lld.c	PLATFORM WSPI subsystem low level driver source	1526
hal_wspi_lld.h	PLATFORM WSPI subsystem low level driver header	1527
halconf.h	HAL configuration header	1528
mcuconf.h		1531
osal.c	OSAL module code	1531
osal.h	OSAL module header	1532

Chapter 7

Topic Documentation

7.1 Compiler portability.

7.1.1 Detailed Description

Compiler abstraction macros

- `#define CC_SECTION(s)`
Allocates a variable or function to a specific section.
- `#define CC_WEAK __attribute__((weak))`
Marks a function or variable as a weak symbol.
- `#define CC_USED __attribute__((used))`
Marks a function or variable as used.
- `#define CC_ALIGN_DATA(n)`
Enforces alignment of the variable declared afterward.
- `#define CC_ALIGN_CODE(n)`
Enforces alignment of a function declared afterward.
- `#define CC_PACK __attribute__((packed))`
Enforces packing of the structure declared afterward.
- `#define CC_NO_INLINE __attribute__((noinline))`
Marks a function as not inlineable.
- `#define CC_FORCE_INLINE __attribute__((always_inline))`
Enforces a function inline.
- `#define CC_NO_RETURN __attribute__((noreturn))`
Marks a function as non-returning.
- `#define CC_ROMCONST const`
Enforces a variable in a ROM area.
- `#define CC_LIKELY(x)`
Marks a boolean expression as likely true.
- `#define CC_UNLIKELY(x)`
Marks a boolean expression as likely false.

7.1.2 Macro Definition Documentation

7.1.2.1 CC_SECTION

```
#define CC_SECTION(  
    s)
```

Value:

```
__attribute__((section(s)))
```

Allocates a variable or function to a specific section.

Note

If the compiler does not support such a feature then this macro must not be defined or it could originate errors.

7.1.2.2 CC_WEAK

```
#define CC_WEAK __attribute__((weak))
```

Marks a function or variable as a weak symbol.

Note

If the compiler does not support such a feature then this macro must not be defined or it could originate errors.

7.1.2.3 CC_USED

```
#define CC_USED __attribute__((used))
```

Marks a function or variable as used.

The compiler or linker shall not remove the marked function or variable regardless if it is referred or not in the code.

Note

If the compiler does not support such a feature then this macro must not be defined or it could originate errors.

7.1.2.4 CC_ALIGN_DATA

```
#define CC_ALIGN_DATA(  
    n)
```

Value:

```
__attribute__((aligned(n)))
```

Enforces alignment of the variable declared afterward.

Note

If the compiler does not support such a feature then this macro must not be defined or it could originate errors.

7.1.2.5 CC_ALIGN_CODE

```
#define CC_ALIGN_CODE(  
    n)
```

Value:

```
__attribute__((aligned(n)))
```

Enforces alignment of a function declared afterward.

Note

If the compiler does not support such a feature then this macro must not be defined or it could originate errors.

7.1.2.6 CC_PACK

```
#define CC_PACK __attribute__((packed))
```

Enforces packing of the structure declared afterward.

Note

If the compiler does not support such a feature then this macro must not be defined or it could originate errors.

7.1.2.7 CC_NO_INLINE

```
#define CC_NO_INLINE __attribute__((noinline))
```

Marks a function as not inlineable.

Note

Can be implemented as an empty macro if not supported by the compiler.

7.1.2.8 CC_FORCE_INLINE

```
#define CC_FORCE_INLINE __attribute__((always_inline))
```

Enforces a function inline.

Note

Can be implemented as an empty macro if not supported by the compiler.

7.1.2.9 CC_NO_RETURN

```
#define CC_NO_RETURN __attribute__((noreturn))
```

Marks a function as non-returning.

Note

Can be implemented as an empty macro if not supported by the compiler.

7.1.2.10 CC_ROMCONST

```
#define CC_ROMCONST const
```

Enforces a variable in a ROM area.

Note

Can be implemented as an empty macro if not supported by the compiler.

7.1.2.11 CC_LIKELY

```
#define CC_LIKELY(  
    x)
```

Value:

```
__builtin_expect(!!(x), 1)
```

Marks a boolean expression as likely true.

Parameters

in	x	a valid expression

7.1.2.12 CC_UNLIKELY

```
#define CC_UNLIKELY(  
    x)
```

Value:

```
__builtin_expect(!!(x), 0)
```

Marks a boolean expression as likely false.

Parameters

in	x	a valid expression

7.2 HAL

Hardware Abstraction Layer.

7.2.1 Detailed Description

Hardware Abstraction Layer.

Under ChibiOS the set of the various device driver interfaces is called the HAL subsystem: Hardware Abstraction Layer. The HAL is the abstract interface between ChibiOS applications and hardware.

7.2.2 HAL Device Drivers Architecture

The HAL contains several kind of modules:

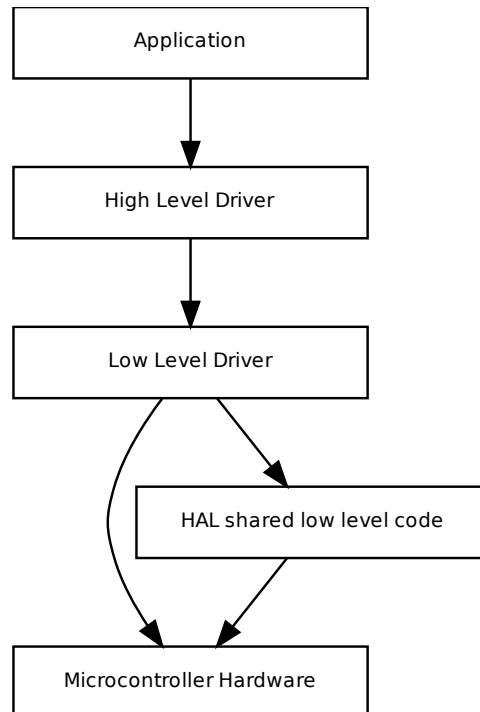
- Normal Device Drivers
- Complex Device Drivers
- Interfaces
- Inner Code

7.2.3 HAL Normal Device Drivers

Normal device are meant to interface the application to the underlying hardware through an high level API. Normal Device Drivers are split in two layers:

- High Level Device Driver (**HLD**). This layer contains the definitions of the driver's APIs and the platform independent part of the driver.
An HLD is composed by two files:
 - `hal_<driver>.c`, the HLD implementation file. This file must be included in the Makefile in order to use the driver.
 - `hal_<driver>.h`, the HLD header file. This file is implicitly included by the HAL header file [hal.h](#).
- Low Level Device Driver (**LLD**). This layer contains the platform dependent part of the driver.
A LLD is composed by two files:
 - `hal_<driver>_lld.c`, the LLD implementation file. This file must be included in the Makefile in order to use the driver.
 - `hal_<driver>_lld.h`, the LLD header file. This file is implicitly included by the HLD header file.

7.2.3.1 Diagram



7.2.4 HAL Complex Device Drivers

It is a class of device drivers that offer an high level API but do not use the hardware directly. Complex device drivers use other drivers for accessing the machine resources.

7.2.5 HAL Interfaces

An interface is a binary structure allowing the access to a service using virtual functions. This allows to create drivers that can be accessed using a common interface. The concept of interface is commonly found in object-oriented languages like Java or C++, their meaning in ChibiOS/HAL is exactly the same.

7.2.6 HAL Inner Code

Some modules are shared among multiple device drivers and are not necessarily meant to be used by the application layer.

Topics

- [Peripheral Interfaces](#)
HAL Abstract Peripheral Interfaces.
- [Configuration](#)
HAL Configuration.
- [Normal Drivers](#)
HAL Normal Drivers.
- [Complex Drivers](#)
HAL Complex Drivers.
- [Interfaces and Classes](#)
HAL Interfaces and Classes.
- [Inner Code](#)
HAL Inner Code.
- [Support Code](#)
HAL Support Code.
- [OSAL](#)
Operating System Abstraction Layer.

7.2.7 Peripheral Interfaces

HAL Abstract Peripheral Interfaces.

7.2.7.1 Detailed Description

HAL Abstract Peripheral Interfaces.

Topics

- [Generic Flash Interface](#)
HAL Generic Flash Driver Interface.

7.2.7.2 Generic Flash Interface

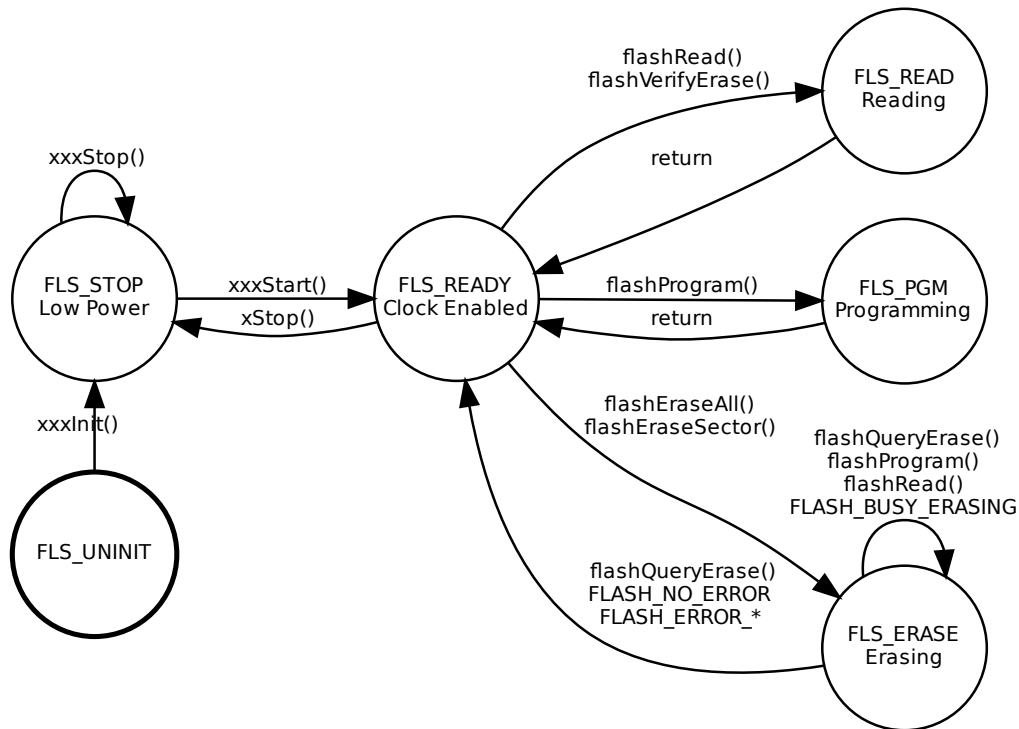
HAL Generic Flash Driver Interface.

7.2.7.2.1 Detailed Description

HAL Generic Flash Driver Interface.

7.2.7.2.2 Driver State Machine

The flash driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



Flash attributes

- `#define FLASH_ATTR_ERASED_IS_ONE 0x00000001U`
Defines one as the erased bit state.
- `#define FLASH_ATTR_MEMORY_MAPPED 0x00000002U`
The memory is accessible in a memory mapped mode.
- `#define FLASH_ATTR_REWRITABLE 0x00000004U`
Programmed pages can be programmed again.
- `#define FLASH_ATTR_ECC_CAPABLE 0x00000008U`
The memory is protected by an ECC mechanism.
- `#define FLASH_ATTR_ECC_ZERO_LINE_CAPABLE 0x00000010U`
The device is able to overwrite zero to a line.
- `#define FLASH_ATTR_SUSPEND_ERASE_CAPABLE 0x00000020U`
The device is able to suspend erase operations.

Macro Functions (BaseFlash)

- `#define getBaseFlash(ip)`
Instance getter.
- `#define flashGetDescriptor(ip)`
Gets the flash descriptor structure.
- `#define flashRead(ip, offset, n, rp)`
Read operation.
- `#define flashProgram(ip, offset, n, pp)`
Program operation.
- `#define flashStartEraseAll(ip)`
Starts a whole-device erase operation.
- `#define flashStartEraseSector(ip, sector)`
Starts an sector erase operation.
- `#define flashQueryErase(ip, msec)`
Queries the driver for erase operation progress.
- `#define flashVerifyErase(ip, sector)`
Returns the erase state of a sector.
- `#define flashAcquireExclusive(ip)`
Acquires exclusive access to flash.
- `#define flashReleaseExclusive(ip)`
Releases exclusive access to flash.

Data Structures

- `struct flash\_sector\_descriptor\_t`
Flash sector descriptor.
- `struct flash\_descriptor\_t`
Type of a flash device descriptor.
- `struct BaseFlashVMT`
[BaseFlash](#) virtual methods table.
- `struct BaseFlash`
Base flash class.

Macros

- `#define \_base\_flash\_methods\_alone`
[BaseFlash](#) specific methods.
- `#define \_base\_flash\_methods`
[BaseFlash](#) specific methods with inherited ones.
- `#define \_base\_flash\_data`
[BaseFlash](#) specific data.

Typedefs

- `typedef uint32_t flash\_offset\_t`
Type of a flash offset.
- `typedef uint32_t flash\_sector\_t`
Type of a flash sector number.

Enumerations

- enum `flash_state_t` {
`FLASH_UNINIT` = 0 , `FLASH_STOP` = 1 , `FLASH_READY` = 2 , `FLASH_READ` = 3 ,
`FLASH_PGM` = 4 , `FLASH_ERASE` = 5 }
Driver state machine possible states.
- enum `flash_error_t` {
`FLASH_NO_ERROR` = 0 , `FLASH_BUSY_ERASING` = 1 , `FLASH_ERROR_READ` = 2 , `FLASH_ERROR_PROGRAM`
= 3 ,
`FLASH_ERROR_ERASE` = 4 , `FLASH_ERROR_VERIFY` = 5 , `FLASH_ERROR_HW_FAILURE` = 6 ,
`FLASH_ERROR_UNIMPLEMENTED` = 7 }
Type of a flash error code.

Functions

- `flash_error_t flashWaitErase (BaseFlash *devp)`
Waits until the current erase operation is finished.
- `flash_offset_t flashGetSectorOffset (BaseFlash *devp, flash_sector_t sector)`
Returns the offset of a sector.
- `uint32_t flashGetSectorSize (BaseFlash *devp, flash_sector_t sector)`
Returns the size of a sector.
- `flash_sector_t flashGetOffsetSector (BaseFlash *devp, flash_offset_t offset)`
Returns the sector of an offset.
- `void * flashGetOffsetAddress (BaseFlash *devp, flash_offset_t offset)`
Get absolute address from offset.
- `flash_offset_t flashGetAddressOffset (BaseFlash *devp, void *addr)`
Get offset from absolute address.

7.2.7.2.3 Macro Definition Documentation

7.2.7.2.3.1 FLASH_ATTR_ERASED_IS_ONE

```
#define FLASH_ATTR_ERASED_IS_ONE 0x00000001U
```

Defines one as the erased bit state.

7.2.7.2.3.2 FLASH_ATTR_MEMORY_MAPPED

```
#define FLASH_ATTR_MEMORY_MAPPED 0x00000002U
```

The memory is accessible in a memory mapped mode.

7.2.7.2.3.3 FLASH_ATTR_REWRITABLE

```
#define FLASH_ATTR_REWRITABLE 0x00000004U
```

Programmed pages can be programmed again.

Note

This is incompatible and alternative to `FLASH_ATTR_ECC_CAPABLE`.

7.2.7.2.3.4 FLASH_ATTR_ECC_CAPABLE

```
#define FLASH_ATTR_ECC_CAPABLE 0x00000008U
```

The memory is protected by an ECC mechanism.

Note

This usually puts restrictions on the program operations.

- Program operations can only happen at offsets aligned to write page boundaries.
- The programmed data size must be a multiple of the write page size.
- Programmed pages cannot be re-programmed.

7.2.7.2.3.5 FLASH_ATTR_ECC_ZERO_LINE_CAPABLE

```
#define FLASH_ATTR_ECC_ZERO_LINE_CAPABLE 0x00000010U
```

The device is able to overwrite zero to a line.

Note

This attribute is only meaningful for those devices that support ECC, so also `FLASH_ATTR_ECC_CAPABLE` must be specified.

7.2.7.2.3.6 FLASH_ATTR_SUSPEND_ERASE_CAPABLE

```
#define FLASH_ATTR_SUSPEND_ERASE_CAPABLE 0x00000020U
```

The device is able to suspend erase operations.

7.2.7.2.3.7 _base_flash_methods_alone

```
#define _base_flash_methods_alone
```

Value:

```
/* Get flash device attributes.*/
const flash_descriptor_t * (*get_descriptor)(void *instance);
/* Read operation.*/
flash_error_t (*read)(void *instance, flash_offset_t offset,
                      size_t n, uint8_t *rp);
/* Program operation.*/
flash_error_t (*program)(void *instance, flash_offset_t offset,
                        size_t n, const uint8_t *pp);
/* Erase whole flash device.*/
flash_error_t (*start_erase_all)(void *instance);
/* Erase single sector.*/
flash_error_t (*start_erase_sector)(void *instance,
                                   flash_sector_t sector);
flash_error_t (*query_erase)(void *instance, uint32_t *wait_time);
/* Verify erase single sector.*/
flash_error_t (*verify_erase)(void *instance, flash_sector_t sector);
/* Acquire exclusive use of flash.*/
flash_error_t (*acquire_exclusive)(void *instance);
/* Release exclusive use of flash.*/
flash_error_t (*release_exclusive)(void *instance);
```

BaseFlash specific methods.

7.2.7.2.3.8 `_base_flash_methods`

```
#define _base_flash_methods
```

Value:

```
_base_object_methods          \
_base_flash_methods_alone
```

[BaseFlash](#) specific methods with inherited ones.

7.2.7.2.3.9 `_base_flash_data`

```
#define _base_flash_data
```

Value:

```
_base_object_data          \
/* Driver state.*/          \
flash_state_t               state;
```

[BaseFlash](#) specific data.

7.2.7.2.3.10 `getBaseFlash`

```
#define getBaseFlash(
    ip)
```

Value:

```
((BaseFlash *) &(ip)->vmt)
```

Instance getter.

This special method is used to get the instance of this class object from a derived class.

7.2.7.2.3.11 `flashGetDescriptor`

```
#define flashGetDescriptor(
    ip)
```

Value:

```
(ip)->vmt->get_descriptor(ip)
```

Gets the flash descriptor structure.

Parameters

in	<i>ip</i>	pointer to a BaseFlash or derived class

Returns

A flash device descriptor.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.3.12 flashRead

```
#define flashRead(  
    ip,  
    offset,  
    n,  
    rp)
```

Value:

```
(ip)->vmt->read(ip, offset, n, rp)
```

Read operation.

Parameters

in	<i>ip</i>	pointer to a BaseFlash or derived class
in	<i>offset</i>	flash offset
in	<i>n</i>	number of bytes to be read
out	<i>rp</i>	pointer to the data buffer

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_READ</i>	if the read operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.3.13 flashProgram

```
#define flashProgram(  
    ip,  
    offset,  
    n,  
    pp)
```

Value:

```
(ip)->vmt->program(ip, offset, n, pp)
```

Program operation.

Parameters

in	<i>ip</i>	pointer to a BaseFlash or derived class
in	<i>offset</i>	flash offset
in	<i>n</i>	number of bytes to be programmed
in	<i>pp</i>	pointer to the data buffer

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_PROGRAM</i>	if the program operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.3.14 flashStartEraseAll

```
#define flashStartEraseAll(  
    ip)
```

Value:

```
(ip)->vmt->start_erase_all(ip)
```

Starts a whole-device erase operation.

Parameters

in	<i>ip</i>	pointer to a BaseFlash or derived class

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.3.15 flashStartEraseSector

```
#define flashStartEraseSector(  
    ip,  
    sector)
```

Value:

```
(ip)->vmt->start_erase_sector(ip, sector)
```

Starts an sector erase operation.

Parameters

in	<i>ip</i>	pointer to a BaseFlash or derived class
in	<i>sector</i>	sector to be erased

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.3.16 flashQueryErase

```
#define flashQueryErase(
    ip,
    msec)
```

Value:

```
(ip)->vmt->query_erase(ip, msec)
```

Queries the driver for erase operation progress.

Parameters

in	<i>ip</i>	pointer to a BaseFlash or derived class
out	<i>msec</i>	recommended time, in milliseconds, that should be spent before calling this function again, can be NULL

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_ERASE</i>	if the erase operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.3.17 flashVerifyErase

```
#define flashVerifyErase(  
    ip,  
    sector)
```

Value:

```
(ip)->vmt->verify_erase(ip, sector)
```

Returns the erase state of a sector.

Parameters

in	<i>ip</i>	pointer to a BaseFlash or derived class
in	<i>sector</i>	sector to be verified

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if the sector is erased.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_VERIFY</i>	if the verify operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.3.18 flashAcquireExclusive

```
#define flashAcquireExclusive(  
    ip)
```

Value:

```
(ip)->vmt->acquire_exclusive(ip)
```

Acquires exclusive access to flash.

Parameters

in	<i>ip</i>	pointer to a BaseFlash or derived class

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if the access is obtained.
<i>FLASH_ERROR_UNIMPLEMENTED</i>	if exclusive access not enabled

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.3.19 flashReleaseExclusive

```
#define flashReleaseExclusive(  
    ip)
```

Value:

```
(ip)->vmt->release_exclusive(ip)
```

Releases exclusive access to flash.

Parameters

in	<i>ip</i>	pointer to a BaseFlash or derived class

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if the access is released.
<i>FLASH_ERROR_UNIMPLEMENTED</i>	if exclusive access not enabled

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.4 Typedef Documentation

7.2.7.2.4.1 flash_offset_t

```
typedef uint32_t flash_offset_t
```

Type of a flash offset.

7.2.7.2.4.2 flash_sector_t

```
typedef uint32_t flash_sector_t
```

Type of a flash sector number.

7.2.7.2.5 Enumeration Type Documentation

7.2.7.2.5.1 flash_state_t

```
enum flash_state_t
```

Driver state machine possible states.

Enumerator

FLASH_UNINIT	
FLASH_STOP	
FLASH_READY	
FLASH_READ	
FLASH_PGM	
FLASH_ERASE	

7.2.7.2.5.2 flash_error_t

enum `flash_error_t`

Type of a flash error code.

Enumerator

FLASH_NO_ERROR	
FLASH_BUSY_ERASING	
FLASH_ERROR_READ	
FLASH_ERROR_PROGRAM	
FLASH_ERROR_ERASE	
FLASH_ERROR_VERIFY	
FLASH_ERROR_HW_FAILURE	
FLASH_ERROR_UNIMPLEMENTED	

7.2.7.2.6 Function Documentation

7.2.7.2.6.1 flashWaitErase()

```
flash_error_t flashWaitErase (
    BaseFlash * devp)
```

Waits until the current erase operation is finished.

Parameters

in	<code>devp</code>	pointer to a <code>BaseFlash</code> object

Returns

An error code.

Return values

<code>FLASH_NO_ERROR</code>	if there is no erase operation in progress.
<code>FLASH_ERROR_ERASE</code>	if the erase operation failed.
<code>FLASH_ERROR_HW_FAILURE</code>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.6.2 flashGetSectorOffset()

```
flash_offset_t flashGetSectorOffset (  
    BaseFlash * devp,  
    flash_sector_t sector)
```

Returns the offset of a sector.

Parameters

in	<i>devp</i>	pointer to a BaseFlash object
in	<i>sector</i>	flash sector number

Returns

Sector offset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.6.3 flashGetSectorSize()

```
uint32_t flashGetSectorSize (  
    BaseFlash * devp,  
    flash_sector_t sector)
```

Returns the size of a sector.

Parameters

in	<i>devp</i>	pointer to a BaseFlash object
in	<i>sector</i>	flash sector number

Returns

Sector size.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.6.4 flashGetOffsetSector()

```
flash_sector_t flashGetOffsetSector (  
    BaseFlash * devp,  
    flash_offset_t offset)
```

Returns the sector of an offset.

Parameters

in	<i>devp</i>	pointer to a BaseFlash object
in	<i>offset</i>	flash offset

Returns

Flash sector.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.7.2.6.5 flashGetOffsetAddress()

```
void * flashGetOffsetAddress (
    BaseFlash * devp,
    flash_offset_t offset)
```

Get absolute address from offset.

Parameters

in	<i>devp</i>	pointer to a BaseFlash object
in	<i>offset</i>	flash offset

Returns

A pointer to the offset.

Deprecated**7.2.7.2.6.6 flashGetAddressOffset()**

```
flash_offset_t flashGetAddressOffset (
    BaseFlash * devp,
    void * addr)
```

Get offset from absolute address.

Parameters

in	<i>devp</i>	pointer to a BaseFlash object
in	<i>addr</i>	pointer

Returns

flash offset

Deprecated

7.2.8 Configuration

HAL Configuration.

7.2.8.1 Detailed Description

HAL Configuration.

The file `halconf.h` contains the high level settings for all the drivers supported by the HAL. The low level, platform dependent, settings are contained in the `mcuconf.h` file instead and are describe in the various platforms reference manuals.

Macros

- `#define _CHIBIOS_HAL_CONF_`
- `#define _CHIBIOS_HAL_CONF_VER_9_0_`
- `#define HAL_USE_PAL TRUE`
Enables the PAL subsystem.
- `#define HAL_USE_ADC TRUE`
Enables the ADC subsystem.
- `#define HAL_USE_CAN TRUE`
Enables the CAN subsystem.
- `#define HAL_USE_CRY TRUE`
Enables the cryptographic subsystem.
- `#define HAL_USE_DAC TRUE`
Enables the DAC subsystem.
- `#define HAL_USE_EFL TRUE`
Enables the EFlash subsystem.
- `#define HAL_USE_GPT TRUE`
Enables the GPT subsystem.
- `#define HAL_USE_I2C TRUE`
Enables the I2C subsystem.
- `#define HAL_USE_I2S TRUE`
Enables the I2S subsystem.
- `#define HAL_USE_ICU TRUE`
Enables the ICU subsystem.
- `#define HAL_USE_MAC TRUE`
Enables the MAC subsystem.
- `#define HAL_USE_MMC_SPI TRUE`
Enables the MMC_SPI subsystem.
- `#define HAL_USE_PWM TRUE`
Enables the PWM subsystem.
- `#define HAL_USE_RTC TRUE`
Enables the RTC subsystem.
- `#define HAL_USE_SDC TRUE`
Enables the SDC subsystem.
- `#define HAL_USE_SERIAL TRUE`
Enables the SERIAL subsystem.
- `#define HAL_USE_SERIAL_USB TRUE`
Enables the SERIAL over USB subsystem.

- `#define HAL_USE_SIO TRUE`
Enables the SIO subsystem.
- `#define HAL_USE_SPI TRUE`
Enables the SPI subsystem.
- `#define HAL_USE_TRNG TRUE`
Enables the TRNG subsystem.
- `#define HAL_USE_UART TRUE`
Enables the UART subsystem.
- `#define HAL_USE_USB TRUE`
Enables the USB subsystem.
- `#define HAL_USE_WDG TRUE`
Enables the WDG subsystem.
- `#define HAL_USE_WSPI TRUE`
Enables the WSPI subsystem.
- `#define PAL_USE_CALLBACKS FALSE`
Enables synchronous APIs.
- `#define PAL_USE_WAIT FALSE`
Enables synchronous APIs.
- `#define ADC_USE_WAIT TRUE`
Enables synchronous APIs.
- `#define ADC_USE_MUTUAL_EXCLUSION TRUE`
Enables the `adcAcquireBus()` and `adcReleaseBus()` APIs.
- `#define CAN_USE_SLEEP_MODE TRUE`
Sleep mode related APIs inclusion switch.
- `#define CAN_ENFORCE_USE_CALLBACKS FALSE`
Enforces the driver to use direct callbacks rather than OSAL events.
- `#define HAL_CRY_USE_FALLBACK FALSE`
Enables the SW fall-back of the cryptographic driver.
- `#define HAL_CRY_ENFORCE_FALLBACK FALSE`
Makes the driver forcibly use the fall-back implementations.
- `#define DAC_USE_WAIT TRUE`
Enables synchronous APIs.
- `#define DAC_USE_MUTUAL_EXCLUSION TRUE`
Enables the `dacAcquireBus()` and `dacReleaseBus()` APIs.
- `#define I2C_ENABLE_SLAVE_MODE FALSE`
Slave mode API enable switch.
- `#define I2C_USE_MUTUAL_EXCLUSION TRUE`
Enables the mutual exclusion APIs on the I2C bus.
- `#define MAC_USE_ZERO_COPY TRUE`
Enables the zero-copy API.
- `#define MAC_USE_EVENTS TRUE`
Enables an event sources for incoming packets.
- `#define MMC_IDLE_TIMEOUT_MS 1000`
Timeout before assuming a failure while waiting for card idle.
- `#define MMC_USE_MUTUAL_EXCLUSION TRUE`
Mutual exclusion on the SPI bus.
- `#define SDC_INIT_RETRY 100`
Number of initialization attempts before rejecting the card.
- `#define SDC_MMC_SUPPORT TRUE`
Include support for MMC cards.
- `#define SDC_NICE_WAITING TRUE`

- Delays insertions.*
- #define `SDC_INIT_OCR_V20` `0x50FF8000U`
OCR initialization constant for V20 cards.
- #define `SDC_INIT_OCR` `0x80100000U`
OCR initialization constant for non-V20 cards.
- #define `SERIAL_DEFAULT_BITRATE` `38400`
Default bit rate.
- #define `SERIAL_BUFFERS_SIZE` `16`
Serial buffers size.
- #define `SIO_DEFAULT_BITRATE` `38400`
Default bit rate.
- #define `SIO_USE_SYNCHRONIZATION` `TRUE`
Support for thread synchronization API.
- #define `SERIAL_USB_BUFFERS_SIZE` `256`
Serial over USB buffers size.
- #define `SERIAL_USB_BUFFERS_NUMBER` `2`
Serial over USB number of buffers.
- #define `SPI_USE_WAIT` `TRUE`
Enables synchronous APIs.
- #define `SPI_USE_ASSERT_ON_ERROR` `TRUE`
Inserts an assertion on function errors before returning.
- #define `SPI_USE_MUTUAL_EXCLUSION` `TRUE`
Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.
- #define `SPI_SELECT_MODE` `SPI_SELECT_MODE_PAD`
Handling method for SPI CS line.
- #define `UART_USE_WAIT` `TRUE`
Enables synchronous APIs.
- #define `UART_USE_MUTUAL_EXCLUSION` `TRUE`
Enables the `uartAcquireBus()` and `uartReleaseBus()` APIs.
- #define `USB_USE_WAIT` `TRUE`
Enables synchronous APIs.
- #define `WSPI_USE_WAIT` `TRUE`
Enables synchronous APIs.
- #define `WSPI_USE_MUTUAL_EXCLUSION` `TRUE`
Enables the `wspiAcquireBus()` and `wspiReleaseBus()` APIs.

7.2.8.2 Macro Definition Documentation

7.2.8.2.1 `_CHIBIOS_HAL_CONF_`

```
#define _CHIBIOS_HAL_CONF_
```

7.2.8.2.2 `_CHIBIOS_HAL_CONF_VER_9_0_`

```
#define _CHIBIOS_HAL_CONF_VER_9_0_
```

7.2.8.2.3 HAL_USE_PAL

```
#define HAL_USE_PAL TRUE
```

Enables the PAL subsystem.

7.2.8.2.4 HAL_USE_ADC

```
#define HAL_USE_ADC TRUE
```

Enables the ADC subsystem.

7.2.8.2.5 HAL_USE_CAN

```
#define HAL_USE_CAN TRUE
```

Enables the CAN subsystem.

7.2.8.2.6 HAL_USE_CRY

```
#define HAL_USE_CRY TRUE
```

Enables the cryptographic subsystem.

7.2.8.2.7 HAL_USE_DAC

```
#define HAL_USE_DAC TRUE
```

Enables the DAC subsystem.

7.2.8.2.8 HAL_USE_EFL

```
#define HAL_USE_EFL TRUE
```

Enables the EFlash subsystem.

7.2.8.2.9 HAL_USE_GPT

```
#define HAL_USE_GPT TRUE
```

Enables the GPT subsystem.

7.2.8.2.10 HAL_USE_I2C

```
#define HAL_USE_I2C TRUE
```

Enables the I2C subsystem.

7.2.8.2.11 HAL_USE_I2S

```
#define HAL_USE_I2S TRUE
```

Enables the I2S subsystem.

7.2.8.2.12 HAL_USE_ICU

```
#define HAL_USE_ICU TRUE
```

Enables the ICU subsystem.

7.2.8.2.13 HAL_USE_MAC

```
#define HAL_USE_MAC TRUE
```

Enables the MAC subsystem.

7.2.8.2.14 HAL_USE_MMC_SPI

```
#define HAL_USE_MMC_SPI TRUE
```

Enables the MMC_SPI subsystem.

7.2.8.2.15 HAL_USE_PWM

```
#define HAL_USE_PWM TRUE
```

Enables the PWM subsystem.

7.2.8.2.16 HAL_USE_RTC

```
#define HAL_USE_RTC TRUE
```

Enables the RTC subsystem.

7.2.8.2.17 HAL_USE_SDC

```
#define HAL_USE_SDC TRUE
```

Enables the SDC subsystem.

7.2.8.2.18 HAL_USE_SERIAL

```
#define HAL_USE_SERIAL TRUE
```

Enables the SERIAL subsystem.

7.2.8.2.19 HAL_USE_SERIAL_USB

```
#define HAL_USE_SERIAL_USB TRUE
```

Enables the SERIAL over USB subsystem.

7.2.8.2.20 HAL_USE_SIO

```
#define HAL_USE_SIO TRUE
```

Enables the SIO subsystem.

7.2.8.2.21 HAL_USE_SPI

```
#define HAL_USE_SPI TRUE
```

Enables the SPI subsystem.

7.2.8.2.22 HAL_USE_TRNG

```
#define HAL_USE_TRNG TRUE
```

Enables the TRNG subsystem.

7.2.8.2.23 HAL_USE_UART

```
#define HAL_USE_UART TRUE
```

Enables the UART subsystem.

7.2.8.2.24 HAL_USE_USB

```
#define HAL_USE_USB TRUE
```

Enables the USB subsystem.

7.2.8.2.25 HAL_USE_WDG

```
#define HAL_USE_WDG TRUE
```

Enables the WDG subsystem.

7.2.8.2.26 HAL_USE_WSPI

```
#define HAL_USE_WSPI TRUE
```

Enables the WSPI subsystem.

7.2.8.2.27 PAL_USE_CALLBACKS

```
#define PAL_USE_CALLBACKS FALSE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.28 PAL_USE_WAIT

```
#define PAL_USE_WAIT FALSE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.29 ADC_USE_WAIT

```
#define ADC_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.30 ADC_USE_MUTUAL_EXCLUSION

```
#define ADC_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `adcAcquireBus()` and `adcReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.31 CAN_USE_SLEEP_MODE

```
#define CAN_USE_SLEEP_MODE TRUE
```

Sleep mode related APIs inclusion switch.

7.2.8.2.32 CAN_ENFORCE_USE_CALLBACKS

```
#define CAN_ENFORCE_USE_CALLBACKS FALSE
```

Enforces the driver to use direct callbacks rather than OSAL events.

7.2.8.2.33 HAL_CRY_USE_FALLBACK

```
#define HAL_CRY_USE_FALLBACK FALSE
```

Enables the SW fall-back of the cryptographic driver.

When enabled, this option, activates a fall-back software implementation for algorithms not supported by the underlying hardware.

Note

Fall-back implementations may not be present for all algorithms.

7.2.8.2.34 HAL_CRY_ENFORCE_FALLBACK

```
#define HAL_CRY_ENFORCE_FALLBACK FALSE
```

Makes the driver forcibly use the fall-back implementations.

7.2.8.2.35 DAC_USE_WAIT

```
#define DAC_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.36 DAC_USE_MUTUAL_EXCLUSION

```
#define DAC_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `dacAcquireBus()` and `dacReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.37 I2C_ENABLE_SLAVE_MODE

```
#define I2C_ENABLE_SLAVE_MODE FALSE
```

Slave mode API enable switch.

Note

The low level driver must support this capability.

7.2.8.2.38 I2C_USE_MUTUAL_EXCLUSION

```
#define I2C_USE_MUTUAL_EXCLUSION TRUE
```

Enables the mutual exclusion APIs on the I2C bus.

7.2.8.2.39 MAC_USE_ZERO_COPY

```
#define MAC_USE_ZERO_COPY TRUE
```

Enables the zero-copy API.

7.2.8.2.40 MAC_USE_EVENTS

```
#define MAC_USE_EVENTS TRUE
```

Enables an event sources for incoming packets.

7.2.8.2.41 MMC_IDLE_TIMEOUT_MS

```
#define MMC_IDLE_TIMEOUT_MS 1000
```

Timeout before assuming a failure while waiting for card idle.

Note

Time is in milliseconds.

7.2.8.2.42 MMC_USE_MUTUAL_EXCLUSION

```
#define MMC_USE_MUTUAL_EXCLUSION TRUE
```

Mutual exclusion on the SPI bus.

7.2.8.2.43 SDC_INIT_RETRY

```
#define SDC_INIT_RETRY 100
```

Number of initialization attempts before rejecting the card.

Note

Attempts are performed at 10mS intervals.

7.2.8.2.44 SDC_MMC_SUPPORT

```
#define SDC_MMC_SUPPORT TRUE
```

Include support for MMC cards.

Note

MMC support is not yet implemented so this option must be kept at FALSE.

7.2.8.2.45 SDC_NICE_WAITING

```
#define SDC_NICE_WAITING TRUE
```

Delays insertions.

If enabled this options inserts delays into the MMC waiting routines releasing some extra CPU time for the threads with lower priority, this may slow down the driver a bit however.

7.2.8.2.46 SDC_INIT_OCR_V20

```
#define SDC_INIT_OCR_V20 0x50FF8000U
```

OCR initialization constant for V20 cards.

7.2.8.2.47 SDC_INIT_OCR

```
#define SDC_INIT_OCR 0x80100000U
```

OCR initialization constant for non-V20 cards.

7.2.8.2.48 SERIAL_DEFAULT_BITRATE

```
#define SERIAL_DEFAULT_BITRATE 38400
```

Default bit rate.

Configuration parameter, this is the baud rate selected for the default configuration.

7.2.8.2.49 SERIAL_BUFFERS_SIZE

```
#define SERIAL_BUFFERS_SIZE 16
```

Serial buffers size.

Configuration parameter, you can change the depth of the queue buffers depending on the requirements of your application.

Note

The default is 16 bytes for both the transmission and receive buffers.

7.2.8.2.50 SIO_DEFAULT_BITRATE

```
#define SIO_DEFAULT_BITRATE 38400
```

Default bit rate.

Configuration parameter, this is the baud rate selected for the default configuration.

7.2.8.2.51 SIO_USE_SYNCHRONIZATION

```
#define SIO_USE_SYNCHRONIZATION TRUE
```

Support for thread synchronization API.

7.2.8.2.52 SERIAL_USB_BUFFERS_SIZE

```
#define SERIAL_USB_BUFFERS_SIZE 256
```

Serial over USB buffers size.

Configuration parameter, the buffer size must be a multiple of the USB data endpoint maximum packet size.

Note

The default is 256 bytes for both the transmission and receive buffers.

7.2.8.2.53 SERIAL_USB_BUFFERS_NUMBER

```
#define SERIAL_USB_BUFFERS_NUMBER 2
```

Serial over USB number of buffers.

Note

The default is 2 buffers.

7.2.8.2.54 SPI_USE_WAIT

```
#define SPI_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.55 SPI_USE_ASSERT_ON_ERROR

```
#define SPI_USE_ASSERT_ON_ERROR TRUE
```

Inserts an assertion on function errors before returning.

7.2.8.2.56 SPI_USE_MUTUAL_EXCLUSION

```
#define SPI_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.57 SPI_SELECT_MODE

```
#define SPI_SELECT_MODE SPI_SELECT_MODE_PAD
```

Handling method for SPI CS line.

Note

Disabling this option saves both code and data space.

7.2.8.2.58 UART_USE_WAIT

```
#define UART_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.59 UART_USE_MUTUAL_EXCLUSION

```
#define UART_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `uartAcquireBus()` and `uartReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.60 USB_USE_WAIT

```
#define USB_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.61 WSPI_USE_WAIT

```
#define WSPI_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.8.2.62 WSPI_USE_MUTUAL_EXCLUSION

```
#define WSPI_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `wspiAcquireBus()` and `wspiReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.9 Normal Drivers

HAL Normal Drivers.

7.2.9.1 Detailed Description

HAL Normal Drivers.

Topics

- [HAL Driver](#)
Hardware Abstraction Layer.
- [ADC Driver](#)
Generic ADC Driver.
- [CAN Driver](#)
Generic CAN Driver.
- [Crypto Driver](#)
Generic Crypto Driver.
- [DAC Driver](#)
Generic DAC Driver.
- [EFL Driver](#)
Generic Embedded Flash Driver.
- [GPT Driver](#)
Generic GPT Driver.
- [I2C Driver](#)
Generic I2C Driver.
- [I2S Driver](#)
Generic I2S Driver.
- [ICU Driver](#)
Generic ICU Driver.
- [MAC Driver](#)
Generic MAC Driver.
- [PAL Driver](#)
I/O Ports Abstraction Layer.
- [PWM Driver](#)
Generic PWM Driver.
- [RTC Driver](#)
Generic RTC Driver.
- [SDC Driver](#)
Generic SD Card Driver.
- [Serial Driver](#)
Generic Serial Driver.
- [SIO Driver](#)
Generic SIO Driver.
- [SPI Driver Selector](#)
SPI Driver selector.
- [SPiV1 Driver](#)
Generic SPI Driver (legacy).
- [SPiV2 Driver](#)
Generic SPI Driver.
- [ST Driver](#)
Generic System Tick Driver.
- [TRNG Driver](#)
Generic True Random Numbers Generator Driver.
- [UART Driver](#)
Generic UART Driver.
- [USB Driver](#)
Generic USB Driver.
- [WDG Driver](#)
Generic WDG Driver.
- [WSPI Driver](#)
Generic WSPI Driver.

7.2.9.2 HAL Driver

Hardware Abstraction Layer.

7.2.9.2.1 Detailed Description

Hardware Abstraction Layer.

The HAL (Hardware Abstraction Layer) driver performs the system initialization and includes the platform support code shared by the other drivers. This driver does contain any API function except for a general initialization function `halInit()` that must be invoked before any HAL service can be used, usually the HAL initialization should be performed immediately before the kernel initialization.

Some HAL driver implementations also offer a custom early clock setup function that can be invoked before the C runtime initialization in order to accelerate the startup time.

ChibiOS/HAL version identification

- `#define CH_HAL_VERSION "9.0.0"`
HAL version string.
- `#define CH_HAL_MAJOR 9`
HAL version major number.
- `#define CH_HAL_MINOR 0`
HAL version minor number.
- `#define CH_HAL_PATCH 0`
HAL version patch number.

Return codes

- `#define HAL_SUCCESS false`
HAL operation success.
- `#define HAL_FAILED true`
HAL operation failed.

Return codes for HAL functions

- `#define HAL_RET_SUCCESS MSG_OK`
- `#define HAL_RET_CONFIG_ERROR (msg_t)-16`
Configuration error.
- `#define HAL_RET_NO_RESOURCE (msg_t)-17`
A required resource is not available.
- `#define HAL_RET_HW_BUSY (msg_t)-18`
The peripheral is busy.
- `#define HAL_RET_HW_FAILURE (msg_t)-19`
Peripheral failure.
- `#define HAL_RET_UNKNOWN_CTL (msg_t)-20`
Unknown control code.
- `#define HAL_RET_IS_INVALID (msg_t)-21`
Invalid instance pointer.

Platform identification macros

- `#define PLATFORM_NAME "templates"`

Data Structures

- struct `halclkcfg_t`
Type of a clock configuration structure.

Macros

- `#define __CHIBIOS_HAL__`
ChibiOS/HAL identification macro.
- `#define CH_HAL_STABLE 1`
Stable release flag.
- `#define HAL_USE_PAL FALSE`
- `#define HAL_USE_ADC FALSE`
- `#define HAL_USE_CAN FALSE`
- `#define HAL_USE_CRY FALSE`
- `#define HAL_USE_DAC FALSE`
- `#define HAL_USE_EFL FALSE`
- `#define HAL_USE_GPT FALSE`
- `#define HAL_USE_I2C FALSE`
- `#define HAL_USE_I2S FALSE`
- `#define HAL_USE_ICU FALSE`
- `#define HAL_USE_MAC FALSE`
- `#define HAL_USE_PWM FALSE`
- `#define HAL_USE_RTC FALSE`
- `#define HAL_USE_SERIAL FALSE`
- `#define HAL_USE_SDC FALSE`
- `#define HAL_USE_SIO FALSE`
- `#define HAL_USE_SPI FALSE`
- `#define HAL_USE_TRNG FALSE`
- `#define HAL_USE_UART FALSE`
- `#define HAL_USE_USB FALSE`
- `#define HAL_USE_WDG FALSE`
- `#define HAL_USE_WSPI FALSE`
- `#define HAL_LLD_SELECT_SPI_V2 TRUE`
Requires use of SPIv2 driver model.
- `#define HAL_LLD_USE_CLOCK_MANAGEMENT`
Specifies implementation of dynamic clock management.

Typedefs

- typedef unsigned `halclkpt_t`
Type of a clock point identifier.
- typedef uint32_t `halfreq_t`
Type of a clock point frequency in Hz.

Functions

- void `halInit` (void)
HAL initialization.
- static `halfreq_t` `halClockGetPointX` (`halclkpt_t` clkpt)
Returns the frequency of a clock point in Hz.
- static bool `halClockSwitchMode` (const `halclkcfg_t` *ccp)
Switches to a different clock configuration.
- void `hal_lld_init` (void)
Low level HAL driver initialization.
- bool `hal_lld_clock_switch_mode` (const `halclkcfg_t` *ccp)
Switches to a different clock configuration.
- `halfreq_t` `hal_lld_get_clock_point` (`halclkpt_t` clkpt)
Returns the frequency of a clock point in Hz.

7.2.9.2.2 Macro Definition Documentation

7.2.9.2.2.1 __CHIBIOS_HAL__

```
#define __CHIBIOS_HAL__
```

ChibiOS/HAL identification macro.

7.2.9.2.2.2 CH_HAL_STABLE

```
#define CH_HAL_STABLE 1
```

Stable release flag.

7.2.9.2.2.3 CH_HAL_VERSION

```
#define CH_HAL_VERSION "9.0.0"
```

HAL version string.

7.2.9.2.2.4 CH_HAL_MAJOR

```
#define CH_HAL_MAJOR 9
```

HAL version major number.

7.2.9.2.2.5 CH_HAL_MINOR

```
#define CH_HAL_MINOR 0
```

HAL version minor number.

7.2.9.2.2.6 CH_HAL_PATCH

```
#define CH_HAL_PATCH 0
```

HAL version patch number.

7.2.9.2.2.7 HAL_SUCCESS

```
#define HAL_SUCCESS false
```

HAL operation success.

Deprecated

7.2.9.2.2.8 HAL_FAILED

```
#define HAL_FAILED true
```

HAL operation failed.

Deprecated

7.2.9.2.2.9 HAL_RET_SUCCESS

```
#define HAL_RET_SUCCESS MSG_OK
```

7.2.9.2.2.10 HAL_RET_CONFIG_ERROR

```
#define HAL_RET_CONFIG_ERROR (msg_t)-16
```

Configuration error.

An error has been detected in the driver configuration structure.

7.2.9.2.2.11 HAL_RET_NO_RESOURCE

```
#define HAL_RET_NO_RESOURCE (msg_t)-17
```

A required resource is not available.

One of the resources required for driver operations is not available.

7.2.9.2.2.12 HAL_RET_HW_BUSY

```
#define HAL_RET_HW_BUSY (msg_t)-18
```

The peripheral is busy.

The peripheral is not available or taken by some other system actor.

7.2.9.2.2.13 HAL_RET_HW_FAILURE

```
#define HAL_RET_HW_FAILURE (msg_t)-19
```

Peripheral failure.

Peripheral failed, for example HW timeouts.

7.2.9.2.2.14 HAL_RET_UNKNOWN_CTL

```
#define HAL_RET_UNKNOWN_CTL (msg_t)-20
```

Unknown control code.

7.2.9.2.2.15 HAL_RET_IS_INVALID

```
#define HAL_RET_IS_INVALID (msg_t)-21
```

Invalid instance pointer.

7.2.9.2.2.16 HAL_USE_PAL

```
#define HAL_USE_PAL FALSE
```

7.2.9.2.2.17 HAL_USE_ADC

```
#define HAL_USE_ADC FALSE
```

7.2.9.2.2.18 HAL_USE_CAN

```
#define HAL_USE_CAN FALSE
```

7.2.9.2.2.19 HAL_USE_CRY

```
#define HAL_USE_CRY FALSE
```

7.2.9.2.2.20 HAL_USE_DAC

```
#define HAL_USE_DAC FALSE
```

7.2.9.2.2.21 HAL_USE_EFL

```
#define HAL_USE_EFL FALSE
```

7.2.9.2.2.22 HAL_USE_GPT

```
#define HAL_USE_GPT FALSE
```

7.2.9.2.2.23 HAL_USE_I2C

```
#define HAL_USE_I2C FALSE
```

7.2.9.2.2.24 HAL_USE_I2S

```
#define HAL_USE_I2S FALSE
```

7.2.9.2.2.25 HAL_USE_ICU

```
#define HAL_USE_ICU FALSE
```

7.2.9.2.2.26 HAL_USE_MAC

```
#define HAL_USE_MAC FALSE
```

7.2.9.2.2.27 HAL_USE_PWM

```
#define HAL_USE_PWM FALSE
```

7.2.9.2.2.28 HAL_USE_RTC

```
#define HAL_USE_RTC FALSE
```

7.2.9.2.2.29 HAL_USE_SERIAL

```
#define HAL_USE_SERIAL FALSE
```

7.2.9.2.2.30 HAL_USE_SDC

```
#define HAL_USE_SDC FALSE
```

7.2.9.2.2.31 HAL_USE_SIO

```
#define HAL_USE_SIO FALSE
```

7.2.9.2.2.32 HAL_USE_SPI

```
#define HAL_USE_SPI FALSE
```

7.2.9.2.2.33 HAL_USE_TRNG

```
#define HAL_USE_TRNG FALSE
```

7.2.9.2.2.34 HAL_USE_UART

```
#define HAL_USE_UART FALSE
```

7.2.9.2.2.35 HAL_USE_USB

```
#define HAL_USE_USB FALSE
```

7.2.9.2.2.36 HAL_USE_WDG

```
#define HAL_USE_WDG FALSE
```

7.2.9.2.2.37 HAL_USE_WSPI

```
#define HAL_USE_WSPI FALSE
```

7.2.9.2.2.38 HAL_LLD_SELECT_SPI_V2

```
#define HAL_LLD_SELECT_SPI_V2 TRUE
```

Requires use of SPIv2 driver model.

7.2.9.2.2.39 HAL_LLD_USE_CLOCK_MANAGEMENT

```
#define HAL_LLD_USE_CLOCK_MANAGEMENT
```

Specifies implementation of dynamic clock management.

7.2.9.2.2.40 PLATFORM_NAME

```
#define PLATFORM_NAME "templates"
```

7.2.9.2.3 Typedef Documentation

7.2.9.2.3.1 halclkpt_t

```
typedef unsigned halclkpt_t
```

Type of a clock point identifier.

7.2.9.2.3.2 halfreq_t

```
typedef uint32_t halfreq_t
```

Type of a clock point frequency in Hz.

7.2.9.2.4 Function Documentation

7.2.9.2.4.1 hallnit()

```
void hallnit (  
    void )
```

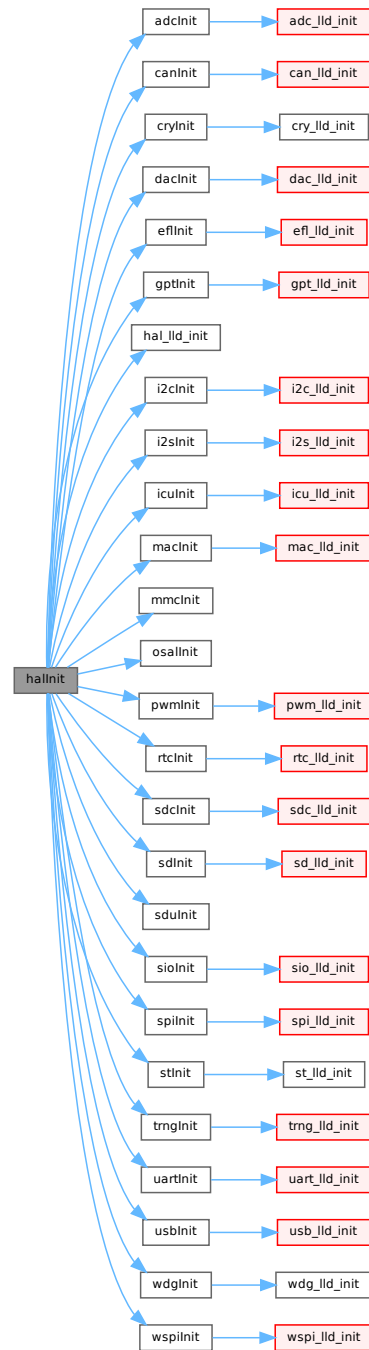
HAL initialization.

This function invokes the low level initialization code then initializes all the drivers enabled in the HAL. Finally the board-specific initialization is performed by invoking `boardInit()` (usually defined in `board.c`).

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.2.4.2 halClockGetPointX()**

```

halfreq_t halClockGetPointX (
    halclkpt_t clkpt) [inline], [static]

```

Returns the frequency of a clock point in Hz.

Parameters

in	<i>clkpt</i>	clock point to be returned

Returns

The clock point frequency in Hz or zero if the frequency is unknown.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

Here is the call graph for this function:



7.2.9.2.4.3 halClockSwitchMode()

```
bool halClockSwitchMode (  
    const halclkcfg\_t * ccp) [inline], [static]
```

Switches to a different clock configuration.

Parameters

in	<i>ccp</i>	pointer to clock a halclkcfg_t structure

Returns

The clock switch result.

Return values

<i>false</i>	if the clock switch succeeded
<i>true</i>	if the clock switch failed

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:

**7.2.9.2.4.4 hal_lld_init()**

```
void hal_lld_init (
    void )
```

Low level HAL driver initialization.

Function Class:

Not an API, this function is for internal use only.

7.2.9.2.4.5 hal_lld_clock_switch_mode()

```
bool hal_lld_clock_switch_mode (
    const halclkcfg_t * ccp)
```

Switches to a different clock configuration.

Parameters

in	ccp	pointer to clock a <code>halclkcfg_t</code> structure

Returns

The clock switch result.

Return values

<i>false</i>	if the clock switch succeeded
<i>true</i>	if the clock switch failed

Function Class:

Not an API, this function is for internal use only.

7.2.9.2.4.6 hal_lld_get_clock_point()

```
halfreq_t hal_lld_get_clock_point (
    halclkpt_t clkpt)
```

Returns the frequency of a clock point in Hz.

Parameters

in	<i>clkpt</i>	clock point to be returned

Returns

The clock point frequency in Hz or zero if the frequency is unknown.

Function Class:

Not an API, this function is for internal use only.

7.2.9.3 ADC Driver

Generic ADC Driver.

7.2.9.3.1 Detailed Description

Generic ADC Driver.

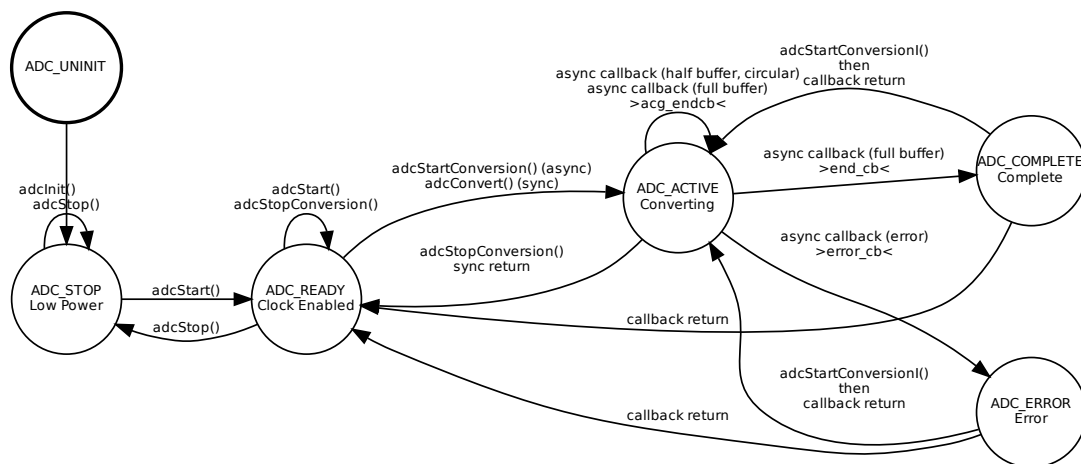
This module implements a generic ADC (Analog to Digital Converter) driver supporting a variety of buffer and conversion modes.

Precondition

In order to use the ADC driver the `HAL_USE_ADC` option must be enabled in `halconf.h`.

7.2.9.3.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



7.2.9.3.3 ADC Operations

The ADC driver is quite complex, an explanation of the terminology and of the operational details follows.

7.2.9.3.3.1 ADC Conversion Groups

The `ADCCConversionGroup` is the objects that specifies a physical conversion operation. This structure contains some standard fields and several implementation-dependent fields.

The standard fields define the CG mode, the number of channels belonging to the CG and the optional callbacks. The implementation-dependent fields specify the physical ADC operation mode, the analog channels belonging to the group and any other implementation-specific setting. Usually the extra fields just mirror the physical ADC registers, please refer to the vendor's MCU Reference Manual for details about the available settings. Details are also available into the documentation of the ADC low level drivers and in the various sample applications.

7.2.9.3.3.2 ADC Conversion Modes

The driver supports several conversion modes:

- **One Shot**, the driver performs a single group conversion then stops.
- **Linear Buffer**, the driver performs a series of group conversions then stops. This mode is like a one shot conversion repeated N times, the buffer pointer increases after each conversion. The buffer is organized as an $S(CG) \times N$ samples matrix, when $S(CG)$ is the conversion group size (number of channels) and N is the buffer depth (number of repeated conversions).
- **Circular Buffer**, much like the linear mode but the operation does not stop when the buffer is filled, it is automatically restarted with the buffer pointer wrapping back to the buffer base.

7.2.9.3.3.3 ADC Callbacks

The driver is able to invoke callbacks during the conversion process. A callback is invoked when the operation has been completed or, in circular mode, when the buffer has been filled and the operation is restarted. In circular mode a callback is also invoked when the buffer is half filled.

The "half filled" and "filled" callbacks in circular mode allow to implement "streaming processing" of the sampled data, while the driver is busy filling one half of the buffer the application can process the other half, this allows for continuous interleaved operations.

The driver is not thread safe for performance reasons, if you need to access the ADC bus from multiple threads then use the `adcAcquireBus()` and `adcReleaseBus()` APIs in order to gain exclusive access.

ADC configuration options

- `#define ADC_USE_WAIT TRUE`
Enables synchronous APIs.
- `#define ADC_USE_MUTUAL_EXCLUSION TRUE`
Enables the `adcAcquireBus()` and `adcReleaseBus()` APIs.

Macro Functions

- `#define adcsBufferComplete(adcp)`
Buffer state.

Low level driver helper macros

- `#define _adc_reset_i(adcp)`
Resumes a thread waiting for a conversion completion.
- `#define _adc_reset_s(adcp)`
Resumes a thread waiting for a conversion completion.
- `#define _adc_wakeup_isr(adcp)`
Wakes up the waiting thread.
- `#define _adc_timeout_isr(adcp)`
Wakes up the waiting thread with a timeout message.
- `#define _adc_isr_half_code(adcp)`
Common ISR code, half buffer event.
- `#define _adc_isr_full_code(adcp)`
Common ISR code, full buffer event.
- `#define _adc_isr_error_code(adcp, err)`
Common ISR code, error event.

Possible ADC errors mask bits.

- `#define ADC_ERR_DMAFAILURE 1U`
- `#define ADC_ERR_OVERFLOW 2U`
- `#define ADC_ERR_AWD 4U`

PLATFORM configuration options

- `#define PLATFORM_ADC_USE_ADC1 FALSE`
ADC1 driver enable switch.

Data Structures

- struct `hal_adc_configuration_group`
Conversion group configuration structure.
- struct `hal_adc_config`
Driver configuration structure.
- struct `hal_adc_driver`
Structure representing an ADC driver.

Macros

- `#define adc_lld_driver_fields`
Low level fields of the ADC driver structure.
- `#define adc_lld_config_fields`
Low level fields of the ADC configuration structure.
- `#define adc_lld_configuration_group_fields`
Low level fields of the ADC configuration structure.

Typedefs

- typedef struct [hal_adc_driver](#) ADCDriver
Type of a structure representing an ADC driver.
- typedef struct [hal_adc_config](#) ADCConfig
Type of a structure representing an ADC driver configuration.
- typedef struct [hal_adc_configuration_group](#) ADCConversionGroup
Conversion group configuration structure.
- typedef void(* [adccallback_t](#)) (ADCDriver *adcp)
Type of an ADC notification callback.
- typedef void(* [adcerrorcallback_t](#)) (ADCDriver *adcp, [adcerror_t](#) err)
Type of an ADC error callback.
- typedef uint16_t [adcsample_t](#)
ADC sample data type.
- typedef uint16_t [adc_channels_num_t](#)
Channels number in a conversion group.
- typedef uint32_t [adcerror_t](#)
Type of an ADC error mask.

Enumerations

- enum [adcstate_t](#) {
 [ADC_UNINIT](#) = 0 , [ADC_STOP](#) = 1 , [ADC_READY](#) = 2 , [ADC_ACTIVE](#) = 3 ,
 [ADC_COMPLETE](#) = 4 , [ADC_ERROR](#) = 5 }
Driver state machine possible states.

Functions

- void [adclnit](#) (void)
ADC Driver initialization.
- void [adcObjectInit](#) (ADCDriver *adcp)
Initializes the standard part of a [ADCDriver](#) structure.
- [msg_t](#) [adcStart](#) (ADCDriver *adcp, const [ADCConfig](#) *config)
Configures and activates the ADC peripheral.
- void [adcStop](#) (ADCDriver *adcp)
Deactivates the ADC peripheral.
- void [adcStartConversion](#) (ADCDriver *adcp, const [ADCConversionGroup](#) *grpp, [adcsample_t](#) *samples, [size_t](#) depth)
Starts an ADC conversion.
- void [adcStartConversionI](#) (ADCDriver *adcp, const [ADCConversionGroup](#) *grpp, [adcsample_t](#) *samples, [size_t](#) depth)
Starts an ADC conversion.
- void [adcStopConversion](#) (ADCDriver *adcp)
Stops an ongoing conversion.
- void [adcStopConversionI](#) (ADCDriver *adcp)
Stops an ongoing conversion.
- [msg_t](#) [adcConvert](#) (ADCDriver *adcp, const [ADCConversionGroup](#) *grpp, [adcsample_t](#) *samples, [size_t](#) depth)
Performs an ADC conversion.
- void [adcAcquireBus](#) (ADCDriver *adcp)

- Gains exclusive access to the ADC peripheral.*
- void `adcReleaseBus` (`ADCDriver *adcp`)
Releases exclusive access to the ADC peripheral.
- void `adc_lld_init` (void)
Low level ADC driver initialization.
- void `adc_lld_start` (`ADCDriver *adcp`)
Configures and activates the ADC peripheral.
- void `adc_lld_stop` (`ADCDriver *adcp`)
Deactivates the ADC peripheral.
- void `adc_lld_start_conversion` (`ADCDriver *adcp`)
Starts an ADC conversion.
- void `adc_lld_stop_conversion` (`ADCDriver *adcp`)
Stops an ongoing conversion.

Variables

- `ADCDriver ADCD1`
ADC1 driver identifier.

7.2.9.3.4 Macro Definition Documentation

7.2.9.3.4.1 ADC_USE_WAIT

```
#define ADC_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.9.3.4.2 ADC_USE_MUTUAL_EXCLUSION

```
#define ADC_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `adcAcquireBus()` and `adcReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.9.3.4.3 adcIsBufferComplete

```
#define adcIsBufferComplete(  
    adcp)
```

Value:

```
((bool)((adcp)->state == ADC_COMPLETE))
```

Buffer state.

Note

This function is meant to be called from the ADC callback only.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Returns

The buffer state.

Return values

<i>false</i>	if the driver filled/sent the first half of the buffer.
<i>true</i>	if the driver filled/sent the second half of the buffer.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.3.4.4 _adc_reset_i

```
#define _adc_reset_i(  
    adcp)
```

Value:

```
osalThreadResumeI (& (adcp) -> thread, MSG_RESET)
```

Resumes a thread waiting for a conversion completion.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.4.5 _adc_reset_s

```
#define _adc_reset_s(  
    adcp)
```

Value:

```
osalThreadResumeS (& (adcp) -> thread, MSG_RESET)
```

Resumes a thread waiting for a conversion completion.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.4.6 `_adc_wakeup_isr`

```
#define _adc_wakeup_isr(  
    adcp)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI (& (adcp)->thread, MSG_OK);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.4.7 `_adc_timeout_isr`

```
#define _adc_timeout_isr(  
    adcp)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI (& (adcp)->thread, MSG_TIMEOUT);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread with a timeout message.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.4.8 `_adc_isr_half_code`

```
#define _adc_isr_half_code(  
    adcp)
```

Value:

```
{  
    if ((adcp)->grpp->end_cb != NULL) {  
        (adcp)->grpp->end_cb(adcp);  
    }  
}
```

Common ISR code, half buffer event.

This code handles the portable part of the ISR code:

- Callback invocation.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.4.9 `_adc_isr_full_code`

```
#define _adc_isr_full_code(
    adcp)
```

Value:

```
{
    if ((adcp)->grpp->circular) {
        /* Callback handling.*/
        if ((adcp)->grpp->end_cb != NULL) {
            (adcp)->state = ADC_COMPLETE;
            (adcp)->grpp->end_cb(adcp);
            if ((adcp)->state == ADC_COMPLETE) {
                (adcp)->state = ADC_ACTIVE;
            }
        }
    }
    else {
        /* End conversion.*/
        adc_llc_stop_conversion(adcp);
        if ((adcp)->grpp->end_cb != NULL) {
            (adcp)->state = ADC_COMPLETE;
            (adcp)->grpp->end_cb(adcp);
            if ((adcp)->state == ADC_COMPLETE) {
                (adcp)->state = ADC_READY;
                (adcp)->grpp = NULL;
            }
        }
        else {
            (adcp)->state = ADC_READY;
            (adcp)->grpp = NULL;
        }
        _adc_wakeup_isr(adcp);
    }
}
```

Common ISR code, full buffer event.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.4.10 `_adc_isr_error_code`

```
#define _adc_isr_error_code(
    adcp,
    err)
```

Value:

```
{
    adc_llc_stop_conversion(adcp); \
    if ((adcp)->grpp->error_cb != NULL) { \
        (adcp)->state = ADC_ERROR; \
        (adcp)->grpp->error_cb(adcp, err); \
        if ((adcp)->state == ADC_ERROR) { \
            (adcp)->state = ADC_READY; \
            (adcp)->grpp = NULL; \
        } \
    } \
    else { \
        (adcp)->state = ADC_READY; \
        (adcp)->grpp = NULL; \
    } \
    _adc_timeout_isr(adcp); \
}
```

Common ISR code, error event.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread timeout signaling, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object
in	<i>err</i>	platform dependent error code

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.4.11 `ADC_ERR_DMAFAILURE`

```
#define ADC_ERR_DMAFAILURE 1U
```

DMA operations failure.

7.2.9.3.4.12 `ADC_ERR_OVERFLOW`

```
#define ADC_ERR_OVERFLOW 2U
```

ADC overflow condition.

7.2.9.3.4.13 ADC_ERR_AWD

```
#define ADC_ERR_AWD 4U
```

Watchdog triggered.

7.2.9.3.4.14 PLATFORM_ADC_USE_ADC1

```
#define PLATFORM_ADC_USE_ADC1 FALSE
```

ADC1 driver enable switch.

If set to TRUE the support for ADC1 is included.

Note

The default is FALSE.

7.2.9.3.4.15 adc_lld_driver_fields

```
#define adc_lld_driver_fields
```

Value:

```
/* Dummy field, it is not needed.*/  
uint32_t dummy;
```

Low level fields of the ADC driver structure.

7.2.9.3.4.16 adc_lld_config_fields

```
#define adc_lld_config_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/  
uint32_t dummy;
```

Low level fields of the ADC configuration structure.

7.2.9.3.4.17 adc_lld_configuration_group_fields

```
#define adc_lld_configuration_group_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/  
uint32_t dummy;
```

Low level fields of the ADC configuration structure.

7.2.9.3.5 Typedef Documentation

7.2.9.3.5.1 ADCDriver

```
typedef struct hal_adc_driver ADCDriver
```

Type of a structure representing an ADC driver.

7.2.9.3.5.2 ADCConfig

```
typedef struct hal_adc_config ADCConfig
```

Type of a structure representing an ADC driver configuration.

7.2.9.3.5.3 ADCConversionGroup

```
typedef struct hal_adc_configuration_group ADCConversionGroup
```

Conversion group configuration structure.

This implementation-dependent structure describes a conversion operation.

Note

The use of this configuration structure requires knowledge of STM32 ADC cell registers interface, please refer to the STM32 reference manual for details.

7.2.9.3.5.4 adccallback_t

```
typedef void(* adccallback_t) (ADCDriver *adcp)
```

Type of an ADC notification callback.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object triggering the callback

7.2.9.3.5.5 adcerrorcallback_t

```
typedef void(* adcerrorcallback_t) (ADCDriver *adcp, adcerror_t err)
```

Type of an ADC error callback.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object triggering the callback
in	<i>err</i>	ADC error code

7.2.9.3.5.6 adcsample_t

```
typedef uint16_t adcsample_t
```

ADC sample data type.

7.2.9.3.5.7 adc_channels_num_t

```
typedef uint16_t adc_channels_num_t
```

Channels number in a conversion group.

7.2.9.3.5.8 adcerror_t

```
typedef uint32_t adcerror_t
```

Type of an ADC error mask.

7.2.9.3.6 Enumeration Type Documentation

7.2.9.3.6.1 adcstate_t

```
enum adcstate_t
```

Driver state machine possible states.

Enumerator

ADC_UNINIT	Not initialized.
ADC_STOP	Stopped.
ADC_READY	Ready.
ADC_ACTIVE	Converting.
ADC_COMPLETE	Conversion complete.
ADC_ERROR	Conversion error.

7.2.9.3.7 Function Documentation

7.2.9.3.7.1 adcInit()

```
void adcInit (  
    void )
```

ADC Driver initialization.

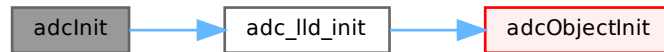
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.3.7.2 adcObjectInit()**

```
void adcObjectInit (
    ADCDriver * adcp)
```

Initializes the standard part of a `ADCDriver` structure.

Parameters

out	<i>adcp</i>	pointer to the <code>ADCDriver</code> object

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.3.7.3 adcStart()**

```
msg_t adcStart (
    ADCDriver * adcp,
    const ADCConfig * config)
```

Configures and activates the ADC peripheral.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object
in	<i>config</i>	pointer to the ADCConfig object. Depending on the implementation the value can be <code>NULL</code> .

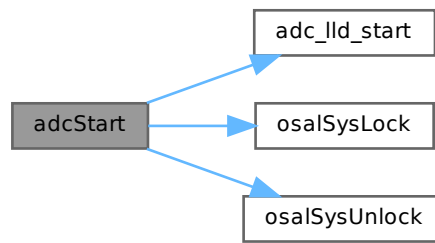
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.3.7.4 adcStop()**

```
void adcStop (  
    ADCDriver * adcp)
```

Deactivates the ADC peripheral.

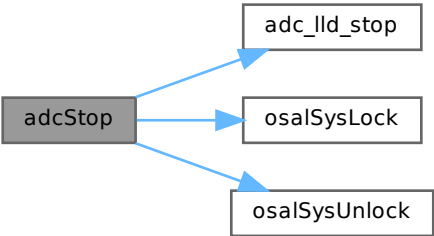
Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.3.7.5 **adcStartConversion()**

```
void adcStartConversion (
    ADCDriver * adcp,
    const ADCCConversionGroup * grpp,
    adcsample_t * samples,
    size_t depth)
```

Starts an ADC conversion.

Starts an asynchronous conversion operation.

Note

The buffer is organized as a matrix of M*N elements where M is the channels number configured into the conversion group and N is the buffer depth. The samples are sequentially written into the buffer with no gaps.

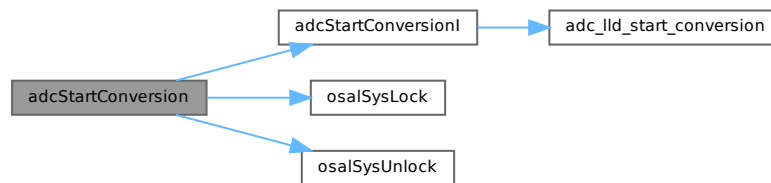
Parameters

in	<i>adcp</i>	pointer to the ADCDriver object
in	<i>grpp</i>	pointer to a ADCCConversionGroup object
out	<i>samples</i>	pointer to the samples buffer
in	<i>depth</i>	buffer depth (matrix rows number). The buffer depth must be one or an even number.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.3.7.6 adcStartConversionI()**

```

void adcStartConversionI (
    ADCDriver * adcp,
    const ADCCConversionGroup * grpp,
    adcsample_t * samples,
    size_t depth)
  
```

Starts an ADC conversion.

Starts an asynchronous conversion operation.

Postcondition

The callbacks associated to the conversion group will be invoked on buffer fill and error events.

Note

The buffer is organized as a matrix of M*N elements where M is the channels number configured into the conversion group and N is the buffer depth. The samples are sequentially written into the buffer with no gaps.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object
in	<i>grpp</i>	pointer to a ADCCConversionGroup object
out	<i>samples</i>	pointer to the samples buffer
in	<i>depth</i>	buffer depth (matrix rows number). The buffer depth must be one or an even number.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.3.7.7 adcStopConversion()**

```
void adcStopConversion (  
    ADCDriver * adcp)
```

Stops an ongoing conversion.

This function stops the currently ongoing conversion and returns the driver in the `ADC_READY` state. If there was no conversion being processed then the function does nothing.

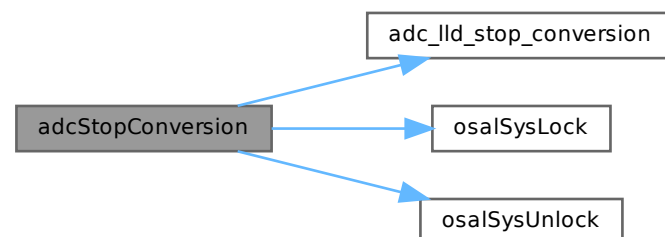
Parameters

in	<i>adcp</i>	pointer to the <code>ADCDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.3.7.8 adcStopConversionI()

```
void adcStopConversionI (
    ADCDriver * adcp)
```

Stops an ongoing conversion.

This function stops the currently ongoing conversion and returns the driver in the `ADC_READY` state. If there was no conversion being processed then the function does nothing.

Parameters

in	<i>adcp</i>	pointer to the <code>ADCDriver</code> object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.3.7.9 adcConvert()

```
msg_t adcConvert (
    ADCDriver * adcp,
    const ADCConversionGroup * grpp,
    adcsample_t * samples,
    size_t depth)
```

Performs an ADC conversion.

Performs a synchronous conversion operation.

Note

The buffer is organized as a matrix of M*N elements where M is the channels number configured into the conversion group and N is the buffer depth. The samples are sequentially written into the buffer with no gaps.

Parameters

in	<i>adcp</i>	pointer to the <code>ADCDriver</code> object
in	<i>grpp</i>	pointer to a <code>ADCConversionGroup</code> object
out	<i>samples</i>	pointer to the samples buffer
in	<i>depth</i>	buffer depth (matrix rows number). The buffer depth must be one or an even number.

Returns

The operation result.

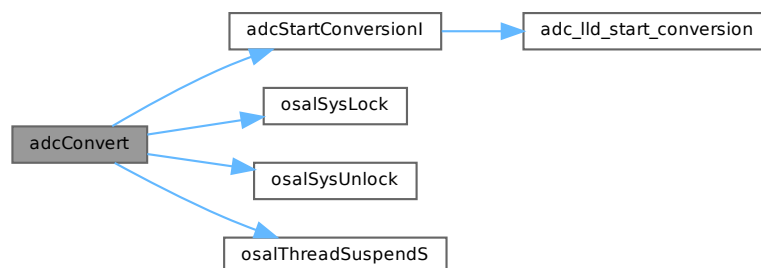
Return values

<i>MSG_OK</i>	Conversion finished.
<i>MSG_RESET</i>	The conversion has been stopped using <code>acdStopConversion()</code> or <code>acdStopConversionI()</code> , the result buffer may contain incorrect data.
<i>MSG_TIMEOUT</i>	The conversion has been stopped because an hardware error.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

7.2.9.3.7.10 `adcAcquireBus()`

```
void adcAcquireBus (
    ADCDriver * adcp)
```

Gains exclusive access to the ADC peripheral.

This function tries to gain ownership to the ADC bus, if the bus is already being used then the invoking thread is queued.

Precondition

In order to use this function the option `ADC_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>adcp</i>	pointer to the <code>ADCDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.3.7.11 adcReleaseBus()**

```
void adcReleaseBus (  
    ADCDriver * adcp)
```

Releases exclusive access to the ADC peripheral.

Precondition

In order to use this function the option `ADC_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>adcp</i>	pointer to the <code>ADCDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.3.7.12 adc_lld_init()

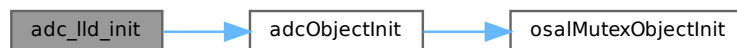
```
void adc_lld_init (
    void )
```

Low level ADC driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.3.7.13 adc_lld_start()**

```
void adc_lld_start (
    ADCDriver * adcp)
```

Configures and activates the ADC peripheral.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.7.14 adc_lld_stop()

```
void adc_lld_stop (
    ADCDriver * adcp)
```

Deactivates the ADC peripheral.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.7.15 adc_lld_start_conversion()

```
void adc_lld_start_conversion (
    ADCDriver * adcp)
```

Starts an ADC conversion.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.7.16 adc_llt_stop_conversion()

```
void adc_llt_stop_conversion (  
    ADCDriver * adcp)
```

Stops an ongoing conversion.

Parameters

in	<i>adcp</i>	pointer to the ADCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.3.8 Variable Documentation**7.2.9.3.8.1 ADCD1**

[ADCDriver](#) ADCD1

ADC1 driver identifier.

7.2.9.4 CAN Driver

Generic CAN Driver.

7.2.9.4.1 Detailed Description

Generic CAN Driver.

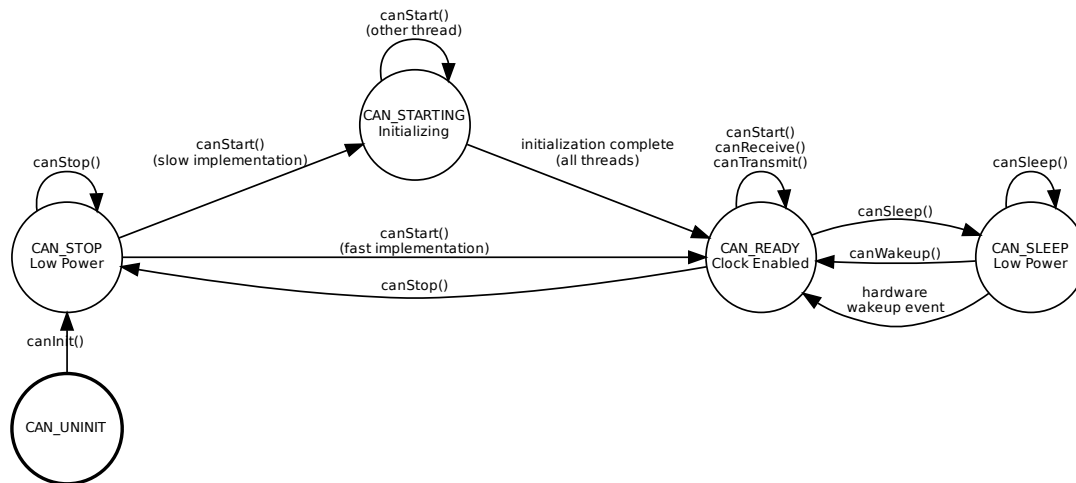
This module implements a generic CAN (Controller Area Network) driver allowing the exchange of information at frame level.

Precondition

In order to use the CAN driver the `HAL_USE_CAN` option must be enabled in [halconf.h](#).

7.2.9.4.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



CAN status flags

- `#define CAN_LIMIT_WARNING 1U`
Errors rate warning.
- `#define CAN_LIMIT_ERROR 2U`
Errors rate error.
- `#define CAN_BUS_OFF_ERROR 4U`
Bus off condition reached.
- `#define CAN_FRAMING_ERROR 8U`
Framing error of some kind on the CAN bus.
- `#define CAN_OVERFLOW_ERROR 16U`
Overflow in receive queue.

CAN configuration options

- `#define CAN_USE_SLEEP_MODE TRUE`
Sleep mode related APIs inclusion switch.
- `#define CAN_ENFORCE_USE_CALLBACKS FALSE`
Enforces the driver to use direct callbacks rather than OSAL events.

Macro Functions

- `#define CAN_MAILBOX_TO_MASK(mbx)`
Converts a mailbox index to a bit mask.
- `#define canTransmit(canp, mailbox, ctfp, timeout)`
Legacy name for `canTransmitTimeout()`.
- `#define canReceive(canp, mailbox, crfp, timeout)`
Legacy name for `canReceiveTimeout()`.

Low level driver helper macros

- `#define _can_tx_empty_isr(canp, flags)`
TX mailbox empty event.
- `#define _can_rx_full_isr(canp, flags)`
RX mailbox empty full event.
- `#define _can_wakeup_isr(canp)`
Wakeup event.
- `#define _can_error_isr(canp, flags)`
Error event.

PLATFORM configuration options

- `#define PLATFORM_CAN_USE_CAN1 FALSE`
CAN1 driver enable switch.

Data Structures

- struct `CANTxFrame`
CAN transmission frame.
- struct `CANRxFrame`
CAN received frame.
- struct `hal_can_config`
Type of a CAN configuration structure.
- struct `hal_can_driver`
Structure representing an CAN driver.

Macros

- `#define CAN_ANY_MAILBOX 0U`
Special mailbox identifier.
- `#define CAN_TX_MAILBOXES 1`
Number of transmit mailboxes.
- `#define CAN_RX_MAILBOXES 1`
Number of receive mailboxes.

Typedefs

- typedef struct `hal_can_driver` `CANDriver`
Type of a structure representing an CAN driver.
- typedef uint32_t `canmbx_t`
Type of a transmission mailbox index.
- typedef void(* `can_callback_t`) (`CANDriver` *canp, uint32_t flags)
Type of a CAN notification callback.
- typedef struct `hal_can_config` `CANConfig`
Type of a CAN configuration structure.

Enumerations

- enum `canstate_t` {
`CAN_UNINIT` = 0 , `CAN_STOP` = 1 , `CAN_STARTING` = 2 , `CAN_STOPPING` = 3 ,
`CAN_READY` = 4 , `CAN_SLEEP` = 5 }

Driver state machine possible states.

Functions

- void `canInit` (void)
CAN Driver initialization.
- void `canObjectInit` (`CANDriver` *canp)
Initializes the standard part of a `CANDriver` structure.
- `msg_t` `canStart` (`CANDriver` *canp, const `CANConfig` *config)
Configures and activates the CAN peripheral.
- void `canStop` (`CANDriver` *canp)
Deactivates the CAN peripheral.
- bool `canTryTransmitI` (`CANDriver` *canp, `canmbx_t` mailbox, const `CANTxFrame` *ctfp)
Can frame transmission attempt.
- bool `canTryReceiveI` (`CANDriver` *canp, `canmbx_t` mailbox, `CANRxFrame` *crfp)
Can frame receive attempt.
- void `canTryAbortX` (`CANDriver` *canp, `canmbx_t` mailbox)
Tries to abort an ongoing transmission.
- `msg_t` `canTransmitTimeout` (`CANDriver` *canp, `canmbx_t` mailbox, const `CANTxFrame` *ctfp, `sysinterval_t` timeout)
Can frame transmission.
- `msg_t` `canReceiveTimeout` (`CANDriver` *canp, `canmbx_t` mailbox, `CANRxFrame` *crfp, `sysinterval_t` timeout)
Can frame receive.
- void `canSleep` (`CANDriver` *canp)
Enters the sleep mode.
- void `canWakeup` (`CANDriver` *canp)
Enforces leaving the sleep mode.
- void `can_llc_init` (void)
Low level CAN driver initialization.
- void `can_llc_start` (`CANDriver` *canp)
Configures and activates the CAN peripheral.
- void `can_llc_stop` (`CANDriver` *canp)
Deactivates the CAN peripheral.
- bool `can_llc_is_tx_empty` (`CANDriver` *canp, `canmbx_t` mailbox)
Determines whether a frame can be transmitted.
- void `can_llc_transmit` (`CANDriver` *canp, `canmbx_t` mailbox, const `CANTxFrame` *ctfp)
Inserts a frame into the transmit queue.
- bool `can_llc_is_rx_nonempty` (`CANDriver` *canp, `canmbx_t` mailbox)
Determines whether a frame has been received.
- void `can_llc_receive` (`CANDriver` *canp, `canmbx_t` mailbox, `CANRxFrame` *crfp)
Receives a frame from the input queue.
- void `can_llc_abort` (`CANDriver` *canp, `canmbx_t` mailbox)
Tries to abort an ongoing transmission.
- void `can_llc_sleep` (`CANDriver` *canp)
Enters the sleep mode.
- void `can_llc_wakeup` (`CANDriver` *canp)
Enforces leaving the sleep mode.

Variables

- [CANDriver CAND1](#)

CAN1 driver identifier.

7.2.9.4.3 Macro Definition Documentation

7.2.9.4.3.1 CAN_LIMIT_WARNING

```
#define CAN_LIMIT_WARNING 1U
```

Errors rate warning.

7.2.9.4.3.2 CAN_LIMIT_ERROR

```
#define CAN_LIMIT_ERROR 2U
```

Errors rate error.

7.2.9.4.3.3 CAN_BUS_OFF_ERROR

```
#define CAN_BUS_OFF_ERROR 4U
```

Bus off condition reached.

7.2.9.4.3.4 CAN_FRAMING_ERROR

```
#define CAN_FRAMING_ERROR 8U
```

Framing error of some kind on the CAN bus.

7.2.9.4.3.5 CAN_OVERFLOW_ERROR

```
#define CAN_OVERFLOW_ERROR 16U
```

Overflow in receive queue.

7.2.9.4.3.6 CAN_ANY_MAILBOX

```
#define CAN_ANY_MAILBOX 0U
```

Special mailbox identifier.

7.2.9.4.3.7 CAN_USE_SLEEP_MODE

```
#define CAN_USE_SLEEP_MODE TRUE
```

Sleep mode related APIs inclusion switch.

This option can only be enabled if the CAN implementation supports the sleep mode, see the macro `CAN_↔SUPPORTS_SLEEP` exported by the underlying implementation.

7.2.9.4.3.8 CAN_ENFORCE_USE_CALLBACKS

```
#define CAN_ENFORCE_USE_CALLBACKS FALSE
```

Enforces the driver to use direct callbacks rather than OSAL events.

7.2.9.4.3.9 CAN_MAILBOX_TO_MASK

```
#define CAN_MAILBOX_TO_MASK(  
    mbx)
```

Value:

```
(1U « ((mbx) - 1U))
```

Converts a mailbox index to a bit mask.

7.2.9.4.3.10 canTransmit

```
#define canTransmit(  
    canp,  
    mailbox,  
    ctfp,  
    timeout)
```

Value:

```
canTransmitTimeout(canp, mailbox, ctfp, timeout)
```

Legacy name for `canTransmitTimeout()`.

Deprecated

7.2.9.4.3.11 canReceive

```
#define canReceive(  
    canp,  
    mailbox,  
    crfp,  
    timeout)
```

Value:

```
canReceiveTimeout(canp, mailbox, crfp, timeout)
```

Legacy name for `canReceiveTimeout()`.

Deprecated

7.2.9.4.3.12 _can_tx_empty_isr

```
#define _can_tx_empty_isr(
    canp,
    flags)
```

Value:

```
{
    osalSysLockFromISR();
    osalThreadDequeueAllI (&(canp)->txqueue, MSG_OK);
    osalEventBroadcastFlagsI (&(canp)->txempty_event, flags);
    osalSysUnlockFromISR();
}
```

TX mailbox empty event.

7.2.9.4.3.13 _can_rx_full_isr

```
#define _can_rx_full_isr(
    canp,
    flags)
```

Value:

```
{
    osalSysLockFromISR();
    osalThreadDequeueAllI (&(canp)->rxqueue, MSG_OK);
    osalEventBroadcastFlagsI (&(canp)->rxfull_event, flags);
    osalSysUnlockFromISR();
}
```

RX mailbox empty full event.

7.2.9.4.3.14 _can_wakeup_isr

```
#define _can_wakeup_isr(
    canp)
```

Value:

```
{
    osalSysLockFromISR();
    osalEventBroadcastFlagsI (&(canp)->wakeup_event, 0U);
    osalSysUnlockFromISR();
}
```

Wakeup event.

7.2.9.4.3.15 _can_error_isr

```
#define _can_error_isr(
    canp,
    flags)
```

Value:

```
{
    osalSysLockFromISR();
    osalEventBroadcastFlagsI (&(canp)->error_event, flags);
    osalSysUnlockFromISR();
}
```

Error event.

7.2.9.4.3.16 CAN_TX_MAILBOXES

```
#define CAN_TX_MAILBOXES 1
```

Number of transmit mailboxes.

7.2.9.4.3.17 CAN_RX_MAILBOXES

```
#define CAN_RX_MAILBOXES 1
```

Number of receive mailboxes.

7.2.9.4.3.18 PLATFORM_CAN_USE_CAN1

```
#define PLATFORM_CAN_USE_CAN1 FALSE
```

CAN1 driver enable switch.

If set to `TRUE` the support for CAN1 is included.

Note

The default is `FALSE`.

7.2.9.4.4 Typedef Documentation

7.2.9.4.4.1 CANDriver

```
typedef struct hal_can_driver CANDriver
```

Type of a structure representing an CAN driver.

7.2.9.4.4.2 canmbx_t

```
typedef uint32_t canmbx_t
```

Type of a transmission mailbox index.

7.2.9.4.4.3 can_callback_t

```
typedef void(* can_callback_t) (CANDriver *canp, uint32_t flags)
```

Type of a CAN notification callback.

Parameters

in	<i>canp</i>	pointer to the CANDriver object triggering the callback
in	<i>flags</i>	flags associated to the mailbox callback

7.2.9.4.4 CANConfig

```
typedef struct hal_can_config CANConfig
```

Type of a CAN configuration structure.

7.2.9.4.5 Enumeration Type Documentation

7.2.9.4.5.1 canstate_t

```
enum canstate_t
```

Driver state machine possible states.

Enumerator

CAN_UNINIT	Not initialized.
CAN_STOP	Stopped.
CAN_STARTING	Starting.
CAN_STOPPING	Stopping.
CAN_READY	Ready.
CAN_SLEEP	Sleep state.

7.2.9.4.6 Function Documentation

7.2.9.4.6.1 canInit()

```
void canInit (
    void )
```

CAN Driver initialization.

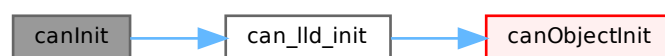
Note

This function is implicitly invoked by [halInit\(\)](#), there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.4.6.2 canObjectInit()

```
void canObjectInit (
    CANDriver * canp)
```

Initializes the standard part of a `CANDriver` structure.

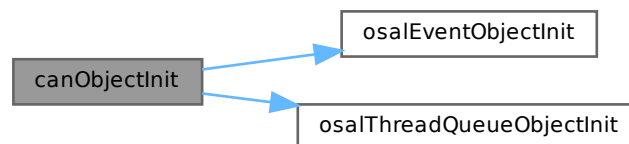
Parameters

out	<i>canp</i>	pointer to the <code>CANDriver</code> object

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.4.6.3 canStart()

```
msg_t canStart (
    CANDriver * canp,
    const CANConfig * config)
```

Configures and activates the CAN peripheral.

Note

Activating the CAN bus can be a slow operation.

Unlike other drivers it is not possible to restart the CAN driver without first stopping it using `canStop()`.

Parameters

in	<i>canp</i>	pointer to the <code>CANDriver</code> object
in	<i>config</i>	pointer to the <code>CANConfig</code> object. Depending on the implementation the value can be <code>NULL</code> .

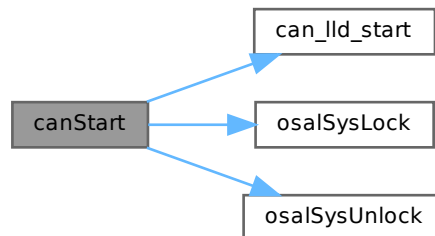
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.4.6.4 canStop()**

```
void canStop (
    CANDriver * canp)
```

Deactivates the CAN peripheral.

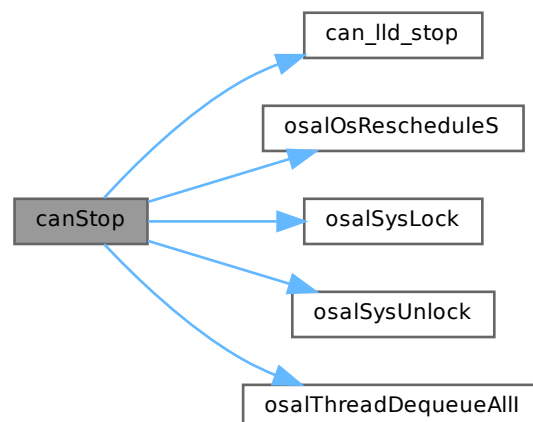
Parameters

in	<i>canp</i>	pointer to the <code>CANDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.4.6.5 canTryTransmitI()

```
bool canTryTransmitI (
    CANDriver * canp,
    canmbx_t mailbox,
    const CANTxFrame * ctfp)
```

Can frame transmission attempt.

The specified frame is queued for transmission, if the hardware queue is full then the function fails.

Parameters

in	<i>canp</i>	pointer to the <code>CANDriver</code> object
in	<i>mailbox</i>	mailbox number, <code>CAN_ANY_MAILBOX</code> for any mailbox
in	<i>ctfp</i>	pointer to the CAN frame to be transmitted

Returns

The operation result.

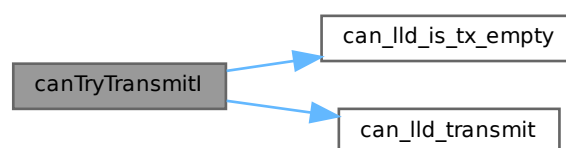
Return values

<i>false</i>	Frame transmitted.
<i>true</i>	Mailbox full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.4.6.6 canTryReceiveI()

```
bool canTryReceiveI (
    CANDriver * canp,
    canmbx_t mailbox,
    CANRxFrame * crfp)
```

Can frame receive attempt.

The function tries to fetch a frame from a mailbox.

Parameters

in	<i>canp</i>	pointer to the CANDriver object
in	<i>mailbox</i>	mailbox number, CAN_ANY_MAILBOX for any mailbox
out	<i>crfp</i>	pointer to the buffer where the CAN frame is copied

Returns

The operation result.

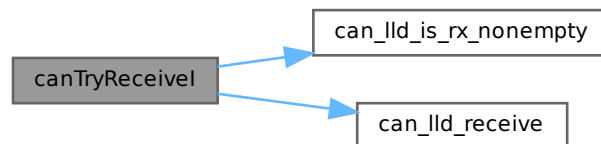
Return values

<i>false</i>	Frame fetched.
<i>true</i>	Mailbox empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.4.6.7 canTryAbortX()

```
void canTryAbortX (  
    CANDriver * canp,  
    canmbx\_t mailbox)
```

Tries to abort an ongoing transmission.

Parameters

in	<i>canp</i>	pointer to the CANDriver object
in	<i>mailbox</i>	mailbox number

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

Here is the call graph for this function:

**7.2.9.4.6.8 canTransmitTimeout()**

```

msg_t canTransmitTimeout (
    CANDriver * canp,
    canmbx_t mailbox,
    const CANTxFrame * ctfp,
    sysinterval_t timeout)
  
```

Can frame transmission.

The specified frame is queued for transmission, if the hardware queue is full then the invoking thread is queued.

Note

Trying to transmit while in sleep mode simply enqueues the thread.

Parameters

in	<i>canp</i>	pointer to the <code>CANDriver</code> object
in	<i>mailbox</i>	mailbox number, <code>CAN_ANY_MAILBOX</code> for any mailbox
in	<i>ctfp</i>	pointer to the CAN frame to be transmitted
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <code>TIME_IMMEDIATE</code> immediate timeout. • <code>TIME_INFINITE</code> no timeout.

Returns

The operation result.

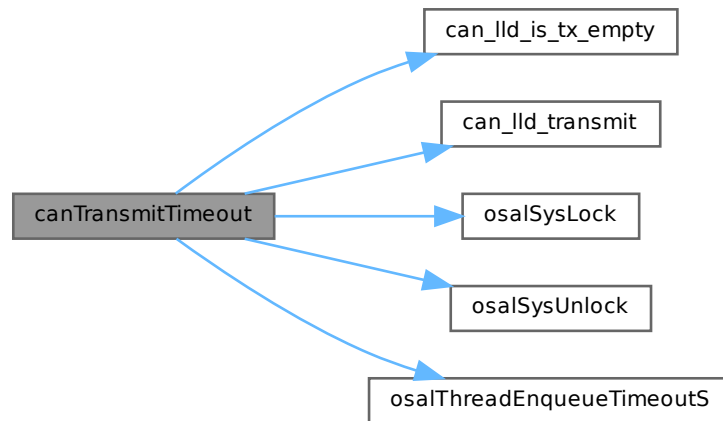
Return values

<code>MSG_OK</code>	the frame has been queued for transmission.
<code>MSG_TIMEOUT</code>	The operation has timed out.
<code>MSG_RESET</code>	The driver has been stopped while waiting.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.4.6.9 canReceiveTimeout()**

```

msg_t canReceiveTimeout (
    CANDriver * canp,
    canmbx_t mailbox,
    CANRxFrame * crfp,
    sysinterval_t timeout)
  
```

Can frame receive.

The function waits until a frame is received.

Note

Trying to receive while in sleep mode simply enqueues the thread.

Parameters

in	<i>canp</i>	pointer to the <code>CANDriver</code> object
in	<i>mailbox</i>	mailbox number, <code>CAN_ANY_MAILBOX</code> for any mailbox
out	<i>crfp</i>	pointer to the buffer where the CAN frame is copied
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <code>TIME_IMMEDIATE</code> immediate timeout (useful in an event driven scenario where a thread never blocks for I/O). • <code>TIME_INFINITE</code> no timeout.

Returns

The operation result.

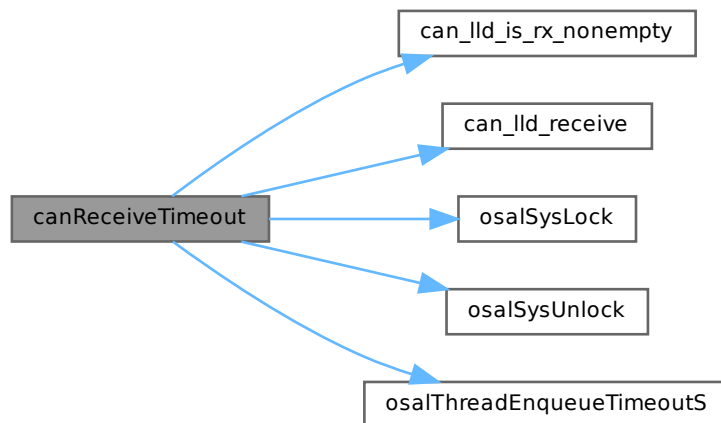
Return values

<i>MSG_OK</i>	a frame has been received and placed in the buffer.
<i>MSG_TIMEOUT</i>	The operation has timed out.
<i>MSG_RESET</i>	The driver has been stopped while waiting.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.4.6.10 canSleep()

```
void canSleep (
    CANDriver * canp)
```

Enters the sleep mode.

This function puts the CAN driver in sleep mode and broadcasts the `sleep_event` event source.

Precondition

In order to use this function the option `CAN_USE_SLEEP_MODE` must be enabled and the `CAN_↔SUPPORTS_SLEEP` mode must be supported by the low level driver.

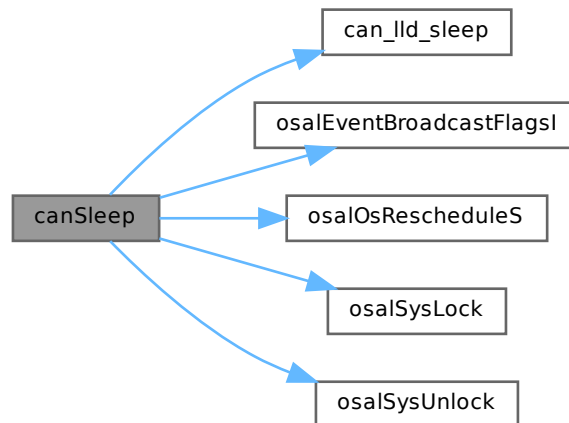
Parameters

in	<i>canp</i>	pointer to the <code>CANDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.4.6.11 canWakeup()**

```
void canWakeup (
    CANDriver * canp)
```

Enforces leaving the sleep mode.

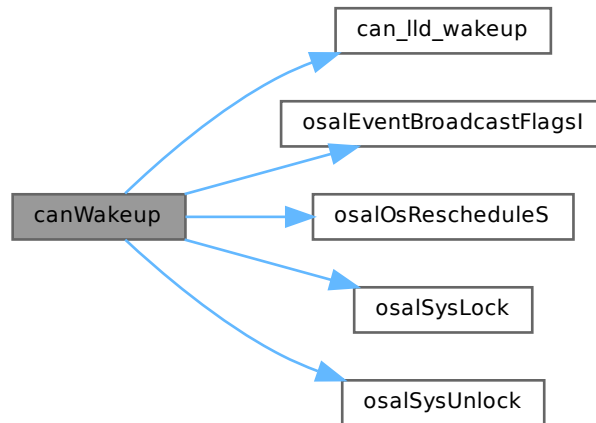
Note

The sleep mode is supposed to be usually exited automatically by an hardware event.

Parameters

in	<i>canp</i>	pointer to the <code>CANDriver</code> object

Here is the call graph for this function:



7.2.9.4.6.12 `can_lld_init()`

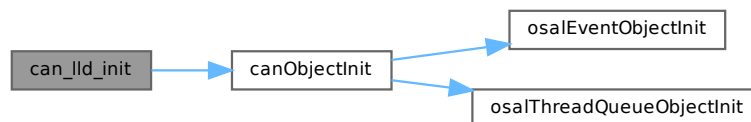
```
void can_lld_init (
    void )
```

Low level CAN driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.4.6.13 `can_lld_start()`

```
void can_lld_start (
    CANDriver * canp)
```

Configures and activates the CAN peripheral.

Parameters

in	<i>canp</i>	pointer to the CANDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.4.6.14 can_lld_stop()

```
void can_lld_stop (
    CANDriver * canp)
```

Deactivates the CAN peripheral.

Parameters

in	<i>canp</i>	pointer to the CANDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.4.6.15 can_lld_is_tx_empty()

```
bool can_lld_is_tx_empty (
    CANDriver * canp,
    canmbx_t mailbox)
```

Determines whether a frame can be transmitted.

Parameters

in	<i>canp</i>	pointer to the CANDriver object
in	<i>mailbox</i>	mailbox number, CAN_ANY_MAILBOX for any mailbox

Returns

The queue space availability.

Return values

<i>false</i>	no space in the transmit queue.
<i>true</i>	transmit slot available.

Function Class:

Not an API, this function is for internal use only.

7.2.9.4.6.16 can_lld_transmit()

```
void can_lld_transmit (
    CANDriver * canp,
    canmbx_t mailbox,
    const CANTxFrame * ctfp)
```

Inserts a frame into the transmit queue.

Parameters

in	<i>canp</i>	pointer to the CANDriver object
in	<i>ctfp</i>	pointer to the CAN frame to be transmitted
in	<i>mailbox</i>	mailbox number, CAN_ANY_MAILBOX for any mailbox

Function Class:

Not an API, this function is for internal use only.

7.2.9.4.6.17 can_lld_is_rx_nonempty()

```
bool can_lld_is_rx_nonempty (
    CANDriver * canp,
    canmbx_t mailbox)
```

Determines whether a frame has been received.

Parameters

in	<i>canp</i>	pointer to the CANDriver object
in	<i>mailbox</i>	mailbox number, CAN_ANY_MAILBOX for any mailbox

Returns

The queue space availability.

Return values

<i>false</i>	no space in the transmit queue.
<i>true</i>	transmit slot available.

Function Class:

Not an API, this function is for internal use only.

7.2.9.4.6.18 can_lld_receive()

```
void can_lld_receive (
    CANDriver * canp,
    canmbx_t mailbox,
    CANRxFrame * crfp)
```

Receives a frame from the input queue.

Parameters

in	<i>canp</i>	pointer to the CANDriver object
in	<i>mailbox</i>	mailbox number, CAN_ANY_MAILBOX for any mailbox
out	<i>crfp</i>	pointer to the buffer where the CAN frame is copied

Function Class:

Not an API, this function is for internal use only.

7.2.9.4.6.19 can_llc_abort()

```
void can_llc_abort (  
    CANDriver * canp,  
    canmbx_t mailbox)
```

Tries to abort an ongoing transmission.

Parameters

in	<i>canp</i>	pointer to the CANDriver object
in	<i>mailbox</i>	mailbox number

Function Class:

Not an API, this function is for internal use only.

7.2.9.4.6.20 can_llc_sleep()

```
void can_llc_sleep (  
    CANDriver * canp)
```

Enters the sleep mode.

Parameters

in	<i>canp</i>	pointer to the CANDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.4.6.21 can_llc_wakeup()

```
void can_llc_wakeup (  
    CANDriver * canp)
```

Enforces leaving the sleep mode.

Parameters

in	<i>canp</i>	pointer to the CANDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.4.7 Variable Documentation

7.2.9.4.7.1 CAND1

`CANDriver` CAND1

CAN1 driver identifier.

7.2.9.5 Crypto Driver

Generic Crypto Driver.

7.2.9.5.1 Detailed Description

Generic Crypto Driver.

This module implements a generic Cryptography driver.

Precondition

In order to use the crypto driver the `HAL_USE_CRY` option must be enabled in `halconf.h`.

Driver capability switches

- `#define CRY_LLD_SUPPORTS_AES TRUE`
- `#define CRY_LLD_SUPPORTS_AES_ECB TRUE`
- `#define CRY_LLD_SUPPORTS_AES_CBC TRUE`
- `#define CRY_LLD_SUPPORTS_AES_CFB TRUE`
- `#define CRY_LLD_SUPPORTS_AES_CTR TRUE`
- `#define CRY_LLD_SUPPORTS_AES_GCM TRUE`
- `#define CRY_LLD_SUPPORTS_DES TRUE`
- `#define CRY_LLD_SUPPORTS_DES_ECB TRUE`
- `#define CRY_LLD_SUPPORTS_DES_CBC TRUE`
- `#define CRY_LLD_SUPPORTS_SHA1 TRUE`
- `#define CRY_LLD_SUPPORTS_SHA256 TRUE`
- `#define CRY_LLD_SUPPORTS_SHA512 TRUE`
- `#define CRY_LLD_SUPPORTS_HMAC_SHA256 TRUE`
- `#define CRY_LLD_SUPPORTS_HMAC_SHA512 TRUE`

PLATFORM configuration options

- `#define PLATFORM_CRY_USE_CRY1 FALSE`
CRY1 driver enable switch.

Data Structures

- struct [SHA1Context](#)
Type of a SHA1 context.
- struct [SHA256Context](#)
Type of a SHA256 context.
- struct [SHA512Context](#)
Type of a SHA512 context.
- struct [HMACSHA256Context](#)
Type of a HMAC_SHA256 context.
- struct [HMACSHA512Context](#)
Type of a HMAC_SHA512 context.
- struct [CRYConfig](#)
Driver configuration structure.
- struct [CRYDriver](#)
Structure representing an CRY driver.

Macros

- #define [HAL_CRY_USE_FALLBACK FALSE](#)
Enables the SW fall-back of the cryptographic driver.
- #define [HAL_CRY_USE_FALLBACK TRUE](#)
Enables the SW fall-back of the cryptographic driver.
- #define [HAL_CRY_ENFORCE_FALLBACK FALSE](#)
Makes the driver forcibly use the fall-back implementations.

Typedefs

- typedef size_t [bitsize_t](#)
Size, in bits, of a crypto field or message.
- typedef uint32_t [crykey_t](#)
CRY key identifier type.
- typedef struct CRYDriver [CRYDriver](#)
Type of a structure representing an CRY driver.

Enumerations

- enum [crystate_t](#) { [CRY_UNINIT](#) = 0 , [CRY_STOP](#) = 1 , [CRY_READY](#) = 2 }
Driver state machine possible states.
- enum [cryerror_t](#) {
 [CRY_NOERROR](#) = 0 , [CRY_ERR_INV_ALGO](#) = 1 , [CRY_ERR_INV_KEY_SIZE](#) = 2 , [CRY_ERR_INV_KEY_TYPE](#) = 3 ,
 [CRY_ERR_INV_KEY_ID](#) = 4 , [CRY_ERR_AUTH_FAILED](#) = 5 , [CRY_ERR_OP_FAILURE](#) = 6 }
Driver error codes.
- enum [cryalgorithm_t](#) { [cry_algo_none](#) = 0 , [cry_algo_aes](#) , [cry_algo_des](#) , [cry_algo_hmac](#) }
Type of an algorithm identifier.

Functions

- void [cryInit](#) (void)
Cryptographic Driver initialization.
- void [cryObjectInit](#) (CRYDriver *cryp)
Initializes the standard part of a CRYDriver structure.
- [msg_t cryStart](#) (CRYDriver *cryp, const CRYConfig *config)
Configures and activates the cryptographic peripheral.
- void [cryStop](#) (CRYDriver *cryp)
Deactivates the cryptographic peripheral.
- [cryerror_t cryLoadAESTransientKey](#) (CRYDriver *cryp, size_t size, const uint8_t *keyp)
Initializes the AES transient key.
- [cryerror_t cryEncryptAES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Encryption of a single block using AES.
- [cryerror_t cryDecryptAES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Decryption of a single block using AES.
- [cryerror_t cryEncryptAES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Encryption operation using AES-ECB.
- [cryerror_t cryDecryptAES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Decryption operation using AES-ECB.
- [cryerror_t cryEncryptAES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CBC.
- [cryerror_t cryDecryptAES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CBC.
- [cryerror_t cryEncryptAES_CFB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CFB.
- [cryerror_t cryDecryptAES_CFB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CFB.
- [cryerror_t cryEncryptAES_CTR](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CTR.
- [cryerror_t cryDecryptAES_CTR](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CTR.
- [cryerror_t cryEncryptAES_GCM](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, uint8_t *tag_out)
Encryption operation using AES-GCM.
- [cryerror_t cryDecryptAES_GCM](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, const uint8_t *tag_in)
Decryption operation using AES-GCM.
- [cryerror_t cryLoadDESTransientKey](#) (CRYDriver *cryp, size_t size, const uint8_t *keyp)
Initializes the DES transient key.
- [cryerror_t cryEncryptDES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Encryption of a single block using (T)DES.
- [cryerror_t cryDecryptDES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)

Decryption of a single block using (T)DES.

- `cryerror_t cryEncryptDES_ECB (CRYDriver *cryp, crykey_t key_id, size_t size, const uint8_t *in, uint8_t *out)`

Encryption operation using (T)DES-ECB.

- `cryerror_t cryDecryptDES_ECB (CRYDriver *cryp, crykey_t key_id, size_t size, const uint8_t *in, uint8_t *out)`

Decryption operation using (T)DES-ECB.

- `cryerror_t cryEncryptDES_CBC (CRYDriver *cryp, crykey_t key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)`

Encryption operation using (T)DES-CBC.

- `cryerror_t cryDecryptDES_CBC (CRYDriver *cryp, crykey_t key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)`

Decryption operation using (T)DES-CBC.

- `cryerror_t crySHA1Init (CRYDriver *cryp, SHA1Context *sha1ctxp)`

Hash initialization using SHA1.

- `cryerror_t crySHA1Update (CRYDriver *cryp, SHA1Context *sha1ctxp, size_t size, const uint8_t *in)`

Hash update using SHA1.

- `cryerror_t crySHA1Final (CRYDriver *cryp, SHA1Context *sha1ctxp, uint8_t *out)`

Hash finalization using SHA1.

- `cryerror_t crySHA256Init (CRYDriver *cryp, SHA256Context *sha256ctxp)`

Hash initialization using SHA256.

- `cryerror_t crySHA256Update (CRYDriver *cryp, SHA256Context *sha256ctxp, size_t size, const uint8_t *in)`

Hash update using SHA256.

- `cryerror_t crySHA256Final (CRYDriver *cryp, SHA256Context *sha256ctxp, uint8_t *out)`

Hash finalization using SHA256.

- `cryerror_t crySHA512Init (CRYDriver *cryp, SHA512Context *sha512ctxp)`

Hash initialization using SHA512.

- `cryerror_t crySHA512Update (CRYDriver *cryp, SHA512Context *sha512ctxp, size_t size, const uint8_t *in)`

Hash update using SHA512.

- `cryerror_t crySHA512Final (CRYDriver *cryp, SHA512Context *sha512ctxp, uint8_t *out)`

Hash finalization using SHA512.

- `cryerror_t cryLoadHMACTransientKey (CRYDriver *cryp, size_t size, const uint8_t *keyp)`

Initializes the HMAC transient key.

- `cryerror_t cryHMACSHA256Init (CRYDriver *cryp, HMACSHA256Context *hmacsha256ctxp)`

Hash initialization using HMAC_SHA256.

- `cryerror_t cryHMACSHA256Update (CRYDriver *cryp, HMACSHA256Context *hmacsha256ctxp, size_t size, const uint8_t *in)`

Hash update using HMAC.

- `cryerror_t cryHMACSHA256Final (CRYDriver *cryp, HMACSHA256Context *hmacsha256ctxp, uint8_t *out)`

Hash finalization using HMAC.

- `cryerror_t cryHMACSHA512Init (CRYDriver *cryp, HMACSHA512Context *hmacsha512ctxp)`

Hash initialization using HMAC_SHA512.

- `cryerror_t cryHMACSHA512Update (CRYDriver *cryp, HMACSHA512Context *hmacsha512ctxp, size_t size, const uint8_t *in)`

Hash update using HMAC.

- `cryerror_t cryHMACSHA512Final (CRYDriver *cryp, HMACSHA512Context *hmacsha512ctxp, uint8_t *out)`

Hash finalization using HMAC.

- `void cry_lld_init (void)`

Low level crypto driver initialization.

- `void cry_lld_start (CRYDriver *cryp)`

Configures and activates the crypto peripheral.

- void `cry_lld_stop` (`CRYDriver` *cryp)

Deactivates the crypto peripheral.
- `cryerror_t` `cry_lld_aes_loadkey` (`CRYDriver` *cryp, `size_t` size, `const uint8_t` *keyp)

Initializes the AES transient key.
- `cryerror_t` `cry_lld_encrypt_AES` (`CRYDriver` *cryp, `crykey_t` key_id, `const uint8_t` *in, `uint8_t` *out)

Encryption of a single block using AES.
- `cryerror_t` `cry_lld_decrypt_AES` (`CRYDriver` *cryp, `crykey_t` key_id, `const uint8_t` *in, `uint8_t` *out)

Decryption of a single block using AES.
- `cryerror_t` `cry_lld_encrypt_AES_ECB` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out)

Encryption operation using AES-ECB.
- `cryerror_t` `cry_lld_decrypt_AES_ECB` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out)

Decryption operation using AES-ECB.
- `cryerror_t` `cry_lld_encrypt_AES_CBC` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out, `const uint8_t` *iv)

Encryption operation using AES-CBC.
- `cryerror_t` `cry_lld_decrypt_AES_CBC` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out, `const uint8_t` *iv)

Decryption operation using AES-CBC.
- `cryerror_t` `cry_lld_encrypt_AES_CFB` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out, `const uint8_t` *iv)

Encryption operation using AES-CFB.
- `cryerror_t` `cry_lld_decrypt_AES_CFB` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out, `const uint8_t` *iv)

Decryption operation using AES-CFB.
- `cryerror_t` `cry_lld_encrypt_AES_CTR` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out, `const uint8_t` *iv)

Encryption operation using AES-CTR.
- `cryerror_t` `cry_lld_decrypt_AES_CTR` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out, `const uint8_t` *iv)

Decryption operation using AES-CTR.
- `cryerror_t` `cry_lld_encrypt_AES_GCM` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` auth_size, `const uint8_t` *auth_in, `size_t` text_size, `const uint8_t` *text_in, `uint8_t` *text_out, `const uint8_t` *iv, `size_t` tag_size, `uint8_t` *tag_out)

Encryption operation using AES-GCM.
- `cryerror_t` `cry_lld_decrypt_AES_GCM` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` auth_size, `const uint8_t` *auth_in, `size_t` text_size, `const uint8_t` *text_in, `uint8_t` *text_out, `const uint8_t` *iv, `size_t` tag_size, `const uint8_t` *tag_in)

Decryption operation using AES-GCM.
- `cryerror_t` `cry_lld_des_loadkey` (`CRYDriver` *cryp, `size_t` size, `const uint8_t` *keyp)

Initializes the DES transient key.
- `cryerror_t` `cry_lld_encrypt_DES` (`CRYDriver` *cryp, `crykey_t` key_id, `const uint8_t` *in, `uint8_t` *out)

Encryption of a single block using (T)DES.
- `cryerror_t` `cry_lld_decrypt_DES` (`CRYDriver` *cryp, `crykey_t` key_id, `const uint8_t` *in, `uint8_t` *out)

Decryption of a single block using (T)DES.
- `cryerror_t` `cry_lld_encrypt_DES_ECB` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out)

Encryption operation using (T)DES-ECB.
- `cryerror_t` `cry_lld_decrypt_DES_ECB` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, `const uint8_t` *in, `uint8_t` *out)

Decryption operation using (T)DES-ECB.

- `cryerror_t cry_lld_encrypt_DES_CBC` (`CRYDriver *cry`, `crykey_t key_id`, `size_t size`, `const uint8_t *in`, `uint8_t *out`, `const uint8_t *iv`)
Encryption operation using (T)DES-CBC.
- `cryerror_t cry_lld_decrypt_DES_CBC` (`CRYDriver *cry`, `crykey_t key_id`, `size_t size`, `const uint8_t *in`, `uint8_t *out`, `const uint8_t *iv`)
Decryption operation using (T)DES-CBC.
- `cryerror_t cry_lld_SHA1_init` (`CRYDriver *cry`, `SHA1Context *sha1ctxp`)
Hash initialization using SHA1.
- `cryerror_t cry_lld_SHA1_update` (`CRYDriver *cry`, `SHA1Context *sha1ctxp`, `size_t size`, `const uint8_t *in`)
Hash update using SHA1.
- `cryerror_t cry_lld_SHA1_final` (`CRYDriver *cry`, `SHA1Context *sha1ctxp`, `uint8_t *out`)
Hash finalization using SHA1.
- `cryerror_t cry_lld_SHA256_init` (`CRYDriver *cry`, `SHA256Context *sha256ctxp`)
Hash initialization using SHA256.
- `cryerror_t cry_lld_SHA256_update` (`CRYDriver *cry`, `SHA256Context *sha256ctxp`, `size_t size`, `const uint8_t *in`)
Hash update using SHA256.
- `cryerror_t cry_lld_SHA256_final` (`CRYDriver *cry`, `SHA256Context *sha256ctxp`, `uint8_t *out`)
Hash finalization using SHA256.
- `cryerror_t cry_lld_SHA512_init` (`CRYDriver *cry`, `SHA512Context *sha512ctxp`)
Hash initialization using SHA512.
- `cryerror_t cry_lld_SHA512_update` (`CRYDriver *cry`, `SHA512Context *sha512ctxp`, `size_t size`, `const uint8_t *in`)
Hash update using SHA512.
- `cryerror_t cry_lld_SHA512_final` (`CRYDriver *cry`, `SHA512Context *sha512ctxp`, `uint8_t *out`)
Hash finalization using SHA512.
- `cryerror_t cry_lld_hmac_loadkey` (`CRYDriver *cry`, `size_t size`, `const uint8_t *keyp`)
Initializes the HMAC transient key.
- `cryerror_t cry_lld_HMACSHA256_init` (`CRYDriver *cry`, `HMACSHA256Context *hmacsha256ctxp`)
Hash initialization using HMAC_SHA256.
- `cryerror_t cry_lld_HMACSHA256_update` (`CRYDriver *cry`, `HMACSHA256Context *hmacsha256ctxp`, `size_t size`, `const uint8_t *in`)
Hash update using HMAC.
- `cryerror_t cry_lld_HMACSHA256_final` (`CRYDriver *cry`, `HMACSHA256Context *hmacsha256ctxp`, `uint8_t *out`)
Hash finalization using HMAC.
- `cryerror_t cry_lld_HMACSHA512_init` (`CRYDriver *cry`, `HMACSHA512Context *hmacsha512ctxp`)
Hash initialization using HMAC_SHA512.
- `cryerror_t cry_lld_HMACSHA512_update` (`CRYDriver *cry`, `HMACSHA512Context *hmacsha512ctxp`, `size_t size`, `const uint8_t *in`)
Hash update using HMAC.
- `cryerror_t cry_lld_HMACSHA512_final` (`CRYDriver *cry`, `HMACSHA512Context *hmacsha512ctxp`, `uint8_t *out`)
Hash finalization using HMAC.

Variables

- `CRYDriver CRYD1`
CRY1 driver identifier.

7.2.9.5.2 Macro Definition Documentation

7.2.9.5.2.1 HAL_CRY_USE_FALLBACK [1/2]

```
#define HAL_CRY_USE_FALLBACK FALSE
```

Enables the SW fall-back of the cryptographic driver.

When enabled, this option, activates a fall-back software implementation for algorithms not supported by the underlying hardware.

Note

Fall-back implementations may not be present for all algorithms.

7.2.9.5.2.2 HAL_CRY_USE_FALLBACK [2/2]

```
#define HAL_CRY_USE_FALLBACK TRUE
```

Enables the SW fall-back of the cryptographic driver.

When enabled, this option, activates a fall-back software implementation for algorithms not supported by the underlying hardware.

Note

Fall-back implementations may not be present for all algorithms.

7.2.9.5.2.3 HAL_CRY_ENFORCE_FALLBACK

```
#define HAL_CRY_ENFORCE_FALLBACK FALSE
```

Makes the driver forcibly use the fall-back implementations.

Note

If enabled then the LLD driver is not included at all.

7.2.9.5.2.4 CRY_LLD_SUPPORTS_AES

```
#define CRY_LLD_SUPPORTS_AES TRUE
```

7.2.9.5.2.5 CRY_LLD_SUPPORTS_AES_ECB

```
#define CRY_LLD_SUPPORTS_AES_ECB TRUE
```

7.2.9.5.2.6 CRY_LLD_SUPPORTS_AES_CBC

```
#define CRY_LLD_SUPPORTS_AES_CBC TRUE
```

7.2.9.5.2.7 CRY_LLD_SUPPORTS_AES_CFB

```
#define CRY_LLD_SUPPORTS_AES_CFB TRUE
```

7.2.9.5.2.8 CRY_LLD_SUPPORTS_AES_CTR

```
#define CRY_LLD_SUPPORTS_AES_CTR TRUE
```

7.2.9.5.2.9 CRY_LLD_SUPPORTS_AES_GCM

```
#define CRY_LLD_SUPPORTS_AES_GCM TRUE
```

7.2.9.5.2.10 CRY_LLD_SUPPORTS_DES

```
#define CRY_LLD_SUPPORTS_DES TRUE
```

7.2.9.5.2.11 CRY_LLD_SUPPORTS_DES_ECB

```
#define CRY_LLD_SUPPORTS_DES_ECB TRUE
```

7.2.9.5.2.12 CRY_LLD_SUPPORTS_DES_CBC

```
#define CRY_LLD_SUPPORTS_DES_CBC TRUE
```

7.2.9.5.2.13 CRY_LLD_SUPPORTS_SHA1

```
#define CRY_LLD_SUPPORTS_SHA1 TRUE
```

7.2.9.5.2.14 CRY_LLD_SUPPORTS_SHA256

```
#define CRY_LLD_SUPPORTS_SHA256 TRUE
```

7.2.9.5.2.15 CRY_LLD_SUPPORTS_SHA512

```
#define CRY_LLD_SUPPORTS_SHA512 TRUE
```

7.2.9.5.2.16 CRY_LLD_SUPPORTS_HMAC_SHA256

```
#define CRY_LLD_SUPPORTS_HMAC_SHA256 TRUE
```

7.2.9.5.2.17 CRY_LLD_SUPPORTS_HMAC_SHA512

```
#define CRY_LLD_SUPPORTS_HMAC_SHA512 TRUE
```

7.2.9.5.2.18 PLATFORM_CRY_USE_CRY1

```
#define PLATFORM_CRY_USE_CRY1 FALSE
```

CRY1 driver enable switch.

If set to TRUE the support for CRY1 is included.

Note

The default is FALSE.

7.2.9.5.3 Typedef Documentation

7.2.9.5.3.1 bitsize_t

```
typedef size_t bitsize_t
```

Size, in bits, of a crypto field or message.

Note

It is assumed, for simplicity, that this type is equivalent to a `size_t`.

7.2.9.5.3.2 crykey_t

```
typedef uint32_t crykey_t
```

CRY key identifier type.

7.2.9.5.3.3 CRYDriver

```
typedef struct CRYDriver CRYDriver
```

Type of a structure representing an CRY driver.

7.2.9.5.4 Enumeration Type Documentation

7.2.9.5.4.1 crystate_t

```
enum crystate_t
```

Driver state machine possible states.

Enumerator

CRY_UNINIT	Not initialized.
CRY_STOP	Stopped.
CRY_READY	Ready.

7.2.9.5.4.2 cryerror_t

enum `cryerror_t`

Driver error codes.

Enumerator

CRY_NOERROR	No error.
CRY_ERR_INV_ALGO	Invalid cypher/mode.
CRY_ERR_INV_KEY_SIZE	Invalid key size.
CRY_ERR_INV_KEY_TYPE	Invalid key type.
CRY_ERR_INV_KEY_ID	Invalid key identifier.
CRY_ERR_AUTH_FAILED	Failed authentication.
CRY_ERR_OP_FAILURE	Failed operation.

7.2.9.5.4.3 cryalgorithm_t

enum `cryalgorithm_t`

Type of an algorithm identifier.

Note

It is only used to determine the key required for operations.

Enumerator

cry_algo_none	
cry_algo_aes	AES 128, 192, 256 bits.
cry_algo_des	DES 56, TDES 112, 168 bits.
cry_algo_hmac	HMAC variable size.

7.2.9.5.5 Function Documentation

7.2.9.5.5.1 cryInit()

```
void cryInit (  
    void )
```

Cryptographic Driver initialization.

Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.5.5.2 cryObjectInit()

```
void cryObjectInit (  
    CRYDriver * cryp)
```

Initializes the standard part of a `CRYDriver` structure.

Parameters

out	<i>cryp</i>	pointer to the <code>CRYDriver</code> object

Function Class:

Object or module initializer function.

7.2.9.5.5.3 cryStart()

```
msg_t cryStart (  
    CRYDriver * cryp,  
    const CRYConfig * config)
```

Configures and activates the cryptographic peripheral.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>config</i>	pointer to the CRYConfig object. Depending on the implementation the value can be <code>NULL</code> .

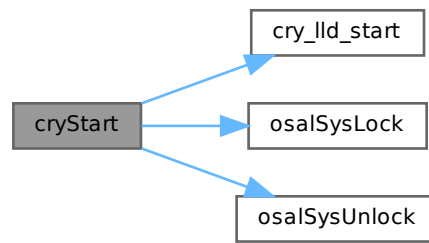
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.4 cryStop()**

```
void cryStop (  
    CRYDriver * cryp)
```

Deactivates the cryptographic peripheral.

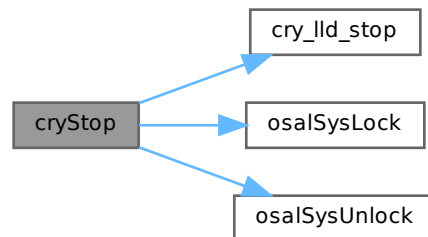
Parameters

in	<i>cryp</i>	pointer to the CRYDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5 cryLoadAESTransientKey()**

```

cryerror_t cryLoadAESTransientKey (
    CRYDriver * cryp,
    size_t size,
    const uint8_t * keyp)
  
```

Initializes the AES transient key.

Note

It is the underlying implementation to decide which key sizes are allowable.

Parameters

in	<i>cryp</i>	pointer to the <code>CRYDriver</code> object
in	<i>size</i>	key size in bytes
in	<i>keyp</i>	pointer to the key data

Returns

The operation status.

Return values

<code>CRY_NOERROR</code>	if the operation succeeded.
<code>CRY_ERR_INV_ALGO</code>	if the algorithm is unsupported.
<code>CRY_ERR_INV_KEY_SIZE</code>	if the specified key size is invalid for the specified algorithm.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.6 cryEncryptAES()**

```

cryerror_t cryEncryptAES (
    CRYDriver * cryp,
    crykey_t key_id,
    const uint8_t * in,
    uint8_t * out)
  
```

Encryption of a single block using AES.

Note

The implementation of this function must guarantee that it can be called from any context.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext

Returns

The operation status.

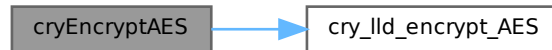
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:

**7.2.9.5.5.7 cryDecryptAES()**

```

cryerror_t cryDecryptAES (
    CRYDriver * cryp,
    crykey_t key_id,
    const uint8_t * in,
    uint8_t * out)
  
```

Decryption of a single block using AES.

Note

The implementation of this function must guarantee that it can be called from any context.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext

Returns

The operation status.

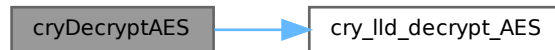
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:

**7.2.9.5.5.8 cryEncryptAES_ECB()**

```

cryerror_t cryEncryptAES_ECB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out)
  
```

Encryption operation using AES-ECB.

Note

The function operates on data buffers whose length is a multiple of an AES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the <code>CRYDriver</code> object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 16
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext

Returns

The operation status.

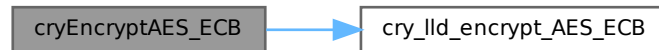
Return values

<code>CRY_NOERROR</code>	if the operation succeeded.
<code>CRY_ERR_INV_ALGO</code>	if the operation is unsupported on this device instance.
<code>CRY_ERR_INV_KEY_TYPE</code>	the selected key is invalid for this operation.
<code>CRY_ERR_INV_KEY_ID</code>	if the specified key identifier is invalid or refers to an empty key slot.
<code>CRY_ERR_OP_FAILURE</code>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.9 cryDecryptAES_ECB()**

```

cryerror_t cryDecryptAES_ECB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out)
  
```

Decryption operation using AES-ECB.

Note

The function operates on data buffers whose length is a multiple of an AES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 16
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext

Returns

The operation status.

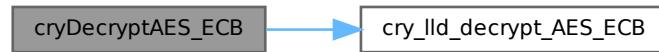
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.10 cryEncryptAES_CBC()**

```

cryerror_t cryEncryptAES_CBC (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
  
```

Encryption operation using AES-CBC.

Note

The function operates on data buffers whose length is a multiple of an AES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 16
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits input vector

Returns

The operation status.

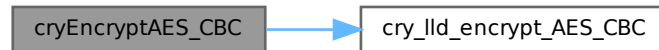
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.11 cryDecryptAES_CBC()**

```

cryerror_t cryDecryptAES_CBC (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
  
```

Decryption operation using AES-CBC.

Note

The function operates on data buffers whose length is a multiple of an AES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 16
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext
in	<i>iv</i>	128 bits input vector

Returns

The operation status.

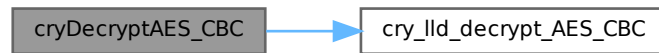
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.12 cryEncryptAES_CFB()**

```

cryerror_t cryEncryptAES_CFB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
  
```

Encryption operation using AES-CFB.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits input vector

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.13 cryDecryptAES_CFB()**

```

cryerror_t cryDecryptAES_CFB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
  
```

Decryption operation using AES-CFB.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext
in	<i>iv</i>	128 bits input vector

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.14 cryEncryptAES_CTR()**

```

cryerror_t cryEncryptAES_CTR (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
  
```

Encryption operation using AES-CTR.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits input vector + counter, it contains a 96 bits IV and a 32 bits counter

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.15 cryDecryptAES_CTR()**

```

cryerror_t cryDecryptAES_CTR (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
  
```

Decryption operation using AES-CTR.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext
in	<i>iv</i>	128 bits input vector + counter, it contains a 96 bits IV and a 32 bits counter

Returns

The operation status.

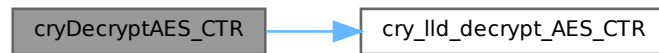
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.16 cryEncryptAES_GCM()**

```

cryerror_t cryEncryptAES_GCM (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t auth_size,
    const uint8_t * auth_in,
    size_t text_size,
    const uint8_t * text_in,
    uint8_t * text_out,
    const uint8_t * iv,
    size_t tag_size,
    uint8_t * tag_out)
  
```

Encryption operation using AES-GCM.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the <code>CRYDriver</code> object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>auth_size</i>	size of the data buffer to be authenticated
in	<i>auth_in</i>	buffer containing the data to be authenticated
in	<i>text_size</i>	size of the text buffer
in	<i>text_in</i>	buffer containing the input plaintext
out	<i>text_out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits input vector
in	<i>tag_size</i>	size of the authentication tag, this number must be between 1 and 16
out	<i>tag_out</i>	buffer for the generated authentication tag

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.17 cryDecryptAES_GCM()

```

cryerror_t cryDecryptAES_GCM (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t auth_size,
    const uint8_t * auth_in,
    size_t text_size,
    const uint8_t * text_in,
    uint8_t * text_out,
    const uint8_t * iv,
    size_t tag_size,
    const uint8_t * tag_in)
  
```

Decryption operation using AES-GCM.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the <i>CRYDriver</i> object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>auth_size</i>	size of the data buffer to be authenticated
in	<i>auth_in</i>	buffer containing the data to be authenticated

Parameters

in	<i>text_size</i>	size of the text buffer
in	<i>text_in</i>	buffer containing the input plaintext
out	<i>text_out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits input vector
in	<i>tag_size</i>	size of the authentication tag, this number must be between 1 and 16
in	<i>tag_in</i>	buffer for the generated authentication tag

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_AUTH_FAILED</i>	authentication failed
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.18 cryLoadDESTransientKey()

```

cryerror_t cryLoadDESTransientKey (
    CRYDriver * cryp,
    size_t size,
    const uint8_t * keyp)
  
```

Initializes the DES transient key.

Note

It is the underlying implementation to decide which key sizes are allowable.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>size</i>	key size in bytes
in	<i>keyp</i>	pointer to the key data

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the algorithm is unsupported.
<i>CRY_ERR_INV_KEY_SIZE</i>	if the specified key size is invalid for the specified algorithm.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.19 cryEncryptDES()**

```

cryerror_t cryEncryptDES (
    CRYDriver * cryp,
    crykey_t key_id,
    const uint8_t * in,
    uint8_t * out)
  
```

Encryption of a single block using (T)DES.

Note

The implementation of this function must guarantee that it can be called from any context.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext

Returns

The operation status.

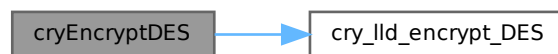
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:

**7.2.9.5.5.20 cryDecryptDES()**

```

cryerror_t cryDecryptDES (
    CRYDriver * cryp,
    crykey_t key_id,
    const uint8_t * in,
    uint8_t * out)
  
```

Decryption of a single block using (T)DES.

Note

The implementation of this function must guarantee that it can be called from any context.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:



7.2.9.5.5.21 cryEncryptDES_ECB()

```

cryerror_t cryEncryptDES_ECB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out)
  
```

Encryption operation using (T)DES-ECB.

Note

The function operates on data buffers whose length is a multiple of an DES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 8
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext

Returns

The operation status.

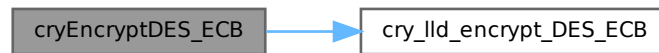
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.22 cryDecryptDES_ECB()

```

cryerror_t cryDecryptDES_ECB (
    CRYDriver * crip,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out)
  
```

Decryption operation using (T)DES-ECB.

Note

The function operates on data buffers whose length is a multiple of an DES block, this means that padding must be done by the caller.

Parameters

in	<i>crip</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 8
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.23 cryEncryptDES_CBC()

```

cryerror_t cryEncryptDES_CBC (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
  
```

Encryption operation using (T)DES-CBC.

Note

The function operates on data buffers whose length is a multiple of an DES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 8
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	64 bits input vector

Returns

The operation status.

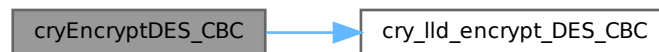
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.24 cryDecryptDES_CBC()

```

cryerror_t cryDecryptDES_CBC (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
  
```

Decryption operation using (T)DES-CBC.

Note

The function operates on data buffers whose length is a multiple of an DES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the <code>CRYDriver</code> object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 8
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext
in	<i>iv</i>	64 bits input vector

Returns

The operation status.

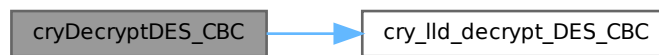
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.25 crySHA1Init()

```

cryerror_t crySHA1Init (
    CRYDriver * cryp,
    SHA1Context * sha1ctxp)
  
```

Hash initialization using SHA1.

Note

Use of this algorithm is not recommended because proven weak.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
out	<i>sha1ctxp</i>	pointer to a SHA1 context to be initialized

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.26 crySHA1Update()**

```

cryerror_t crySHA1Update (
    CRYDriver * cryp,
    SHA1Context * sha1ctxp,
    size_t size,
    const uint8_t * in)
  
```

Hash update using SHA1.

Note

Use of this algorithm is not recommended because proven weak.

Parameters

in	<i>cryp</i>	pointer to the <code>CRYDriver</code> object
in	<i>sha1ctxp</i>	pointer to a SHA1 context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

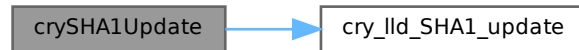
Return values

<code>CRY_NOERROR</code>	if the operation succeeded.
<code>CRY_ERR_INV_ALGO</code>	if the operation is unsupported on this device instance.
<code>CRY_ERR_OP_FAILURE</code>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.27 crySHA1Final()**

```

cryerror_t crySHA1Final (
    CRYDriver * cryp,
    SHA1Context * sha1ctxp,
    uint8_t * out)
  
```

Hash finalization using SHA1.

Note

Use of this algorithm is not recommended because proven weak.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>sha1ctxp</i>	pointer to a SHA1 context
out	<i>out</i>	160 bits output buffer

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.28 crySHA256Init()**

```

cryerror_t crySHA256Init (
    CRYDriver * cryp,
    SHA256Context * sha256ctxp)
  
```

Hash initialization using SHA256.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
out	<i>sha256ctxp</i>	pointer to a SHA256 context to be initialized

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.29 crySHA256Update()

```

cryerror_t crySHA256Update (
    CRYDriver * cryp,
    SHA256Context * sha256ctxp,
    size_t size,
    const uint8_t * in)

```

Hash update using SHA256.

Parameters

in	<i>cryp</i>	pointer to the <code>CRYDriver</code> object
in	<i>sha256ctxp</i>	pointer to a SHA256 context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

Return values

<code>CRY_NOERROR</code>	if the operation succeeded.
<code>CRY_ERR_INV_ALGO</code>	if the operation is unsupported on this device instance.
<code>CRY_ERR_OP_FAILURE</code>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.30 crySHA256Final()

```

cryerror_t crySHA256Final (
    CRYDriver * cryp,
    SHA256Context * sha256ctxp,
    uint8_t * out)

```

Hash finalization using SHA256.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>sha256ctxp</i>	pointer to a SHA256 context
out	<i>out</i>	256 bits output buffer

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.31 crySHA512Init()**

```
cryerror_t crySHA512Init (  
    CRYDriver * cryp,  
    SHA512Context * sha512ctxp)
```

Hash initialization using SHA512.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
out	<i>sha512ctxp</i>	pointer to a SHA512 context to be initialized

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.32 crySHA512Update()

```

cryerror_t crySHA512Update (
    CRYDriver * cryp,
    SHA512Context * sha512ctxp,
    size_t size,
    const uint8_t * in)
  
```

Hash update using SHA512.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>sha512ctxp</i>	pointer to a SHA512 context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.33 crySHA512Final()**

```

cryerror_t crySHA512Final (
    CRYDriver * cryp,
    SHA512Context * sha512ctxp,
    uint8_t * out)
  
```

Hash finalization using SHA512.

Parameters

in	<i>cryp</i>	pointer to the <code>CRYDriver</code> object
in	<i>sha512ctxp</i>	pointer to a SHA512 context
out	<i>out</i>	512 bits output buffer

Returns

The operation status.

Return values

<code>CRY_NOERROR</code>	if the operation succeeded.
<code>CRY_ERR_INV_ALGO</code>	if the operation is unsupported on this device instance.
<code>CRY_ERR_OP_FAILURE</code>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.34 cryLoadHMACTransientKey()

```
cryerror_t cryLoadHMACTransientKey (
    CRYDriver * cryp,
    size_t size,
    const uint8_t * keyp)
```

Initializes the HMAC transient key.

Note

It is the underlying implementation to decide which key sizes are allowable.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>size</i>	key size in bytes
in	<i>keyp</i>	pointer to the key data

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the algorithm is unsupported.
<i>CRY_ERR_INV_KEY_SIZE</i>	if the specified key size is invalid for the specified algorithm.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.35 cryHMACSHA256Init()

```
cryerror_t cryHMACSHA256Init (
    CRYDriver * cryp,
    HMACSHA256Context * hmacsha256ctxp)
```

Hash initialization using HMAC_SHA256.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
out	<i>hmacsha256ctxp</i>	pointer to a HMAC_SHA256 context to be initialized

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.36 cryHMACSHA256Update()

```

cryerror_t cryHMACSHA256Update (
    CRYDriver * cryp,
    HMACSHA256Context * hmacsha256ctxp,
    size_t size,
    const uint8_t * in)
  
```

Hash update using HMAC.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>hmacsha256ctxp</i>	pointer to a HMAC_SHA256 context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.37 cryHMACSHA256Final()

```

cryerror_t cryHMACSHA256Final (
    CRYDriver * cryp,
    HMACSHA256Context * hmacsha256ctxp,
    uint8_t * out)
  
```

Hash finalization using HMAC.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>hmacsha256ctxp</i>	pointer to a HMAC_SHA256 context
out	<i>out</i>	256 bits output buffer

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.38 cryHMACSHA512Init()**

```

cryerror_t cryHMACSHA512Init (
    CRYDriver * cryp,
    HMACSHA512Context * hmacsha512ctxp)
  
```

Hash initialization using HMAC_SHA512.

Parameters

in	<i>cryp</i>	pointer to the <code>CRYDriver</code> object
out	<i>hmacsha512ctxp</i>	pointer to a HMAC_SHA512 context to be initialized

Returns

The operation status.

Return values

<code>CRY_NOERROR</code>	if the operation succeeded.
<code>CRY_ERR_INV_ALGO</code>	if the operation is unsupported on this device instance.
<code>CRY_ERR_OP_FAILURE</code>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.39 cryHMACSHA512Update()

```

cryerror_t cryHMACSHA512Update (
    CRYDriver * cryp,
    HMACSHA512Context * hmacsha512ctxp,
    size_t size,
    const uint8_t * in)

```

Hash update using HMAC.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>hmacsha512ctxp</i>	pointer to a HMAC_SHA512 context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.5.5.40 cryHMACSHA512Final()

```

cryerror_t cryHMACSHA512Final (
    CRYDriver * cryp,
    HMACSHA512Context * hmacsha512ctxp,
    uint8_t * out)

```

Hash finalization using HMAC.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>hmacsha512ctxp</i>	pointer to a HMAC_SHA512 context
out	<i>out</i>	512 bits output buffer

Returns

The operation status.

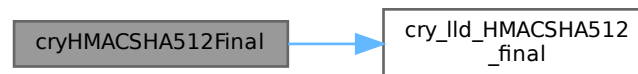
Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.5.5.41 cry_lld_init()**

```
void cry_lld_init (  
    void )
```

Low level crypto driver initialization.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.42 cry_lld_start()

```
void cry_lld_start (  
    CRYDriver * cryp)
```

Configures and activates the crypto peripheral.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.43 cry_lld_stop()

```
void cry_lld_stop (  
    CRYDriver * cryp)
```

Deactivates the crypto peripheral.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.44 cry_lld_aes_loadkey()

```
cryerror\_t cry_lld_aes_loadkey (  
    CRYDriver * cryp,  
    size\_t size,  
    const uint8\_t * keyp)
```

Initializes the AES transient key.

Note

It is the underlying implementation to decide which key sizes are allowable.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>size</i>	key size in bytes
in	<i>keyp</i>	pointer to the key data

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the algorithm is unsupported.
<i>CRY_ERR_INV_KEY_SIZE</i>	if the specified key size is invalid for the specified algorithm.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.45 **cry_lld_encrypt_AES()**

```

cryerror_t cry_lld_encrypt_AES (
    CRYDriver * cryp,
    crykey_t key_id,
    const uint8_t * in,
    uint8_t * out)

```

Encryption of a single block using AES.

Note

The implementation of this function must guarantee that it can be called from any context.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.46 cry_lld_decrypt_AES()

```
cryerror_t cry_lld_decrypt_AES (
    CRYDriver * cryp,
    crykey_t key_id,
    const uint8_t * in,
    uint8_t * out)
```

Decryption of a single block using AES.

Note

The implementation of this function must guarantee that it can be called from any context.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.47 cry_lld_encrypt_AES_ECB()

```
cryerror_t cry_lld_encrypt_AES_ECB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out)
```

Encryption operation using AES-ECB.

Note

The function operates on data buffers whose length is a multiple of an AES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 16
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.48 cry_lld_decrypt_AES_ECB()

```

cryerror_t cry_lld_decrypt_AES_ECB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out)

```

Decryption operation using AES-ECB.

Note

The function operates on data buffers whose length is a multiple of an AES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 16
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.49 cry_lld_encrypt_AES_CBC()

```

cryerror_t cry_lld_encrypt_AES_CBC (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)

```

Encryption operation using AES-CBC.

Note

The function operates on data buffers whose length is a multiple of an AES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 16
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits initial vector

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.50 cry_llid_decrypt_AES_CBC()

```

cryerror_t cry_llid_decrypt_AES_CBC (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)

```

Decryption operation using AES-CBC.

Note

The function operates on data buffers whose length is a multiple of an AES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers, this number must be a multiple of 16
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits initial vector

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.51 cry_llid_encrypt_AES_CFB()

```

cryerror_t cry_llid_encrypt_AES_CFB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)

```

Encryption operation using AES-CFB.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits initial vector

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.52 `cry_llc_decrypt_AES_CFB()`

```

cryerror_t cry_llc_decrypt_AES_CFB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)

```

Decryption operation using AES-CFB.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits initial vector

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.53 cry_lld_encrypt_AES_CTR()

```
cryerror_t cry_lld_encrypt_AES_CTR (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
```

Encryption operation using AES-CTR.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits initial vector + counter, it contains a 96 bits IV and a 32 bits counter

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.54 cry_llid_decrypt_AES_CTR()

```

cryerror_t cry_llid_decrypt_AES_CTR (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)

```

Decryption operation using AES-CTR.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of both buffers
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext
in	<i>iv</i>	128 bits initial vector + counter, it contains a 96 bits IV and a 32 bits counter

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.55 cry_llid_encrypt_AES_GCM()

```

cryerror_t cry_llid_encrypt_AES_GCM (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t auth_size,
    const uint8_t * auth_in,
    size_t text_size,
    const uint8_t * text_in,
    uint8_t * text_out,
    const uint8_t * iv,
    size_t tag_size,
    uint8_t * tag_out)

```

Encryption operation using AES-GCM.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>auth_size</i>	size of the data buffer to be authenticated
in	<i>auth_in</i>	buffer containing the data to be authenticated
in	<i>text_size</i>	size of the text buffer
in	<i>text_in</i>	buffer containing the input plaintext
out	<i>text_out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits input vector
in	<i>tag_size</i>	size of the authentication tag, this number must be between 1 and 16
out	<i>tag_out</i>	buffer for the generated authentication tag

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.56 cry_lld_decrypt_AES_GCM()

```

cryerror_t cry_lld_decrypt_AES_GCM (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t auth_size,
    const uint8_t * auth_in,
    size_t text_size,
    const uint8_t * text_in,
    uint8_t * text_out,
    const uint8_t * iv,
    size_t tag_size,
    const uint8_t * tag_in)

```

Decryption operation using AES-GCM.

Note

This is a stream cipher, there are no size restrictions.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>auth_size</i>	size of the data buffer to be authenticated
in	<i>auth_in</i>	buffer containing the data to be authenticated
in	<i>text_size</i>	size of the text buffer
in	<i>text_in</i>	buffer containing the input plaintext
out	<i>text_out</i>	buffer for the output ciphertext
in	<i>iv</i>	128 bits input vector
in	<i>tag_size</i>	size of the authentication tag, this number must be between 1 and 16
in	<i>tag_in</i>	buffer for the generated authentication tag

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_AUTH_FAILED</i>	authentication failed
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.57 cry_lld_des_loadkey()

```
cryerror_t cry_lld_des_loadkey (
    CRYDriver * cryp,
    size_t size,
    const uint8_t * keyp)
```

Initializes the DES transient key.

Note

It is the underlying implementation to decide which key sizes are allowable.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>size</i>	key size in bytes
in	<i>keyp</i>	pointer to the key data

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the algorithm is unsupported.
<i>CRY_ERR_INV_KEY_SIZE</i>	if the specified key size is invalid for the specified algorithm.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.58 cry_lld_encrypt_DES()

```
cryerror_t cry_lld_encrypt_DES (
    CRYDriver * cryp,
    crykey_t key_id,
    const uint8_t * in,
    uint8_t * out)
```

Encryption of a single block using (T)DES.

Note

The implementation of this function must guarantee that it can be called from any context.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.59 cry_lld_decrypt_DES()

```

cryerror_t cry_lld_decrypt_DES (
    CRYDriver * cryp,
    crykey_t key_id,
    const uint8_t * in,
    uint8_t * out)

```

Decryption of a single block using (T)DES.

Note

The implementation of this function must guarantee that it can be called from any context.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.60 cry_lld_encrypt_DES_ECB()

```

cryerror_t cry_lld_encrypt_DES_ECB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out)

```

Encryption operation using (T)DES-ECB.

Note

The function operates on data buffers whose length is a multiple of an DES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of the plaintext buffer, this number must be a multiple of 8
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.61 cry_llc_decrypt_DES_ECB()

```
cryerror_t cry_llc_decrypt_DES_ECB (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out)
```

Decryption operation using (T)DES-ECB.

Note

The function operates on data buffers whose length is a multiple of an DES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of the plaintext buffer, this number must be a multiple of 8
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.62 cry_lld_encrypt_DES_CBC()

```
cryerror_t cry_lld_encrypt_DES_CBC (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
```

Encryption operation using (T)DES-CBC.

Note

The function operates on data buffers whose length is a multiple of an DES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔ _id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of the plaintext buffer, this number must be a multiple of 8
in	<i>in</i>	buffer containing the input plaintext
out	<i>out</i>	buffer for the output ciphertext
in	<i>iv</i>	64 bits input vector

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.63 cry_llid_decrypt_DES_CBC()

```
cryerror_t cry_llid_decrypt_DES_CBC (
    CRYDriver * cryp,
    crykey_t key_id,
    size_t size,
    const uint8_t * in,
    uint8_t * out,
    const uint8_t * iv)
```

Decryption operation using (T)DES-CBC.

Note

The function operates on data buffers whose length is a multiple of an DES block, this means that padding must be done by the caller.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>key↔_id</i>	the key to be used for the operation, zero is the transient key, other values are keys stored in an unspecified way
in	<i>size</i>	size of the plaintext buffer, this number must be a multiple of 8
in	<i>in</i>	buffer containing the input ciphertext
out	<i>out</i>	buffer for the output plaintext
in	<i>iv</i>	64 bits input vector

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_INV_KEY_TYPE</i>	the selected key is invalid for this operation.
<i>CRY_ERR_INV_KEY_ID</i>	if the specified key identifier is invalid or refers to an empty key slot.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.64 cry_llid_SHA1_init()

```
cryerror_t cry_llid_SHA1_init (
    CRYDriver * cryp,
    SHA1Context * sha1ctxp)
```

Hash initialization using SHA1.

Note

Use of this algorithm is not recommended because proven weak.

Parameters

in	<i>crypt</i>	pointer to the CRYDriver object
out	<i>sha1ctxp</i>	pointer to a SHA1 context to be initialized

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.65 **cry_llc_SHA1_update()**

```

cryerror_t cry_llc_SHA1_update (
    CRYDriver * crypt,
    SHA1Context * sha1ctxp,
    size_t size,
    const uint8_t * in)

```

Hash update using SHA1.

Note

Use of this algorithm is not recommended because proven weak.

Parameters

in	<i>crypt</i>	pointer to the CRYDriver object
in	<i>sha1ctxp</i>	pointer to a SHA1 context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.66 cry_llc_SHA1_final()

```
cryerror_t cry_llc_SHA1_final (  
    CRYDriver * cryp,  
    SHA1Context * sha1ctxp,  
    uint8_t * out)
```

Hash finalization using SHA1.

Note

Use of this algorithm is not recommended because proven weak.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>sha1ctxp</i>	pointer to a SHA1 context
out	<i>out</i>	160 bits output buffer

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.67 cry_llc_SHA256_init()

```
cryerror_t cry_llc_SHA256_init (  
    CRYDriver * cryp,  
    SHA256Context * sha256ctxp)
```

Hash initialization using SHA256.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
out	<i>sha256ctxp</i>	pointer to a SHA256 context to be initialized

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.68 **cry_lld_SHA256_update()**

```

cryerror_t cry_lld_SHA256_update (
    CRYDriver * cryp,
    SHA256Context * sha256ctxp,
    size_t size,
    const uint8_t * in)

```

Hash update using SHA256.

Parameters

in	<i>cryp</i>	pointer to the <i>CRYDriver</i> object
in	<i>sha256ctxp</i>	pointer to a SHA256 context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.69 **cry_lld_SHA256_final()**

```

cryerror_t cry_lld_SHA256_final (
    CRYDriver * cryp,
    SHA256Context * sha256ctxp,
    uint8_t * out)

```

Hash finalization using SHA256.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>sha256ctxp</i>	pointer to a SHA256 context
out	<i>out</i>	256 bits output buffer

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.70 cry_llc_SHA512_init()

```
cryerror_t cry_llc_SHA512_init (  
    CRYDriver * cryp,  
    SHA512Context * sha512ctxp)
```

Hash initialization using SHA512.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
out	<i>sha512ctxp</i>	pointer to a SHA512 context to be initialized

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.71 cry_llc_SHA512_update()

```
cryerror_t cry_llc_SHA512_update (  
    CRYDriver * cryp,  
    SHA512Context * sha512ctxp,  
    size_t size,  
    const uint8_t * in)
```

Hash update using SHA512.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>sha512ctxp</i>	pointer to a SHA512 context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.72 cry_llc_SHA512_final()

```
cryerror_t cry_llc_SHA512_final (  
    CRYDriver * cryp,  
    SHA512Context * sha512ctxp,  
    uint8_t * out)
```

Hash finalization using SHA512.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>sha512ctxp</i>	pointer to a SHA512 context
out	<i>out</i>	512 bits output buffer

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.73 cry_lld_hmac_loadkey()

```
cryerror_t cry_lld_hmac_loadkey (  
    CRYDriver * cryp,  
    size_t size,  
    const uint8_t * keyp)
```

Initializes the HMAC transient key.

Note

It is the underlying implementation to decide which key sizes are allowable.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>size</i>	key size in bytes
in	<i>keyp</i>	pointer to the key data

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the algorithm is unsupported.
<i>CRY_ERR_INV_KEY_SIZE</i>	if the specified key size is invalid for the specified algorithm.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.74 cry_lld_HMACSHA256_init()

```
cryerror_t cry_lld_HMACSHA256_init (  
    CRYDriver * cryp,  
    HMACSHA256Context * hmacsha256ctxp)
```

Hash initialization using HMAC_SHA256.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
out	<i>hmacsha256ctxp</i>	pointer to a HMAC_SHA256 context to be initialized

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.75 `cry_lld_HMACSHA256_update()`

```

cryerror_t cry_lld_HMACSHA256_update (
    CRYDriver * cryp,
    HMACSHA256Context * hmacsha256ctxp,
    size_t size,
    const uint8_t * in)

```

Hash update using HMAC.

Parameters

in	<i>cryp</i>	pointer to the <code>CRYDriver</code> object
in	<i>hmacsha256ctxp</i>	pointer to a <code>HMAC_SHA256</code> context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.76 `cry_lld_HMACSHA256_final()`

```

cryerror_t cry_lld_HMACSHA256_final (
    CRYDriver * cryp,
    HMACSHA256Context * hmacsha256ctxp,
    uint8_t * out)

```

Hash finalization using HMAC.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>hmacsha256ctxp</i>	pointer to a HMAC_SHA256 context
out	<i>out</i>	256 bits output buffer

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.77 cry_lld_HMACSHA512_init()

```
cryerror_t cry_lld_HMACSHA512_init (
    CRYDriver * cryp,
    HMACSHA512Context * hmacsha512ctxp)
```

Hash initialization using HMAC_SHA512.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
out	<i>hmacsha512ctxp</i>	pointer to a HMAC_SHA512 context to be initialized

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.78 cry_lld_HMACSHA512_update()

```
cryerror_t cry_lld_HMACSHA512_update (
    CRYDriver * cryp,
    HMACSHA512Context * hmacsha512ctxp,
    size_t size,
    const uint8_t * in)
```

Hash update using HMAC.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>hmacsha512ctxp</i>	pointer to a HMAC_SHA512 context
in	<i>size</i>	size of input buffer
in	<i>in</i>	buffer containing the input text

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.5.79 cry_lld_HMACSHA512_final()

```

cryerror_t cry_lld_HMACSHA512_final (
    CRYDriver * cryp,
    HMACSHA512Context * hmacsha512ctxp,
    uint8_t * out)

```

Hash finalization using HMAC.

Parameters

in	<i>cryp</i>	pointer to the CRYDriver object
in	<i>hmacsha512ctxp</i>	pointer to a HMAC_SHA512 context
out	<i>out</i>	512 bits output buffer

Returns

The operation status.

Return values

<i>CRY_NOERROR</i>	if the operation succeeded.
<i>CRY_ERR_INV_ALGO</i>	if the operation is unsupported on this device instance.
<i>CRY_ERR_OP_FAILURE</i>	if the operation failed, implementation dependent.

Function Class:

Not an API, this function is for internal use only.

7.2.9.5.6 Variable Documentation

7.2.9.5.6.1 CRYD1

`CRYDriver` CRYD1

CRY1 driver identifier.

7.2.9.6 DAC Driver

Generic DAC Driver.

7.2.9.6.1 Detailed Description

Generic DAC Driver.

This module implements a generic DAC (Digital to Analog Converter) driver.

Precondition

In order to use the DAC driver the `HAL_USE_DAC` option must be enabled in `halconf.h`.

DAC configuration options

- `#define DAC_USE_SYNCHRONIZATION FALSE`
Support for thread synchronization API.
- `#define DAC_USE_MUTUAL_EXCLUSION TRUE`
Enables the `dacAcquireBus()` and `dacReleaseBus()` APIs.

Low level driver helper macros

- `#define dacsBufferComplete(dacp)`
Buffer state.
- `#define _dac_wait_s(dacp)`
Waits for operation completion.
- `#define _dac_reset_i(dacp)`
Resumes a thread waiting for a conversion completion.
- `#define _dac_reset_s(dacp)`
Resumes a thread waiting for a conversion completion.
- `#define _dac_wakeup_isr(dacp)`
Wakes up the waiting thread.
- `#define _dac_timeout_isr(dacp)`
Wakes up the waiting thread with a timeout message.
- `#define _dac_isr_half_code(dacp)`
Common ISR code, half buffer event.
- `#define _dac_isr_full_code(dacp)`
Common ISR code, full buffer event.
- `#define _dac_isr_error_code(dacp, err)`
Common ISR code, error event.

Configuration options

- #define `PLATFORM_DAC_USE_DAC1` `FALSE`
DAC1 CH1 driver enable switch.

Data Structures

- struct `hal_dac_conversion_group`
DAC Conversion group structure.
- struct `hal_dac_config`
Driver configuration structure.
- struct `hal_dac_driver`
Structure representing a DAC driver.

Macros

- #define `DAC_MAX_CHANNELS` `2`
Maximum number of DAC channels per unit.
- #define `dac_lld_driver_fields`
Low level fields of the DAC driver structure.
- #define `dac_lld_config_fields`
Low level fields of the DAC configuration structure.
- #define `dac_lld_conversion_group_fields`
Low level fields of the DAC group configuration structure.

Typedefs

- typedef struct `hal_dac_driver` `DACDriver`
Type of a structure representing an DAC driver.
- typedef struct `hal_dac_config` `DACConfig`
Type of a structure representing an DAC driver configuration.
- typedef struct `hal_dac_conversion_group` `DACConversionGroup`
Type of a DAC conversion group.
- typedef void(* `daccallback_t`) (`DACDriver` *dACP)
DAC notification callback type.
- typedef void(* `dacerrorcallback_t`) (`DACDriver` *dACP, `dacerror_t` err)
DAC error callback type.
- typedef uint32_t `dacchannel_t`
Type of a DAC channel index.
- typedef uint16_t `dacsample_t`
Type representing a DAC sample.

Enumerations

- enum `dacstate_t` {
 `DAC_UNINIT` = 0 , `DAC_STOP` = 1 , `DAC_READY` = 2 , `DAC_ACTIVE` = 3 ,
 `DAC_COMPLETE` = 4 , `DAC_ERROR` = 5 }
Driver state machine possible states.
- enum `dacerror_t` { `DAC_ERR_DMAFAILURE` = 0 , `DAC_ERR_UNDERFLOW` = 1 }
Possible DAC failure causes.

Functions

- void `daclnit` (void)
DAC Driver initialization.
- void `dacObjectInit` (DACDriver *daccp)
Initializes the standard part of a `DACDriver` structure.
- `msg_t` `dacStart` (DACDriver *daccp, const DACConfig *config)
Configures and activates the DAC peripheral.
- void `dacStop` (DACDriver *daccp)
Deactivates the DAC peripheral.
- `msg_t` `dacPutChannelX` (DACDriver *daccp, `dacchannel_t` channel, `dacsample_t` sample)
Outputs a value directly on a DAC channel.
- `msg_t` `dacStartConversion` (DACDriver *daccp, const DACConversionGroup *grpp, `dacsample_t` *samples, `size_t` depth)
Starts a DAC conversion.
- `msg_t` `dacStartConversionI` (DACDriver *daccp, const DACConversionGroup *grpp, `dacsample_t` *samples, `size_t` depth)
Starts a DAC conversion.
- void `dacStopConversion` (DACDriver *daccp)
Stops an ongoing conversion.
- void `dacStopConversionI` (DACDriver *daccp)
Stops an ongoing conversion.
- `msg_t` `dacConvert` (DACDriver *daccp, const DACConversionGroup *grpp, `dacsample_t` *samples, `size_t` depth)
Performs a DAC conversion.
- `msg_t` `dacSynchronizeS` (DACDriver *daccp, `sysinterval_t` timeout)
Synchronize to a conversion completion.
- `msg_t` `dacSynchronize` (DACDriver *daccp, `sysinterval_t` timeout)
Synchronize to a conversion completion.
- void `dacAcquireBus` (DACDriver *daccp)
Gains exclusive access to the DAC bus.
- void `dacReleaseBus` (DACDriver *daccp)
Releases exclusive access to the DAC bus.
- void `dac_lld_init` (void)
Low level DAC driver initialization.
- void `dac_lld_start` (DACDriver *daccp)
Configures and activates the DAC peripheral.
- void `dac_lld_stop` (DACDriver *daccp)
Deactivates the DAC peripheral.
- void `dac_lld_put_channel` (DACDriver *daccp, `dacchannel_t` channel, `dacsample_t` sample)
Outputs a value directly on a DAC channel.
- void `dac_lld_start_conversion` (DACDriver *daccp)
Starts a DAC conversion.
- void `dac_lld_stop_conversion` (DACDriver *daccp)
Stops an ongoing conversion.

Variables

- `DACDriver` `DACD1`
DAC1 driver identifier.

7.2.9.6.2 Macro Definition Documentation

7.2.9.6.2.1 DAC_USE_SYNCHRONIZATION

```
#define DAC_USE_SYNCHRONIZATION FALSE
```

Support for thread synchronization API.

7.2.9.6.2.2 DAC_USE_MUTUAL_EXCLUSION

```
#define DAC_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `dacAcquireBus()` and `dacReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.9.6.2.3 `dacIsBufferComplete`

```
#define dacIsBufferComplete(  
    dacp)
```

Value:

```
((bool)((dacp)->state == DAC_COMPLETE))
```

Buffer state.

Note

This function is meant to be called from the DAC callback only.

Parameters

<i>in</i>	<i>dacp</i>	pointer to the <code>DACDriver</code> object

Returns

The buffer state.

Return values

<i>false</i>	if the driver filled/sent the first half of the buffer.
<i>true</i>	if the driver filled/sent the second half of the buffer.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.6.2.4 `_dac_wait_s`

```
#define _dac_wait_s(  
    dacp)
```

Value:

```
osalThreadSuspendS(&(dacp)->thread)
```

Waits for operation completion.

This function waits for the driver to complete the current operation.

Precondition

An operation must be running while the function is invoked.

Note

No more than one thread can wait on a DAC driver using this function.

Parameters

in	<i>dacp</i>	pointer to the <code>DACDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.2.5 `_dac_reset_i`

```
#define _dac_reset_i(  
    dacp)
```

Value:

```
osalThreadResumeI(&(dacp)->thread, MSG_RESET)
```

Resumes a thread waiting for a conversion completion.

Parameters

in	<i>dacp</i>	pointer to the <code>DACDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.2.6 `_dac_reset_s`

```
#define _dac_reset_s(  
    dacp)
```

Value:

```
osalThreadResumeS(&(dacp)->thread, MSG_RESET)
```

Resumes a thread waiting for a conversion completion.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.2.7 _dac_wakeup_isr

```
#define _dac_wakeup_isr(  
    dacp)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI (& (dacp)->thread, MSG_OK);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.2.8 _dac_timeout_isr

```
#define _dac_timeout_isr(  
    dacp)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI (& (dacp)->thread, MSG_TIMEOUT);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread with a timeout message.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.2.9 `_dac_isr_half_code`

```
#define _dac_isr_half_code(  
    dacp)
```

Value:

```
{  
    if ((dacp)->grpp->end_cb != NULL) {  
        (dacp)->grpp->end_cb(dacp);  
    }  
}
```

Common ISR code, half buffer event.

This code handles the portable part of the ISR code:

- Callback invocation.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.2.10 `_dac_isr_full_code`

```
#define _dac_isr_full_code(  
    dacp)
```

Value:

```
{  
    if ((dacp)->grpp->end_cb) {  
        (dacp)->state = DAC\_COMPLETE;  
        (dacp)->grpp->end_cb(dacp);  
        if ((dacp)->state == DAC\_COMPLETE)  
            (dacp)->state = DAC\_ACTIVE;  
    }  
    _dac_wakeup_isr(dacp);  
}
```

Common ISR code, full buffer event.

This code handles the portable part of the ISR code:

- Callback invocation.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.2.11 _dac_isr_error_code

```
#define _dac_isr_error_code(
    dacp,
    err)
```

Value:

```
{
    dac_llc_stop_conversion(dacp);
    if ((dacp)->grp->error_cb != NULL) {
        (dacp)->state = DAC_ERROR;
        (dacp)->grp->error_cb(dacp, err);
        if ((dacp)->state == DAC_ERROR)
            (dacp)->state = DAC_READY;
    }
    (dacp)->grp = NULL;
    _dac_timeout_isr(dacp);
}
```

Common ISR code, error event.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread timeout signalling, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object
in	<i>err</i>	platform dependent error code

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.2.12 DAC_MAX_CHANNELS

```
#define DAC_MAX_CHANNELS 2
```

Maximum number of DAC channels per unit.

7.2.9.6.2.13 PLATFORM_DAC_USE_DAC1

```
#define PLATFORM_DAC_USE_DAC1 FALSE
```

DAC1 CH1 driver enable switch.

If set to TRUE the support for DAC1 channel 1 is included.

Note

The default is FALSE.

7.2.9.6.2.14 dac_lld_driver_fields

```
#define dac_lld_driver_fields
```

Value:

```
/* Dummy field, it is not needed.*/
uint32_t dummy;
```

Low level fields of the DAC driver structure.

7.2.9.6.2.15 dac_lld_config_fields

```
#define dac_lld_config_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/
uint32_t dummy;
```

Low level fields of the DAC configuration structure.

7.2.9.6.2.16 dac_lld_conversion_group_fields

```
#define dac_lld_conversion_group_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/
uint32_t dummy;
```

Low level fields of the DAC group configuration structure.

7.2.9.6.3 Typedef Documentation

7.2.9.6.3.1 DACDriver

```
typedef struct hal_dac_driver DACDriver
```

Type of a structure representing an DAC driver.

7.2.9.6.3.2 DACConfig

```
typedef struct hal_dac_config DACConfig
```

Type of a structure representing an DAC driver configuration.

7.2.9.6.3.3 DACConversionGroup

```
typedef struct hal_dac_conversion_group DACConversionGroup
```

Type of a DAC conversion group.

7.2.9.6.3.4 daccallback_t

```
typedef void(* daccallback_t) (DACDriver *daccp)
```

DAC notification callback type.

Parameters

in	<i>daccp</i>	pointer to the DACDriver object triggering the

7.2.9.6.3.5 dacerrorcallback_t

```
typedef void(* dacerrorcallback_t) (DACDriver *daccp, dacerror_t err)
```

DAC error callback type.

Parameters

in	<i>daccp</i>	pointer to the DACDriver object triggering the callback
in	<i>err</i>	DAC error code

7.2.9.6.3.6 dacchannel_t

```
typedef uint32_t dacchannel_t
```

Type of a DAC channel index.

7.2.9.6.3.7 dacsample_t

```
typedef uint16_t dacsample_t
```

Type representing a DAC sample.

7.2.9.6.4 Enumeration Type Documentation**7.2.9.6.4.1 dacstate_t**

```
enum dacstate_t
```

Driver state machine possible states.

Enumerator

DAC_UNINIT	Not initialized.
DAC_STOP	Stopped.
DAC_READY	Ready.
DAC_ACTIVE	Exchanging data.
DAC_COMPLETE	Asynchronous operation complete.
DAC_ERROR	Error.

7.2.9.6.4.2 dacerror_t

```
enum dacerror_t
```

Possible DAC failure causes.

Note

Error codes are architecture dependent and should not relied upon.

Enumerator

DAC_ERR_DMAFAILURE	DMA operations failure.
DAC_ERR_UNDERFLOW	DAC overflow condition.

7.2.9.6.5 Function Documentation**7.2.9.6.5.1 dacInit()**

```
void dacInit (  
    void )
```

DAC Driver initialization.

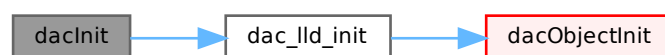
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.6.5.2 dacObjectInit()

```
void dacObjectInit (  
    DACDriver * dacp)
```

Initializes the standard part of a `DACDriver` structure.

Parameters

out	<i>dacp</i>	pointer to the <code>DACDriver</code> object

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.6.5.3 dacStart()

```
msg_t dacStart (  
    DACDriver * dacp,  
    const DACConfig * config)
```

Configures and activates the DAC peripheral.

Parameters

in	<i>dacp</i>	pointer to the <code>DACDriver</code> object
in	<i>config</i>	pointer to the <code>DACConfig</code> object, it can be <code>NULL</code> if the low level driver implementation supports a default configuration

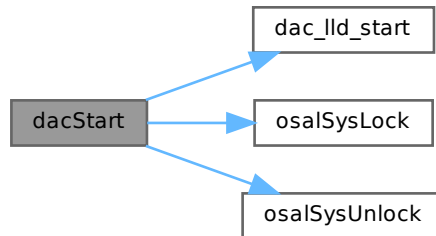
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.6.5.4 dacStop()**

```
void dacStop (  
    DACDriver * daccp)
```

Deactivates the DAC peripheral.

Note

Deactivating the peripheral also enforces a release of the slave select line.

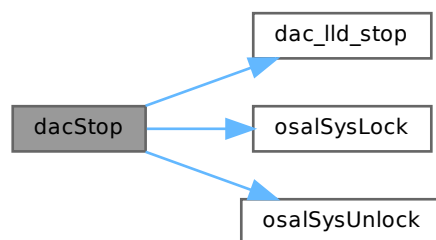
Parameters

in	<i>daccp</i>	pointer to the <code>DACDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.6.5.5 dacPutChannelX()

```
msg_t dacPutChannelX (
    DACDriver * dacp,
    dacchannel_t channel,
    dacsample_t sample)
```

Outputs a value directly on a DAC channel.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object
in	<i>channel</i>	DAC channel number
in	<i>sample</i>	value to be output

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

Here is the call graph for this function:



7.2.9.6.5.6 dacStartConversion()

```
msg_t dacStartConversion (
    DACDriver * dacp,
    const DACConversionGroup * grpp,
    dacsample_t * samples,
    size_t depth)
```

Starts a DAC conversion.

Starts an asynchronous conversion operation.

Note

The buffer is organized as a matrix of M*N elements where M is the channels number configured into the conversion group and N is the buffer depth. The samples are sequentially written into the buffer with no gaps.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object
in	<i>grpp</i>	pointer to a DACConversionGroup object
in	<i>samples</i>	pointer to the samples buffer
in	<i>depth</i>	buffer depth (matrix rows number). The buffer depth must be one or an even number.

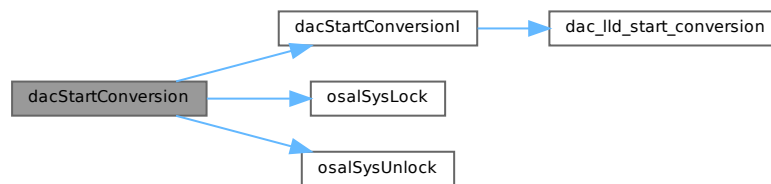
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.6.5.7 dacStartConversionI()**

```

msg_t dacStartConversionI (
    DACDriver * daccp,
    const DACConversionGroup * grpp,
    dacsample_t * samples,
    size_t depth)
  
```

Starts a DAC conversion.

Starts an asynchronous conversion operation.

Postcondition

The callbacks associated to the conversion group will be invoked on buffer complete and error events.

Note

The buffer is organized as a matrix of M*N elements where M is the channels number configured into the conversion group and N is the buffer depth. The samples are sequentially organised in the buffer with no gaps.

Parameters

in	<i>daccp</i>	pointer to the DACDriver object
in	<i>grpp</i>	pointer to a DACConversionGroup object
in	<i>samples</i>	pointer to the samples buffer
in	<i>depth</i>	buffer depth (matrix rows number). The buffer depth must be one or an even number.

Returns

The operation status.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.6.5.8 dacStopConversion()**

```
void dacStopConversion (
    DACDriver * dacp)
```

Stops an ongoing conversion.

This function stops the currently ongoing conversion and returns the driver in the `DAC_READY` state. If there was no conversion being processed then the function does nothing.

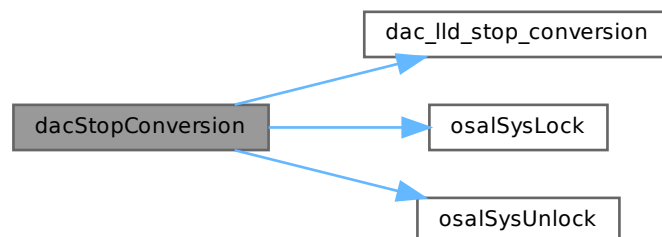
Parameters

in	<i>dacp</i>	pointer to the <code>DACDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.6.5.9 dacStopConversionI()

```
void dacStopConversionI (
    DACDriver * daccp)
```

Stops an ongoing conversion.

This function stops the currently ongoing conversion and returns the driver in the `DAC_READY` state. If there was no conversion being processed then the function does nothing.

Parameters

in	<i>daccp</i>	pointer to the <code>DACDriver</code> object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.6.5.10 dacConvert()

```
msg_t dacConvert (
    DACDriver * daccp,
    const DACConversionGroup * grpp,
    dacsamples_t * samples,
    size_t depth)
```

Performs a DAC conversion.

Performs a synchronous conversion operation.

Note

The buffer is organized as a matrix of M*N elements where M is the channels number configured into the conversion group and N is the buffer depth. The samples are sequentially organised in the buffer with no gaps.

Parameters

in	<i>daccp</i>	pointer to the <code>DACDriver</code> object
in	<i>grpp</i>	pointer to a <code>DACConversionGroup</code> object
in	<i>samples</i>	pointer to the samples buffer
in	<i>depth</i>	buffer depth (matrix rows number). The buffer depth must be one or an even number.

Returns

The operation result.

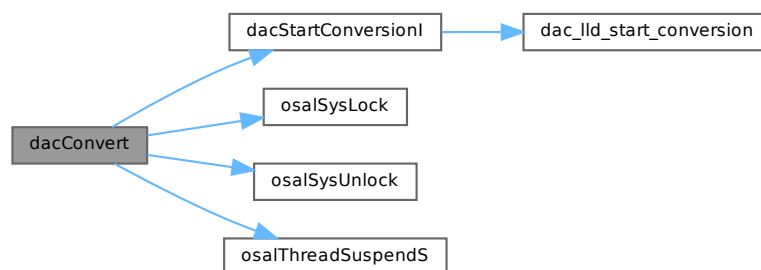
Return values

<i>MSG_OK</i>	Conversion finished.
<i>MSG_RESET</i>	The conversion has been stopped using <code>dacStopConversion()</code> or <code>dacStopConversionI()</code> .
<i>MSG_TIMEOUT</i>	The conversion has been stopped because an hardware error.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

7.2.9.6.5.11 `dacSynchronizeS()`

```

msg_t dacSynchronizeS (
    DACDriver * dacp,
    sysinterval_t timeout)

```

Synchronize to a conversion completion.

Note

This function can only be called by a single thread at time.

Parameters

in	<i>dacp</i>	pointer to the <code>DACDriver</code> object
in	<i>timeout</i>	wait timeout

Returns

The wait result.

Return values

<i>MSG_OK</i>	if operation completed without errors.
<i>MSG_TIMEOUT</i>	if synchronization request timed out.
<i>MSG_RESET</i>	if the conversion has been stopped.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

Here is the call graph for this function:



7.2.9.6.5.12 dacSynchronize()

```

msg_t dacSynchronize (
    DACDriver * dacp,
    sysinterval_t timeout)
  
```

Synchronize to a conversion completion.

Note

This function can only be called by a single thread at time.

Parameters

in	<i>dacp</i>	pointer to the <code>DACDriver</code> object
in	<i>timeout</i>	wait timeout

Returns

The wait result.

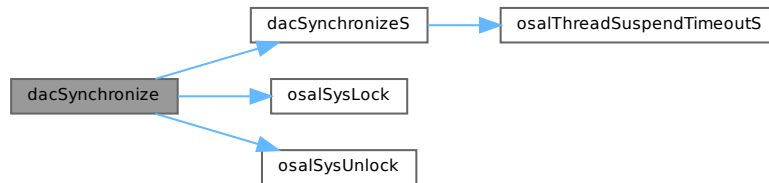
Return values

<i>MSG_OK</i>	if operation completed without errors.
<i>MSG_TIMEOUT</i>	if synchronization request timed out.
<i>MSG_RESET</i>	if the conversion has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.6.5.13 dacAcquireBus()**

```
void dacAcquireBus (
    DACDriver * dacp)
```

Gains exclusive access to the DAC bus.

This function tries to gain ownership to the DAC bus, if the bus is already being used then the invoking thread is queued.

Precondition

In order to use this function the option `DAC_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>dacp</i>	pointer to the <code>DACDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.6.5.14 dacReleaseBus()

```
void dacReleaseBus (  
    DACDriver * dacp)
```

Releases exclusive access to the DAC bus.

Precondition

In order to use this function the option `DAC_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>dacp</i>	pointer to the <code>DACDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.6.5.15 dac_lld_init()

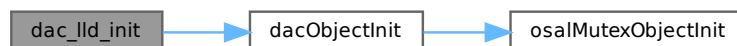
```
void dac_lld_init (  
    void )
```

Low level DAC driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.6.5.16 dac_lld_start()

```
void dac_lld_start (  
    DACDriver * dacp)
```

Configures and activates the DAC peripheral.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.5.17 dac_lld_stop()

```
void dac_lld_stop (
    DACDriver * dacp)
```

Deactivates the DAC peripheral.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.5.18 dac_lld_put_channel()

```
void dac_lld_put_channel (
    DACDriver * dacp,
    dacchannel_t channel,
    dacsample_t sample)
```

Outputs a value directly on a DAC channel.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object
in	<i>channel</i>	DAC channel number
in	<i>sample</i>	value to be output

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.6.5.19 dac_lld_start_conversion()

```
void dac_lld_start_conversion (
    DACDriver * dacp)
```

Starts a DAC conversion.

Starts an asynchronous conversion operation.

Note

In `DAC_DHRM_8BIT_RIGHT` mode the parameters passed to the callback are wrong because two samples are packed in a single [dacsample_t](#) element. This will not be corrected, do not rely on those parameters.

In `DAC_DHRM_8BIT_RIGHT_DUAL` mode two samples are treated as a single 16 bits sample and packed into a single [dacsample_t](#) element. The `num_channels` must be set to one in the group conversion configuration structure.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.6.5.20 dac_lld_stop_conversion()

```
void dac_lld_stop_conversion (  
    DACDriver * dacp)
```

Stops an ongoing conversion.

This function stops the currently ongoing conversion and returns the driver in the `DAC_READY` state. If there was no conversion being processed then the function does nothing.

Parameters

in	<i>dacp</i>	pointer to the DACDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.6.6 Variable Documentation**7.2.9.6.6.1 DACD1**

```
DACDriver DACD1
```

DAC1 driver identifier.

7.2.9.7 EFL Driver

Generic Embedded Flash Driver.

7.2.9.7.1 Detailed Description

Generic Embedded Flash Driver.

This module implements a generic embedded flash driver.

Precondition

In order to use the EFL driver the `HAL_USE_EFL` option must be enabled in [halconf.h](#).

PLATFORM configuration options

- #define `PLATFORM_EFL_USE_EFL1 FALSE`
EFL1 driver enable switch.

Data Structures

- struct `EFlashDriverVMT`
EFlash virtual methods table.
- struct `hal_efl_config`
Type of a structure representing a flash driver configuration.
- struct `hal_efl_driver`
Structure representing an embedded flash driver.

Macros

- #define `EFL_USE_MUTUAL_EXCLUSION TRUE`
Enables the `eflAcquireUse()` and `eflReleaseUse()` APIs.
- #define `_efl_flash_methods_alone`
EFlashDriver specific methods.
- #define `_efl_flash_methods`
EFlashDriver specific methods with inherited ones.
- #define `_efl_driver_data`
EFlashDriver specific data.
- #define `efl_lld_driver_fields`
Low level fields of the embedded flash driver structure.
- #define `efl_lld_config_fields`
Low level fields of the embedded flash configuration structure.

Typedefs

- typedef struct `hal_efl_driver` `EFlashDriver`
Type of external flash driver class.
- typedef struct `hal_efl_config` `EFlashConfig`
Type of a structure representing a flash driver configuration.

Functions

- static `flash_error_t efl_acquire_exclusive` (void *instance)
- static `flash_error_t efl_release_exclusive` (void *instance)
- void `eflInit` (void)
Embedded Flash Driver initialization.
- void `eflObjectInit` (`EFlashDriver` *eflp)
Initializes a generic EFlashDriver object.
- `msg_t eflStart` (`EFlashDriver` *eflp, const `EFlashConfig` *config)
Configures and starts the driver.
- void `eflStop` (`EFlashDriver` *eflp)
Stops the driver.
- void `efl_lld_init` (void)

- Low level Embedded Flash driver initialization.*
- void `efl_lld_start` (`EFlashDriver *eflp`)
Configures and activates the Embedded Flash peripheral.
- void `efl_lld_stop` (`EFlashDriver *eflp`)
Deactivates the Embedded Flash peripheral.
- const `flash_descriptor_t * efl_lld_get_descriptor` (void *instance)
Gets the flash descriptor structure.
- `flash_error_t efl_lld_read` (void *instance, `flash_offset_t` offset, `size_t` n, `uint8_t *rp`)
Read operation.
- `flash_error_t efl_lld_program` (void *instance, `flash_offset_t` offset, `size_t` n, const `uint8_t *pp`)
Program operation.
- `flash_error_t efl_lld_start_erase_all` (void *instance)
Starts a whole-device erase operation.
- `flash_error_t efl_lld_start_erase_sector` (void *instance, `flash_sector_t` sector)
Starts an sector erase operation.
- `flash_error_t efl_lld_query_erase` (void *instance, `uint32_t *msec`)
Queries the driver for erase operation progress.
- `flash_error_t efl_lld_verify_erase` (void *instance, `flash_sector_t` sector)
Returns the erase state of a sector.

Variables

- static const struct `EFlashDriverVMT vmt`
- `EFlashDriver EFLD1`
EFL1 driver identifier.
- static const `flash_descriptor_t efl_lld_descriptor`

7.2.9.7.2 Macro Definition Documentation

7.2.9.7.2.1 EFL_USE_MUTUAL_EXCLUSION

```
#define EFL_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `eflAcquireUse()` and `eflReleaseUse()` APIs.

Note

Disabling this option saves both code and data space.

7.2.9.7.2.2 _efl_flash_methods_alone

```
#define _efl_flash_methods_alone
```

`EFlashDriver` specific methods.

7.2.9.7.2.3 _efl_flash_methods

```
#define _efl_flash_methods
```

Value:

```
_base_flash_methods
_efl_flash_methods_alone
```

[EFlashDriver](#) specific methods with inherited ones.

7.2.9.7.2.4 _efl_driver_data

```
#define _efl_driver_data
```

Value:

```
_base_flash_data
/* Current configuration data.*/
const EFlashConfig *config;
```

[EFlashDriver](#) specific data.

7.2.9.7.2.5 PLATFORM_EFL_USE_EFL1

```
#define PLATFORM_EFL_USE_EFL1 FALSE
```

EFL1 driver enable switch.

If set to TRUE the support for EFL1 is included.

Note

The default is FALSE.

7.2.9.7.2.6 efl_lld_driver_fields

```
#define efl_lld_driver_fields
```

Value:

```
/* Dummy field, it is not needed.*/
uint32_t dummy;
```

Low level fields of the embedded flash driver structure.

7.2.9.7.2.7 efl_lld_config_fields

```
#define efl_lld_config_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/
uint32_t dummy;
```

Low level fields of the embedded flash configuration structure.

7.2.9.7.3 Typedef Documentation

7.2.9.7.3.1 EFlashDriver

```
typedef struct hal_efl_driver EFlashDriver
```

Type of external flash driver class.

7.2.9.7.3.2 EFlashConfig

```
typedef struct hal_efl_config EFlashConfig
```

Type of a structure representing a flash driver configuration.

7.2.9.7.4 Function Documentation

7.2.9.7.4.1 efl_acquire_exclusive()

```
flash_error_t efl_acquire_exclusive (  
    void * instance) [static]
```

Here is the call graph for this function:



7.2.9.7.4.2 efl_release_exclusive()

```
flash_error_t efl_release_exclusive (  
    void * instance) [static]
```

Here is the call graph for this function:



7.2.9.7.4.3 eflInit()

```
void eflInit (
    void )
```

Embedded Flash Driver initialization.

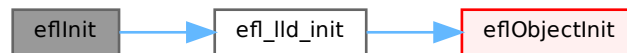
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.7.4.4 eflObjectInit()

```
void eflObjectInit (
    EFlashDriver * eflp)
```

Initializes a generic `EFlashDriver` object.

Parameters

out	<i>eflp</i>	pointer to a <code>EFlashDriver</code> structure

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.7.4.5 eflStart()

```
msg_t eflStart (  
    EFlashDriver * eflp,  
    const EFlashConfig * config)
```

Configures and starts the driver.

Parameters

in	<i>eflp</i>	pointer to a EFlashDriver structure
in	<i>config</i>	pointer to a configuration structure. If this parameter is set to <code>NULL</code> then a default configuration is used.

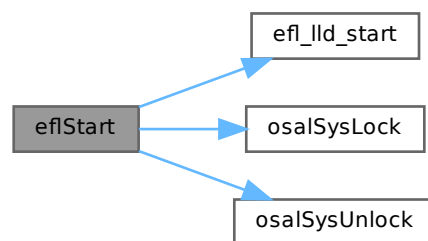
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.7.4.6 eflStop()

```
void eflStop (  
    EFlashDriver * eflp)
```

Stops the driver.

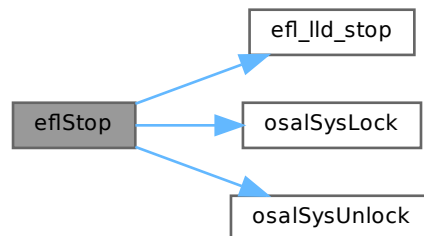
Parameters

in	<i>eflp</i>	pointer to a EFlashDriver structure

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.7.4.7 efl_ild_init()**

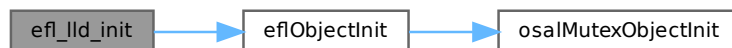
```
void efl_ild_init (  
    void )
```

Low level Embedded Flash driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.7.4.8 efl_ild_start()**

```
void efl_ild_start (  
    EFlashDriver * eflp)
```

Configures and activates the Embedded Flash peripheral.

Parameters

in	<i>eflp</i>	pointer to a EFlashDriver structure

Function Class:

Not an API, this function is for internal use only.

7.2.9.7.4.9 efl_llt_stop()

```
void efl_llt_stop (  
    EFlashDriver * eflp)
```

Deactivates the Embedded Flash peripheral.

Parameters

in	<i>eflp</i>	pointer to a EFlashDriver structure

Function Class:

Not an API, this function is for internal use only.

7.2.9.7.4.10 efl_llt_get_descriptor()

```
const flash_descriptor_t * efl_llt_get_descriptor (  
    void * instance)
```

Gets the flash descriptor structure.

Parameters

in	<i>instance</i>	pointer to a EFlashDriver instance

Returns

A flash device descriptor.

Function Class:

Not an API, this function is for internal use only.

7.2.9.7.4.11 efl_llt_read()

```
flash_error_t efl_llt_read (  
    void * instance,  
    flash_offset_t offset,  
    size_t n,  
    uint8_t * rp)
```

Read operation.

Parameters

in	<i>instance</i>	pointer to a EFlashDriver instance
in	<i>offset</i>	flash offset
in	<i>n</i>	number of bytes to be read
out	<i>rp</i>	pointer to the data buffer

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_READ</i>	if the read operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.7.4.12 `efl_lld_program()`

```
flash_error_t efl_lld_program (
    void * instance,
    flash_offset_t offset,
    size_t n,
    const uint8_t * pp)
```

Program operation.

Note

The device supports ECC, it is only possible to write erased pages once except when writing all zeroes.

Parameters

in	<i>instance</i>	pointer to a EFlashDriver instance
in	<i>offset</i>	flash offset
in	<i>n</i>	number of bytes to be programmed
in	<i>pp</i>	pointer to the data buffer

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_PROGRAM</i>	if the program operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.7.4.13 efl_lld_start_erase_all()

```
flash_error_t efl_lld_start_erase_all (
    void * instance)
```

Starts a whole-device erase operation.

Note

This function only erases bank 2 if it is present. Bank 1 is not touched because it is where the program is running on. Pages on bank 1 can be individually erased.

Parameters

in	<i>instance</i>	pointer to a EFlashDriver instance

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.7.4.14 efl_lld_start_erase_sector()

```
flash_error_t efl_lld_start_erase_sector (
    void * instance,
    flash_sector_t sector)
```

Starts an sector erase operation.

Parameters

in	<i>instance</i>	pointer to a EFlashDriver instance
in	<i>sector</i>	sector to be erased

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.7.4.15 `efl_lld_query_erase()`

```
flash_error_t efl_lld_query_erase (
    void * instance,
    uint32_t * msec)
```

Queries the driver for erase operation progress.

Parameters

in	<i>instance</i>	pointer to a EFlashDriver instance
out	<i>msec</i>	recommended time, in milliseconds, that should be spent before calling this function again, can be NULL

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_ERASE</i>	if the erase operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.7.4.16 `efl_lld_verify_erase()`

```
flash_error_t efl_lld_verify_erase (
    void * instance,
    flash_sector_t sector)
```

Returns the erase state of a sector.

Parameters

in	<i>instance</i>	pointer to a EFlashDriver instance
in	<i>sector</i>	sector to be verified

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	if the sector is erased.
<i>FLASH_BUSY_ERASING</i>	if there is an erase operation in progress.
<i>FLASH_ERROR_VERIFY</i>	if the verify operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.7.5 Variable Documentation**7.2.9.7.5.1 vmt**

```
const struct EFlashDriverVMT vmt [static]
```

Initial value:

```
= {
    (size_t)0,
    efl_lld_get_descriptor,
    efl_lld_read,
    efl_lld_program,
    efl_lld_start_erase_all,
    efl_lld_start_erase_sector,
    efl_lld_query_erase,
    efl_lld_verify_erase,
    efl_acquire_exclusive,
    efl_release_exclusive
}
```

7.2.9.7.5.2 EFLD1

```
EFlashDriver EFLD1
```

EFL1 driver identifier.

7.2.9.7.5.3 efl_lld_descriptor

```
const flash\_descriptor\_t efl_lld_descriptor [static]
```

Initial value:

```
= {
    .attributes          = FLASH_ATTR_ERASED_IS_ONE |
                          FLASH_ATTR_MEMORY_MAPPED |
                          FLASH_ATTR_ECC_CAPABLE   |
                          FLASH_ATTR_ECC_ZERO_LINE_CAPABLE,
    .page_size           = 0,
    .sectors_count       = 0,
    .sectors             = NULL,
    .sectors_size        = 0,
    .address             = (uint8_t *)0,
    .size                = 0
}
```


7.2.9.8 GPT Driver

Generic GPT Driver.

7.2.9.8.1 Detailed Description

Generic GPT Driver.

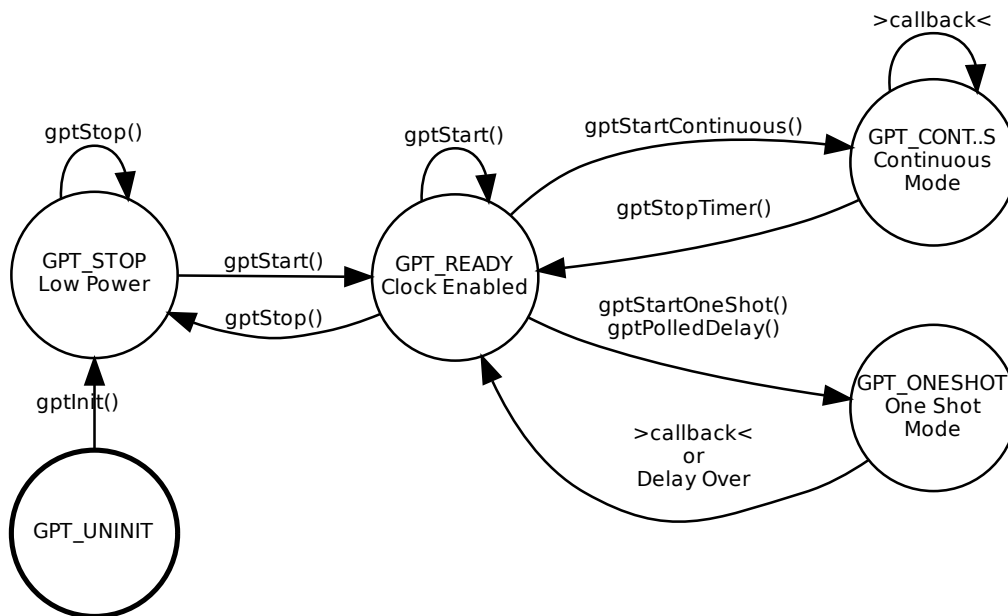
This module implements a generic GPT (General Purpose Timer) driver. The timer can be programmed in order to trigger callbacks after a specified time period or continuously with a specified interval.

Precondition

In order to use the GPT driver the `HAL_USE_GPT` option must be enabled in `halconf.h`.

7.2.9.8.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



7.2.9.8.3 GPT Operations.

This driver abstracts a generic timer composed of:

- A clock prescaler.
- A main up counter.
- A comparator register that resets the main counter to zero when the limit is reached. A callback is invoked when this happens.

The timer can operate in three different modes:

- **Continuous Mode**, a periodic callback is invoked until the driver is explicitly stopped.
- **One Shot Mode**, a callback is invoked after the programmed period and then the timer automatically stops.
- **Delay Mode**, the timer is used for inserting a brief delay into the execution flow, no callback is invoked in this mode.

PLATFORM configuration options

- #define `PLATFORM_GPT_USE_GPT1 FALSE`
GPTD1 driver enable switch.

Data Structures

- struct `GPTConfig`
Driver configuration structure.
- struct `GPTDriver`
Structure representing a GPT driver.

Macros

- #define `gptChangeIntervall(gptp, interval)`
Changes the interval of GPT peripheral.
- #define `gptGetIntervalX(gptp)`
Returns the interval of GPT peripheral.
- #define `gptGetCounterX(gptp)`
Returns the counter value of GPT peripheral.
- #define `_gpt_isr_invoke_cb(gptp)`
Common ISR code, GPT period event.
- #define `gpt_lld_change_interval(gptp, interval)`
Changes the interval of GPT peripheral.

Typedefs

- typedef struct GPTDriver [GPTDriver](#)
Type of a structure representing a GPT driver.
- typedef void(* [gptcallback_t](#)) ([GPTDriver](#) *gptp)
GPT notification callback type.
- typedef uint32_t [gptfreq_t](#)
GPT frequency type.
- typedef uint16_t [gptcnt_t](#)
GPT counter type.

Enumerations

- enum [gptstate_t](#) {
 [GPT_UNINIT](#) = 0 , [GPT_STOP](#) = 1 , [GPT_READY](#) = 2 , [GPT_CONTINUOUS](#) = 3 ,
 [GPT_ONESHOT](#) = 4 }
Driver state machine possible states.

Functions

- void [gptInit](#) (void)
GPT Driver initialization.
- void [gptObjectInit](#) ([GPTDriver](#) *gptp)
Initializes the standard part of a [GPTDriver](#) structure.
- [msg_t](#) [gptStart](#) ([GPTDriver](#) *gptp, const [GPTConfig](#) *config)
Configures and activates the GPT peripheral.
- void [gptStop](#) ([GPTDriver](#) *gptp)
Deactivates the GPT peripheral.
- void [gptChangeInterval](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Changes the interval of GPT peripheral.
- void [gptStartContinuous](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in continuous mode.
- void [gptStartContinuousl](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in continuous mode.
- void [gptStartOneShot](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in one shot mode.
- void [gptStartOneShotl](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in one shot mode.
- void [gptStopTimer](#) ([GPTDriver](#) *gptp)
Stops the timer.
- void [gptStopTimerl](#) ([GPTDriver](#) *gptp)
Stops the timer.
- void [gptPolledDelay](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in one shot mode and waits for completion.
- void [gpt_lld_init](#) (void)
Low level GPT driver initialization.
- void [gpt_lld_start](#) ([GPTDriver](#) *gptp)
Configures and activates the GPT peripheral.
- void [gpt_lld_stop](#) ([GPTDriver](#) *gptp)
Deactivates the GPT peripheral.

- void `gpt_lld_start_timer` (`GPTDriver *gtp`, `gptcnt_t` interval)
Starts the timer in continuous mode.
- void `gpt_lld_stop_timer` (`GPTDriver *gtp`)
Stops the timer.
- void `gpt_lld_polled_delay` (`GPTDriver *gtp`, `gptcnt_t` interval)
Starts the timer in one shot mode and waits for completion.

Variables

- `GPTDriver GPTD1`
GPTD1 driver identifier.

7.2.9.8.4 Macro Definition Documentation

7.2.9.8.4.1 gptChangeIntervall

```
#define gptChangeIntervalI(  
    gtp,  
    interval)
```

Value:

```
{  
    gpt_lld_change_interval(gtp, interval);  
}
```

Changes the interval of GPT peripheral.

This function changes the interval of a running GPT unit.

Precondition

The GPT unit must be running in continuous mode.

Postcondition

The GPT unit interval is changed to the new value.

Parameters

in	<i>gtp</i>	pointer to a <code>GPTDriver</code> object
in	<i>interval</i>	new cycle time in timer ticks

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.8.4.2 gptGetIntervalX

```
#define gptGetIntervalX(  
    gtp)
```

Value:

```
gpt_lld_get_interval(gtp)
```

Returns the interval of GPT peripheral.

Precondition

The GPT unit must be running in continuous mode.

Parameters

in	<i>gtp</i>	pointer to a GPTDriver object

Returns

The current interval.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.8.4.3 gptGetCounterX

```
#define gptGetCounterX(  
    gtp)
```

Value:

```
gpt_lld_get_counter(gtp)
```

Returns the counter value of GPT peripheral.

Precondition

The GPT unit must be running in continuous mode.

Note

The nature of the counter is not defined, it may count upward or downward, it could be continuously running or not.

Parameters

in	<i>gtp</i>	pointer to a GPTDriver object

Returns

The current counter value.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.8.4.4 _gpt_isr_invoke_cb

```
#define _gpt_isr_invoke_cb(  
    gtp)
```

Value:

```
do {  
    if ((gtp->state == GPT_ONESHOT) {  
        (gtp->state = GPT_READY;  
        gpt_lld_stop_timer(gtp);  
    }  
    if ((gtp->config->callback != NULL) {  
        (gtp->config->callback(gtp);  
    }  
} while (0)
```

Common ISR code, GPT period event.

Parameters

in	<i>gptp</i>	pointer to the GPTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.8.4.5 PLATFORM_GPT_USE_GPT1

```
#define PLATFORM_GPT_USE_GPT1 FALSE
```

GPTD1 driver enable switch.

If set to TRUE the support for GPTD1 is included.

Note

The default is FALSE.

7.2.9.8.4.6 gpt_lld_change_interval

```
#define gpt_lld_change_interval(  
    gptp,  
    interval)
```

Value:

```
{  
    (void)gptp;  
    (void)interval;  
}
```

Changes the interval of GPT peripheral.

This function changes the interval of a running GPT unit.

Precondition

The GPT unit must have been activated using [gptStart\(\)](#).

The GPT unit must have been running in continuous mode using [gptStartContinuous\(\)](#).

Postcondition

The GPT unit interval is changed to the new value.

Note

The function has effect at the next cycle start.

Parameters

in	<i>gtp</i>	pointer to a GPTDriver object
in	<i>interval</i>	new cycle time in timer ticks

Function Class:

Not an API, this function is for internal use only.

7.2.9.8.5 Typedef Documentation

7.2.9.8.5.1 GPTDriver

```
typedef struct GPTDriver GPTDriver
```

Type of a structure representing a GPT driver.

7.2.9.8.5.2 gptcallback_t

```
typedef void(* gptcallback_t) (GPTDriver *gtp)
```

GPT notification callback type.

Parameters

in	<i>gtp</i>	pointer to a GPTDriver object

7.2.9.8.5.3 gptfreq_t

```
typedef uint32_t gptfreq_t
```

GPT frequency type.

7.2.9.8.5.4 gptcnt_t

```
typedef uint16_t gptcnt_t
```

GPT counter type.

7.2.9.8.6 Enumeration Type Documentation

7.2.9.8.6.1 gptstate_t

```
enum gptstate_t
```

Driver state machine possible states.

Enumerator

GPT_UNINIT	Not initialized.
GPT_STOP	Stopped.
GPT_READY	Ready.
GPT_CONTINUOUS	Active in continuous mode.
GPT_ONESHOT	Active in one shot mode.

7.2.9.8.7 Function Documentation**7.2.9.8.7.1 gptInit()**

```
void gptInit (
    void )
```

GPT Driver initialization.

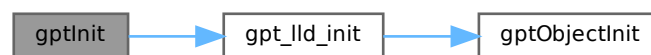
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.8.7.2 gptObjectInit()**

```
void gptObjectInit (
    GPTDriver * gtp)
```

Initializes the standard part of a `GPTDriver` structure.

Parameters

out	<i>gtp</i>	pointer to the <code>GPTDriver</code> object

Function Class:

Object or module initializer function.

7.2.9.8.7.3 gptStart()

```
msg_t gptStart (  
    GPTDriver * gtp,  
    const GPTConfig * config)
```

Configures and activates the GPT peripheral.

Parameters

in	<i>gtp</i>	pointer to the <code>GPTDriver</code> object
in	<i>config</i>	pointer to the <code>GPTConfig</code> object

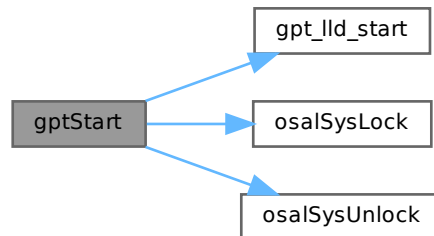
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.8.7.4 gptStop()

```
void gptStop (  
    GPTDriver * gtp)
```

Deactivates the GPT peripheral.

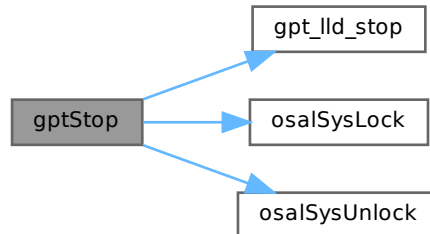
Parameters

in	<i>gtp</i>	pointer to the <code>GPTDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.8.7.5 gptChangeInterval()**

```
void gptChangeInterval (
    GPTDriver * gtp,
    gptcnt_t interval)
```

Changes the interval of GPT peripheral.

This function changes the interval of a running GPT unit.

Precondition

The GPT unit must be running in continuous mode.

Postcondition

The GPT unit interval is changed to the new value.

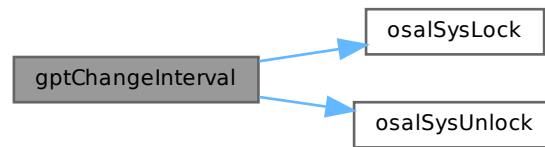
Parameters

in	<i>gtp</i>	pointer to a <code>GPTDriver</code> object
in	<i>interval</i>	new cycle time in timer ticks

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.8.7.6 gptStartContinuous()**

```
void gptStartContinuous (
    GPTDriver * gtp,
    gptcnt_t interval)
```

Starts the timer in continuous mode.

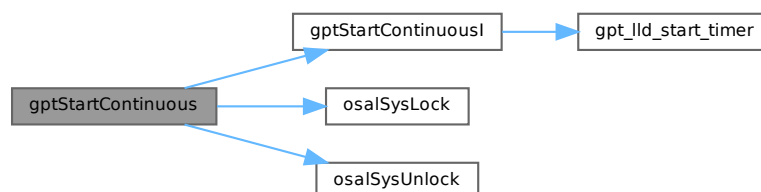
Parameters

in	<i>gtp</i>	pointer to the <code>GPTDriver</code> object
in	<i>interval</i>	period in ticks

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.8.7.7 gptStartContinuousI()**

```
void gptStartContinuousI (
    GPTDriver * gtp,
    gptcnt_t interval)
```

Starts the timer in continuous mode.

Parameters

in	<i>gtp</i>	pointer to the GPTDriver object
in	<i>interval</i>	period in ticks

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.8.7.8 gptStartOneShot()**

```

void gptStartOneShot (
    GPTDriver * gtp,
    gptcnt_t interval)
  
```

Starts the timer in one shot mode.

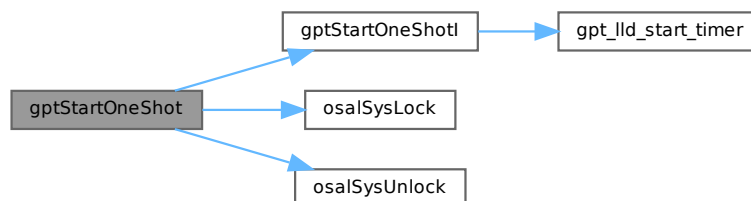
Parameters

in	<i>gtp</i>	pointer to the GPTDriver object
in	<i>interval</i>	time interval in ticks

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.8.7.9 gptStartOneShotI()

```
void gptStartOneShotI (  
    GPTDriver * gtp,  
    gptcnt_t interval)
```

Starts the timer in one shot mode.

Parameters

in	<i>gtp</i>	pointer to the <code>GPTDriver</code> object
in	<i>interval</i>	time interval in ticks

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.8.7.10 gptStopTimer()

```
void gptStopTimer (  
    GPTDriver * gtp)
```

Stops the timer.

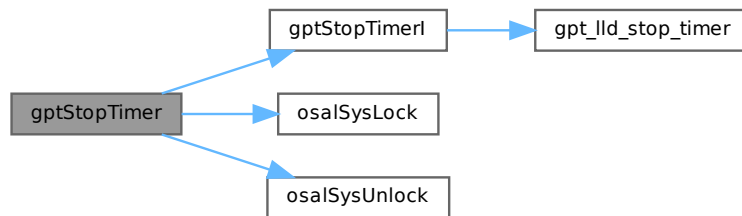
Parameters

in	<i>gtp</i>	pointer to the <code>GPTDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.8.7.11 gptStopTimerI()**

```
void gptStopTimerI (
    GPTDriver * gtp)
```

Stops the timer.

Parameters

in	<i>gtp</i>	pointer to the <code>GPTDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.8.7.12 gptPolledDelay()

```
void gptPolledDelay (  
    GPTDriver * gtp,  
    gptcnt_t interval)
```

Starts the timer in one shot mode and waits for completion.

This function specifically polls the timer waiting for completion in order to not have extra delays caused by interrupt servicing, this function is only recommended for short delays.

Note

The configured callback is not invoked when using this function.

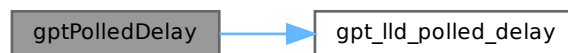
Parameters

in	<i>gtp</i>	pointer to the GPTDriver object
in	<i>interval</i>	time interval in ticks

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.8.7.13 gpt_lld_init()

```
void gpt_lld_init (  
    void )
```

Low level GPT driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.8.7.14 gpt_lld_start()

```
void gpt_lld_start (
    GPTDriver * gtp)
```

Configures and activates the GPT peripheral.

Parameters

in	<i>gtp</i>	pointer to the GPTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.8.7.15 gpt_lld_stop()

```
void gpt_lld_stop (
    GPTDriver * gtp)
```

Deactivates the GPT peripheral.

Parameters

in	<i>gtp</i>	pointer to the GPTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.8.7.16 gpt_lld_start_timer()

```
void gpt_lld_start_timer (
    GPTDriver * gtp,
    gptcnt_t interval)
```

Starts the timer in continuous mode.

Parameters

in	<i>gtp</i>	pointer to the GPTDriver object
in	<i>interval</i>	period in ticks

Function Class:

Not an API, this function is for internal use only.

7.2.9.8.7.17 gpt_lld_stop_timer()

```
void gpt_lld_stop_timer (
    GPTDriver * gtp)
```

Stops the timer.

Parameters

in	<i>gtp</i>	pointer to the GPTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.8.7.18 gpt_lld_polled_delay()

```
void gpt_lld_polled_delay (  
    GPTDriver * gtp,  
    gptcnt\_t interval)
```

Starts the timer in one shot mode and waits for completion.

This function specifically polls the timer waiting for completion in order to not have extra delays caused by interrupt servicing, this function is only recommended for short delays.

Parameters

in	<i>gtp</i>	pointer to the GPTDriver object
in	<i>interval</i>	time interval in ticks

Function Class:

Not an API, this function is for internal use only.

7.2.9.8.8 Variable Documentation**7.2.9.8.8.1 GPTD1**

```
GPTDriver GPTD1
```

GPTD1 driver identifier.

7.2.9.9 I2C Driver

Generic I2C Driver.

7.2.9.9.1 Detailed Description

Generic I2C Driver.

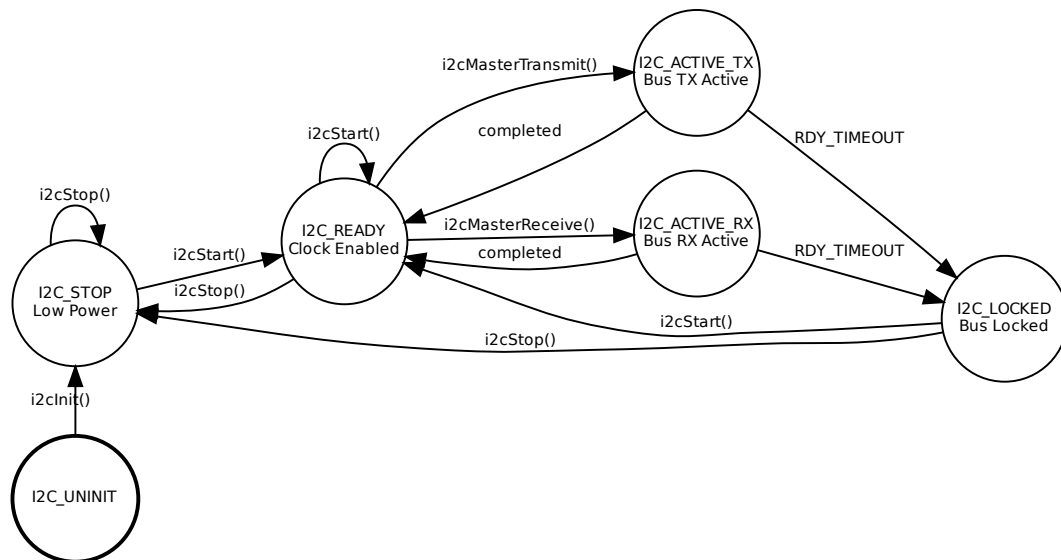
This module implements a generic I2C (Inter-Integrated Circuit) driver.

Precondition

In order to use the I2C driver the `HAL_USE_I2C` option must be enabled in [halconf.h](#).

7.2.9.9.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



The driver is not thread safe for performance reasons, if you need to access the I2C bus from multiple threads then use the `i2cAcquireBus()` and `i2cReleaseBus()` APIs in order to gain exclusive access.

I2C bus error conditions

- `#define I2C_NO_ERROR 0x00`
No error.
- `#define I2C_BUS_ERROR 0x01`
Bus Error.
- `#define I2C_ARBITRATION_LOST 0x02`
Arbitration Lost.
- `#define I2C_ACK_FAILURE 0x04`
Acknowledge Failure.
- `#define I2C_OVERRUN 0x08`
Overrun/Underrun.
- `#define I2C_PEC_ERROR 0x10`
PEC Error in reception.
- `#define I2C_TIMEOUT 0x20`
Hardware timeout.
- `#define I2C_SMB_ALERT 0x40`
SMBus Alert.

Macro Functions

- `#define _i2c_wakeup_isr(i2cp)`
Wakes up the waiting thread notifying no errors.
- `#define _i2c_wakeup_error_isr(i2cp)`
Wakes up the waiting thread notifying errors.
- `#define i2cMasterTransmit(i2cp, addr, txbuf, txbytes, rxbuf, rxbytes)`
Wrap i2cMasterTransmitTimeout function with TIME_INFINITE timeout.
- `#define i2cMasterReceive(i2cp, addr, rxbuf, rxbytes)`
Wrap i2cMasterReceiveTimeout function with TIME_INFINITE timeout.
- `#define i2cSlavesAnswerRequired(i2cp)`
Answer required.

PLATFORM configuration options

- `#define PLATFORM_I2C_USE_I2C1 FALSE`
I2C1 driver enable switch.

Data Structures

- struct `hal_i2c_config`
I2C driver configuration structure.
- struct `hal_i2c_driver`
Structure representing an I2C driver.

Macros

- `#define I2C_USE_MUTUAL_EXCLUSION TRUE`
Enables the mutual exclusion APIs on the I2C bus.
- `#define I2C_ENABLE_SLAVE_MODE FALSE`
Slave mode API enable switch.
- `#define I2C_SUPPORTS_SLAVE_MODE FALSE`
- `#define i2c_lld_get_errors(i2cp)`
Get errors from I2C driver.

Typedefs

- typedef uint16_t `i2caddr_t`
Type representing an I2C address.
- typedef uint32_t `i2cflags_t`
Type of I2C Driver condition flags.
- typedef struct `hal_i2c_config` `I2CConfig`
Type of a structure representing an I2C configuration.
- typedef struct `hal_i2c_driver` `I2CDriver`
Type of a structure representing an I2C driver.

Enumerations

- enum `i2cstate_t` {
`I2C_UNINIT` = 0 , `I2C_STOP` = 1 , `I2C_READY` = 2 , `I2C_ACTIVE_TX` = 3 ,
`I2C_ACTIVE_RX` = 4 , `I2C_LOCKED` = 5 }

Driver state machine possible states.

Functions

- void `i2cInit` (void)
I2C Driver initialization.
- void `i2cObjectInit` (I2CDriver *i2cp)
Initializes the standard part of a I2CDriver structure.
- msg_t `i2cStart` (I2CDriver *i2cp, const I2CConfig *config)
Configures and activates the I2C peripheral.
- void `i2cStop` (I2CDriver *i2cp)
Deactivates the I2C peripheral.
- i2cflags_t `i2cGetErrors` (I2CDriver *i2cp)
Returns the errors mask associated to the previous operation.
- msg_t `i2cMasterTransmitTimeout` (I2CDriver *i2cp, i2caddr_t addr, const uint8_t *txbuf, size_t txbytes, uint8_t *rxbuf, size_t rxbytes, sysinterval_t timeout)
Sends data via the I2C bus.
- msg_t `i2cMasterReceiveTimeout` (I2CDriver *i2cp, i2caddr_t addr, uint8_t *rxbuf, size_t rxbytes, sysinterval_t timeout)
Receives data from the I2C bus.
- void `i2cAcquireBus` (I2CDriver *i2cp)
Gains exclusive access to the I2C bus.
- void `i2cReleaseBus` (I2CDriver *i2cp)
Releases exclusive access to the I2C bus.
- msg_t `i2cSlaveMatchAddress` (I2CDriver *i2cp, i2caddr_t addr)
Listen I2C bus for address match.
- msg_t `i2cSlaveReceiveTimeout` (I2CDriver *i2cp, uint8_t *rxbuf, size_t rxbytes, sysinterval_t timeout)
Receive data via the I2C bus as slave and call handler.
- msg_t `i2cSlaveTransmitTimeout` (I2CDriver *i2cp, const uint8_t *txbuf, size_t txbytes, sysinterval_t timeout)
Transmits data via the I2C bus as slave.
- void `i2c_llc_init` (void)
Low level I2C driver initialization.
- void `i2c_llc_start` (I2CDriver *i2cp)
Configures and activates the I2C peripheral.
- void `i2c_llc_stop` (I2CDriver *i2cp)
Deactivates the I2C peripheral.
- msg_t `i2c_llc_master_receive_timeout` (I2CDriver *i2cp, i2caddr_t addr, uint8_t *rxbuf, size_t rxbytes, sysinterval_t timeout)
Receives data via the I2C bus as master.
- msg_t `i2c_llc_master_transmit_timeout` (I2CDriver *i2cp, i2caddr_t addr, const uint8_t *txbuf, size_t txbytes, uint8_t *rxbuf, size_t rxbytes, sysinterval_t timeout)
Transmits data via the I2C bus as master.

Variables

- I2CDriver `I2CD1`
I2C1 driver identifier.

7.2.9.9.3 Macro Definition Documentation

7.2.9.9.3.1 I2C_NO_ERROR

```
#define I2C_NO_ERROR 0x00
```

No error.

7.2.9.9.3.2 I2C_BUS_ERROR

```
#define I2C_BUS_ERROR 0x01
```

Bus Error.

7.2.9.9.3.3 I2C_ARBITRATION_LOST

```
#define I2C_ARBITRATION_LOST 0x02
```

Arbitration Lost.

7.2.9.9.3.4 I2C_ACK_FAILURE

```
#define I2C_ACK_FAILURE 0x04
```

Acknowledge Failure.

7.2.9.9.3.5 I2C_OVERRUN

```
#define I2C_OVERRUN 0x08
```

Overrun/Underrun.

7.2.9.9.3.6 I2C_PEC_ERROR

```
#define I2C_PEC_ERROR 0x10
```

PEC Error in reception.

7.2.9.9.3.7 I2C_TIMEOUT

```
#define I2C_TIMEOUT 0x20
```

Hardware timeout.

7.2.9.9.3.8 I2C_SMB_ALERT

```
#define I2C_SMB_ALERT 0x40
```

SMBus Alert.

7.2.9.9.3.9 I2C_USE_MUTUAL_EXCLUSION

```
#define I2C_USE_MUTUAL_EXCLUSION TRUE
```

Enables the mutual exclusion APIs on the I2C bus.

7.2.9.9.3.10 I2C_ENABLE_SLAVE_MODE

```
#define I2C_ENABLE_SLAVE_MODE FALSE
```

Slave mode API enable switch.

Note

The low level driver must support this capability.

7.2.9.9.3.11 I2C_SUPPORTS_SLAVE_MODE

```
#define I2C_SUPPORTS_SLAVE_MODE FALSE
```

7.2.9.9.3.12 _i2c_wakeup_isr

```
#define _i2c_wakeup_isr(  
    i2cp)
```

Value:

```
do {  
    osalSysLockFromISR();  
    osalThreadResumeI(&(i2cp)->thread, MSG_OK);  
    osalSysUnlockFromISR();  
} while (0)
```

Wakes up the waiting thread notifying no errors.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.9.3.13 _i2c_wakeup_error_isr

```
#define _i2c_wakeup_error_isr(  
    i2cp)
```

Value:

```
do {  
    osalSysLockFromISR();  
    osalThreadResumeI(&(i2cp)->thread, MSG_RESET);  
    osalSysUnlockFromISR();  
} while (0)
```

Wakes up the waiting thread notifying errors.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.9.3.14 i2cMasterTransmit

```
#define i2cMasterTransmit(
    i2cp,
    addr,
    txbuf,
    txbytes,
    rxbuf,
    rxbytes)
```

Value:

```
(i2cMasterTransmitTimeout(i2cp, addr, txbuf, txbytes, rxbuf, rxbytes, \
    TIME_INFINITE))
```

Wrap `i2cMasterTransmitTimeout` function with `TIME_INFINITE` timeout.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.9.3.15 i2cMasterReceive

```
#define i2cMasterReceive(
    i2cp,
    addr,
    rxbuf,
    rxbytes)
```

Value:

```
(i2cMasterReceiveTimeout(i2cp, addr, rxbuf, rxbytes, TIME_INFINITE))
```

Wrap `i2cMasterReceiveTimeout` function with `TIME_INFINITE` timeout.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.9.3.16 i2cSlaveIsAnswerRequired

```
#define i2cSlaveIsAnswerRequired(
    i2cp)
```

Value:

```
((i2cp)->reply_required)
```

Answer required.

Note

This function is meant to be called after slave receive only.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object

Returns

Slave answer required.

Return values

<i>false</i>	if the slave must not answer.
<i>true</i>	if the slave must answer.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.9.3.17 PLATFORM_I2C_USE_I2C1

```
#define PLATFORM_I2C_USE_I2C1 FALSE
```

I2C1 driver enable switch.

If set to `TRUE` the support for I2C1 is included.

Note

The default is `FALSE`.

7.2.9.9.3.18 i2c_lld_get_errors

```
#define i2c_lld_get_errors(  
    i2cp)
```

Value:

```
((i2cp)->errors)
```

Get errors from I2C driver.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.9.4 Typedef Documentation

7.2.9.9.4.1 i2caddr_t

```
typedef uint16_t i2caddr_t
```

Type representing an I2C address.

7.2.9.9.4.2 i2cflags_t

```
typedef uint32_t i2cflags_t
```

Type of I2C Driver condition flags.

7.2.9.9.4.3 I2CConfig

```
typedef struct hal_i2c_config I2CConfig
```

Type of a structure representing an I2C configuration.

7.2.9.9.4.4 I2CDriver

```
typedef struct hal_i2c_driver I2CDriver
```

Type of a structure representing an I2C driver.

7.2.9.9.5 Enumeration Type Documentation

7.2.9.9.5.1 i2cstate_t

```
enum i2cstate_t
```

Driver state machine possible states.

Enumerator

I2C_UNINIT	Not initialized.
I2C_STOP	Stopped.
I2C_READY	Ready.
I2C_ACTIVE_TX	Transmitting.
I2C_ACTIVE_RX	Receiving.
I2C_LOCKED	Bus locked.

7.2.9.9.6 Function Documentation

7.2.9.9.6.1 i2cInit()

```
void i2cInit (
    void )
```

I2C Driver initialization.

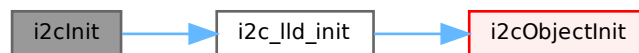
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.9.6.2 i2cObjectInit()

```
void i2cObjectInit (
    I2CDriver * i2cp)
```

Initializes the standard part of a `I2CDriver` structure.

Parameters

out	<i>i2cp</i>	pointer to the <code>I2CDriver</code> object

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.9.6.3 i2cStart()

```
msg_t i2cStart (  
    I2CDriver * i2cp,  
    const I2CConfig * config)
```

Configures and activates the I2C peripheral.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object
in	<i>config</i>	pointer to the I2CConfig object

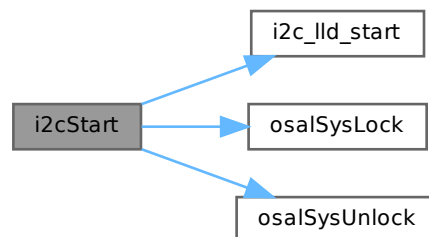
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.9.6.4 i2cStop()

```
void i2cStop (  
    I2CDriver * i2cp)
```

Deactivates the I2C peripheral.

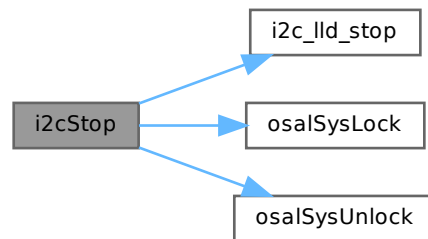
Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.9.6.5 i2cGetErrors()**

```
i2cflags_t i2cGetErrors (
    I2CDriver * i2cp)
```

Returns the errors mask associated to the previous operation.

Parameters

in	<i>i2cp</i>	pointer to the <code>I2CDriver</code> object

Returns

The errors mask.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.9.6.6 i2cMasterTransmitTimeout()

```
msg_t i2cMasterTransmitTimeout (
    I2CDriver * i2cp,
    i2caddr_t addr,
    const uint8_t * txbuf,
    size_t txbytes,
    uint8_t * rxbuf,
    size_t rxbytes,
    sysinterval_t timeout)
```

Sends data via the I2C bus.

Function designed to realize "read-through-write" transfer paradigm. If you want transmit data without any further read, than set **rxbytes** field to 0.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object
in	<i>addr</i>	slave device address (7 bits) without R/W bit
in	<i>txbuf</i>	pointer to transmit buffer
in	<i>txbytes</i>	number of bytes to be transmitted
out	<i>rxbuf</i>	pointer to receive buffer
in	<i>rxbytes</i>	number of bytes to be received, set it to 0 if you want transmit only
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

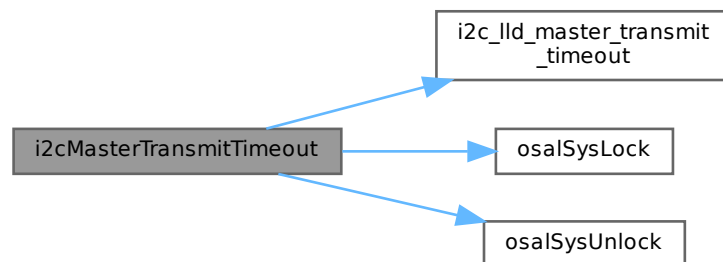
Return values

<i>MSG_OK</i>	if the function succeeded.
<i>MSG_RESET</i>	if one or more I2C errors occurred, the errors can be retrieved using i2cGetErrors() .
<i>MSG_TIMEOUT</i>	if a timeout occurred before operation end.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.9.6.7 i2cMasterReceiveTimeout()

```

msg_t i2cMasterReceiveTimeout (
    I2CDriver * i2cp,
    i2caddr_t addr,
    uint8_t * rxbuf,
    size_t rxbytes,
    sysinterval_t timeout)

```

Receives data from the I2C bus.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object
in	<i>addr</i>	slave device address (7 bits) without R/W bit
out	<i>rxbuf</i>	pointer to receive buffer
in	<i>rxbytes</i>	number of bytes to be received
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

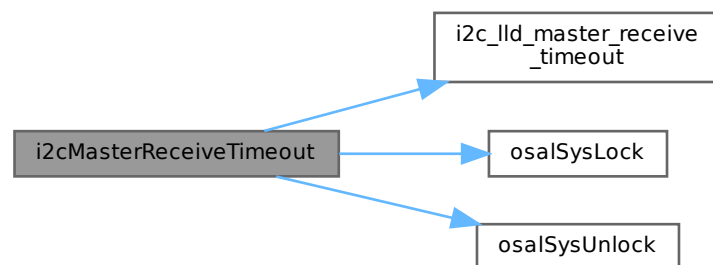
Return values

<i>MSG_OK</i>	if the function succeeded.
<i>MSG_RESET</i>	if one or more I2C errors occurred, the errors can be retrieved using i2cGetErrors() .
<i>MSG_TIMEOUT</i>	if a timeout occurred before operation end.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.9.6.8 i2cAcquireBus()**

```
void i2cAcquireBus (
    I2CDriver * i2cp)
```

Gains exclusive access to the I2C bus.

This function tries to gain ownership to the I2C bus, if the bus is already being used then the invoking thread is queued.

Precondition

In order to use this function the option `I2C_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.9.6.9 i2cReleaseBus()**

```
void i2cReleaseBus (  
    I2CDriver * i2cp)
```

Releases exclusive access to the I2C bus.

Precondition

In order to use this function the option `I2C_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.9.6.10 i2cSlaveMatchAddress()

```
msg_t i2cSlaveMatchAddress (
    I2CDriver * i2cp,
    i2caddr_t addr)
```

Listen I2C bus for address match.

Use 7 bit address (10 bit,dual and general call address doesn't implement yet) .

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object
in	<i>addr</i>	slave device address

Returns

The operation status.

Return values

<i>MSG_OK</i>	if the function succeeded.
<i>MSG_RESET</i>	if one or more I2C errors occurred, the errors can be retrieved using i2cGetErrors() .

Function Class:

Not an API, this function is for internal use only.

7.2.9.9.6.11 i2cSlaveReceiveTimeout()

```
msg_t i2cSlaveReceiveTimeout (
    I2CDriver * i2cp,
    uint8_t * rxbuf,
    size_t rxbytes,
    sysinterval_t timeout)
```

Receive data via the I2C bus as slave and call handler.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object
out	<i>rxbuf</i>	pointer to the receive buffer
in	<i>rxbytes</i>	size of receive buffer
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

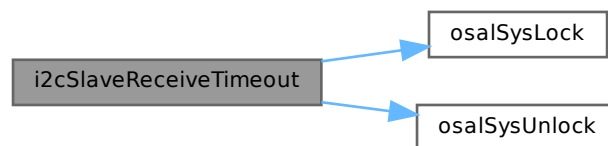
Return values

<i>MSG_OK</i>	if the function succeeded.
<i>MSG_RESET</i>	if one or more I2C errors occurred, the errors can be retrieved using <code>i2cGetErrors()</code> .
<i>MSG_TIMEOUT</i>	if a timeout occurred before operation end. After a timeout the driver must be stopped and restarted because the bus is in an uncertain state.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.9.6.12 i2cSlaveTransmitTimeout()

```

msg_t i2cSlaveTransmitTimeout (
    I2CDriver * i2cp,
    const uint8_t * txbuf,
    size_t txbytes,
    sysinterval_t timeout)
  
```

Transmits data via the I2C bus as slave.

Call this function when Master request data (in request handler)

Parameters

in	<i>i2cp</i>	pointer to the <code>I2CDriver</code> object
in	<i>txbuf</i>	pointer to the transmit buffer
in	<i>txbytes</i>	number of bytes to be transmitted
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <code>TIME_INFINITE</code> no timeout.

Returns

The operation status.

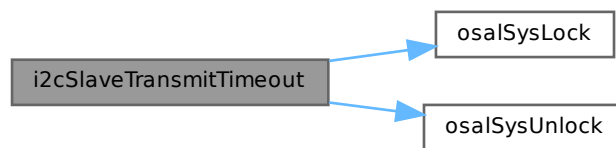
Return values

<i>MSG_OK</i>	if the function succeeded.
<i>MSG_RESET</i>	if one or more I2C errors occurred, the errors can be retrieved using i2cGetErrors() .
<i>MSG_TIMEOUT</i>	if a timeout occurred before operation end. After a timeout the driver must be stopped and restarted because the bus is in an uncertain state.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.9.6.13 i2c_lld_init()

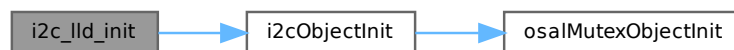
```
void i2c_lld_init (
    void )
```

Low level I2C driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.9.6.14 i2c_lld_start()

```
void i2c_lld_start (
    I2CDriver * i2cp)
```

Configures and activates the I2C peripheral.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.9.6.15 i2c_lld_stop()

```
void i2c_lld_stop (
    I2CDriver * i2cp)
```

Deactivates the I2C peripheral.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.9.6.16 i2c_lld_master_receive_timeout()

```
msg_t i2c_lld_master_receive_timeout (
    I2CDriver * i2cp,
    i2caddr_t addr,
    uint8_t * rxbuf,
    size_t rxbytes,
    sysinterval_t timeout)
```

Receives data via the I2C bus as master.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object
in	<i>addr</i>	slave device address
out	<i>rxbuf</i>	pointer to the receive buffer
in	<i>rxbytes</i>	number of bytes to be received
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

Return values

<i>MSG_OK</i>	if the function succeeded.
<i>MSG_RESET</i>	if one or more I2C errors occurred, the errors can be retrieved using i2cGetErrors() .
<i>MSG_TIMEOUT</i>	if a timeout occurred before operation end. After a timeout the driver must be stopped and restarted because the bus is in an uncertain state.

Function Class:

Not an API, this function is for internal use only.

7.2.9.9.6.17 i2c_llc_master_transmit_timeout()

```
msg_t i2c_llc_master_transmit_timeout (
    I2CDriver * i2cp,
    i2caddr_t addr,
    const uint8_t * txbuf,
    size_t txbytes,
    uint8_t * rxbuf,
    size_t rxbytes,
    sysinterval_t timeout)
```

Transmits data via the I2C bus as master.

Parameters

in	<i>i2cp</i>	pointer to the I2CDriver object
in	<i>addr</i>	slave device address
in	<i>txbuf</i>	pointer to the transmit buffer
in	<i>txbytes</i>	number of bytes to be transmitted
out	<i>rxbuf</i>	pointer to the receive buffer
in	<i>rxbytes</i>	number of bytes to be received
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

Return values

<i>MSG_OK</i>	if the function succeeded.
<i>MSG_RESET</i>	if one or more I2C errors occurred, the errors can be retrieved using i2cGetErrors() .
<i>MSG_TIMEOUT</i>	if a timeout occurred before operation end. After a timeout the driver must be stopped and restarted because the bus is in an uncertain state.

Function Class:

Not an API, this function is for internal use only.

7.2.9.9.7 Variable Documentation

7.2.9.9.7.1 I2CD1

`I2CDriver` I2CD1

I2C1 driver identifier.

7.2.9.10 I2S Driver

Generic I2S Driver.

7.2.9.10.1 Detailed Description

Generic I2S Driver.

This module implements a generic I2S driver.

Precondition

In order to use the I2S driver the `HAL_USE_I2S` option must be enabled in `halconf.h`.

7.2.9.10.2 Driver State Machine

I2S modes

- `#define I2S_MODE_SLAVE` 0
- `#define I2S_MODE_MASTER` 1

Macro Functions

- `#define i2sIsBufferComplete(i2sp)`
Buffer state.
- `#define i2sStartExchange(i2sp)`
Starts a I2S data exchange.
- `#define i2sStopExchange(i2sp)`
Stops the ongoing data exchange.
- `#define _i2s_isr_half_code(i2sp)`
Common ISR code, half buffer event.
- `#define _i2s_isr_full_code(i2sp)`
Common ISR code.

PLATFORM configuration options

- `#define PLATFORM_I2S_USE_I2S1` FALSE
I2SD1 driver enable switch.

Data Structures

- struct [hal_i2s_driver](#)
Structure representing an I2S driver.
- struct [hal_i2s_config](#)
Driver configuration structure.

Macros

- `#define i2s_llt_driver_fields`
Low level fields of the I2S driver structure.
- `#define i2s_llt_config_fields`
Low level fields of the I2S configuration structure.

Typedefs

- typedef struct [hal_i2s_driver](#) I2SDriver
Type of a structure representing an I2S driver.
- typedef struct [hal_i2s_config](#) I2SConfig
Type of a structure representing an I2S driver configuration.
- typedef void(* [i2scallback_t](#)) (I2SDriver *i2sp)
I2S notification callback type.

Enumerations

- enum [i2sstate_t](#) {
 [I2S_UNINIT](#) = 0 , [I2S_STOP](#) = 1 , [I2S_READY](#) = 2 , [I2S_ACTIVE](#) = 3 ,
 [I2S_COMPLETE](#) = 4 }
Driver state machine possible states.

Functions

- void [i2sInit](#) (void)
I2S Driver initialization.
- void [i2sObjectInit](#) (I2SDriver *i2sp)
Initializes the standard part of a [I2SDriver](#) structure.
- `msg_t` [i2sStart](#) (I2SDriver *i2sp, const I2SConfig *config)
Configures and activates the I2S peripheral.
- void [i2sStop](#) (I2SDriver *i2sp)
Deactivates the I2S peripheral.
- void [i2sStartExchange](#) (I2SDriver *i2sp)
Starts a I2S data exchange.
- void [i2sStopExchange](#) (I2SDriver *i2sp)
Stops the ongoing data exchange.
- void [i2s_llt_init](#) (void)
Low level I2S driver initialization.
- void [i2s_llt_start](#) (I2SDriver *i2sp)
Configures and activates the I2S peripheral.
- void [i2s_llt_stop](#) (I2SDriver *i2sp)
Deactivates the I2S peripheral.
- void [i2s_llt_start_exchange](#) (I2SDriver *i2sp)
Starts a I2S data exchange.
- void [i2s_llt_stop_exchange](#) (I2SDriver *i2sp)
Stops the ongoing data exchange.

Variables

- [I2SDriver I2SD1](#)
I2S2 driver identifier.

7.2.9.10.3 Macro Definition Documentation**7.2.9.10.3.1 I2S_MODE_SLAVE**

```
#define I2S_MODE_SLAVE 0
```

7.2.9.10.3.2 I2S_MODE_MASTER

```
#define I2S_MODE_MASTER 1
```

7.2.9.10.3.3 i2sIsBufferComplete

```
#define i2sIsBufferComplete(  
    i2sp)
```

Value:

```
((bool)((i2sp)->state == I2S\_COMPLETE))
```

Buffer state.

Note

This function is meant to be called from the SPI callback only.

Parameters

<i>in</i>	<i>i2sp</i>	pointer to the I2SDriver object

Returns

The buffer state.

Return values

<i>false</i>	if the driver filled/sent the first half of the buffer.
<i>true</i>	if the driver filled/sent the second half of the buffer.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.10.3.4 i2sStartExchangeI

```
#define i2sStartExchangeI(  
    i2sp)
```

Value:

```
{  
    i2s_lld_start_exchange(i2sp);  
    (i2sp)->state = I2S\_ACTIVE;  
}
```

Starts a I2S data exchange.

Parameters

in	<i>i2sp</i>	pointer to the I2SDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.10.3.5 i2sStopExchangeI

```
#define i2sStopExchangeI(  
    i2sp)
```

Value:

```
{  
    i2s_lld_stop_exchange(i2sp);  
    (i2sp)->state = I2S_READY;  
}
```

Stops the ongoing data exchange.

The ongoing data exchange, if any, is stopped, if the driver was not active the function does nothing.

Parameters

in	<i>i2sp</i>	pointer to the I2SDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.10.3.6 _i2s_isr_half_code

```
#define _i2s_isr_half_code(  
    i2sp)
```

Value:

```
{  
    if ((i2sp)->config->end_cb != NULL) {  
        (i2sp)->config->end_cb(i2sp);  
    }  
}
```

Common ISR code, half buffer event.

This code handles the portable part of the ISR code:

- Callback invocation.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>i2sp</i>	pointer to the I2CDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.10.3.7 `_i2s_isr_full_code`

```
#define _i2s_isr_full_code(  
    i2sp)
```

Value:

```
{  
    if ((i2sp)->config->end_cb) {  
        (i2sp)->state = I2S\_COMPLETE;  
        (i2sp)->config->end_cb(i2sp);  
        if ((i2sp)->state == I2S\_COMPLETE) {  
            (i2sp)->state = I2S\_ACTIVE;  
        }  
    }  
}
```

Common ISR code.

This code handles the portable part of the ISR code:

- Callback invocation.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>i2sp</i>	pointer to the I2CDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.10.3.8 `PLATFORM_I2S_USE_I2S1`

```
#define PLATFORM_I2S_USE_I2S1 FALSE
```

I2SD1 driver enable switch.

If set to `TRUE` the support for I2S1 is included.

Note

The default is `FALSE`.

7.2.9.10.3.9 i2s_llt_driver_fields

```
#define i2s_llt_driver_fields
```

Value:

```
/* Dummy field, it is not needed.*/
uint32_t dummy;
```

Low level fields of the I2S driver structure.

7.2.9.10.3.10 i2s_llt_config_fields

```
#define i2s_llt_config_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/
uint32_t dummy;
```

Low level fields of the I2S configuration structure.

7.2.9.10.4 Typedef Documentation

7.2.9.10.4.1 I2SDriver

```
typedef struct hal_i2s_driver I2SDriver
```

Type of a structure representing an I2S driver.

7.2.9.10.4.2 I2SConfig

```
typedef struct hal_i2s_config I2SConfig
```

Type of a structure representing an I2S driver configuration.

7.2.9.10.4.3 i2scallback_t

```
typedef void(* i2scallback_t) (I2SDriver *i2sp)
```

I2S notification callback type.

Parameters

in	<i>i2sp</i>	pointer to the I2SDriver object

7.2.9.10.5 Enumeration Type Documentation

7.2.9.10.5.1 i2sstate_t

```
enum i2sstate_t
```

Driver state machine possible states.

Enumerator

I2S_UNINIT	Not initialized.
I2S_STOP	Stopped.
I2S_READY	Ready.
I2S_ACTIVE	Active.
I2S_COMPLETE	Transmission complete.

7.2.9.10.6 Function Documentation

7.2.9.10.6.1 i2sInit()

```
void i2sInit (  
    void )
```

I2S Driver initialization.

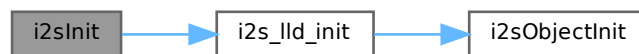
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.10.6.2 i2sObjectInit()

```
void i2sObjectInit (  
    I2SDriver * i2sp)
```

Initializes the standard part of a `I2SDriver` structure.

Parameters

out	<i>i2sp</i>	pointer to the <code>I2SDriver</code> object

Function Class:

Object or module initializer function.

7.2.9.10.6.3 i2sStart()

```
msg_t i2sStart (  
    I2SDriver * i2sp,  
    const I2SConfig * config)
```

Configures and activates the I2S peripheral.

Parameters

in	<i>i2sp</i>	pointer to the I2SDriver object
in	<i>config</i>	pointer to the I2SConfig object

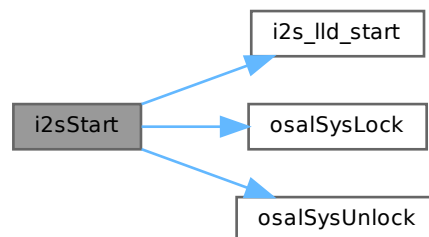
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.10.6.4 i2sStop()

```
void i2sStop (  
    I2SDriver * i2sp)
```

Deactivates the I2S peripheral.

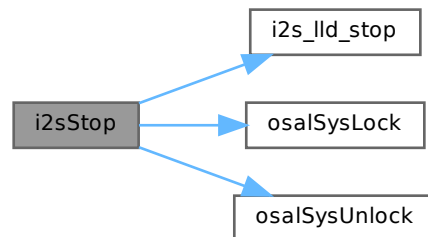
Parameters

in	<i>i2sp</i>	pointer to the I2SDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.10.6.5 i2sStartExchange()**

```
void i2sStartExchange (
    I2SDriver * i2sp)
```

Starts a I2S data exchange.

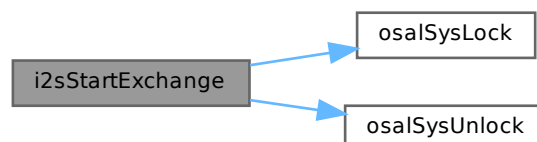
Parameters

in	<i>i2sp</i>	pointer to the <code>I2SDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.10.6.6 i2sStopExchange()

```
void i2sStopExchange (  
    I2SDriver * i2sp)
```

Stops the ongoing data exchange.

The ongoing data exchange, if any, is stopped, if the driver was not active the function does nothing.

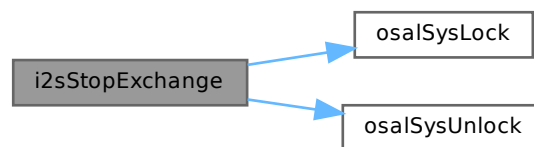
Parameters

in	<i>i2sp</i>	pointer to the <i>I2SDriver</i> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.10.6.7 i2s_lld_init()

```
void i2s_lld_init (  
    void )
```

Low level I2S driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.10.6.8 i2s_ll_start()

```
void i2s_ll_start (  
    I2SDriver * i2sp)
```

Configures and activates the I2S peripheral.

Parameters

in	<i>i2sp</i>	pointer to the I2SDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.10.6.9 i2s_ll_stop()

```
void i2s_ll_stop (  
    I2SDriver * i2sp)
```

Deactivates the I2S peripheral.

Parameters

in	<i>i2sp</i>	pointer to the I2SDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.10.6.10 i2s_ll_start_exchange()

```
void i2s_ll_start_exchange (  
    I2SDriver * i2sp)
```

Starts a I2S data exchange.

Parameters

in	<i>i2sp</i>	pointer to the I2SDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.10.6.11 i2s_ll_stop_exchange()

```
void i2s_ll_stop_exchange (  
    I2SDriver * i2sp)
```

Stops the ongoing data exchange.

The ongoing data exchange, if any, is stopped, if the driver was not active the function does nothing.

Parameters

in	<i>i2sp</i>	pointer to the I2SDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.10.7 Variable Documentation

7.2.9.10.7.1 I2SD1

[I2SDriver](#) I2SD1

I2S2 driver identifier.

7.2.9.11 ICU Driver

Generic ICU Driver.

7.2.9.11.1 Detailed Description

Generic ICU Driver.

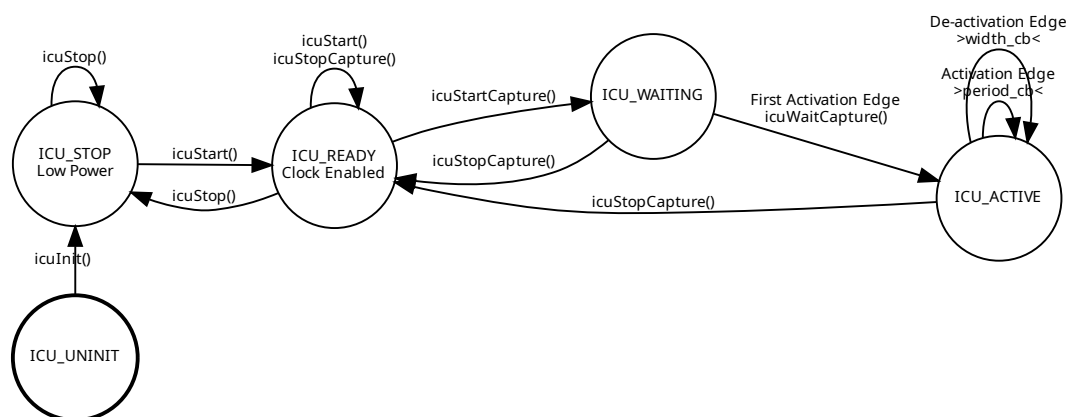
This module implements a generic ICU (Input Capture Unit) driver. The purpose of the driver is to measure period and duty cycle of an input digital signal (PWM input).

Precondition

In order to use the ICU driver the `HAL_USE_ICU` option must be enabled in [halconf.h](#).

7.2.9.11.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



7.2.9.11.3 ICU Operations.

This driver abstracts a generic Input Capture Unit composed of:

- A clock prescaler.
- A main up counter.
- Two capture registers triggered by the rising and falling edges on the sampled input.

The ICU unit can be programmed to synchronize on the rising or falling edge of the sample input:

- **ICU_INPUT_ACTIVE_HIGH**, a rising edge is the start signal.
- **ICU_INPUT_ACTIVE_LOW**, a falling edge is the start signal.

Callbacks are optionally invoked when:

- On the PWM de-activation edge.
- On the PWM activation edge, measurements for the previous cycle are available from this callback and can be retrieved using `icuGetPeriodX()` and `icuGetWidthX()`.

Macro Functions

- `#define icuStartCaptureI(icup)`
Starts the input capture.
- `#define icuStopCaptureI(icup)`
Stops the input capture.
- `#define icuEnableNotificationsI(icup)`
Enables notifications.
- `#define icuDisableNotificationsI(icup)`
Disables notifications.
- `#define icuAreNotificationsEnabledX(icup)`
Check on notifications status.
- `#define icuGetWidthX(icup)`
Returns the width of the latest pulse.
- `#define icuGetPeriodX(icup)`
Returns the width of the latest cycle.

Low level driver helper macros

- `#define _icu_isr_invoke_width_cb(icup)`
Common ISR code, ICU width event.
- `#define _icu_isr_invoke_period_cb(icup)`
Common ISR code, ICU period event.
- `#define _icu_isr_invoke_overflow_cb(icup)`
Common ISR code, ICU timer overflow event.

PLATFORM configuration options

- `#define PLATFORM_ICU_USE_ICU1 FALSE`
ICUD1 driver enable switch.

Data Structures

- struct `ICUConfig`
Driver configuration structure.
- struct `ICUDriver`
Structure representing an ICU driver.

Macros

- `#define ICU_USE_OVERFLOW_SCALING FALSE`
- `#define icu_ild_get_width(icup)`
Returns the width of the latest pulse.
- `#define icu_ild_get_period(icup)`
Returns the width of the latest cycle.
- `#define icu_ild_are_notifications_enabled(icup)`
Check on notifications status.

Typedefs

- typedef struct `ICUDriver` `ICUDriver`
Type of a structure representing an ICU driver.
- typedef void(* `icucallback_t`) (`ICUDriver` *icup)
ICU notification callback type.
- typedef uint32_t `icufreq_t`
ICU frequency type.
- typedef uint32_t `icucnt_t`
ICU counter type.

Enumerations

- enum `icustate_t` {
 `ICU_UNINIT` = 0 , `ICU_STOP` = 1 , `ICU_READY` = 2 , `ICU_WAITING` = 3 ,
 `ICU_ACTIVE` = 4 }
Driver state machine possible states.
- enum `icumode_t` { `ICU_INPUT_ACTIVE_HIGH` = 0 , `ICU_INPUT_ACTIVE_LOW` = 1 }
ICU driver mode.

Functions

- void `icuinIt` (void)
ICU Driver initialization.
- void `icuObjectInit` (ICUDriver *icup)
Initializes the standard part of a `ICUDriver` structure.
- `msg_t` `icuStart` (ICUDriver *icup, const ICUConfig *config)
Configures and activates the ICU peripheral.
- void `icuStop` (ICUDriver *icup)
Deactivates the ICU peripheral.
- void `icuStartCapture` (ICUDriver *icup)
Starts the input capture.
- bool `icuWaitCapture` (ICUDriver *icup)
Waits for a completed capture.
- void `icuStopCapture` (ICUDriver *icup)
Stops the input capture.
- void `icuEnableNotifications` (ICUDriver *icup)
Enables notifications.
- void `icuDisableNotifications` (ICUDriver *icup)
Disables notifications.
- void `icu_lld_init` (void)
Low level ICU driver initialization.
- void `icu_lld_start` (ICUDriver *icup)
Configures and activates the ICU peripheral.
- void `icu_lld_stop` (ICUDriver *icup)
Deactivates the ICU peripheral.
- void `icu_lld_start_capture` (ICUDriver *icup)
Starts the input capture.
- bool `icu_lld_wait_capture` (ICUDriver *icup)
Waits for a completed capture.
- void `icu_lld_stop_capture` (ICUDriver *icup)
Stops the input capture.
- void `icu_lld_enable_notifications` (ICUDriver *icup)
Enables notifications.
- void `icu_lld_disable_notifications` (ICUDriver *icup)
Disables notifications.

Variables

- `ICUDriver ICUD1`
ICUD1 driver identifier.

7.2.9.11.4 Macro Definition Documentation

7.2.9.11.4.1 ICU_USE_OVERFLOW_SCALING

```
#define ICU_USE_OVERFLOW_SCALING FALSE
```

7.2.9.11.4.2 icuStartCaptureI

```
#define icuStartCaptureI(  
    icup)
```

Value:

```
do {  
    icu_ll_start_capture(icup);  
    (icup)->state = ICU_WAITING;  
} while (false)
```

Starts the input capture.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.11.4.3 icuStopCaptureI

```
#define icuStopCaptureI(  
    icup)
```

Value:

```
do {  
    icu_ll_stop_capture(icup);  
    (icup)->state = ICU_READY;  
} while (false)
```

Stops the input capture.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.11.4.4 icuEnableNotificationsI

```
#define icuEnableNotificationsI(  
    icup)
```

Value:

```
icu_ll_enable_notifications(icup)
```

Enables notifications.

Precondition

The ICU unit must have been activated using [icuStart\(\)](#) and the capture started using [icuStartCapture\(\)](#).

Note

If the notification is already enabled then the call has no effect.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.11.4.5 icuDisableNotificationsI

```
#define icuDisableNotificationsI(  
    icup)
```

Value:

[icu_lld_disable_notifications](#)(*icup*)

Disables notifications.

Precondition

The ICU unit must have been activated using [icuStart\(\)](#) and the capture started using [icuStartCapture\(\)](#).

Note

If the notification is already disabled then the call has no effect.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.11.4.6 icuAreNotificationsEnabledX

```
#define icuAreNotificationsEnabledX(  
    icup)
```

Value:

[icu_lld_are_notifications_enabled](#)(*icup*)

Check on notifications status.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Returns

The notifications status.

Return values

<i>false</i>	if notifications are not enabled.
<i>true</i>	if notifications are enabled.

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.4.7 icuGetWidthX

```
#define icuGetWidthX(  
    icup)
```

Value:

```
icu_lld_get_width(icup)
```

Returns the width of the latest pulse.

The pulse width is defined as number of ticks between the start edge and the stop edge.

Note

This function is meant to be invoked from the width capture callback.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Returns

The number of ticks.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.11.4.8 icuGetPeriodX

```
#define icuGetPeriodX(  
    icup)
```

Value:

```
icu_lld_get_period(icup)
```

Returns the width of the latest cycle.

The cycle width is defined as number of ticks between a start edge and the next start edge.

Note

This function is meant to be invoked from the width capture callback.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Returns

The number of ticks.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.11.4.9 _icu_isr_invoke_width_cb

```
#define _icu_isr_invoke_width_cb(  
    icup)
```

Value:

```
do {  
    if (((icup)->state == ICU\_ACTIVE) &&  
        ((icup)->config->width_cb != NULL))  
        (icup)->config->width_cb(icup);  
} while (0)
```

```
\  
\  
\
```

Common ISR code, ICU width event.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.4.10 _icu_isr_invoke_period_cb

```
#define _icu_isr_invoke_period_cb(  
    icup)
```

Value:

```
do {  
    if (((icup)->state == ICU\_ACTIVE) &&  
        ((icup)->config->period_cb != NULL))  
        (icup)->config->period_cb(icup);  
        (icup)->state = ICU\_ACTIVE;  
} while (0)
```

```
\  
\  
\
```

Common ISR code, ICU period event.

Note

A period event brings the driver into the [ICU_ACTIVE](#) state.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.4.11 _icu_isr_invoke_overflow_cb

```
#define _icu_isr_invoke_overflow_cb(  
    icup)
```

Value:

```
do {  
    if ((icup->config->overflow_cb != NULL && (icup->state == ICU\_ACTIVE) \  
        (icup->config->overflow_cb(icup); \  
    else \  
        (icup->state = ICU\_WAITING; \  
} while (0)
```

Common ISR code, ICU timer overflow event.

Note

An overflow when in [ICU_ACTIVE](#) state with a callback set executes the callback code (supports overflow scaling). Otherwise overflow brings the driver back to the [ICU_WAITING](#) state.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.4.12 PLATFORM_ICU_USE_ICU1

```
#define PLATFORM_ICU_USE_ICU1 FALSE
```

ICUD1 driver enable switch.

If set to [TRUE](#) the support for ICUD1 is included.

Note

The default is [FALSE](#).

7.2.9.11.4.13 icu_lld_get_width

```
#define icu_lld_get_width(  
    icup)
```

Value:

0

Returns the width of the latest pulse.

The pulse width is defined as number of ticks between the start edge and the stop edge.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Returns

The number of ticks.

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.4.14 icu_lld_get_period

```
#define icu_lld_get_period(  
    icup)
```

Value:

0

Returns the width of the latest cycle.

The cycle width is defined as number of ticks between a start edge and the next start edge.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Returns

The number of ticks.

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.4.15 icu_lld_are_notifications_enabled

```
#define icu_lld_are_notifications_enabled(  
    icup)
```

Value:

false

Check on notifications status.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Returns

The notifications status.

Return values

<i>false</i>	if notifications are not enabled.
<i>true</i>	if notifications are enabled.

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.5 Typedef Documentation**7.2.9.11.5.1 ICUDriver**

```
typedef struct ICUDriver ICUDriver
```

Type of a structure representing an ICU driver.

7.2.9.11.5.2 icucallback_t

```
typedef void(* icucallback_t) (ICUDriver *icup)
```

ICU notification callback type.

Parameters

in	<i>icup</i>	pointer to a ICUDriver object

7.2.9.11.5.3 icufreq_t

```
typedef uint32_t icufreq_t
```

ICU frequency type.

7.2.9.11.5.4 icucnt_t

```
typedef uint32_t icucnt_t
```

ICU counter type.

7.2.9.11.6 Enumeration Type Documentation**7.2.9.11.6.1 icustate_t**

```
enum icustate_t
```

Driver state machine possible states.

Enumerator

ICU_UNINIT	Not initialized.
ICU_STOP	Stopped.
ICU_READY	Ready.
ICU_WAITING	Waiting for first front.
ICU_ACTIVE	First front detected.

7.2.9.11.6.2 icumode_t

```
enum icumode_t
```

ICU driver mode.

Enumerator

ICU_INPUT_ACTIVE_HIGH	Trigger on rising edge.
ICU_INPUT_ACTIVE_LOW	Trigger on falling edge.

7.2.9.11.7 Function Documentation

7.2.9.11.7.1 icuInit()

```
void icuInit (
    void )
```

ICU Driver initialization.

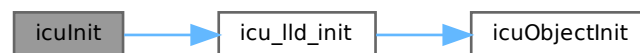
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.11.7.2 icuObjectInit()

```
void icuObjectInit (
    ICUDriver * icup)
```

Initializes the standard part of a `ICUDriver` structure.

Parameters

out	<i>icup</i>	pointer to the ICUDriver object

Function Class:

Object or module initializer function.

7.2.9.11.7.3 icuStart()

```
msg_t icuStart (  
    ICUDriver * icup,  
    const ICUConfig * config)
```

Configures and activates the ICU peripheral.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object
in	<i>config</i>	pointer to the ICUConfig object

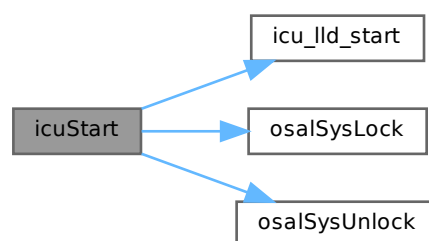
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.11.7.4 icuStop()**

```
void icuStop (  
    ICUDriver * icup)
```

Deactivates the ICU peripheral.

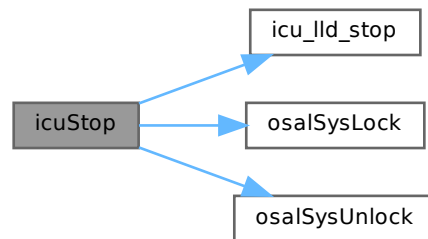
Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.11.7.5 icuStartCapture()**

```
void icuStartCapture (  
    ICUDriver * icup)
```

Starts the input capture.

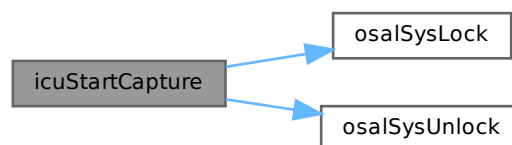
Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.11.7.6 icuWaitCapture()

```
bool icuWaitCapture (
    ICUDriver * icup)
```

Waits for a completed capture.

Note

The operation could be performed in polled mode depending on.
In order to use this function notifications must be disabled.

Precondition

The driver must be in `ICU_WAITING` or `ICU_ACTIVE` states.

Postcondition

After the capture is available the driver is in `ICU_ACTIVE` state. If a capture fails then the driver is in `ICU_WAITING` state.

Parameters

in	<i>icup</i>	pointer to the <code>ICUDriver</code> object

Returns

The capture status.

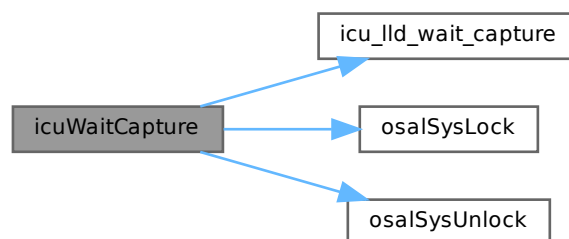
Return values

<i>false</i>	if the capture is successful.
<i>true</i>	if a timer overflow occurred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.11.7.7 icuStopCapture()

```
void icuStopCapture (  
    ICUDriver * icup)
```

Stops the input capture.

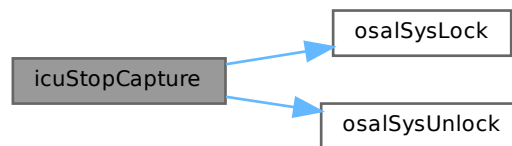
Parameters

in	<i>icup</i>	pointer to the <code>ICUDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.11.7.8 icuEnableNotifications()

```
void icuEnableNotifications (  
    ICUDriver * icup)
```

Enables notifications.

Precondition

The ICU unit must have been activated using `icuStart()` and the capture started using `icuStartCapture()`.

Note

If the notification is already enabled then the call has no effect.

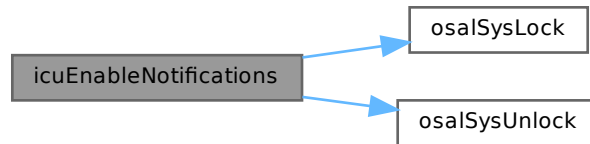
Parameters

in	<i>icup</i>	pointer to the <code>ICUDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.11.7.9 icuDisableNotifications()**

```
void icuDisableNotifications (  
    ICUDriver * icup)
```

Disables notifications.

Precondition

The ICU unit must have been activated using `icuStart()` and the capture started using `icuStartCapture()`.

Note

If the notification is already disabled then the call has no effect.

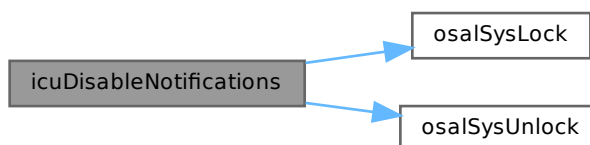
Parameters

in	<i>icup</i>	pointer to the <code>ICUDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.11.7.10 icu_lld_init()

```
void icu_lld_init (  
    void )
```

Low level ICU driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.11.7.11 icu_lld_start()

```
void icu_lld_start (  
    ICUDriver * icup)
```

Configures and activates the ICU peripheral.

Parameters

in	<i>icup</i>	pointer to the <code>ICUDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.7.12 icu_lld_stop()

```
void icu_lld_stop (  
    ICUDriver * icup)
```

Deactivates the ICU peripheral.

Parameters

in	<i>icup</i>	pointer to the <code>ICUDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.7.13 icu_lld_start_capture()

```
void icu_lld_start_capture (  
    ICUDriver * icup)
```

Starts the input capture.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.7.14 icu_lld_wait_capture()

```
bool icu_lld_wait_capture (  
    ICUDriver * icup)
```

Waits for a completed capture.

Note

The operation is performed in polled mode.

In order to use this function notifications must be disabled.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Returns

The capture status.

Return values

<i>false</i>	if the capture is successful.
<i>true</i>	if a timer overflow occurred.

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.7.15 icu_lld_stop_capture()

```
void icu_lld_stop_capture (  
    ICUDriver * icup)
```

Stops the input capture.

Parameters

in	<i>icup</i>	pointer to the ICUDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.11.7.16 icu_lld_enable_notifications()

```
void icu_lld_enable_notifications (
    ICUDriver * icup)
```

Enables notifications.

Precondition

The ICU unit must have been activated using `icuStart()` and the capture started using `icuStartCapture()`.

Note

If the notification is already enabled then the call has no effect.

Parameters

in	<i>icup</i>	pointer to the <code>ICUDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.11.7.17 icu_lld_disable_notifications()

```
void icu_lld_disable_notifications (
    ICUDriver * icup)
```

Disables notifications.

Precondition

The ICU unit must have been activated using `icuStart()` and the capture started using `icuStartCapture()`.

Note

If the notification is already disabled then the call has no effect.

Parameters

in	<i>icup</i>	pointer to the <code>ICUDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.11.8 Variable Documentation

7.2.9.11.8.1 ICUD1

`ICUDriver` ICUD1

ICUD1 driver identifier.

Note

The driver ICUD1 allocates the complex timer TIM1 when enabled.

7.2.9.12 MAC Driver

Generic MAC Driver.

7.2.9.12.1 Detailed Description

Generic MAC Driver.

This module implements a generic MAC (Media Access Control) driver for Ethernet controllers.

Precondition

In order to use the MAC driver the `HAL_USE_MAC` option must be enabled in `halconf.h`.

MAC event flags

- `#define MAC_FLAGS_TX` (1U << 0)
- `#define MAC_FLAGS_RX` (1U << 1)

MAC configuration options

- `#define MAC_USE_ZERO_COPY` FALSE
Enables an event sources for incoming packets.
- `#define MAC_USE_EVENTS` TRUE
Enables an event sources for incoming packets.

Low level driver helper macros

- `#define __mac_callback(macp)`
MAC callback.
- `#define __mac_tx_wakeup(macp)`
MAC TX wakeup and event.
- `#define __mac_rx_wakeup(macp)`
MAC RX wakeup and event.

Macro Functions

- `#define macSetCallbackX(macp, f)`
Associates a callback to the MAC instance.
- `#define macGetEventSource(macp)`
Returns the driver events source.
- `#define macGetTransmitDescriptorX(macp, tdp)`
Returns a transmission descriptor.
- `#define macReleaseTransmitDescriptorX(tdp)`
Releases a transmit descriptor and starts the transmission of the enqueued data as a single frame.
- `#define macGetReceiveDescriptorX(macp, rdp)`
Returns a receive descriptor.
- `#define macReleaseReceiveDescriptorX(rdp)`
Releases a receive descriptor.
- `#define macWriteTransmitDescriptor(tdp, buf, size)`
Writes to a transmit descriptor's stream.
- `#define macReadReceiveDescriptor(rdp, buf, size)`
Reads from a receive descriptor's stream.
- `#define macGetNextTransmitBuffer(tdp, size, sizep)`
Returns a pointer to the next transmit buffer in the descriptor chain.
- `#define macGetNextReceiveBuffer(rdp, sizep)`
Returns a pointer to the next receive buffer in the descriptor chain.

PLATFORM configuration options

- `#define PLATFORM_MAC_USE_MAC1 FALSE`
MAC driver enable switch.

Data Structures

- struct `hal_mac_config`
Driver configuration structure.
- struct `hal_mac_driver`
Structure representing a MAC driver.
- struct `hal_mac_transmit_descriptor`
Structure representing a MAC transmit descriptor.
- struct `hal_mac_receive_descriptor`
Structure representing a MAC receive descriptor.

Macros

- `#define MAC_SUPPORTS_ZERO_COPY TRUE`
This implementation supports the zero-copy mode API.
- `#define mac_lld_driver_fields`
Low level fields of the MAC driver structure.
- `#define mac_lld_config_fields`
Low level fields of the MAC configuration structure.
- `#define mac_lld_transmit_descriptor_fields`
Low level fields of the MAC transmit descriptor structure.
- `#define mac_lld_receive_descriptor_fields`
Low level fields of the MAC receive descriptor structure.

Typedefs

- typedef struct [hal_mac_driver](#) [MACDriver](#)
Type of a structure representing a MAC driver.
- typedef struct [hal_mac_config](#) [MACConfig](#)
Type of structure representing a MAC configuration.
- typedef struct [hal_mac_transmit_descriptor](#) [MACTransmitDescriptor](#)
Type of structure representing a MAC transmit descriptor.
- typedef struct [hal_mac_receive_descriptor](#) [MACReceiveDescriptor](#)
Type of structure representing a MAC receive descriptor.
- typedef void(* [maccb_t](#)) ([MACDriver](#) *macp)
Generic ETH notification callback type.

Enumerations

- enum [macstate_t](#) { [MAC_UNINIT](#) = 0 , [MAC_STOP](#) = 1 , [MAC_ACTIVE](#) = 2 }
Driver state machine possible states.

Functions

- void [macInit](#) (void)
MAC Driver initialization.
- void [macObjectInit](#) ([MACDriver](#) *macp)
Initialize the standard part of a [MACDriver](#) structure.
- [msg_t](#) [macStart](#) ([MACDriver](#) *macp, const [MACConfig](#) *config)
Configures and activates the MAC peripheral.
- void [macStop](#) ([MACDriver](#) *macp)
Deactivates the MAC peripheral.
- [eventflags_t](#) [macGetAndClearEvents](#) ([MACDriver](#) *macp)
Get and clears MAC event flags.
- [msg_t](#) [macWaitTransmitDescriptor](#) ([MACDriver](#) *macp, [MACTransmitDescriptor](#) *tdp, [sysinterval_t](#) timeout)
Allocates a transmission descriptor.
- [msg_t](#) [macWaitReceiveDescriptor](#) ([MACDriver](#) *macp, [MACReceiveDescriptor](#) *rdp, [sysinterval_t](#) timeout)
Waits for a received frame.
- bool [macPollLinkStatus](#) ([MACDriver](#) *macp)
Updates and returns the link status.
- void [macReleaseTransmitDescriptor](#) ([MACTransmitDescriptor](#) *tdp)
- void [macReleaseReceiveDescriptor](#) ([MACReceiveDescriptor](#) *rdp)
- void [mac_llc_init](#) (void)
Low level MAC initialization.
- void [mac_llc_start](#) ([MACDriver](#) *macp)
Configures and activates the MAC peripheral.
- void [mac_llc_stop](#) ([MACDriver](#) *macp)
Deactivates the MAC peripheral.
- [msg_t](#) [mac_llc_get_transmit_descriptor](#) ([MACDriver](#) *macp, [MACTransmitDescriptor](#) *tdp)
Returns a transmission descriptor.
- void [mac_llc_release_transmit_descriptor](#) ([MACTransmitDescriptor](#) *tdp)
Releases a transmit descriptor and starts the transmission of the enqueued data as a single frame.
- [msg_t](#) [mac_llc_get_receive_descriptor](#) ([MACDriver](#) *macp, [MACReceiveDescriptor](#) *rdp)
Returns a receive descriptor.

- void `mac_lld_release_receive_descriptor` (`MACReceiveDescriptor` *rdp)
Releases a receive descriptor.
- bool `mac_lld_poll_link_status` (`MACDriver` *macp)
Updates and returns the link status.
- size_t `mac_lld_write_transmit_descriptor` (`MACTransmitDescriptor` *tdp, uint8_t *buf, size_t size)
Writes to a transmit descriptor's stream.
- size_t `mac_lld_read_receive_descriptor` (`MACReceiveDescriptor` *rdp, uint8_t *buf, size_t size)
Reads from a receive descriptor's stream.
- uint8_t * `mac_lld_get_next_transmit_buffer` (`MACTransmitDescriptor` *tdp, size_t size, size_t *sizep)
Returns a pointer to the next transmit buffer in the descriptor chain.
- const uint8_t * `mac_lld_get_next_receive_buffer` (`MACReceiveDescriptor` *rdp, size_t *sizep)
Returns a pointer to the next receive buffer in the descriptor chain.

Variables

- `MACDriver ETHD1`
MAC1 driver identifier.

7.2.9.12.2 Macro Definition Documentation

7.2.9.12.2.1 MAC_FLAGS_TX

```
#define MAC_FLAGS_TX (1U << 0)
```

7.2.9.12.2.2 MAC_FLAGS_RX

```
#define MAC_FLAGS_RX (1U << 1)
```

7.2.9.12.2.3 MAC_USE_ZERO_COPY

```
#define MAC_USE_ZERO_COPY FALSE
```

Enables an event sources for incoming packets.

7.2.9.12.2.4 MAC_USE_EVENTS

```
#define MAC_USE_EVENTS TRUE
```

Enables an event sources for incoming packets.

7.2.9.12.2.5 __mac_callback

```
#define __mac_callback(  
    macp)
```

Value:

```
do {  
    if ((macp)->cb != NULL) {  
        (macp)->cb(macp);  
    }  
} while (false)
```

MAC callback.

Parameters

in	<i>macp</i>	pointer to the MACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.2.6 __mac_tx_wakeup

```
#define __mac_tx_wakeup(  
    macp)
```

Value:

```
do {  
    osalSysLockFromISR();  
    osalThreadDequeueAllI (&(macp)->tdqueue, MSG_OK);  
    (macp)->flags |= MAC_FLAGS_TX;  
    osalEventBroadcastFlagsI (&(macp)->es, MAC_FLAGS_TX);  
    osalSysUnlockFromISR();  
} while (false)
```

MAC TX wakeup and event.

Parameters

in	<i>macp</i>	pointer to the MACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.2.7 __mac_rx_wakeup

```
#define __mac_rx_wakeup(  
    macp)
```

Value:

```
do {  
    osalSysLockFromISR();  
    osalThreadDequeueAllI (&(macp)->rdqueue, MSG_OK);  
    (macp)->flags |= MAC_FLAGS_RX;  
    osalEventBroadcastFlagsI (&(macp)->es, MAC_FLAGS_RX);  
    osalSysUnlockFromISR();  
} while (false)
```

MAC RX wakeup and event.

Parameters

in	<i>macp</i>	pointer to the MACDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.2.8 macSetCallbackX

```
#define macSetCallbackX(  
    macp,  
    f)
```

Value:

```
(siop)->cb = (f)
```

Associates a callback to the MAC instance.

Parameters

in	<i>macp</i>	pointer to the MACDriver object
in	<i>f</i>	callback to be associated

7.2.9.12.2.9 macGetEventSource

```
#define macGetEventSource(  
    macp)
```

Value:

```
(&(macp)->es)
```

Returns the driver events source.

Parameters

in	<i>macp</i>	pointer to the MACDriver object

Returns

The pointer to the `EventSource` structure.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.12.2.10 macGetTransmitDescriptorX

```
#define macGetTransmitDescriptorX(  
    macp,  
    tdp)
```

Value:

```
mac\_llc\_get\_transmit\_descriptor(macp, tdp)
```

Returns a transmission descriptor.

One of the available transmission descriptors is locked and returned.

Parameters

in	<i>macp</i>	pointer to the MACDriver object
out	<i>tdp</i>	pointer to a MACTransmitDescriptor structure

Returns

The operation status.

Return values

<i>MSG_OK</i>	the descriptor has been obtained.
<i>MSG_TIMEOUT</i>	descriptor not available.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.12.2.11 macReleaseTransmitDescriptorX

```
#define macReleaseTransmitDescriptorX(  
    tdp)
```

Value:

```
mac_lld_release_transmit_descriptor(tdp)
```

Releases a transmit descriptor and starts the transmission of the enqueued data as a single frame.

Parameters

in	<i>tdp</i>	the pointer to the MACTransmitDescriptor structure

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.2.12 macGetReceiveDescriptorX

```
#define macGetReceiveDescriptorX(  
    macp,  
    rdp)
```

Value:

```
mac_lld_get_receive_descriptor(macp, rdp)
```

Returns a receive descriptor.

Parameters

in	<i>macp</i>	pointer to the MACDriver object
out	<i>rdp</i>	pointer to a MACReceiveDescriptor structure

Returns

The operation status.

Return values

<i>MSG_OK</i>	the descriptor has been obtained.
<i>MSG_TIMEOUT</i>	descriptor not available.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.12.2.13 macReleaseReceiveDescriptorX

```
#define macReleaseReceiveDescriptorX(  
    rdp)
```

Value:

```
mac_lld_release_receive_descriptor(rdp)
```

Releases a receive descriptor.

The descriptor and its buffer are made available for more incoming frames.

Parameters

in	<i>rdp</i>	the pointer to the MACReceiveDescriptor structure

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.12.2.14 macWriteTransmitDescriptor

```
#define macWriteTransmitDescriptor(  
    tdp,  
    buf,  
    size)
```

Value:

```
mac_lld_write_transmit_descriptor(tdp, buf, size)
```

Writes to a transmit descriptor's stream.

Parameters

in	<i>tdp</i>	pointer to a MACTransmitDescriptor structure
in	<i>buf</i>	pointer to the buffer containing the data to be written
in	<i>size</i>	number of bytes to be written

Returns

The number of bytes written into the descriptor's stream, this value can be less than the amount specified in the parameter `size` if the maximum frame size is reached.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.12.2.15 macReadReceiveDescriptor

```
#define macReadReceiveDescriptor(  
    rdp,  
    buf,  
    size)
```

Value:

```
mac_llid_read_receive_descriptor(rdp, buf, size)
```

Reads from a receive descriptor's stream.

Parameters

in	<i>rdp</i>	pointer to a MACReceiveDescriptor structure
in	<i>buf</i>	pointer to the buffer that will receive the read data
in	<i>size</i>	number of bytes to be read

Returns

The number of bytes read from the descriptor's stream, this value can be less than the amount specified in the parameter `size` if there are no more bytes to read.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.12.2.16 macGetNextTransmitBuffer

```
#define macGetNextTransmitBuffer(  
    tdp,  
    size,  
    sizep)
```

Value:

```
mac_llid_get_next_transmit_buffer(tdp, size, sizep)
```

Returns a pointer to the next transmit buffer in the descriptor chain.

Note

The API guarantees that enough buffers can be requested to fill a whole frame.

Parameters

in	<i>tdp</i>	pointer to a MACTransmitDescriptor structure
in	<i>size</i>	size of the requested buffer. Specify the frame size on the first call then scale the value down subtracting the amount of data already copied into the previous buffers.
out	<i>sizep</i>	pointer to variable receiving the real buffer size. The returned value can be less than the amount requested, this means that more buffers must be requested in order to fill the frame data entirely.

Returns

Pointer to the returned buffer.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.12.2.17 macGetNextReceiveBuffer

```
#define macGetNextReceiveBuffer(
    rdp,
    sizep)
```

Value:

```
mac_llt_get_next_receive_buffer(rdp, sizep)
```

Returns a pointer to the next receive buffer in the descriptor chain.

Note

The API guarantees that the descriptor chain contains a whole frame.

Parameters

in	<i>rdp</i>	pointer to a MACReceiveDescriptor structure
out	<i>sizep</i>	pointer to variable receiving the buffer size, it is zero when the last buffer has already been returned.

Returns

Pointer to the returned buffer.

Return values

<i>NULL</i>	if the buffer chain has been entirely scanned.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.12.2.18 MAC_SUPPORTS_ZERO_COPY

```
#define MAC_SUPPORTS_ZERO_COPY TRUE
```

This implementation supports the zero-copy mode API.

7.2.9.12.2.19 PLATFORM_MAC_USE_MAC1

```
#define PLATFORM_MAC_USE_MAC1 FALSE
```

MAC driver enable switch.

If set to TRUE the support for MAC1 is included.

Note

The default is FALSE.

7.2.9.12.2.20 mac_lld_driver_fields

```
#define mac_lld_driver_fields
```

Low level fields of the MAC driver structure.

7.2.9.12.2.21 mac_lld_config_fields

```
#define mac_lld_config_fields
```

Low level fields of the MAC configuration structure.

7.2.9.12.2.22 mac_lld_transmit_descriptor_fields

```
#define mac_lld_transmit_descriptor_fields
```

Low level fields of the MAC transmit descriptor structure.

7.2.9.12.2.23 mac_lld_receive_descriptor_fields

```
#define mac_lld_receive_descriptor_fields
```

Low level fields of the MAC receive descriptor structure.

7.2.9.12.3 Typedef Documentation

7.2.9.12.3.1 MACDriver

```
typedef struct hal_mac_driver MACDriver
```

Type of a structure representing a MAC driver.

7.2.9.12.3.2 MACConfig

```
typedef struct hal_mac_config MACConfig
```

Type of structure representing a MAC configuration.

7.2.9.12.3.3 MACTransmitDescriptor

```
typedef struct hal_mac_transmit_descriptor MACTransmitDescriptor
```

Type of structure representing a MAC transmit descriptor.

7.2.9.12.3.4 MACReceiveDescriptor

```
typedef struct hal_mac_receive_descriptor MACReceiveDescriptor
```

Type of structure representing a MAC receive descriptor.

7.2.9.12.3.5 maccb_t

```
typedef void(* maccb_t) (MACDriver *macp)
```

Generic ETH notification callback type.

Parameters

in	<i>macp</i>	pointer to the <code>MACDriver</code> object

7.2.9.12.4 Enumeration Type Documentation

7.2.9.12.4.1 macstate_t

```
enum macstate_t
```

Driver state machine possible states.

Enumerator

MAC_UNINIT	Not initialized.
MAC_STOP	Stopped.
MAC_ACTIVE	Active.

7.2.9.12.5 Function Documentation

7.2.9.12.5.1 macInit()

```
void macInit (
    void )
```

MAC Driver initialization.

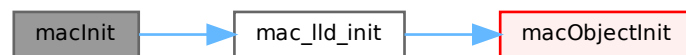
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.12.5.2 macObjectInit()

```
void macObjectInit (
    MACDriver * macp)
```

Initialize the standard part of a `MACDriver` structure.

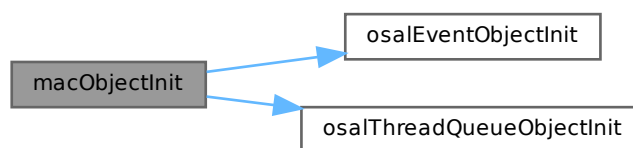
Parameters

out	<i>macp</i>	pointer to the <code>MACDriver</code> object

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.12.5.3 macStart()

```
msg_t macStart (  
    MACDriver * macp,  
    const MACConfig * config)
```

Configures and activates the MAC peripheral.

Parameters

in	<i>macp</i>	pointer to the <code>MACDriver</code> object
in	<i>config</i>	pointer to the <code>MACConfig</code> object

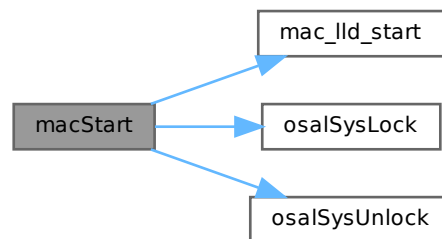
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.12.5.4 macStop()

```
void macStop (  
    MACDriver * macp)
```

Deactivates the MAC peripheral.

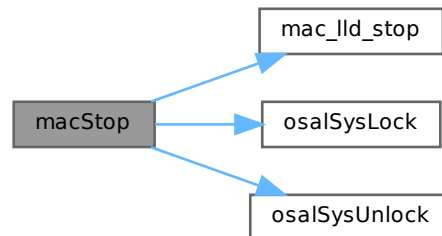
Parameters

in	<i>macp</i>	pointer to the <code>MACDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.12.5.5 macGetAndClearEventsI()**

```
eventflags_t macGetAndClearEventsI (
    MACDriver * macp)
```

Get and clears MAC event flags.

Parameters

in	<code>macp</code>	pointer to the <code>MACDriver</code> object

Returns

The pending event flags.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.12.5.6 macWaitTransmitDescriptor()

```
msg_t macWaitTransmitDescriptor (
    MACDriver * macp,
    MACTransmitDescriptor * tdp,
    sysinterval_t timeout)
```

Allocates a transmission descriptor.

One of the available transmission descriptors is locked and returned. If a descriptor is not currently available then the invoking thread is queued until one is freed.

Parameters

in	<i>macp</i>	pointer to the MACDriver object
out	<i>tdp</i>	pointer to a MACTransmitDescriptor structure
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

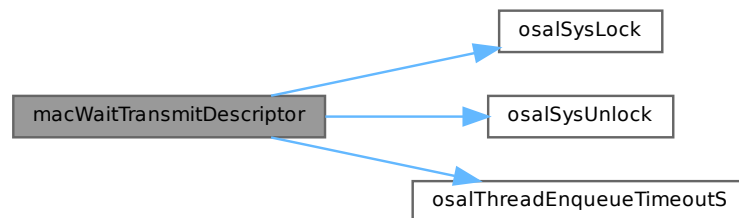
Return values

<i>MSG_OK</i>	the descriptor was obtained.
<i>MSG_TIMEOUT</i>	the operation timed out, descriptor not initialized.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.12.5.7 macWaitReceiveDescriptor()

```

msg_t macWaitReceiveDescriptor (
    MACDriver * macp,
    MACReceiveDescriptor * rdp,
    sysinterval_t timeout)
  
```

Waits for a received frame.

Stops until a frame is received and buffered. If a frame is not immediately available then the invoking thread is queued until one is received.

Parameters

in	<i>macp</i>	pointer to the MACDriver object
out	<i>rdp</i>	pointer to a MACReceiveDescriptor structure
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

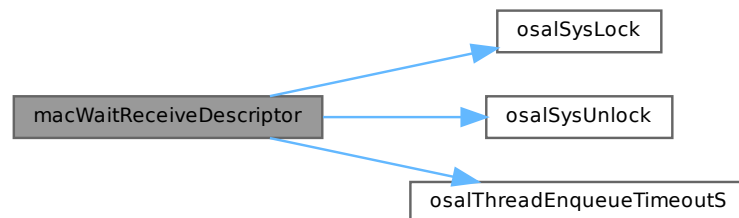
Return values

<i>MSG_OK</i>	the descriptor was obtained.
<i>MSG_TIMEOUT</i>	the operation timed out, descriptor not initialized.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.12.5.8 macPollLinkStatus()**

```
bool macPollLinkStatus (
    MACDriver * macp)
```

Updates and returns the link status.

Parameters

in	<i>macp</i>	pointer to the MACDriver object

Returns

The link status.

Return values

<i>true</i>	if the link is active.
<i>false</i>	if the link is down.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.12.5.9 macReleaseTransmitDescriptor()**

```
void macReleaseTransmitDescriptor (
    MACTransmitDescriptor * tdp)
```

7.2.9.12.5.10 macReleaseReceiveDescriptor()

```
void macReleaseReceiveDescriptor (
    MACReceiveDescriptor * rdp)
```

7.2.9.12.5.11 mac_lld_init()

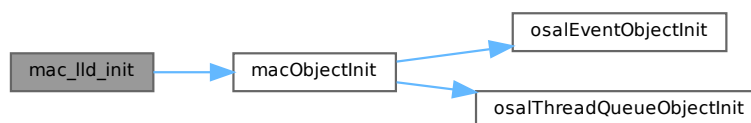
```
void mac_lld_init (
    void )
```

Low level MAC initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.12.5.12 `mac_lld_start()`

```
void mac_lld_start (  
    MACDriver * macp)
```

Configures and activates the MAC peripheral.

Parameters

in	<i>macp</i>	pointer to the <code>MACDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.13 `mac_lld_stop()`

```
void mac_lld_stop (  
    MACDriver * macp)
```

Deactivates the MAC peripheral.

Parameters

in	<i>macp</i>	pointer to the <code>MACDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.14 `mac_lld_get_transmit_descriptor()`

```
msg_t mac_lld_get_transmit_descriptor (  
    MACDriver * macp,  
    MACTransmitDescriptor * tdp)
```

Returns a transmission descriptor.

One of the available transmission descriptors is locked and returned.

Parameters

in	<i>macp</i>	pointer to the <code>MACDriver</code> object
out	<i>tdp</i>	pointer to a <code>MACTransmitDescriptor</code> structure

Returns

The operation status.

Return values

<i>MSG_OK</i>	the descriptor has been obtained.
<i>MSG_TIMEOUT</i>	descriptor not available.

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.15 mac_lld_release_transmit_descriptor()

```
void mac_lld_release_transmit_descriptor (  
    MACTransmitDescriptor * tdp)
```

Releases a transmit descriptor and starts the transmission of the enqueued data as a single frame.

Parameters

in	<i>tdp</i>	the pointer to the MACTransmitDescriptor structure

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.16 mac_lld_get_receive_descriptor()

```
msg_t mac_lld_get_receive_descriptor (  
    MACDriver * macp,  
    MACReceiveDescriptor * rdp)
```

Returns a receive descriptor.

Parameters

in	<i>macp</i>	pointer to the MACDriver object
out	<i>rdp</i>	pointer to a MACReceiveDescriptor structure

Returns

The operation status.

Return values

<i>MSG_OK</i>	the descriptor has been obtained.
<i>MSG_TIMEOUT</i>	descriptor not available.

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.17 `mac_lld_release_receive_descriptor()`

```
void mac_lld_release_receive_descriptor (  
    MACReceiveDescriptor * rdp)
```

Releases a receive descriptor.

The descriptor and its buffer are made available for more incoming frames.

Parameters

in	<i>rdp</i>	the pointer to the MACReceiveDescriptor structure

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.18 mac_llt_poll_link_status()

```
bool mac_llt_poll_link_status (
    MACDriver * macp)
```

Updates and returns the link status.

Parameters

in	<i>macp</i>	pointer to the MACDriver object

Returns

The link status.

Return values

<i>true</i>	if the link is active.
<i>false</i>	if the link is down.

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.19 mac_llt_write_transmit_descriptor()

```
size_t mac_llt_write_transmit_descriptor (
    MACTransmitDescriptor * tdp,
    uint8_t * buf,
    size_t size)
```

Writes to a transmit descriptor's stream.

Parameters

in	<i>tdp</i>	pointer to a MACTransmitDescriptor structure
in	<i>buf</i>	pointer to the buffer containing the data to be written
in	<i>size</i>	number of bytes to be written

Returns

The number of bytes written into the descriptor's stream, this value can be less than the amount specified in the parameter *size* if the maximum frame size is reached.

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.20 `mac_llid_read_receive_descriptor()`

```
size_t mac_llid_read_receive_descriptor (  
    MACReceiveDescriptor * rdp,  
    uint8_t * buf,  
    size_t size)
```

Reads from a receive descriptor's stream.

Parameters

in	<i>rdp</i>	pointer to a MACReceiveDescriptor structure
in	<i>buf</i>	pointer to the buffer that will receive the read data
in	<i>size</i>	number of bytes to be read

Returns

The number of bytes read from the descriptor's stream, this value can be less than the amount specified in the parameter `size` if there are no more bytes to read.

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.21 `mac_llid_get_next_transmit_buffer()`

```
uint8_t * mac_llid_get_next_transmit_buffer (  
    MACTransmitDescriptor * tdp,  
    size_t size,  
    size_t * sizep)
```

Returns a pointer to the next transmit buffer in the descriptor chain.

Note

The API guarantees that enough buffers can be requested to fill a whole frame.

Parameters

in	<i>tdp</i>	pointer to a MACTransmitDescriptor structure
in	<i>size</i>	size of the requested buffer. Specify the frame size on the first call then scale the value down subtracting the amount of data already copied into the previous buffers.
out	<i>sizep</i>	pointer to variable receiving the buffer size, it is zero when the last buffer has already been returned. Note that a returned size lower than the amount requested means that more buffers must be requested in order to fill the frame data entirely.

Returns

Pointer to the returned buffer.

Return values

<code>NULL</code>	if the buffer chain has been entirely scanned.

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.5.22 `mac_lld_get_next_receive_buffer()`

```
const uint8_t * mac_lld_get_next_receive_buffer (  
    MACReceiveDescriptor * rdp,  
    size_t * sizep)
```

Returns a pointer to the next receive buffer in the descriptor chain.

Note

The API guarantees that the descriptor chain contains a whole frame.

Parameters

in	<code>rdp</code>	pointer to a <code>MACReceiveDescriptor</code> structure
out	<code>sizep</code>	pointer to variable receiving the buffer size, it is zero when the last buffer has already been returned.

Returns

Pointer to the returned buffer.

Return values

<code>NULL</code>	if the buffer chain has been entirely scanned.

Function Class:

Not an API, this function is for internal use only.

7.2.9.12.6 Variable Documentation**7.2.9.12.6.1 `ETHD1`**

`MACDriver` `ETHD1`

MAC1 driver identifier.

7.2.9.13 PAL Driver

I/O Ports Abstraction Layer.

7.2.9.13.1 Detailed Description

I/O Ports Abstraction Layer.

This module defines an abstract interface for digital I/O ports. Note that most I/O ports functions are just macros. The macros have default software implementations that can be redefined in a PAL Low Level Driver if the target hardware supports special features like, for example, atomic bit set/reset/masking. Please refer to the ports specific documentation for details.

The [PAL Driver](#) driver has the advantage to make the access to the I/O ports platform independent and still be optimized for the specific architectures.

Note that the PAL Low Level Driver may also offer non standard macro and functions in order to support specific features but, of course, the use of such interfaces would not be portable. Such interfaces shall be marked with the architecture name inside the function names.

Precondition

In order to use the PAL driver the `HAL_USE_PAL` option must be enabled in `halconf.h`.

7.2.9.13.2 Implementation Rules

In implementing a PAL Low Level Driver there are some rules/behaviors that should be respected.

7.2.9.13.2.1 Writing on input pads

The behavior is not specified but there are implementations better than others, this is the list of possible implementations, preferred options are on top:

1. The written value is not actually output but latched, should the pads be reprogrammed as outputs the value would be in effect.
2. The write operation is ignored.
3. The write operation has side effects, as example disabling/enabling pull up/down resistors or changing the pad direction. This scenario is discouraged, please try to avoid this scenario.

7.2.9.13.2.2 Reading from output pads

The behavior is not specified but there are implementations better than others, this is the list of possible implementations, preferred options are on top:

1. The actual pads states are read (not the output latch).
2. The output latch value is read (regardless of the actual pads states).
3. Unspecified, please try to avoid this scenario.

7.2.9.13.2.3 Writing unused or unimplemented port bits

The behavior is not specified.

7.2.9.13.2.4 Reading from unused or unimplemented port bits

The behavior is not specified.

7.2.9.13.2.5 Reading or writing on pins associated to other functionalities

The behavior is not specified.

Pads mode constants

- #define `PAL_MODE_RESET` 0U
After reset state.
- #define `PAL_MODE_UNCONNECTED` 1U
*Safe state for **unconnected** pads.*
- #define `PAL_MODE_INPUT` 2U
Regular input high-Z pad.
- #define `PAL_MODE_INPUT_PULLUP` 3U
Input pad with weak pull up resistor.
- #define `PAL_MODE_INPUT_PULLDOWN` 4U
Input pad with weak pull down resistor.
- #define `PAL_MODE_INPUT_ANALOG` 5U
Analog input mode.
- #define `PAL_MODE_OUTPUT_PUSHPULL` 6U
Push-pull output pad.
- #define `PAL_MODE_OUTPUT_OPENDRAIN` 7U
Open-drain output pad.

Logic level constants

- #define `PAL_LOW` 0U
Logical low state.
- #define `PAL_HIGH` 1U
Logical high state.

PAL event modes

- #define `PAL_EVENT_MODE_EDGES_MASK` 3U
Mask of edges field.
- #define `PAL_EVENT_MODE_DISABLED` 0U
Channel disabled.
- #define `PAL_EVENT_MODE_RISING_EDGE` 1U
Rising edge callback.
- #define `PAL_EVENT_MODE_FALLING_EDGE` 2U
Falling edge callback.
- #define `PAL_EVENT_MODE_BOTH_EDGES` 3U
Both edges callback.

PAL configuration options

- `#define PAL_USE_CALLBACKS TRUE`
Enables synchronous APIs.
- `#define PAL_USE_WAIT TRUE`
Enables synchronous APIs.

Low level driver helper macros

- `#define _pal_init_event(e)`
Initializes a PAL event object.
- `#define _pal_init_event(e)`
Initializes a PAL event object.
- `#define _pal_init_event(e)`
Initializes a PAL event object.
- `#define _pal_clear_event(e)`
Clears a PAL event object.
- `#define _pal_clear_event(e)`
Clears a PAL event object.
- `#define _pal_clear_event(e)`
Clears a PAL event object.
- `#define _pal_isr_code(e)`
Common ISR code.
- `#define _pal_isr_code(e)`
Common ISR code.
- `#define _pal_isr_code(e)`
Common ISR code.

Macro Functions

- `#define palInit()`
PAL subsystem initialization.
- `#define palReadPort(port)`
Reads the physical I/O port states.
- `#define palReadLatch(port)`
Reads the output latch.
- `#define palWritePort(port, bits)`
Writes a bits mask on a I/O port.
- `#define palSetPort(port, bits)`
Sets a bits mask on a I/O port.
- `#define palClearPort(port, bits)`
Clears a bits mask on a I/O port.
- `#define palTogglePort(port, bits)`
Toggles a bits mask on a I/O port.
- `#define palReadGroup(port, mask, offset)`
Reads a group of bits.
- `#define palReadGroupLatch(port, mask, offset)`
Reads the group latch.
- `#define palWriteGroup(port, mask, offset, bits)`
Writes a group of bits.

- #define `palSetGroupMode`(port, mask, offset, mode)
Pads group mode setup.
- #define `palReadPad`(port, pad)
Reads an input pad logic state.
- #define `palWritePad`(port, pad, bit)
Writes a logic state on an output pad.
- #define `palSetPad`(port, pad)
Sets a pad logic state to `PAL_HIGH`.
- #define `palClearPad`(port, pad)
Clears a pad logic state to `PAL_LOW`.
- #define `palTogglePad`(port, pad)
Toggles a pad logic state.
- #define `palSetPadMode`(port, pad, mode)
Pad mode setup.
- #define `palReadLine`(line)
Reads an input line logic state.
- #define `palWriteLine`(line, bit)
Writes a logic state on an output line.
- #define `palSetLine`(line)
Sets a line logic state to `PAL_HIGH`.
- #define `palClearLine`(line)
Clears a line logic state to `PAL_LOW`.
- #define `palToggleLine`(line)
Toggles a line logic state.
- #define `palSetLineMode`(line, mode)
Line mode setup.
- #define `palEnablePadEventI`(port, pad, mode)
Pad event enable.
- #define `palDisablePadEventI`(port, pad)
Pad event disable.
- #define `palEnablePadEvent`(port, pad, mode)
Pad event enable.
- #define `palDisablePadEvent`(port, pad)
Pad event disable.
- #define `palEnableLineEventI`(line, mode)
Line event enable.
- #define `palDisableLineEventI`(line)
Line event disable.
- #define `palEnableLineEvent`(line, mode)
Line event enable.
- #define `palDisableLineEvent`(line)
Line event disable.
- #define `palIsPadEventEnabledX`(port, pad)
Pad event enable check.
- #define `palIsLineEventEnabledX`(line)
Line event enable check.
- #define `palSetPadCallback`(port, pad, cb, arg)
Associates a callback to a pad.
- #define `palSetLineCallback`(line, cb, arg)
Associates a callback to a line.

Port related definitions

- #define `PAL_IOPORTS_WIDTH` 16U
Width, in bits, of an I/O port.
- #define `PAL_WHOLE_PORT` ((ioportmask_t)0xFFFFU)
Whole port mask.

Line handling macros

- #define `PAL_LINE`(port, pad)
Forms a line identifier.
- #define `PAL_PORT`(line)
Decodes a port identifier from a line identifier.
- #define `PAL_PAD`(line)
Decodes a pad identifier from a line identifier.
- #define `PAL_NOLINE` 0U
Value identifying an invalid line.

Data Structures

- struct `palevent_t`
Type of a PAL event record.
- struct `IOBus`
I/O bus descriptor.
- struct `PALConfig`
Generic I/O ports static initializer.

Macros

- #define `PAL_PORT_BIT`(n)
Port bit helper macro.
- #define `PAL_GROUP_MASK`(width)
Bits group mask helper.
- #define `_IOBUS_DATA`(name, port, width, offset)
Data part of a static I/O bus initializer.
- #define `IOBUS_DECL`(name, port, width, offset)
Static I/O bus initializer.
- #define `PAL_NEW_INIT`
- #define `IOPORT1` 0
First I/O port identifier.
- #define `pal_ild_init`()
Low level PAL subsystem initialization.
- #define `pal_ild_readport`(port)
Reads the physical I/O port states.
- #define `pal_ild_readlatch`(port)
Reads the output latch.
- #define `pal_ild_writeport`(port, bits)
Writes a bits mask on a I/O port.
- #define `pal_ild_setport`(port, bits)

- Sets a bits mask on a I/O port.*

 - #define `pal_lld_clearport`(port, bits)
- Clears a bits mask on a I/O port.*

 - #define `pal_lld_toggleport`(port, bits)
- Toggles a bits mask on a I/O port.*

 - #define `pal_lld_readgroup`(port, mask, offset)
- Reads a group of bits.*

 - #define `pal_lld_writegroup`(port, mask, offset, bits)
- Writes a group of bits.*

 - #define `pal_lld_setgroupmode`(port, mask, offset, mode)
- Pads group mode setup.*

 - #define `pal_lld_readpad`(port, pad)
- Reads a logical state from an I/O pad.*

 - #define `pal_lld_writepad`(port, pad, bit)
- Writes a logical state on an output pad.*

 - #define `pal_lld_setpad`(port, pad)
- Sets a pad logical state to `PAL_HIGH`.*

 - #define `pal_lld_clearpad`(port, pad)
- Clears a pad logical state to `PAL_LOW`.*

 - #define `pal_lld_togglepad`(port, pad)
- Toggles a pad logical state.*

 - #define `pal_lld_setpadmode`(port, pad, mode)
- Pad mode setup.*

 - #define `pal_lld_get_pad_event`(port, pad)
- Returns a PAL event structure associated to a pad.*

 - #define `pal_lld_get_line_event`(line)
- Returns a PAL event structure associated to a line.*

Typedefs

- typedef void(* `palcallback_t`) (void *arg)

Type of a PAL event callback.
- typedef uint32_t `ioportmask_t`

Digital I/O port sized unsigned type.
- typedef uint32_t `iomode_t`

Digital I/O modes.
- typedef uint32_t `ioline_t`

Type of an I/O line.
- typedef uint32_t `ioportid_t`

Port Identifier.
- typedef uint32_t `iopadid_t`

Type of an pad identifier.

Functions

- `ioportmask_t palReadBus` (const `IOBus` *bus)
Read from an I/O bus.
- void `palWriteBus` (const `IOBus` *bus, `ioportmask_t` bits)
Write to an I/O bus.
- void `palSetBusMode` (const `IOBus` *bus, `iomode_t` mode)
Programs a bus with the specified mode.
- void `palSetPadCallbackI` (`ioportid_t` port, `iopadid_t` pad, `palcallback_t` cb, void *arg)
Associates a callback to a port/pad.
- void `palSetLineCallbackI` (`ioline_t` line, `palcallback_t` cb, void *arg)
Associates a callback to a line.
- `msg_t palWaitPadTimeoutS` (`ioportid_t` port, `iopadid_t` pad, `sysinterval_t` timeout)
Waits for an edge on the specified port/pad.
- `msg_t palWaitPadTimeout` (`ioportid_t` port, `iopadid_t` pad, `sysinterval_t` timeout)
Waits for an edge on the specified port/pad.
- `msg_t palWaitLineTimeoutS` (`ioline_t` line, `sysinterval_t` timeout)
Waits for an edge on the specified line.
- `msg_t palWaitLineTimeout` (`ioline_t` line, `sysinterval_t` timeout)
Waits for an edge on the specified line.
- void `_pal_lld_init` (void)
STM32 I/O ports configuration.
- void `_pal_lld_setgroupmode` (`ioportid_t` port, `ioportmask_t` mask, `iomode_t` mode)
Pads mode setup.

7.2.9.13.3 Macro Definition Documentation

7.2.9.13.3.1 PAL_MODE_RESET

```
#define PAL_MODE_RESET 0U
```

After reset state.

The state itself is not specified and is architecture dependent, it is guaranteed to be equal to the after-reset state. It is usually an input state.

7.2.9.13.3.2 PAL_MODE_UNCONNECTED

```
#define PAL_MODE_UNCONNECTED 1U
```

Safe state for **unconnected** pads.

The state itself is not specified and is architecture dependent, it may be mapped on `PAL_MODE_INPUT_PULLUP`, `PAL_MODE_INPUT_PULLDOWN` or `PAL_MODE_OUTPUT_PUSHPULL` for example.

7.2.9.13.3.3 PAL_MODE_INPUT

```
#define PAL_MODE_INPUT 2U
```

Regular input high-Z pad.

7.2.9.13.3.4 PAL_MODE_INPUT_PULLUP

```
#define PAL_MODE_INPUT_PULLUP 3U
```

Input pad with weak pull up resistor.

7.2.9.13.3.5 PAL_MODE_INPUT_PULLDOWN

```
#define PAL_MODE_INPUT_PULLDOWN 4U
```

Input pad with weak pull down resistor.

7.2.9.13.3.6 PAL_MODE_INPUT_ANALOG

```
#define PAL_MODE_INPUT_ANALOG 5U
```

Analog input mode.

7.2.9.13.3.7 PAL_MODE_OUTPUT_PUSHPULL

```
#define PAL_MODE_OUTPUT_PUSHPULL 6U
```

Push-pull output pad.

7.2.9.13.3.8 PAL_MODE_OUTPUT_OPENDRAIN

```
#define PAL_MODE_OUTPUT_OPENDRAIN 7U
```

Open-drain output pad.

7.2.9.13.3.9 PAL_LOW

```
#define PAL_LOW 0U
```

Logical low state.

7.2.9.13.3.10 PAL_HIGH

```
#define PAL_HIGH 1U
```

Logical high state.

7.2.9.13.3.11 PAL_EVENT_MODE_EDGES_MASK

```
#define PAL_EVENT_MODE_EDGES_MASK 3U
```

Mask of edges field.

7.2.9.13.3.12 PAL_EVENT_MODE_DISABLED

```
#define PAL_EVENT_MODE_DISABLED 0U
```

Channel disabled.

7.2.9.13.3.13 PAL_EVENT_MODE_RISING_EDGE

```
#define PAL_EVENT_MODE_RISING_EDGE 1U
```

Rising edge callback.

7.2.9.13.3.14 PAL_EVENT_MODE_FALLING_EDGE

```
#define PAL_EVENT_MODE_FALLING_EDGE 2U
```

Falling edge callback.

7.2.9.13.3.15 PAL_EVENT_MODE_BOTH_EDGES

```
#define PAL_EVENT_MODE_BOTH_EDGES 3U
```

Both edges callback.

7.2.9.13.3.16 PAL_USE_CALLBACKS

```
#define PAL_USE_CALLBACKS TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.9.13.3.17 PAL_USE_WAIT

```
#define PAL_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.9.13.3.18 PAL_PORT_BIT

```
#define PAL_PORT_BIT(  
    n)
```

Value:

```
((ioportmask_t)(1U << (n)))
```

Port bit helper macro.

This macro calculates the mask of a bit within a port.

Parameters

in	<i>n</i>	bit position within the port

Returns

The bit mask.

7.2.9.13.3.19 PAL_GROUP_MASK

```
#define PAL_GROUP_MASK(  
    width)
```

Value:

```
((ioportmask_t)(1U << (width)) - 1U)
```

Bits group mask helper.

This macro calculates the mask of a bits group.

Parameters

in	<i>width</i>	group width

Returns

The group mask.

7.2.9.13.3.20 _IOBUS_DATA

```
#define _IOBUS_DATA(  
    name,  
    port,  
    width,  
    offset)
```

Value:

```
{port, PAL_GROUP_MASK(width), offset}
```

Data part of a static I/O bus initializer.

This macro should be used when statically initializing an I/O bus that is part of a bigger structure.

Parameters

in	<i>name</i>	name of the IOBus variable
in	<i>port</i>	I/O port descriptor
in	<i>width</i>	bus width in bits
in	<i>offset</i>	bus bit offset within the port

Parameters

in	<i>e</i>	event index

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.24 _pal_init_event [3/3]

```
#define _pal_init_event(  
    e)
```

Value:

```
do {
    osalThreadQueueObjectInit(&_pal_events[e].threads);
} while (false)
```

Initializes a PAL event object.

Parameters

in	e	event index

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.25 _pal_clear_event [1/3]

```
#define _pal_clear_event(  
    e)
```

Value:

```
do {
    osalThreadDequeueAllI(&_pal_events[pad].threads, MSG_RESET);
    _pal_events[e].cb = NULL;
    _pal_events[e].arg = NULL;
} while (false)
```

Clears a PAL event object.

Parameters

in	e	event index

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.26 `_pal_clear_event` [2/3]

```
#define _pal_clear_event(  
    e)
```

Value:

```
do {
    _pal_events[e].cb = NULL;
    _pal_events[e].arg = NULL;
} while (false)
```

Clears a PAL event object.

Parameters

in	<i>e</i>	event index

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.27 _pal_clear_event [3/3]

```
#define _pal_clear_event(  
    e)
```

Value:

```
do {  
    osalThreadDequeueAllI(&_pal_events[e].threads, MSG_RESET);  
} while (false)
```

```
\  
\  

```

Clears a PAL event object.

Parameters

in	<i>e</i>	event index

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.28 _pal_isr_code [1/3]

```
#define _pal_isr_code(  
    e)
```

Value:

```
do {  
    if (_pal_events[e].cb != NULL) {  
        _pal_events[e].cb(_pal_events[e].arg);  
    }  
    osalSysLockFromISR();  
    osalThreadDequeueAllI(&_pal_events[e].threads, MSG_OK);  
    osalSysUnlockFromISR();  
} while (false)
```

```
/  
/  
/  
/  
/  
/  

```

Common ISR code.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>e</i>	event index

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.29 `_pal_isr_code` [2/3]

```
#define _pal_isr_code(
    e)
```

Value:

```
do {
    if (_pal_events[e].cb != NULL) {
        _pal_events[e].cb(_pal_events[e].arg);
    }
} while (false)
```

Common ISR code.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

<code>in</code>	<code>e</code>	event index

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.30 `_pal_isr_code` [3/3]

```
#define _pal_isr_code(
    e)
```

Value:

```
do {
    osalSysLockFromISR();
    osalThreadDequeueAllI(&_pal_events[e].threads, MSG_OK);
    osalSysUnlockFromISR();
} while (false)
```

Common ISR code.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

<code>in</code>	<code>e</code>	event index

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.31 palInit

```
#define palInit()
```

Value:

```
pal_lld_init()
```

PAL subsystem initialization.

Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

7.2.9.13.3.32 palReadPort

```
#define palReadPort(  
    port)
```

Value:

```
((void) (port), 0U)
```

Reads the physical I/O port states.

Note

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier

Returns

The port logic states.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.33 palReadLatch

```
#define palReadLatch(  
    port)
```

Value:

```
((void) (port), 0U)
```

Reads the output latch.

The purpose of this function is to read back the latched output value.

Note

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier

Returns

The latched logic states.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.34 palWritePort

```
#define palWritePort(
    port,
    bits)
```

Value:

```
((void) (port), (void) (bits))
```

Writes a bits mask on a I/O port.

Note

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>bits</i>	bits to be written on the specified port

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.35 palSetPort

```
#define palSetPort(
    port,
    bits)
```

Value:

```
palWritePort(port, palReadLatch(port) | (bits))
```

Sets a bits mask on a I/O port.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>bits</i>	bits to be ORed on the specified port

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.36 palClearPort

```
#define palClearPort(
    port,
    bits)
```

Value:

```
palWritePort(port, palReadLatch(port) & ~(bits))
```

Clears a bits mask on a I/O port.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>bits</i>	bits to be cleared on the specified port

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.37 palTogglePort

```
#define palTogglePort(
    port,
    bits)
```

Value:

```
palWritePort(port, palReadLatch(port) ^ (bits))
```

Toggles a bits mask on a I/O port.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>bits</i>	bits to be XORed on the specified port

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.38 palReadGroup

```
#define palReadGroup(  
    port,  
    mask,  
    offset)
```

Value:

```
((palReadPort(port) » (offset)) & (mask))
```

Reads a group of bits.

Note

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>mask</i>	group mask, a logic AND is performed on the input data
in	<i>offset</i>	group bit offset within the port

Returns

The group logic states.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.39 palReadGroupLatch

```
#define palReadGroupLatch(  
    port,  
    mask,  
    offset)
```

Value:

```
((palReadLatch(port) » (offset)) & (mask))
```

Reads the group latch.

Note

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>mask</i>	group mask, a logic AND is performed on the input data
in	<i>offset</i>	group bit offset within the port

Returns

The group logic states.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.40 palWriteGroup

```
#define palWriteGroup(
    port,
    mask,
    offset,
    bits)
```

Value:

```
palWritePort(port, (palReadLatch(port) & ~((mask) << (offset))) | \
                (((bits) & (mask)) << (offset)))
```

Writes a group of bits.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>mask</i>	group mask, a logic AND is performed on the output data
in	<i>offset</i>	group bit offset within the port
in	<i>bits</i>	bits to be written. Values exceeding the group width are masked.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.41 palSetGroupMode

```
#define palSetGroupMode(
    port,
    mask,
    offset,
    mode)
```

Value:

```
do {
    (void) (port);
    (void) (mask);
    (void) (offset);
    (void) (port);
} while (mode)
```

Pads group mode setup.

This function programs a pads group belonging to the same port with the specified mode.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

Programming an unknown or unsupported mode is silently ignored.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>mask</i>	group mask
in	<i>offset</i>	group bit offset within the port
in	<i>mode</i>	group mode

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.42 palReadPad

```
#define palReadPad(
    port,
    pad)
```

Value:

```
((palReadPort(port) >> (pad)) & 1U)
```

Reads an input pad logic state.

Note

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Returns

The logic state.

Return values

<i>PAL_LOW</i>	low logic state.
<i>PAL_HIGH</i>	high logic state.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.43 palWritePad

```
#define palWritePad(
    port,
    pad,
    bit)
```

Value:

```
palWritePort(port, (palReadLatch(port) & ~PAL_PORT_BIT(pad)) | \
    ((bit) & 1U) << pad)
```

Writes a logic state on an output pad.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>bit</i>	logic value, the value must be <code>PAL_LOW</code> or <code>PAL_HIGH</code>

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.44 palSetPad

```
#define palSetPad(  
    port,  
    pad)
```

Value:

`palSetPort(port, PAL_PORT_BIT(pad))`

Sets a pad logic state to `PAL_HIGH`.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.45 palClearPad

```
#define palClearPad(  
    port,  
    pad)
```

Value:

`palClearPort(port, PAL_PORT_BIT(pad))`

Clears a pad logic state to `PAL_LOW`.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.46 palTogglePad

```
#define palTogglePad(  
    port,  
    pad)
```

Value:

```
palTogglePort(port, PAL_PORT_BIT(pad))
```

Toggles a pad logic state.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.47 palSetPadMode

```
#define palSetPadMode(  
    port,  
    pad,  
    mode)
```

Value:

```
palSetGroupMode(port, PAL_PORT_BIT(pad), 0U, mode)
```

Pad mode setup.

This function programs a pad with the specified mode.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

Programming an unknown or unsupported mode is silently ignored.

The function can be called from any context.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>mode</i>	pad mode

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.48 palReadLine

```
#define palReadLine(  
    line)
```

Value:

```
palReadPad(PAL_PORT(line), PAL_PAD(line))
```

Reads an input line logic state.

Note

The function can be called from any context.

Parameters

in	<i>line</i>	line identifier

Returns

The logic state.

Return values

<i>PAL_LOW</i>	low logic state.
<i>PAL_HIGH</i>	high logic state.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.49 palWriteLine

```
#define palWriteLine(  
    line,  
    bit)
```

Value:

```
palWritePad(PAL_PORT(line), PAL_PAD(line), bit)
```

Writes a logic state on an output line.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>line</i>	line identifier
in	<i>bit</i>	logic value, the value must be PAL_LOW or PAL_HIGH

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.50 palSetLine

```
#define palSetLine(  
    line)
```

Value:

```
palSetPad(PAL_PORT(line), PAL_PAD(line))
```

Sets a line logic state to PAL_HIGH.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>line</i>	line identifier

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.51 palClearLine

```
#define palClearLine(  
    line)
```

Value:

```
palClearPad(PAL_PORT(line), PAL_PAD(line))
```

Clears a line logic state to PAL_LOW.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>line</i>	line identifier

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.52 palToggleLine

```
#define palToggleLine(  
    line)
```

Value:

```
palTogglePad(PAL_PORT(line), PAL_PAD(line))
```

Toggles a line logic state.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>line</i>	line identifier

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.53 palSetLineMode

```
#define palSetLineMode(  
    line,  
    mode)
```

Value:

```
palSetPadMode(PAL_PORT(line), PAL_PAD(line), mode)
```

Line mode setup.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function can be called from any context.

Parameters

in	<i>line</i>	line identifier
in	<i>mode</i>	pad mode

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.3.54 palEnablePadEventI

```
#define palEnablePadEventI(  
    port,  
    pad,  
    mode)
```

Pad event enable.

Note

Programming an unknown or unsupported mode is silently ignored.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>mode</i>	pad event mode

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.13.3.55 palDisablePadEventI

```
#define palDisablePadEventI(  
    port,  
    pad)
```

Pad event disable.

This function also disables previously programmed event callbacks.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.13.3.56 palEnablePadEvent

```
#define palEnablePadEvent(  
    port,  
    pad,  
    mode)
```

Value:

```
do {
    osalSysLock();
    palEnablePadEventI(port, pad, mode);
    osalSysUnlock();
} while (false)
```

1111

Pad event enable.

Note

Programming an unknown or unsupported mode is silently ignored.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>mode</i>	pad event mode

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.13.3.57 palDisablePadEvent

```
#define palDisablePadEvent(  
    port,  
    pad)
```

Value:

```
do {
    osalSysLock();
    palDisablePadEventI(port, pad);
    osalSysUnlock();
} while (false)
```

/ / / /

Pad event disable.

This function also disables previously programmed event callbacks.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.13.3.58 palEnableLineEventI

```
#define palEnableLineEventI(
    line,
    mode)
```

Value:

```
palEnablePadEventI(PAL_PORT(line), PAL_PAD(line), mode)
```

Line event enable.

Note

Programming an unknown or unsupported mode is silently ignored.

Parameters

in	<i>line</i>	line identifier
in	<i>mode</i>	line event mode

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.13.3.59 palDisableLineEventI

```
#define palDisableLineEventI(
    line)
```

Value:

```
palDisablePadEventI(PAL_PORT(line), PAL_PAD(line))
```

Line event disable.

This function also disables previously programmed event callbacks.

Parameters

in	<i>line</i>	line identifier

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.13.3.60 palEnableLineEvent

```
#define palEnableLineEvent(
    line,
    mode)
```

Value:

```
do {
    osalSysLock();
    palEnableLineEventI(line, mode);
    osalSysUnlock();
} while (false)
```

```
\\
\\
\\
```

Line event enable.

Note

Programming an unknown or unsupported mode is silently ignored.

Parameters

in	<i>line</i>	line identifier
in	<i>mode</i>	line event mode

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.13.3.61 palDisableLineEvent

```
#define palDisableLineEvent(  
    line)
```

Value:

```
do {  
    osalSysLock();  
    palDisableLineEventI(line);  
    osalSysUnlock();  
} while (false)
```

```
///  
///  
///  
///
```

Line event disable.

This function also disables previously programmed event callbacks.

Parameters

in	<i>line</i>	line identifier

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.13.3.62 palIsPadEventEnabledX

```
#define palIsPadEventEnabledX(  
    port,  
    pad)
```

Value:

```
false
```

Pad event enable check.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Returns

Pad event status.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>cb</i>	event callback function
in	<i>arg</i>	callback argument

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.13.3.65 palSetLineCallback

```
#define palSetLineCallback(
    line,
    cb,
    arg)
```

Value:

```
do {
    osalSysLock();
    palSetLineCallbackI(line, cb, arg);
    osalSysUnlock();
} while (false)
```

```

/
/
/
/
```

Associates a callback to a line.

Parameters

in	<i>line</i>	line identifier
in	<i>cb</i>	event callback function
in	<i>arg</i>	callback argument

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.13.3.66 PAL_NEW_INIT

```
#define PAL_NEW_INIT
```

7.2.9.13.3.67 PAL_IOPORTS_WIDTH

```
#define PAL_IOPORTS_WIDTH 16U
```

Width, in bits, of an I/O port.

7.2.9.13.3.68 PAL_WHOLE_PORT

```
#define PAL_WHOLE_PORT ((ioportmask_t) 0xFFFFU)
```

Whole port mask.

This macro specifies all the valid bits into a port.

7.2.9.13.3.69 PAL_LINE

```
#define PAL_LINE(  
    port,  
    pad)
```

Value:

```
((ioline_t)((uint32_t)(port)) | ((uint32_t)(pad)))
```

Forms a line identifier.

A port/pad pair are encoded into an `ioline_t` type. The encoding of this type is platform-dependent.

7.2.9.13.3.70 PAL_PORT

```
#define PAL_PORT(  
    line)
```

Value:

```
((stm32_gpio_t *)(((uint32_t)(line)) & 0xFFFFFFF0U))
```

Decodes a port identifier from a line identifier.

7.2.9.13.3.71 PAL_PAD

```
#define PAL_PAD(  
    line)
```

Value:

```
((uint32_t)((uint32_t)(line) & 0x0000000FU))
```

Decodes a pad identifier from a line identifier.

7.2.9.13.3.72 PAL_NOLINE

```
#define PAL_NOLINE 0U
```

Value identifying an invalid line.

7.2.9.13.3.73 IOPORT1

```
#define IOPORT1 0
```

First I/O port identifier.

Low level drivers can define multiple ports, it is suggested to use this naming convention.

7.2.9.13.3.74 pal_lld_init

```
#define pal_lld_init()
```

Value:

```
_pal_lld_init()
```

Low level PAL subsystem initialization.

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.75 pal_lld_readport

```
#define pal_lld_readport(  
    port)
```

Value:

```
0U
```

Reads the physical I/O port states.

Parameters

in	<i>port</i>	port identifier

Returns

The port bits.

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.76 pal_lld_readlatch

```
#define pal_lld_readlatch(  
    port)
```

Value:

```
0U
```

Reads the output latch.

The purpose of this function is to read back the latched output value.

Parameters

in	<i>port</i>	port identifier

Returns

The latched logical states.

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.77 pal_lld_writeport

```
#define pal_lld_writeport(
    port,
    bits)
```

Value:

```
do {
    (void)port;
    (void)bits;
} while (false)
```

```
\\
```

Writes a bits mask on a I/O port.

Parameters

in	<i>port</i>	port identifier
in	<i>bits</i>	bits to be written on the specified port

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.78 pal_lld_setport

```
#define pal_lld_setport(
    port,
    bits)
```

Value:

```
do {
    (void)port;
    (void)bits;
} while (false)
```

```
\\
```

Sets a bits mask on a I/O port.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>bits</i>	bits to be ORed on the specified port

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.79 pal_lld_clearport

```
#define pal_lld_clearport(  
    port,  
    bits)
```

Value:

```
do {  
    (void)port;  
    (void)bits;  
} while (false)
```

\

\

\

Clears a bits mask on a I/O port.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>bits</i>	bits to be cleared on the specified port

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.80 pal_lld_toggleport

```
#define pal_lld_toggleport(  
    port,  
    bits)
```

Value:

```
do {  
    (void)port;  
    (void)bits;  
} while (false)
```

\

\

\

Toggles a bits mask on a I/O port.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>bits</i>	bits to be XORed on the specified port

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.81 pal_lld_readgroup

```
#define pal_lld_readgroup(  
    port,  
    mask,  
    offset)
```

Value:

0U

Reads a group of bits.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>mask</i>	group mask
in	<i>offset</i>	group bit offset within the port

Returns

The group logical states.

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.82 pal_lld_writegroup

```
#define pal_lld_writegroup(  
    port,  
    mask,  
    offset,  
    bits)
```

Value:

```
do {  
    (void)port;  
    (void)mask;  
    (void)offset;  
    (void)bits;  
} while (false)
```

/
/
/
/
/

Writes a group of bits.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>mask</i>	group mask
in	<i>offset</i>	group bit offset within the port
in	<i>bits</i>	bits to be written. Values exceeding the group width are masked.

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.83 pal_lld_setgroupmode

```
#define pal_lld_setgroupmode(  
    port,  
    mask,  
    offset,  
    mode)
```

Value:

```
_pal_lld_setgroupmode(port, mask << offset, mode)
```

Pads group mode setup.

This function programs a pads group belonging to the same port with the specified mode.

Note

Programming an unknown or unsupported mode is silently ignored.

Parameters

in	<i>port</i>	port identifier
in	<i>mask</i>	group mask
in	<i>offset</i>	group bit offset within the port
in	<i>mode</i>	group mode

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.84 pal_lld_readpad

```
#define pal_lld_readpad(  
    port,  
    pad)
```

Value:

```
PAL_LOW
```

Reads a logical state from an I/O pad.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Returns

The logical state.

Return values

<i>PAL_LOW</i>	low logical state.
<i>PAL_HIGH</i>	high logical state.

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.85 pal_lld_writepad

```
#define pal_lld_writepad(  
    port,  
    pad,  
    bit)
```

Value:

```
do {  
    (void)port;  
    (void)pad;  
    (void)bit;  
} while (false)
```

```
///  
///  
///
```

Writes a logical state on an output pad.

Note

This function is not meant to be invoked directly by the application code.

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>bit</i>	logical value, the value must be <code>PAL_LOW</code> or <code>PAL_HIGH</code>

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.86 pal_lld_setpad

```
#define pal_lld_setpad(
    port,
    pad)
```

Value:

```
do {
    (void)port;
    (void)pad;
} while (false)
```

```
\
\
\
```

Sets a pad logical state to `PAL_HIGH`.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.87 pal_lld_clearpad

```
#define pal_lld_clearpad(
    port,
    pad)
```

Value:

```
do {
    (void)port;
    (void)pad;
} while (false)
```

```
\
\
\
```

Clears a pad logical state to `PAL_LOW`.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.88 pal_lld_togglepad

```
#define pal_lld_togglepad(  
    port,  
    pad)
```

Value:

```
do {  
    (void)port;  
    (void)pad;  
} while (false)
```

Toggles a pad logical state.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.89 pal_lld_setpadmode

```
#define pal_lld_setpadmode(  
    port,  
    pad,  
    mode)
```

Value:

```
do {  
    (void)port;  
    (void)pad;  
    (void)mode;  
} while (false)
```

Pad mode setup.

This function programs a pad with the specified mode.

Note

The [PAL Driver](#) provides a default software implementation of this functionality, implement this function if can optimize it by using special hardware functionalities or special coding.

Programming an unknown or unsupported mode is silently ignored.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>mode</i>	pad mode

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.90 pal_lld_get_pad_event

```
#define pal_lld_get_pad_event(  
    port,  
    pad)
```

Value:

```
&_pal_events[0]; (void) (port); (void) pad
```

Returns a PAL event structure associated to a pad.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.3.91 pal_lld_get_line_event

```
#define pal_lld_get_line_event(  
    line)
```

Value:

```
&_pal_events[0]; (void) line
```

Returns a PAL event structure associated to a line.

Parameters

in	<i>line</i>	line identifier

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.4 Typedef Documentation

7.2.9.13.4.1 palcallback_t

```
typedef void(* palcallback_t) (void *arg)
```

Type of a PAL event callback.

7.2.9.13.4.2 ioportmask_t

```
typedef uint32_t ioportmask_t
```

Digital I/O port sized unsigned type.

7.2.9.13.4.3 iomode_t

```
typedef uint32_t iomode_t
```

Digital I/O modes.

7.2.9.13.4.4 ioline_t

```
typedef uint32_t ioline_t
```

Type of an I/O line.

7.2.9.13.4.5 ioportid_t

```
typedef uint32_t ioportid_t
```

Port Identifier.

This type can be a scalar or some kind of pointer, do not make any assumption about it, use the provided macros when populating variables of this type.

7.2.9.13.4.6 iopadid_t

```
typedef uint32_t iopadid_t
```

Type of an pad identifier.

7.2.9.13.5 Function Documentation

7.2.9.13.5.1 palReadBus()

```
ioportmask_t palReadBus (  
    const IOBus * bus)
```

Read from an I/O bus.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between `osalSysLock()` and `osalSysUnlock()`.

The function internally uses the `palReadGroup()` macro. The use of this function is preferred when you value code size, readability and error checking over speed.

The function can be called from any context.

Parameters

in	<i>bus</i>	the I/O bus, pointer to a IOBus structure

Returns

The bus logical states.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.5.2 palWriteBus()

```
void palWriteBus (
    const IOBus * bus,
    ioportmask_t bits)
```

Write to an I/O bus.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between [osalSysLock\(\)](#) and [osalSysUnlock\(\)](#).

The default implementation is non atomic and not necessarily optimal. Low level drivers may optimize the function by using specific hardware or coding.

The function can be called from any context.

Parameters

in	<i>bus</i>	the I/O bus, pointer to a IOBus structure
in	<i>bits</i>	the bits to be written on the I/O bus. Values exceeding the bus width are masked so most significant bits are lost.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.5.3 palSetBusMode()

```
void palSetBusMode (
    const IOBus * bus,
    iomode_t mode)
```

Programs a bus with the specified mode.

Note

The operation is not guaranteed to be atomic on all the architectures, for atomicity and/or portability reasons you may need to enclose port I/O operations between [osalSysLock\(\)](#) and [osalSysUnlock\(\)](#).

The default implementation is non atomic and not necessarily optimal. Low level drivers may optimize the function by using specific hardware or coding.

The function can be called from any context.

Parameters

in	<i>bus</i>	the I/O bus, pointer to a IOBus structure
in	<i>mode</i>	the mode

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.13.5.4 palSetPadCallbackI()

```
void palSetPadCallbackI (
    ioportid_t port,
    iopadid_t pad,
    palcallback_t cb,
    void * arg)
```

Associates a callback to a port/pad.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>cb</i>	event callback function
in	<i>arg</i>	callback argument

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.13.5.5 palSetLineCallbackI()

```
void palSetLineCallbackI (
    ioline_t line,
    palcallback_t cb,
    void * arg)
```

Associates a callback to a line.

Parameters

in	<i>line</i>	line identifier
in	<i>cb</i>	event callback function
in	<i>arg</i>	callback argument

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.13.5.6 palWaitPadTimeoutS()

```
msg_t palWaitPadTimeoutS (
    ioportid_t port,
    iopadid_t pad,
    sysinterval_t timeout)
```

Waits for an edge on the specified port/pad.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation state.

Return values

<i>MSG_OK</i>	if an edge has been detected.
<i>MSG_TIMEOUT</i>	if a timeout occurred before an edge could be detected.
<i>MSG_RESET</i>	if the event has been disabled while the thread was waiting for an edge.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

Here is the call graph for this function:



7.2.9.13.5.7 palWaitPadTimeout()

```
msg_t palWaitPadTimeout (
    ioportid_t port,
    iopadid_t pad,
    sysinterval_t timeout)
```

Waits for an edge on the specified port/pad.

Parameters

in	<i>port</i>	port identifier
in	<i>pad</i>	pad number within the port
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation state.

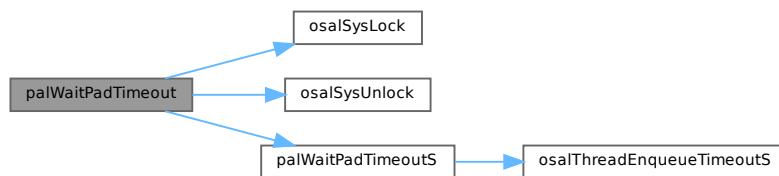
Return values

<i>MSG_OK</i>	if an edge has been detected.
<i>MSG_TIMEOUT</i>	if a timeout occurred before an edge could be detected.
<i>MSG_RESET</i>	if the event has been disabled while the thread was waiting for an edge.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.13.5.8 palWaitLineTimeoutS()**

```

msg_t palWaitLineTimeoutS (
    ioline_t line,
    sysinterval_t timeout)

```

Waits for an edge on the specified line.

Parameters

in	<i>line</i>	line identifier
in	<i>timeout</i>	operation timeout

Returns

The operation state.

Return values

<i>MSG_OK</i>	if an edge has been detected.
<i>MSG_TIMEOUT</i>	if a timeout occurred before an edge could be detected.
<i>MSG_RESET</i>	if the event has been disabled while the thread was waiting for an edge.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

Here is the call graph for this function:



7.2.9.13.5.9 palWaitLineTimeout()

```

msg_t palWaitLineTimeout (
    ioline_t line,
    sysinterval_t timeout)
  
```

Waits for an edge on the specified line.

Parameters

in	<i>line</i>	line identifier
in	<i>timeout</i>	operation timeout

Returns

The operation state.

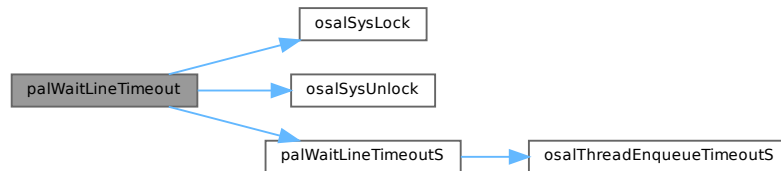
Return values

<i>MSG_OK</i>	if an edge has been detected.
<i>MSG_TIMEOUT</i>	if a timeout occurred before an edge could be detected.
<i>MSG_RESET</i>	if the event has been disabled while the thread was waiting for an edge.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.13.5.10 _pal_lld_init()**

```
void _pal_lld_init (
    void )
```

STM32 I/O ports configuration.

Function Class:

Not an API, this function is for internal use only.

7.2.9.13.5.11 _pal_lld_setgroupmode()

```
void _pal_lld_setgroupmode (
    ioportid_t port,
    ioportmask_t mask,
    iomode_t mode)
```

Pads mode setup.

This function programs a pads group belonging to the same port with the specified mode.

Parameters

in	<i>port</i>	the port identifier
in	<i>mask</i>	the group mask
in	<i>mode</i>	the mode

Function Class:

Not an API, this function is for internal use only.

7.2.9.14 PWM Driver

Generic PWM Driver.

7.2.9.14.1 Detailed Description

Generic PWM Driver.

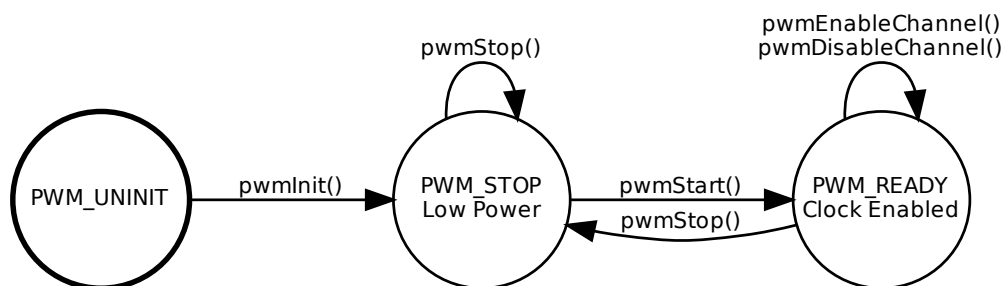
This module implements a generic PWM (Pulse Width Modulation) driver.

Precondition

In order to use the PWM driver the `HAL_USE_PWM` option must be enabled in `halconf.h`.

7.2.9.14.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



7.2.9.14.3 PWM Operations.

This driver abstracts a generic PWM timer composed of:

- A clock prescaler.
- A main up counter.
- A comparator register that resets the main counter to zero when the limit is reached. An optional callback can be generated when this happens.
- An array of `PWM_CHANNELS` PWM channels, each channel has an output, a comparator and is able to invoke an optional callback when a comparator match with the main counter happens.

A PWM channel output can be in two different states:

- **IDLE**, when the channel is disabled or after a match occurred.
- **ACTIVE**, when the channel is enabled and a match didn't occur yet in the current PWM cycle.

Note that the two states can be associated to both logical zero or one in the `PWMChannelConfig` structure.

PWM output mode macros

- #define `PWM_OUTPUT_MASK` 0x0FU
Standard output modes mask.
- #define `PWM_OUTPUT_DISABLED` 0x00U
Output not driven, callback only.
- #define `PWM_OUTPUT_ACTIVE_HIGH` 0x01U
Positive PWM logic, active is logic level one.
- #define `PWM_OUTPUT_ACTIVE_LOW` 0x02U
Inverse PWM logic, active is logic level zero.

PWM duty cycle conversion

- #define `PWM_FRACTION_TO_WIDTH`(pwmp, denominator, numerator)
Converts from fraction to pulse width.
- #define `PWM_DEGREES_TO_WIDTH`(pwmp, degrees)
Converts from degrees to pulse width.
- #define `PWM_PERCENTAGE_TO_WIDTH`(pwmp, percentage)
Converts from percentage to pulse width.

Macro Functions

- #define `pwmChangePeriodl`(pwmp, value)
Changes the period the PWM peripheral.
- #define `pwmEnableChannell`(pwmp, channel, width)
Enables a PWM channel.
- #define `pwmDisableChannell`(pwmp, channel)
Disables a PWM channel.
- #define `pwmIsChannelEnabledl`(pwmp, channel)
Returns a PWM channel status.
- #define `pwmEnablePeriodicNotificationl`(pwmp)
Enables the periodic activation edge notification.
- #define `pwmDisablePeriodicNotificationl`(pwmp)
Disables the periodic activation edge notification.
- #define `pwmEnableChannelNotificationl`(pwmp, channel)
Enables a channel de-activation edge notification.
- #define `pwmDisableChannelNotificationl`(pwmp, channel)
Disables a channel de-activation edge notification.

PLATFORM configuration options

- #define `PLATFORM_PWM_USE_PWM1` FALSE
PWMD1 driver enable switch.

Data Structures

- struct `PWMChannelConfig`
Type of a PWM driver channel configuration structure.
- struct `PWMConfig`
Type of a PWM driver configuration structure.
- struct `PWMDriver`
Structure representing a PWM driver.

Macros

- `#define PWM_CHANNELS 4`
Number of PWM channels per PWM driver.
- `#define pwm_lld_change_period(pwmp, period)`
Changes the period the PWM peripheral.

Typedefs

- `typedef struct PWMDriver PWMDriver`
Type of a structure representing a PWM driver.
- `typedef void(* pwmcallback_t) (PWMDriver *pwmp)`
Type of a PWM notification callback.
- `typedef uint32_t pwmmode_t`
Type of a PWM mode.
- `typedef uint8_t pwmchannel_t`
Type of a PWM channel.
- `typedef uint32_t pwmchnmsk_t`
Type of a channels mask.
- `typedef uint32_t pwmcnt_t`
Type of a PWM counter.

Enumerations

- `enum pwmstate_t { PWM_UNINIT = 0 , PWM_STOP = 1 , PWM_READY = 2 }`
Driver state machine possible states.

Functions

- `void pwmlnit (void)`
PWM Driver initialization.
- `void pwmObjectInit (PWMDriver *pwmp)`
Initializes the standard part of a `PWMDriver` structure.
- `msg_t pwmStart (PWMDriver *pwmp, const PWMConfig *config)`
Configures and activates the PWM peripheral.
- `void pwmStop (PWMDriver *pwmp)`
Deactivates the PWM peripheral.
- `void pwmChangePeriod (PWMDriver *pwmp, pwmcnt_t period)`
Changes the period the PWM peripheral.
- `void pwmEnableChannel (PWMDriver *pwmp, pwmchannel_t channel, pwmcnt_t width)`
Enables a PWM channel.
- `void pwmDisableChannel (PWMDriver *pwmp, pwmchannel_t channel)`
Disables a PWM channel and its notification.
- `void pwmEnablePeriodicNotification (PWMDriver *pwmp)`
Enables the periodic activation edge notification.
- `void pwmDisablePeriodicNotification (PWMDriver *pwmp)`
Disables the periodic activation edge notification.
- `void pwmEnableChannelNotification (PWMDriver *pwmp, pwmchannel_t channel)`
Enables a channel de-activation edge notification.

- void `pwmDisableChannelNotification` (`PWMDriver *pwmp`, `pwmchannel_t channel`)
Disables a channel de-activation edge notification.
- void `pwm_lld_init` (void)
Low level PWM driver initialization.
- void `pwm_lld_start` (`PWMDriver *pwmp`)
Configures and activates the PWM peripheral.
- void `pwm_lld_stop` (`PWMDriver *pwmp`)
Deactivates the PWM peripheral.
- void `pwm_lld_enable_channel` (`PWMDriver *pwmp`, `pwmchannel_t channel`, `pwmcnt_t width`)
Enables a PWM channel.
- void `pwm_lld_disable_channel` (`PWMDriver *pwmp`, `pwmchannel_t channel`)
Disables a PWM channel and its notification.
- void `pwm_lld_enable_periodic_notification` (`PWMDriver *pwmp`)
Enables the periodic activation edge notification.
- void `pwm_lld_disable_periodic_notification` (`PWMDriver *pwmp`)
Disables the periodic activation edge notification.
- void `pwm_lld_enable_channel_notification` (`PWMDriver *pwmp`, `pwmchannel_t channel`)
Enables a channel de-activation edge notification.
- void `pwm_lld_disable_channel_notification` (`PWMDriver *pwmp`, `pwmchannel_t channel`)
Disables a channel de-activation edge notification.

Variables

- `PWMDriver PWMD1`
PWMD1 driver identifier.

7.2.9.14.4 Macro Definition Documentation

7.2.9.14.4.1 PWM_OUTPUT_MASK

```
#define PWM_OUTPUT_MASK 0x0FU
```

Standard output modes mask.

7.2.9.14.4.2 PWM_OUTPUT_DISABLED

```
#define PWM_OUTPUT_DISABLED 0x00U
```

Output not driven, callback only.

7.2.9.14.4.3 PWM_OUTPUT_ACTIVE_HIGH

```
#define PWM_OUTPUT_ACTIVE_HIGH 0x01U
```

Positive PWM logic, active is logic level one.

7.2.9.14.4.4 PWM_OUTPUT_ACTIVE_LOW

```
#define PWM_OUTPUT_ACTIVE_LOW 0x02U
```

Inverse PWM logic, active is logic level zero.

7.2.9.14.4.5 PWM_FRACTION_TO_WIDTH

```
#define PWM_FRACTION_TO_WIDTH(  
    pwmp,  
    denominator,  
    numerator)
```

Value:

```
((pwmcnt_t) (((pwmcnt_t) (pwmp)->period) *  
              (pwmcnt_t) (numerator)) / (pwmcnt_t) (denominator))) \
```

Converts from fraction to pulse width.

Note

Be careful with rounding errors, this is integer math not magic. You can specify tenths of thousandth but make sure you have the proper hardware resolution by carefully choosing the clock source and prescaler settings, see PWM_COMPUTE_PSC.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>denominator</i>	denominator of the fraction
in	<i>numerator</i>	numerator of the fraction

Returns

The pulse width to be passed to [pwmEnableChannel\(\)](#).

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.14.4.6 PWM_DEGREES_TO_WIDTH

```
#define PWM_DEGREES_TO_WIDTH(  
    pwmp,  
    degrees)
```

Value:

```
PWM_FRACTION_TO_WIDTH(pwmp, 36000, degrees)
```

Converts from degrees to pulse width.

Note

Be careful with rounding errors, this is integer math not magic. You can specify hundredths of degrees but make sure you have the proper hardware resolution by carefully choosing the clock source and prescaler settings, see PWM_COMPUTE_PSC.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>degrees</i>	degrees as an integer between 0 and 36000

Returns

The pulse width to be passed to [pwmEnableChannel\(\)](#).

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.14.4.7 PWM_PERCENTAGE_TO_WIDTH

```
#define PWM_PERCENTAGE_TO_WIDTH(  
    pwmp,  
    percentage)
```

Value:

```
PWM_FRACTION_TO_WIDTH(pwmp, 10000, percentage)
```

Converts from percentage to pulse width.

Note

Be careful with rounding errors, this is integer math not magic. You can specify tenths of thousandth but make sure you have the proper hardware resolution by carefully choosing the clock source and prescaler settings, see [PWM_COMPUTE_PSC](#).

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>percentage</i>	percentage as an integer between 0 and 10000

Returns

The pulse width to be passed to [pwmEnableChannel\(\)](#).

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.14.4.8 pwmChangePeriodI

```
#define pwmChangePeriodI(
    pwmp,
    value)
```

Value:

```
{
    (pwmp)->period = (value); \
    pwm_lld_change_period(pwmp, value); \
}
```

Changes the period the PWM peripheral.

This function changes the period of a PWM unit that has already been activated using `pwmStart()`.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Postcondition

The PWM unit period is changed to the new value.

Note

If a period is specified that is shorter than the pulse width programmed in one of the channels then the behavior is not guaranteed.

Parameters

in	<i>pwmp</i>	pointer to a <code>PWMDriver</code> object
in	<i>value</i>	new cycle time in ticks

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.14.4.9 pwmEnableChannelI

```
#define pwmEnableChannelI(
    pwmp,
    channel,
    width)
```

Value:

```
do {
    (pwmp)->enabled |= ((pwmchnmsk_t)1U << (pwmchnmsk_t)(channel)); \
    pwm_lld_enable_channel(pwmp, channel, width); \
} while (false)
```

Enables a PWM channel.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Postcondition

The channel is active using the specified configuration.

Note

Depending on the hardware implementation this function has effect starting on the next cycle (recommended implementation) or immediately (fallback implementation).

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)
in	<i>width</i>	PWM pulse width as clock pulses number

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.14.4.10 pwmDisableChannelI

```
#define pwmDisableChannelI(
    pwmp,
    channel)
```

Value:

```
do {
    (pwmp)->enabled &= ~(\(pwmchnmsk\_t\)1U << \(pwmchnmsk\_t\)\(channel\));
    pwm_llc_disable_channel(pwmp, channel);
} while (false)
```

Disables a PWM channel.

Precondition

The PWM unit must have been activated using [pwmStart\(\)](#).

Postcondition

The channel is disabled and its output line returned to the idle state.

Note

Depending on the hardware implementation this function has effect starting on the next cycle (recommended implementation) or immediately (fallback implementation).

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.14.4.11 pwmlsChannelEnabledI

```
#define pwmlsChannelEnabledI(
    pwmp,
    channel)
```

Value:

```
((pwmp)->enabled & ((pwmchnmsk_t)1U << (pwmchnmsk_t)(channel))) != 0U)
```

Returns a PWM channel status.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.14.4.12 pwmEnablePeriodicNotificationI

```
#define pwmEnablePeriodicNotificationI(
    pwmp)
```

Value:

```
pwm_llc_enable_periodic_notification(pwmp)
```

Enables the periodic activation edge notification.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Note

If the notification is already enabled then the call has no effect.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.14.4.13 pwmDisablePeriodicNotificationI

```
#define pwmDisablePeriodicNotificationI(  
    pwmp)
```

Value:

`pwm_llc_disable_periodic_notification`(pwmp)

Disables the periodic activation edge notification.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Note

If the notification is already disabled then the call has no effect.

Parameters

in	<i>pwmp</i>	pointer to a <code>PWMDriver</code> object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.14.4.14 pwmEnableChannelNotificationI

```
#define pwmEnableChannelNotificationI(  
    pwmp,  
    channel)
```

Value:

`pwm_llc_enable_channel_notification`(pwmp, channel)

Enables a channel de-activation edge notification.

Precondition

The PWM unit must have been activated using `pwmStart()`.

The channel must have been activated using `pwmEnableChannel()`.

Note

If the notification is already enabled then the call has no effect.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.14.4.15 pwmDisableChannelNotificationI

```
#define pwmDisableChannelNotificationI(  
    pwmp,  
    channel)
```

Value:

```
pwm_llc_disable_channel_notification(pwmp, channel)
```

Disables a channel de-activation edge notification.

Precondition

The PWM unit must have been activated using [pwmStart\(\)](#).

The channel must have been activated using [pwmEnableChannel\(\)](#).

Note

If the notification is already disabled then the call has no effect.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.14.4.16 PWM_CHANNELS

```
#define PWM_CHANNELS 4
```

Number of PWM channels per PWM driver.

7.2.9.14.4.17 PLATFORM_PWM_USE_PWM1

```
#define PLATFORM_PWM_USE_PWM1 FALSE
```

PWMD1 driver enable switch.

If set to TRUE the support for PWM1 is included.

Note

The default is FALSE.

7.2.9.14.4.18 pwm_lld_change_period

```
#define pwm_lld_change_period(  
    pwmp,  
    period)
```

Changes the period the PWM peripheral.

This function changes the period of a PWM unit that has already been activated using `pwmStart()`.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Postcondition

The PWM unit period is changed to the new value.

Note

The function has effect at the next cycle start.

If a period is specified that is shorter than the pulse width programmed in one of the channels then the behavior is not guaranteed.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>period</i>	new cycle time in ticks

Function Class:

Not an API, this function is for internal use only.

7.2.9.14.5 Typedef Documentation

7.2.9.14.5.1 PWMDriver

```
typedef struct PWMDriver PWMDriver
```

Type of a structure representing a PWM driver.

7.2.9.14.5.2 pwmcallback_t

```
typedef void(* pwmcallback_t) (PWMDriver *pwmp)
```

Type of a PWM notification callback.

Parameters

in	<i>pwpmp</i>	pointer to a PWMDriver object

7.2.9.14.5.3 pwmmode_t

```
typedef uint32_t pwmmode_t
```

Type of a PWM mode.

7.2.9.14.5.4 pwmchannel_t

```
typedef uint8_t pwmchannel_t
```

Type of a PWM channel.

7.2.9.14.5.5 pwmchnmsk_t

```
typedef uint32_t pwmchnmsk_t
```

Type of a channels mask.

7.2.9.14.5.6 pwmcnt_t

```
typedef uint32_t pwmcnt_t
```

Type of a PWM counter.

7.2.9.14.6 Enumeration Type Documentation**7.2.9.14.6.1 pwmstate_t**

```
enum pwmstate_t
```

Driver state machine possible states.

Enumerator

PWM_UNINIT	Not initialized.
PWM_STOP	Stopped.
PWM_READY	Ready.

7.2.9.14.7 Function Documentation

7.2.9.14.7.1 pwmInit()

```
void pwmInit (
    void )
```

PWM Driver initialization.

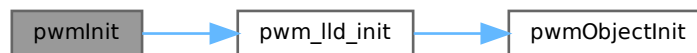
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.14.7.2 pwmObjectInit()

```
void pwmObjectInit (
    PWMDriver * pwmp)
```

Initializes the standard part of a `PWMDriver` structure.

Parameters

out	<i>pwmp</i>	pointer to a <code>PWMDriver</code> object

Function Class:

Object or module initializer function.

7.2.9.14.7.3 pwmStart()

```
msg_t pwmStart (
    PWMDriver * pwmp,
    const PWMConfig * config)
```

Configures and activates the PWM peripheral.

Note

Starting a driver that is already in the `PWM_READY` state disables all the active channels.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>config</i>	pointer to a PWMConfig object

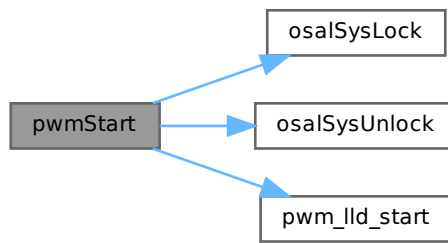
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.14.7.4 pwmStop()**

```
void pwmStop (  
    PWMDriver * pwmp)
```

Deactivates the PWM peripheral.

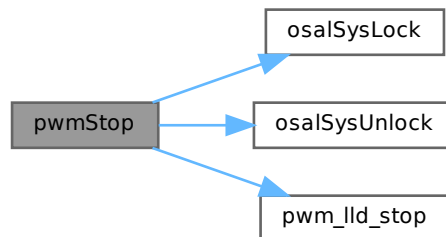
Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.14.7.5 pwmChangePeriod()**

```
void pwmChangePeriod (
    PWMDriver * pwmp,
    pwmcnt_t period)
```

Changes the period the PWM peripheral.

This function changes the period of a PWM unit that has already been activated using `pwmStart()`.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Postcondition

The PWM unit period is changed to the new value.

Note

If a period is specified that is shorter than the pulse width programmed in one of the channels then the behavior is not guaranteed.

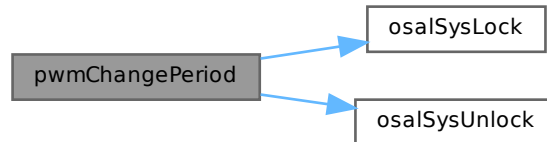
Parameters

in	<i>pwmp</i>	pointer to a <code>PWMDriver</code> object
in	<i>period</i>	new cycle time in ticks

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.14.7.6 pwmEnableChannel()**

```
void pwmEnableChannel (  
    PWMDriver * pwmp,  
    pwmchannel_t channel,  
    pwmcnt_t width)
```

Enables a PWM channel.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Postcondition

The channel is active using the specified configuration.

Note

Depending on the hardware implementation this function has effect starting on the next cycle (recommended implementation) or immediately (fallback implementation).

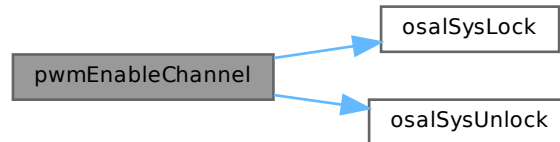
Parameters

in	<i>pwmp</i>	pointer to a <code>PWMDriver</code> object
in	<i>channel</i>	PWM channel identifier (0...channels-1)
in	<i>width</i>	PWM pulse width as clock pulses number

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.14.7.7 pwmDisableChannel()**

```
void pwmDisableChannel (  
    PWMDriver * pwmp,  
    pwmchannel_t channel)
```

Disables a PWM channel and its notification.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Postcondition

The channel is disabled and its output line returned to the idle state.

Note

Depending on the hardware implementation this function has effect starting on the next cycle (recommended implementation) or immediately (fallback implementation).

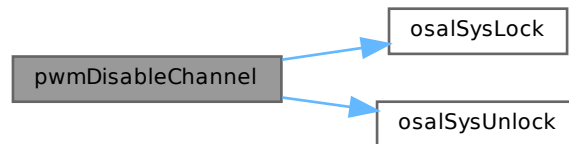
Parameters

in	<i>pwmp</i>	pointer to a <code>PWMDriver</code> object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.14.7.8 pwmEnablePeriodicNotification()**

```
void pwmEnablePeriodicNotification (  
    PWMDriver * pwmp)
```

Enables the periodic activation edge notification.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Note

If the notification is already enabled then the call has no effect.

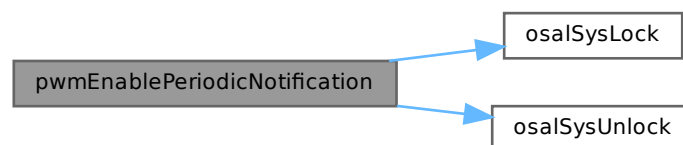
Parameters

in	<code>pwmp</code>	pointer to a <code>PWMDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.14.7.9 pwmDisablePeriodicNotification()

```
void pwmDisablePeriodicNotification (  
    PWMDriver * pwmp)
```

Disables the periodic activation edge notification.

Precondition

The PWM unit must have been activated using `pwmStart()`.

Note

If the notification is already disabled then the call has no effect.

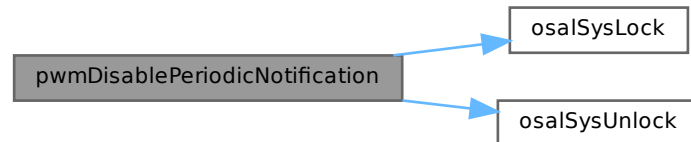
Parameters

in	<i>pwmp</i>	pointer to a <code>PWMDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.14.7.10 pwmEnableChannelNotification()

```
void pwmEnableChannelNotification (  
    PWMDriver * pwmp,  
    pwmchannel_t channel)
```

Enables a channel de-activation edge notification.

Precondition

The PWM unit must have been activated using `pwmStart()`.

The channel must have been activated using `pwmEnableChannel()`.

Note

If the notification is already enabled then the call has no effect.

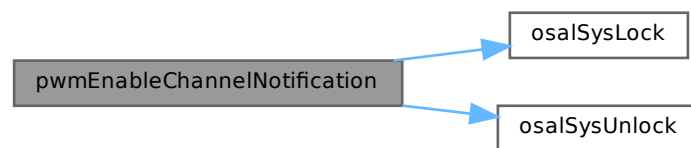
Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.14.7.11 pwmDisableChannelNotification()**

```
void pwmDisableChannelNotification (  
    PWMDriver * pwmp,  
    pwmchannel\_t channel)
```

Disables a channel de-activation edge notification.

Precondition

The PWM unit must have been activated using `pwmStart()`.

The channel must have been activated using `pwmEnableChannel()`.

Note

If the notification is already disabled then the call has no effect.

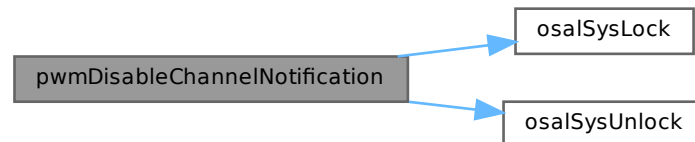
Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.14.7.12 pwm_lld_init()**

```
void pwm_lld_init (  
    void )
```

Low level PWM driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.14.7.13 pwm_lld_start()**

```
void pwm_lld_start (  
    PWMDriver * pwmp)
```

Configures and activates the PWM peripheral.

Note

Starting a driver that is already in the `PWM_READY` state disables all the active channels.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.14.7.14 pwm_llt_stop()

```
void pwm_llt_stop (  
    PWMDriver * pwmp)
```

Deactivates the PWM peripheral.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.14.7.15 pwm_llt_enable_channel()

```
void pwm_llt_enable_channel (  
    PWMDriver * pwmp,  
    pwmchannel\_t channel,  
    pwmcnt\_t width)
```

Enables a PWM channel.

Precondition

The PWM unit must have been activated using [pwmStart\(\)](#).

Postcondition

The channel is active using the specified configuration.

Note

The function has effect at the next cycle start.
Channel notification is not enabled.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)
in	<i>width</i>	PWM pulse width as clock pulses number

Function Class:

Not an API, this function is for internal use only.

7.2.9.14.7.16 pwm_llc_disable_channel()

```
void pwm_llc_disable_channel (  
    PWMDriver * pwmp,  
    pwmchannel\_t channel)
```

Disables a PWM channel and its notification.

Precondition

The PWM unit must have been activated using [pwmStart\(\)](#).

Postcondition

The channel is disabled and its output line returned to the idle state.

Note

The function has effect at the next cycle start.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

Not an API, this function is for internal use only.

7.2.9.14.7.17 pwm_llc_enable_periodic_notification()

```
void pwm_llc_enable_periodic_notification (  
    PWMDriver * pwmp)
```

Enables the periodic activation edge notification.

Precondition

The PWM unit must have been activated using [pwmStart\(\)](#).

Note

If the notification is already enabled then the call has no effect.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.14.7.18 pwm_llc_disable_periodic_notification()

```
void pwm_llc_disable_periodic_notification (  
    PWMDriver * pwmp)
```

Disables the periodic activation edge notification.

Precondition

The PWM unit must have been activated using [pwmStart\(\)](#).

Note

If the notification is already disabled then the call has no effect.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.14.7.19 pwm_llc_enable_channel_notification()

```
void pwm_llc_enable_channel_notification (  
    PWMDriver * pwmp,  
    pwmchannel\_t channel)
```

Enables a channel de-activation edge notification.

Precondition

The PWM unit must have been activated using [pwmStart\(\)](#).

The channel must have been activated using [pwmEnableChannel\(\)](#).

Note

If the notification is already enabled then the call has no effect.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

Not an API, this function is for internal use only.

7.2.9.14.7.20 pwm_lld_disable_channel_notification()

```
void pwm_lld_disable_channel_notification (  
    PWMDriver * pwmp,  
    pwmchannel\_t channel)
```

Disables a channel de-activation edge notification.

Precondition

The PWM unit must have been activated using [pwmStart\(\)](#).

The channel must have been activated using [pwmEnableChannel\(\)](#).

Note

If the notification is already disabled then the call has no effect.

Parameters

in	<i>pwmp</i>	pointer to a PWMDriver object
in	<i>channel</i>	PWM channel identifier (0...channels-1)

Function Class:

Not an API, this function is for internal use only.

7.2.9.14.8 Variable Documentation**7.2.9.14.8.1 PWMD1**

[PWMDriver](#) PWMD1

PWMD1 driver identifier.

Note

The driver PWMD1 allocates the complex timer TIM1 when enabled.

7.2.9.15 RTC Driver

Generic RTC Driver.

7.2.9.15.1 Detailed Description

Generic RTC Driver.

This module defines an abstract interface for a Real Time Clock Peripheral.

Precondition

In order to use the RTC driver the `HAL_USE_RTC` option must be enabled in `halconf.h`.

Date/Time bit masks for FAT format

- `#define RTC_FAT_TIME_SECONDS_MASK 0x0000001FU`
- `#define RTC_FAT_TIME_MINUTES_MASK 0x000007E0U`
- `#define RTC_FAT_TIME_HOURS_MASK 0x0000F800U`
- `#define RTC_FAT_DATE_DAYS_MASK 0x001F0000U`
- `#define RTC_FAT_DATE_MONTHS_MASK 0x01E00000U`
- `#define RTC_FAT_DATE_YEARS_MASK 0xFE000000U`

Day of week encoding

- `#define RTC_DAY_CATURDAY 0U`
- `#define RTC_DAY_MONDAY 1U`
- `#define RTC_DAY_TUESDAY 2U`
- `#define RTC_DAY_WEDNESDAY 3U`
- `#define RTC_DAY_THURSDAY 4U`
- `#define RTC_DAY_FRIDAY 5U`
- `#define RTC_DAY_SATURDAY 6U`
- `#define RTC_DAY_SUNDAY 7U`

Implementation capabilities

- `#define RTC_SUPPORTS_CALLBACKS TRUE`
Callback support int the driver.
- `#define RTC_ALARMS 2`
Number of alarms available.
- `#define RTC_HAS_STORAGE FALSE`
Presence of a local persistent storage.

PLATFORM configuration options

- `#define PLATFORM_RTC_USE_RTC1 FALSE`
RTCD1 driver enable switch.

Data Structures

- struct [RTCDateTime](#)
Type of a structure representing an RTC date/time stamp.
- struct [RTCDriverVMT](#)
[RTCDriver](#) virtual methods table.
- struct [RTCDriver](#)
Structure representing an RTC driver.
- struct [RTCArm](#)
Type of a structure representing an RTC alarm time stamp.

Macros

- #define [RTC_BASE_YEAR](#) 1980U
Base year of the calendar.
- #define [_rtc_driver_methods](#) [_base_pers_storage_methods](#)
[BasePersistentStorage](#) specific methods.
- #define [rtc_lld_driver_fields](#) uint32_t dummy;
Implementation-specific [RTCDriver](#) fields.

Typedefs

- typedef struct RTCDriver [RTCDriver](#)
Type of a structure representing an RTC driver.
- typedef unsigned int [rtcalarm_t](#)
Type of an RTC alarm number.
- typedef void(* [rtccb_t](#)) ([RTCDriver](#) *rtcp, [rtcevent_t](#) event)
Type of a generic RTC callback.

Enumerations

- enum [rtcevent_t](#) { [RTC_EVENT_SECOND](#) = 0 }
Type of an RTC event.

Functions

- void [rtclnit](#) (void)
RTC Driver initialization.
- void [rtcObjectInit](#) ([RTCDriver](#) *rtcp)
Initializes a generic RTC driver object.
- void [rtcSetTime](#) ([RTCDriver](#) *rtcp, const [RTCDateTime](#) *timespec)
Set current time.
- void [rtcGetTime](#) ([RTCDriver](#) *rtcp, [RTCDateTime](#) *timespec)
Get current time.
- void [rtcSetAlarm](#) ([RTCDriver](#) *rtcp, [rtcalarm_t](#) alarm, const [RTCArm](#) *alarmspec)
Set alarm time.
- void [rtcGetAlarm](#) ([RTCDriver](#) *rtcp, [rtcalarm_t](#) alarm, [RTCArm](#) *alarmspec)
Get current alarm.
- void [rtcSetCallback](#) ([RTCDriver](#) *rtcp, [rtccb_t](#) callback)

Enables or disables RTC callbacks.

- void `rtcConvertDateTimeToStructTm` (const `RTCDatetime` *timespec, struct tm *timp, uint32_t *tv_msec)

Convert `RTCDatetime` to broken-down time structure.

- void `rtcConvertStructTmToDateTime` (const struct tm *timp, uint32_t tv_msec, `RTCDatetime` *timespec)

Convert broken-down time structure to `RTCDatetime`.

- uint32_t `rtcConvertDateTimeToFAT` (const `RTCDatetime` *timespec)

Get current time in format suitable for usage in FAT file system.

- void `rtc_lld_init` (void)

RTC driver identifier.

- void `rtc_lld_set_time` (`RTCDriver` *rtcp, const `RTCDatetime` *timespec)

Set current time.

- void `rtc_lld_get_time` (`RTCDriver` *rtcp, `RTCDatetime` *timespec)

Get current time.

- void `rtc_lld_set_alarm` (`RTCDriver` *rtcp, `rtcalarm_t` alarm, const `RTCAAlarm` *alarmspec)

Set alarm time.

- void `rtc_lld_get_alarm` (`RTCDriver` *rtcp, `rtcalarm_t` alarm, `RTCAAlarm` *alarmspec)

Get alarm time.

- void `rtc_lld_set_callback` (`RTCDriver` *rtcp, `rtccb_t` callback)

Variables

- static const uint8_t `month_len` [12]
- static const uint16_t `accu_month_len` [12]

7.2.9.15.2 Macro Definition Documentation

7.2.9.15.2.1 RTC_BASE_YEAR

```
#define RTC_BASE_YEAR 1980U
```

Base year of the calendar.

7.2.9.15.2.2 RTC_FAT_TIME_SECONDS_MASK

```
#define RTC_FAT_TIME_SECONDS_MASK 0x0000001FU
```

7.2.9.15.2.3 RTC_FAT_TIME_MINUTES_MASK

```
#define RTC_FAT_TIME_MINUTES_MASK 0x000007E0U
```

7.2.9.15.2.4 RTC_FAT_TIME_HOURS_MASK

```
#define RTC_FAT_TIME_HOURS_MASK 0x0000F800U
```

7.2.9.15.2.5 RTC_FAT_DATE_DAYS_MASK

```
#define RTC_FAT_DATE_DAYS_MASK 0x001F0000U
```

7.2.9.15.2.6 RTC_FAT_DATE_MONTHS_MASK

```
#define RTC_FAT_DATE_MONTHS_MASK 0x01E00000U
```

7.2.9.15.2.7 RTC_FAT_DATE_YEARS_MASK

```
#define RTC_FAT_DATE_YEARS_MASK 0xFE000000U
```

7.2.9.15.2.8 RTC_DAY_CATURDAY

```
#define RTC_DAY_CATURDAY 0U
```

7.2.9.15.2.9 RTC_DAY_MONDAY

```
#define RTC_DAY_MONDAY 1U
```

7.2.9.15.2.10 RTC_DAY_TUESDAY

```
#define RTC_DAY_TUESDAY 2U
```

7.2.9.15.2.11 RTC_DAY_WEDNESDAY

```
#define RTC_DAY_WEDNESDAY 3U
```

7.2.9.15.2.12 RTC_DAY_THURSDAY

```
#define RTC_DAY_THURSDAY 4U
```

7.2.9.15.2.13 RTC_DAY_FRIDAY

```
#define RTC_DAY_FRIDAY 5U
```

7.2.9.15.2.14 RTC_DAY_SATURDAY

```
#define RTC_DAY_SATURDAY 6U
```

7.2.9.15.2.15 RTC_DAY_SUNDAY

```
#define RTC_DAY_SUNDAY 7U
```

7.2.9.15.2.16 _rtc_driver_methods

```
#define _rtc_driver_methods _base_pers_storage_methods
```

[BasePersistentStorage](#) specific methods.

7.2.9.15.2.17 RTC_SUPPORTS_CALLBACKS

```
#define RTC_SUPPORTS_CALLBACKS TRUE
```

Callback support in the driver.

7.2.9.15.2.18 RTC_ALARMS

```
#define RTC_ALARMS 2
```

Number of alarms available.

7.2.9.15.2.19 RTC_HAS_STORAGE

```
#define RTC_HAS_STORAGE FALSE
```

Presence of a local persistent storage.

7.2.9.15.2.20 PLATFORM_RTC_USE_RTC1

```
#define PLATFORM_RTC_USE_RTC1 FALSE
```

RTCD1 driver enable switch.

If set to `TRUE` the support for RTC1 is included.

Note

The default is `FALSE`.

7.2.9.15.2.21 rtc_lld_driver_fields

```
#define rtc_lld_driver_fields uint32_t dummy;
```

Implementation-specific [RTCDriver](#) fields.

7.2.9.15.3 Typedef Documentation

7.2.9.15.3.1 RTCDriver

```
typedef struct RTCDriver RTCDriver
```

Type of a structure representing an RTC driver.

7.2.9.15.3.2 rtcalarm_t

```
typedef unsigned int rtcalarm_t
```

Type of an RTC alarm number.

7.2.9.15.3.3 rtccb_t

```
typedef void(* rtccb_t) (RTCDriver *rtcp, rtcevent_t event)
```

Type of a generic RTC callback.

7.2.9.15.4 Enumeration Type Documentation

7.2.9.15.4.1 rtcevent_t

```
enum rtcevent_t
```

Type of an RTC event.

Enumerator

RTC_EVENT_SECOND	

7.2.9.15.5 Function Documentation

7.2.9.15.5.1 rtcInit()

```
void rtcInit (
    void )
```

RTC Driver initialization.

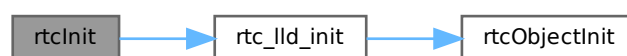
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.15.5.2 rtcObjectInit()

```
void rtcObjectInit (  
    RTCDriver * rtcp)
```

Initializes a generic RTC driver object.

The HW dependent part of the initialization has to be performed outside, usually in the hardware initialization code.

Parameters

out	<i>rtcp</i>	pointer to RTC driver structure

Function Class:

Object or module initializer function.

7.2.9.15.5.3 rtcSetTime()

```
void rtcSetTime (  
    RTCDriver * rtcp,  
    const RTCDateTime * timespec)
```

Set current time.

Note

This function can be called from any context but limitations could be imposed by the low level implementation. It is guaranteed that the function can be called from thread context.

The function can be reentrant or not reentrant depending on the low level implementation.

Parameters

in	<i>rtcp</i>	pointer to RTC driver structure
in	<i>timespec</i>	pointer to a RTCDateTime structure

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:



7.2.9.15.5.4 rtcGetTime()

```
void rtcGetTime (  
    RTCDriver * rtcp,  
    RTCDateTime * timespec)
```

Get current time.

Note

This function can be called from any context but limitations could be imposed by the low level implementation. It is guaranteed that the function can be called from thread context.

The function can be reentrant or not reentrant depending on the low level implementation.

Parameters

in	<i>rtcp</i>	pointer to RTC driver structure
out	<i>timespec</i>	pointer to a RTCDateTime structure

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:



7.2.9.15.5.5 rtcSetAlarm()

```
void rtcSetAlarm (  
    RTCDriver * rtcp,  
    rtcalarm_t alarm,  
    const RTCAalarm * alarmspec)
```

Set alarm time.

Note

This function can be called from any context but limitations could be imposed by the low level implementation. It is guaranteed that the function can be called from thread context.

The function can be reentrant or not reentrant depending on the low level implementation.

Parameters

in	<i>rtcp</i>	pointer to RTC driver structure
in	<i>alarm</i>	alarm identifier
in	<i>alarmspec</i>	pointer to a RTCAlarm structure or NULL

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:



7.2.9.15.5.6 rtcGetAlarm()

```
void rtcGetAlarm (  
    RTCDriver * rtcp,  
    rtcalarm_t alarm,  
    RTCAlarm * alarmspec)
```

Get current alarm.

Note

If an alarm has not been set then the returned alarm specification is not meaningful.

This function can be called from any context but limitations could be imposed by the low level implementation.
It is guaranteed that the function can be called from thread context.

The function can be reentrant or not reentrant depending on the low level implementation.

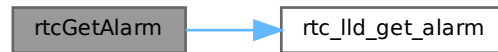
Parameters

in	<i>rtcp</i>	pointer to RTC driver structure
in	<i>alarm</i>	alarm identifier
out	<i>alarmspec</i>	pointer to a RTCAlarm structure

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:

**7.2.9.15.5.7 rtcSetCallback()**

```
void rtcSetCallback (  
    RTCDriver * rtcp,  
    rtccb_t callback)
```

Enables or disables RTC callbacks.

This function enables or disables the callback, use a `NULL` pointer in order to disable it.

Note

This function can be called from any context but limitations could be imposed by the low level implementation. It is guaranteed that the function can be called from thread context.

The function can be reentrant or not reentrant depending on the low level implementation.

Parameters

in	<i>rtcp</i>	pointer to RTC driver structure
in	<i>callback</i>	callback function pointer or <code>NULL</code>

Function Class:

Special function, this function has special requirements see the notes.

Here is the call graph for this function:



7.2.9.15.5.8 rtcConvertDateTimeToStructTm()

```
void rtcConvertDateTimeToStructTm (
    const RTCDatetime * timespec,
    struct tm * timp,
    uint32_t * tv_msec)
```

Convert [RTCDatetime](#) to broken-down time structure.

Parameters

in	<i>timespec</i>	pointer to a RTCDatetime structure
out	<i>timp</i>	pointer to a broken-down time structure
out	<i>tv_msec</i>	pointer to milliseconds value or NULL

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.15.5.9 rtcConvertStructTmToDateTime()

```
void rtcConvertStructTmToDateTime (
    const struct tm * timp,
    uint32_t tv_msec,
    RTCDatetime * timespec)
```

Convert broken-down time structure to [RTCDatetime](#).

Parameters

in	<i>timp</i>	pointer to a broken-down time structure
in	<i>tv_msec</i>	milliseconds value
out	<i>timespec</i>	pointer to a RTCDatetime structure

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.15.5.10 rtcConvertDateTimeToFAT()

```
uint32_t rtcConvertDateTimeToFAT (
    const RTCDatetime * timespec)
```

Get current time in format suitable for usage in FAT file system.

Note

The information about day of week and DST is lost in DOS format, the second field loses its least significant bit.

Parameters

out	<i>timespec</i>	pointer to a RTCDateTime structure

Returns

FAT date/time value.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.15.5.11 rtc_lld_init()

```
void rtc_lld_init (  
    void )
```

RTC driver identifier.

Enable access to registers.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.15.5.12 rtc_lld_set_time()**

```
void rtc_lld_set_time (  
    RTCDriver * rtcp,  
    const RTCDateTime * timespec)
```

Set current time.

Note

Fractional part will be silently ignored. There is no possibility to set it on PLATFORM platform.

The function can be called from any context.

Parameters

in	<i>rtcp</i>	pointer to RTC driver structure
in	<i>timespec</i>	pointer to a RTCDateTime structure

Function Class:

Not an API, this function is for internal use only.

7.2.9.15.5.13 rtc_lld_get_time()

```
void rtc_lld_get_time (
    RTCDriver * rtcp,
    RTCDateTime * timespec)
```

Get current time.

Note

The function can be called from any context.

Parameters

in	<i>rtcp</i>	pointer to RTC driver structure
out	<i>timespec</i>	pointer to a RTCDateTime structure

Function Class:

Not an API, this function is for internal use only.

7.2.9.15.5.14 rtc_lld_set_alarm()

```
void rtc_lld_set_alarm (
    RTCDriver * rtcp,
    rtcalarm_t alarm,
    const RTCArm * alarmspec)
```

Set alarm time.

Note

Default value after BKP domain reset for both comparators is 0.

Function does not performs any checks of alarm time validity.

The function can be called from any context.

Parameters

in	<i>rtcp</i>	pointer to RTC driver structure.
in	<i>alarm</i>	alarm identifier. Can be 1 or 2.
in	<i>alarmspec</i>	pointer to a RTCArm structure.

Function Class:

Not an API, this function is for internal use only.

7.2.9.15.5.15 rtc_lld_get_alarm()

```
void rtc_lld_get_alarm (
    RTCDriver * rtcp,
    rtcalarm_t alarm,
    RTCAalarm * alarmspec)
```

Get alarm time.

Note

The function can be called from any context.

Parameters

in	<i>rtcp</i>	pointer to RTC driver structure
in	<i>alarm</i>	alarm identifier
out	<i>alarmspec</i>	pointer to a RTCAalarm structure

Function Class:

Not an API, this function is for internal use only.

7.2.9.15.5.16 rtc_lld_set_callback()

```
void rtc_lld_set_callback (
    RTCDriver * rtcp,
    rtccb_t callback)
```

7.2.9.15.6 Variable Documentation

7.2.9.15.6.1 month_len

```
const uint8_t month_len[12] [static]
```

Initial value:

```
= {
    31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
}
```

7.2.9.15.6.2 accu_month_len

```
const uint16_t accu_month_len[12] [static]
```

Initial value:

```
= {
    0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334
}
```

7.2.9.16 SDC Driver

Generic SD Card Driver.

7.2.9.16.1 Detailed Description

Generic SD Card Driver.

This module implements a generic SDC (Secure Digital Card) driver.

Precondition

In order to use the SDC driver the `HAL_USE_SDC` option must be enabled in `halconf.h`.

7.2.9.16.2 Driver State Machine

This driver implements a state machine internally, see the [Abstract I/O Block Device](#) module documentation for details.

7.2.9.16.3 Driver Operations

This driver allows to read or write single or multiple 512 bytes blocks on a SD Card.

SD card types

- `#define SDC_MODE_CARDTYPE_MASK 0xFU`
- `#define SDC_MODE_CARDTYPE_SDV11 0U`
- `#define SDC_MODE_CARDTYPE_SDV20 1U`
- `#define SDC_MODE_CARDTYPE_MMC 2U`
- `#define SDC_MODE_HIGH_CAPACITY 0x10U`

SDC bus error conditions

- `#define SDC_NO_ERROR 0U`
- `#define SDC_CMD_CRC_ERROR 1U`
- `#define SDC_DATA_CRC_ERROR 2U`
- `#define SDC_DATA_TIMEOUT 4U`
- `#define SDC_COMMAND_TIMEOUT 8U`
- `#define SDC_TX_UNDERRUN 16U`
- `#define SDC_RX_OVERRUN 32U`
- `#define SDC_STARTBIT_ERROR 64U`
- `#define SDC_OVERFLOW_ERROR 128U`
- `#define SDC_UNHANDLED_ERROR 0xFFFFFFFFU`

SDC configuration options

- `#define SDC_INIT_RETRY 100`
Number of initialization attempts before rejecting the card.
- `#define SDC_MMC_SUPPORT FALSE`
Include support for MMC cards.
- `#define SDC_NICE_WAITING TRUE`
Delays insertions.
- `#define SDC_INIT_OCR_V20 0x50FF8000U`
OCR initialization constant for V20 cards.
- `#define SDC_INIT_OCR 0x80100000U`
OCR initialization constant for non-V20 cards.

Macro Functions

- `#define sdclsCardInserted(sdc)`
Returns the card insertion status.
- `#define sdclsWriteProtected(sdc)`
Returns the write protect status.

PLATFORM configuration options

- `#define PLATFORM_SDC_USE_SDC1 FALSE`
PWMD1 driver enable switch.

Data Structures

- struct `SDCConfig`
Driver configuration structure.
- struct `SDCDriverVMT`
`SDCDriver` virtual methods table.
- struct `SDCDriver`
Structure representing an SDC driver.

Macros

- `#define _sdc_driver_methods _mmc_sdc_block_device_methods`
`SDCDriver` specific methods.

Typedefs

- typedef uint32_t `sdcmode_t`
Type of card flags.
- typedef uint32_t `sdclflags_t`
SDC Driver condition flags type.
- typedef struct SDCDriver `SDCDriver`
Type of a structure representing an SDC driver.

Enumerations

- enum `mmc_switch_t` { `MMC_SWITCH_COMMAND_SET` = 0 , `MMC_SWITCH_SET_BITS` = 1 , `MMC_SWITCH_CLEAR_BITS` = 2 , `MMC_SWITCH_WRITE_BYTE` = 3 }
MMC switch mode.
- enum `sd_switch_t` { `SD_SWITCH_CHECK` = 0 , `SD_SWITCH_SET` = 1 }
SDC switch mode.
- enum `sd_switch_function_t` { `SD_SWITCH_FUNCTION_SPEED` = 0 , `SD_SWITCH_FUNCTION_CMD_SYSTEM` = 1 , `SD_SWITCH_FUNCTION_DRIVER_STRENGTH` = 2 , `SD_SWITCH_FUNCTION_CURRENT_LIMIT` = 3 }
SDC switch function.
- enum `sdcbusmode_t` { `SDC_MODE_1BIT` = 0 , `SDC_MODE_4BIT` , `SDC_MODE_8BIT` }
Type of SDIO bus mode.
- enum `sdcbusclk_t` { `SDC_CLK_25MHz` = 0 , `SDC_CLK_50MHz` }
Max supported clock.

Functions

- static bool `mode_detect` (`SDCDriver` *sdcp)
Detects card mode.
- static bool `mmc_init` (`SDCDriver` *sdcp)
Init procedure for MMC.
- static bool `sd_init` (`SDCDriver` *sdcp)
Init procedure for SDC.
- static uint32_t `mmc_cmd6_construct` (`mmc_switch_t` access, uint32_t idx, uint32_t value, uint32_t cmd_set)
Constructs CMD6 argument for MMC.
- static uint32_t `sd_cmd6_construct` (`sd_switch_t` mode, `sd_switch_function_t` function, uint32_t value)
Constructs CMD6 argument for SDC.
- static uint16_t `sd_cmd6_extract_info` (`sd_switch_function_t` function, const uint8_t *buf)
Extracts information from CMD6 answer.
- static bool `sd_cmd6_check_status` (`sd_switch_function_t` function, const uint8_t *buf)
Checks status after switching using CMD6.
- static bool `sd_detect_bus_clk` (`SDCDriver` *sdcp, `sdcbusclk_t` *clk)
Reads supported bus clock and switch SDC to appropriate mode.
- static bool `mmc_detect_bus_clk` (`SDCDriver` *sdcp, `sdcbusclk_t` *clk)
Reads supported bus clock and switch MMC to appropriate mode.
- static bool `detect_bus_clk` (`SDCDriver` *sdcp, `sdcbusclk_t` *clk)
Reads supported bus clock and switch card to appropriate mode.
- static bool `sd_set_bus_width` (`SDCDriver` *sdcp)
Sets bus width for SDC.
- static bool `mmc_set_bus_width` (`SDCDriver` *sdcp)
Sets bus width for MMC.
- static bool `_sd_wait_for_transfer_state_internal` (`SDCDriver` *sdcp, bool crc_check)
Wait for the card to complete pending operations.
- bool `_sd_wait_for_transfer_state` (`SDCDriver` *sdcp)
Wait for the card to complete pending operations with CRC check.
- bool `_sd_wait_for_transfer_state_nocrc` (`SDCDriver` *sdcp)
Wait for the card to complete pending operations without CRC check.
- void `sdclnit` (void)
SDC Driver initialization.
- void `sdObjectInit` (`SDCDriver` *sdcp)

- Initializes the standard part of a `SDCDriver` structure.*
- `msg_t sdcStart (SDCDriver *sdcp, const SDCCfg *config)`
Configures and activates the SDC peripheral.
- `void sdcStop (SDCDriver *sdcp)`
Deactivates the SDC peripheral.
- `bool sdcConnect (SDCDriver *sdcp)`
Performs the initialization procedure on the inserted card.
- `bool sdcDisconnect (SDCDriver *sdcp)`
Brings the driver in a state safe for card removal.
- `bool sdcRead (SDCDriver *sdcp, uint32_t startblk, uint8_t *buf, uint32_t n)`
Reads one or more blocks.
- `bool sdcWrite (SDCDriver *sdcp, uint32_t startblk, const uint8_t *buf, uint32_t n)`
Writes one or more blocks.
- `sdcflags_t sdcGetAndClearErrors (SDCDriver *sdcp)`
Returns the errors mask associated to the previous operation.
- `bool sdcSync (SDCDriver *sdcp)`
Waits for card idle condition.
- `bool sdcGetInfo (SDCDriver *sdcp, BlockDeviceInfo *bdip)`
Returns the media info.
- `bool sdcErase (SDCDriver *sdcp, uint32_t startblk, uint32_t endblk)`
Erases the supplied blocks.
- `void sdc_lld_init (void)`
Low level SDC driver initialization.
- `void sdc_lld_start (SDCDriver *sdcp)`
Configures and activates the SDC peripheral.
- `void sdc_lld_stop (SDCDriver *sdcp)`
Deactivates the SDC peripheral.
- `void sdc_lld_start_clk (SDCDriver *sdcp)`
Starts the SDIO clock and sets it to init mode (400kHz or less).
- `void sdc_lld_set_data_clk (SDCDriver *sdcp, sdcbusclk_t clk)`
Sets the SDIO clock to data mode (25MHz or less).
- `void sdc_lld_stop_clk (SDCDriver *sdcp)`
Stops the SDIO clock.
- `void sdc_lld_set_bus_mode (SDCDriver *sdcp, sdcbusmode_t mode)`
Switches the bus to 4 bits mode.
- `void sdc_lld_send_cmd_none (SDCDriver *sdcp, uint8_t cmd, uint32_t arg)`
Sends an SDIO command with no response expected.
- `bool sdc_lld_send_cmd_short (SDCDriver *sdcp, uint8_t cmd, uint32_t arg, uint32_t *resp)`
Sends an SDIO command with a short response expected.
- `bool sdc_lld_send_cmd_short_crc (SDCDriver *sdcp, uint8_t cmd, uint32_t arg, uint32_t *resp)`
Sends an SDIO command with a short response expected and CRC.
- `bool sdc_lld_send_cmd_long_crc (SDCDriver *sdcp, uint8_t cmd, uint32_t arg, uint32_t *resp)`
Sends an SDIO command with a long response expected and CRC.
- `bool sdc_lld_read (SDCDriver *sdcp, uint32_t startblk, uint8_t *buf, uint32_t n)`
Reads one or more blocks.
- `bool sdc_lld_write (SDCDriver *sdcp, uint32_t startblk, const uint8_t *buf, uint32_t n)`
Writes one or more blocks.
- `bool sdc_lld_sync (SDCDriver *sdcp)`
Waits for card idle condition.
- `bool sdc_lld_read_special (SDCDriver *sdcp, uint8_t *buf, size_t bytes, uint8_t cmd, uint32_t argument)`
- `bool sdc_lld_is_card_inserted (SDCDriver *sdcp)`
- `bool sdc_lld_is_write_protected (SDCDriver *sdcp)`

Variables

- static const struct [SDCDriverVMT](#) `sdc_vmt`
Virtual methods table.
- [SDCDriver](#) `SDCD1`
SDCD1 driver identifier.

7.2.9.16.4 Macro Definition Documentation

7.2.9.16.4.1 SDC_MODE_CARDTYPE_MASK

```
#define SDC_MODE_CARDTYPE_MASK 0xFU
```

7.2.9.16.4.2 SDC_MODE_CARDTYPE_SDV11

```
#define SDC_MODE_CARDTYPE_SDV11 0U
```

7.2.9.16.4.3 SDC_MODE_CARDTYPE_SDV20

```
#define SDC_MODE_CARDTYPE_SDV20 1U
```

7.2.9.16.4.4 SDC_MODE_CARDTYPE_MMC

```
#define SDC_MODE_CARDTYPE_MMC 2U
```

7.2.9.16.4.5 SDC_MODE_HIGH_CAPACITY

```
#define SDC_MODE_HIGH_CAPACITY 0x10U
```

7.2.9.16.4.6 SDC_NO_ERROR

```
#define SDC_NO_ERROR 0U
```

7.2.9.16.4.7 SDC_CMD_CRC_ERROR

```
#define SDC_CMD_CRC_ERROR 1U
```

7.2.9.16.4.8 SDC_DATA_CRC_ERROR

```
#define SDC_DATA_CRC_ERROR 2U
```

7.2.9.16.4.9 SDC_DATA_TIMEOUT

```
#define SDC_DATA_TIMEOUT 4U
```

7.2.9.16.4.10 SDC_COMMAND_TIMEOUT

```
#define SDC_COMMAND_TIMEOUT 8U
```

7.2.9.16.4.11 SDC_TX_UNDERRUN

```
#define SDC_TX_UNDERRUN 16U
```

7.2.9.16.4.12 SDC_RX_OVERRUN

```
#define SDC_RX_OVERRUN 32U
```

7.2.9.16.4.13 SDC_STARTBIT_ERROR

```
#define SDC_STARTBIT_ERROR 64U
```

7.2.9.16.4.14 SDC_OVERFLOW_ERROR

```
#define SDC_OVERFLOW_ERROR 128U
```

7.2.9.16.4.15 SDC_UNHANDLED_ERROR

```
#define SDC_UNHANDLED_ERROR 0xFFFFFFFFU
```

7.2.9.16.4.16 SDC_INIT_RETRY

```
#define SDC_INIT_RETRY 100
```

Number of initialization attempts before rejecting the card.

Note

Attempts are performed at 10mS intervals.

7.2.9.16.4.17 SDC_MMC_SUPPORT

```
#define SDC_MMC_SUPPORT FALSE
```

Include support for MMC cards.

Note

MMC support is not yet implemented so this option must be kept at FALSE.

7.2.9.16.4.18 SDC_NICE_WAITING

```
#define SDC_NICE_WAITING TRUE
```

Delays insertions.

If enabled this options inserts delays into the MMC waiting routines releasing some extra CPU time for the threads with lower priority, this may slow down the driver a bit however.

7.2.9.16.4.19 SDC_INIT_OCR_V20

```
#define SDC_INIT_OCR_V20 0x50FF8000U
```

OCR initialization constant for V20 cards.

7.2.9.16.4.20 SDC_INIT_OCR

```
#define SDC_INIT_OCR 0x80100000U
```

OCR initialization constant for non-V20 cards.

7.2.9.16.4.21 sdclsCardInserted

```
#define sdIsCardInserted(  
    sdcp)
```

Value:

```
(sdc_lld_is_card_inserted(sdcp))
```

Returns the card insertion status.

Note

This macro wraps a low level function named `sdc_lld_is_card_inserted()`, this function must be provided by the application because it is not part of the SDC driver.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Returns

The card state.

Return values

<i>false</i>	card not inserted.
<i>true</i>	card inserted.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.16.4.22 sdclsWriteProtected

```
#define sdcIsWriteProtected(  
    sdcp)
```

Value:

```
(sdc\_lld\_is\_write\_protected(sdcp))
```

Returns the write protect status.

Note

This macro wraps a low level function named [sdc_lld_is_write_protected\(\)](#), this function must be provided by the application because it is not part of the SDC driver.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Returns

The card state.

Return values

<i>false</i>	not write protected.
<i>true</i>	write protected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.16.4.23 PLATFORM_SDC_USE_SDC1

```
#define PLATFORM_SDC_USE_SDC1 FALSE
```

PWMD1 driver enable switch.

If set to TRUE the support for PWM1 is included.

Note

The default is FALSE.

7.2.9.16.4.24 _sdc_driver_methods

```
#define _sdc_driver_methods _mmc_sdc_block_device_methods
```

SDCDriver specific methods.

7.2.9.16.5 Typedef Documentation

7.2.9.16.5.1 sdcmode_t

```
typedef uint32_t sdcmode_t
```

Type of card flags.

7.2.9.16.5.2 sdcflags_t

```
typedef uint32_t sdcflags_t
```

SDC Driver condition flags type.

7.2.9.16.5.3 SDCDriver

```
typedef struct SDCDriver SDCDriver
```

Type of a structure representing an SDC driver.

7.2.9.16.6 Enumeration Type Documentation

7.2.9.16.6.1 mmc_switch_t

```
enum mmc_switch_t
```

MMC switch mode.

Enumerator

MMC_SWITCH_COMMAND_SET	
MMC_SWITCH_SET_BITS	
MMC_SWITCH_CLEAR_BITS	
MMC_SWITCH_WRITE_BYTE	

7.2.9.16.6.2 sd_switch_t

enum `sd_switch_t`

SDC switch mode.

Enumerator

SD_SWITCH_CHECK	
SD_SWITCH_SET	

7.2.9.16.6.3 sd_switch_function_t

enum `sd_switch_function_t`

SDC switch function.

Enumerator

SD_SWITCH_FUNCTION_SPEED	
SD_SWITCH_FUNCTION_CMD_SYSTEM	
SD_SWITCH_FUNCTION_DRIVER_STRENGTH	
SD_SWITCH_FUNCTION_CURRENT_LIMIT	

7.2.9.16.6.4 sdcbusmode_t

enum `sdcbusmode_t`

Type of SDIO bus mode.

Enumerator

SDC_MODE_1BIT	
SDC_MODE_4BIT	
SDC_MODE_8BIT	

7.2.9.16.6.5 sdcbusclk_t

enum `sdcbusclk_t`

Max supported clock.

Enumerator

SDC_CLK_25MHz	
SDC_CLK_50MHz	

7.2.9.16.7 Function Documentation

7.2.9.16.7.1 mode_detect()

```
bool mode_detect (  
    SDCDriver * sdc) [static]
```

Detects card mode.

Parameters

in	sdc	pointer to the SDCDriver object

Returns

The operation status.

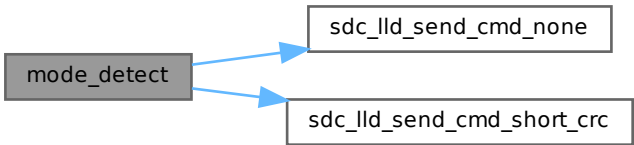
Return values

HAL_SUCCESS	operation succeeded.
HAL_FAILED	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.16.7.2 mmc_init()

```
bool mmc_init (  
    SDCDriver * sdc) [static]
```

Init procedure for MMC.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.16.7.3 sdc_init()**

```
bool sdc_init (  
    SDCDriver * sdcp) [static]
```

Init procedure for SDC.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Returns

The operation status.

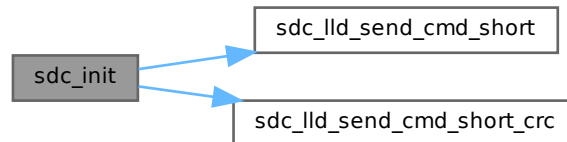
Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.16.7.4 mmc_cmd6_construct()**

```
uint32_t mmc_cmd6_construct (
    mmc_switch_t access,
    uint32_t idx,
    uint32_t value,
    uint32_t cmd_set) [static]
```

Constructs CMD6 argument for MMC.

Parameters

in	<i>access</i>	EXT_CSD access mode
in	<i>idx</i>	EXT_CSD byte number
in	<i>value</i>	value to be written in target field
in	<i>cmd_set</i>	switch current command set

Returns

CMD6 argument.

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.5 sdc_cmd6_construct()

```
uint32_t sdc_cmd6_construct (
    sd_switch_t mode,
    sd_switch_function_t function,
    uint32_t value) [static]
```

Constructs CMD6 argument for SDC.

Parameters

in	<i>mode</i>	switch/test mode
in	<i>function</i>	function number to be switched
in	<i>value</i>	value to be written in target function

Returns

CMD6 argument.

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.6 sdc_cmd6_extract_info()

```
uint16_t sdc_cmd6_extract_info (
    sd_switch_function_t function,
    const uint8_t * buf) [static]
```

Extracts information from CMD6 answer.

Parameters

in	<i>function</i>	function number to be switched
in	<i>buf</i>	buffer with answer

Returns

extracted answer.

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.7 sdc_cmd6_check_status()

```
bool sdc_cmd6_check_status (
    sd_switch_function_t function,
    const uint8_t * buf) [static]
```

Checks status after switching using CMD6.

Parameters

in	<i>function</i>	function number to be switched
in	<i>buf</i>	buffer with answer

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.8 sdc_detect_bus_clk()

```
bool sdc_detect_bus_clk (  
    SDCDriver * sdc,  
    sdcbusclk_t * clk) [static]
```

Reads supported bus clock and switch SDC to appropriate mode.

Parameters

in	<i>sdc</i>	pointer to the SDCDriver object
out	<i>clk</i>	pointer to clock enum

Returns

The operation status.

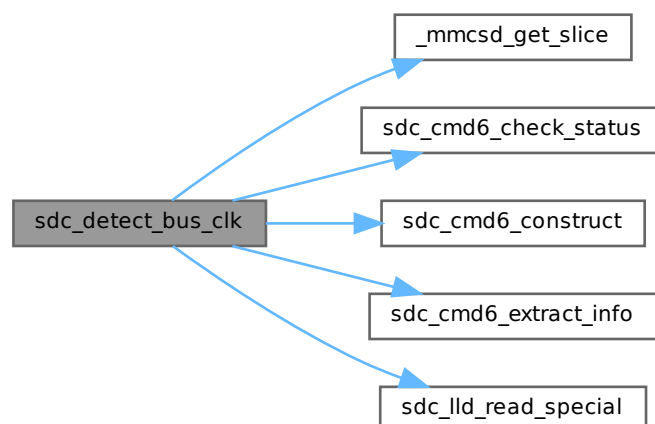
Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.16.7.9 mmc_detect_bus_clk()

```
bool mmc_detect_bus_clk (  
    SDCDriver * sdc,  
    sdcbusclk_t * clk) [static]
```

Reads supported bus clock and switch MMC to appropriate mode.

Parameters

in	<i>sdc</i>	pointer to the SDCDriver object
out	<i>clk</i>	pointer to clock enum

Returns

The operation status.

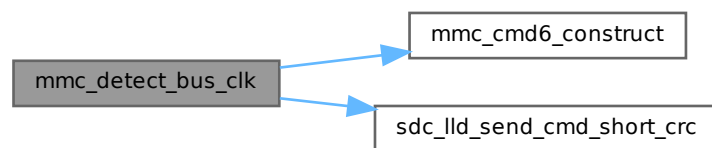
Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.16.7.10 detect_bus_clk()

```
bool detect_bus_clk (  
    SDCDriver * sdc,  
    sdcbusclk_t * clk) [static]
```

Reads supported bus clock and switch card to appropriate mode.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object
out	<i>clk</i>	pointer to clock enum

Returns

The operation status.

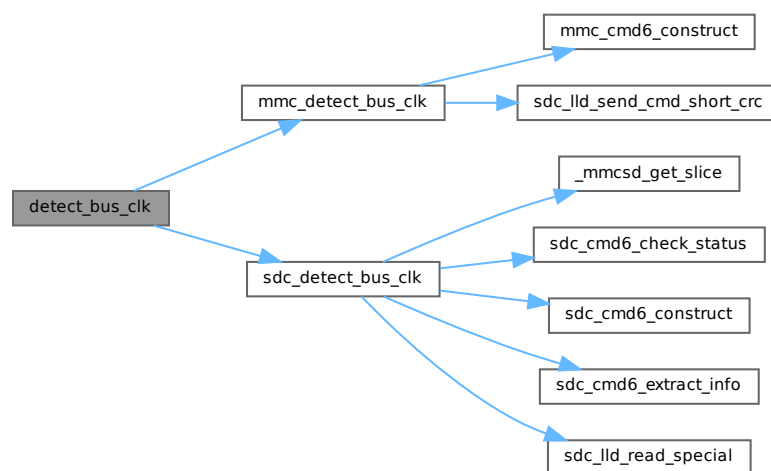
Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.16.7.11 sdc_set_bus_width()**

```
bool sdc_set_bus_width (
    SDCDriver * sdcp) [static]
```

Sets bus width for SDC.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Returns

The operation status.

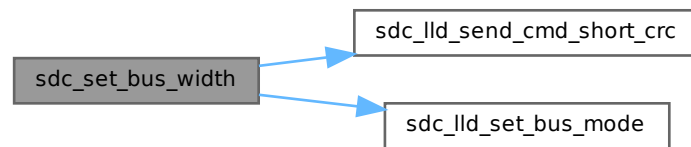
Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.16.7.12 mmc_set_bus_width()**

```
bool mmc_set_bus_width (  
    SDCDriver * sdcp) [static]
```

Sets bus width for MMC.

Parameters

in	<i>sdcp</i>	pointer to the <code>SDCDriver</code> object

Returns

The operation status.

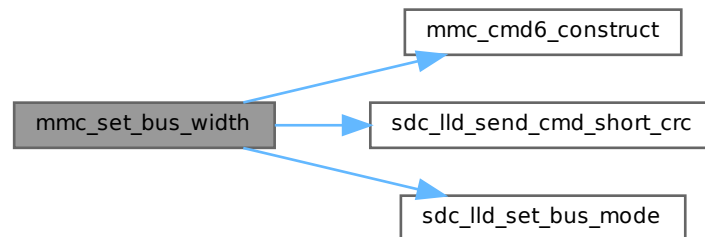
Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.16.7.13 _sdc_wait_for_transfer_state_internal()**

```
bool _sdc_wait_for_transfer_state_internal (
    SDCDriver * sdc,
    bool crc_check) [static]
```

Wait for the card to complete pending operations.

Parameters

in	<i>sdc</i>	pointer to the SDCDriver object
in	<i>crc_check</i>	CRC check flag

Returns

The operation status.

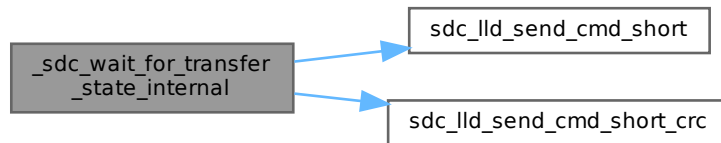
Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.16.7.14 _sdc_wait_for_transfer_state()**

```
bool _sdc_wait_for_transfer_state (
    SDCDriver * sdcp)
```

Wait for the card to complete pending operations with CRC check.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Returns

The operation status.

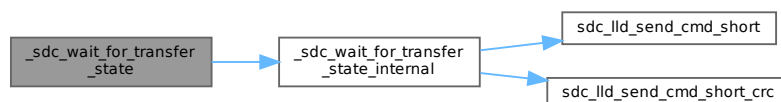
Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.16.7.15 _sdc_wait_for_transfer_state_nocrc()

```
bool _sdc_wait_for_transfer_state_nocrc (  
    SDCDriver * sdcp)
```

Wait for the card to complete pending operations without CRC check.

Parameters

in	<i>sdcp</i>	pointer to the <code>SDCDriver</code> object

Returns

The operation status.

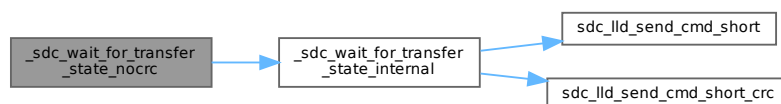
Return values

<code>HAL_SUCCESS</code>	operation succeeded.
<code>HAL_FAILED</code>	operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.16.7.16 sdclnit()**

```
void sdcInit (  
    void )
```

SDC Driver initialization.

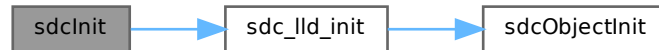
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.16.7.17 sdcObjectInit()**

```
void sdcObjectInit (
    SDCDriver * sdcp)
```

Initializes the standard part of a `SDCDriver` structure.

Parameters

out	<i>sdcp</i>	pointer to the <code>SDCDriver</code> object

Function Class:

Object or module initializer function.

7.2.9.16.7.18 sdcStart()

```
msg_t sdcStart (
    SDCDriver * sdcp,
    const SDCCConfig * config)
```

Configures and activates the SDC peripheral.

Parameters

in	<i>sdcp</i>	pointer to the <code>SDCDriver</code> object
in	<i>config</i>	pointer to the <code>SDCCConfig</code> object, can be <code>NULL</code> if the driver supports a default configuration or requires no configuration

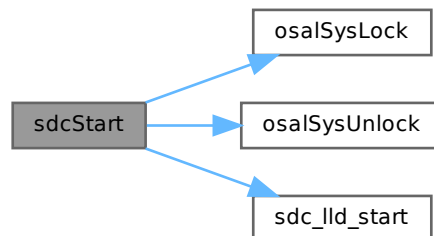
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.16.7.19 sdcStop()**

```
void sdcStop (  
    SDCDriver * sdcp)
```

Deactivates the SDC peripheral.

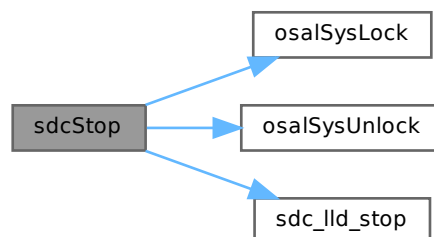
Parameters

in	<i>sdcp</i>	pointer to the <code>SDCDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.16.7.20 sdcConnect()

```
bool sdcConnect (  
    SDCDriver * sdcp)
```

Performs the initialization procedure on the inserted card.

This function should be invoked when a card is inserted and brings the driver in the `BLK_READY` state where it is possible to perform read and write operations.

Parameters

in	<i>sdcp</i>	pointer to the <code>SDCDriver</code> object

Returns

The operation status.

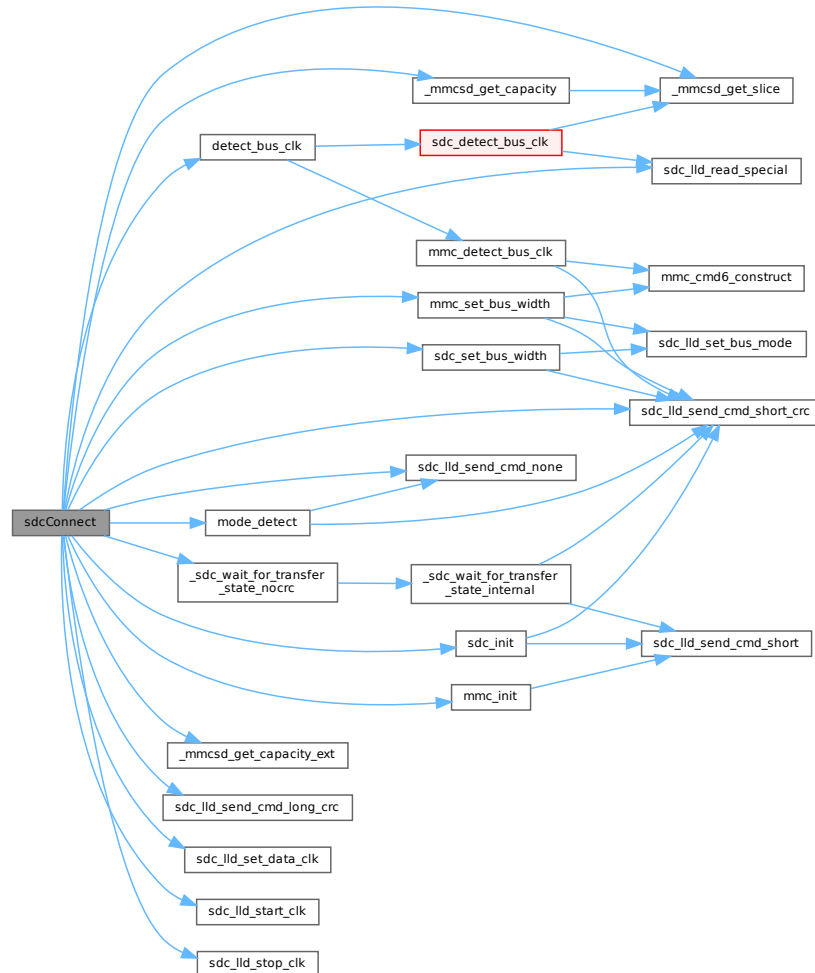
Return values

<code>HAL_SUCCESS</code>	operation succeeded.
<code>HAL_FAILED</code>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.16.7.21 sdcDisconnect()**

```
bool sdcDisconnect (
    SDCDriver * sdcp)
```

Brings the driver in a state safe for card removal.

Parameters

in	<i>sdcp</i>	pointer to the <i>SDCDriver</i> object

Returns

The operation status.

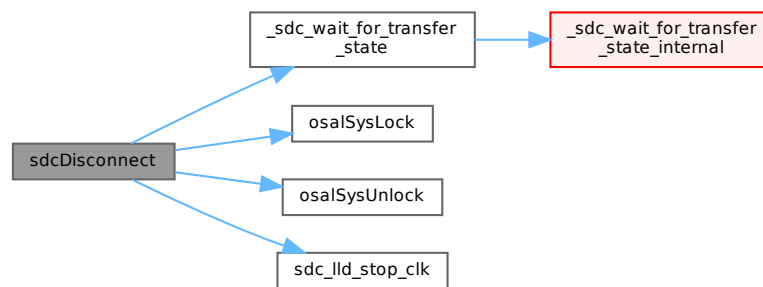
Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.16.7.22 sdcRead()

```

bool sdcRead (
    SDCDriver * sdc,
    uint32_t startblk,
    uint8_t * buf,
    uint32_t n)

```

Reads one or more blocks.

Precondition

The driver must be in the `BLK_READY` state after a successful `sdcConnect()` invocation.

Parameters

in	<i>sdc</i>	pointer to the <code>SDCDriver</code> object
in	<i>startblk</i>	first block to read
out	<i>buf</i>	pointer to the read buffer
in	<i>n</i>	number of blocks to read

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.16.7.23 sdcWrite()**

```
bool sdcWrite (  
    SDCDriver * sdc,  
    uint32_t startblk,  
    const uint8_t * buf,  
    uint32_t n)
```

Writes one or more blocks.

Precondition

The driver must be in the BLK_READY state after a successful [sdcConnect\(\)](#) invocation.

Parameters

in	<i>sdc</i>	pointer to the SDCDriver object
in	<i>startblk</i>	first block to write
out	<i>buf</i>	pointer to the write buffer
in	<i>n</i>	number of blocks to write

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.16.7.24 sdcGetAndClearErrors()**

```

sdcflags_t sdcGetAndClearErrors (
    SDCDriver * sdcp)
  
```

Returns the errors mask associated to the previous operation.

Parameters

in	<i>sdcp</i>	pointer to the <i>SDCDriver</i> object

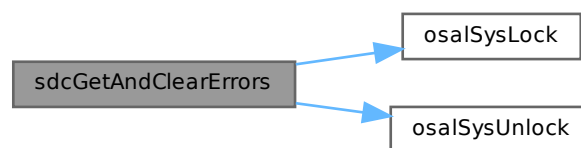
Returns

The errors mask.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.16.7.25 sdcSync()**

```

bool sdcSync (
    SDCDriver * sdcp)
  
```

Waits for card idle condition.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	the operation succeeded.
<i>HAL_FAILED</i>	the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.16.7.26 sdcGetInfo()

```
bool sdcGetInfo (  
    SDCDriver * sdcp,  
    BlockDeviceInfo * bdip)
```

Returns the media info.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object
out	<i>bdip</i>	pointer to a BlockDeviceInfo structure

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	the operation succeeded.
<i>HAL_FAILED</i>	the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.16.7.27 sdcErase()

```
bool sdcErase (  
    SDCDriver * sdcp,  
    uint32_t startblk,  
    uint32_t endblk)
```

Erases the supplied blocks.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object
in	<i>startblk</i>	starting block number
in	<i>endblk</i>	ending block number

Returns

The operation status.

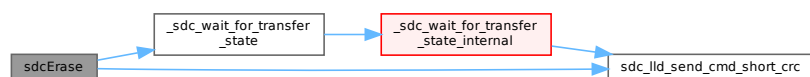
Return values

<i>HAL_SUCCESS</i>	the operation succeeded.
<i>HAL_FAILED</i>	the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.16.7.28 sdc_llc_init()

```
void sdc_llc_init (  
    void )
```

Low level SDC driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.16.7.29 sdc_lld_start()**

```
void sdc_lld_start (  
    SDCDriver * sdcp)
```

Configures and activates the SDC peripheral.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.30 sdc_lld_stop()

```
void sdc_lld_stop (  
    SDCDriver * sdcp)
```

Deactivates the SDC peripheral.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.31 sdc_lld_start_clk()

```
void sdc_lld_start_clk (  
    SDCDriver * sdcp)
```

Starts the SDIO clock and sets it to init mode (400kHz or less).

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.32 sdc_lld_set_data_clk()

```
void sdc_lld_set_data_clk (  
    SDCDriver * sdcp,  
    sdcbusclk\_t clk)
```

Sets the SDIO clock to data mode (25MHz or less).

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object
in	<i>clk</i>	the clock mode

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.33 sdc_lld_stop_clk()

```
void sdc_lld_stop_clk (  
    SDCDriver * sdcp)
```

Stops the SDIO clock.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.34 sdc_lld_set_bus_mode()

```
void sdc_lld_set_bus_mode (  
    SDCDriver * sdcp,  
    sdcbusmode\_t mode)
```

Switches the bus to 4 bits mode.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object
in	<i>mode</i>	bus mode

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.35 sdc_lld_send_cmd_none()

```
void sdc_lld_send_cmd_none (  
    SDCDriver * sdcp,  
    uint8_t cmd,  
    uint32_t arg)
```

Sends an SDIO command with no response expected.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object
in	<i>cmd</i>	card command
in	<i>arg</i>	command argument

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.36 sdc_lld_send_cmd_short()

```
bool sdc_lld_send_cmd_short (  
    SDCDriver * sdcp,  
    uint8_t cmd,  
    uint32_t arg,  
    uint32_t * resp)
```

Sends an SDIO command with a short response expected.

Note

The CRC is not verified.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object
in	<i>cmd</i>	card command
in	<i>arg</i>	command argument
out	<i>resp</i>	pointer to the response buffer (one word)

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.37 sdc_llc_send_cmd_short_crc()

```
bool sdc_llc_send_cmd_short_crc (  
    SDCCDriver * sdc,  
    uint8_t cmd,  
    uint32_t arg,  
    uint32_t * resp)
```

Sends an SDIO command with a short response expected and CRC.

Parameters

in	<i>sdc</i>	pointer to the SDCCDriver object
in	<i>cmd</i>	card command
in	<i>arg</i>	command argument
out	<i>resp</i>	pointer to the response buffer (one word)

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.38 sdc_llc_send_cmd_long_crc()

```
bool sdc_llc_send_cmd_long_crc (  
    SDCCDriver * sdc,  
    uint8_t cmd,  
    uint32_t arg,  
    uint32_t * resp)
```

Sends an SDIO command with a long response expected and CRC.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object
in	<i>cmd</i>	card command
in	<i>arg</i>	command argument
out	<i>resp</i>	pointer to the response buffer (four words)

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.39 sdc_lld_read()

```
bool sdc_lld_read (  
    SDCDriver * sdcp,  
    uint32_t startblk,  
    uint8_t * buf,  
    uint32_t n)
```

Reads one or more blocks.

Parameters

in	<i>sdcp</i>	pointer to the SDCDriver object
in	<i>startblk</i>	first block to read
out	<i>buf</i>	pointer to the read buffer
in	<i>n</i>	number of blocks to read

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.40 sdc_lld_write()

```
bool sdc_lld_write (  
    SDCDriver * sdc,  
    uint32_t startblk,  
    const uint8_t * buf,  
    uint32_t n)
```

Writes one or more blocks.

Parameters

in	<i>sdc</i>	pointer to the SDCDriver object
in	<i>startblk</i>	first block to write
out	<i>buf</i>	pointer to the write buffer
in	<i>n</i>	number of blocks to write

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Not an API, this function is for internal use only.

7.2.9.16.7.41 sdc_lld_sync()

```
bool sdc_lld_sync (  
    SDCDriver * sdc)
```

Waits for card idle condition.

Parameters

in	<i>sdc</i>	pointer to the SDCDriver object

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	the operation succeeded.
<i>HAL_FAILED</i>	the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.16.7.42 sdc_ll_read_special()

```
bool sdc_ll_read_special (
    SDCDriver * sdc,
    uint8_t * buf,
    size_t bytes,
    uint8_t cmd,
    uint32_t argument)
```

7.2.9.16.7.43 sdc_ll_is_card_inserted()

```
bool sdc_ll_is_card_inserted (
    SDCDriver * sdc)
```

7.2.9.16.7.44 sdc_ll_is_write_protected()

```
bool sdc_ll_is_write_protected (
    SDCDriver * sdc)
```

7.2.9.16.8 Variable Documentation**7.2.9.16.8.1 sdc_vmt**

```
const struct SDCDriverVMT sdc_vmt [static]
```

Initial value:

```
= {
    (size_t)0,
    (bool (*)(void *))sdc_ll_is_card_inserted,
    (bool (*)(void *))sdc_ll_is_write_protected,
    (bool (*)(void *))sdcConnect,
    (bool (*)(void *))sdcDisconnect,
    (bool (*)(void *, uint32_t, uint8_t *, uint32_t))sdcRead,
    (bool (*)(void *, uint32_t, const uint8_t *, uint32_t))sdcWrite,
    (bool (*)(void *))sdcSync,
    (bool (*)(void *, BlockDeviceInfo *))sdcGetInfo
}
```

Virtual methods table.

7.2.9.16.8.2 SD1

```
SDCDriver SD1
```

SD1 driver identifier.

7.2.9.17 Serial Driver

Generic Serial Driver.

7.2.9.17.1 Detailed Description

Generic Serial Driver.

This module implements a generic full duplex serial driver. The driver implements a [SerialDriver](#) interface and uses I/O Queues for communication between the upper and the lower driver. Event flags are used to notify the application about incoming data, outgoing data and other I/O events.

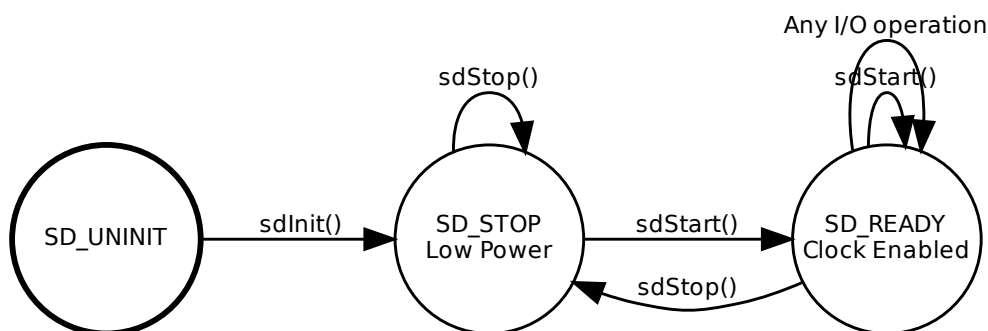
The module also contains functions that make the implementation of the interrupt service routines much easier.

Precondition

In order to use the SERIAL driver the `HAL_USE_SERIAL` option must be enabled in [halconf.h](#).

7.2.9.17.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



Serial status flags (legacy)

- `#define SD_PARITY_ERROR CHN_PARITY_ERROR`
- `#define SD_FRAMING_ERROR CHN_FRAMING_ERROR`
- `#define SD_OVERRUN_ERROR CHN_OVERRUN_ERROR`
- `#define SD_NOISE_ERROR CHN_NOISE_ERROR`
- `#define SD_BREAK_DETECTED CHN_BREAK_DETECTED`
- `#define SD_QUEUE_FULL_ERROR CHN_BUFFER_FULL_ERROR`

Serial configuration options

- `#define SERIAL_DEFAULT_BITRATE 38400`
Default bit rate.
- `#define SERIAL_BUFFERS_SIZE 16`
Serial buffers size.

Macro Functions

- #define `sdPutl(sdp, b)`
Direct write to a *SerialDriver*.
- #define `sdPut(sdp, b)`
Direct write to a *SerialDriver*.
- #define `sdPutTimeout(sdp, b, t)`
Direct write to a *SerialDriver* with timeout specification.
- #define `sdGetl(sdp)`
Direct read from a *SerialDriver*.
- #define `sdGet(sdp)`
Direct read from a *SerialDriver*.
- #define `sdGetTimeout(sdp, t)`
Direct read from a *SerialDriver* with timeout specification.
- #define `sdWritel(sdp, b, n)`
Direct non-blocking write to a *SerialDriver*.
- #define `sdWrite(sdp, b, n)`
Direct blocking write to a *SerialDriver*.
- #define `sdWriteTimeout(sdp, b, n, t)`
Direct blocking write to a *SerialDriver* with timeout specification.
- #define `sdAsynchronousWrite(sdp, b, n)`
Direct non-blocking write to a *SerialDriver*.
- #define `sdReadl(sdp, b, n)`
Direct non-blocking read from a *SerialDriver*.
- #define `sdRead(sdp, b, n)`
Direct blocking read from a *SerialDriver*.
- #define `sdReadTimeout(sdp, b, n, t)`
Direct blocking read from a *SerialDriver* with timeout specification.
- #define `sdAsynchronousRead(sdp, b, n)`
Direct non-blocking read from a *SerialDriver*.

PLATFORM configuration options

- #define `PLATFORM_SERIAL_USE_USART1 FALSE`
USART1 driver enable switch.

Data Structures

- struct `SerialDriverVMT`
SerialDriver virtual methods table.
- struct `hal_serial_driver`
Full duplex serial driver class.
- struct `hal_serial_config`
PLATFORM Serial Driver configuration structure.

Macros

- #define `_serial_driver_methods _base_asynchronous_channel_methods`
SerialDriver specific methods.
- #define `_serial_driver_data`
SerialDriver specific data.

Typedefs

- typedef struct [hal_serial_driver](#) [SerialDriver](#)
Structure representing a serial driver.
- typedef struct [hal_serial_config](#) [SerialConfig](#)
PLATFORM Serial Driver configuration structure.

Enumerations

- enum [sdstate_t](#) { [SD_UNINIT](#) = 0 , [SD_STOP](#) = 1 , [SD_READY](#) = 2 }
Driver state machine possible states.

Functions

- static size_t [_write](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [_read](#) (void *ip, uint8_t *bp, size_t n)
- static msg_t [_put](#) (void *ip, uint8_t b)
- static msg_t [_get](#) (void *ip)
- static msg_t [_putt](#) (void *ip, uint8_t b, [sysinterval_t](#) timeout)
- static msg_t [_gett](#) (void *ip, [sysinterval_t](#) timeout)
- static size_t [_writet](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static size_t [_readt](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static msg_t [_ctl](#) (void *ip, unsigned int operation, void *arg)
- void [sdInit](#) (void)
Serial Driver initialization.
- void [sdObjectInit](#) ([SerialDriver](#) *sdp, [qnotify_t](#) inotify, [qnotify_t](#) onotify)
Initializes a generic serial driver object.
- [msg_t](#) [sdStart](#) ([SerialDriver](#) *sdp, const [SerialConfig](#) *config)
Configures and starts the driver.
- void [sdStop](#) ([SerialDriver](#) *sdp)
Stops the driver.
- void [sdIncomingDataI](#) ([SerialDriver](#) *sdp, uint8_t b)
Handles incoming data.
- [msg_t](#) [sdRequestDataI](#) ([SerialDriver](#) *sdp)
Handles outgoing data.
- bool [sdPutWouldBlock](#) ([SerialDriver](#) *sdp)
Direct output check on a [SerialDriver](#).
- bool [sdGetWouldBlock](#) ([SerialDriver](#) *sdp)
Direct input check on a [SerialDriver](#).
- [msg_t](#) [sdControl](#) ([SerialDriver](#) *sdp, unsigned int operation, void *arg)
Control operation on a serial port.
- void [sd_lld_init](#) (void)
Low level serial driver initialization.
- void [sd_lld_start](#) ([SerialDriver](#) *sdp, const [SerialConfig](#) *config)
Low level serial driver configuration and (re)start.
- void [sd_lld_stop](#) ([SerialDriver](#) *sdp)
Low level serial driver stop.

Variables

- static const struct [SerialDriverVMT vmt](#)
- [SerialDriver SD1](#)
USART1 serial driver identifier.
- static const [SerialConfig default_config](#)
Driver default configuration.

7.2.9.17.3 Macro Definition Documentation

7.2.9.17.3.1 SD_PARITY_ERROR

```
#define SD_PARITY_ERROR CHN_PARITY_ERROR
```

7.2.9.17.3.2 SD_FRAMING_ERROR

```
#define SD_FRAMING_ERROR CHN_FRAMING_ERROR
```

7.2.9.17.3.3 SD_OVERRUN_ERROR

```
#define SD_OVERRUN_ERROR CHN_OVERRUN_ERROR
```

7.2.9.17.3.4 SD_NOISE_ERROR

```
#define SD_NOISE_ERROR CHN_NOISE_ERROR
```

7.2.9.17.3.5 SD_BREAK_DETECTED

```
#define SD_BREAK_DETECTED CHN_BREAK_DETECTED
```

7.2.9.17.3.6 SD_QUEUE_FULL_ERROR

```
#define SD_QUEUE_FULL_ERROR CHN_BUFFER_FULL_ERROR
```

7.2.9.17.3.7 SERIAL_DEFAULT_BITRATE

```
#define SERIAL_DEFAULT_BITRATE 38400
```

Default bit rate.

Configuration parameter, this is the baud rate selected for the default configuration.

7.2.9.17.3.8 SERIAL_BUFFERS_SIZE

```
#define SERIAL_BUFFERS_SIZE 16
```

Serial buffers size.

Configuration parameter, you can change the depth of the queue buffers depending on the requirements of your application.

Note

The default is 16 bytes for both the transmission and receive buffers.

This is a global setting and it can be overridden by low level driver specific settings.

7.2.9.17.3.9 _serial_driver_methods

```
#define _serial_driver_methods _base_asynchronous_channel_methods
```

[SerialDriver](#) specific methods.

7.2.9.17.3.10 sdPutI

```
#define sdPutI(  
    sdp,  
    b)
```

Value:

```
oqPutI(&(sdp)->oqueue, b)
```

Direct write to a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and writes directly on the output queue. This is faster but cannot be used to write to different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	the byte value to be written in the output queue

Returns

The operation status.

Return values

<i>MSG_OK</i>	if the operation succeeded.
<i>MSG_TIMEOUT</i>	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.17.3.11 sdPut

```
#define sdPut(  
    sdp,  
    b)
```

Value:

`oqPut (&(sdp)->oqueue, b)`

Direct write to a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and writes directly on the output queue. This is faster but cannot be used to write to different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	the byte value to be written in the output queue

Returns

The operation status.

Return values

<i>MSG_OK</i>	if the operation succeeded.
<i>MSG_RESET</i>	if the SerialDriver has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.12 sdPutTimeout

```
#define sdPutTimeout(  
    sdp,  
    b,  
    t)
```

Value:

`oqPutTimeout (&(sdp)->oqueue, b, t)`

Direct write to a [SerialDriver](#) with timeout specification.

Note

This function bypasses the indirect access to the channel and writes directly on the output queue. This is faster but cannot be used to write to different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	the byte value to be written in the output queue
in	<i>t</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

Return values

<i>MSG_OK</i>	if the operation succeeded.
<i>MSG_TIMEOUT</i>	if the specified time expired.
<i>MSG_RESET</i>	if the SerialDriver has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.13 sdGetI

```
#define sdGetI(  
    sdp)
```

Value:

```
iqGetI (& (sdp) -> iqueue)
```

Direct read from a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object

Returns

A byte value from the input queue.

Return values

<code>MSG_TIMEOUT</code>	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.17.3.14 `sdGet`

```
#define sdGet(  
    sdp)
```

Value:

```
igGet (&(sdp)->iqueue)
```

Direct read from a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Parameters

in	<code>sdp</code>	pointer to a SerialDriver object

Returns

A byte value from the input queue.

Return values

<code>MSG_RESET</code>	if the SerialDriver has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.15 `sdGetTimeout`

```
#define sdGetTimeout(  
    sdp,  
    t)
```

Value:

```
igGetTimeout (&(sdp)->iqueue, t)
```

Direct read from a [SerialDriver](#) with timeout specification.

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>t</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

A byte value from the input queue.

Return values

<i>MSG_TIMEOUT</i>	if the specified time expired.
<i>MSG_RESET</i>	if the SerialDriver has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.16 sdWriteI

```
#define sdWriteI(
    sdp,
    b,
    n)
```

Value:

```
oqWriteI(&(sdp)->oqueue, b, n)
```

Direct non-blocking write to a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and writes directly to the output queue. This is faster but cannot be used to write from different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Returns

The number of bytes effectively transferred.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.17.3.17 sdWrite

```
#define sdWrite(
    sdp,
    b,
    n)
```

Value:

```
oqWriteTimeout(&(sdp)->oqueue, b, n, TIME_INFINITE)
```

Direct blocking write to a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and writes directly to the output queue. This is faster but cannot be used to write from different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.18 sdWriteTimeout

```
#define sdWriteTimeout(
    sdp,
    b,
    n,
    t)
```

Value:

```
oqWriteTimeout(&(sdp)->oqueue, b, n, t)
```

Direct blocking write to a [SerialDriver](#) with timeout specification.

Note

This function bypasses the indirect access to the channel and writes directly to the output queue. This is faster but cannot be used to write to different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved
in	<i>t</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The number of bytes effectively transferred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.19 sdAsynchronousWrite

```
#define sdAsynchronousWrite(  
    sdp,  
    b,  
    n)
```

Value:

```
oqWriteTimeout(&(sdp)->oqueue, b, n, TIME_IMMEDIATE)
```

Direct non-blocking write to a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and writes directly to the output queue. This is faster but cannot be used to write to different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.20 sdReadI

```
#define sdReadI(  
    sdp,  
    b,  
    n)
```

Value:

```
iqReadI(&(sdp)->iqueue, b, n)
```

Direct non-blocking read from a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Returns

The number of bytes effectively transferred.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.17.3.21 sdRead

```
#define sdRead(
    sdp,
    b,
    n)
```

Value:

```
iqReadTimeout(&(sdp)->iqueue, b, n, TIME_INFINITE)
```

Direct blocking read from a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.22 sdReadTimeout

```
#define sdReadTimeout(
    sdp,
    b,
    n,
    t)
```

Value:

```
iqReadTimeout(&(sdp)->iqueue, b, n, t)
```

Direct blocking read from a [SerialDriver](#) with timeout specification.

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved
in	<i>t</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The number of bytes effectively transferred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.23 sdAsynchronousRead

```
#define sdAsynchronousRead(
    sdp,
    b,
    n)
```

Value:

```
iqReadTimeout(&(sdp)->iqueue, b, n, TIME_IMMEDIATE)
```

Direct non-blocking read from a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>b</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Returns

The number of bytes effectively transferred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.17.3.24 PLATFORM_SERIAL_USE_USART1

```
#define PLATFORM_SERIAL_USE_USART1 FALSE
```

USART1 driver enable switch.

If set to `TRUE` the support for USART1 is included.

Note

The default is `FALSE`.

7.2.9.17.3.25 _serial_driver_data

```
#define _serial_driver_data
```

Value:

```
_base_asynchronous_channel_data
/* Driver state.*/
sdstate_t                state;
/* Input queue.*/
input_queue_t            iqueue;
/* Output queue.*/
output_queue_t           oqueue;
/* Input circular buffer.*/
uint8_t                  ib[SERIAL_BUFFERS_SIZE];
/* Output circular buffer.*/
uint8_t                  ob[SERIAL_BUFFERS_SIZE];
/* End of the mandatory fields.*/
```

`SerialDriver` specific data.

7.2.9.17.4 Typedef Documentation

7.2.9.17.4.1 SerialDriver

```
typedef struct hal_serial_driver SerialDriver
```

Structure representing a serial driver.

7.2.9.17.4.2 SerialConfig

```
typedef struct hal_serial_config SerialConfig
```

PLATFORM Serial Driver configuration structure.

An instance of this structure must be passed to `sdStart()` in order to configure and start a serial driver operations.

Note

This structure content is architecture dependent, each driver implementation defines its own version and the custom static initializers.

7.2.9.17.5 Enumeration Type Documentation

7.2.9.17.5.1 sdstate_t

```
enum sdstate_t
```

Driver state machine possible states.

Enumerator

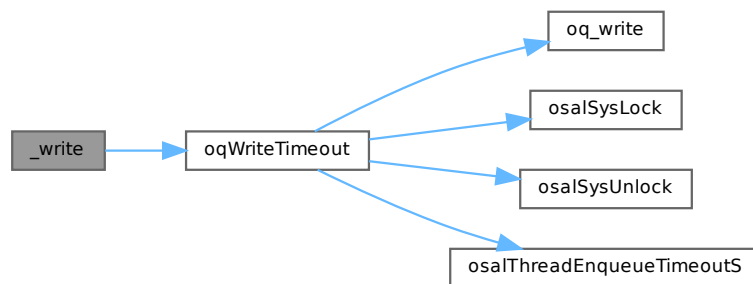
SD_UNINIT	Not initialized.
SD_STOP	Stopped.
SD_READY	Ready.

7.2.9.17.6 Function Documentation

7.2.9.17.6.1 _write()

```
size_t _write (  
    void * ip,  
    const uint8_t * bp,  
    size_t n) [static]
```

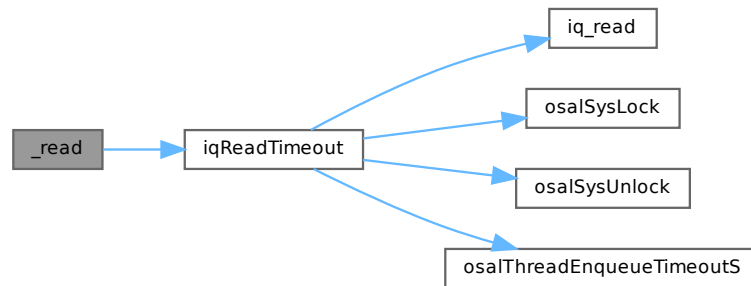
Here is the call graph for this function:



7.2.9.17.6.2 _read()

```
size_t _read (  
    void * ip,  
    uint8_t * bp,  
    size_t n) [static]
```

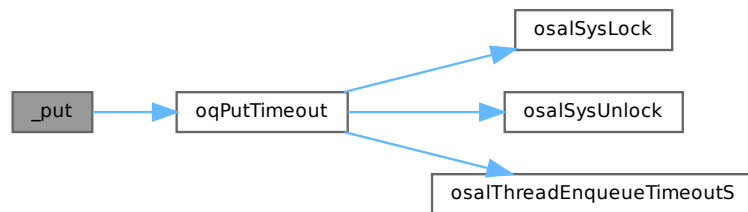
Here is the call graph for this function:



7.2.9.17.6.3 `_put()`

```
msg_t _put (  
    void * ip,  
    uint8_t b) [static]
```

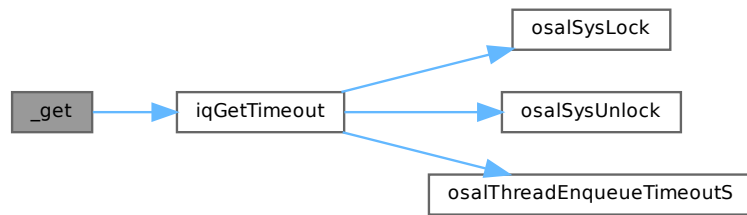
Here is the call graph for this function:



7.2.9.17.6.4 `_get()`

```
msg_t _get (  
    void * ip) [static]
```

Here is the call graph for this function:

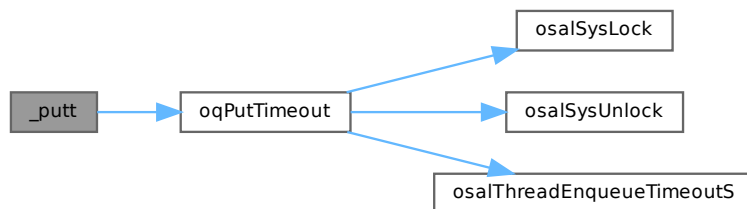


7.2.9.17.6.5 _putt()

```

msg_t _putt (
    void * ip,
    uint8_t b,
    sysinterval_t timeout) [static]
  
```

Here is the call graph for this function:

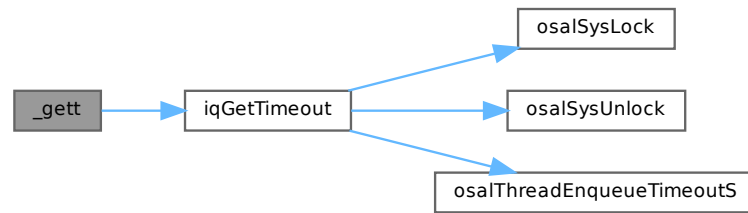


7.2.9.17.6.6 _gett()

```

msg_t _gett (
    void * ip,
    sysinterval_t timeout) [static]
  
```

Here is the call graph for this function:

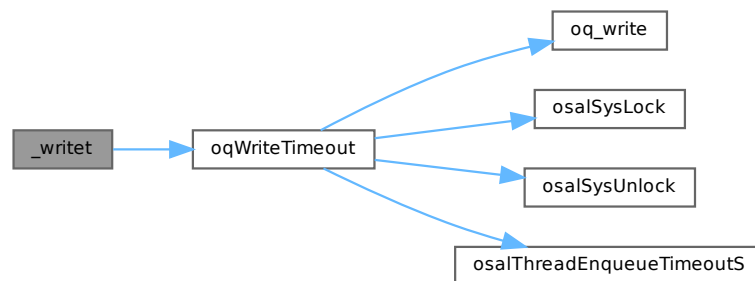


7.2.9.17.6.7 _writet()

```

size_t _writet (
    void * ip,
    const uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]
  
```

Here is the call graph for this function:

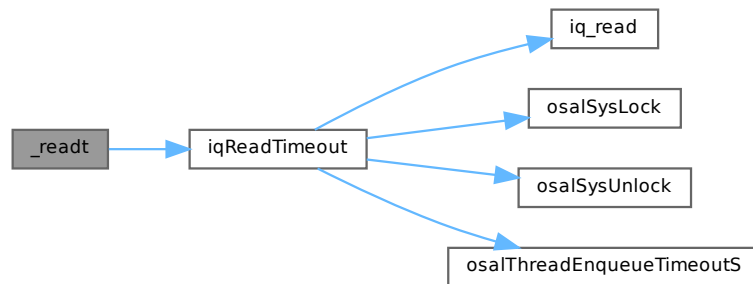


7.2.9.17.6.8 _readt()

```

size_t _readt (
    void * ip,
    uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]
  
```

Here is the call graph for this function:



7.2.9.17.6.9 _ctl()

```

msg_t _ctl (
    void * ip,
    unsigned int operation,
    void * arg) [static]
  
```

7.2.9.17.6.10 sdInit()

```

void sdInit (
    void )
  
```

Serial Driver initialization.

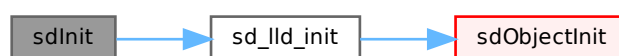
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.17.6.11 sdObjectInit()

```
void sdObjectInit (
    SerialDriver * sdp,
    qnotify_t inotify,
    qnotify_t onotify)
```

Initializes a generic serial driver object.

The HW dependent part of the initialization has to be performed outside, usually in the hardware initialization code.

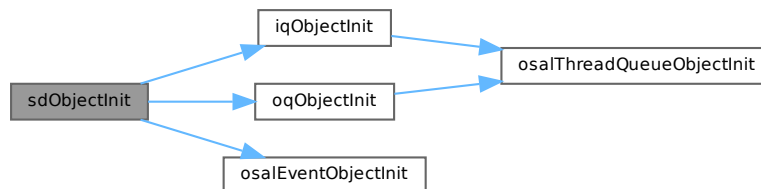
Parameters

out	<i>sdp</i>	pointer to a SerialDriver structure
in	<i>inotify</i>	pointer to a callback function that is invoked when some data is read from the Queue. The value can be <code>NULL</code> .
in	<i>onotify</i>	pointer to a callback function that is invoked when some data is written in the Queue. The value can be <code>NULL</code> .

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.17.6.12 sdStart()**

```
msg_t sdStart (
    SerialDriver * sdp,
    const SerialConfig * config)
```

Configures and starts the driver.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>config</i>	the architecture-dependent serial driver configuration. If this parameter is set to <code>NULL</code> then a default configuration is used.

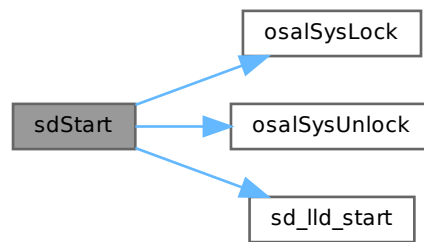
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.17.6.13 sdStop()**

```
void sdStop (  
    SerialDriver * sdp)
```

Stops the driver.

Any thread waiting on the driver's queues will be awakened with the message `MSG_RESET`.

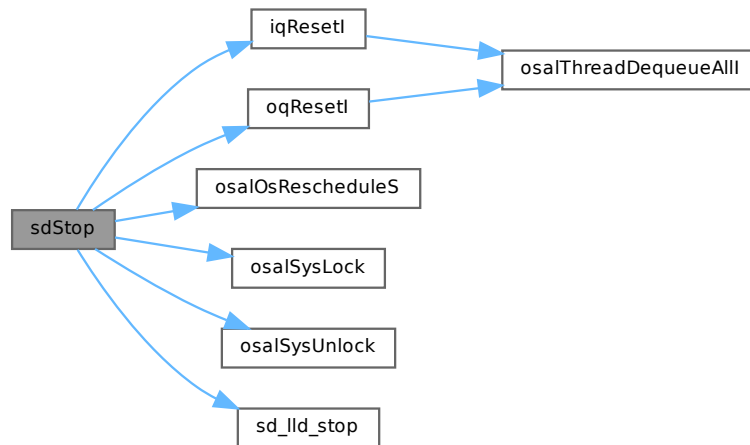
Parameters

in	<i>sdp</i>	pointer to a <i>SerialDriver</i> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.17.6.14 sdIncomingDataI()**

```
void sdIncomingDataI (
    SerialDriver * sdp,
    uint8_t b)
```

Handles incoming data.

This function must be called from the input interrupt service routine in order to enqueue incoming data and generate the related events.

Note

The incoming data event is only generated when the input queue becomes non-empty.

In order to gain some performance it is suggested to not use this function directly but copy this code directly into the interrupt service routine.

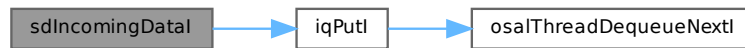
Parameters

in	<i>sdp</i>	pointer to a SerialDriver structure
in	<i>b</i>	the byte to be written in the driver's Input Queue

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.17.6.15 sdRequestDataI()**

```
msg_t sdRequestDataI (
    SerialDriver * sdp)
```

Handles outgoing data.

Must be called from the output interrupt service routine in order to get the next byte to be transmitted.

Note

In order to gain some performance it is suggested to not use this function directly but copy this code directly into the interrupt service routine.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver structure

Returns

The byte value read from the driver's output queue.

Return values

<i>MSG_TIMEOUT</i>	if the queue is empty (the lower driver usually disables the interrupt source when this happens).

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.17.6.16 sdPutWouldBlock()

```
bool sdPutWouldBlock (
    SerialDriver * sdp)
```

Direct output check on a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and checks directly the output queue. This is faster but cannot be used to check different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver structure

Returns

The queue status.

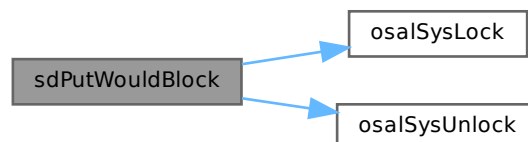
Return values

<i>false</i>	if the next write operation would not block.
<i>true</i>	if the next write operation would block.

Deprecated**Function Class:**

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.17.6.17 sdGetWouldBlock()**

```
bool sdGetWouldBlock (
    SerialDriver * sdp)
```

Direct input check on a [SerialDriver](#).

Note

This function bypasses the indirect access to the channel and checks directly the input queue. This is faster but cannot be used to check different channels implementations.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver structure

Returns

The queue status.

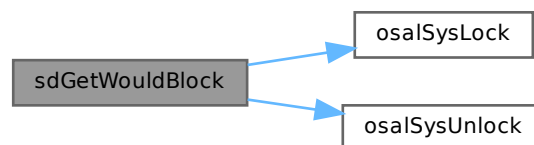
Return values

<i>false</i>	if the next write operation would not block.
<i>true</i>	if the next write operation would block.

Deprecated**Function Class:**

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.17.6.18 sdControl()**

```
msg_t sdControl (  
    SerialDriver * sdp,  
    unsigned int operation,  
    void * arg)
```

Control operation on a serial port.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>operation</i>	control operation code
in, out	<i>arg</i>	operation argument

Returns

The control operation status.

Return values

<i>MSG_OK</i>	in case of success.
<i>MSG_TIMEOUT</i>	in case of operation timeout.
<i>MSG_RESET</i>	in case of operation reset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.17.6.19 sd_lld_init()

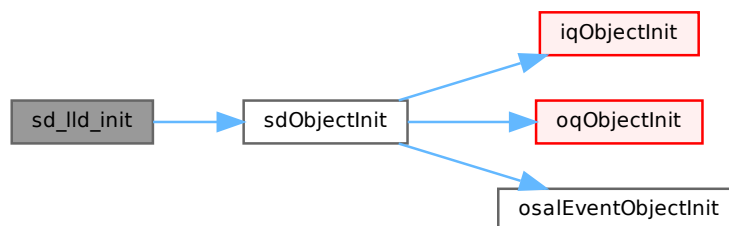
```
void sd_lld_init (
    void )
```

Low level serial driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.17.6.20 sd_lld_start()

```
void sd_lld_start (
    SerialDriver * sdp,
    const SerialConfig * config)
```

Low level serial driver configuration and (re)start.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object
in	<i>config</i>	the architecture-dependent serial driver configuration. If this parameter is set to <code>NULL</code> then a default configuration is used.

Function Class:

Not an API, this function is for internal use only.

7.2.9.17.6.21 sd_llc_stop()

```
void sd_llc_stop (  
    SerialDriver * sdp)
```

Low level serial driver stop.

De-initializes the USART, stops the associated clock, resets the interrupt vector.

Parameters

in	<i>sdp</i>	pointer to a SerialDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.17.7 Variable Documentation**7.2.9.17.7.1 vmt**

```
const struct SerialDriverVMT vmt [static]
```

Initial value:

```
= {  
    (size_t)0,  
    _write, _read, _put, _get,  
    _putt, _gett, _writet, _readt,  
    _ctl  
}
```

7.2.9.17.7.2 SD1

```
SerialDriver SD1
```

USART1 serial driver identifier.

7.2.9.17.7.3 default_config

```
const SerialConfig default_config [static]
```

Initial value:

```
= {
    38400
}
```

Driver default configuration.

7.2.9.18 SIO Driver

Generic SIO Driver.

7.2.9.18.1 Detailed Description

Generic SIO Driver.

This driver abstracts a generic serial communication channel, usually an UART, this driver is similar to Serial and UART drivers but follows a different concept:

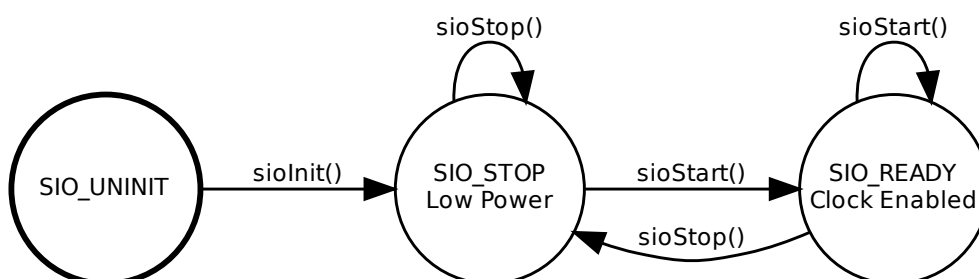
- Very close to HW.
- No buffering done in SW, the driver relies on the peripheral internal FIFO, if any.
- Asynchronous, the API is always non blocking.
- Callbacks capable, operations completion and other events are notified using callbacks.
- Very short code paths, especially in ISRs.

Precondition

In order to use the SIO driver the `HAL_USE_SIO` option must be enabled in `halconf.h`.

7.2.9.18.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



SIO masks offsets

- `#define SIO_EV_RXNOTEMPY_POS 2 /* CHN_INPUT_AVAILABLE */`
- `#define SIO_EV_TXNOTFULL_POS 3 /* CHN_OUTPUT_EMPTY */`
- `#define SIO_EV_TXDONE_POS 4 /* CHN_TRANSMISSION_END */`
- `#define SIO_EV_ALL_ERRORS_POS SIO_EV_PARITY_ERR_POS`
- `#define SIO_EV_PARITY_ERR_POS 5 /* CHN_PARITY_ERROR */`
- `#define SIO_EV_FRAMING_ERR_POS 6 /* CHN_FRAMING_ERROR */`
- `#define SIO_EV_NOISE_ERR_POS 7 /* CHN_NOISE_ERROR */`
- `#define SIO_EV_OVERRUN_ERR_POS 8 /* CHN_OVERRUN_ERROR */`
- `#define SIO_EV_RXIDLE_POS 9 /* CHN_IDLE_DETECTED */`
- `#define SIO_EV_RXBREAK_POS 10 /* CHN_BREAK_DETECTED */`

Event flags (compatible with channel and serial events)

- `#define SIO_EV_NONE 0U`
- `#define SIO_EV_RXNOTEMPY (1U << SIO_EV_RXNOTEMPY_POS)`
- `#define SIO_EV_TXNOTFULL (1U << SIO_EV_TXNOTFULL_POS)`
- `#define SIO_EV_ALL_DATA (SIO_EV_RXNOTEMPY | SIO_EV_TXNOTFULL)`
- `#define SIO_EV_TXDONE (1U << SIO_EV_TXDONE_POS)`
- `#define SIO_EV_PARITY_ERR (1U << SIO_EV_PARITY_ERR_POS)`
- `#define SIO_EV_FRAMING_ERR (1U << SIO_EV_FRAMING_ERR_POS)`
- `#define SIO_EV_NOISE_ERR (1U << SIO_EV_NOISE_ERR_POS)`
- `#define SIO_EV_OVERRUN_ERR (1U << SIO_EV_OVERRUN_ERR_POS)`
- `#define SIO_EV_RXIDLE (1U << SIO_EV_RXIDLE_POS)`
- `#define SIO_EV_RXBREAK (1U << SIO_EV_RXBREAK_POS)`
- `#define SIO_EV_ALL_ERRORS`
- `#define SIO_EV_ALL_EVENTS`

Additional messages

- `#define SIO_MSG_ERRORS (msg_t)1`

SIO configuration options

- `#define SIO_DEFAULT_BITRATE 38400`
Default bit rate.
- `#define SIO_USE_SYNCHRONIZATION TRUE`
Support for thread synchronization API.
- `#define SIO_USE_STREAMS_INTERFACE SIO_USE_SYNCHRONIZATION`
Support for streams interface.

Low level driver helper macros

- `#define __sio_callback(siop)`
SIO callback.
- `#define __sio_wakeup_errors(siop)`
Wakes up because RX errors.
- `#define __sio_wakeup_rx(siop)`
Wakes up the RX-waiting thread.
- `#define __sio_wakeup_rxidle(siop)`
Wakes up the RX-idle-waiting thread.
- `#define __sio_wakeup_tx(siop)`
Wakes up the TX-waiting thread.
- `#define __sio_wakeup_txend(siop)`
Wakes up the TXend-waiting thread.

PLATFORM configuration options

- #define `PLATFORM_SIO_USE_SIO1` `FALSE`
SIO driver enable switch.

Data Structures

- struct `hal_sio_config`
Driver configuration structure.
- struct `sio_driver_vmt`
SIODriver virtual methods table.
- struct `hal_sio_driver`
Structure representing a SIO driver.

Macros

- #define `_sio_driver_methods_base_channel_methods`
SIODriver specific methods.
- #define `sioSetCallbackX`(siop, f)
Associates a callback to the SIO instance.
- #define `siolsRXEmptyX`(siop)
Determines the state of the RX FIFO.
- #define `siolsRXIdleX`(siop)
Determines the activity state of the receiver.
- #define `sioHasRXErrorsX`(siop)
Determines if RX has pending errors to be read and cleared.
- #define `siolsTXFullX`(siop)
Determines the state of the TX FIFO.
- #define `siolsTXOngoingX`(siop)
Determines the transmission state.
- #define `sioWriteEnableFlagsX`(siop, mask)
Writes the enabled events mask.
- #define `sioSetEnableFlagsX`(siop, mask)
Sets flags into the enabled events flags mask.
- #define `sioClearEnableFlagsX`(siop, mask)
Clears flags from the enabled events flags mask.
- #define `sioGetEnableFlagsX`(siop)
Return the enabled condition flags mask.
- #define `sioGetAndClearErrorsX`(siop)
Get and clears SIO error event flags.
- #define `sioGetAndClearEventsX`(siop)
Get and clears SIO event flags.
- #define `sioGetEventsX`(siop)
Returns the pending SIO event flags.
- #define `sioGetX`(siop)
Returns one frame from the RX FIFO.
- #define `sioPutX`(siop, data)
Pushes one frame into the TX FIFO.
- #define `sioAsyncReadX`(siop, size, buffer)
Reads data from the RX FIFO.

- `#define sioAsyncWriteX(siop, size, buffer)`
Writes data into the TX FIFO.
- `#define sioControlX(siop, operation, arg)`
Control operation on a serial port.
- `#define __sio_reloc_field(v, m, s, d)`
Relocates a bit field.
- `#define sio_lld_driver_fields uint32_t dummy;`
Low level fields of the SIO driver structure.
- `#define sio_lld_config_fields uint32_t dummy;`
Low level fields of the SIO configuration structure.
- `#define sio_lld_is_rx_empty(siop)`
Determines the state of the RX FIFO.
- `#define sio_lld_is_rx_idle(siop)`
Determines the activity state of the receiver.
- `#define sio_lld_has_rx_errors(siop)`
Determines if RX has pending error events to be read and cleared.
- `#define sio_lld_is_tx_full(siop)`
Determines the state of the TX FIFO.
- `#define sio_lld_is_tx_ongoing(siop)`
Determines the transmission state.

Typedefs

- `typedef eventflags_t sioevents_t`
Type of event flags.
- `typedef struct hal_sio_driver SIODriver`
Type of structure representing a SIO driver.
- `typedef struct hal_sio_config SIOConfig`
Type of structure representing a SIO configuration.
- `typedef void(* siocb_t) (SIODriver *siop)`
Generic SIO notification callback type.

Enumerations

- `enum siostate_t { SIO_UNINIT = 0 , SIO_STOP = 1 , SIO_READY = 2 }`
Driver state machine possible states.

Functions

- `static size_t sync_write (void *ip, const uint8_t *bp, size_t n, sysinterval_t timeout)`
- `static size_t sync_read (void *ip, uint8_t *bp, size_t n, sysinterval_t timeout)`
- `static size_t __write (void *ip, const uint8_t *bp, size_t n)`
- `static size_t __read (void *ip, uint8_t *bp, size_t n)`
- `static msg_t __put (void *ip, uint8_t b)`
- `static msg_t __get (void *ip)`
- `static msg_t __putt (void *ip, uint8_t b, sysinterval_t timeout)`
- `static msg_t __gett (void *ip, sysinterval_t timeout)`
- `static size_t __writet (void *ip, const uint8_t *bp, size_t n, sysinterval_t timeout)`
- `static size_t __readt (void *ip, uint8_t *bp, size_t n, sysinterval_t timeout)`
- `static msg_t __ctl (void *ip, unsigned int operation, void *arg)`

- void `sioInit` (void)
SIO Driver initialization.
- void `sioObjectInit` (SIODriver *siop)
Initializes the standard part of a SIODriver structure.
- `msg_t` `sioStart` (SIODriver *siop, const SIOConfig *config)
Configures and activates the SIO peripheral.
- void `sioStop` (SIODriver *siop)
Deactivates the SIO peripheral.
- void `sioWriteEnableFlags` (SIODriver *siop, `sioevents_t` mask)
Writes the enabled events flags mask.
- void `sioSetEnableFlags` (SIODriver *siop, `sioevents_t` mask)
Sets flags into the enabled events flags mask.
- void `sioClearEnableFlags` (SIODriver *siop, `sioevents_t` mask)
Clears flags from the enabled events flags mask.
- `sioevents_t` `sioGetAndClearErrors` (SIODriver *siop)
Get and clears SIO error event flags.
- `sioevents_t` `sioGetAndClearEvents` (SIODriver *siop)
Get and clears SIO event flags.
- `sioevents_t` `sioGetEvents` (SIODriver *siop)
Returns the pending SIO event flags.
- `size_t` `sioAsyncRead` (SIODriver *siop, `uint8_t` *buffer, `size_t` n)
Reads data from the RX FIFO.
- `size_t` `sioAsyncWrite` (SIODriver *siop, const `uint8_t` *buffer, `size_t` n)
Writes data into the TX FIFO.
- `msg_t` `sioSynchronizeRX` (SIODriver *siop, `sysinterval_t` timeout)
Synchronizes with RX FIFO data availability.
- `msg_t` `sioSynchronizeRXIdle` (SIODriver *siop, `sysinterval_t` timeout)
Synchronizes with RX going idle.
- `msg_t` `sioSynchronizeTX` (SIODriver *siop, `sysinterval_t` timeout)
Synchronizes with TX FIFO space availability.
- `msg_t` `sioSynchronizeTXEnd` (SIODriver *siop, `sysinterval_t` timeout)
Synchronizes with TX completion.
- void `sio_llc_init` (void)
Low level SIO driver initialization.
- `msg_t` `sio_llc_start` (SIODriver *siop)
Configures and activates the SIO peripheral.
- void `sio_llc_stop` (SIODriver *siop)
Deactivates the SIO peripheral.
- void `sio_llc_update_enable_flags` (SIODriver *siop)
Enable flags change notification.
- `sioevents_t` `sio_llc_get_and_clear_errors` (SIODriver *siop)
Get and clears SIO error event flags.
- `sioevents_t` `sio_llc_get_and_clear_events` (SIODriver *siop)
Get and clears SIO event flags.
- `sioevents_t` `sio_llc_get_events` (SIODriver *siop)
Returns the pending SIO event flags.
- `size_t` `sio_llc_read` (SIODriver *siop, `uint8_t` *buffer, `size_t` n)
Reads data from the RX FIFO.
- `size_t` `sio_llc_write` (SIODriver *siop, const `uint8_t` *buffer, `size_t` n)
Writes data into the TX FIFO.
- `msg_t` `sio_llc_get` (SIODriver *siop)

- *Returns one frame from the RX FIFO.*
- void [sio_lld_put](#) ([SIODriver](#) *siop, uint_fast16_t data)
Pushes one frame into the TX FIFO.
- [msg_t sio_lld_control](#) ([SIODriver](#) *siop, unsigned int operation, void *arg)
Control operation on a serial port.
- void [sio_lld_serve_interrupt](#) ([SIODriver](#) *siop)
Serves an UART interrupt.

Variables

- static const struct [sio_driver_vmt](#) vmt
- [SIODriver](#) SIOD1
SIO1 driver identifier.

7.2.9.18.3 Macro Definition Documentation

7.2.9.18.3.1 SIO_EV_RXNOTEMPTY_POS

```
#define SIO_EV_RXNOTEMPTY_POS 2 /* CHN_INPUT_AVAILABLE */
```

7.2.9.18.3.2 SIO_EV_TXNOTFULL_POS

```
#define SIO_EV_TXNOTFULL_POS 3 /* CHN_OUTPUT_EMPTY */
```

7.2.9.18.3.3 SIO_EV_TXDONE_POS

```
#define SIO_EV_TXDONE_POS 4 /* CHN_TRANSMISSION_END */
```

7.2.9.18.3.4 SIO_EV_ALL_ERRORS_POS

```
#define SIO_EV_ALL_ERRORS_POS SIO_EV_PARITY_ERR_POS
```

7.2.9.18.3.5 SIO_EV_PARITY_ERR_POS

```
#define SIO_EV_PARITY_ERR_POS 5 /* CHN_PARITY_ERROR */
```

7.2.9.18.3.6 SIO_EV_FRAMING_ERR_POS

```
#define SIO_EV_FRAMING_ERR_POS 6 /* CHN_FRAMING_ERROR */
```

7.2.9.18.3.7 SIO_EV_NOISE_ERR_POS

```
#define SIO_EV_NOISE_ERR_POS 7 /* CHN_NOISE_ERROR */
```

7.2.9.18.3.8 SIO_EV_OVERRUN_ERR_POS

```
#define SIO_EV_OVERRUN_ERR_POS 8 /* CHN_OVERRUN_ERROR */
```

7.2.9.18.3.9 SIO_EV_RXIDLE_POS

```
#define SIO_EV_RXIDLE_POS 9 /* CHN_IDLE_DETECTED */
```

7.2.9.18.3.10 SIO_EV_RXBREAK_POS

```
#define SIO_EV_RXBREAK_POS 10 /* CHN_BREAK_DETECTED */
```

7.2.9.18.3.11 SIO_EV_NONE

```
#define SIO_EV_NONE 0U
```

7.2.9.18.3.12 SIO_EV_RXNOTEMPTY

```
#define SIO_EV_RXNOTEMPTY (1U << SIO_EV_RXNOTEMPTY_POS)
```

7.2.9.18.3.13 SIO_EV_TXNOTFULL

```
#define SIO_EV_TXNOTFULL (1U << SIO_EV_TXNOTFULL_POS)
```

7.2.9.18.3.14 SIO_EV_ALL_DATA

```
#define SIO_EV_ALL_DATA (SIO_EV_RXNOTEMPTY | SIO_EV_TXNOTFULL)
```

7.2.9.18.3.15 SIO_EV_TXDONE

```
#define SIO_EV_TXDONE (1U << SIO_EV_TXDONE_POS)
```

7.2.9.18.3.16 SIO_EV_PARITY_ERR

```
#define SIO_EV_PARITY_ERR (1U << SIO_EV_PARITY_ERR_POS)
```

7.2.9.18.3.17 SIO_EV_FRAMING_ERR

```
#define SIO_EV_FRAMING_ERR (1U << SIO_EV_FRAMING_ERR_POS)
```

7.2.9.18.3.18 SIO_EV_NOISE_ERR

```
#define SIO_EV_NOISE_ERR (1U << SIO_EV_NOISE_ERR_POS)
```

7.2.9.18.3.19 SIO_EV_OVERRUN_ERR

```
#define SIO_EV_OVERRUN_ERR (1U << SIO_EV_OVERRUN_ERR_POS)
```

7.2.9.18.3.20 SIO_EV_RXIDLE

```
#define SIO_EV_RXIDLE (1U << SIO_EV_RXIDLE_POS)
```

7.2.9.18.3.21 SIO_EV_RXBREAK

```
#define SIO_EV_RXBREAK (1U << SIO_EV_RXBREAK_POS)
```

7.2.9.18.3.22 SIO_EV_ALL_ERRORS

```
#define SIO_EV_ALL_ERRORS
```

Value:

```
(SIO_EV_PARITY_ERR |
SIO_EV_FRAMING_ERR |
SIO_EV_OVERRUN_ERR |
SIO_EV_NOISE_ERR   |
SIO_EV_RXBREAK)
```

7.2.9.18.3.23 SIO_EV_ALL_EVENTS

```
#define SIO_EV_ALL_EVENTS
```

Value:

```
(SIO_EV_ALL_DATA |
SIO_EV_ALL_ERRORS |
SIO_EV_TXDONE   |
SIO_EV_RXIDLE)
```

7.2.9.18.3.24 SIO_MSG_ERRORS

```
#define SIO_MSG_ERRORS (msg_t)1
```

7.2.9.18.3.25 SIO_DEFAULT_BITRATE

```
#define SIO_DEFAULT_BITRATE 38400
```

Default bit rate.

Configuration parameter, this is the baud rate selected for the default configuration.

7.2.9.18.3.26 SIO_USE_SYNCHRONIZATION

```
#define SIO_USE_SYNCHRONIZATION TRUE
```

Support for thread synchronization API.

7.2.9.18.3.27 SIO_USE_STREAMS_INTERFACE

```
#define SIO_USE_STREAMS_INTERFACE SIO_USE_SYNCHRONIZATION
```

Support for streams interface.

7.2.9.18.3.28 _sio_driver_methods

```
#define _sio_driver_methods _base_channel_methods
```

[SIODriver](#) specific methods.

7.2.9.18.3.29 sioSetCallbackX

```
#define sioSetCallbackX(  
    siop,  
    f)
```

Value:

```
(siop)->cb = (f)
```

Associates a callback to the SIO instance.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>f</i>	callback to be associated

7.2.9.18.3.30 sioIsRXEmptyX

```
#define sioIsRXEmptyX(  
    siop)
```

Value:

```
sio_ll_d_is_rx_empty(siop)
```

Determines the state of the RX FIFO.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The RX FIFO state.

Return values

<i>false</i>	if RX FIFO is not empty.
<i>true</i>	if RX FIFO is empty.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.31 siolsRXIdleX

```
#define siolsRXIdleX(  
    siop)
```

Value:

`sio_llc_is_rx_idle`(siop)

Determines the activity state of the receiver.

Parameters

in	<i>siop</i>	pointer to the <code>SIODriver</code> object

Returns

The RX activity state.

Return values

<i>false</i>	if RX is in active state.
<i>true</i>	if RX is in idle state.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.32 sioHasRXErrorsX

```
#define sioHasRXErrorsX(  
    siop)
```

Value:

`sio_llc_has_rx_errors`(siop)

Determines if RX has pending errors to be read and cleared.

Note

Only errors are handled, data and idle events are not considered.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The RX error events.

Return values

<i>false</i>	if RX has no pending events.
<i>true</i>	if RX has pending events.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.33 siolsTXFullX

```
#define siolsTXFullX(  
    siop)
```

Value:

```
sio\_lld\_is\_tx\_full(siop)
```

Determines the state of the TX FIFO.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The TX FIFO state.

Return values

<i>false</i>	if TX FIFO is not full.
<i>true</i>	if TX FIFO is full.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.34 siolsTXOngoingX

```
#define siolsTXOngoingX(  
    siop)
```

Value:

```
sio\_lld\_is\_tx\_ongoing(siop)
```

Determines the transmission state.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The TX FIFO state.

Return values

<i>false</i>	if transmission is over.
<i>true</i>	if transmission is ongoing.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.35 sioWriteEnableFlagsX

```
#define sioWriteEnableFlagsX(  
    siop,  
    mask)
```

Value:

```
do {  
    (siop)->enabled = (mask);  
    sio_llld_update_enable_flags(siop);  
} while (false)
```

Writes the enabled events mask.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>mask</i>	enabled events mask to be written

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.36 sioSetEnableFlagsX

```
#define sioSetEnableFlagsX(  
    siop,  
    mask)
```

Value:

```
do {  
    (siop)->enabled |= (mask);  
    sio_llld_update_enable_flags(siop);  
} while (false)
```

Sets flags into the enabled events flags mask.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>mask</i>	enabled events mask to be set

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.37 **sioClearEnableFlagsX**

```
#define sioClearEnableFlagsX(
    siop,
    mask)
```

Value:

```
do {
    (siop)->enabled &= ~(mask);
    sio_lld_update_enable_flags(siop);
} while (false)
```

Clears flags from the enabled events flags mask.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>mask</i>	enabled events mask to be cleared

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.38 **sioGetEnableFlagsX**

```
#define sioGetEnableFlagsX(
    siop)
```

Value:

```
(siop)->enabled
```

Return the enabled condition flags mask.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.39 **sioGetAndClearErrorsX**

```
#define sioGetAndClearErrorsX(
    siop)
```

Value:

```
sio_lld_get_and_clear_errors(siop)
```

Get and clears SIO error event flags.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The pending error event flags.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.40 sioGetAndClearEventsX

```
#define sioGetAndClearEventsX(  
    siop)
```

Value:

[sio_lld_get_and_clear_events](#)(*siop*)

Get and clears SIO event flags.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The pending event flags.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.41 sioGetEventsX

```
#define sioGetEventsX(  
    siop)
```

Value:

[sio_lld_get_events](#)(*siop*)

Returns the pending SIO event flags.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The pending event flags.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.42 sioGetX

```
#define sioGetX(  
    siop)
```

Value:

`sio_lld_get`(*siop*)

Returns one frame from the RX FIFO.

Note

If the FIFO is empty then the returned value is unpredictable.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The frame from RX FIFO.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.43 sioPutX

```
#define sioPutX(  
    siop,  
    data)
```

Value:

`sio_lld_put`(*siop*, *data*)

Pushes one frame into the TX FIFO.

Note

If the FIFO is full then the behavior is unpredictable.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>data</i>	frame to be written

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.44 sioAsyncReadX

```
#define sioAsyncReadX(  
    siop,  
    size,  
    buffer)
```

Value:

`sio_ll_read`(siop, size, buffer)

Reads data from the RX FIFO.

This function is non-blocking, data is read if present and the effective amount is returned.

Note

This function can be called from any context but it is meant to be called from the `rxne_cb` callback handler.

Parameters

in	<i>siop</i>	pointer to the <code>SIODriver</code> object
in	<i>size</i>	maximum number of frames to read
in	<i>buffer</i>	buffer for the received data

Returns

The number of received frames.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.45 sioAsyncWriteX

```
#define sioAsyncWriteX(  
    siop,  
    size,  
    buffer)
```

Value:

`sio_ll_write`(siop, size, buffer)

Writes data into the TX FIFO.

This function is non-blocking, data is written if there is space in the FIFO and the effective amount is returned.

Note

This function can be called from any context but it is meant to be called from the `txnf_cb` callback handler.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>size</i>	maximum number of frames to read
out	<i>buffer</i>	buffer containing the data to be transmitted

Returns

The number of transmitted frames.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.46 sioControlX

```
#define sioControlX(
    siop,
    operation,
    arg)
```

Value:

[sio_lld_control](#)(siop, operation, arg)

Control operation on a serial port.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>operation</i>	control operation code
in, out	<i>arg</i>	operation argument

Returns

The control operation status.

Return values

<i>MSG_OK</i>	in case of success.
<i>MSG_TIMEOUT</i>	in case of operation timeout.
<i>MSG_RESET</i>	in case of operation reset.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.18.3.47 __sio_callback

```
#define __sio_callback(
    siop)
```

Value:

```
do {
    if ((siop)->cb != NULL) {
        (siop)->cb(siop);
    }
} while (false)
```

SIO callback.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.48 __sio_wakeup_errors

```
#define __sio_wakeup_errors(  
    siop)
```

Value:

```
do {  
    osalSysLockFromISR();  
    osalThreadResumeI(&(siop)->sync_rx, SIO_MSG_ERRORS);  
    osalThreadResumeI(&(siop)->sync_rxidle, SIO_MSG_ERRORS);  
    osalSysUnlockFromISR();  
} while (false)
```

Wakes up because RX errors.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.49 __sio_wakeup_rx

```
#define __sio_wakeup_rx(  
    siop)
```

Value:

```
do {  
    osalSysLockFromISR();  
    osalThreadResumeI(&(siop)->sync_rx, MSG_OK);  
    osalSysUnlockFromISR();  
} while (false)
```

Wakes up the RX-waiting thread.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.50 __sio_wakeup_rxidle

```
#define __sio_wakeup_rxidle(  
    siop)
```

Value:

```
do {  
    osalSysLockFromISR();  
    osalThreadResumeI(&(siop)->sync_rxidle, MSG_OK);  
    osalSysUnlockFromISR();  
} while (false)
```

Wakes up the RX-idle-waiting thread.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.51 __sio_wakeup_tx

```
#define __sio_wakeup_tx(  
    siop)
```

Value:

```
do {  
    osalSysLockFromISR();  
    osalThreadResumeI(&(siop)->sync_tx, MSG_OK);  
    osalSysUnlockFromISR();  
} while (false)
```

Wakes up the TX-waiting thread.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.52 __sio_wakeup_txend

```
#define __sio_wakeup_txend(  
    siop)
```

Value:

```
do {  
    osalSysLockFromISR();  
    osalThreadResumeI(&(siop)->sync_txend, MSG_OK);  
    osalSysUnlockFromISR();  
} while (false)
```

Wakes up the TXend-waiting thread.

Parameters

in	<i>sio</i> p	pointer to the SIODriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.53 __sio_reloc_field

```
#define __sio_reloc_field(
    v,
    m,
    s,
    d)
```

Value:

```
(( ((v) & m) » (s)) « (d))
```

Relocates a bit field.

Parameters

in	<i>v</i>	value
in	<i>m</i>	mask of the bit field
in	<i>s</i>	source bit offset
in	<i>d</i>	destination bit offset

7.2.9.18.3.54 PLATFORM_SIO_USE_SIO1

```
#define PLATFORM_SIO_USE_SIO1 FALSE
```

SIO driver enable switch.

If set to `TRUE` the support for SIO1 is included.

Note

The default is `FALSE`.

7.2.9.18.3.55 sio_lld_driver_fields

```
#define sio_lld_driver_fields uint32_t dummy;
```

Low level fields of the SIO driver structure.

7.2.9.18.3.56 sio_llc_config_fields

```
#define sio_llc_config_fields  uint32_t dummy;
```

Low level fields of the SIO configuration structure.

7.2.9.18.3.57 sio_llc_is_rx_empty

```
#define sio_llc_is_rx_empty(  
    siop)
```

Value:

false

Determines the state of the RX FIFO.

Parameters

in	siop	pointer to the SIODriver object

Returns

The RX FIFO state.

Return values

false	if RX FIFO is not empty
true	if RX FIFO is empty

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.58 sio_llc_is_rx_idle

```
#define sio_llc_is_rx_idle(  
    siop)
```

Value:

false

Determines the activity state of the receiver.

Parameters

in	siop	pointer to the SIODriver object

Returns

The RX activity state.

Return values

<i>false</i>	if RX is in active state.
<i>true</i>	if RX is in idle state.

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.59 sio_lld_has_rx_errors

```
#define sio_lld_has_rx_errors(  
    siop)
```

Value:

false

Determines if RX has pending error events to be read and cleared.

Note

Only error and protocol errors are handled, data events are not considered.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The RX error events.

Return values

<i>false</i>	if RX has no pending events
<i>true</i>	if RX has pending events

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.60 sio_lld_is_tx_full

```
#define sio_lld_is_tx_full(  
    siop)
```

Value:

false

Determines the state of the TX FIFO.

Parameters

in	<i>sio</i> p	pointer to the SIODriver object

Returns

The TX FIFO state.

Return values

<i>false</i>	if TX FIFO is not full
<i>true</i>	if TX FIFO is full

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.3.61 sio_llc_is_tx_ongoing

```
#define sio_llc_is_tx_ongoing(  
    sio
```

Value:

false

Determines the transmission state.

Parameters

in	<i>sio</i> p	pointer to the SIODriver object

Returns

The TX FIFO state.

Return values

<i>false</i>	if transmission is idle
<i>true</i>	if transmission is ongoing

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.4 Typedef Documentation**7.2.9.18.4.1 sioevents_t**

```
typedef eventflags_t sioevents_t
```

Type of event flags.

7.2.9.18.4.2 SIODriver

```
typedef struct hal_sio_driver SIODriver
```

Type of structure representing a SIO driver.

7.2.9.18.4.3 SIOConfig

```
typedef struct hal_sio_config SIOConfig
```

Type of structure representing a SIO configuration.

7.2.9.18.4.4 siocb_t

```
typedef void(* siocb_t) (SIODriver *siop)
```

Generic SIO notification callback type.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

7.2.9.18.5 Enumeration Type Documentation

7.2.9.18.5.1 siostate_t

```
enum siostate_t
```

Driver state machine possible states.

Enumerator

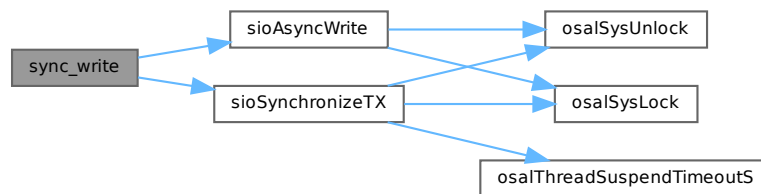
SIO_UNINIT	Not initialized.
SIO_STOP	Stopped.
SIO_READY	Ready.

7.2.9.18.6 Function Documentation

7.2.9.18.6.1 sync_write()

```
size_t sync_write (
    void * ip,
    const uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]
```

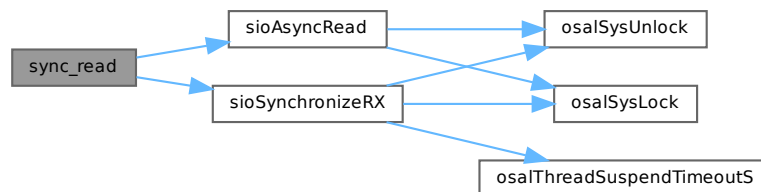
Here is the call graph for this function:



7.2.9.18.6.2 sync_read()

```
size_t sync_read (
    void * ip,
    uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]
```

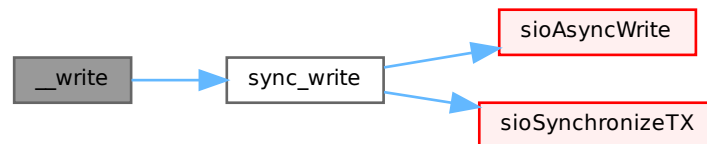
Here is the call graph for this function:



7.2.9.18.6.3 __write()

```
size_t __write (  
    void * ip,  
    const uint8_t * bp,  
    size_t n) [static]
```

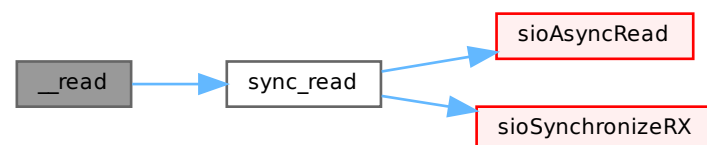
Here is the call graph for this function:



7.2.9.18.6.4 __read()

```
size_t __read (  
    void * ip,  
    uint8_t * bp,  
    size_t n) [static]
```

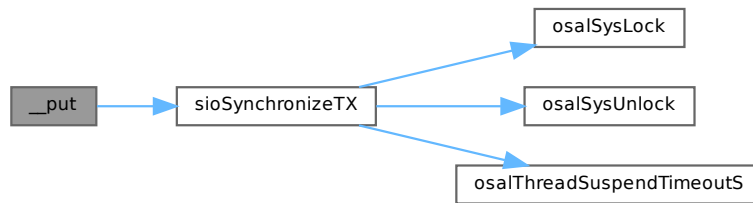
Here is the call graph for this function:



7.2.9.18.6.5 __put()

```
msg_t __put (  
    void * ip,  
    uint8_t b) [static]
```


Here is the call graph for this function:

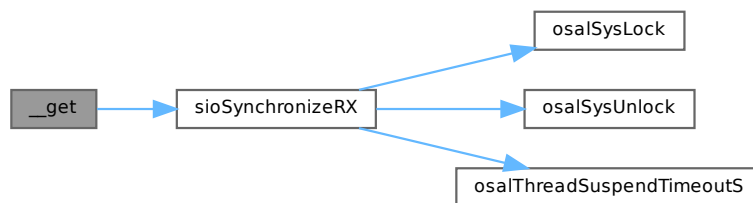


7.2.9.18.6.6 __get()

```

msg_t __get (
    void * ip) [static]
  
```

Here is the call graph for this function:

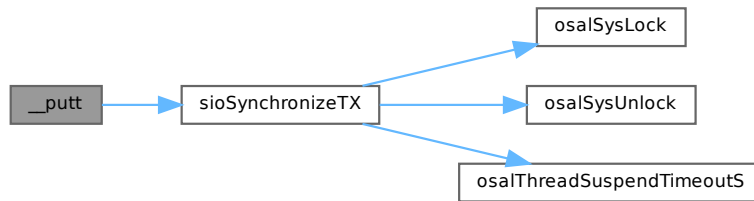


7.2.9.18.6.7 __putt()

```

msg_t __putt (
    void * ip,
    uint8_t b,
    sysinterval_t timeout) [static]
  
```

Here is the call graph for this function:

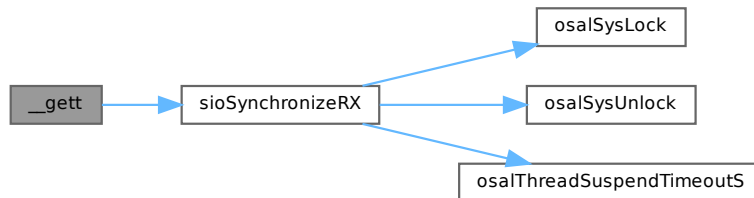


7.2.9.18.6.8 __gett()

```

msg_t __gett (
    void * ip,
    sysinterval_t timeout) [static]
  
```

Here is the call graph for this function:

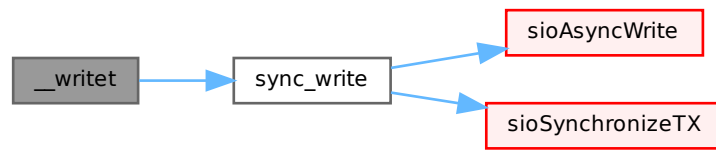


7.2.9.18.6.9 __writet()

```

size_t __writet (
    void * ip,
    const uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]
  
```

Here is the call graph for this function:

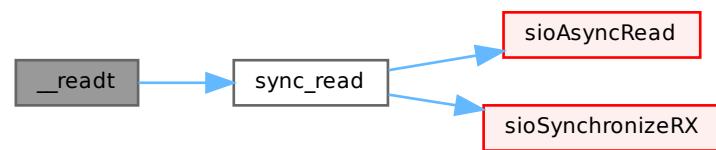


7.2.9.18.6.10 __readt()

```

size_t __readt (
    void * ip,
    uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]
  
```

Here is the call graph for this function:



7.2.9.18.6.11 __ctl()

```

msg_t __ctl (
    void * ip,
    unsigned int operation,
    void * arg) [static]
  
```

Here is the call graph for this function:



7.2.9.18.6.12 `sioInit()`

```
void sioInit (
    void )
```

SIO Driver initialization.

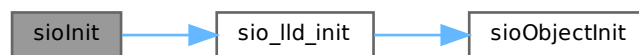
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.18.6.13 `sioObjectInit()`

```
void sioObjectInit (
    SIODriver * siop)
```

Initializes the standard part of a `SIODriver` structure.

Parameters

out	<i>siop</i>	pointer to the <code>SIODriver</code> object

Function Class:

Object or module initializer function.

7.2.9.18.6.14 `sioStart()`

```
msg_t sioStart (
    SIODriver * siop,
    const SIOConfig * config)
```

Configures and activates the SIO peripheral.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>config</i>	pointer to the SIOConfig object, can be <code>NULL</code> if the default configuration is desired

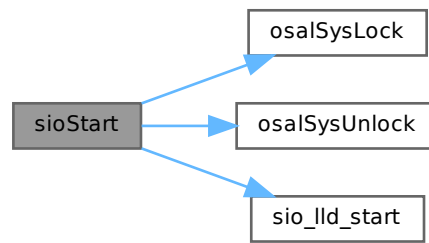
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.18.6.15 sioStop()**

```
void sioStop (  
    SIODriver * siop)
```

Deactivates the SIO peripheral.

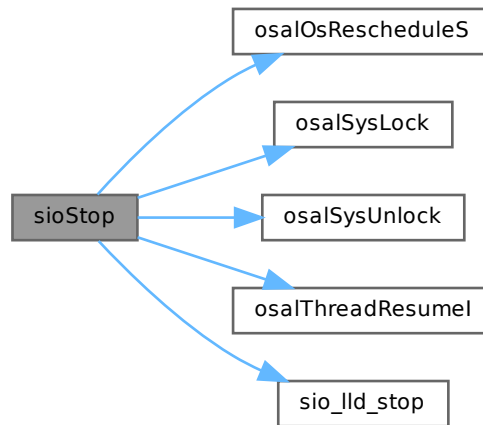
Parameters

in	<i>siop</i>	pointer to the SIODriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.18.6.16 sioWriteEnableFlags()**

```
void sioWriteEnableFlags (
    SIODriver * siop,
    sioevents_t mask)
```

Writes the enabled events flags mask.

Parameters

in	<i>siop</i>	pointer to the <code>SIODriver</code> object
in	<i>mask</i>	enabled events mask to be written

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.18.6.17 sioSetEnableFlags()

```
void sioSetEnableFlags (  
    SIODriver * siop,  
    sioevents_t mask)
```

Sets flags into the enabled events flags mask.

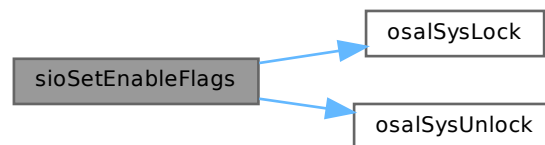
Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>mask</i>	enabled events mask to be set

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.18.6.18 sioClearEnableFlags()**

```
void sioClearEnableFlags (  
    SIODriver * siop,  
    sioevents_t mask)
```

Clears flags from the enabled events flags mask.

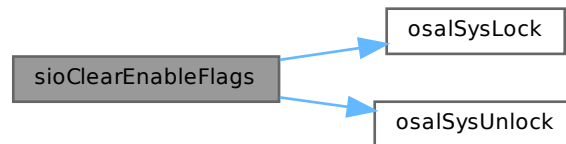
Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>mask</i>	enabled events mask to be cleared

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.18.6.19 sioGetAndClearErrors()**

```
sioevents_t sioGetAndClearErrors (  
    SIODriver * siop)
```

Get and clears SIO error event flags.

Parameters

in	<i>siop</i>	pointer to the <code>SIODriver</code> object

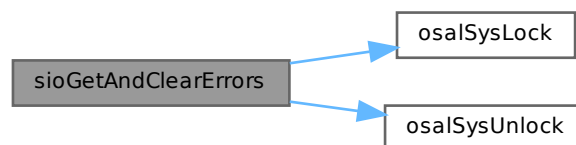
Returns

The pending error event flags.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.18.6.20 sioGetAndClearEvents()

```
sioevents_t sioGetAndClearEvents (  
    SIODriver * sio)
```

Get and clears SIO event flags.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

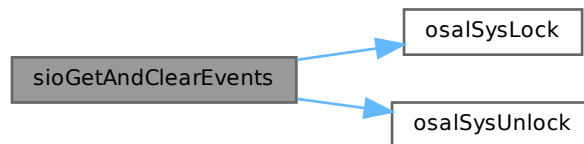
Returns

The pending event flags.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.18.6.21 sioGetEvents()**

```
sioevents_t sioGetEvents (  
    SIODriver * siop)
```

Returns the pending SIO event flags.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

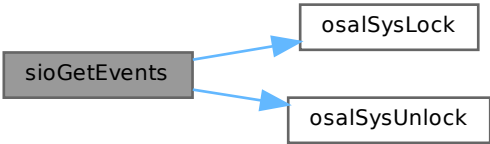
Returns

The pending event flags.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.18.6.22 sioAsyncRead()

```
size_t sioAsyncRead (  
    SIODriver * siop,  
    uint8_t * buffer,  
    size_t n)
```

Reads data from the RX FIFO.

This function is non-blocking, data is read if present and the effective amount is returned.

Note

This function can be called from any context but it is meant to be called from the rxne_cb callback handler.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>buffer</i>	buffer for the received data
in	<i>n</i>	maximum number of frames to read

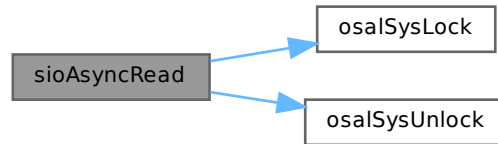
Returns

The number of received frames.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.18.6.23 sioAsyncWrite()**

```
size_t sioAsyncWrite (  
    SIODriver * siop,  
    const uint8_t * buffer,  
    size_t n)
```

Writes data into the TX FIFO.

This function is non-blocking, data is written if there is space in the FIFO and the effective amount is returned.

Note

This function can be called from any context but it is meant to be called from the `txnf_cb` callback handler.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
out	<i>buffer</i>	buffer containing the data to be transmitted
in	<i>n</i>	maximum number of frames to read

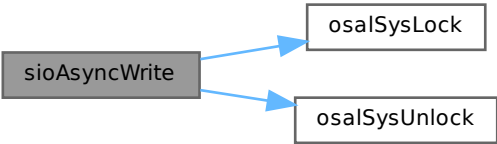
Returns

The number of transmitted frames.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.18.6.24 sioSynchronizeRX()

```
msg_t sioSynchronizeRX (  
    SIODriver * siop,  
    sysinterval_t timeout)
```

Synchronizes with RX FIFO data availability.

Note

The exact behavior depends on low level FIFO settings such as thresholds, etc.

This function can only be called by a single thread at time.

Parameters

in	siop	pointer to an SIODriver structure
in	timeout	synchronization timeout

Returns

The synchronization result.

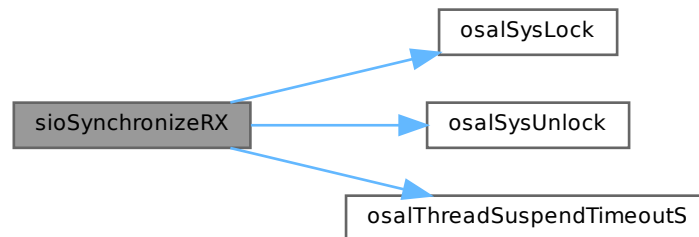
Return values

MSG_OK	if there is data in the RX FIFO.
MSG_TIMEOUT	if synchronization timed out.
MSG_RESET	it the operation has been stopped while waiting.
SIO_MSG_ERRORS	if RX errors occurred before or during wait.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.18.6.25 sioSynchronizeRXIdle()**

```

msg_t sioSynchronizeRXIdle (
    SIODriver * siop,
    sysinterval_t timeout)
  
```

Synchronizes with RX going idle.

Note

This function can only be called by a single thread at time.

Parameters

in	<i>siop</i>	pointer to an SIODriver structure
in	<i>timeout</i>	synchronization timeout

Returns

The synchronization result.

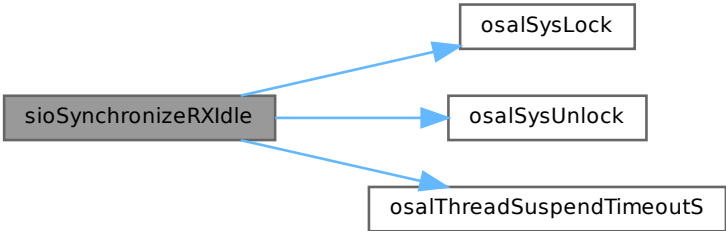
Return values

<i>MSG_OK</i>	if RW went idle.
<i>MSG_TIMEOUT</i>	if synchronization timed out.
<i>MSG_RESET</i>	if the operation has been stopped while waiting.
<i>SIO_MSG_ERRORS</i>	if RX errors occurred before or during wait.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.18.6.26 sioSynchronizeTX()

```
msg_t sioSynchronizeTX (  
    SIODriver * siop,  
    sysinterval_t timeout)
```

Synchronizes with TX FIFO space availability.

Note

The exact behavior depends on low level FIFO settings such as thresholds, etc.
This function can only be called by a single thread at time.

Parameters

in	siop	pointer to an SIODriver structure
in	timeout	synchronization timeout

Returns

The synchronization result.

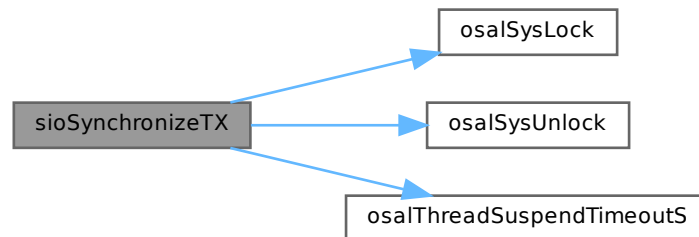
Return values

<i>MSG_OK</i>	if there is space in the TX FIFO.
<i>MSG_TIMEOUT</i>	if synchronization timed out.
<i>MSG_RESET</i>	operation has been stopped while waiting.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.18.6.27 sioSynchronizeTXEnd()**

```

msg_t sioSynchronizeTXEnd (
    SIODriver * siop,
    sysinterval_t timeout)
  
```

Synchronizes with TX completion.

Note

This function can only be called by a single thread at time.

Parameters

in	<i>siop</i>	pointer to an <code>SIODriver</code> structure
in	<i>timeout</i>	synchronization timeout

Returns

The synchronization result.

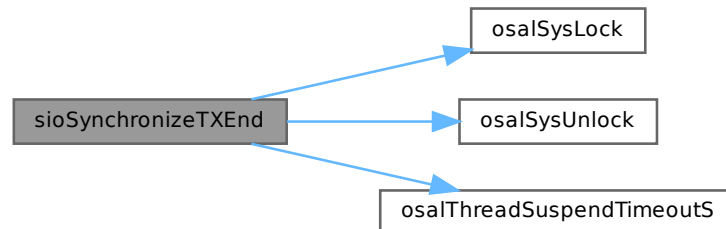
Return values

<code>MSG_OK</code>	if TX operation finished.
<code>MSG_TIMEOUT</code>	if synchronization timed out.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.18.6.28 sio_lld_init()**

```
void sio_lld_init (  
    void )
```

Low level SIO driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.18.6.29 sio_lld_start()**

```
msg_t sio_lld_start (  
    SIODriver * siop)
```

Configures and activates the SIO peripheral.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.6.30 sio_llc_stop()

```
void sio_llc_stop (  
    SIODriver * siop)
```

Deactivates the SIO peripheral.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.6.31 sio_llc_update_enable_flags()

```
void sio_llc_update_enable_flags (  
    SIODriver * siop)
```

Enable flags change notification.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

7.2.9.18.6.32 sio_llc_get_and_clear_errors()

```
sioevents\_t sio_llc_get_and_clear_errors (  
    SIODriver * siop)
```

Get and clears SIO error event flags.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The pending event flags.

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.6.33 [sio_lld_get_and_clear_events\(\)](#)

```
sioevents_t sio_lld_get_and_clear_events (  
    SIODriver * siop)
```

Get and clears SIO event flags.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The pending event flags.

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.6.34 [sio_lld_get_events\(\)](#)

```
sioevents_t sio_lld_get_events (  
    SIODriver * siop)
```

Returns the pending SIO event flags.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The pending event flags.

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.6.35 [sio_lld_read\(\)](#)

```
size_t sio_lld_read (  
    SIODriver * siop,  
    uint8_t * buffer,  
    size_t n)
```

Reads data from the RX FIFO.

The function is not blocking, it writes frames until there is space available without waiting.

Parameters

in	<i>siop</i>	pointer to an SIODriver structure
in	<i>buffer</i>	pointer to the buffer for read frames
in	<i>n</i>	maximum number of frames to be read

Returns

The number of frames copied from the buffer.

Return values

0	if the TX FIFO is full.

7.2.9.18.6.36 sio_lld_write()

```
size_t sio_lld_write (  
    SIODriver * siop,  
    const uint8_t * buffer,  
    size_t n)
```

Writes data into the TX FIFO.

The function is not blocking, it writes frames until there is space available without waiting.

Parameters

in	<i>siop</i>	pointer to an SIODriver structure
in	<i>buffer</i>	pointer to the buffer for read frames
in	<i>n</i>	maximum number of frames to be written

Returns

The number of frames copied from the buffer.

Return values

0	if the TX FIFO is full.

7.2.9.18.6.37 sio_lld_get()

```
msg_t sio_lld_get (  
    SIODriver * siop)
```

Returns one frame from the RX FIFO.

Note

If the FIFO is empty then the returned value is unpredictable.

Parameters

in	<i>siop</i>	pointer to the SIODriver object

Returns

The frame from RX FIFO.

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.6.38 sio_lld_put()

```
void sio_lld_put (  
    SIODriver * siop,  
    uint_fast16_t data)
```

Pushes one frame into the TX FIFO.

Note

If the FIFO is full then the behavior is unpredictable.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>data</i>	frame to be written

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.6.39 sio_lld_control()

```
msg_t sio_lld_control (  
    SIODriver * siop,  
    unsigned int operation,  
    void * arg)
```

Control operation on a serial port.

Parameters

in	<i>siop</i>	pointer to the SIODriver object
in	<i>operation</i>	control operation code
in, out	<i>arg</i>	operation argument

Returns

The control operation status.

Return values

<i>MSG_OK</i>	in case of success.
<i>MSG_TIMEOUT</i>	in case of operation timeout.
<i>MSG_RESET</i>	in case of operation reset.

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.6.40 sio_lld_serve_interrupt()

```
void sio_lld_serve_interrupt (  
    SIODriver * siop)
```

Serves an UART interrupt.

Parameters

in	<i>siop</i>	pointer to the <i>SIODriver</i> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.18.7 Variable Documentation**7.2.9.18.7.1 vmt**

```
const struct sio_driver_vmt vmt [static]
```

Initial value:

```
= {  
    (size_t)0,  
    __write, __read, __put, __get,  
    __putt, __gett, __writet, __readt,  
    __ctl  
}
```

7.2.9.18.7.2 SIOD1

```
SIODriver SIOD1
```

SIO1 driver identifier.

7.2.9.19 SPI Driver Selector

SPI Driver selector.

SPI Driver selector.

This module is a selector between SPI drivers version 1 and version 2.

7.2.9.20 SPIv1 Driver

Generic SPI Driver (legacy).

7.2.9.20.1 Detailed Description

Generic SPI Driver (legacy).

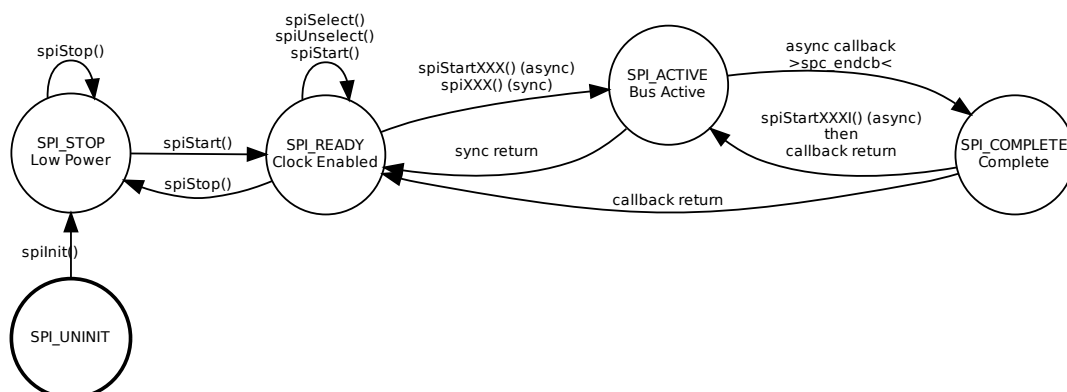
This module implements a generic SPI (Serial Peripheral Interface) driver allowing bidirectional and monodirectional transfers, complex atomic transactions are supported as well.

Precondition

In order to use the SPI driver the `HAL_USE_SPI` option must be enabled in `halconf.h`.

7.2.9.20.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



The driver is not thread safe for performance reasons, if you need to access the SPI bus from multiple threads then use the `spiAcquireBus()` and `spiReleaseBus()` APIs in order to gain exclusive access.

Chip Select modes

- `#define SPI_SELECT_MODE_NONE`
- `#define SPI_SELECT_MODE_PAD 1 /** @brief Legacy mode. */`
- `#define SPI_SELECT_MODE_PORT 2 /** @brief Fastest mode. */`
- `#define SPI_SELECT_MODE_LINE 3 /** @brief Packed mode. */`
- `#define SPI_SELECT_MODE_LLD 4 /** @brief LLD-defined mode.*/`

SPI configuration options

- `#define SPI_USE_WAIT TRUE`
Enables synchronous APIs.
- `#define SPI_USE_MUTUAL_EXCLUSION TRUE`
Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.
- `#define SPI_SELECT_MODE_SPI_SELECT_MODE_PAD`
Handling method for SPI CS line.

Macro Functions

- `#define spilsBufferComplete(spi)`
Buffer state.
- `#define spiSelectl(spi)`
Asserts the slave select signal and prepares for transfers.
- `#define spiUnselectl(spi)`
Deasserts the slave select signal.
- `#define spiStartIgnorel(spi, n)`
Ignores data on the SPI bus.
- `#define spiStartExchangel(spi, n, txbuf, rxbuf)`
Exchanges data on the SPI bus.
- `#define spiStartSendl(spi, n, txbuf)`
Sends data over the SPI bus.
- `#define spiStartReceiveI(spi, n, rxbuf)`
Receives data from the SPI bus.
- `#define spiPolledExchange(spi, frame)`
Exchanges one frame using a polled wait.

Low level driver helper macros

- `#define _spi_wakeup_isr(spi)`
Wakes up the waiting thread.
- `#define _spi_isr_code(spi)`
Common ISR code when circular mode is not supported.
- `#define _spi_isr_half_code(spi)`
Half buffer filled ISR code in circular mode.
- `#define _spi_isr_full_code(spi)`
Full buffer filled ISR code in circular mode.

PLATFORM configuration options

- `#define PLATFORM_SPI_USE_SPI1 FALSE`
SPI1 driver enable switch.

Data Structures

- struct `hal_spi_config`
Driver configuration structure.
- struct `hal_spi_driver`
Structure representing an SPI driver.

Macros

- `#define SPI_SUPPORTS_CIRCULAR TRUE`
Circular mode support flag.
- `#define spi_lld_driver_fields`
Low level fields of the SPI driver structure.
- `#define spi_lld_config_fields`
Low level fields of the SPI configuration structure.

Typedefs

- `typedef struct hal_spi_driver SPIDriver`
Type of a structure representing an SPI driver.
- `typedef struct hal_spi_config SPIConfig`
Type of a SPI driver configuration structure.
- `typedef void(* spicallback_t) (SPIDriver *spip)`
SPI notification callback type.

Enumerations

- `enum spistate_t {`
`SPI_UNINIT = 0 , SPI_STOP = 1 , SPI_READY = 2 , SPI_ACTIVE = 3 ,`
`SPI_COMPLETE = 4 }`
Driver state machine possible states.

Functions

- void `spiInit` (void)
SPI Driver initialization.
- void `spiObjectInit` (SPIDriver *spip)
Initializes the standard part of a `SPIDriver` structure.
- `msg_t spiStart` (SPIDriver *spip, const SPIConfig *config)
Configures and activates the SPI peripheral.
- void `spiStop` (SPIDriver *spip)
Deactivates the SPI peripheral.
- void `spiSelect` (SPIDriver *spip)
Asserts the slave select signal and prepares for transfers.
- void `spiUnselect` (SPIDriver *spip)
Deasserts the slave select signal.
- void `spiStartIgnore` (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- void `spiStartExchange` (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- void `spiStartSend` (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- void `spiStartReceive` (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- void `spiAbortI` (SPIDriver *spip)
Aborts the ongoing SPI operation.
- void `spiAbort` (SPIDriver *spip)
Aborts the ongoing SPI operation, if any.
- void `spiIgnore` (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- void `spiExchange` (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- void `spiSend` (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- void `spiReceive` (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- void `spiAcquireBus` (SPIDriver *spip)
Gains exclusive access to the SPI bus.
- void `spiReleaseBus` (SPIDriver *spip)
Releases exclusive access to the SPI bus.
- void `spi_ll_init` (void)
Low level SPI driver initialization.
- void `spi_ll_start` (SPIDriver *spip)
Configures and activates the SPI peripheral.
- void `spi_ll_stop` (SPIDriver *spip)
Deactivates the SPI peripheral.
- void `spi_ll_select` (SPIDriver *spip)
Asserts the slave select signal and prepares for transfers.
- void `spi_ll_unselect` (SPIDriver *spip)
Deasserts the slave select signal.
- void `spi_ll_ignore` (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- void `spi_ll_exchange` (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)

Exchanges data on the SPI bus.

- void `spi_llc_send` (`SPIDriver` *spip, size_t n, const void *txbuf)

Sends data over the SPI bus.

- void `spi_llc_receive` (`SPIDriver` *spip, size_t n, void *rxbuf)

Receives data from the SPI bus.

- void `spi_llc_abort` (`SPIDriver` *spip)

Aborts the ongoing SPI operation, if any.

- uint16_t `spi_llc_polled_exchange` (`SPIDriver` *spip, uint16_t frame)

Exchanges one frame using a polled wait.

Variables

- `SPIDriver SPID1`

SPI1 driver identifier.

7.2.9.20.3 Macro Definition Documentation

7.2.9.20.3.1 SPI_SELECT_MODE_NONE

```
#define SPI_SELECT_MODE_NONE
```

Value:

```
0    /** @brief @p spiSelect() and
    @p spiUnselect() do
    nothing.                */
```

7.2.9.20.3.2 SPI_SELECT_MODE_PAD

```
#define SPI_SELECT_MODE_PAD 1 /** @brief Legacy mode. */
```

7.2.9.20.3.3 SPI_SELECT_MODE_PORT

```
#define SPI_SELECT_MODE_PORT 2 /** @brief Fastest mode. */
```

7.2.9.20.3.4 SPI_SELECT_MODE_LINE

```
#define SPI_SELECT_MODE_LINE 3 /** @brief Packed mode. */
```

7.2.9.20.3.5 SPI_SELECT_MODE_LLD

```
#define SPI_SELECT_MODE_LLD 4 /** @brief LLD-defined mode.*/
```

7.2.9.20.3.6 SPI_USE_WAIT

```
#define SPI_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.9.20.3.7 SPI_USE_MUTUAL_EXCLUSION

```
#define SPI_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.9.20.3.8 SPI_SELECT_MODE

```
#define SPI_SELECT_MODE SPI_SELECT_MODE_PAD
```

Handling method for SPI CS line.

Note

Disabling this option saves both code and data space.

7.2.9.20.3.9 spiIsBufferComplete

```
#define spiIsBufferComplete(  
    spip)
```

Value:

```
((bool)((spip)->state == SPI_COMPLETE))
```

Buffer state.

Note

This function is meant to be called from the SPI callback only.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Returns

The buffer state.

Return values

<i>false</i>	if the driver filled/sent the first half of the buffer.
<i>true</i>	if the driver filled/sent the second half of the buffer.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.20.3.10 spiSelectI

```
#define spiSelectI(  
    spip)
```

Value:

```
do {  
    spi_llc_select(spip);  
} while (false)
```

\
\

Asserts the slave select signal and prepares for transfers.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.20.3.11 spiUnselectI

```
#define spiUnselectI(  
    spip)
```

Value:

```
do {  
    spi_llc_unselect(spip);  
} while (false)
```

\
\

Deasserts the slave select signal.

The previously selected peripheral is unselected.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.20.3.12 spiStartIgnoreI

```
#define spiStartIgnoreI(
    spip,
    n)
```

Value:

```
{
    (spip)->state = SPI_ACTIVE;
    spi_llc_ignore(spip, n);
}
```

Ignores data on the SPI bus.

This asynchronous function starts the transmission of a series of idle words on the SPI bus and ignores the received data.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to be ignored

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.20.3.13 spiStartExchangel

```
#define spiStartExchangeI(
    spip,
    n,
    txbuf,
    rxbuf)
```

Value:

```
{
    (spip)->state = SPI_ACTIVE;
    spi_llc_exchange(spip, n, txbuf, rxbuf);
}
```

Exchanges data on the SPI bus.

This asynchronous function starts a simultaneous transmit/receive operation.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to be exchanged
in	<i>txbuf</i>	the pointer to the transmit buffer
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.20.3.14 spiStartSendI

```
#define spiStartSendI(
    spip,
    n,
    txbuf)
```

Value:

```
{
    (spip)->state = SPI_ACTIVE;
    spi_llc_send(spip, n, txbuf);
}
```

Sends data over the SPI bus.

This asynchronous function starts a transmit operation.

Precondition

A slave must have been selected using [spiSelect\(\)](#) or [spiSelectI\(\)](#).

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.20.3.15 spiStartReceiveI

```
#define spiStartReceiveI(
    spip,
    n,
    rxbuf)
```

Value:

```
{
    (spip)->state = SPI_ACTIVE; \
    spi_llt_receive(spip, n, rxbuf); \
}
```

Receives data from the SPI bus.

This asynchronous function starts a receive operation.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to receive
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.20.3.16 spiPolledExchange

```
#define spiPolledExchange(
    spip,
    frame)
```

Value:

```
spi_llt_polled_exchange(spip, frame)
```

Exchanges one frame using a polled wait.

This synchronous function exchanges one frame using a polled synchronization method. This function is useful when exchanging small amount of data on high speed channels, usually in this situation is much more efficient just wait for completion using polling than suspending the thread waiting for an interrupt.

Note

This API is implemented as a macro in order to minimize latency.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>frame</i>	the data frame to send over the SPI bus

Returns

The received data frame from the SPI bus.

7.2.9.20.3.17 _spi_wakeup_isr

```
#define _spi_wakeup_isr(  
    spip)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI(&(spip)->thread, MSG_OK);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.3.18 _spi_isr_code

```
#define _spi_isr_code(  
    spip)
```

Value:

```
{  
    if ((spip)->config->end_cb) {  
        (spip)->state = SPI_COMPLETE;  
        (spip)->config->end_cb(spip);  
        if ((spip)->state == SPI_COMPLETE)  
            (spip)->state = SPI_READY;  
    }  
    else  
        (spip)->state = SPI_READY;  
    _spi_wakeup_isr(spip);  
}
```

Common ISR code when circular mode is not supported.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.3.19 _spi_isr_half_code

```
#define _spi_isr_half_code(  
    spip)
```

Value:

```
{  
    if ((spip)->config->end_cb) {  
        (spip)->config->end_cb(spip);  
    }  
}
```

Half buffer filled ISR code in circular mode.

This code handles the portable part of the ISR code:

- Callback invocation.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.3.20 _spi_isr_full_code

```
#define _spi_isr_full_code(  
    spip)
```

Value:

```
{  
    if ((spip)->config->end_cb) {  
        (spip)->state = SPI_COMPLETE;  
        (spip)->config->end_cb(spip);  
        if ((spip)->state == SPI_COMPLETE)  
            (spip)->state = SPI_ACTIVE;  
    }  
}
```

Full buffer filled ISR code in circular mode.

This code handles the portable part of the ISR code:

- Callback invocation.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.3.21 SPI_SUPPORTS_CIRCULAR

```
#define SPI_SUPPORTS_CIRCULAR TRUE
```

Circular mode support flag.

7.2.9.20.3.22 PLATFORM_SPI_USE_SPI1

```
#define PLATFORM_SPI_USE_SPI1 FALSE
```

SPI1 driver enable switch.

If set to `TRUE` the support for SPI1 is included.

Note

The default is `FALSE`.

7.2.9.20.3.23 spi_llc_driver_fields

```
#define spi_llc_driver_fields
```

Value:

```
/* Dummy field, it is not needed.*/
uint32_t dummy
```

Low level fields of the SPI driver structure.

7.2.9.20.3.24 spi_llc_config_fields

```
#define spi_llc_config_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/
uint32_t dummy
```

Low level fields of the SPI configuration structure.

7.2.9.20.4 Typedef Documentation

7.2.9.20.4.1 SPIDriver

```
typedef struct hal_spi_driver SPIDriver
```

Type of a structure representing an SPI driver.

7.2.9.20.4.2 SPIConfig

```
typedef struct hal_spi_config SPIConfig
```

Type of a SPI driver configuration structure.

7.2.9.20.4.3 spicallback_t

```
typedef void(* spicallback_t) (SPIDriver *spip)
```

SPI notification callback type.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object triggering the callback

7.2.9.20.5 Enumeration Type Documentation

7.2.9.20.5.1 spistate_t

```
enum spistate_t
```

Driver state machine possible states.

Enumerator

SPI_UNINIT	Not initialized.
SPI_STOP	Stopped.
SPI_READY	Ready.
SPI_ACTIVE	Exchanging data.
SPI_COMPLETE	Asynchronous operation complete.

7.2.9.20.6 Function Documentation

7.2.9.20.6.1 spiInit()

```
void spiInit (  
    void )
```

SPI Driver initialization.

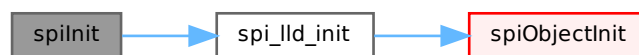
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.20.6.2 spiObjectInit()

```
void spiObjectInit (  
    SPIDriver * spip)
```

Initializes the standard part of a `SPIDriver` structure.

Parameters

out	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.20.6.3 spiStart()

```
msg_t spiStart (  
    SPIDriver * spip,  
    const SPIConfig * config)
```

Configures and activates the SPI peripheral.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>config</i>	pointer to the SPIConfig object

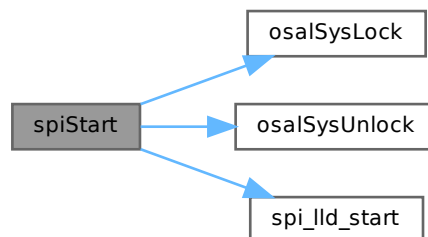
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.20.6.4 spiStop()

```
void spiStop (  
    SPIDriver * spip)
```

Deactivates the SPI peripheral.

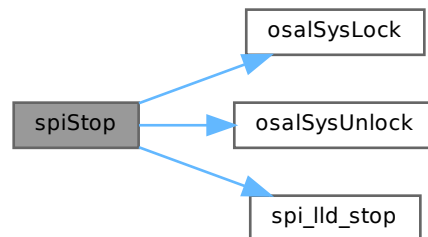
Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.20.6.5 spiSelect()**

```
void spiSelect (  
    SPIDriver * spip)
```

Asserts the slave select signal and prepares for transfers.

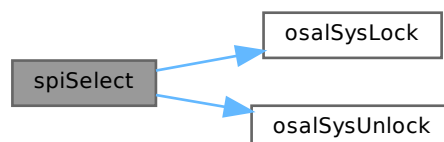
Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.20.6.6 spiUnselect()

```
void spiUnselect (  
    SPIDriver * spip)
```

Deasserts the slave select signal.

The previously selected peripheral is unselected.

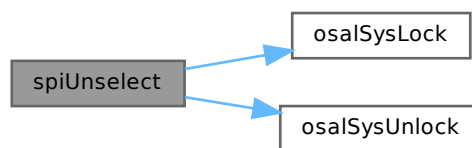
Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.20.6.7 spiStartIgnore()

```
void spiStartIgnore (  
    SPIDriver * spip,  
    size_t n)
```

Ignores data on the SPI bus.

This asynchronous function starts the transmission of a series of idle words on the SPI bus and ignores the received data.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

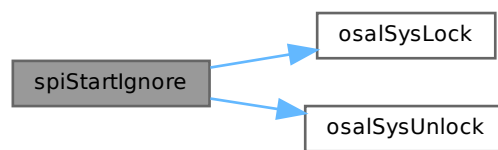
Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to be ignored

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.20.6.8 spiStartExchange()**

```
void spiStartExchange (  
    SPIDriver * spip,  
    size_t n,  
    const void * txbuf,  
    void * rxbuf)
```

Exchanges data on the SPI bus.

This asynchronous function starts a simultaneous transmit/receive operation.

Precondition

A slave must have been selected using [spiSelect\(\)](#) or [spiSelectI\(\)](#).

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

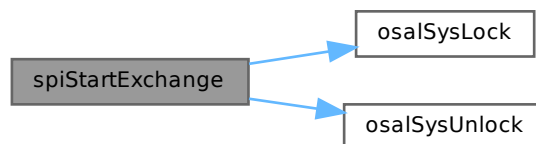
Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to be exchanged
in	<i>txbuf</i>	the pointer to the transmit buffer
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.20.6.9 spiStartSend()**

```
void spiStartSend (  
    SPIDriver * spip,  
    size_t n,  
    const void * txbuf)
```

Sends data over the SPI bus.

This asynchronous function starts a transmit operation.

Precondition

A slave must have been selected using [spiSelect\(\)](#) or [spiSelectI\(\)](#).

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

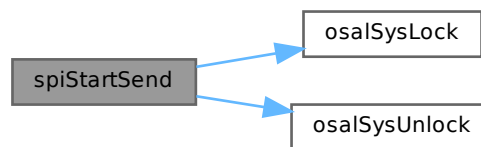
Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.20.6.10 spiStartReceive()**

```

void spiStartReceive (
    SPIDriver * spip,
    size_t n,
    void * rxbuf)
  
```

Receives data from the SPI bus.

This asynchronous function starts a receive operation.

Precondition

A slave must have been selected using [spiSelect\(\)](#) or [spiSelectI\(\)](#).

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

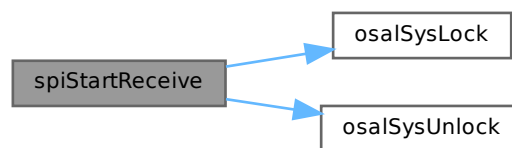
Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to receive
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.20.6.11 spiAbortI()**

```
void spiAbortI (  
    SPIDriver * spip)
```

Aborts the ongoing SPI operation.

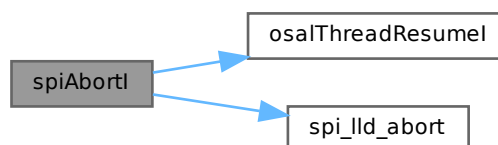
Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.20.6.12 spiAbort()

```
void spiAbort (  
    SPIDriver * spip)
```

Aborts the ongoing SPI operation, if any.

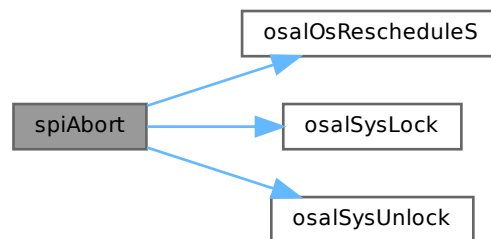
Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.20.6.13 spilgnore()

```
void spilgnore (  
    SPIDriver * spip,  
    size_t n)
```

Ignores data on the SPI bus.

This synchronous function performs the transmission of a series of idle words on the SPI bus and ignores the received data.

Precondition

In order to use this function the option `SPI_USE_WAIT` must be enabled.

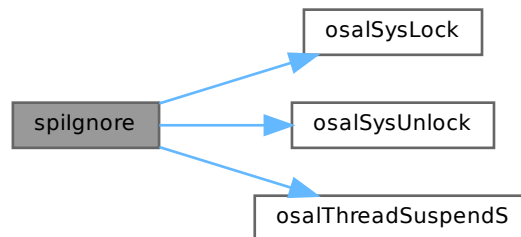
Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to be ignored

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.20.6.14 spiExchange()**

```

void spiExchange (
    SPIDriver * spip,
    size_t n,
    const void * txbuf,
    void * rxbuf)
  
```

Exchanges data on the SPI bus.

This synchronous function performs a simultaneous transmit/receive operation.

Precondition

In order to use this function the option `SPI_USE_WAIT` must be enabled.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

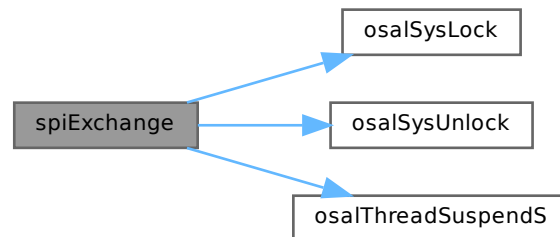
Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to be exchanged
in	<i>txbuf</i>	the pointer to the transmit buffer
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.20.6.15 spiSend()**

```

void spiSend (
    SPIDriver * spip,
    size_t n,
    const void * txbuf)
  
```

Sends data over the SPI bus.

This synchronous function performs a transmit operation.

Precondition

In order to use this function the option `SPI_USE_WAIT` must be enabled.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

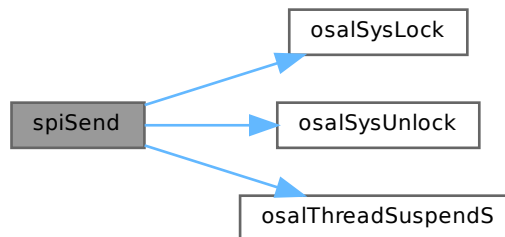
Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.20.6.16 spiReceive()**

```

void spiReceive (
    SPIDriver * spip,
    size_t n,
    void * rxbuf)
  
```

Receives data from the SPI bus.

This synchronous function performs a receive operation.

Precondition

In order to use this function the option `SPI_USE_WAIT` must be enabled.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

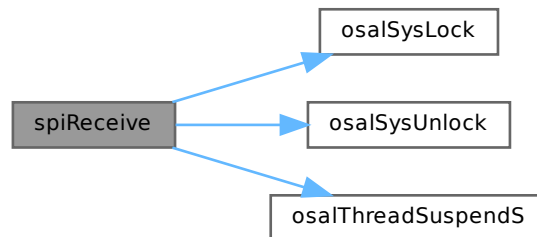
Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to receive
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.20.6.17 spiAcquireBus()**

```
void spiAcquireBus (
    SPIDriver * spip)
```

Gains exclusive access to the SPI bus.

This function tries to gain ownership to the SPI bus, if the bus is already being used then the invoking thread is queued.

Precondition

In order to use this function the option `SPI_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.20.6.18 spiReleaseBus()

```
void spiReleaseBus (  
    SPIDriver * spip)
```

Releases exclusive access to the SPI bus.

Precondition

In order to use this function the option `SPI_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.20.6.19 spi_lld_init()

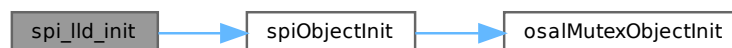
```
void spi_lld_init (  
    void )
```

Low level SPI driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.20.6.20 spi_lld_start()

```
void spi_lld_start (  
    SPIDriver * spip)
```

Configures and activates the SPI peripheral.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.6.21 spi_llc_stop()

```
void spi_llc_stop (  
    SPIDriver * spip)
```

Deactivates the SPI peripheral.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.6.22 spi_llc_select()

```
void spi_llc_select (  
    SPIDriver * spip)
```

Asserts the slave select signal and prepares for transfers.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.6.23 spi_llc_unselect()

```
void spi_llc_unselect (  
    SPIDriver * spip)
```

Deasserts the slave select signal.

The previously selected peripheral is unselected.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.6.24 spi_llc_ignore()

```
void spi_llc_ignore (  
    SPIDriver * spip,  
    size_t n)
```

Ignores data on the SPI bus.

This asynchronous function starts the transmission of a series of idle words on the SPI bus and ignores the received data.

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to be ignored

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.6.25 spi_llc_exchange()

```
void spi_llc_exchange (  
    SPIDriver * spip,  
    size_t n,  
    const void * txbuf,  
    void * rxbuf)
```

Exchanges data on the SPI bus.

This asynchronous function starts a simultaneous transmit/receive operation.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to be exchanged
in	<i>txbuf</i>	the pointer to the transmit buffer
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.6.26 spi_llc_send()

```
void spi_llc_send (  
    SPIDriver * spip,  
    size_t n,  
    const void * txbuf)
```

Sends data over the SPI bus.

This asynchronous function starts a transmit operation.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as uint8_t arrays for data sizes below or equal to 8 bits else it is organized as uint16_t arrays.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.6.27 spi_llc_receive()

```
void spi_llc_receive (  
    SPIDriver * spip,  
    size_t n,  
    void * rxbuf)
```

Receives data from the SPI bus.

This asynchronous function starts a receive operation.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as uint8_t arrays for data sizes below or equal to 8 bits else it is organized as uint16_t arrays.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to receive
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.6.28 spi_llc_abort()

```
void spi_llc_abort (  
    SPIDriver * spip)
```

Aborts the ongoing SPI operation, if any.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.6.29 spi_llc_polled_exchange()

```
uint16_t spi_llc_polled_exchange (  
    SPIDriver * spip,  
    uint16_t frame)
```

Exchanges one frame using a polled wait.

This synchronous function exchanges one frame using a polled synchronization method. This function is useful when exchanging small amount of data on high speed channels, usually in this situation is much more efficient just wait for completion using polling than suspending the thread waiting for an interrupt.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>frame</i>	the data frame to send over the SPI bus

Returns

The received data frame from the SPI bus.

Function Class:

Not an API, this function is for internal use only.

7.2.9.20.7 Variable Documentation

7.2.9.20.7.1 SPID1

`SPIDriver` SPID1

SPI1 driver identifier.

7.2.9.21 SPIv2 Driver

Generic SPI Driver.

7.2.9.21.1 Detailed Description

Generic SPI Driver.

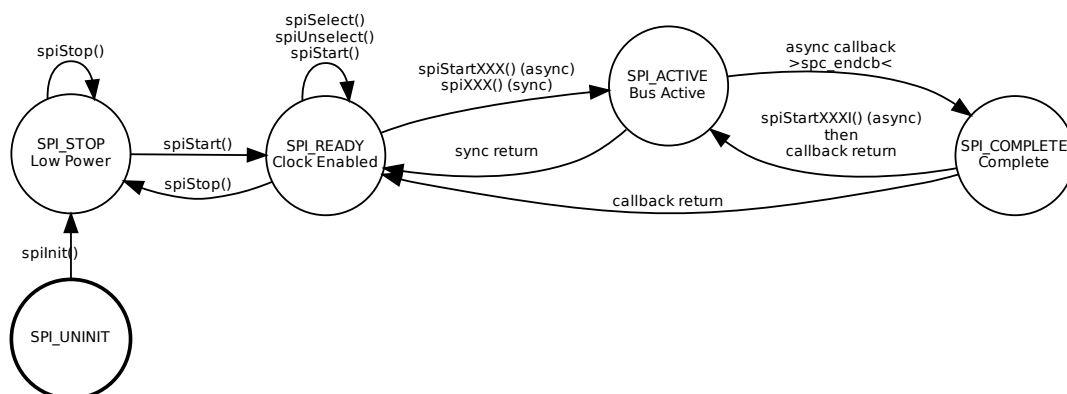
This module implements a generic SPI (Serial Peripheral Interface) driver allowing bidirectional and monodirectional transfers, complex atomic transactions are supported as well.

Precondition

In order to use the SPI driver the `HAL_USE_SPI` option must be enabled in `halconf.h`.

7.2.9.21.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



The driver is not thread safe for performance reasons, if you need to access the SPI bus from multiple threads then use the `spiAcquireBus()` and `spiReleaseBus()` APIs in order to gain exclusive access.

Chip Select modes

- `#define SPI_SELECT_MODE_NONE`
- `#define SPI_SELECT_MODE_PAD 1 /** @brief Legacy mode. */`
- `#define SPI_SELECT_MODE_PORT 2 /** @brief Fastest mode. */`
- `#define SPI_SELECT_MODE_LINE 3 /** @brief Packed mode. */`
- `#define SPI_SELECT_MODE_LLD 4 /** @brief LLD-defined mode.*/`

SPI configuration options

- `#define SPI_USE_SYNCHRONIZATION FALSE`
Support for thread synchronization API.
- `#define SPI_USE_ASSERT_ON_ERROR TRUE`
Inserts an assertion on function errors before returning.
- `#define SPI_USE_MUTUAL_EXCLUSION TRUE`
Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.
- `#define SPI_SELECT_MODE SPI_SELECT_MODE_PAD`
Handling method for SPI CS line.

Macro Functions

- `#define spilsBufferComplete(spip)`
Buffer state.
- `#define spiSelectl(spip)`
Asserts the slave select signal and prepares for transfers.
- `#define spiUnselectl(spip)`
Deasserts the slave select signal.
- `#define spiPolledExchange(spip, frame)`
Exchanges one frame using a polled wait.
- `#define spiAbortl(spip)`
Compatibility API with SPI driver v1.
- `#define spiAbort(spip)`
Compatibility API with SPI driver v1.

Low level driver helper macros

- `#define __spi_wakeup_isr(spip, msg)`
Wakes up the waiting thread.
- `#define __spi_isr_complete_code(spip)`
Common ISR code in linear mode.
- `#define __spi_isr_half_code(spip)`
Half buffer filled ISR code in circular mode.
- `#define __spi_isr_full_code(spip)`
Full buffer filled ISR code in circular mode.
- `#define __spi_isr_error_code(spip, msg)`
ISR error reporting code..

PLATFORM configuration options

- #define `PLATFORM_SPI_USE_SPI1` `FALSE`
SPI1 driver enable switch.

Data Structures

- struct `hal_spi_config`
Driver configuration structure.
- struct `hal_spi_driver`
Structure representing an SPI driver.

Macros

- #define `SPI_SUPPORTS_CIRCULAR` `TRUE`
Circular mode support flag.
- #define `SPI_SUPPORTS_SLAVE_MODE` `TRUE`
Slave mode support flag.
- #define `spi_lld_driver_fields`
Low level fields of the SPI driver structure.
- #define `spi_lld_config_fields`
Low level fields of the SPI configuration structure.

Typedefs

- typedef struct `hal_spi_driver` `SPIDriver`
Type of a structure representing an SPI driver.
- typedef struct `hal_spi_config` `SPISConfig`
Type of a SPI driver configuration structure.
- typedef void(* `spicb_t`) (`SPIDriver` *spip)
SPI notification callback type.

Enumerations

- enum `spistate_t` {
 `SPI_UNINIT` = 0 , `SPI_STOP` = 1 , `SPI_READY` = 2 , `SPI_ACTIVE` = 3 ,
 `SPI_COMPLETE` = 4 }
Driver state machine possible states.

Functions

- void `spiInit` (void)
SPI Driver initialization.
- void `spiObjectInit` (SPIDriver *spip)
Initializes the standard part of a `SPIDriver` structure.
- `msg_t spiStart` (SPIDriver *spip, const SPIConfig *config)
Configures and activates the SPI peripheral.
- void `spiStop` (SPIDriver *spip)
Deactivates the SPI peripheral.
- void `spiSelect` (SPIDriver *spip)
Asserts the slave select signal and prepares for transfers.
- void `spiUnselect` (SPIDriver *spip)
Deasserts the slave select signal.
- `msg_t spiStartIgnoreI` (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- `msg_t spiStartIgnore` (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- `msg_t spiStartExchangeI` (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- `msg_t spiStartExchange` (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- `msg_t spiStartSendI` (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- `msg_t spiStartSend` (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- `msg_t spiStartReceiveI` (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- `msg_t spiStartReceive` (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- `msg_t spiStopTransferI` (SPIDriver *spip, size_t *sizep)
Stops the ongoing SPI operation.
- `msg_t spiStopTransfer` (SPIDriver *spip, size_t *sizep)
Stops the ongoing SPI operation, if any.
- `msg_t spiSynchronizeS` (SPIDriver *spip, sysinterval_t timeout)
Synchronizes with current transfer completion.
- `msg_t spiSynchronize` (SPIDriver *spip, sysinterval_t timeout)
Synchronizes with current transfer completion.
- `msg_t spiIgnore` (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- `msg_t spiExchange` (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- `msg_t spiSend` (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- `msg_t spiReceive` (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- void `spiAcquireBus` (SPIDriver *spip)
Gains exclusive access to the SPI bus.
- void `spiReleaseBus` (SPIDriver *spip)
Releases exclusive access to the SPI bus.
- void `spi_llc_init` (void)

Low level SPI driver initialization.

- `msg_t spi_ll_start (SPIDriver *spip)`
Configures and activates the SPI peripheral.
- `void spi_ll_stop (SPIDriver *spip)`
Deactivates the SPI peripheral.
- `void spi_ll_select (SPIDriver *spip)`
Asserts the slave select signal and prepares for transfers.
- `void spi_ll_unselect (SPIDriver *spip)`
Deasserts the slave select signal.
- `msg_t spi_ll_ignore (SPIDriver *spip, size_t n)`
Ignores data on the SPI bus.
- `msg_t spi_ll_exchange (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)`
Exchanges data on the SPI bus.
- `msg_t spi_ll_send (SPIDriver *spip, size_t n, const void *txbuf)`
Sends data over the SPI bus.
- `msg_t spi_ll_receive (SPIDriver *spip, size_t n, void *rxbuf)`
Receives data from the SPI bus.
- `msg_t spi_ll_stop_transfer (SPIDriver *spip, size_t *sizep)`
Aborts the ongoing SPI operation, if any.
- `uint16_t spi_ll_polled_exchange (SPIDriver *spip, uint16_t frame)`
Exchanges one frame using a polled wait.

Variables

- `SPIDriver SPID1`
SPI1 driver identifier.

7.2.9.21.3 Macro Definition Documentation

7.2.9.21.3.1 SPI_SELECT_MODE_NONE

```
#define SPI_SELECT_MODE_NONE
```

Value:

```
0 /** @brief @p spiSelect() and
    @p spiUnselect() do
    nothing. */
```

7.2.9.21.3.2 SPI_SELECT_MODE_PAD

```
#define SPI_SELECT_MODE_PAD 1 /** @brief Legacy mode. */
```

7.2.9.21.3.3 SPI_SELECT_MODE_PORT

```
#define SPI_SELECT_MODE_PORT 2 /** @brief Fastest mode. */
```

7.2.9.21.3.4 SPI_SELECT_MODE_LINE

```
#define SPI_SELECT_MODE_LINE 3 /** @brief Packed mode. */
```

7.2.9.21.3.5 SPI_SELECT_MODE_LLD

```
#define SPI_SELECT_MODE_LLD 4 /** @brief LLD-defined mode.*/
```

7.2.9.21.3.6 SPI_USE_SYNCHRONIZATION

```
#define SPI_USE_SYNCHRONIZATION FALSE
```

Support for thread synchronization API.

7.2.9.21.3.7 SPI_USE_ASSERT_ON_ERROR

```
#define SPI_USE_ASSERT_ON_ERROR TRUE
```

Inserts an assertion on function errors before returning.

7.2.9.21.3.8 SPI_USE_MUTUAL_EXCLUSION

```
#define SPI_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.9.21.3.9 SPI_SELECT_MODE

```
#define SPI_SELECT_MODE SPI_SELECT_MODE_PAD
```

Handling method for SPI CS line.

Note

Disabling this option saves both code and data space.

7.2.9.21.3.10 spiIsBufferComplete

```
#define spiIsBufferComplete(  
    spip)
```

Value:

```
((bool)((spip)->state == SPI_COMPLETE))
```

Buffer state.

Note

This function is meant to be called from the SPI callback only.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Returns

The buffer state.

Return values

<i>false</i>	if the driver filled/sent the first half of the buffer.
<i>true</i>	if the driver filled/sent the second half of the buffer.

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.21.3.11 spiSelectI

```
#define spiSelectI(
    spip)
```

Value:

```
do {
    spi_ll_d_select(spip);
} while (false)
```

\
\

Asserts the slave select signal and prepares for transfers.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.21.3.12 spiUnselectI

```
#define spiUnselectI(
    spip)
```

Value:

```
do {
    spi_ll_d_unselect(spip);
} while (false)
```

\
\

Deasserts the slave select signal.

The previously selected peripheral is unselected.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.21.3.13 spiPolledExchange

```
#define spiPolledExchange(  
    spip,  
    frame)
```

Value:

[spi_llc_polled_exchange](#)(spip, frame)

Exchanges one frame using a polled wait.

This synchronous function exchanges one frame using a polled synchronization method. This function is useful when exchanging small amount of data on high speed channels, usually in this situation is much more efficient just wait for completion using polling than suspending the thread waiting for an interrupt.

Note

This API is implemented as a macro in order to minimize latency.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>frame</i>	the data frame to send over the SPI bus

Returns

The received data frame from the SPI bus.

7.2.9.21.3.14 spiAbortI

```
#define spiAbortI(  
    spip)
```

Value:

[spiStopTransferI](#)(spip, NULL)

Compatibility API with SPI driver v1.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.21.3.15 spiAbort

```
#define spiAbort(  
    spip)
```

Value:

```
spiStopTransfer(spip, NULL)
```

Compatibility API with SPI driver v1.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.21.3.16 __spi_wakeup_isr

```
#define __spi_wakeup_isr(  
    spip,  
    msg)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI(&(spip)->sync_transfer, msg);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>msg</i>	wakeup message

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.3.17 __spi_isr_complete_code

```
#define __spi_isr_complete_code(  
    spip)
```

Value:

```
{  
    if ((spip)->config->data_cb) {  
        (spip)->state = SPI_COMPLETE;  
        (spip)->config->data_cb(spip);  
        if ((spip)->state == SPI_COMPLETE)  
            (spip)->state = SPI_READY;  
    }  
    else {  
        (spip)->state = SPI_READY;  
    }  
    __spi_wakeup_isr(spip, MSG_OK);  
}
```

Common ISR code in linear mode.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.3.18 __spi_isr_half_code

```
#define __spi_isr_half_code(  
    spip)
```

Value:

```
{  
    if ((spip)->config->data_cb) {  
        (spip)->config->data_cb(spip);  
    }  
}
```

Half buffer filled ISR code in circular mode.

This code handles the portable part of the ISR code:

- Callback invocation.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.3.19 __spi_isr_full_code

```
#define __spi_isr_full_code(
    spip)
```

Value:

```
{
    if ((spip)->config->data_cb) {
        (spip)->state = SPI_COMPLETE;
        (spip)->config->data_cb(spip);
        if ((spip)->state == SPI_COMPLETE) {
            (spip)->state = SPI_ACTIVE;
        }
    }
}
```

Full buffer filled ISR code in circular mode.

This code handles the portable part of the ISR code:

- Callback invocation.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.3.20 __spi_isr_error_code

```
#define __spi_isr_error_code(
    spip,
    msg)
```

Value:

```
{
    if ((spip)->config->error_cb) {
        (spip)->config->error_cb(spip);
    }
    __spi_wakeup_isr(spip, msg);
}
```

ISR error reporting code..

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>msg</i>	error code

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.3.21 SPI_SUPPORTS_CIRCULAR

```
#define SPI_SUPPORTS_CIRCULAR TRUE
```

Circular mode support flag.

7.2.9.21.3.22 SPI_SUPPORTS_SLAVE_MODE

```
#define SPI_SUPPORTS_SLAVE_MODE TRUE
```

Slave mode support flag.

7.2.9.21.3.23 PLATFORM_SPI_USE_SPI1

```
#define PLATFORM_SPI_USE_SPI1 FALSE
```

SPI1 driver enable switch.

If set to [TRUE](#) the support for SPI1 is included.

Note

The default is [FALSE](#).

7.2.9.21.3.24 spi_llc_driver_fields

```
#define spi_llc_driver_fields
```

Value:

```
/* Dummy field, it is not needed.*/  
uint32_t dummy;
```

Low level fields of the SPI driver structure.

7.2.9.21.3.25 spi_llc_config_fields

```
#define spi_llc_config_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/
uint32_t dummy;
```

Low level fields of the SPI configuration structure.

7.2.9.21.4 Typedef Documentation

7.2.9.21.4.1 SPIDriver

```
typedef struct hal_spi_driver SPIDriver
```

Type of a structure representing an SPI driver.

7.2.9.21.4.2 SPIConfig

```
typedef struct hal_spi_config SPIConfig
```

Type of a SPI driver configuration structure.

7.2.9.21.4.3 spicb_t

```
typedef void(* spicb_t) (SPIDriver *spip)
```

SPI notification callback type.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object triggering the callback

7.2.9.21.5 Enumeration Type Documentation

7.2.9.21.5.1 spistate_t

```
enum spistate_t
```

Driver state machine possible states.

Enumerator

SPI_UNINIT	Not initialized.
SPI_STOP	Stopped.
SPI_READY	Ready.
SPI_ACTIVE	Exchanging data.
SPI_COMPLETE	Asynchronous operation complete.

7.2.9.21.6 Function Documentation

7.2.9.21.6.1 spiInit()

```
void spiInit (  
    void )
```

SPI Driver initialization.

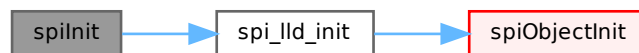
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.21.6.2 spiObjectInit()

```
void spiObjectInit (  
    SPIDriver * spip)
```

Initializes the standard part of a `SPIDriver` structure.

Parameters

out	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.21.6.3 spiStart()

```
msg_t spiStart (  
    SPIDriver * spip,  
    const SPIConfig * config)
```

Configures and activates the SPI peripheral.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>config</i>	pointer to the SPIConfig object

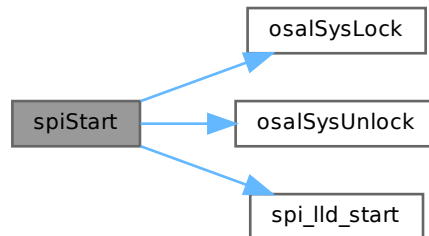
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.4 spiStop()

```
void spiStop (  
    SPIDriver * spip)
```

Deactivates the SPI peripheral.

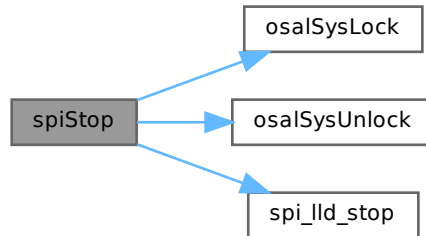
Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.21.6.5 spiSelect()**

```
void spiSelect (  
    SPIDriver * spip)
```

Asserts the slave select signal and prepares for transfers.

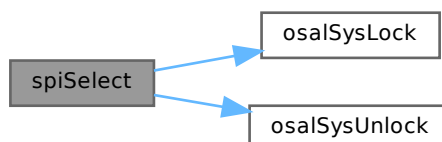
Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.6 spiUnselect()

```
void spiUnselect (  
    SPIDriver * spip)
```

Deasserts the slave select signal.

The previously selected peripheral is unselected.

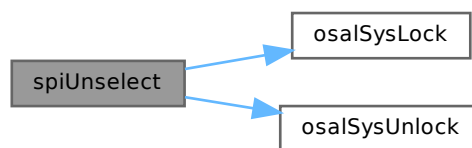
Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.7 spiStartIgnoreI()

```
msg_t spiStartIgnoreI (  
    SPIDriver * spip,  
    size_t n)
```

Ignores data on the SPI bus.

This asynchronous function starts the transmission of a series of idle words on the SPI bus and ignores the received data.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to be ignored

Returns

The operation status.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.21.6.8 spiStartIgnore()**

```
msg_t spiStartIgnore (  
    SPIDriver * spip,  
    size_t n)
```

Ignores data on the SPI bus.

This asynchronous function starts the transmission of a series of idle words on the SPI bus and ignores the received data.

Precondition

A slave must have been selected using [spiSelect\(\)](#) or [spiSelectI\(\)](#).

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to be ignored

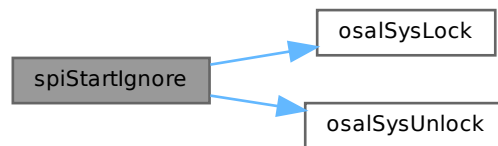
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.21.6.9 spiStartExchangeI()**

```

msg_t spiStartExchangeI (
    SPIDriver * spip,
    size_t n,
    const void * txbuf,
    void * rxbuf)
  
```

Exchanges data on the SPI bus.

This asynchronous function starts a simultaneous transmit/receive operation.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to be exchanged
in	<i>txbuf</i>	the pointer to the transmit buffer
out	<i>rxbuf</i>	the pointer to the receive buffer

Returns

The operation status.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.21.6.10 spiStartExchange()**

```
msg_t spiStartExchange (  
    SPIDriver * spip,  
    size_t n,  
    const void * txbuf,  
    void * rxbuf)
```

Exchanges data on the SPI bus.

This asynchronous function starts a simultaneous transmit/receive operation.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to be exchanged
in	<i>txbuf</i>	the pointer to the transmit buffer
out	<i>rxbuf</i>	the pointer to the receive buffer

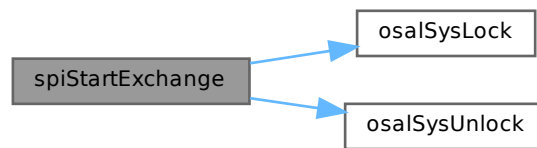
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.21.6.11 spiStartSendI()**

```
msg_t spiStartSendI (  
    SPIDriver * spip,  
    size_t n,  
    const void * txbuf)
```

Sends data over the SPI bus.

This asynchronous function starts a transmit operation.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Returns

The operation status.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.21.6.12 spiStartSend()**

```
msg_t spiStartSend (  
    SPIDriver * spip,  
    size_t n,  
    const void * txbuf)
```

Sends data over the SPI bus.

This asynchronous function starts a transmit operation.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to send
in	<i>txbuf</i>	the pointer to the transmit buffer

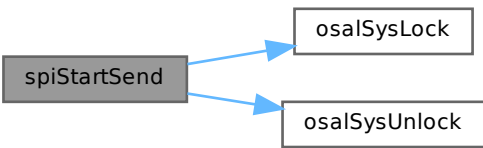
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.13 spiStartReceiveI()

```
msg_t spiStartReceiveI (  
    SPIDriver * spip,  
    size_t n,  
    void * rxbuf)
```

Receives data from the SPI bus.

This asynchronous function starts a receive operation.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to receive
out	<i>rxbuf</i>	the pointer to the receive buffer

Returns

The operation status.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.21.6.14 spiStartReceive()**

```
msg_t spiStartReceive (  
    SPIDriver * spip,  
    size_t n,  
    void * rxbuf)
```

Receives data from the SPI bus.

This asynchronous function starts a receive operation.

Precondition

A slave must have been selected using `spiSelect()` or `spiSelectI()`.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to receive
out	<i>rxbuf</i>	the pointer to the receive buffer

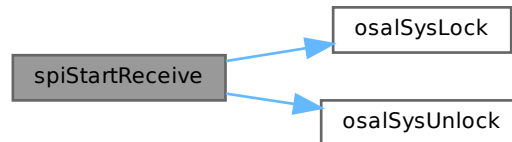
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.21.6.15 spiStopTransferI()**

```

msg_t spiStopTransferI (
    SPIDriver * spip,
    size_t * sizep)
  
```

Stops the ongoing SPI operation.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
out	<i>sizep</i>	pointer to the counter of frames not yet transferred or <code>NULL</code>

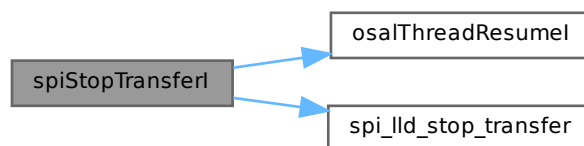
Returns

The operation status.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.21.6.16 spiStopTransfer()

```
msg_t spiStopTransfer (
    SPIDriver * spip,
    size_t * sizep)
```

Stops the ongoing SPI operation, if any.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
out	<i>sizep</i>	pointer to the counter of frames not yet transferred or <code>NULL</code>

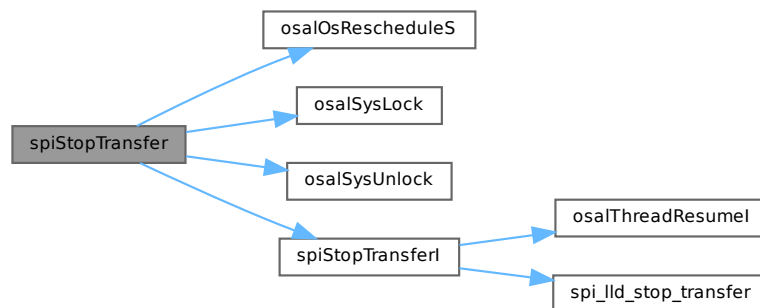
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.17 spiSynchronizeS()

```
msg_t spiSynchronizeS (
    SPIDriver * spip,
    sysinterval_t timeout)
```

Synchronizes with current transfer completion.

Note

This function can only be called by a single thread at time.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>timeout</i>	synchronization timeout

Returns

The synchronization result.

Return values

<i>MSG_OK</i>	if operation completed without errors.
<i>MSG_TIMEOUT</i>	if synchronization timed out.
<i>MSG_RESET</i>	if the transfer has been stopped.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

Here is the call graph for this function:



7.2.9.21.6.18 spiSynchronize()

```
msg_t spiSynchronize (  
    SPIDriver * spip,  
    sysinterval_t timeout)
```

Synchronizes with current transfer completion.

Note

This function can only be called by a single thread at time.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>timeout</i>	synchronization timeout

Returns

The synchronization result.

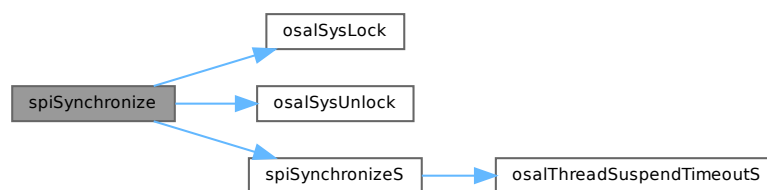
Return values

<i>MSG_OK</i>	if operation completed without errors.
<i>MSG_TIMEOUT</i>	if synchronization timed out.
<i>MSG_RESET</i>	if the transfer has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

7.2.9.21.6.19 `spiIgnore()`

```

msg_t spiIgnore (
    SPIDriver * spip,
    size_t n)

```

Ignores data on the SPI bus.

This synchronous function performs the transmission of a series of idle words on the SPI bus and ignores the received data.

Precondition

In order to use this function the option `SPI_USE_SYNCHRONIZATION` must be enabled.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to be ignored

Returns

The operation status.

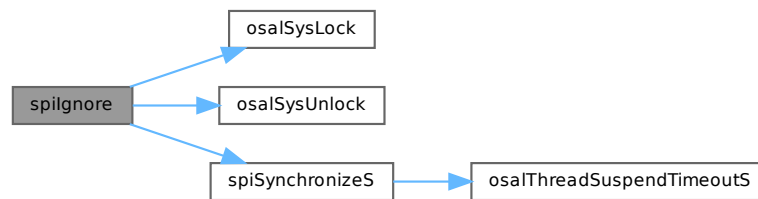
Return values

<i>MSG_OK</i>	if operation completed without errors.
<i>MSG_TIMEOUT</i>	if synchronization timed out.
<i>MSG_RESET</i>	if the transfer has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.20 spiExchange()

```

msg_t spiExchange (
    SPIDriver * spip,
    size_t n,
    const void * txbuf,
    void * rxbuf)
  
```

Exchanges data on the SPI bus.

This synchronous function performs a simultaneous transmit/receive operation.

Precondition

In order to use this function the option `SPI_USE_SYNCHRONIZATION` must be enabled.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to be exchanged
in	<i>txbuf</i>	the pointer to the transmit buffer
out	<i>rxbuf</i>	the pointer to the receive buffer

Returns

The operation status.

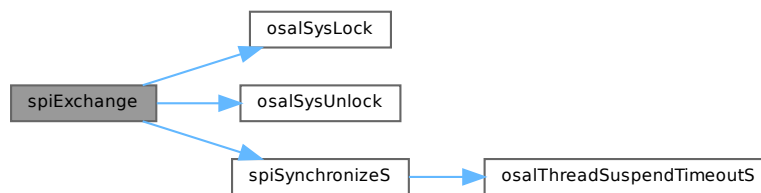
Return values

<i>MSG_OK</i>	if operation completed without errors.
<i>MSG_TIMEOUT</i>	if synchronization timed out.
<i>MSG_RESET</i>	if the transfer has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.21 spiSend()

```

msg_t spiSend (
    SPIDriver * spip,
    size_t n,
    const void * txbuf)

```

Sends data over the SPI bus.

This synchronous function performs a transmit operation.

Precondition

In order to use this function the option `SPI_USE_SYNCHRONIZATION` must be enabled.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Returns

The operation status.

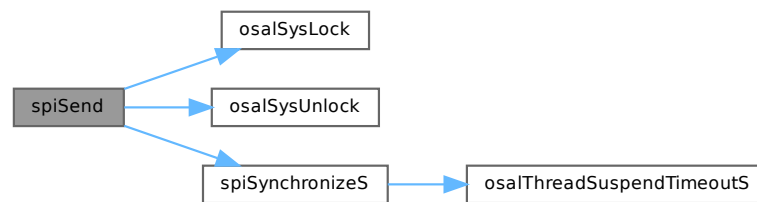
Return values

<i>MSG_OK</i>	if operation completed without errors.
<i>MSG_TIMEOUT</i>	if synchronization timed out.
<i>MSG_RESET</i>	if the transfer has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.22 spiReceive()

```

msg_t spiReceive (
    SPIDriver * spip,
    size_t n,
    void * rxbuf)

```

Receives data from the SPI bus.

This synchronous function performs a receive operation.

Precondition

In order to use this function the option `SPI_USE_SYNCHRONIZATION` must be enabled.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>n</i>	number of words to receive
out	<i>rxbuf</i>	the pointer to the receive buffer

Returns

The operation status.

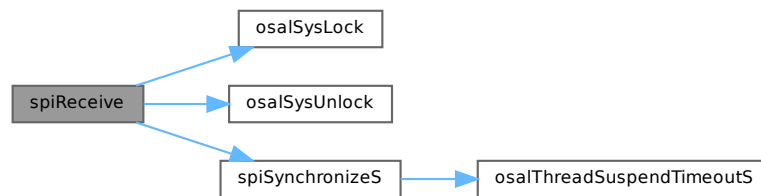
Return values

<i>MSG_OK</i>	if operation completed without errors.
<i>MSG_TIMEOUT</i>	if synchronization timed out.
<i>MSG_RESET</i>	if the transfer has been stopped.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.23 spiAcquireBus()

```
void spiAcquireBus (
    SPIDriver * spip)
```

Gains exclusive access to the SPI bus.

This function tries to gain ownership to the SPI bus, if the bus is already being used then the invoking thread is queued.

Precondition

In order to use this function the option `SPI_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.24 spiReleaseBus()

```
void spiReleaseBus (  
    SPIDriver * spip)
```

Releases exclusive access to the SPI bus.

Precondition

In order to use this function the option `SPI_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.21.6.25 spi_lld_init()

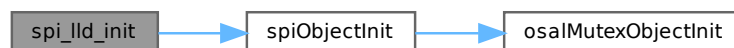
```
void spi_lld_init (  
    void )
```

Low level SPI driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.21.6.26 spi_lld_start()

```
msg_t spi_lld_start (  
    SPIDriver * spip)
```

Configures and activates the SPI peripheral.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.6.27 spi_llc_stop()

```
void spi_llc_stop (  
    SPIDriver * spip)
```

Deactivates the SPI peripheral.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.6.28 spi_llc_select()

```
void spi_llc_select (  
    SPIDriver * spip)
```

Asserts the slave select signal and prepares for transfers.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.6.29 spi_llc_unselect()

```
void spi_llc_unselect (  
    SPIDriver * spip)
```

Deasserts the slave select signal.

The previously selected peripheral is unselected.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.6.30 spi_llc_ignore()

```
msg_t spi_llc_ignore (  
    SPIDriver * spip,  
    size_t n)
```

Ignores data on the SPI bus.

This synchronous function performs the transmission of a series of idle words on the SPI bus and ignores the received data.

Precondition

In order to use this function the option `SPI_USE_SYNCHRONIZATION` must be enabled.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to be ignored

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.6.31 spi_llc_exchange()

```
msg_t spi_llc_exchange (  
    SPIDriver * spip,  
    size_t n,  
    const void * txbuf,  
    void * rxbuf)
```

Exchanges data on the SPI bus.

This asynchronous function starts a simultaneous transmit/receive operation.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to be exchanged
in	<i>txbuf</i>	the pointer to the transmit buffer
out	<i>rxbuf</i>	the pointer to the receive buffer

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.6.32 spi_llc_send()

```
msg_t spi_llc_send (  
    SPIDriver * spip,  
    size_t n,  
    const void * txbuf)
```

Sends data over the SPI bus.

This asynchronous function starts a transmit operation.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as uint8_t arrays for data sizes below or equal to 8 bits else it is organized as uint16_t arrays.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.6.33 spi_llt_receive()

```
msg_t spi_llt_receive (  
    SPIDriver * spip,  
    size_t n,  
    void * rxbuf)
```

Receives data from the SPI bus.

This asynchronous function starts a receive operation.

Postcondition

At the end of the operation the configured callback is invoked.

Note

The buffers are organized as uint8_t arrays for data sizes below or equal to 8 bits else it is organized as uint16_t arrays.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
in	<i>n</i>	number of words to receive
out	<i>rxbuf</i>	the pointer to the receive buffer

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.6.34 spi_llt_stop_transfer()

```
msg_t spi_llt_stop_transfer (  
    SPIDriver * spip,  
    size_t * sizep)
```

Aborts the ongoing SPI operation, if any.

Parameters

in	<i>spip</i>	pointer to the SPIDriver object
out	<i>sizep</i>	pointer to the counter of frames not yet transferred or NULL

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

7.2.9.21.6.35 spi_llc_polled_exchange()

```
uint16_t spi_llc_polled_exchange (
    SPIDriver * spip,
    uint16_t frame)
```

Exchanges one frame using a polled wait.

This synchronous function exchanges one frame using a polled synchronization method. This function is useful when exchanging small amount of data on high speed channels, usually in this situation is much more efficient just wait for completion using polling than suspending the thread waiting for an interrupt.

Parameters

in	<i>spip</i>	pointer to the <code>SPIDriver</code> object
in	<i>frame</i>	the data frame to send over the SPI bus

Returns

The received data frame from the SPI bus.

7.2.9.21.7 Variable Documentation

7.2.9.21.7.1 SPID1

```
SPIDriver SPID1
```

SPI1 driver identifier.

7.2.9.22 ST Driver

Generic System Tick Driver.

7.2.9.22.1 Detailed Description

Generic System Tick Driver.

This module implements a system tick timer in order to support the underlying operating system.

Macros

- #define `ST_LLD_NUM_ALARMS` 1

Typedefs

- typedef void(* `st_callback_t`) (unsigned alarm)

Functions

- void [stInit](#) (void)
ST Driver initialization.
- void [stBind](#) (void)
Enables an alarm interrupt on the invoking core.
- [systime_t](#) [stGetCounter](#) (void)
Returns the time counter value.
- void [stStartAlarm](#) ([systime_t](#) abstime)
Starts the alarm zero.
- void [stStopAlarm](#) (void)
Stops the alarm zero interrupt.
- void [stSetAlarm](#) ([systime_t](#) abstime)
Sets the alarm zero time.
- [systime_t](#) [stGetAlarm](#) (void)
Returns the alarm zero current time.
- bool [stIsAlarmActive](#) (void)
Determines if the alarm zero is active.
- void [stSetCallback](#) (unsigned alarm, [st_callback_t](#) cb)
Associates a callback to an alarm.
- void [stBindAlarmN](#) (unsigned alarm)
Enables an alarm interrupt on the invoking core.
- void [stStartAlarmN](#) (unsigned alarm, [systime_t](#) abstime)
Starts an additional alarm.
- void [stStopAlarmN](#) (unsigned alarm)
Stops an additional alarm.
- void [stSetAlarmN](#) (unsigned alarm, [systime_t](#) abstime)
Sets an additional alarm time.
- [systime_t](#) [stGetAlarmN](#) (unsigned alarm)
Returns an additional alarm current time.
- bool [stIsAlarmActiveN](#) (unsigned alarm)
Determines if the specified alarm is active.
- void [st_llc_init](#) (void)
Low level ST driver initialization.
- static [systime_t](#) [st_llc_get_counter](#) (void)
Returns the time counter value.
- static void [st_llc_start_alarm](#) ([systime_t](#) abstime)
Starts the alarm.
- static void [st_llc_stop_alarm](#) (void)
Stops the alarm interrupt.
- static void [st_llc_set_alarm](#) ([systime_t](#) abstime)
Sets the alarm time.
- static [systime_t](#) [st_llc_get_alarm](#) (void)
Returns the current alarm time.
- static bool [st_llc_is_alarm_active](#) (void)
Determines if the alarm is active.

Variables

- [st_callback_t](#) [st_callbacks](#) [[ST_LLC_NUM_ALARMS](#)]
Callback pointers for each alarm.

7.2.9.22.2 Macro Definition Documentation

7.2.9.22.2.1 ST_LLD_NUM_ALARMS

```
#define ST_LLD_NUM_ALARMS 1
```

7.2.9.22.3 Typedef Documentation

7.2.9.22.3.1 st_callback_t

```
typedef void(* st_callback_t) (unsigned alarm)
```

7.2.9.22.4 Function Documentation

7.2.9.22.4.1 stInit()

```
void stInit (
    void )
```

ST Driver initialization.

Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.22.4.2 stBind()

```
void stBind (
    void )
```

Enables an alarm interrupt on the invoking core.

Note

Must be called before any other alarm-related function.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.22.4.3 stGetCounter()

```
systime_t stGetCounter (  
    void )
```

Returns the time counter value.

Note

This functionality is only available in free running mode, the behaviour in periodic mode is undefined.

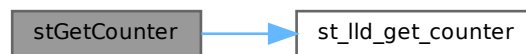
Returns

The counter value.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.22.4.4 stStartAlarm()

```
void stStartAlarm (  
    systime_t abstime)
```

Starts the alarm zero.

Note

Makes sure that no spurious alarms are triggered after this call.

This functionality is only available in free running mode, the behavior in periodic mode is undefined.

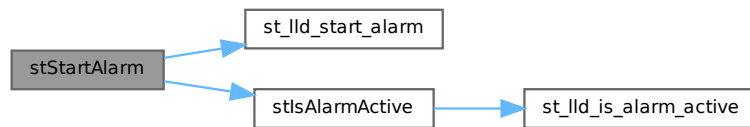
Parameters

in	<i>abstime</i>	the time to be set for the first alarm

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.22.4.5 stStopAlarm()**

```
void stStopAlarm (  
    void )
```

Stops the alarm zero interrupt.

Note

This functionality is only available in free running mode, the behavior in periodic mode is undefined.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.22.4.6 stSetAlarm()**

```
void stSetAlarm (  
    system_time_t abstime)
```

Sets the alarm zero time.

Note

This functionality is only available in free running mode, the behavior in periodic mode is undefined.

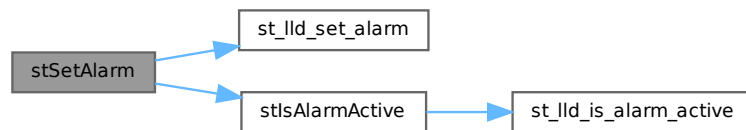
Parameters

in	<i>abstime</i>	the time to be set for the next alarm

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.22.4.7 stGetAlarm()**

```
system_time_t stGetAlarm (
    void )
```

Returns the alarm zero current time.

Note

This functionality is only available in free running mode, the behavior in periodic mode is undefined.

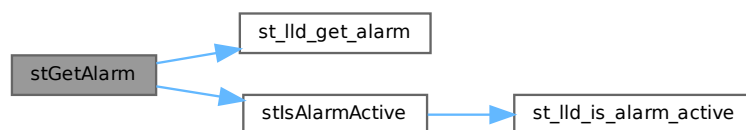
Returns

The currently set alarm time.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.22.4.8 stIsAlarmActive()

```
bool stIsAlarmActive (
    void )
```

Determines if the alarm zero is active.

Returns

The alarm status.

Return values

<i>false</i>	if the alarm is not active.
<i>true</i>	is the alarm is active

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.22.4.9 stSetCallback()

```
void stSetCallback (
    unsigned alarm,
    st_callback_t cb)
```

Associates a callback to an alarm.

Note

Makes sure that no spurious alarms are triggered after this call.

This functionality is only available in free running mode, the behavior in periodic mode is undefined.

Parameters

in	<i>alarm</i>	alarm channel number (0..ST_LLD_NUM_ALARMS-1)
in	<i>cb</i>	alarm callback or NULL

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.22.4.10 stBindAlarmN()

```
void stBindAlarmN (  
    unsigned alarm)
```

Enables an alarm interrupt on the invoking core.

Note

Must be called before any other alarm-related function.

Parameters

in	<i>alarm</i>	alarm channel number (0..ST_LLD_NUM_ALARMS-1)

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.22.4.11 stStartAlarmN()

```
void stStartAlarmN (  
    unsigned alarm,  
    systime_t abstime)
```

Starts an additional alarm.

Note

Makes sure that no spurious alarms are triggered after this call.

This functionality is only available in free running mode, the behavior in periodic mode is undefined.

Parameters

in	<i>alarm</i>	alarm channel number (0..ST_LLD_NUM_ALARMS-1)
in	<i>abstime</i>	the time to be set for the first alarm

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.22.4.12 stStopAlarmN()**

```
void stStopAlarmN (  
    unsigned alarm)
```

Stops an additional alarm.

Note

This functionality is only available in free running mode, the behavior in periodic mode is undefined.

Parameters

in	<i>alarm</i>	alarm channel number (0..ST_LLD_NUM_ALARMS-1)

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.22.4.13 stSetAlarmN()

```
void stSetAlarmN (  
    unsigned alarm,  
    systime_t abstime)
```

Sets an additional alarm time.

Note

This functionality is only available in free running mode, the behavior in periodic mode is undefined.

Parameters

in	<i>alarm</i>	alarm channel number (0..ST_LLD_NUM_ALARMS-1)
in	<i>abstime</i>	the time to be set for the next alarm

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.22.4.14 stGetAlarmN()**

```
sys_time_t stGetAlarmN (  
    unsigned alarm)
```

Returns an additional alarm current time.

Note

This functionality is only available in free running mode, the behavior in periodic mode is undefined.

Parameters

in	<i>alarm</i>	alarm channel number (0..ST_LLD_NUM_ALARMS-1)

Returns

The currently set alarm time.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.22.4.15 stIsAlarmActiveN()

```
bool stIsAlarmActiveN (
    unsigned alarm)
```

Determines if the specified alarm is active.

Parameters

in	<i>alarm</i>	alarm channel number (0..ST_LLD_NUM_ALARMS-1)

Returns

The alarm status.

Return values

<i>false</i>	if the alarm is not active.
<i>true</i>	is the alarm is active

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.22.4.16 st_lld_init()

```
void st_lld_init (
    void )
```

Low level ST driver initialization.

Function Class:

Not an API, this function is for internal use only.

7.2.9.22.4.17 st_lld_get_counter()

```
system_time_t st_lld_get_counter (
    void ) [inline], [static]
```

Returns the time counter value.

Returns

The counter value.

Function Class:

Not an API, this function is for internal use only.

7.2.9.22.4.18 st_lld_start_alarm()

```
void st_lld_start_alarm (
    system_time_t abstime) [inline], [static]
```

Starts the alarm.

Note

Makes sure that no spurious alarms are triggered after this call.

Parameters

in	<i>abstime</i>	the time to be set for the first alarm

Function Class:

Not an API, this function is for internal use only.

7.2.9.22.4.19 st_lld_stop_alarm()

```
void st_lld_stop_alarm (
    void ) [inline], [static]
```

Stops the alarm interrupt.

Function Class:

Not an API, this function is for internal use only.

7.2.9.22.4.20 st_lld_set_alarm()

```
void st_lld_set_alarm (
    systime_t abstime) [inline], [static]
```

Sets the alarm time.

Parameters

in	<i>abstime</i>	the time to be set for the next alarm

Function Class:

Not an API, this function is for internal use only.

7.2.9.22.4.21 st_lld_get_alarm()

```
systime_t st_lld_get_alarm (
    void ) [inline], [static]
```

Returns the current alarm time.

Returns

The currently set alarm time.

Function Class:

Not an API, this function is for internal use only.

7.2.9.22.4.22 st_lld_is_alarm_active()

```
bool st_lld_is_alarm_active (
    void ) [inline], [static]
```

Determines if the alarm is active.

Returns

The alarm status.

Return values

<i>false</i>	if the alarm is not active.
<i>true</i>	is the alarm is active

Function Class:

Not an API, this function is for internal use only.

7.2.9.22.5 Variable Documentation

7.2.9.22.5.1 `st_callbacks`

`st_callback_t st_callbacks[ST_LLD_NUM_ALARMS]`

Callback pointers for each alarm.

Note

If some alarms have static callbacks defined in the LLD then some of the pointers might be unused (never called through).

7.2.9.23 TRNG Driver

Generic True Random Numbers Generator Driver.

7.2.9.23.1 Detailed Description

Generic True Random Numbers Generator Driver.

This module implements a generic TRNG driver.

Precondition

In order to use the TRNG driver the `HAL_USE_TRNG` option must be enabled in `halconf.h`.

PLATFORM configuration options

- `#define PLATFORM_TRNG_USE_TRNG1 FALSE`
TRNGD1 driver enable switch.

Data Structures

- struct `hal_trng_config`
Driver configuration structure.
- struct `hal_trng_driver`
Structure representing a TRNG driver.

Macros

- `#define trng_lld_driver_fields`
Low level fields of the TRNG driver structure.
- `#define trng_lld_config_fields`
Low level fields of the TRNG configuration structure.

Typedefs

- `typedef struct hal_trng_driver TRNGDriver`
Type of a structure representing a TRNG driver.
- `typedef struct hal_trng_config TRNGConfig`
Driver configuration structure.

Enumerations

- `enum trngstate_t { TRNG_UNINIT = 0 , TRNG_STOP = 1 , TRNG_READY = 2 , TRNG_RUNNING = 3 }`
Driver state machine possible states.

Functions

- `void trngInit (void)`
TRNG Driver initialization.
- `void trngObjectInit (TRNGDriver *trngp)`
*Initializes the standard part of a *TRNGDriver* structure.*
- `msg_t trngStart (TRNGDriver *trngp, const TRNGConfig *config)`
Configures and activates the TRNG peripheral.
- `void trngStop (TRNGDriver *trngp)`
Deactivates the TRNG peripheral.
- `bool trngGenerate (TRNGDriver *trngp, size_t size, uint8_t *out)`
True random numbers generator.
- `void trng_lld_init (void)`
Low level TRNG driver initialization.
- `void trng_lld_start (TRNGDriver *trngp)`
Configures and activates the TRNG peripheral.
- `void trng_lld_stop (TRNGDriver *trngp)`
Deactivates the TRNG peripheral.
- `bool trng_lld_generate (TRNGDriver *trngp, size_t size, uint8_t *out)`
True random numbers generator.

Variables

- `TRNGDriver TRNGD1`
TRNGD1 driver identifier.

7.2.9.23.2 Macro Definition Documentation

7.2.9.23.2.1 PLATFORM_TRNG_USE_TRNG1

```
#define PLATFORM_TRNG_USE_TRNG1 FALSE
```

TRNGD1 driver enable switch.

If set to TRUE the support for TRNGD1 is included.

Note

The default is FALSE.

7.2.9.23.2.2 trng_lld_driver_fields

```
#define trng_lld_driver_fields
```

Value:

```
/* Dummy field, it is not needed.*/
uint32_t dummy;
```

Low level fields of the TRNG driver structure.

7.2.9.23.2.3 trng_lld_config_fields

```
#define trng_lld_config_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/
uint32_t dummy;
```

Low level fields of the TRNG configuration structure.

7.2.9.23.3 Typedef Documentation

7.2.9.23.3.1 TRNGDriver

```
typedef struct hal_trng_driver TRNGDriver
```

Type of a structure representing a TRNG driver.

7.2.9.23.3.2 TRNGConfig

```
typedef struct hal_trng_config TRNGConfig
```

Driver configuration structure.

Note

It could be empty on some architectures.

7.2.9.23.4 Enumeration Type Documentation

7.2.9.23.4.1 trngstate_t

```
enum trngstate_t
```

Driver state machine possible states.

Enumerator

TRNG_UNINIT	Not initialized.
TRNG_STOP	Stopped.
TRNG_READY	Ready.
TRNG_RUNNING	Generating random number.

7.2.9.23.5 Function Documentation

7.2.9.23.5.1 trngInit()

```
void trngInit (  
    void )
```

TRNG Driver initialization.

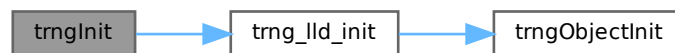
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module nitializer function.

Here is the call graph for this function:



7.2.9.23.5.2 trngObjectInit()

```
void trngObjectInit (  
    TRNGDriver * trngp)
```

Initializes the standard part of a `TRNGDriver` structure.

Parameters

out	<i>trngp</i>	pointer to the <code>TRNGDriver</code> object

Function Class:

Object or module nitializer function.

7.2.9.23.5.3 trngStart()

```
msg_t trngStart (  
    TRNGDriver * trngp,  
    const TRNGConfig * config)
```

Configures and activates the TRNG peripheral.

Parameters

in	<i>trngp</i>	pointer to the TRNGDriver object
in	<i>config</i>	pointer to the TRNGConfig object or <code>NULL</code> for default configuration

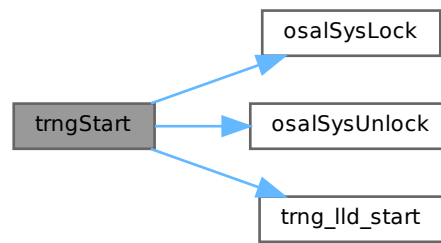
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.23.5.4 trngStop()**

```
void trngStop (  
    TRNGDriver * trngp)
```

Deactivates the TRNG peripheral.

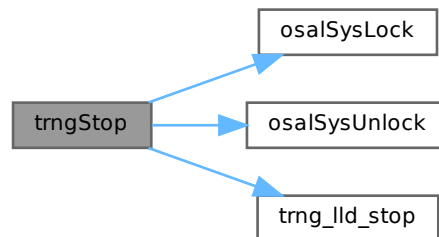
Parameters

in	<i>trngp</i>	pointer to the TRNGDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.23.5.5 trngGenerate()**

```

bool trngGenerate (
    TRNGDriver * trngp,
    size_t size,
    uint8_t * out)
  
```

True random numbers generator.

Note

The function is blocking and likely performs polled waiting inside the low level implementation.

Parameters

in	<i>trngp</i>	pointer to the <code>TRNGDriver</code> object
in	<i>size</i>	size of output buffer
out	<i>out</i>	output buffer

Returns

The operation status.

Return values

<i>false</i>	if a random number has been generated.
<i>true</i>	if an HW error occurred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.23.5.6 trng_lld_init()**

```
void trng_lld_init (  
    void )
```

Low level TRNG driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.23.5.7 trng_lld_start()**

```
void trng_lld_start (  
    TRNGDriver * trngp)
```

Configures and activates the TRNG peripheral.

Parameters

in	<i>trngp</i>	pointer to the TRNGDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.23.5.8 trng_lld_stop()

```
void trng_lld_stop (  
    TRNGDriver * trngp)
```

Deactivates the TRNG peripheral.

Parameters

in	<i>trngp</i>	pointer to the TRNGDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.23.5.9 trng_lld_generate()

```
bool trng_lld_generate (  
    TRNGDriver * trngp,  
    size_t size,  
    uint8_t * out)
```

True random numbers generator.

Note

The function is blocking and likely performs polled waiting inside the low level implementation.

Parameters

in	<i>trngp</i>	pointer to the TRNGDriver object
in	<i>size</i>	size of output buffer
out	<i>out</i>	output buffer

Returns

The operation status.

Return values

<i>false</i>	if a random number has been generated.
<i>true</i>	if an HW error occurred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.23.6 Variable Documentation

7.2.9.23.6.1 TRNGD1

[TRNGDriver](#) TRNGD1

TRNGD1 driver identifier.

7.2.9.24 UART Driver

Generic UART Driver.

7.2.9.24.1 Detailed Description

Generic UART Driver.

This driver abstracts a generic UART (Universal Asynchronous Receiver Transmitter) peripheral, the API is designed to be:

- Unbuffered and copy-less, transfers are always directly performed from/to the application-level buffers without extra copy operations.
- Asynchronous, the API is always non blocking.
- Callbacks capable, operations completion and other events are notified using callbacks.

Special hardware features like deep hardware buffers, DMA transfers are hidden to the user but fully supportable by the low level implementations.

This driver model is best used where communication events are meant to drive an higher level state machine, as example:

- RS485 drivers.
- Multipoint network drivers.
- Serial protocol decoders.

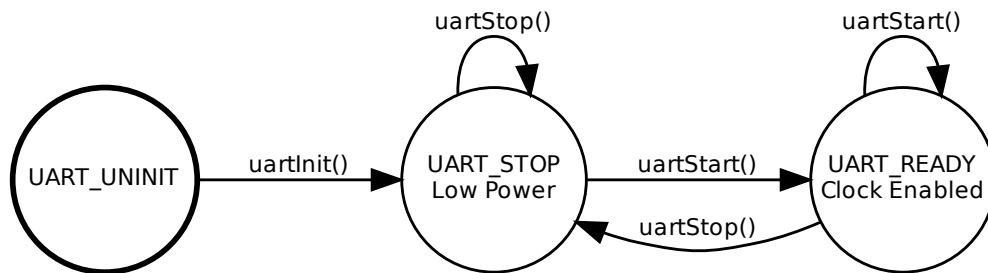
If your application requires a synchronous buffered driver then the [Serial Driver](#) should be used instead.

Precondition

In order to use the UART driver the `HAL_USE_UART` option must be enabled in [halconf.h](#).

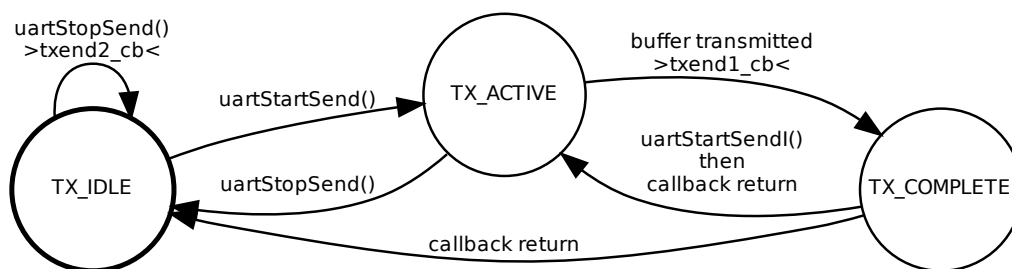
7.2.9.24.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



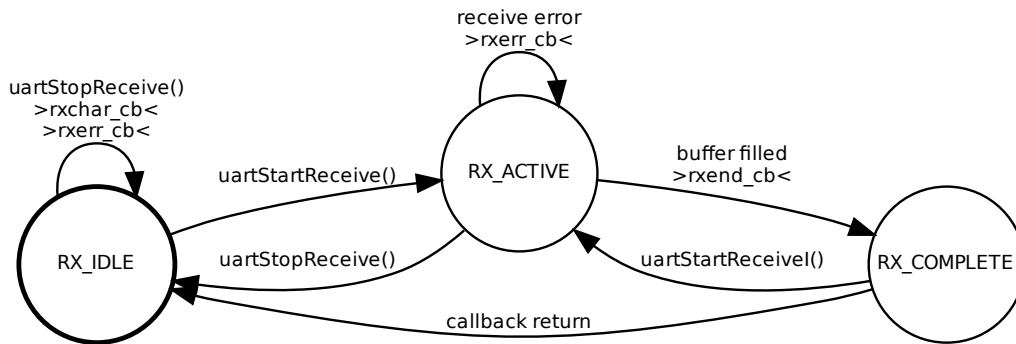
7.2.9.24.2.1 Transmitter sub State Machine

The follow diagram describes the transmitter state machine, this diagram is valid while the driver is in the `UART_READY` state. This state machine is automatically reset to the `TX_IDLE` state each time the driver enters the `UART_READY` state.



7.2.9.24.2.2 Receiver sub State Machine

The follow diagram describes the receiver state machine, this diagram is valid while the driver is in the `UART_READY` state. This state machine is automatically reset to the `RX_IDLE` state each time the driver enters the `UART_READY` state.



UART status flags

- #define `UART_NO_ERROR` 0
No pending conditions.
- #define `UART_PARITY_ERROR` 4
Parity error happened.
- #define `UART_FRAMING_ERROR` 8
Framing error happened.
- #define `UART_OVERRUN_ERROR` 16
Overflow happened.
- #define `UART_NOISE_ERROR` 32
Noise on the line.
- #define `UART_BREAK_DETECTED` 64
Break detected.

UART error conditions

- #define `UART_ERR_NOT_ACTIVE` (size_t)-1

UART configuration options

- #define `UART_USE_WAIT` FALSE
Enables synchronous APIs.
- #define `UART_USE_MUTUAL_EXCLUSION` FALSE
Enables the `uartAcquireBus()` and `uartReleaseBus()` APIs.

Low level driver helper macros

- `#define _uart_wakeup_tx1_isr(uartp)`
Wakes up the waiting thread in case of early TX complete.
- `#define _uart_wakeup_tx2_isr(uartp)`
Wakes up the waiting thread in case of late TX complete.
- `#define _uart_wakeup_rx_complete_isr(uartp)`
Wakes up the waiting thread in case of RX complete.
- `#define _uart_wakeup_rx_error_isr(uartp)`
Wakes up the waiting thread in case of RX error.
- `#define _uart_wakeup_rx_cm_isr(uartp)`
Wakes up the waiting thread in case of RX character match.
- `#define _uart_wakeup_rx_timeout_isr(uartp)`
Wakes up the waiting thread in case of RX timeout.
- `#define _uart_tx1_isr_code(uartp)`
Common ISR code for early TX.
- `#define _uart_tx2_isr_code(uartp)`
Common ISR code for late TX.
- `#define _uart_rx_complete_isr_code(uartp)`
Common ISR code for RX complete.
- `#define _uart_rx_error_isr_code(uartp, errors)`
Common ISR code for RX error.
- `#define _uart_rx_idle_code(uartp)`
Common ISR code for RX on idle.
- `#define _uart_timeout_isr_code(uartp)`
Timeout ISR code for receiver.
- `#define _uart_rx_char_match_isr_code(uartp)`
Character match ISR code for receiver.

PLATFORM configuration options

- `#define PLATFORM_UART_USE_UART1 FALSE`
UART driver enable switch.

Data Structures

- struct `hal_uart_config`
Driver configuration structure.
- struct `hal_uart_driver`
Structure representing an UART driver.

Typedefs

- typedef uint32_t `uartflags_t`
UART driver condition flags type.
- typedef struct `hal_uart_driver` `UARTDriver`
Type of structure representing an UART driver.
- typedef void(* `uartcb_t`) (`UARTDriver` *uartp)
Generic UART notification callback type.
- typedef void(* `uartccb_t`) (`UARTDriver` *uartp, uint16_t c)
Character received UART notification callback type.
- typedef void(* `uartecb_t`) (`UARTDriver` *uartp, `uartflags_t` e)
Receive error UART notification callback type.
- typedef struct `hal_uart_config` `UARTConfig`
Driver configuration structure.

Enumerations

- enum `uartstate_t` { `UART_UNINIT` = 0 , `UART_STOP` = 1 , `UART_READY` = 2 }
Driver state machine possible states.
- enum `uarttxstate_t` { `UART_TX_IDLE` = 0 , `UART_TX_ACTIVE` = 1 , `UART_TX_COMPLETE` = 2 }
Transmitter state machine states.
- enum `uartrxstate_t` { `UART_RX_IDLE` = 0 , `UART_RX_ACTIVE` = 1 , `UART_RX_COMPLETE` = 2 }
Receiver state machine states.

Functions

- void `uartInit` (void)
UART Driver initialization.
- void `uartObjectInit` (`UARTDriver` *uartp)
Initializes the standard part of a `UARTDriver` structure.
- `msg_t` `uartStart` (`UARTDriver` *uartp, const `UARTConfig` *config)
Configures and activates the UART peripheral.
- void `uartStop` (`UARTDriver` *uartp)
Deactivates the UART peripheral.
- void `uartStartSend` (`UARTDriver` *uartp, `size_t` n, const void *txbuf)
Starts a transmission on the UART peripheral.
- void `uartStartSendI` (`UARTDriver` *uartp, `size_t` n, const void *txbuf)
Starts a transmission on the UART peripheral.
- `size_t` `uartStopSend` (`UARTDriver` *uartp)
Stops any ongoing transmission.
- `size_t` `uartStopSendI` (`UARTDriver` *uartp)
Stops any ongoing transmission.
- void `uartStartReceive` (`UARTDriver` *uartp, `size_t` n, void *rxbuf)
Starts a receive operation on the UART peripheral.
- void `uartStartReceiveI` (`UARTDriver` *uartp, `size_t` n, void *rxbuf)
Starts a receive operation on the UART peripheral.
- `size_t` `uartStopReceive` (`UARTDriver` *uartp)
Stops any ongoing receive operation.
- `size_t` `uartStopReceiveI` (`UARTDriver` *uartp)
Stops any ongoing receive operation.
- `msg_t` `uartSendTimeout` (`UARTDriver` *uartp, `size_t` *np, const void *txbuf, `sysinterval_t` timeout)
Performs a transmission on the UART peripheral.
- `msg_t` `uartSendFullTimeout` (`UARTDriver` *uartp, `size_t` *np, const void *txbuf, `sysinterval_t` timeout)
Performs a transmission on the UART peripheral.
- `msg_t` `uartReceiveTimeout` (`UARTDriver` *uartp, `size_t` *np, void *rxbuf, `sysinterval_t` timeout)
Performs a receive operation on the UART peripheral.
- void `uartAcquireBus` (`UARTDriver` *uartp)
Gains exclusive access to the UART bus.
- void `uartReleaseBus` (`UARTDriver` *uartp)
Releases exclusive access to the UART bus.
- void `uart_lld_init` (void)
Low level UART driver initialization.
- void `uart_lld_start` (`UARTDriver` *uartp)
Configures and activates the UART peripheral.
- void `uart_lld_stop` (`UARTDriver` *uartp)

Deactivates the UART peripheral.

- void `uart_lld_start_send` (`UARTDriver` *uartp, size_t n, const void *txbuf)

Starts a transmission on the UART peripheral.

- size_t `uart_lld_stop_send` (`UARTDriver` *uartp)

Stops any ongoing transmission.

- void `uart_lld_start_receive` (`UARTDriver` *uartp, size_t n, void *rxbuf)

Starts a receive operation on the UART peripheral.

- size_t `uart_lld_stop_receive` (`UARTDriver` *uartp)

Stops any ongoing receive operation.

Variables

- `UARTDriver` `UARTD1`

UART1 driver identifier.

7.2.9.24.3 Macro Definition Documentation

7.2.9.24.3.1 UART_NO_ERROR

```
#define UART_NO_ERROR 0
```

No pending conditions.

7.2.9.24.3.2 UART_PARITY_ERROR

```
#define UART_PARITY_ERROR 4
```

Parity error happened.

7.2.9.24.3.3 UART_FRAMING_ERROR

```
#define UART_FRAMING_ERROR 8
```

Framing error happened.

7.2.9.24.3.4 UART_OVERRUN_ERROR

```
#define UART_OVERRUN_ERROR 16
```

Overflow happened.

7.2.9.24.3.5 UART_NOISE_ERROR

```
#define UART_NOISE_ERROR 32
```

Noise on the line.

7.2.9.24.3.6 UART_BREAK_DETECTED

```
#define UART_BREAK_DETECTED 64
```

Break detected.

7.2.9.24.3.7 UART_ERR_NOT_ACTIVE

```
#define UART_ERR_NOT_ACTIVE (size_t)-1
```

7.2.9.24.3.8 UART_USE_WAIT

```
#define UART_USE_WAIT FALSE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.9.24.3.9 UART_USE_MUTUAL_EXCLUSION

```
#define UART_USE_MUTUAL_EXCLUSION FALSE
```

Enables the `uartAcquireBus()` and `uartReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.9.24.3.10 _uart_wakeup_tx1_isr

```
#define _uart_wakeup_tx1_isr(  
    uartp)
```

Value:

```
{  
    if ((uartp)->early == true) {  
        osalSysLockFromISR();  
        osalThreadResumeI(&(uartp)->threadtx, MSG_OK);  
        osalSysUnlockFromISR();  
    }  
}
```

Wakes up the waiting thread in case of early TX complete.

Parameters

in	<code>uartp</code>	pointer to the <code>UARTDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.11 _uart_wakeup_tx2_isr

```
#define _uart_wakeup_tx2_isr(  
    uartp)
```

Value:

```
{  
    if ((uartp)->early == false) {  
        osalSysLockFromISR();  
        osalThreadResumeI (&(uartp)->threadtx, MSG_OK);  
        osalSysUnlockFromISR();  
    }  
}
```

Wakes up the waiting thread in case of late TX complete.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.12 _uart_wakeup_rx_complete_isr

```
#define _uart_wakeup_rx_complete_isr(  
    uartp)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI (&(uartp)->threadrx, MSG_OK);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread in case of RX complete.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.13 _uart_wakeup_rx_error_isr

```
#define _uart_wakeup_rx_error_isr(  
    uartp)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI (&(uartp)->threadrx, MSG_RESET);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread in case of RX error.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.14 _uart_wakeup_rx_cm_isr

```
#define _uart_wakeup_rx_cm_isr(  
    uartp)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI(&(uartp)->threadrx, MSG_TIMEOUT);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread in case of RX character match.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.15 _uart_wakeup_rx_timeout_isr

```
#define _uart_wakeup_rx_timeout_isr(  
    uartp)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI(&(uartp)->threadrx, MSG_TIMEOUT);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread in case of RX timeout.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.16 _uart_tx1_isr_code

```
#define _uart_tx1_isr_code(  
    uartp)
```

Value:

```
{  
    (uartp)->txstate = UART_TX_COMPLETE; \  
    if ((uartp)->config->txendl_cb != NULL) {  
        (uartp)->config->txendl_cb(uartp);  
    }  
    if ((uartp)->txstate == UART_TX_COMPLETE) {  
        (uartp)->txstate = UART_TX_IDLE;  
    }  
    _uart_wakeup_tx1_isr(uartp);  
}
```

Common ISR code for early TX.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.17 _uart_tx2_isr_code

```
#define _uart_tx2_isr_code(  
    uartp)
```

Value:

```
{  
    if ((uartp)->config->txend2_cb != NULL) {  
        (uartp)->config->txend2_cb(uartp);  
    }  
    _uart_wakeup_tx2_isr(uartp);  
}
```

Common ISR code for late TX.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.18 _uart_rx_complete_isr_code

```
#define _uart_rx_complete_isr_code(  
    uartp)
```

Value:

```
{  
    (uartp)->rxstate = UART\_RX\_COMPLETE;  
    if ((uartp)->config->rxend_cb != NULL) {  
        (uartp)->config->rxend_cb(uartp);  
    }  
    if ((uartp)->rxstate == UART\_RX\_COMPLETE) {  
        (uartp)->rxstate = UART\_RX\_IDLE;  
        uart_enter_rx_idle_loop(uartp);  
    }  
    _uart_wakeup_rx_complete_isr(uartp);  
}
```

Common ISR code for RX complete.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.19 _uart_rx_error_isr_code

```
#define _uart_rx_error_isr_code(  
    uartp,  
    errors)
```

Value:

```
{  
    if ((uartp)->config->rxerr_cb != NULL) {  
        (uartp)->config->rxerr_cb(uartp, errors);  
    }  
    _uart_wakeup_rx_error_isr(uartp);  
}
```

Common ISR code for RX error.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object
in	<i>errors</i>	mask of errors to be reported

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.20 _uart_rx_idle_code

```
#define _uart_rx_idle_code(  
    uartp)
```

Value:

```
{  
    if ((uartp)->config->rxchar_cb != NULL)  
        (uartp)->config->rxchar_cb(uartp, (uartp)->rxbuf);  
}
```

Common ISR code for RX on idle.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.21 _uart_timeout_isr_code

```
#define _uart_timeout_isr_code(  
    uartp)
```

Value:

```
{  
    if ((uartp)->config->timeout_cb != NULL) {  
        (uartp)->config->timeout_cb(uartp);  
    }  
    _uart_wakeup_rx_timeout_isr(uartp);  
}
```

Timeout ISR code for receiver.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.22 _uart_rx_char_match_isr_code

```
#define _uart_rx_char_match_isr_code(  
    uartp)
```

Value:

```
{  
    if ((uartp)->config->rx_cm_cb != NULL) {  
        (uartp)->config->rx_cm_cb(uartp);  
    }  
    _uart_wakeup_rx_cm_isr(uartp);  
}
```

Character match ISR code for receiver.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.3.23 PLATFORM_UART_USE_UART1

```
#define PLATFORM_UART_USE_UART1 FALSE
```

UART driver enable switch.

If set to `TRUE` the support for UART1 is included.

Note

The default is `FALSE`.

7.2.9.24.4 Typedef Documentation

7.2.9.24.4.1 uartflags_t

```
typedef uint32_t uartflags_t
```

UART driver condition flags type.

7.2.9.24.4.2 UARTDriver

```
typedef struct hal_uart_driver UARTDriver
```

Type of structure representing an UART driver.

7.2.9.24.4.3 uartcb_t

```
typedef void(* uartcb_t) (UARTDriver *uartp)
```

Generic UART notification callback type.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

7.2.9.24.4.4 uartccb_t

```
typedef void(* uartccb_t) (UARTDriver *uartp, uint16_t c)
```

Character received UART notification callback type.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object triggering the callback
in	<i>c</i>	received character

7.2.9.24.4.5 uartecb_t

```
typedef void(* uartecb_t) (UARTDriver *uartp, uartflags_t e)
```

Receive error UART notification callback type.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object triggering the callback
in	<i>e</i>	receive error mask

7.2.9.24.4.6 UARTConfig

```
typedef struct hal_uart_config UARTConfig
```

Driver configuration structure.

Note

Implementations may extend this structure to contain more, architecture dependent, fields.

7.2.9.24.5 Enumeration Type Documentation**7.2.9.24.5.1 uartstate_t**

```
enum uartstate_t
```

Driver state machine possible states.

Enumerator

UART_UNINIT	Not initialized.
UART_STOP	Stopped.
UART_READY	Ready.

7.2.9.24.5.2 uarttxstate_t

```
enum uarttxstate_t
```

Transmitter state machine states.

Enumerator

UART_TX_IDLE	Not transmitting.
UART_TX_ACTIVE	Transmitting.
UART_TX_COMPLETE	Buffer complete.

7.2.9.24.5.3 uartrxstate_t

```
enum uartrxstate_t
```

Receiver state machine states.

Enumerator

UART_RX_IDLE	Not receiving.
UART_RX_ACTIVE	Receiving.
UART_RX_COMPLETE	Buffer complete.

7.2.9.24.6 Function Documentation**7.2.9.24.6.1 uartInit()**

```
void uartInit (
    void )
```

UART Driver initialization.

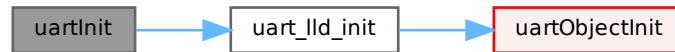
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.24.6.2 uartObjectInit()**

```
void uartObjectInit (
    UARTDriver * uartp)
```

Initializes the standard part of a `UARTDriver` structure.

Parameters

out	uartp	pointer to the <code>UARTDriver</code> object

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.24.6.3 uartStart()**

```
msg_t uartStart (
    UARTDriver * uartp,
    const UARTConfig * config)
```

Configures and activates the UART peripheral.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object
in	<i>config</i>	pointer to the UARTConfig object

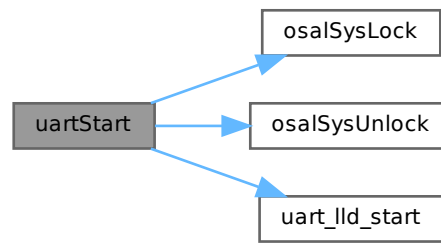
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.24.6.4 uartStop()**

```
void uartStop (  
    UARTDriver * uartp)
```

Deactivates the UART peripheral.

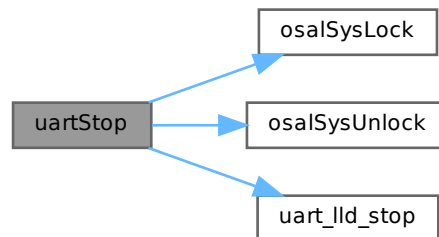
Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.24.6.5 uartStartSend()**

```
void uartStartSend (  
    UARTDriver * uartp,  
    size_t n,  
    const void * txbuf)
```

Starts a transmission on the UART peripheral.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

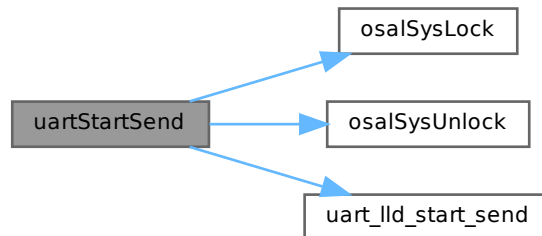
Parameters

in	<i>uartp</i>	pointer to the <code>UARTDriver</code> object
in	<i>n</i>	number of data frames to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.24.6.6 uartStartSendI()**

```
void uartStartSendI (
    UARTDriver * uartp,
    size_t n,
    const void * txbuf)
```

Starts a transmission on the UART peripheral.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

This function has to be invoked from a lock zone.

Parameters

in	<i>uartp</i>	pointer to the <code>UARTDriver</code> object
in	<i>n</i>	number of data frames to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.24.6.7 uartStopSend()

```
size_t uartStopSend (
    UARTDriver * uartp)
```

Stops any ongoing transmission.

Note

Stopping a transmission also suppresses the transmission callbacks.

Parameters

in	uartp	pointer to the <code>UARTDriver</code> object

Returns

The number of data frames not transmitted by the stopped transmit operation.

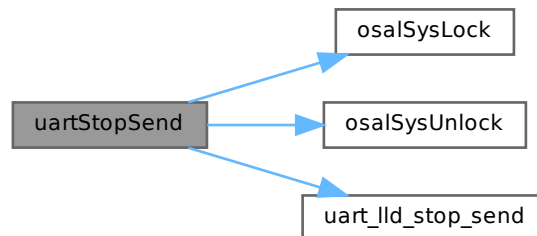
Return values

<code>UART_ERR_NOT_ACTIVE</code>	if there was no transmit operation in progress.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.24.6.8 uartStopSendI()**

```
size_t uartStopSendI (
    UARTDriver * uartp)
```

Stops any ongoing transmission.

Note

Stopping a transmission also suppresses the transmission callbacks.

This function has to be invoked from a lock zone.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Returns

The number of data frames not transmitted by the stopped transmit operation.

Return values

UART_ERR_NOT_ACTIVE	if there was no transmit operation in progress.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.24.6.9 uartStartReceive()**

```

void uartStartReceive (
    UARTDriver * uartp,
    size_t n,
    void * rxbuf)
  
```

Starts a receive operation on the UART peripheral.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

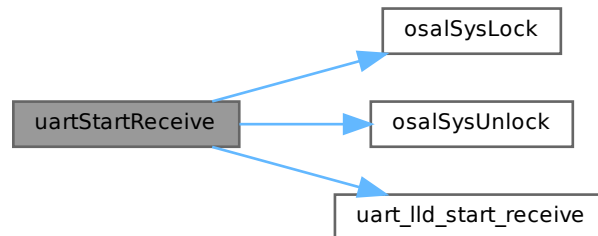
Parameters

in	<i>uartp</i>	pointer to the UARTDriver object
in	<i>n</i>	number of data frames to receive
in	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.24.6.10 uartStartReceiveI()**

```
void uartStartReceiveI (
    UARTDriver * uartp,
    size_t n,
    void * rxbuf)
```

Starts a receive operation on the UART peripheral.

Note

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

This function has to be invoked from a lock zone.

Parameters

in	<i>uartp</i>	pointer to the <code>UARTDriver</code> object
in	<i>n</i>	number of data frames to receive
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.24.6.11 uartStopReceive()

```
size_t uartStopReceive (
    UARTDriver * uartp)
```

Stops any ongoing receive operation.

Note

Stopping a receive operation also suppresses the receive callbacks.

Parameters

in	uartp	pointer to the <code>UARTDriver</code> object

Returns

The number of data frames not received by the stopped receive operation.

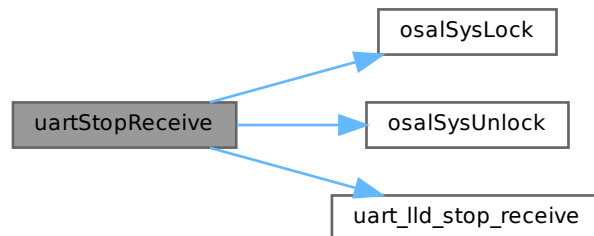
Return values

<code>UART_ERR_NOT_ACTIVE</code>	if there was no receive operation in progress.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.24.6.12 uartStopReceiveI()

```
size_t uartStopReceiveI (
    UARTDriver * uartp)
```

Stops any ongoing receive operation.

Note

Stopping a receive operation also suppresses the receive callbacks.

This function has to be invoked from a lock zone.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Returns

The number of data frames not received by the stopped receive operation.

Return values

<code>UART_ERR_NOT_ACTIVE</code>	if there was no receive operation in progress.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.24.6.13 uartSendTimeout()**

```

msg_t uartSendTimeout (
    UARTDriver * uartp,
    size_t * np,
    const void * txbuf,
    sysinterval_t timeout)
  
```

Performs a transmission on the UART peripheral.

Note

The function returns when the specified number of frames have been sent to the UART or on timeout.

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

This function implements a software timeout, it does not use any underlying HW timeout mechanism.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object
in, out	<i>np</i>	number of data frames to transmit, on exit the number of frames actually transmitted
in	<i>txbuf</i>	the pointer to the transmit buffer
in	<i>timeout</i>	operation timeout

Returns

The operation status.

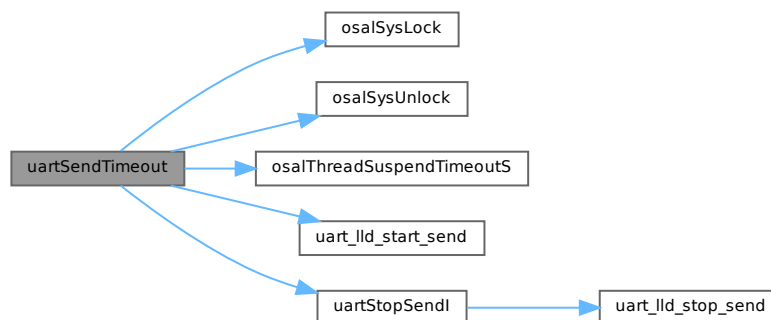
Return values

<i>MSG_OK</i>	if the operation completed successfully.
<i>MSG_TIMEOUT</i>	if the operation timed out.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.24.6.14 uartSendFullTimeout()

```

msg_t uartSendFullTimeout (
    UARTDriver * uartp,
    size_t * np,
    const void * txbuf,
    sysinterval_t timeout)

```

Performs a transmission on the UART peripheral.

Note

The function returns when the specified number of frames have been physically transmitted or on timeout.

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

This function implements a software timeout, it does not use any underlying HW timeout mechanism.

Parameters

in	<i>uartp</i>	pointer to the <code>UARTDriver</code> object
in, out	<i>np</i>	number of data frames to transmit, on exit the number of frames actually transmitted
in	<i>txbuf</i>	the pointer to the transmit buffer
in	<i>timeout</i>	operation timeout

Returns

The operation status.

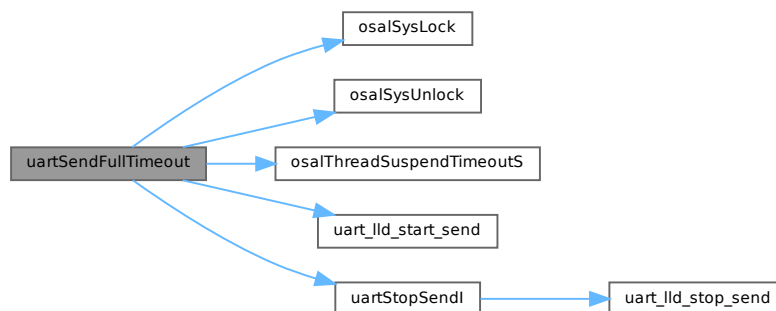
Return values

<code>MSG_OK</code>	if the operation completed successfully.
<code>MSG_TIMEOUT</code>	if the operation timed out.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

7.2.9.24.6.15 `uartReceiveTimeout()`

```

msg_t uartReceiveTimeout (
    UARTDriver * uartp,
    size_t * np,
    void * rxbuf,
    sysinterval_t timeout)

```

Performs a receive operation on the UART peripheral.

Note

The function returns when the specified number of frames have been received or on error/timeout.

The buffers are organized as `uint8_t` arrays for data sizes below or equal to 8 bits else it is organized as `uint16_t` arrays.

This function implements a software timeout, it does not use any underlying HW timeout mechanism.

Parameters

in	<code>uartp</code>	pointer to the <code>UARTDriver</code> object
in, out	<code>np</code>	number of data frames to receive, on exit the number of frames actually received
in	<code>rxbuf</code>	the pointer to the receive buffer
in	<code>timeout</code>	operation timeout

Returns

The operation status.

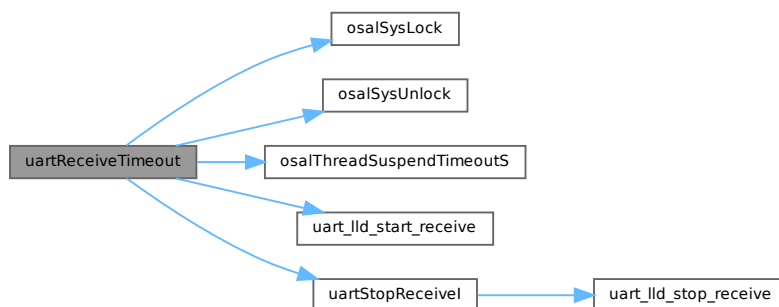
Return values

<i>MSG_OK</i>	if the operation completed successfully.
<i>MSG_TIMEOUT</i>	if the operation timed out.
<i>MSG_RESET</i>	in case of a receive error.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

7.2.9.24.6.16 `uartAcquireBus()`

```
void uartAcquireBus (
    UARTDriver * uartp)
```

Gains exclusive access to the UART bus.

This function tries to gain ownership to the UART bus, if the bus is already being used then the invoking thread is queued.

Precondition

In order to use this function the option `UART_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>uartp</i>	pointer to the <code>UARTDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.24.6.17 uartReleaseBus()**

```
void uartReleaseBus (  
    UARTDriver * uartp)
```

Releases exclusive access to the UART bus.

Precondition

In order to use this function the option `UART_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>uartp</i>	pointer to the <code>UARTDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.24.6.18 uart_lld_init()

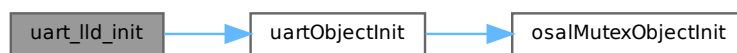
```
void uart_lld_init (  
    void )
```

Low level UART driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.24.6.19 uart_lld_start()

```
void uart_lld_start (  
    UARTDriver * uartp)
```

Configures and activates the UART peripheral.

Parameters

in	<i>uartp</i>	pointer to the <code>UARTDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.6.20 uart_lld_stop()

```
void uart_lld_stop (  
    UARTDriver * uartp)
```

Deactivates the UART peripheral.

Parameters

in	<i>uartp</i>	pointer to the <code>UARTDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.6.21 uart_lld_start_send()

```
void uart_lld_start_send (
    UARTDriver * uartp,
    size_t n,
    const void * txbuf)
```

Starts a transmission on the UART peripheral.

Note

The buffers are organized as uint8_t arrays for data sizes below or equal to 8 bits else it is organized as uint16_t arrays.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object
in	<i>n</i>	number of data frames to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.6.22 uart_lld_stop_send()

```
size_t uart_lld_stop_send (
    UARTDriver * uartp)
```

Stops any ongoing transmission.

Note

Stopping a transmission also suppresses the transmission callbacks.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Returns

The number of data frames not transmitted by the stopped transmit operation.

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.6.23 uart_lld_start_receive()

```
void uart_lld_start_receive (
    UARTDriver * uartp,
    size_t n,
    void * rxbuf)
```

Starts a receive operation on the UART peripheral.

Note

The buffers are organized as uint8_t arrays for data sizes below or equal to 8 bits else it is organized as uint16_t arrays.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object
in	<i>n</i>	number of data frames to send
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.6.24 uart_lld_stop_receive()

```
size_t uart_lld_stop_receive (  
    UARTDriver * uartp)
```

Stops any ongoing receive operation.

Note

Stopping a receive operation also suppresses the receive callbacks.

Parameters

in	<i>uartp</i>	pointer to the UARTDriver object

Returns

The number of data frames not received by the stopped receive operation.

Function Class:

Not an API, this function is for internal use only.

7.2.9.24.7 Variable Documentation**7.2.9.24.7.1 UARTD1**

[UARTDriver](#) UARTD1

UART1 driver identifier.

7.2.9.25 USB Driver

Generic USB Driver.

7.2.9.25.1 Detailed Description

Generic USB Driver.

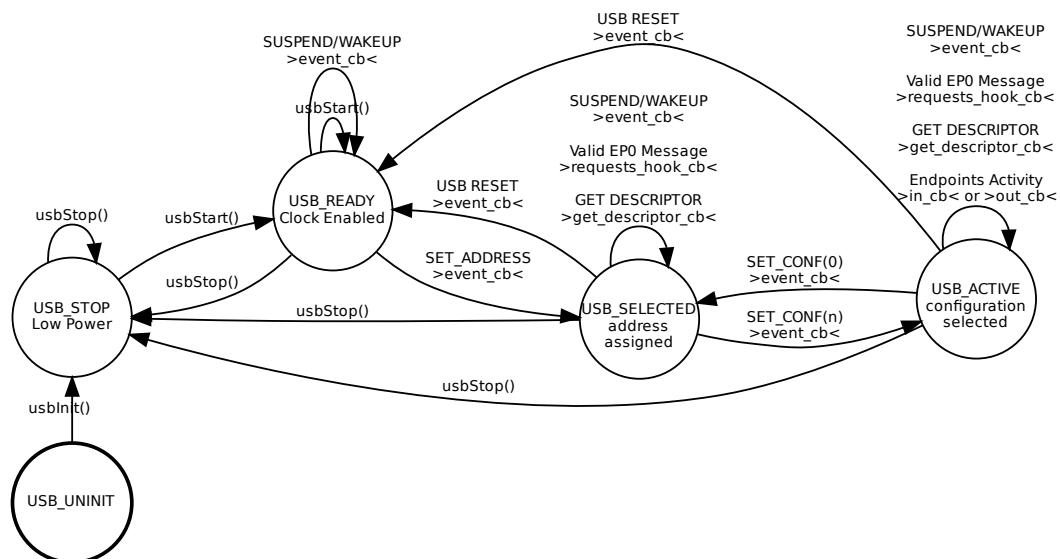
This module implements a generic USB (Universal Serial Bus) driver supporting device-mode operations.

Precondition

In order to use the USB driver the `HAL_USE_USB` option must be enabled in `halconf.h`.

7.2.9.25.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



7.2.9.25.3 USB Operations

The USB driver is quite complex and USB is complex in itself, it is recommended to study the USB specification before trying to use the driver.

7.2.9.25.3.1 USB Implementation

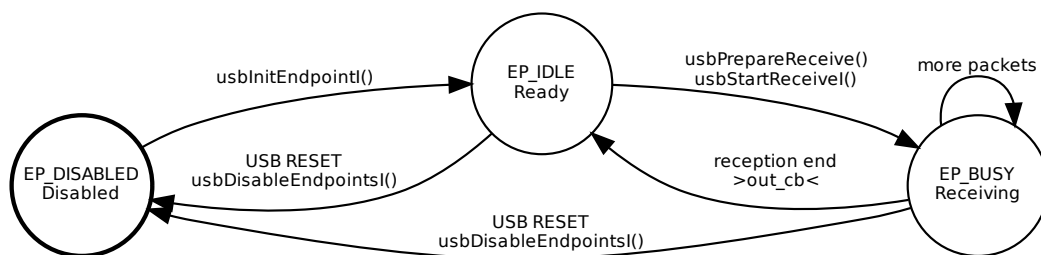
The USB driver abstracts the inner details of the underlying USB hardware. The driver works asynchronously and communicates with the application using callbacks. The application is responsible of the descriptors and strings required by the USB device class to be implemented and of the handling of the specific messages sent over the endpoint zero. Standard messages are handled internally to the driver. The application can use hooks in order to handle custom messages or override the handling of the default handling of standard messages.

7.2.9.25.3.2 USB Endpoints

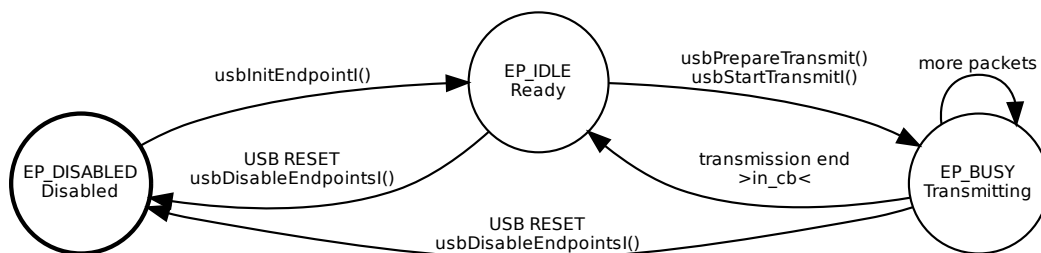
USB endpoints are the objects that the application uses to exchange data with the host. There are two kind of endpoints:

- **IN** endpoints are used by the application to transmit data to the host.
- **OUT** endpoints are used by the application to receive data from the host.

The driver invokes a callback after finishing an IN or OUT transaction. States diagram for OUT endpoints in transaction mode:



States diagram for IN endpoints in transaction mode:



7.2.9.25.3.3 USB Callbacks

The USB driver uses callbacks in order to interact with the application. There are several kinds of callbacks to be handled:

- Driver events callback. As example errors, suspend event, reset event etc.
- Messages Hook callback. This hook allows the application to implement handling of custom messages or to override the default handling of standard messages on endpoint zero.
- Descriptor Requested callback. When the driver endpoint zero handler receives a GET_DESCRIPTOR message and needs to send a descriptor to the host it queries the application using this callback.
- Start of Frame callback. This callback is invoked each time a SOF packet is received.
- Endpoint callbacks. Each endpoint informs the application about I/O conditions using those callbacks.

Helper macros for USB descriptors

- `#define USB_DESC_INDEX(i)`
Helper macro for index values into descriptor strings.
- `#define USB_DESC_BYTE(b)`
Helper macro for byte values into descriptor strings.
- `#define USB_DESC_WORD(w)`
Helper macro for word values into descriptor strings.
- `#define USB_DESC_BCD(bcd)`
Helper macro for BCD values into descriptor strings.
- `#define USB_DESC_DEVICE_SIZE 18U`
- `#define USB_DESC_DEVICE(bcdUSB, bDeviceClass, bDeviceSubClass, bDeviceProtocol, bMaxPacketSize, idVendor, idProduct, bcdDevice, iManufacturer, iProduct, iSerialNumber, bNumConfigurations)`
Device Descriptor helper macro.
- `#define USB_DESC_CONFIGURATION_SIZE 9U`
Configuration Descriptor size.
- `#define USB_DESC_CONFIGURATION(wTotalLength, bNumInterfaces, bConfigurationValue, iConfiguration, bmAttributes, bMaxPower)`
Configuration Descriptor helper macro.
- `#define USB_DESC_INTERFACE_SIZE 9U`
Interface Descriptor size.
- `#define USB_DESC_INTERFACE(bInterfaceNumber, bAlternateSetting, bNumEndpoints, bInterfaceClass, bInterfaceSubClass, bInterfaceProtocol, iInterface)`
Interface Descriptor helper macro.
- `#define USB_DESC_INTERFACE_ASSOCIATION_SIZE 8U`
Interface Association Descriptor size.
- `#define USB_DESC_INTERFACE_ASSOCIATION(bFirstInterface, bInterfaceCount, bFunctionClass, bFunctionSubClass, bFunctionProtocol, iInterface)`
Interface Association Descriptor helper macro.
- `#define USB_DESC_ENDPOINT_SIZE 7U`
Endpoint Descriptor size.
- `#define USB_DESC_ENDPOINT(bEndpointAddress, bmAttributes, wMaxPacketSize, bInterval)`
Endpoint Descriptor helper macro.

Endpoint types and settings

- #define `USB_EP_MODE_TYPE` 0x0003U
- #define `USB_EP_MODE_TYPE_CTRL` 0x0000U
- #define `USB_EP_MODE_TYPE_ISOC` 0x0001U
- #define `USB_EP_MODE_TYPE_BULK` 0x0002U
- #define `USB_EP_MODE_TYPE_INTR` 0x0003U

Macro Functions

- #define `usbGetDriverStateI(usbp)`
Returns the driver state.
- #define `usbConnectBus(usbp)`
Connects the USB device.
- #define `usbDisconnectBus(usbp)`
Disconnect the USB device.
- #define `usbGetFrameNumberX(usbp)`
Returns the current frame number.
- #define `usbGetTransmitStatusI(usbp, ep)`
Returns the status of an IN endpoint.
- #define `usbGetReceiveStatusI(usbp, ep)`
Returns the status of an OUT endpoint.
- #define `usbGetReceiveTransactionSizeX(usbp, ep)`
Returns the exact size of a receive transaction.
- #define `usbSetupTransfer(usbp, buf, n, endcb)`
Request transfer setup.
- #define `usbReadSetup(usbp, ep, buf)`
Reads a setup packet from the dedicated packet buffer.

Low level driver helper macros

- #define `_usb_isr_invoke_event_cb(usbp, evt)`
Common ISR code, usb event callback.
- #define `_usb_isr_invoke_sof_cb(usbp)`
Common ISR code, SOF callback.
- #define `_usb_isr_invoke_setup_cb(usbp, ep)`
Common ISR code, setup packet callback.
- #define `_usb_isr_invoke_in_cb(usbp, ep)`
Common ISR code, IN endpoint callback.
- #define `_usb_isr_invoke_out_cb(usbp, ep)`
Common ISR code, OUT endpoint event.

PLATFORM configuration options

- #define `PLATFORM_USB_USE_USB1` FALSE
USB driver enable switch.

Data Structures

- struct [USBDescriptor](#)
Type of an USB descriptor.
- struct [USBInEndpointState](#)
Type of an IN endpoint state structure.
- struct [USBOutEndpointState](#)
Type of an OUT endpoint state structure.
- struct [USBEndpointConfig](#)
Type of an USB endpoint configuration structure.
- struct [USBConfig](#)
Type of an USB driver configuration structure.
- struct [USBDriver](#)
Structure representing an USB driver.

Macros

- #define [USB_ENDPOINT_OUT](#)(ep)
- #define [USB_ENDPOINT_IN](#)(ep)
- #define [USB_RTYPE_DIR_MASK](#) 0x80U
- #define [USB_RTYPE_DIR_HOST2DEV](#) 0x00U
- #define [USB_RTYPE_DIR_DEV2HOST](#) 0x80U
- #define [USB_RTYPE_TYPE_MASK](#) 0x60U
- #define [USB_RTYPE_TYPE_STD](#) 0x00U
- #define [USB_RTYPE_TYPE_CLASS](#) 0x20U
- #define [USB_RTYPE_TYPE_VENDOR](#) 0x40U
- #define [USB_RTYPE_TYPE_RESERVED](#) 0x60U
- #define [USB_RTYPE_RECIPIENT_MASK](#) 0x1FU
- #define [USB_RTYPE_RECIPIENT_DEVICE](#) 0x00U
- #define [USB_RTYPE_RECIPIENT_INTERFACE](#) 0x01U
- #define [USB_RTYPE_RECIPIENT_ENDPOINT](#) 0x02U
- #define [USB_RTYPE_RECIPIENT_OTHER](#) 0x03U
- #define [USB_REQ_GET_STATUS](#) 0U
- #define [USB_REQ_CLEAR_FEATURE](#) 1U
- #define [USB_REQ_SET_FEATURE](#) 3U
- #define [USB_REQ_SET_ADDRESS](#) 5U
- #define [USB_REQ_GET_DESCRIPTOR](#) 6U
- #define [USB_REQ_SET_DESCRIPTOR](#) 7U
- #define [USB_REQ_GET_CONFIGURATION](#) 8U
- #define [USB_REQ_SET_CONFIGURATION](#) 9U
- #define [USB_REQ_GET_INTERFACE](#) 10U
- #define [USB_REQ_SET_INTERFACE](#) 11U
- #define [USB_REQ_SYNCH_FRAME](#) 12U
- #define [USB_DESCRIPTOR_DEVICE](#) 1U
- #define [USB_DESCRIPTOR_CONFIGURATION](#) 2U
- #define [USB_DESCRIPTOR_STRING](#) 3U
- #define [USB_DESCRIPTOR_INTERFACE](#) 4U
- #define [USB_DESCRIPTOR_ENDPOINT](#) 5U
- #define [USB_DESCRIPTOR_DEVICE_QUALIFIER](#) 6U
- #define [USB_DESCRIPTOR_OTHER_SPEED_CFG](#) 7U
- #define [USB_DESCRIPTOR_INTERFACE_POWER](#) 8U
- #define [USB_DESCRIPTOR_INTERFACE_ASSOCIATION](#) 11U
- #define [USB_FEATURE_ENDPOINT_HALT](#) 0U

- `#define USB_FEATURE_DEVICE_REMOTE_WAKEUP 1U`
- `#define USB_FEATURE_TEST_MODE 2U`
- `#define USB_EARLY_SET_ADDRESS 0`
- `#define USB_LATE_SET_ADDRESS 1`
- `#define USB_EP0_STATUS_STAGE_SW 0`
- `#define USB_EP0_STATUS_STAGE_HW 1`
- `#define USB_SET_ADDRESS_ACK_SW 0`
- `#define USB_SET_ADDRESS_ACK_HW 1`
- `#define USB_IN_STATE 0x08U`
- `#define USB_OUT_STATE 0x10U`
- `#define USB_USE_WAIT FALSE`
Enables synchronous APIs.
- `#define USB_MAX_ENDPOINTS 4`
Maximum endpoint address.
- `#define USB_EP0_STATUS_STAGE USB_EP0_STATUS_STAGE_SW`
Status stage handling method.
- `#define USB_SET_ADDRESS_MODE USB_LATE_SET_ADDRESS`
The address can be changed immediately upon packet reception.
- `#define USB_SET_ADDRESS_ACK_HANDLING USB_SET_ADDRESS_ACK_SW`
Method for set address acknowledge.
- `#define usb_lld_get_frame_number(usbp)`
Returns the current frame number.
- `#define usb_lld_get_transaction_size(usbp, ep)`
Returns the exact size of a receive transaction.
- `#define usb_lld_connect_bus(usbp)`
Connects the USB device.
- `#define usb_lld_disconnect_bus(usbp)`
Disconnect the USB device.
- `#define usb_lld_wakeup_host(usbp)`
Start of host wake-up procedure.

Typedefs

- `typedef struct USBDriver USBDriver`
Type of a structure representing an USB driver.
- `typedef uint8_t usbep_t`
Type of an endpoint identifier.
- `typedef void(* usbcallback_t) (USBDriver *usbp)`
Type of an USB generic notification callback.
- `typedef void(* usbepcallback_t) (USBDriver *usbp, usbep_t ep)`
Type of an USB endpoint callback.
- `typedef void(* usbeventcb_t) (USBDriver *usbp, usbevent_t event)`
Type of an USB event notification callback.
- `typedef bool(* usbreqhandler_t) (USBDriver *usbp)`
Type of a requests handler callback.
- `typedef const USBDescriptor *(* usbgetdescriptor_t) (USBDriver *usbp, uint8_t dtype, uint8_t dindex, uint16_t lang)`
Type of an USB descriptor-retrieving callback.

Enumerations

- enum `usbstate_t` {
`USB_UNINIT` = 0 , `USB_STOP` = 1 , `USB_READY` = 2 , `USB_SELECTED` = 3 ,
`USB_ACTIVE` = 4 , `USB_SUSPENDED` = 5 }
Type of a driver state machine possible states.
- enum `usbepstatus_t` { `EP_STATUS_DISABLED` = 0 , `EP_STATUS_STALLED` = 1 , `EP_STATUS_ACTIVE` = 2 }
Type of an endpoint status.
- enum `usbep0state_t` {
`USB_EP0_STP_WAITING` = 0U , `USB_EP0_IN_TX` = `USB_IN_STATE` | 1U , `USB_EP0_IN_WAITING_TX0`
= `USB_IN_STATE` | 2U , `USB_EP0_IN_SENDING_STS` = `USB_IN_STATE` | 3U ,
`USB_EP0_OUT_WAITING_STS` = `USB_OUT_STATE` | 4U , `USB_EP0_OUT_RX` = `USB_OUT_STATE` | 5U
, `USB_EP0_ERROR` = 6U }
Type of an endpoint zero state machine states.
- enum `usbevent_t` {
`USB_EVENT_RESET` = 0 , `USB_EVENT_ADDRESS` = 1 , `USB_EVENT_CONFIGURED` = 2 ,
`USB_EVENT_UNCONFIGURED` = 3 ,
`USB_EVENT_SUSPEND` = 4 , `USB_EVENT_WAKEUP` = 5 , `USB_EVENT_STALLED` = 6 }
Type of an enumeration of the possible USB events.

Functions

- static uint16_t `get_hword` (uint8_t *p)
- static void `set_address` (USBDriver *usbp)
SET ADDRESS transaction callback.
- static bool `default_handler` (USBDriver *usbp)
Standard requests handler.
- void `usbInit` (void)
USB Driver initialization.
- void `usbObjectInit` (USBDriver *usbp)
Initializes the standard part of a USBDriver structure.
- msg_t `usbStart` (USBDriver *usbp, const USBConfig *config)
Configures and activates the USB peripheral.
- void `usbStop` (USBDriver *usbp)
Deactivates the USB peripheral.
- void `usbInitEndpointI` (USBDriver *usbp, usbep_t ep, const USBEndpointConfig *epcp)
Enables an endpoint.
- void `usbDisableEndpointsI` (USBDriver *usbp)
Disables all the active endpoints.
- void `usbStartReceiveI` (USBDriver *usbp, usbep_t ep, uint8_t *buf, size_t n)
Starts a receive transaction on an OUT endpoint.
- void `usbStartTransmitI` (USBDriver *usbp, usbep_t ep, const uint8_t *buf, size_t n)
Starts a transmit transaction on an IN endpoint.
- msg_t `usbReceive` (USBDriver *usbp, usbep_t ep, uint8_t *buf, size_t n)
Performs a receive transaction on an OUT endpoint.
- msg_t `usbTransmit` (USBDriver *usbp, usbep_t ep, const uint8_t *buf, size_t n)
Performs a transmit transaction on an IN endpoint.
- bool `usbStallReceiveI` (USBDriver *usbp, usbep_t ep)
Stalls an OUT endpoint.
- bool `usbStallTransmitI` (USBDriver *usbp, usbep_t ep)
Stalls an IN endpoint.

- void `usbWakeupHost` (`USBDriver *usbp`)
Host wake-up procedure.
- void `_usb_reset` (`USBDriver *usbp`)
USB reset routine.
- void `_usb_suspend` (`USBDriver *usbp`)
USB suspend routine.
- void `_usb_wakeup` (`USBDriver *usbp`)
USB wake-up routine.
- void `_usb_ep0setup` (`USBDriver *usbp, usbep_t ep`)
Default EP0 SETUP callback.
- void `_usb_ep0in` (`USBDriver *usbp, usbep_t ep`)
Default EP0 IN callback.
- void `_usb_ep0out` (`USBDriver *usbp, usbep_t ep`)
Default EP0 OUT callback.
- void `usbReadSetupI` (`USBDriver *usbp, usbep_t ep, uint8_t *buf`)
- void `usb_llc_init` (void)
Low level USB driver initialization.
- void `usb_llc_start` (`USBDriver *usbp`)
Configures and activates the USB peripheral.
- void `usb_llc_stop` (`USBDriver *usbp`)
Deactivates the USB peripheral.
- void `usb_llc_reset` (`USBDriver *usbp`)
USB low level reset routine.
- void `usb_llc_set_address` (`USBDriver *usbp`)
Sets the USB address.
- void `usb_llc_init_endpoint` (`USBDriver *usbp, usbep_t ep`)
Enables an endpoint.
- void `usb_llc_disable_endpoints` (`USBDriver *usbp`)
Disables all the active endpoints except the endpoint zero.
- `usbepstatus_t` `usb_llc_get_status_out` (`USBDriver *usbp, usbep_t ep`)
Returns the status of an OUT endpoint.
- `usbepstatus_t` `usb_llc_get_status_in` (`USBDriver *usbp, usbep_t ep`)
Returns the status of an IN endpoint.
- void `usb_llc_read_setup` (`USBDriver *usbp, usbep_t ep, uint8_t *buf`)
Reads a setup packet from the dedicated packet buffer.
- void `usb_llc_prepare_receive` (`USBDriver *usbp, usbep_t ep`)
Prepares for a receive operation.
- void `usb_llc_prepare_transmit` (`USBDriver *usbp, usbep_t ep`)
Prepares for a transmit operation.
- void `usb_llc_start_out` (`USBDriver *usbp, usbep_t ep`)
Starts a receive operation on an OUT endpoint.
- void `usb_llc_start_in` (`USBDriver *usbp, usbep_t ep`)
Starts a transmit operation on an IN endpoint.
- void `usb_llc_stall_out` (`USBDriver *usbp, usbep_t ep`)
Brings an OUT endpoint in the stalled state.
- void `usb_llc_stall_in` (`USBDriver *usbp, usbep_t ep`)
Brings an IN endpoint in the stalled state.
- void `usb_llc_clear_out` (`USBDriver *usbp, usbep_t ep`)
Brings an OUT endpoint in the active state.
- void `usb_llc_clear_in` (`USBDriver *usbp, usbep_t ep`)
Brings an IN endpoint in the active state.

Variables

- static const uint8_t `zero_status` [] = {0x00, 0x00}
- static const uint8_t `active_status` [] = {0x00, 0x00}
- static const uint8_t `halted_status` [] = {0x01, 0x00}
- `USBDriver USBD1`
USB1 driver identifier.
- union {
 `USBInEndpointState in`
 IN EP0 state.
 `USBOutEndpointState out`
 OUT EP0 state.
 } `ep0_state`

 EP0 state.
- static const `USBEndpointConfig ep0config`
 EP0 initialization structure.

7.2.9.25.4 Macro Definition Documentation**7.2.9.25.4.1 USB_ENDPOINT_OUT**

```
#define USB_ENDPOINT_OUT(  
    ep)
```

Value:

```
(ep)
```

7.2.9.25.4.2 USB_ENDPOINT_IN

```
#define USB_ENDPOINT_IN(  
    ep)
```

Value:

```
((ep) | 0x80U)
```

7.2.9.25.4.3 USB_RTYPE_DIR_MASK

```
#define USB_RTYPE_DIR_MASK 0x80U
```

7.2.9.25.4.4 USB_RTYPE_DIR_HOST2DEV

```
#define USB_RTYPE_DIR_HOST2DEV 0x00U
```

7.2.9.25.4.5 USB_RTYPE_DIR_DEV2HOST

```
#define USB_RTYPE_DIR_DEV2HOST 0x80U
```

7.2.9.25.4.6 USB_RTYPE_TYPE_MASK

```
#define USB_RTYPE_TYPE_MASK 0x60U
```

7.2.9.25.4.7 USB_RTYPE_TYPE_STD

```
#define USB_RTYPE_TYPE_STD 0x00U
```

7.2.9.25.4.8 USB_RTYPE_TYPE_CLASS

```
#define USB_RTYPE_TYPE_CLASS 0x20U
```

7.2.9.25.4.9 USB_RTYPE_TYPE_VENDOR

```
#define USB_RTYPE_TYPE_VENDOR 0x40U
```

7.2.9.25.4.10 USB_RTYPE_TYPE_RESERVED

```
#define USB_RTYPE_TYPE_RESERVED 0x60U
```

7.2.9.25.4.11 USB_RTYPE_RECIPIENT_MASK

```
#define USB_RTYPE_RECIPIENT_MASK 0x1FU
```

7.2.9.25.4.12 USB_RTYPE_RECIPIENT_DEVICE

```
#define USB_RTYPE_RECIPIENT_DEVICE 0x00U
```

7.2.9.25.4.13 USB_RTYPE_RECIPIENT_INTERFACE

```
#define USB_RTYPE_RECIPIENT_INTERFACE 0x01U
```

7.2.9.25.4.14 USB_RTYPE_RECIPIENT_ENDPOINT

```
#define USB_RTYPE_RECIPIENT_ENDPOINT 0x02U
```

7.2.9.25.4.15 USB_RTYPE_RECIPIENT_OTHER

```
#define USB_RTYPE_RECIPIENT_OTHER 0x03U
```

7.2.9.25.4.16 USB_REQ_GET_STATUS

```
#define USB_REQ_GET_STATUS 0U
```

7.2.9.25.4.17 USB_REQ_CLEAR_FEATURE

```
#define USB_REQ_CLEAR_FEATURE 1U
```

7.2.9.25.4.18 USB_REQ_SET_FEATURE

```
#define USB_REQ_SET_FEATURE 3U
```

7.2.9.25.4.19 USB_REQ_SET_ADDRESS

```
#define USB_REQ_SET_ADDRESS 5U
```

7.2.9.25.4.20 USB_REQ_GET_DESCRIPTOR

```
#define USB_REQ_GET_DESCRIPTOR 6U
```

7.2.9.25.4.21 USB_REQ_SET_DESCRIPTOR

```
#define USB_REQ_SET_DESCRIPTOR 7U
```

7.2.9.25.4.22 USB_REQ_GET_CONFIGURATION

```
#define USB_REQ_GET_CONFIGURATION 8U
```

7.2.9.25.4.23 USB_REQ_SET_CONFIGURATION

```
#define USB_REQ_SET_CONFIGURATION 9U
```

7.2.9.25.4.24 USB_REQ_GET_INTERFACE

```
#define USB_REQ_GET_INTERFACE 10U
```

7.2.9.25.4.25 USB_REQ_SET_INTERFACE

```
#define USB_REQ_SET_INTERFACE 11U
```

7.2.9.25.4.26 USB_REQ_SYNCH_FRAME

```
#define USB_REQ_SYNCH_FRAME 12U
```

7.2.9.25.4.27 USB_DESCRIPTOR_DEVICE

```
#define USB_DESCRIPTOR_DEVICE 1U
```

7.2.9.25.4.28 USB_DESCRIPTOR_CONFIGURATION

```
#define USB_DESCRIPTOR_CONFIGURATION 2U
```

7.2.9.25.4.29 USB_DESCRIPTOR_STRING

```
#define USB_DESCRIPTOR_STRING 3U
```

7.2.9.25.4.30 USB_DESCRIPTOR_INTERFACE

```
#define USB_DESCRIPTOR_INTERFACE 4U
```

7.2.9.25.4.31 USB_DESCRIPTOR_ENDPOINT

```
#define USB_DESCRIPTOR_ENDPOINT 5U
```

7.2.9.25.4.32 USB_DESCRIPTOR_DEVICE_QUALIFIER

```
#define USB_DESCRIPTOR_DEVICE_QUALIFIER 6U
```

7.2.9.25.4.33 USB_DESCRIPTOR_OTHER_SPEED_CFG

```
#define USB_DESCRIPTOR_OTHER_SPEED_CFG 7U
```

7.2.9.25.4.34 USB_DESCRIPTOR_INTERFACE_POWER

```
#define USB_DESCRIPTOR_INTERFACE_POWER 8U
```

7.2.9.25.4.35 USB_DESCRIPTOR_INTERFACE_ASSOCIATION

```
#define USB_DESCRIPTOR_INTERFACE_ASSOCIATION 11U
```

7.2.9.25.4.36 USB_FEATURE_ENDPOINT_HALT

```
#define USB_FEATURE_ENDPOINT_HALT 0U
```

7.2.9.25.4.37 USB_FEATURE_DEVICE_REMOTE_WAKEUP

```
#define USB_FEATURE_DEVICE_REMOTE_WAKEUP 1U
```

7.2.9.25.4.38 USB_FEATURE_TEST_MODE

```
#define USB_FEATURE_TEST_MODE 2U
```

7.2.9.25.4.39 USB_EARLY_SET_ADDRESS

```
#define USB_EARLY_SET_ADDRESS 0
```

7.2.9.25.4.40 USB_LATE_SET_ADDRESS

```
#define USB_LATE_SET_ADDRESS 1
```

7.2.9.25.4.41 USB_EP0_STATUS_STAGE_SW

```
#define USB_EP0_STATUS_STAGE_SW 0
```

7.2.9.25.4.42 USB_EP0_STATUS_STAGE_HW

```
#define USB_EP0_STATUS_STAGE_HW 1
```

7.2.9.25.4.43 USB_SET_ADDRESS_ACK_SW

```
#define USB_SET_ADDRESS_ACK_SW 0
```

7.2.9.25.4.44 USB_SET_ADDRESS_ACK_HW

```
#define USB_SET_ADDRESS_ACK_HW 1
```

7.2.9.25.4.45 USB_DESC_INDEX

```
#define USB_DESC_INDEX(  
    i)
```

Value:

```
((uint8_t)(i))
```

Helper macro for index values into descriptor strings.

7.2.9.25.4.46 USB_DESC_BYTE

```
#define USB_DESC_BYTE(  
    b)
```

Value:

```
((uint8_t) (b))
```

Helper macro for byte values into descriptor strings.

7.2.9.25.4.47 USB_DESC_WORD

```
#define USB_DESC_WORD(  
    w)
```

Value:

```
(uint8_t)((w) & 255U),  
(uint8_t)((w) >> 8) & 255U)
```

Helper macro for word values into descriptor strings.

7.2.9.25.4.48 USB_DESC_BCD

```
#define USB_DESC_BCD(  
    bcd)
```

Value:

```
(uint8_t)((bcd) & 255U),  
(uint8_t)((bcd) >> 8) & 255)
```

Helper macro for BCD values into descriptor strings.

7.2.9.25.4.49 USB_DESC_DEVICE_SIZE

```
#define USB_DESC_DEVICE_SIZE 18U
```

7.2.9.25.4.50 USB_DESC_DEVICE

```
#define USB_DESC_DEVICE(  
    bcdUSB,  
    bDeviceClass,  
    bDeviceSubClass,  
    bDeviceProtocol,  
    bMaxPacketSize,  
    idVendor,  
    idProduct,  
    bcdDevice,  
    iManufacturer,  
    iProduct,  
    iSerialNumber,  
    bNumConfigurations)
```

Value:

```

USB_DESC_BYTE(USB_DESC_DEVICE_SIZE),
USB_DESC_BYTE(USB_DESCRIPTOR_DEVICE),
USB_DESC_BCD(bcdUSB),
USB_DESC_BYTE(bDeviceClass),
USB_DESC_BYTE(bDeviceSubClass),
USB_DESC_BYTE(bDeviceProtocol),
USB_DESC_BYTE(bMaxPacketSize),
USB_DESC_WORD(idVendor),
USB_DESC_WORD(idProduct),
USB_DESC_BCD(bcdDevice),
USB_DESC_INDEX(iManufacturer),
USB_DESC_INDEX(iProduct),
USB_DESC_INDEX(iSerialNumber),
USB_DESC_BYTE(bNumConfigurations)

```

```

/
/
/
/
/
/
/
/
/
/
/
/

```

Device Descriptor helper macro.

7.2.9.25.4.51 USB_DESC_CONFIGURATION_SIZE

```
#define USB_DESC_CONFIGURATION_SIZE 9U
```

Configuration Descriptor size.

7.2.9.25.4.52 USB_DESC_CONFIGURATION

```

#define USB_DESC_CONFIGURATION(
    wTotalLength,
    bNumInterfaces,
    bConfigurationValue,
    iConfiguration,
    bmAttributes,
    bMaxPower)

```

Value:

```

USB_DESC_BYTE(USB_DESC_CONFIGURATION_SIZE),
USB_DESC_BYTE(USB_DESCRIPTOR_CONFIGURATION),
USB_DESC_WORD(wTotalLength),
USB_DESC_BYTE(bNumInterfaces),
USB_DESC_BYTE(bConfigurationValue),
USB_DESC_INDEX(iConfiguration),
USB_DESC_BYTE(bmAttributes),
USB_DESC_BYTE(bMaxPower)

```

```

/
/
/
/
/
/
/
/

```

Configuration Descriptor helper macro.

7.2.9.25.4.53 USB_DESC_INTERFACE_SIZE

```
#define USB_DESC_INTERFACE_SIZE 9U
```

Interface Descriptor size.

7.2.9.25.4.54 USB_DESC_INTERFACE

```
#define USB_DESC_INTERFACE(
    bInterfaceNumber,
    bAlternateSetting,
    bNumEndpoints,
    bInterfaceClass,
    bInterfaceSubClass,
    bInterfaceProtocol,
    iInterface)
```

Value:

```
USB_DESC_BYTE(USB_DESC_INTERFACE_SIZE),
USB_DESC_BYTE(USB_DESCRIPTOR_INTERFACE),
USB_DESC_BYTE(bInterfaceNumber),
USB_DESC_BYTE(bAlternateSetting),
USB_DESC_BYTE(bNumEndpoints),
USB_DESC_BYTE(bInterfaceClass),
USB_DESC_BYTE(bInterfaceSubClass),
USB_DESC_BYTE(bInterfaceProtocol),
USB_DESC_INDEX(iInterface)
```

```

/
/
/
/
/
/
/
/
```

Interface Descriptor helper macro.

7.2.9.25.4.55 USB_DESC_INTERFACE_ASSOCIATION_SIZE

```
#define USB_DESC_INTERFACE_ASSOCIATION_SIZE 8U
```

Interface Association Descriptor size.

7.2.9.25.4.56 USB_DESC_INTERFACE_ASSOCIATION

```
#define USB_DESC_INTERFACE_ASSOCIATION(
    bFirstInterface,
    bInterfaceCount,
    bFunctionClass,
    bFunctionSubClass,
    bFunctionProtocol,
    iInterface)
```

Value:

```
USB_DESC_BYTE(USB_DESC_INTERFACE_ASSOCIATION_SIZE),
USB_DESC_BYTE(USB_DESCRIPTOR_INTERFACE_ASSOCIATION),
USB_DESC_BYTE(bFirstInterface),
USB_DESC_BYTE(bInterfaceCount),
USB_DESC_BYTE(bFunctionClass),
USB_DESC_BYTE(bFunctionSubClass),
USB_DESC_BYTE(bFunctionProtocol),
USB_DESC_INDEX(iInterface)
```

```

/
/
/
/
/
/
/
/
```

Interface Association Descriptor helper macro.

7.2.9.25.4.57 USB_DESC_ENDPOINT_SIZE

```
#define USB_DESC_ENDPOINT_SIZE 7U
```

Endpoint Descriptor size.

7.2.9.25.4.58 USB_DESC_ENDPOINT

```
#define USB_DESC_ENDPOINT(
    bEndpointAddress,
    bmAttributes,
    wMaxPacketSize,
    bInterval)
```

Value:

```
USB_DESC_BYTE(USB_DESC_ENDPOINT_SIZE),
USB_DESC_BYTE(USB_DESCRIPTOR_ENDPOINT),
USB_DESC_BYTE(bEndpointAddress),
USB_DESC_BYTE(bmAttributes),
USB_DESC_WORD(wMaxPacketSize),
USB_DESC_BYTE(bInterval)
```

```

/
/
/
/
/

```

Endpoint Descriptor helper macro.

7.2.9.25.4.59 USB_EP_MODE_TYPE

```
#define USB_EP_MODE_TYPE 0x0003U
```

Endpoint type mask.

7.2.9.25.4.60 USB_EP_MODE_TYPE_CTRL

```
#define USB_EP_MODE_TYPE_CTRL 0x0000U
```

Control endpoint.

7.2.9.25.4.61 USB_EP_MODE_TYPE_ISOC

```
#define USB_EP_MODE_TYPE_ISOC 0x0001U
```

Isochronous endpoint.

7.2.9.25.4.62 USB_EP_MODE_TYPE_BULK

```
#define USB_EP_MODE_TYPE_BULK 0x0002U
```

Bulk endpoint.

7.2.9.25.4.63 USB_EP_MODE_TYPE_INTR

```
#define USB_EP_MODE_TYPE_INTR 0x0003U
```

Interrupt endpoint.

7.2.9.25.4.64 USB_IN_STATE

```
#define USB_IN_STATE 0x08U
```

7.2.9.25.4.65 USB_OUT_STATE

```
#define USB_OUT_STATE 0x10U
```

7.2.9.25.4.66 USB_USE_WAIT

```
#define USB_USE_WAIT FALSE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.9.25.4.67 usbGetDriverStateI

```
#define usbGetDriverStateI(  
    usbp)
```

Value:

```
((usbp)->state)
```

Returns the driver state.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Returns

The driver state.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.25.4.68 usbConnectBus

```
#define usbConnectBus(  
    usbp)
```

Value:

```
usb_lld_connect_bus(usbp)
```

Connects the USB device.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.25.4.69 usbDisconnectBus

```
#define usbDisconnectBus(  
    usbp)
```

Value:

[usb_llc_disconnect_bus](#)(*usbp*)

Disconnect the USB device.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.25.4.70 usbGetFrameNumberX

```
#define usbGetFrameNumberX(  
    usbp)
```

Value:

[usb_llc_get_frame_number](#)(*usbp*)

Returns the current frame number.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Returns

The current frame number.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.25.4.71 usbGetTransmitStatusI

```
#define usbGetTransmitStatusI(  
    usbp,  
    ep)
```

Value:

((*usbp*)->transmitting & (uint16_t)((unsigned)1U << (unsigned)(*ep*))) != 0U

Returns the status of an IN endpoint.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Returns

The operation status.

Return values

<i>false</i>	Endpoint ready.
<i>true</i>	Endpoint transmitting.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.25.4.72 usbGetReceiveStatusI

```
#define usbGetReceiveStatusI(  
    usbp,  
    ep)
```

Value:

```
((usbp)->receiving & (uint16_t)((unsigned)1U << (unsigned)(ep))) != 0U)
```

Returns the status of an OUT endpoint.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Returns

The operation status.

Return values

<i>false</i>	Endpoint ready.
<i>true</i>	Endpoint receiving.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.25.4.73 usbGetReceiveTransactionSizeX

```
#define usbGetReceiveTransactionSizeX(
    usbp,
    ep)
```

Value:

```
usb_lld_get_transaction_size(usbp, ep)
```

Returns the exact size of a receive transaction.

The received size can be different from the size specified in `usbStartReceiveI()` because the last packet could have a size different from the expected one.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number

Returns

Received data size.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.9.25.4.74 usbSetupTransfer

```
#define usbSetupTransfer(
    usbp,
    buf,
    n,
    endcb)
```

Value:

```
{
    (usbp)->ep0next = (buf);
    (usbp)->ep0n    = (n);
    (usbp)->ep0endcb = (endcb);
}
```

Request transfer setup.

This macro is used by the request handling callbacks in order to prepare a transaction over the endpoint zero.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>buf</i>	pointer to a buffer for the transaction data
in	<i>n</i>	number of bytes to be transferred
in	<i>endcb</i>	callback to be invoked after the transfer or <code>NULL</code>

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.25.4.75 usbReadSetup

```
#define usbReadSetup(  
    usbp,  
    ep,  
    buf)
```

Value:

`usb_llc_read_setup(usbp, ep, buf)`

Reads a setup packet from the dedicated packet buffer.

This function must be invoked in the context of the `setup_cb` callback in order to read the received setup packet.

Precondition

In order to use this function the endpoint must have been initialized as a control endpoint.

Note

This function can be invoked both in thread and IRQ context.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number
out	<i>buf</i>	buffer where to copy the packet data

Function Class:

Special function, this function has special requirements see the notes.

7.2.9.25.4.76 _usb_isr_invoke_event_cb

```
#define _usb_isr_invoke_event_cb(  
    usbp,  
    evt)
```

Value:

```
{  
    if (((usbp)->config->event_cb) != NULL) {  
        (usbp)->config->event_cb(usbp, evt);  
    }  
}
```

Common ISR code, usb event callback.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>evt</i>	USB event code

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.4.77 _usb_isr_invoke_sof_cb

```
#define _usb_isr_invoke_sof_cb(  
    usbp)
```

Value:

```
{  
    if ((usbp->config->sof_cb) != NULL) {  
        (usbp->config->sof_cb)(usbp);  
    }  
}
```

Common ISR code, SOF callback.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.4.78 _usb_isr_invoke_setup_cb

```
#define _usb_isr_invoke_setup_cb(  
    usbp,  
    ep)
```

Value:

```
{  
    (usbp->epc[ep]->setup_cb)(usbp, ep);  
}
```

Common ISR code, setup packet callback.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.4.79 _usb_isr_invoke_in_cb

```
#define _usb_isr_invoke_in_cb(  
    usbp,  
    ep)
```

Value:

```
{  
    (usbp->transmitting &= ~(1 << (ep)));  
    if ((usbp->epc[ep]->in_cb) != NULL) {  
        (usbp->epc[ep]->in_cb)(usbp, ep);  
    }  
    osalSysLockFromISR();  
    osalThreadResumeI(&(usbp->epc[ep]->in_state->thread, MSG_OK);  
    osalSysUnlockFromISR();  
}
```

Common ISR code, IN endpoint callback.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.4.80 _usb_isr_invoke_out_cb

```
#define _usb_isr_invoke_out_cb(  
    usbp,  
    ep)
```

Value:

```
{  
    (usbp)->receiving &= ~(1 « (ep)); \  
    if ((usbp)->epc[ep]->out_cb != NULL) {  
        (usbp)->epc[ep]->out_cb(usbp, ep);  
    }  
    osalSysLockFromISR();  
    osalThreadResumeI(&(usbp)->epc[ep]->out_state->thread,  
        usbGetReceiveTransactionSizeX(usbp, ep));  
    osalSysUnlockFromISR();  
}
```

Common ISR code, OUT endpoint event.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.4.81 USB_MAX_ENDPOINTS

```
#define USB_MAX_ENDPOINTS 4
```

Maximum endpoint address.

7.2.9.25.4.82 USB_EP0_STATUS_STAGE

```
#define USB_EP0_STATUS_STAGE USB\_EP0\_STATUS\_STAGE\_SW
```

Status stage handling method.

7.2.9.25.4.83 USB_SET_ADDRESS_MODE

```
#define USB_SET_ADDRESS_MODE USB_LATE_SET_ADDRESS
```

The address can be changed immediately upon packet reception.

7.2.9.25.4.84 USB_SET_ADDRESS_ACK_HANDLING

```
#define USB_SET_ADDRESS_ACK_HANDLING USB_SET_ADDRESS_ACK_SW
```

Method for set address acknowledge.

7.2.9.25.4.85 PLATFORM_USB_USE_USB1

```
#define PLATFORM_USB_USE_USB1 FALSE
```

USB driver enable switch.

If set to `TRUE` the support for USB1 is included.

Note

The default is `FALSE`.

7.2.9.25.4.86 usb_lld_get_frame_number

```
#define usb_lld_get_frame_number(  
    usbp)
```

Value:

0

Returns the current frame number.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Returns

The current frame number.

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.4.87 usb_lld_get_transaction_size

```
#define usb_lld_get_transaction_size(  
    usbp,  
    ep)
```

Value:

```
((usbp)->epc[ep]->out_state->rxcnt)
```

Returns the exact size of a receive transaction.

The received size can be different from the size specified in [usbStartReceiveI\(\)](#) because the last packet could have a size different from the expected one.

Precondition

The OUT endpoint must have been configured in transaction mode in order to use this function.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Returns

Received data size.

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.4.88 usb_lld_connect_bus

```
#define usb_lld_connect_bus(  
    usbp)
```

Connects the USB device.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.25.4.89 usb_lld_disconnect_bus

```
#define usb_lld_disconnect_bus(  
    usbp)
```

Disconnect the USB device.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.25.4.90 usb_lld_wakeup_host

```
#define usb_lld_wakeup_host(  
    usbp)
```

Start of host wake-up procedure.

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.5 Typedef Documentation

7.2.9.25.5.1 USBDriver

```
typedef struct USBDriver USBDriver
```

Type of a structure representing an USB driver.

7.2.9.25.5.2 usbep_t

```
typedef uint8_t usbep_t
```

Type of an endpoint identifier.

7.2.9.25.5.3 usbcallback_t

```
typedef void(* usbcallback_t) (USBDriver *usbp)
```

Type of an USB generic notification callback.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object triggering the callback

7.2.9.25.5.4 usbepcallback_t

```
typedef void(* usbepcallback_t) (USBDriver *usbp, usbep_t ep)
```

Type of an USB endpoint callback.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object triggering the callback
in	<i>ep</i>	endpoint number

7.2.9.25.5.5 usbeventcb_t

```
typedef void(* usbeventcb_t) (USBDriver *usbp, usbevent_t event)
```

Type of an USB event notification callback.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object triggering the callback
in	<i>event</i>	event type

7.2.9.25.5.6 usbreqhandler_t

```
typedef bool(* usbreqhandler_t) (USBDriver *usbp)
```

Type of a requests handler callback.

The request is encoded in the `usb_setup` buffer.

Parameters

<i>in</i>	<i>usbp</i>	pointer to the USBDriver object triggering the callback

Returns

The request handling exit code.

Return values

<i>false</i>	Request not recognized by the handler.
<i>true</i>	Request handled.

7.2.9.25.5.7 usbgetdescriptor_t

```
typedef const USBDescriptor *(* usbgetdescriptor_t) (USBDriver *usbp, uint8_t dtype, uint8_t dindex, uint16_t lang)
```

Type of an USB descriptor-retrieving callback.

7.2.9.25.6 Enumeration Type Documentation**7.2.9.25.6.1 usbstate_t**

```
enum usbstate_t
```

Type of a driver state machine possible states.

Enumerator

USB_UNINIT	Not initialized.
USB_STOP	Stopped.
USB_READY	Ready, after bus reset.
USB_SELECTED	Address assigned.
USB_ACTIVE	Active, configuration selected.
USB_SUSPENDED	Suspended, low power mode.

7.2.9.25.6.2 usbepstatus_t

```
enum usbepstatus_t
```

Type of an endpoint status.

Enumerator

EP_STATUS_DISABLED	Endpoint not active.
EP_STATUS_STALLED	Endpoint opened but stalled.
EP_STATUS_ACTIVE	Active endpoint.

7.2.9.25.6.3 usbep0state_t

```
enum usbep0state_t
```

Type of an endpoint zero state machine states.

Enumerator

USB_EP0_STP_WAITING	Waiting for SETUP data.
USB_EP0_IN_TX	Transmitting.
USB_EP0_IN_WAITING_TX0	Waiting transmit 0.
USB_EP0_IN_SENDING_STS	Sending status.
USB_EP0_OUT_WAITING_STS	Waiting status.
USB_EP0_OUT_RX	Receiving.
USB_EP0_ERROR	Error, EP0 stalled.

7.2.9.25.6.4 `usbevent_t`

```
enum usbevent_t
```

Type of an enumeration of the possible USB events.

Enumerator

USB_EVENT_RESET	Driver has been reset by host.
USB_EVENT_ADDRESS	Address assigned.
USB_EVENT_CONFIGURED	Configuration selected.
USB_EVENT_UNCONFIGURED	Configuration removed.
USB_EVENT_SUSPEND	Entering suspend mode.
USB_EVENT_WAKEUP	Leaving suspend mode.
USB_EVENT_STALLED	Endpoint 0 error, stalled.

7.2.9.25.7 Function Documentation

7.2.9.25.7.1 `get_hword()`

```
uint16_t get_hword (
    uint8_t * p) [static]
```

7.2.9.25.7.2 `set_address()`

```
void set_address (
    USBDriver * usbp) [static]
```

SET ADDRESS transaction callback.

Parameters

in	<code>usbp</code>	pointer to the <code>USBDriver</code> object

Here is the call graph for this function:



7.2.9.25.7.3 default_handler()

```
bool default_handler (  
    USBDriver * usbp) [static]
```

Standard requests handler.

This is the standard requests default handler, most standard requests are handled here, the user can override the standard handling using the `requests_hook_cb` hook in the [USBConfig](#) structure.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

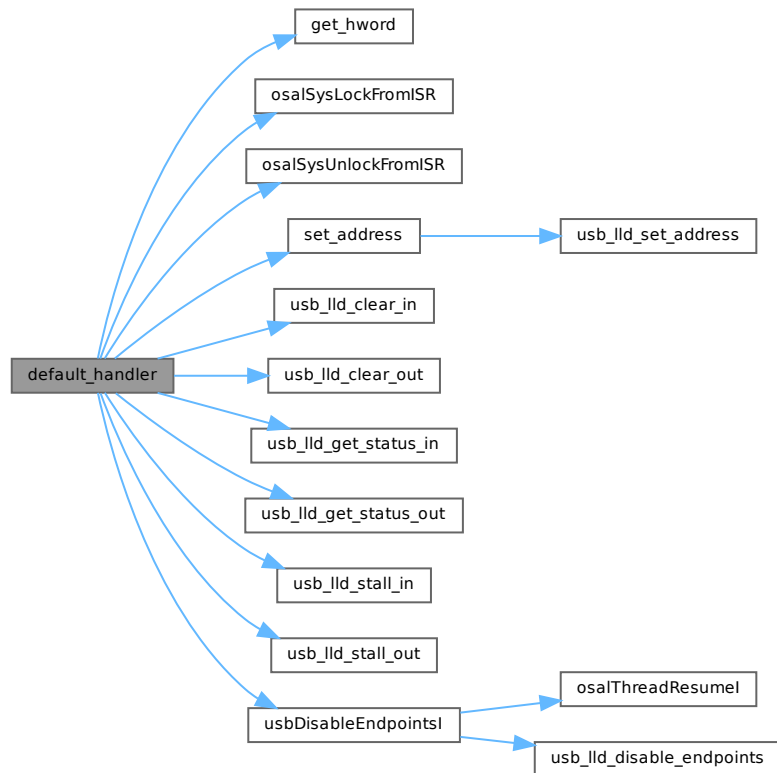
Returns

The request handling exit code.

Return values

<i>false</i>	Request not recognized by the handler or error.
<i>true</i>	Request handled.

Here is the call graph for this function:



7.2.9.25.7.4 usbInit()

```
void usbInit (
    void )
```

USB Driver initialization.

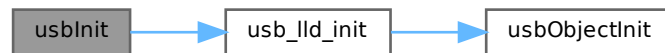
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.25.7.5 usbObjectInit()**

```
void usbObjectInit (  
    USBDriver * usbp)
```

Initializes the standard part of a `USBDriver` structure.

Parameters

out	<i>usbp</i>	pointer to the <code>USBDriver</code> object

Function Class:

Object or module initializer function.

7.2.9.25.7.6 usbStart()

```
msg_t usbStart (  
    USBDriver * usbp,  
    const USBConfig * config)
```

Configures and activates the USB peripheral.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>config</i>	pointer to the <code>USBConfig</code> object

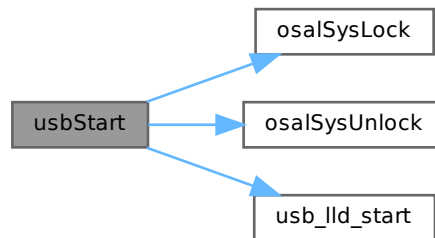
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.25.7.7 usbStop()**

```
void usbStop (
    USBDriver * usbp)
```

Deactivates the USB peripheral.

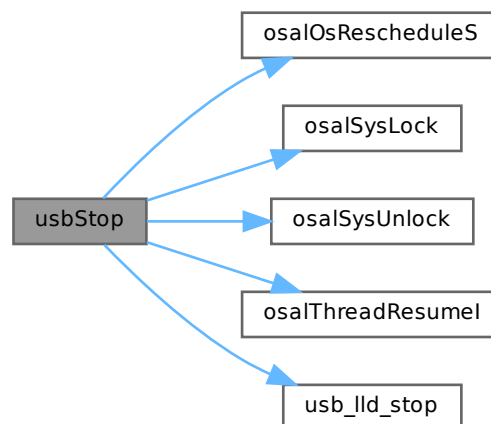
Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.25.7.8 usbInitEndpointI()

```
void usbInitEndpointI (  
    USBDriver * usbp,  
    usbept_t ep,  
    const USBEndpointConfig * epcp)
```

Enables an endpoint.

This function enables an endpoint, both IN and/or OUT directions depending on the configuration structure.

Note

This function must be invoked in response of a SET_CONFIGURATION or SET_INTERFACE message.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number
in	<i>epcp</i>	the endpoint configuration

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.25.7.9 usbDisableEndpointsI()

```
void usbDisableEndpointsI (  
    USBDriver * usbp)
```

Disables all the active endpoints.

This function disables all the active endpoints except the endpoint zero.

Note

This function must be invoked in response of a SET_CONFIGURATION message with configuration number zero.

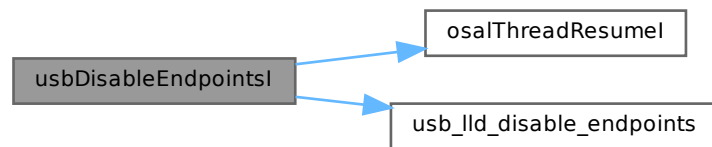
Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.25.7.10 usbStartReceiveI()

```

void usbStartReceiveI (
    USBDriver * usbp,
    usbep_t ep,
    uint8_t * buf,
    size_t n)
  
```

Starts a receive transaction on an OUT endpoint.

Note

This function is meant to be called from ISR context outside critical zones because there is a potentially slow operation inside.

The transaction terminates when one of the following conditions has been met:

- The specified amount of data has been received.
- A short packet has been received.
- A zero-length packet has been received.
- The USB has been reset by host or the driver went into `USB_SUSPENDED` state.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number
out	<i>buf</i>	buffer where to copy the received data
in	<i>n</i>	transaction size. It is recommended a multiple of the packet size because the excess is discarded.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.25.7.11 usbStartTransmitI()**

```

void usbStartTransmitI (
    USBDriver * usbp,
    usbep_t ep,
    const uint8_t * buf,
    size_t n)
  
```

Starts a transmit transaction on an IN endpoint.

Note

This function is meant to be called from ISR context outside critical zones because there is a potentially slow operation inside.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number
in	<i>buf</i>	buffer where to fetch the data to be transmitted
in	<i>n</i>	transaction size

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.9.25.7.12 usbReceive()

```
msg_t usbReceive (
    USBDriver * usbp,
    usbep_t ep,
    uint8_t * buf,
    size_t n)
```

Performs a receive transaction on an OUT endpoint.

Note

The transaction terminates when one of the following conditions has been met:

- The specified amount of data has been received.
- A short packet has been received.
- A zero-length packet has been received.
- The USB has been reset by host or the driver went into `USB_SUSPENDED` state.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number
out	<i>buf</i>	buffer where to copy the received data
in	<i>n</i>	transaction size. It is recommended a multiple of the packet size because the excess is discarded.

Returns

The received effective data size, it can be less than the amount specified.

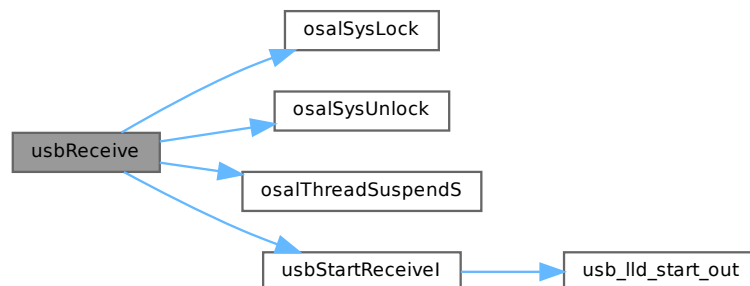
Return values

<i>MSG_RESET</i>	driver not in <code>USB_ACTIVE</code> state or the operation has been aborted by an USB reset or a transition to the <code>USB_SUSPENDED</code> state.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.25.7.13 usbTransmit()

```
msg_t usbTransmit (
    USBDriver * usbp,
    usbep_t ep,
    const uint8_t * buf,
    size_t n)
```

Performs a transmit transaction on an IN endpoint.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number
in	<i>buf</i>	buffer where to fetch the data to be transmitted
in	<i>n</i>	transaction size

Returns

The operation status.

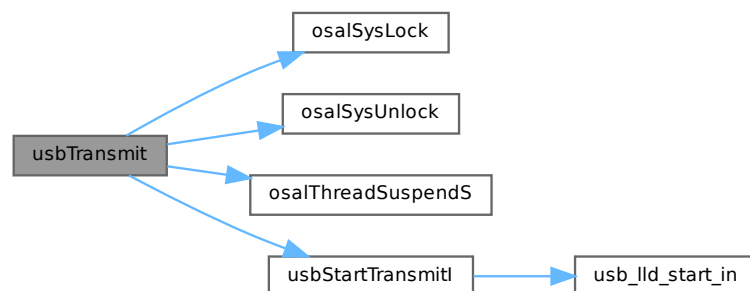
Return values

<i>MSG_OK</i>	operation performed successfully.
<i>MSG_RESET</i>	driver not in <code>USB_ACTIVE</code> state or the operation has been aborted by an USB reset or a transition to the <code>USB_SUSPENDED</code> state.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.25.7.14 usbStallReceive()

```
bool usbStallReceiveI (
    USBDriver * usbp,
    usbep_t ep)
```

Stalls an OUT endpoint.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Returns

The operation status.

Return values

<i>false</i>	Endpoint stalled.
<i>true</i>	Endpoint busy, not stalled.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.25.7.15 usbStallTransmitI()**

```
bool usbStallTransmitI (  
    USBDriver * usbp,  
    usbep\_t ep)
```

Stalls an IN endpoint.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Returns

The operation status.

Return values

<i>false</i>	Endpoint stalled.
<i>true</i>	Endpoint busy, not stalled.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.9.25.7.16 usbWakeupHost()**

```
void usbWakeupHost (  
    USBDriver * usbp)
```

Host wake-up procedure.

Note

It is silently ignored if the USB device is not in the `USB_SUSPENDED` state.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.9.25.7.17 _usb_reset()

```
void _usb_reset (  
    USBDriver * usbp)
```

USB reset routine.

This function must be invoked when an USB bus reset condition is detected.

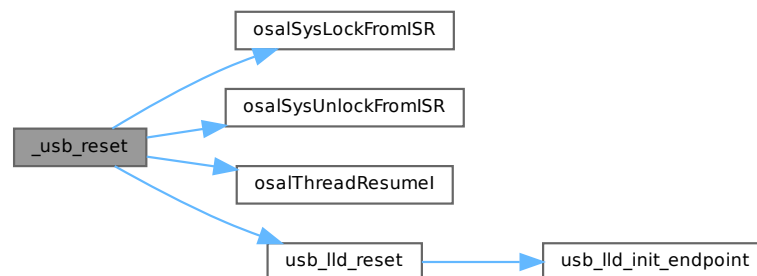
Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.25.7.18 _usb_suspend()**

```
void _usb_suspend (  
    USBDriver * usbp)
```

USB suspend routine.

This function must be invoked when an USB bus suspend condition is detected.

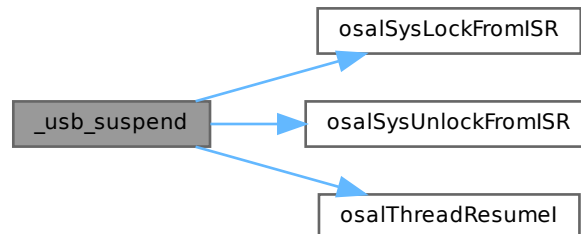
Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.25.7.19 _usb_wakeup()**

```
void _usb_wakeup (
    USBDriver * usbp)
```

USB wake-up routine.

This function must be invoked when an USB bus wake-up condition is detected.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.20 _usb_ep0setup()

```
void _usb_ep0setup (
    USBDriver * usbp,
    usbep_t ep)
```

Default EP0 SETUP callback.

This function is used by the low level driver as default handler for EP0 SETUP events.

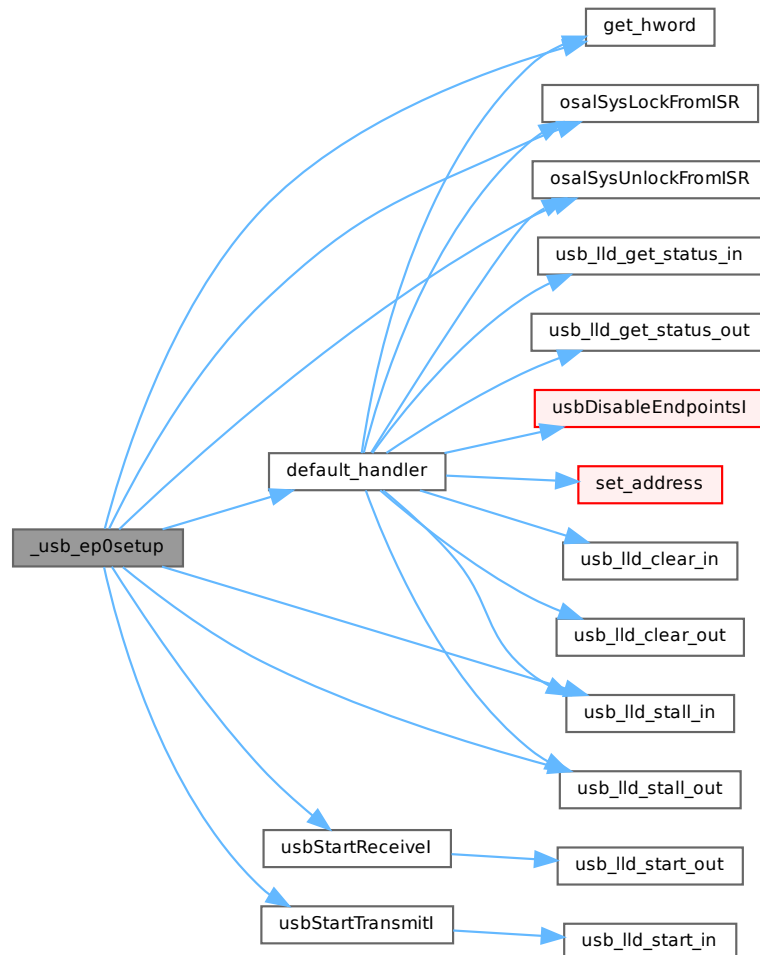
Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number, always zero

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.25.7.21 _usb_ep0in()**

```
void _usb_ep0in (
    USBDriver * usbp,
    usbep_t ep)
```

Default EP0 IN callback.

This function is used by the low level driver as default handler for EP0 IN events.

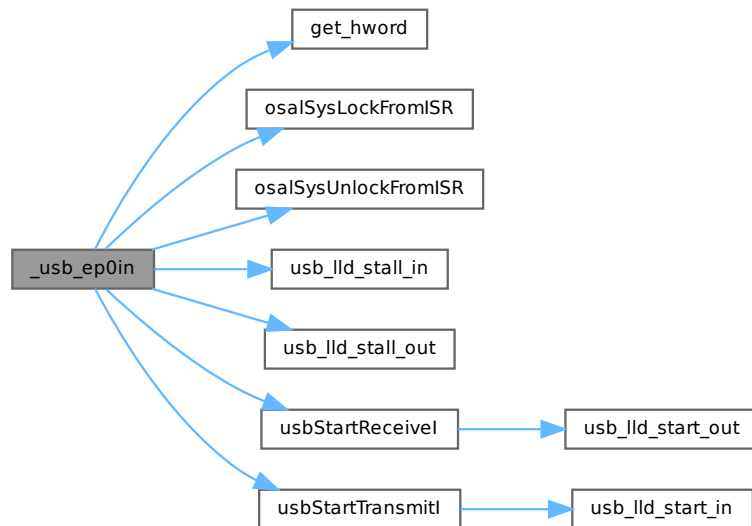
Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number, always zero

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.25.7.22 _usb_ep0out()**

```
void _usb_ep0out (
    USBDriver * usbp,
    usbep_t ep)
```

Default EP0 OUT callback.

This function is used by the low level driver as default handler for EP0 OUT events.

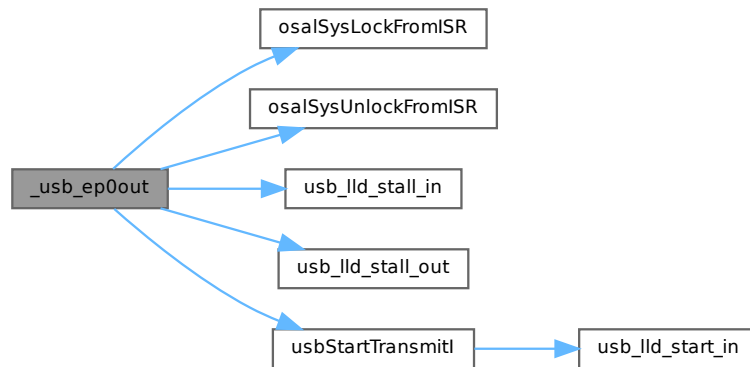
Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number, always zero

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.9.25.7.23 usbReadSetupI()**

```

void usbReadSetupI (
    USBDriver * usbp,
    usbep_t ep,
    uint8_t * buf)
  
```

7.2.9.25.7.24 usb_lld_init()

```

void usb_lld_init (
    void )
  
```

Low level USB driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.25.7.25 usb_ll_start()

```
void usb_ll_start (
    USBDriver * usbp)
```

Configures and activates the USB peripheral.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.26 usb_ll_stop()

```
void usb_ll_stop (
    USBDriver * usbp)
```

Deactivates the USB peripheral.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.27 usb_ll_reset()

```
void usb_ll_reset (
    USBDriver * usbp)
```

USB low level reset routine.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.25.7.28 usb_llc_set_address()

```
void usb_llc_set_address (
    USBDriver * usbp)
```

Sets the USB address.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.29 usb_llc_init_endpoint()

```
void usb_llc_init_endpoint (
    USBDriver * usbp,
    usbep_t ep)
```

Enables an endpoint.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.30 usb_llc_disable_endpoints()

```
void usb_llc_disable_endpoints (
    USBDriver * usbp)
```

Disables all the active endpoints except the endpoint zero.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.31 usb_llc_get_status_out()

```
usbepstatus_t usb_llc_get_status_out (
    USBDriver * usbp,
    usbep_t ep)
```

Returns the status of an OUT endpoint.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Returns

The endpoint status.

Return values

<i>EP_STATUS_DISABLED</i>	The endpoint is not active.
<i>EP_STATUS_STALLED</i>	The endpoint is stalled.
<i>EP_STATUS_ACTIVE</i>	The endpoint is active.

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.32 usb_lld_get_status_in()

```
usbepstatus_t usb_lld_get_status_in (  
    USBDriver * usbp,  
    usbep_t ep)
```

Returns the status of an IN endpoint.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Returns

The endpoint status.

Return values

<i>EP_STATUS_DISABLED</i>	The endpoint is not active.
<i>EP_STATUS_STALLED</i>	The endpoint is stalled.
<i>EP_STATUS_ACTIVE</i>	The endpoint is active.

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.33 usb_ll_read_setup()

```
void usb_ll_read_setup (
    USBDriver * usbp,
    usbep_t ep,
    uint8_t * buf)
```

Reads a setup packet from the dedicated packet buffer.

This function must be invoked in the context of the `setup_cb` callback in order to read the received setup packet.

Precondition

In order to use this function the endpoint must have been initialized as a control endpoint.

Postcondition

The endpoint is ready to accept another packet.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number
out	<i>buf</i>	buffer where to copy the packet data

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.34 usb_ll_prepare_receive()

```
void usb_ll_prepare_receive (
    USBDriver * usbp,
    usbep_t ep)
```

Prepares for a receive operation.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.35 usb_ll_prepare_transmit()

```
void usb_ll_prepare_transmit (
    USBDriver * usbp,
    usbep_t ep)
```

Prepares for a transmit operation.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.36 usb_lld_start_out()

```
void usb_lld_start_out (  
    USBDriver * usbp,  
    usbep\_t ep)
```

Starts a receive operation on an OUT endpoint.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.37 usb_lld_start_in()

```
void usb_lld_start_in (  
    USBDriver * usbp,  
    usbep\_t ep)
```

Starts a transmit operation on an IN endpoint.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.38 usb_lld_stall_out()

```
void usb_lld_stall_out (  
    USBDriver * usbp,  
    usbep\_t ep)
```

Brings an OUT endpoint in the stalled state.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.39 usb_llc_stall_in()

```
void usb_llc_stall_in (  
    USBDriver * usbp,  
    usbep\_t ep)
```

Brings an IN endpoint in the stalled state.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.40 usb_llc_clear_out()

```
void usb_llc_clear_out (  
    USBDriver * usbp,  
    usbep\_t ep)
```

Brings an OUT endpoint in the active state.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.7.41 usb_llc_clear_in()

```
void usb_llc_clear_in (  
    USBDriver * usbp,  
    usbep\_t ep)
```

Brings an IN endpoint in the active state.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object
in	<i>ep</i>	endpoint number

Function Class:

Not an API, this function is for internal use only.

7.2.9.25.8 Variable Documentation**7.2.9.25.8.1 zero_status**

```
const uint8_t zero_status[] = {0x00, 0x00} [static]
```

7.2.9.25.8.2 active_status

```
const uint8_t active_status[] = {0x00, 0x00} [static]
```

7.2.9.25.8.3 halted_status

```
const uint8_t halted_status[] = {0x01, 0x00} [static]
```

7.2.9.25.8.4 USBD1

[USBDriver](#) USBD1

USB1 driver identifier.

7.2.9.25.8.5 [union]

```
union { ... } ep0_state [static]
```

EP0 state.

Note

It is an union because IN and OUT endpoints are never used at the same time for EP0.

7.2.9.25.8.6 []

```
USBInEndpointState { ... } ::in
```

IN EP0 state.

7.2.9.25.8.7 []

```
USBOutEndpointState { ... } ::out
```

OUT EP0 state.

7.2.9.25.8.8 ep0config

```
const USBEndpointConfig ep0config [static]
```

Initial value:

```
= {
    USB_EP_MODE_TYPE_CTRL,
    _usb_ep0setup,
    _usb_ep0in,
    _usb_ep0out,
    0x40,
    0x40,
    &ep0_state.in,
    &ep0_state.out
}
```

EP0 initialization structure.

7.2.9.26 WDG Driver

Generic WDG Driver.

7.2.9.26.1 Detailed Description

Generic WDG Driver.

This module defines an abstract interface for a watchdog timer.

Precondition

In order to use the WDG driver the `HAL_USE_WDG` option must be enabled in `halconf.h`.

Configuration options

- `#define PLATFORM_WDG_USE_WDG1 FALSE`
WDG1 driver enable switch.

Data Structures

- struct `WDGConfig`
Driver configuration structure.
- struct `WDGDriver`
Structure representing an WDG driver.

Macros

- `#define wdgResetI(wdgp)`
Resets WDG's counter.

Typedefs

- `typedef struct WDGDriver WDGDriver`
Type of a structure representing an WDG driver.

Enumerations

- `enum wdgstate_t { WDG_UNINIT = 0 , WDG_STOP = 1 , WDG_READY = 2 }`
Driver state machine possible states.

Functions

- `void wdgInit (void)`
WDG Driver initialization.
- `msg_t wdgStart (WDGDriver *wdgp, const WDGConfig *config)`
Configures and activates the WDG peripheral.
- `void wdgStop (WDGDriver *wdgp)`
Deactivates the WDG peripheral.
- `void wdgReset (WDGDriver *wdgp)`
Resets WDG's counter.
- `void wdg_lld_init (void)`
Low level WDG driver initialization.
- `void wdg_lld_start (WDGDriver *wdgp)`
Configures and activates the WDG peripheral.
- `void wdg_lld_stop (WDGDriver *wdgp)`
Deactivates the WDG peripheral.
- `void wdg_lld_reset (WDGDriver *wdgp)`
Reloads WDG's counter.

Variables

- `WDGDriver WGD1`

7.2.9.26.2 Macro Definition Documentation

7.2.9.26.2.1 wdgResetI

```
#define wdgResetI(  
    wdgp)
```

Value:

`wdg_lld_reset (wdgp)`

Resets WDG's counter.

Parameters

in	<i>wdgp</i>	pointer to the WDGDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.26.2.2 PLATFORM_WDG_USE_WDG1

```
#define PLATFORM_WDG_USE_WDG1 FALSE
```

WDG1 driver enable switch.

Note

The default is FALSE.

7.2.9.26.3 Typedef Documentation**7.2.9.26.3.1 WDGDriver**

```
typedef struct WDGDriver WDGDriver
```

Type of a structure representing an WDG driver.

7.2.9.26.4 Enumeration Type Documentation**7.2.9.26.4.1 wdgstate_t**

```
enum wdgstate_t
```

Driver state machine possible states.

Enumerator

WDG_UNINIT	Not initialized.
WDG_STOP	Stopped.
WDG_READY	Ready.

7.2.9.26.5 Function Documentation

7.2.9.26.5.1 wdgInit()

```
void wdgInit (  
    void )
```

WDG Driver initialization.

Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.26.5.2 wdgStart()

```
msg_t wdgStart (  
    WDGDriver * wdg,   
    const WDGConfig * config)
```

Configures and activates the WDG peripheral.

Parameters

in	<i>wdg</i>	pointer to the <code>WDGDriver</code> object
in	<i>config</i>	pointer to the <code>WDGConfig</code> object

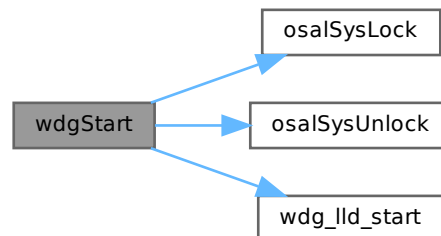
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.26.5.3 wdgStop()**

```
void wdgStop (  
    WDGDriver * wdgp)
```

Deactivates the WDG peripheral.

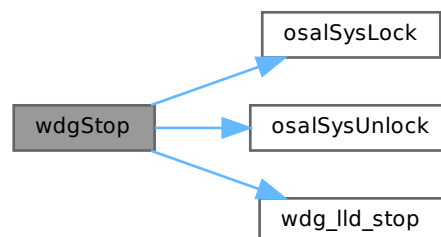
Parameters

in	<code>wdgp</code>	pointer to the <code>WDGDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.26.5.4 wdgReset()

```
void wdgReset (  
    WDGDriver * wdgp)
```

Resets WDG's counter.

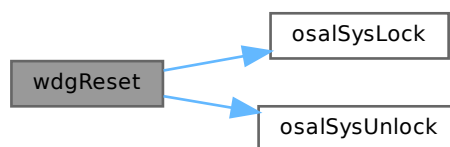
Parameters

in	wdgp	pointer to the WDGDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.26.5.5 wdg_lld_init()

```
void wdg_lld_init (  
    void )
```

Low level WDG driver initialization.

Function Class:

Not an API, this function is for internal use only.

7.2.9.26.5.6 wdg_lld_start()

```
void wdg_lld_start (  
    WDGDriver * wdgp)
```

Configures and activates the WDG peripheral.

Parameters

in	<i>wdgp</i>	pointer to the WDGDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.26.5.7 wdg_llt_stop()

```
void wdg_llt_stop (  
    WDGDriver * wdgp)
```

Deactivates the WDG peripheral.

Parameters

in	<i>wdgp</i>	pointer to the WDGDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.26.5.8 wdg_llt_reset()

```
void wdg_llt_reset (  
    WDGDriver * wdgp)
```

Reloads WDG's counter.

Parameters

in	<i>wdgp</i>	pointer to the WDGDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.26.6 Variable Documentation**7.2.9.26.6.1 WDGD1**

[WDGDriver](#) WDGD1

7.2.9.27 WSPI Driver

Generic WSPI Driver.

7.2.9.27.1 Detailed Description

Generic WSPI Driver.

This module defines an abstract interface for an wide SPI communication interface (Quad SPI, Octal SPI and similar).

Precondition

In order to use the WSPI driver the `HAL_USE_WSPI` option must be enabled in [halconf.h](#).

WSPI configuration options

- `#define WSPI_USE_WAIT TRUE`
Enables synchronous APIs.
- `#define WSPI_USE_MUTUAL_EXCLUSION TRUE`
Enables the `wspiAcquireBus()` and `wspiReleaseBus()` APIs.

Transfer options

Note

The low level driver has the option to override the following definitions and use its own ones. In must take care to use the same name for the same function or compatibility is not ensured.

- `#define WSPI_CFG_CMD_MODE_MASK (7LU << 0LU)`
- `#define WSPI_CFG_CMD_MODE_NONE (0LU << 0LU)`
- `#define WSPI_CFG_CMD_MODE_ONE_LINE (1LU << 0LU)`
- `#define WSPI_CFG_CMD_MODE_TWO_LINES (2LU << 0LU)`
- `#define WSPI_CFG_CMD_MODE_FOUR_LINES (3LU << 0LU)`
- `#define WSPI_CFG_CMD_MODE_EIGHT_LINES (4LU << 0LU)`
- `#define WSPI_CFG_CMD_DTR (1LU << 3LU)`
- `#define WSPI_CFG_CMD_SIZE_MASK (3LU << 4LU)`
- `#define WSPI_CFG_CMD_SIZE_8 (0LU << 4LU)`
- `#define WSPI_CFG_CMD_SIZE_16 (1LU << 4LU)`
- `#define WSPI_CFG_CMD_SIZE_24 (2LU << 4LU)`
- `#define WSPI_CFG_CMD_SIZE_32 (3LU << 4LU)`
- `#define WSPI_CFG_ADDR_MODE_MASK (7LU << 8LU)`
- `#define WSPI_CFG_ADDR_MODE_NONE (0LU << 8LU)`
- `#define WSPI_CFG_ADDR_MODE_ONE_LINE (1LU << 8LU)`
- `#define WSPI_CFG_ADDR_MODE_TWO_LINES (2LU << 8LU)`
- `#define WSPI_CFG_ADDR_MODE_FOUR_LINES (3LU << 8LU)`
- `#define WSPI_CFG_ADDR_MODE_EIGHT_LINES (4LU << 8LU)`
- `#define WSPI_CFG_ADDR_DTR (1LU << 11LU)`
- `#define WSPI_CFG_ADDR_SIZE_MASK (3LU << 12LU)`
- `#define WSPI_CFG_ADDR_SIZE_8 (0LU << 12LU)`
- `#define WSPI_CFG_ADDR_SIZE_16 (1LU << 12LU)`

- #define `WSPI_CFG_ADDR_SIZE_24` (2LU << 12LU)
- #define `WSPI_CFG_ADDR_SIZE_32` (3LU << 12LU)
- #define `WSPI_CFG_ALT_MODE_MASK` (7LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_NONE` (0LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_ONE_LINE` (1LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_TWO_LINES` (2LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_FOUR_LINES` (3LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_EIGHT_LINES` (4LU << 16LU)
- #define `WSPI_CFG_ALT_DTR` (1LU << 19LU)
- #define `WSPI_CFG_ALT_SIZE_MASK` (3LU << 20LU)
- #define `WSPI_CFG_ALT_SIZE_8` (0LU << 20LU)
- #define `WSPI_CFG_ALT_SIZE_16` (1LU << 20LU)
- #define `WSPI_CFG_ALT_SIZE_24` (2LU << 20LU)
- #define `WSPI_CFG_ALT_SIZE_32` (3LU << 20LU)
- #define `WSPI_CFG_DATA_MODE_MASK` (7LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_NONE` (0LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_ONE_LINE` (1LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_TWO_LINES` (2LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_FOUR_LINES` (3LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_EIGHT_LINES` (4LU << 24LU)
- #define `WSPI_CFG_DATA_DTR` (1LU << 27LU)
- #define `WSPI_CFG_DQS_ENABLE` (1LU << 29LU)
- #define `WSPI_CFG_SIOO` (1LU << 31LU)
- #define `WSPI_CFG_ALL_DTR`

Macro Functions

- #define `wspiStartCommandl(wspip, cmdp)`
Sends a command without data phase.
- #define `wspiStartSendl(wspip, cmdp, n, txbuf)`
Sends data over the WSPI bus.
- #define `wspiStartReceive(wspip, cmdp, n, rxbuf)`
Receives data from the WSPI bus.
- #define `wspiMapFlashl(wspip, cmdp, addrp)`
Maps in memory space a WSPI flash device.
- #define `wspiUnmapFlashl(wspip)`
Maps in memory space a WSPI flash device.

Low level driver helper macros

- #define `_wspi_wakeup_isr(wspip, msg)`
Wakes up the waiting thread.
- #define `_wspi_isr_code(wspip)`
Common ISR code.
- #define `_wspi_error_code(wspip)`
Common error ISR code.

WSPI implementation capabilities

- #define `WSPI_SUPPORTS_MEMMAP` TRUE
- #define `WSPI_DEFAULT_CFG_MASKS` TRUE

Configuration options

- #define `PLATFORM_WSPI_USE_WSPID1` `FALSE`
WSPID1 driver enable switch.

Data Structures

- struct `wspi_command_t`
Type of a WSPI command descriptor.
- struct `hal_wspi_config`
Driver configuration structure.
- struct `hal_wspi_driver`
Structure representing an WSPI driver.

Macros

- #define `wspi_lld_driver_fields`
Low level fields of the WSPI driver structure.
- #define `wspi_lld_config_fields`
Low level fields of the WSPI configuration structure.

Typedefs

- typedef struct `hal_wspi_driver` `WSPIDriver`
Type of a structure representing an WSPI driver.
- typedef struct `hal_wspi_config` `WSPIDriverConfig`
Type of a structure representing an WSPI driver configuration.
- typedef void(* `wspicallback_t`) (`WSPIDriver` *wspip)
Type of a WSPI notification callback.

Enumerations

- enum `wspistate_t` {
 `WSPIDriver_UNINIT` = 0 , `WSPIDriver_STOP` = 1 , `WSPIDriver_READY` = 2 , `WSPIDriver_SEND` = 3 ,
 `WSPIDriver_RECEIVE` = 4 , `WSPIDriver_COMPLETE` = 5 , `WSPIDriver_MEMMAP` = 6 }
Driver state machine possible states.

Functions

- void [wspiInit](#) (void)
WSPI Driver initialization.
- void [wspiObjectInit](#) (WSPIDriver *wspip)
Initializes the standard part of a [WSPIDriver](#) structure.
- [msg_t wspiStart](#) (WSPIDriver *wspip, const WSPIDriverConfig *config)
Configures and activates the WSPI peripheral.
- void [wspiStop](#) (WSPIDriver *wspip)
Deactivates the WSPI peripheral.
- void [wspiStartCommand](#) (WSPIDriver *wspip, const wspi_command_t *cmdp)
Sends a command without data phase.
- void [wspiStartSend](#) (WSPIDriver *wspip, const wspi_command_t *cmdp, size_t n, const uint8_t *txbuf)
Sends a command with data over the WSPI bus.
- void [wspiStartReceive](#) (WSPIDriver *wspip, const wspi_command_t *cmdp, size_t n, uint8_t *rxbuf)
Sends a command then receives data over the WSPI bus.
- bool [wspiCommand](#) (WSPIDriver *wspip, const wspi_command_t *cmdp)
Sends a command without data phase.
- bool [wspiSend](#) (WSPIDriver *wspip, const wspi_command_t *cmdp, size_t n, const uint8_t *txbuf)
Sends a command with data over the WSPI bus.
- bool [wspiReceive](#) (WSPIDriver *wspip, const wspi_command_t *cmdp, size_t n, uint8_t *rxbuf)
Sends a command then receives data over the WSPI bus.
- void [wspiMapFlash](#) (WSPIDriver *wspip, const wspi_command_t *cmdp, uint8_t **addrp)
Maps in memory space a WSPI flash device.
- void [wspiUnmapFlash](#) (WSPIDriver *wspip)
Unmaps from memory space a WSPI flash device.
- void [wspiAcquireBus](#) (WSPIDriver *wspip)
Gains exclusive access to the WSPI bus.
- void [wspiReleaseBus](#) (WSPIDriver *wspip)
Releases exclusive access to the WSPI bus.
- void [wspi_lld_init](#) (void)
Low level WSPI driver initialization.
- void [wspi_lld_start](#) (WSPIDriver *wspip)
Configures and activates the WSPI peripheral.
- void [wspi_lld_stop](#) (WSPIDriver *wspip)
Deactivates the WSPI peripheral.
- void [wspi_lld_command](#) (WSPIDriver *wspip, const wspi_command_t *cmdp)
Sends a command without data phase.
- void [wspi_lld_send](#) (WSPIDriver *wspip, const wspi_command_t *cmdp, size_t n, const uint8_t *txbuf)
Sends a command with data over the WSPI bus.
- void [wspi_lld_receive](#) (WSPIDriver *wspip, const wspi_command_t *cmdp, size_t n, uint8_t *rxbuf)
Sends a command then receives data over the WSPI bus.
- void [wspi_lld_map_flash](#) (WSPIDriver *wspip, const wspi_command_t *cmdp, uint8_t **addrp)
Maps in memory space a WSPI flash device.
- void [wspi_lld_unmap_flash](#) (WSPIDriver *wspip)
Unmaps from memory space a WSPI flash device.

Variables

- [WSPIDriver WSPID1](#)
WSPID1 driver identifier.

7.2.9.27.2 Macro Definition Documentation

7.2.9.27.2.1 WSPI_USE_WAIT

```
#define WSPI_USE_WAIT TRUE
```

Enables synchronous APIs.

Note

Disabling this option saves both code and data space.

7.2.9.27.2.2 WSPI_USE_MUTUAL_EXCLUSION

```
#define WSPI_USE_MUTUAL_EXCLUSION TRUE
```

Enables the `wspiAcquireBus()` and `wspiReleaseBus()` APIs.

Note

Disabling this option saves both code and data space.

7.2.9.27.2.3 WSPI_CFG_CMD_MODE_MASK

```
#define WSPI_CFG_CMD_MODE_MASK (7LU << 0LU)
```

7.2.9.27.2.4 WSPI_CFG_CMD_MODE_NONE

```
#define WSPI_CFG_CMD_MODE_NONE (0LU << 0LU)
```

7.2.9.27.2.5 WSPI_CFG_CMD_MODE_ONE_LINE

```
#define WSPI_CFG_CMD_MODE_ONE_LINE (1LU << 0LU)
```

7.2.9.27.2.6 WSPI_CFG_CMD_MODE_TWO_LINES

```
#define WSPI_CFG_CMD_MODE_TWO_LINES (2LU << 0LU)
```

7.2.9.27.2.7 WSPI_CFG_CMD_MODE_FOUR_LINES

```
#define WSPI_CFG_CMD_MODE_FOUR_LINES (3LU << 0LU)
```


7.2.9.27.2.8 WSPI_CFG_CMD_MODE_EIGHT_LINES

```
#define WSPI_CFG_CMD_MODE_EIGHT_LINES (4LU << 0LU)
```

7.2.9.27.2.9 WSPI_CFG_CMD_DTR

```
#define WSPI_CFG_CMD_DTR (1LU << 3LU)
```

7.2.9.27.2.10 WSPI_CFG_CMD_SIZE_MASK

```
#define WSPI_CFG_CMD_SIZE_MASK (3LU << 4LU)
```

7.2.9.27.2.11 WSPI_CFG_CMD_SIZE_8

```
#define WSPI_CFG_CMD_SIZE_8 (0LU << 4LU)
```

7.2.9.27.2.12 WSPI_CFG_CMD_SIZE_16

```
#define WSPI_CFG_CMD_SIZE_16 (1LU << 4LU)
```

7.2.9.27.2.13 WSPI_CFG_CMD_SIZE_24

```
#define WSPI_CFG_CMD_SIZE_24 (2LU << 4LU)
```

7.2.9.27.2.14 WSPI_CFG_CMD_SIZE_32

```
#define WSPI_CFG_CMD_SIZE_32 (3LU << 4LU)
```

7.2.9.27.2.15 WSPI_CFG_ADDR_MODE_MASK

```
#define WSPI_CFG_ADDR_MODE_MASK (7LU << 8LU)
```

7.2.9.27.2.16 WSPI_CFG_ADDR_MODE_NONE

```
#define WSPI_CFG_ADDR_MODE_NONE (0LU << 8LU)
```

7.2.9.27.2.17 WSPI_CFG_ADDR_MODE_ONE_LINE

```
#define WSPI_CFG_ADDR_MODE_ONE_LINE (1LU << 8LU)
```

7.2.9.27.2.18 WSPI_CFG_ADDR_MODE_TWO_LINES

```
#define WSPI_CFG_ADDR_MODE_TWO_LINES (2LU << 8LU)
```

7.2.9.27.2.19 WSPI_CFG_ADDR_MODE_FOUR_LINES

```
#define WSPI_CFG_ADDR_MODE_FOUR_LINES (3LU << 8LU)
```

7.2.9.27.2.20 WSPI_CFG_ADDR_MODE_EIGHT_LINES

```
#define WSPI_CFG_ADDR_MODE_EIGHT_LINES (4LU << 8LU)
```

7.2.9.27.2.21 WSPI_CFG_ADDR_DTR

```
#define WSPI_CFG_ADDR_DTR (1LU << 11LU)
```

7.2.9.27.2.22 WSPI_CFG_ADDR_SIZE_MASK

```
#define WSPI_CFG_ADDR_SIZE_MASK (3LU << 12LU)
```

7.2.9.27.2.23 WSPI_CFG_ADDR_SIZE_8

```
#define WSPI_CFG_ADDR_SIZE_8 (0LU << 12LU)
```

7.2.9.27.2.24 WSPI_CFG_ADDR_SIZE_16

```
#define WSPI_CFG_ADDR_SIZE_16 (1LU << 12LU)
```

7.2.9.27.2.25 WSPI_CFG_ADDR_SIZE_24

```
#define WSPI_CFG_ADDR_SIZE_24 (2LU << 12LU)
```

7.2.9.27.2.26 WSPI_CFG_ADDR_SIZE_32

```
#define WSPI_CFG_ADDR_SIZE_32 (3LU << 12LU)
```

7.2.9.27.2.27 WSPI_CFG_ALT_MODE_MASK

```
#define WSPI_CFG_ALT_MODE_MASK (7LU << 16LU)
```

7.2.9.27.2.28 WSPI_CFG_ALT_MODE_NONE

```
#define WSPI_CFG_ALT_MODE_NONE (0LU << 16LU)
```

7.2.9.27.2.29 WSPI_CFG_ALT_MODE_ONE_LINE

```
#define WSPI_CFG_ALT_MODE_ONE_LINE (1LU << 16LU)
```

7.2.9.27.2.30 WSPI_CFG_ALT_MODE_TWO_LINES

```
#define WSPI_CFG_ALT_MODE_TWO_LINES (2LU << 16LU)
```

7.2.9.27.2.31 WSPI_CFG_ALT_MODE_FOUR_LINES

```
#define WSPI_CFG_ALT_MODE_FOUR_LINES (3LU << 16LU)
```

7.2.9.27.2.32 WSPI_CFG_ALT_MODE_EIGHT_LINES

```
#define WSPI_CFG_ALT_MODE_EIGHT_LINES (4LU << 16LU)
```

7.2.9.27.2.33 WSPI_CFG_ALT_DTR

```
#define WSPI_CFG_ALT_DTR (1LU << 19LU)
```

7.2.9.27.2.34 WSPI_CFG_ALT_SIZE_MASK

```
#define WSPI_CFG_ALT_SIZE_MASK (3LU << 20LU)
```

7.2.9.27.2.35 WSPI_CFG_ALT_SIZE_8

```
#define WSPI_CFG_ALT_SIZE_8 (0LU << 20LU)
```

7.2.9.27.2.36 WSPI_CFG_ALT_SIZE_16

```
#define WSPI_CFG_ALT_SIZE_16 (1LU << 20LU)
```

7.2.9.27.2.37 WSPI_CFG_ALT_SIZE_24

```
#define WSPI_CFG_ALT_SIZE_24 (2LU << 20LU)
```

7.2.9.27.2.38 WSPI_CFG_ALT_SIZE_32

```
#define WSPI_CFG_ALT_SIZE_32 (3LU << 20LU)
```

7.2.9.27.2.39 WSPI_CFG_DATA_MODE_MASK

```
#define WSPI_CFG_DATA_MODE_MASK (7LU << 24LU)
```

7.2.9.27.2.40 WSPI_CFG_DATA_MODE_NONE

```
#define WSPI_CFG_DATA_MODE_NONE (0LU << 24LU)
```

7.2.9.27.2.41 WSPI_CFG_DATA_MODE_ONE_LINE

```
#define WSPI_CFG_DATA_MODE_ONE_LINE (1LU << 24LU)
```

7.2.9.27.2.42 WSPI_CFG_DATA_MODE_TWO_LINES

```
#define WSPI_CFG_DATA_MODE_TWO_LINES (2LU << 24LU)
```

7.2.9.27.2.43 WSPI_CFG_DATA_MODE_FOUR_LINES

```
#define WSPI_CFG_DATA_MODE_FOUR_LINES (3LU << 24LU)
```

7.2.9.27.2.44 WSPI_CFG_DATA_MODE_EIGHT_LINES

```
#define WSPI_CFG_DATA_MODE_EIGHT_LINES (4LU << 24LU)
```

7.2.9.27.2.45 WSPI_CFG_DATA_DTR

```
#define WSPI_CFG_DATA_DTR (1LU << 27LU)
```

7.2.9.27.2.46 WSPI_CFG_DQS_ENABLE

```
#define WSPI_CFG_DQS_ENABLE (1LU << 29LU)
```

7.2.9.27.2.47 WSPI_CFG_SIOO

```
#define WSPI_CFG_SIOO (1LU << 31LU)
```

7.2.9.27.2.48 WSPI_CFG_ALL_DTR

```
#define WSPI_CFG_ALL_DTR
```

Value:

```
(WSPI_CFG_CMD_DTR | \
WSPI_CFG_ADDR_DTR | \
WSPI_CFG_ALT_DTR | \
WSPI_CFG_DATA_DTR)
```

7.2.9.27.2.49 wspiStartCommandI

```
#define wspiStartCommandI(
    wspi,
    cmdp)
```

Value:

```
{
    osalDbgAssert(((cmdp)->cfg & WSPI_CFG_DATA_MODE_MASK) == \
        WSPI_CFG_DATA_MODE_NONE, \
        "data mode specified"); \
    (wspi)->state = WSPI_SEND; \
    wspi_llc_command(wspi, cmdp); \
}
```

Sends a command without data phase.

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>wspi</i>	pointer to the WSPI_Driver object
in	<i>cmdp</i>	pointer to the command descriptor

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.27.2.50 wspiStartSendI

```
#define wspiStartSendI(
    wspi,
    cmdp,
    n,
    txbuf)
```

Value:

```
{
    osalDbgAssert(((cmdp)->cfg & WSPI_CFG_DATA_MODE_MASK) != \
        WSPI_CFG_DATA_MODE_NONE, \
        "data mode required"); \
    (wspi)->state = WSPI_SEND; \
    wspi_llc_send(wspi, cmdp, n, txbuf); \
}
```

Sends data over the WSPI bus.

This asynchronous function starts a transmit operation.

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
in	<i>n</i>	number of bytes to send or zero if no data phase
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.27.2.51 wspiStartReceiveI

```
#define wspiStartReceiveI(
    wspip,
    cmdp,
    n,
    rxbuf)
```

Value:

```
{
    osalDbgAssert(((cmdp)->cfg & WSPI_CFG_DATA_MODE_MASK) !=
        WSPI_CFG_DATA_MODE_NONE,
        "data mode required");
    (wspip)->state = WSPI_RECEIVE;
    wspi_lld_receive(wspip, cmdp, n, rxbuf);
}
```

Receives data from the WSPI bus.

This asynchronous function starts a receive operation.

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
in	<i>n</i>	number of bytes to receive or zero if no data phase
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.27.2.52 wspiMapFlashI

```
#define wspiMapFlashI(  
    wspi,  
    cmdp,  
    addrp)
```

Value:

```
wspi_lld_map_flash(wspi, cmdp, addrp)
```

Maps in memory space a WSPI flash device.

Precondition

The memory flash device must be initialized appropriately before mapping it in memory space.

Parameters

in	<i>wspi</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
out	<i>addrp</i>	pointer to the memory start address of the mapped flash or <code>NULL</code>

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.27.2.53 wspiUnmapFlashI

```
#define wspiUnmapFlashI(  
    wspi)
```

Value:

```
wspi_lld_unmap_flash(wspi)
```

Maps in memory space a WSPI flash device.

Postcondition

The memory flash device must be re-initialized for normal commands exchange.

Parameters

in	<i>wspi</i>	pointer to the WSPIDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.9.27.2.54 _wspi_wakeup_isr

```
#define _wspi_wakeup_isr(  
    wspi,  
    msg)
```

Value:

```
{  
    osalSysLockFromISR();  
    osalThreadResumeI(&(wspi)->thread, msg);  
    osalSysUnlockFromISR();  
}
```

Wakes up the waiting thread.

Parameters

in	<i>wspi</i>	pointer to the WSPIDriver object
in	<i>msg</i>	the wakeup message

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.2.55 _wspi_isr_code

```
#define _wspi_isr_code(  
    wspi)
```

Value:

```
{  
    if ((wspi)->config->end_cb) {  
        (wspi)->state = WSPI\_COMPLETE;  
        (wspi)->config->end_cb(wspi);  
        if ((wspi)->state == WSPI\_COMPLETE)  
            (wspi)->state = WSPI\_READY;  
    }  
    else  
        (wspi)->state = WSPI\_READY;  
    _wspi_wakeup_isr(wspi, MSG\_OK);  
}
```

Common ISR code.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.2.56 `_wspi_error_code`

```
#define _wspi_error_code(  
    wspip)
```

Value:

```
{  
    if ((wspip)->config->error_cb) {  
        (wspip)->state = WSPI_COMPLETE;  
        (wspip)->config->error_cb(wspip);  
        if ((wspip)->state == WSPI_COMPLETE)  
            (wspip)->state = WSPI_READY;  
    }  
    else  
        (wspip)->state = WSPI_READY;  
    _wspi_wakeup_isr(wspip, MSG_RESET);  
}
```

Common error ISR code.

This code handles the portable part of the ISR code:

- Callback invocation.
- Waiting thread wakeup, if any.
- Driver state transitions.

Note

This macro is meant to be used in the low level drivers implementation only.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.2.57 `WSPI_SUPPORTS_MEMMAP`

```
#define WSPI_SUPPORTS_MEMMAP TRUE
```

7.2.9.27.2.58 WSPI_DEFAULT_CFG_MASKS

```
#define WSPI_DEFAULT_CFG_MASKS TRUE
```

7.2.9.27.2.59 PLATFORM_WSPI_USE_WSPI1

```
#define PLATFORM_WSPI_USE_WSPI1 FALSE
```

WSPID1 driver enable switch.

If set to TRUE the support for WSPID1 is included.

Note

The default is FALSE.

7.2.9.27.2.60 wspi_ll_driver_fields

```
#define wspi_ll_driver_fields
```

Value:

```
/* Dummy field, it is not needed.*/
uint32_t dummy;
```

Low level fields of the WSPI driver structure.

7.2.9.27.2.61 wspi_ll_config_fields

```
#define wspi_ll_config_fields
```

Value:

```
/* Dummy configuration, it is not needed.*/
uint32_t dummy;
```

Low level fields of the WSPI configuration structure.

7.2.9.27.3 Typedef Documentation**7.2.9.27.3.1 WSPIDriver**

```
typedef struct hal_wspi_driver WSPIDriver
```

Type of a structure representing an WSPI driver.

7.2.9.27.3.2 WSPIDriverConfig

```
typedef struct hal_wspi_config WSPIDriverConfig
```

Type of a structure representing an WSPI driver configuration.

7.2.9.27.3.3 wspi_callback_t

```
typedef void(* wspi_callback_t) (WSPIDriver *wspip)
```

Type of a WSPI notification callback.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object triggering the callback

7.2.9.27.4 Enumeration Type Documentation

7.2.9.27.4.1 `wspistate_t`

```
enum wspistate_t
```

Driver state machine possible states.

Enumerator

WSPi_UNINIT	Not initialized.
WSPi_STOP	Stopped.
WSPi_READY	Ready.
WSPi_SEND	Sending data.
WSPi_RECEIVE	Receiving data.
WSPi_COMPLETE	Asynchronous operation complete.
WSPi_MEMMAP	In memory mapped mode.

7.2.9.27.5 Function Documentation

7.2.9.27.5.1 `wspilnit()`

```
void wspilnit (
    void )
```

WSPi Driver initialization.

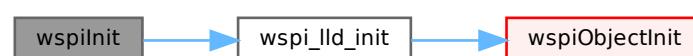
Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.9.27.5.2 wspiObjectInit()

```
void wspiObjectInit (  
    WSPIDriver * wspip)
```

Initializes the standard part of a `WSPIDriver` structure.

Parameters

out	<i>wspip</i>	pointer to the WSPIDriver object

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.9.27.5.3 wspiStart()**

```

msg_t wspiStart (
    WSPIDriver * wspip,
    const WSPIDriverConfig * config)
  
```

Configures and activates the WSPID peripheral.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object
in	<i>config</i>	pointer to the WSPIDriverConfig object

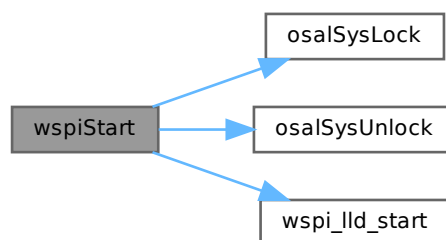
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.27.5.4 wspiStop()

```
void wspiStop (  
    WSPIDriver * wspi)
```

Deactivates the WSPI peripheral.

Note

Deactivating the peripheral also enforces a release of the slave select line.

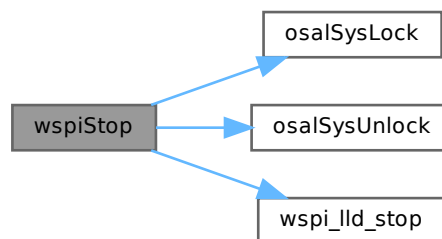
Parameters

in	<i>wspi</i>	pointer to the <code>WSPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.27.5.5 wspiStartCommand()

```
void wspiStartCommand (  
    WSPIDriver * wspi,  
    const wspi_command_t * cmdp)
```

Sends a command without data phase.

Postcondition

At the end of the operation the configured callback is invoked.

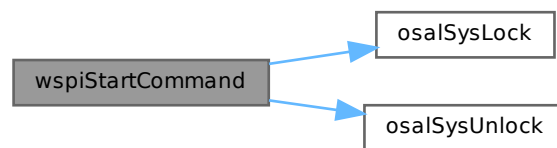
Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.27.5.6 wspiStartSend()**

```
void wspiStartSend (  
    WSPIDriver * wspip,  
    const wspi_command_t * cmdp,  
    size_t n,  
    const uint8_t * txbuf)
```

Sends a command with data over the WSPI bus.

Postcondition

At the end of the operation the configured callback is invoked.

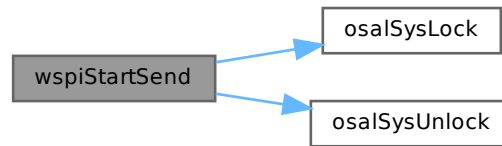
Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
in	<i>n</i>	number of bytes to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.27.5.7 wspiStartReceive()**

```

void wspiStartReceive (
    WSPIDriver * wspi,
    const wspi_command_t * cmdp,
    size_t n,
    uint8_t * rxbuf)
  
```

Sends a command then receives data over the WSPI bus.

Postcondition

At the end of the operation the configured callback is invoked.

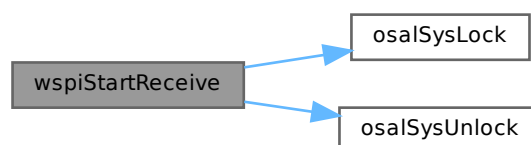
Parameters

in	<i>wspi</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
in	<i>n</i>	number of bytes to send
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.27.5.8 wspiCommand()

```
bool wspiCommand (  
    WSPIDriver * wspi,  
    const wspi_command_t * cmdp)
```

Sends a command without data phase.

Precondition

In order to use this function the option `WSP_I_USE_WAIT` must be enabled.

In order to use this function the driver must have been configured without callbacks (`end_cb = NULL`).

Parameters

in	<i>wspi</i>	pointer to the <code>WSPIDriver</code> object
in	<i>cmdp</i>	pointer to the command descriptor

Returns

The operation status.

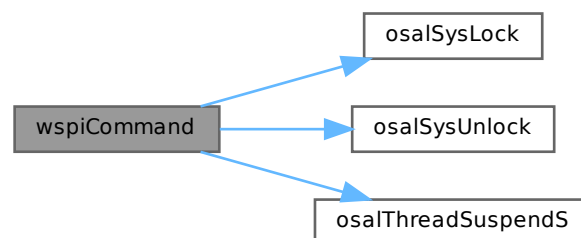
Return values

<i>false</i>	if the operation succeeded.
<i>true</i>	if the operation failed because HW issues.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.27.5.9 wspiSend()

```
bool wspiSend (  
    WSPIDriver * wspi,  
    const wspi_command_t * cmdp,  
    size_t n,  
    const uint8_t * txbuf)
```

Sends a command with data over the WSPI bus.

Precondition

In order to use this function the option `WSPI_USE_WAIT` must be enabled.

In order to use this function the driver must have been configured without callbacks (`end_cb = NULL`).

Parameters

in	<i>wspi</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
in	<i>n</i>	number of bytes to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Returns

The operation status.

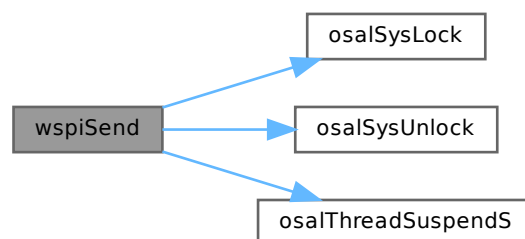
Return values

<i>false</i>	if the operation succeeded.
<i>true</i>	if the operation failed because HW issues.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.27.5.10 wspiReceive()

```
bool wspiReceive (
    WSPIDriver * wspi,
    const wspi_command_t * cmdp,
    size_t n,
    uint8_t * rxbuf)
```

Sends a command then receives data over the WSPI bus.

Precondition

In order to use this function the option `WSPI_USE_WAIT` must be enabled.

In order to use this function the driver must have been configured without callbacks (`end_cb = NULL`).

Parameters

in	<i>wspi</i>	pointer to the <code>WSPIDriver</code> object
in	<i>cmdp</i>	pointer to the command descriptor
in	<i>n</i>	number of bytes to send
out	<i>rxbuf</i>	the pointer to the receive buffer

Returns

The operation status.

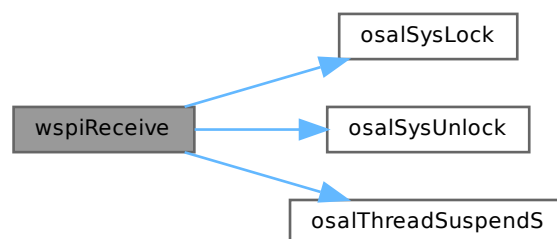
Return values

<i>false</i>	if the operation succeeded.
<i>true</i>	if the operation failed because HW issues.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.27.5.11 wspiMapFlash()

```
void wspiMapFlash (  
    WSPIDriver * wspi,  
    const wspi_command_t * cmdp,  
    uint8_t ** addrp)
```

Maps in memory space a WSPI flash device.

Precondition

The memory flash device must be initialized appropriately before mapping it in memory space.

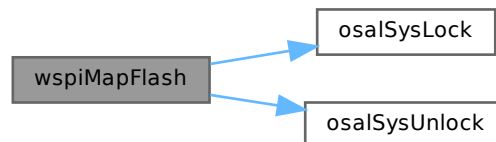
Parameters

in	<i>wspi</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
out	<i>addrp</i>	pointer to the memory start address of the mapped flash or NULL

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.27.5.12 wspiUnmapFlash()

```
void wspiUnmapFlash (  
    WSPIDriver * wspi)
```

Unmaps from memory space a WSPI flash device.

Postcondition

The memory flash device must be re-initialized for normal commands exchange.

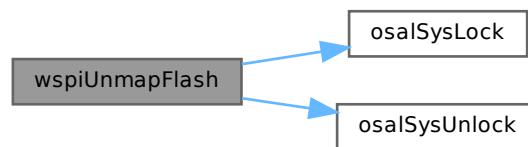
Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.9.27.5.13 wspiAcquireBus()**

```
void wspiAcquireBus (  
    WSPIDriver * wspip)
```

Gains exclusive access to the WSPI bus.

This function tries to gain ownership to the WSPI bus, if the bus is already being used then the invoking thread is queued.

Precondition

In order to use this function the option `WSPi_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.27.5.14 wspiReleaseBus()

```
void wspiReleaseBus (  
    WSPIDriver * wspi)
```

Releases exclusive access to the WSPI bus.

Precondition

In order to use this function the option `WSP_I_USE_MUTUAL_EXCLUSION` must be enabled.

Parameters

in	<i>wspi</i>	pointer to the <code>WSPIDriver</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.9.27.5.15 wspi_lld_init()

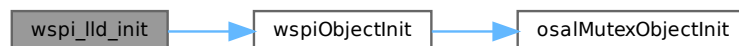
```
void wspi_lld_init (  
    void )
```

Low level WSPI driver initialization.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.9.27.5.16 wspi_lld_start()

```
void wspi_lld_start (  
    WSPIDriver * wspi)
```

Configures and activates the WSPI peripheral.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.5.17 wspi_lld_stop()

```
void wspi_lld_stop (  
    WSPIDriver * wspip)
```

Deactivates the WSPI peripheral.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.5.18 wspi_lld_command()

```
void wspi_lld_command (  
    WSPIDriver * wspip,  
    const wspi\_command\_t * cmdp)
```

Sends a command without data phase.

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.5.19 wspi_lld_send()

```
void wspi_lld_send (  
    WSPIDriver * wspip,  
    const wspi\_command\_t * cmdp,  
    size_t n,  
    const uint8_t * txbuf)
```

Sends a command with data over the WSPI bus.

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
in	<i>n</i>	number of bytes to send
in	<i>txbuf</i>	the pointer to the transmit buffer

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.5.20 wspi_llc_receive()

```
void wspi_llc_receive (
    WSPIDriver * wspip,
    const wspi_command_t * cmdp,
    size_t n,
    uint8_t * rxbuf)
```

Sends a command then receives data over the WSPI bus.

Postcondition

At the end of the operation the configured callback is invoked.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
in	<i>n</i>	number of bytes to send
out	<i>rxbuf</i>	the pointer to the receive buffer

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.5.21 wspi_llc_map_flash()

```
void wspi_llc_map_flash (
    WSPIDriver * wspip,
    const wspi_command_t * cmdp,
    uint8_t ** addrp)
```

Maps in memory space a WSPI flash device.

Precondition

The memory flash device must be initialized appropriately before mapping it in memory space.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object
in	<i>cmdp</i>	pointer to the command descriptor
out	<i>addrp</i>	pointer to the memory start address of the mapped flash or NULL

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.5.22 wspi_lld_unmap_flash()

```
void wspi_lld_unmap_flash (  
    WSPIDriver * wspip)
```

Unmaps from memory space a WSPI flash device.

Postcondition

The memory flash device must be re-initialized for normal commands exchange.

Parameters

in	<i>wspip</i>	pointer to the WSPIDriver object

Function Class:

Not an API, this function is for internal use only.

7.2.9.27.6 Variable Documentation**7.2.9.27.6.1 WSPID1**

```
WSPIDriver WSPID1
```

WSPID1 driver identifier.

7.2.10 Complex Drivers

HAL Complex Drivers.

7.2.10.1 Detailed Description

HAL Complex Drivers.

Topics

- [Buffered SIO Driver](#)
Buffered SIO Driver.
- [Managed Flash Storage Driver](#)
Managed Flash Storage Driver.
- [MMC over SPI Driver](#)
Generic MMC driver.
- [Serial NOR Flash Driver](#)
Serial NOR Flash driver.
- [Serial over USB Driver](#)
Serial over USB Driver.
- [Improved Serial NOR Driver](#)
Serial NOR Flash memories framework. This module provides support for a variety of SNOR devices. Features:

7.2.10.2 Buffered SIO Driver

Buffered SIO Driver.

7.2.10.2.1 Detailed Description

Buffered SIO Driver.

This module implements circular buffers and events on top of a SIO driver. The behavior of the legacy Serial driver is emulated.

Macro Functions

- #define [bsioPutl](#)(bsiop, b)
Direct write to a [BufferedSIODriver](#).
- #define [bsioPut](#)(bsiop, b)
Direct write to a [BufferedSIODriver](#).
- #define [bsioPutTimeout](#)(bsiop, b, t)
Direct write to a [BufferedSIODriver](#) with timeout specification.
- #define [bsioGetl](#)(bsiop)
Direct read from a [BufferedSIODriver](#).
- #define [bsioGet](#)(bsiop)
Direct read from a [BufferedSIODriver](#).
- #define [bsioGetTimeout](#)(bsiop, t)
Direct read from a [BufferedSIODriver](#) with timeout specification.
- #define [bsioWritel](#)(bsiop, b, n)
Direct blocking write to a [BufferedSIODriver](#).
- #define [bsioWrite](#)(bsiop, b, n)
Direct blocking write to a [BufferedSIODriver](#).
- #define [bsioWriteTimeout](#)(bsiop, b, n, t)
Direct blocking write to a [BufferedSIODriver](#) with timeout specification.
- #define [bsioAsynchronousWrite](#)(bsiop, b, n)
Direct non-blocking write to a [BufferedSIODriver](#).
- #define [bsioReadl](#)(bsiop, b, n)

- *Direct blocking read from a [BufferedSIODriver](#).*
- `#define bsioRead(bsiop, b, n)`
Direct blocking read from a [BufferedSIODriver](#).
- `#define bsioReadTimeout(bsiop, b, n, t)`
Direct blocking read from a [BufferedSIODriver](#) with timeout specification.
- `#define bsioAsynchronousRead(bsiop, b, n)`
Direct non-blocking read from a [BufferedSIODriver](#).

Data Structures

- struct [BufferedSIODriverVMT](#)
[BufferedSIODriver](#) virtual methods table.
- struct [hal_buffered_siol_driver](#)
Buffered SIO driver class.

Macros

- `#define __buffered_sio_driver_methods __buffered_serial_methods`
[BufferedSIODriver](#) specific methods.
- `#define __buffered_sio_driver_data`
[SerialDriver](#) specific data.

Typedefs

- typedef struct [hal_buffered_siol_driver](#) [BufferedSIODriver](#)
Buffered SIO driver class.
- typedef [SIOConfig](#) [BufferedSIOConfig](#)
Type of a buffered SIO configuration.

Functions

- static void [__bsio_push_data](#) ([BufferedSIODriver](#) *bsiop)
- static void [__bsio_pop_data](#) ([BufferedSIODriver](#) *bsiop)
- static void [__bsio_default_cb](#) ([SIODriver](#) *siop)
- static void [__bsio_onotify](#) ([io_queue_t](#) *qp)
- static size_t [__bsio_write](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [__bsio_read](#) (void *ip, uint8_t *bp, size_t n)
- static msg_t [__bsio_put](#) (void *ip, uint8_t b)
- static msg_t [__bsio_get](#) (void *ip)
- static msg_t [__bsio_putt](#) (void *ip, uint8_t b, [sysinterval_t](#) timeout)
- static msg_t [__bsio_gett](#) (void *ip, [sysinterval_t](#) timeout)
- static size_t [__bsio_writet](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static size_t [__bsio_readt](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static msg_t [__bsio_ctl](#) (void *ip, unsigned int operation, void *arg)
- void [bsioObjectInit](#) ([BufferedSIODriver](#) *bsiop, [SIODriver](#) *siop, uint8_t *ib, size_t ibsize, uint8_t *ob, size_t obsize)
Initializes a generic serial driver object.
- [msg_t bsioStart](#) ([BufferedSIODriver](#) *bsiop, const [BufferedSIOConfig](#) *config)
Configures and starts the driver.
- void [bsioStop](#) ([BufferedSIODriver](#) *bsiop)
Stops the driver.

Variables

- static const struct [BufferedSIODriverVMT](#) vmt

7.2.10.2.2 Macro Definition Documentation

7.2.10.2.2.1 `__buffered_sio_driver_methods`

```
#define __buffered_sio_driver_methods __buffered_serial_methods
```

[BufferedSIODriver](#) specific methods.

7.2.10.2.2.2 `__buffered_sio_driver_data`

```
#define __buffered_sio_driver_data
```

Value:

```
__buffered_serial_data
SIODriver          *sio;
```

[SerialDriver](#) specific data.

7.2.10.2.2.3 `bsioPutI`

```
#define bsioPutI(
    bsio,
    b)
```

Value:

```
oqPutI(&(bsio)->oqueue, b)
```

Direct write to a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and writes directly on the output queue. This is faster but cannot be used to write to different channels implementations.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.10.2.2.4 bsioPut

```
#define bsioPut(  
    bsiop,  
    b)
```

Value:

```
oqPut(&(bsiop)->oqueue, b)
```

Direct write to a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and writes directly on the output queue. This is faster but cannot be used to write to different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.2.5 bsioPutTimeout

```
#define bsioPutTimeout(  
    bsiop,  
    b,  
    t)
```

Value:

```
oqPutTimeout(&(bsiop)->oqueue, b, t)
```

Direct write to a [BufferedSIODriver](#) with timeout specification.

Note

This function bypasses the indirect access to the channel and writes directly on the output queue. This is faster but cannot be used to write to different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.2.6 bsioGetI

```
#define bsioGetI(  
    bsiop)
```

Value:

```
iqGetI(&(bsiop)->iqueue)
```

Direct read from a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.10.2.2.7 bsioGet

```
#define bsioGet(  
    bsiop)
```

Value:

```
iqGet(&(bsiop)->iqueue)
```

Direct read from a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.2.8 bsioGetTimeout

```
#define bsioGetTimeout(  
    bsiop,  
    t)
```

Value:

```
iqGetTimeout(&(bsiop)->iqueue, t)
```

Direct read from a [BufferedSIODriver](#) with timeout specification.

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.2.9 bsioWriteI

```
#define bsioWriteI(  
    bsiop,  
    b,  
    n)
```

Value:

```
oqWriteI(&(bsiop)->oqueue, b, n)
```

Direct blocking write to a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and writes directly to the output queue. This is faster but cannot be used to write from different channels implementations.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.10.2.2.10 bsioWrite

```
#define bsioWrite(  
    bsiop,  
    b,  
    n)
```

Value:

`oqWriteTimeout` (&(bsiop)->oqueue, *b*, *n*, `TIME_INFINITE`)

Direct blocking write to a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and writes directly to the output queue. This is faster but cannot be used to write from different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.2.11 bsioWriteTimeout

```
#define bsioWriteTimeout(  
    bsiop,  
    b,  
    n,  
    t)
```

Value:

`oqWriteTimeout` (&(bsiop)->oqueue, *b*, *n*, *t*)

Direct blocking write to a [BufferedSIODriver](#) with timeout specification.

Note

This function bypasses the indirect access to the channel and writes directly to the output queue. This is faster but cannot be used to write to different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.2.12 bsioAsynchronousWrite

```
#define bsioAsynchronousWrite(  
    bsiop,  
    b,  
    n)
```

Value:

```
oqWriteTimeout(&(bsiop)->oqueue, b, n, TIME_IMMEDIATE)
```

Direct non-blocking write to a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and writes directly to the output queue. This is faster but cannot be used to write to different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.2.13 bsioReadI

```
#define bsioReadI(  
    bsiop,  
    b,  
    n)
```

Value:

```
iqReadI(&(bsiop)->iqueue, b, n, TIME_INFINITE)
```

Direct blocking read from a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.10.2.2.14 bsioRead

```
#define bsioRead(  
    bsiop,  
    b,  
    n)
```

Value:

`iqReadTimeout(&(bsiop)->iqueue, b, n, TIME_INFINITE)`

Direct blocking read from a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.2.15 bsioReadTimeout

```
#define bsioReadTimeout(  
    bsiop,  
    b,  
    n,  
    t)
```

Value:

`iqReadTimeout(&(bsiop)->iqueue, b, n, t)`

Direct blocking read from a [BufferedSIODriver](#) with timeout specification.

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.2.16 bsioAsynchronousRead

```
#define bsioAsynchronousRead(
    bsiop,
    b,
    n)
```

Value:

```
iqReadTimeout(&(bsiop)->iqueue, b, n, TIME_IMMEDIATE)
```

Direct non-blocking read from a [BufferedSIODriver](#).

Note

This function bypasses the indirect access to the channel and reads directly from the input queue. This is faster but cannot be used to read from different channels implementations.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.2.3 Typedef Documentation

7.2.10.2.3.1 BufferedSIODriver

```
typedef struct hal_buffered_siol_driver BufferedSIODriver
```

Buffered SIO driver class.

7.2.10.2.3.2 BufferedSIOConfig

```
typedef SIOConfig BufferedSIOConfig
```

Type of a buffered SIO configuration.

7.2.10.2.4 Function Documentation

7.2.10.2.4.1 __bsio_push_data()

```
void __bsio_push_data (
    BufferedSIODriver * bsiop) [static]
```

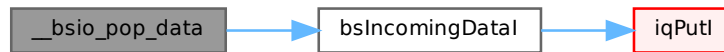
Here is the call graph for this function:



7.2.10.2.4.2 __bsio_pop_data()

```
void __bsio_pop_data (
    BufferedSIODriver * bsio) [static]
```

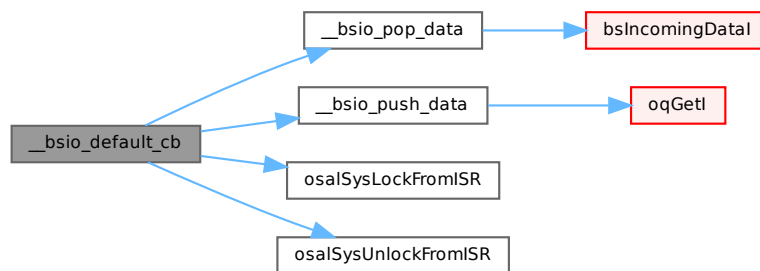
Here is the call graph for this function:



7.2.10.2.4.3 __bsio_default_cb()

```
void __bsio_default_cb (
    SIODriver * siop) [static]
```

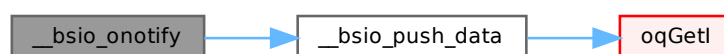
Here is the call graph for this function:



7.2.10.2.4.4 __bsio_onotify()

```
void __bsio_onotify (
    io_queue_t * qp) [static]
```

Here is the call graph for this function:



7.2.10.2.4.5 __bsio_write()

```
size_t __bsio_write (  
    void * ip,  
    const uint8_t * bp,  
    size_t n) [static]
```

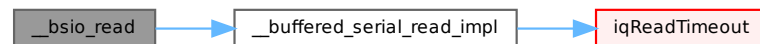
Here is the call graph for this function:



7.2.10.2.4.6 __bsio_read()

```
size_t __bsio_read (  
    void * ip,  
    uint8_t * bp,  
    size_t n) [static]
```

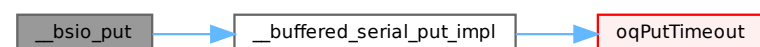
Here is the call graph for this function:



7.2.10.2.4.7 __bsio_put()

```
msg_t __bsio_put (  
    void * ip,  
    uint8_t b) [static]
```

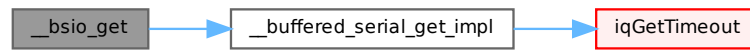
Here is the call graph for this function:



7.2.10.2.4.8 __bsio_get()

```
msg_t __bsio_get (
    void * ip) [static]
```

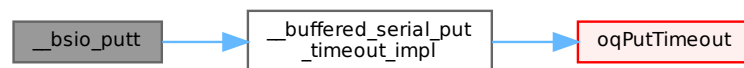
Here is the call graph for this function:



7.2.10.2.4.9 __bsio_putt()

```
msg_t __bsio_putt (
    void * ip,
    uint8_t b,
    sysinterval_t timeout) [static]
```

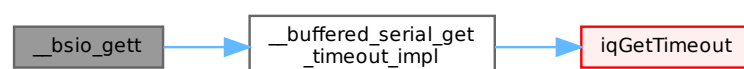
Here is the call graph for this function:



7.2.10.2.4.10 __bsio_gett()

```
msg_t __bsio_gett (
    void * ip,
    sysinterval_t timeout) [static]
```

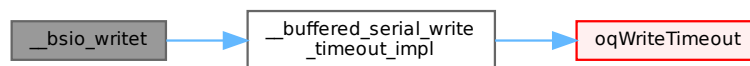
Here is the call graph for this function:



7.2.10.2.4.11 __bsio_writet()

```
size_t __bsio_writet (
    void * ip,
    const uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]
```

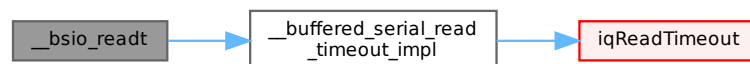
Here is the call graph for this function:



7.2.10.2.4.12 __bsio_readt()

```
size_t __bsio_readt (
    void * ip,
    uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]
```

Here is the call graph for this function:



7.2.10.2.4.13 __bsio_ctl()

```
msg_t __bsio_ctl (
    void * ip,
    unsigned int operation,
    void * arg) [static]
```

7.2.10.2.4.14 bsioObjectInit()

```
void bsioObjectInit (
    BufferedSIODriver * bsiop,
    SIODriver * siop,
    uint8_t * ib,
    size_t ibsize,
    uint8_t * ob,
    size_t obsize)
```

Initializes a generic serial driver object.

The HW dependent part of the initialization has to be performed outside, usually in the hardware initialization code.

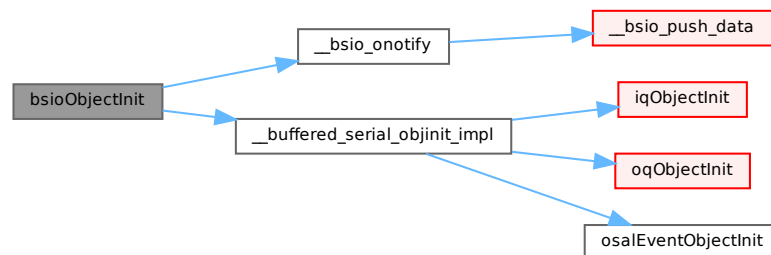
Parameters

out	<i>bsiop</i>	pointer to a BufferedSIODriver structure
in	<i>siop</i>	pointer to the SIODriver object
in	<i>ib</i>	pointer to the input buffer
in	<i>ibsize</i>	size of the input buffer
in	<i>ob</i>	pointer to the output buffer
in	<i>obsize</i>	size of the output buffer

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.10.2.4.15 bsioStart()

```
msg_t bsioStart (
    BufferedSIODriver * bsiop,
    const BufferedSIOConfig * config)
```

Configures and starts the driver.

Parameters

out	<i>bsiop</i>	pointer to a BufferedSIODriver structure
in	<i>config</i>	the architecture-dependent serial driver configuration. If this parameter is set to <code>NULL</code> then a default configuration is used.

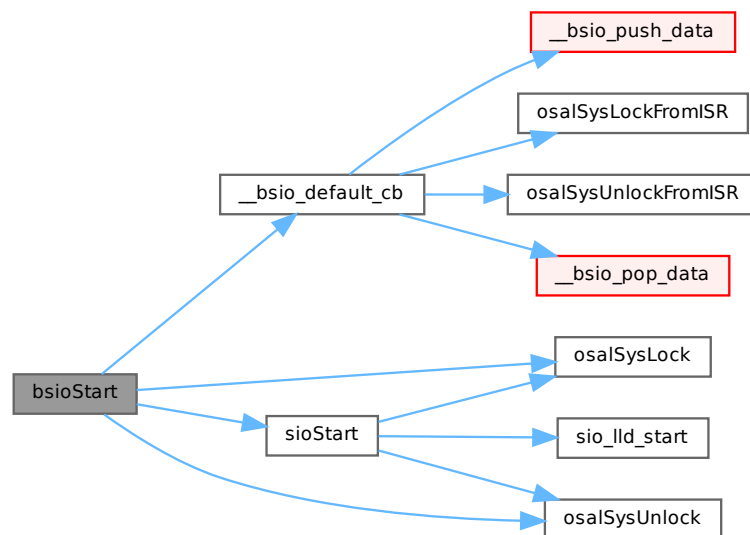
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.2.4.16 bsioStop()

```
void bsioStop (
    BufferedSIODriver * bsiop)
```

Stops the driver.

Any thread waiting on the driver's queues will be awakened with the message `MSG_RESET`.

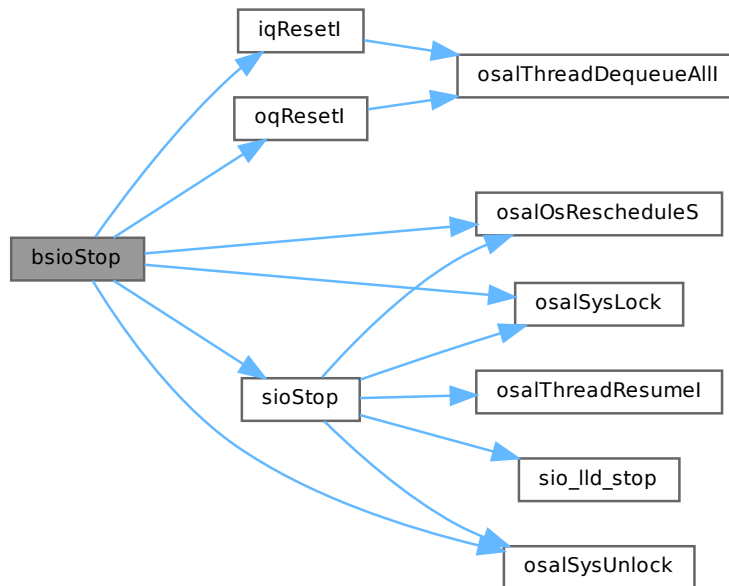
Parameters

out	<i>bsiop</i>	pointer to a BufferedSIODriver structure

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.2.5 Variable Documentation****7.2.10.2.5.1 vmt**

```
const struct BufferedSIODriverVMT vmt [static]
```

Initial value:

```
= {
    (size_t)0,
    __bsio_write, __bsio_read, __bsio_put, __bsio_get,
    __bsio_putt, __bsio_gett, __bsio_writet, __bsio_readt,
    __bsio_ctl
}
```

7.2.10.3 Managed Flash Storage Driver

Managed Flash Storage Driver.

7.2.10.3.1 Detailed Description

Managed Flash Storage Driver.

This module implements a managed flash storage able to store a finite number of variable-size records. Records are retrieved by their index number.

The driver is automatically performs:

- Wear leveling.
- Auto repair after power loss.
- Garbage collection in order to remove erased data.

Configuration options

- #define `MFS_CFG_MAX_RECORDS` 32
Maximum number of indexed records in the managed storage.
- #define `MFS_CFG_MAX_REPAIR_ATTEMPTS` 3
Maximum number of repair attempts on partition mount.
- #define `MFS_CFG_WRITE_VERIFY` TRUE
Verify written data.
- #define `MFS_CFG_STRONG_CHECKING` TRUE
Enables a stronger and slower check procedure on mount.
- #define `MFS_CFG_BUFFER_SIZE` 32
Size of the buffer used for data copying.
- #define `MFS_CFG_MEMORY_ALIGNMENT` 2
Enforced memory alignment.
- #define `MFS_CFG_TRANSACTION_MAX` 16
Maximum number of objects writable in a single transaction.

Error codes handling macros

- #define `MFS_IS_ERROR`(err)
- #define `MFS_IS_WARNING`(err)

Alignment macros

- #define `MFS_ALIGN_MASK` ((uint32_t)`MFS_CFG_MEMORY_ALIGNMENT` - 1U)
- #define `MFS_IS_ALIGNED`(v)
- #define `MFS_ALIGN_PREV`(v)
- #define `MFS_ALIGN_NEXT`(v)

Data Structures

- union [mfs_bank_header_t](#)
Type of a bank header.
- union [mfs_data_header_t](#)
Type of a data block header.
- struct [mfs_record_descriptor_t](#)
- struct [MFSConfig](#)
Type of a MFS configuration structure.
- struct [mfs_transaction_op_t](#)
Type of a buffered write/erase operation within a transaction.
- union [mfs_nocache_buffer](#)
Type of an non-cacheable MFS buffer.
- struct [MFSDriver](#)
Type of an MFS instance.

Macros

- #define [ALIGNED_REC_SIZE](#)(n)
Data record size aligned.
- #define [ALIGNED_DHDR_SIZE](#) [ALIGNED_REC_SIZE](#)(0)
Data record header size aligned.
- #define [ALIGNED_SIZEOF](#)(t)
Aligned size of a type.
- #define [PAIR](#)(a, b)
Combines two values (0..3) in one (0..15).
- #define [RET_ON_ERROR](#)(err)
Error check helper.
- #define [MFS_BANK_MAGIC_1](#) 0xEC705ADEU
- #define [MFS_BANK_MAGIC_2](#) 0xF0339CC5U
- #define [MFS_HEADER_MAGIC_1](#) 0x5FAE45F0U
- #define [MFS_HEADER_MAGIC_2](#) 0xF045AE5FU

Typedefs

- typedef uint32_t [mfs_id_t](#)
Type of a record identifier.
- typedef union [mfs_nocache_buffer](#) [mfs_nocache_buffer_t](#)
Type of an non-cacheable MFS buffer.

Enumerations

- enum [mfs_bank_t](#) { [MFS_BANK_0](#) = 0 , [MFS_BANK_1](#) = 1 }
 - enum [mfs_state_t](#) {
 [MFS_UNINIT](#) = 0 , [MFS_STOP](#) = 1 , [MFS_READY](#) = 2 , [MFS_TRANSACTION](#) = 3 ,
 [MFS_ERROR](#) = 4 }
- Type of driver state machine states.*

- enum `mfs_error_t` {
`MFS_NO_ERROR` = 0 , `MFS_WARN_REPAIR` = 1 , `MFS_WARN_GC` = 2 , `MFS_ERR_INV_STATE` = -1 ,
`MFS_ERR_INV_SIZE` = -2 , `MFS_ERR_NOT_FOUND` = -3 , `MFS_ERR_OUT_OF_MEM` = -4 ,
`MFS_ERR_TRANSACTION_NUM` = -5 ,
`MFS_ERR_TRANSACTION_SIZE` = -6 , `MFS_ERR_NOT_ERASED` = -7 , `MFS_ERR_FLASH_FAILURE` =
-8 , `MFS_ERR_INTERNAL` = -9 }
Type of an MFS error code.
- enum `mfs_bank_state_t` { `MFS_BANK_ERASED` = 0 , `MFS_BANK_OK` = 1 , `MFS_BANK_GARBAGE` = 2 }
Type of a bank state assessment.

Functions

- `uint16_t crc16` (`uint16_t` crc, `const uint8_t *`data, `size_t` n)
- static void `mfs_state_reset` (`MFSDriver *`mfsp)
- static `flash_offset_t` `mfs_flash_get_bank_offset` (`MFSDriver *`mfsp, `mfs_bank_t` bank)
- static `mfs_error_t` `mfs_flash_read` (`MFSDriver *`mfsp, `flash_offset_t` offset, `size_t` n, `uint8_t *`rp)
Flash read.
- static `mfs_error_t` `mfs_flash_write` (`MFSDriver *`mfsp, `flash_offset_t` offset, `size_t` n, `const uint8_t *`wp)
Flash write.
- static `mfs_error_t` `mfs_flash_copy` (`MFSDriver *`mfsp, `flash_offset_t` doffset, `flash_offset_t` soffset, `uint32_t` n)
Flash copy.
- static `mfs_error_t` `mfs_bank_erase` (`MFSDriver *`mfsp, `mfs_bank_t` bank)
Erases and verifies all sectors belonging to a bank.
- static `mfs_error_t` `mfs_bank_verify_erase` (`MFSDriver *`mfsp, `mfs_bank_t` bank)
Erases and verifies all sectors belonging to a bank.
- static `mfs_error_t` `mfs_bank_write_header` (`MFSDriver *`mfsp, `mfs_bank_t` bank, `uint32_t` cnt)
Writes the validation header in a bank.
- static `mfs_bank_state_t` `mfs_bank_check_header` (`MFSDriver *`mfsp)
Checks integrity of the header in the shared buffer.
- static `mfs_error_t` `mfs_bank_scan_records` (`MFSDriver *`mfsp, `mfs_bank_t` bank, `bool *`wflagp)
Scans blocks searching for records.
- static `mfs_error_t` `mfs_bank_get_state` (`MFSDriver *`mfsp, `mfs_bank_t` bank, `mfs_bank_state_t *`statep, `uint32_t *`cntp)
Determines the state of a bank.
- static `mfs_error_t` `mfs_garbage_collect` (`MFSDriver *`mfsp)
Enforces a garbage collection.
- static `mfs_error_t` `mfs_try_mount` (`MFSDriver *`mfsp)
Performs a flash partition mount attempt.
- `mfs_error_t` `mfs_mount` (`MFSDriver *`mfsp)
Configures and activates a MFS driver.
- void `mfsObjectInit` (`MFSDriver *`mfsp, `mfs_nocache_buffer_t *`ncbuf)
Initializes an instance.
- `mfs_error_t` `mfsStart` (`MFSDriver *`mfsp, `const MFSCfg *`config)
Configures and activates a MFS driver.
- void `mfsStop` (`MFSDriver *`mfsp)
Deactivates a MFS driver.
- `mfs_error_t` `mfsErase` (`MFSDriver *`mfsp)
Destroys the state of the managed storage by erasing the flash.
- `mfs_error_t` `mfsReadRecord` (`MFSDriver *`mfsp, `mfs_id_t` id, `size_t` np, `uint8_t *`buffer)
Retrieves and reads a data record.
- `mfs_error_t` `mfsWriteRecord` (`MFSDriver *`mfsp, `mfs_id_t` id, `size_t` n, `const uint8_t *`buffer)

- Creates or updates a data record.*

 - `mfs_error_t mfsEraseRecord (MFSDriver *mfsp, mfs_id_t id)`

Erases a data record.
- `mfs_error_t mfsPerformGarbageCollection (MFSDriver *mfsp)`

Enforces a garbage collection operation.
- `mfs_error_t mfsStartTransaction (MFSDriver *mfsp, size_t size)`

Puts the driver in transaction mode.
- `mfs_error_t mfsCommitTransaction (MFSDriver *mfsp)`

A transaction is committed and finalized atomically.
- `mfs_error_t mfsRollbackTransaction (MFSDriver *mfsp)`

A transaction is rolled back atomically.

Variables

- static const uint16_t `crc16_table` [256]

7.2.10.3.2 Macro Definition Documentation

7.2.10.3.2.1 ALIGNED_REC_SIZE

```
#define ALIGNED_REC_SIZE(  
    n)
```

Value:

```
(flash_offset_t)MFS_ALIGN_NEXT(sizeof (mfs_data_header_t) + (size_t) (n))
```

Data record size aligned.

7.2.10.3.2.2 ALIGNED_DHDR_SIZE

```
#define ALIGNED_DHDR_SIZE  ALIGNED_REC_SIZE(0)
```

Data record header size aligned.

7.2.10.3.2.3 ALIGNED_SIZEOF

```
#define ALIGNED_SIZEOF(  
    t)
```

Value:

```
((sizeof (t) - 1U) | MFS_ALIGN_MASK) + 1U)
```

Aligned size of a type.

7.2.10.3.2.4 PAIR

```
#define PAIR(  
    a,  
    b)
```

Value:

`((unsigned)(a) << 2U) | (unsigned)(b)`

Combines two values (0..3) in one (0..15).

7.2.10.3.2.5 RET_ON_ERROR

```
#define RET_ON_ERROR(  
    err)
```

Value:

```
do {  
    mfs_error_t e = (err);  
    if (e != MFS_NO_ERROR) {  
        return e;  
    }  
} while (false)
```

\\
\\
\\
\\

Error check helper.

7.2.10.3.2.6 MFS_BANK_MAGIC_1

```
#define MFS_BANK_MAGIC_1 0xEC705ADEU
```

7.2.10.3.2.7 MFS_BANK_MAGIC_2

```
#define MFS_BANK_MAGIC_2 0xF0339CC5U
```

7.2.10.3.2.8 MFS_HEADER_MAGIC_1

```
#define MFS_HEADER_MAGIC_1 0x5FAE45F0U
```

7.2.10.3.2.9 MFS_HEADER_MAGIC_2

```
#define MFS_HEADER_MAGIC_2 0xF045AE5FU
```

7.2.10.3.2.10 MFS_CFG_MAX_RECORDS

```
#define MFS_CFG_MAX_RECORDS 32
```

Maximum number of indexed records in the managed storage.

Note

Record indexes go from 1 to `MFS_CFG_MAX_RECORDS`.

7.2.10.3.2.11 MFS_CFG_MAX_REPAIR_ATTEMPTS

```
#define MFS_CFG_MAX_REPAIR_ATTEMPTS 3
```

Maximum number of repair attempts on partition mount.

7.2.10.3.2.12 MFS_CFG_WRITE_VERIFY

```
#define MFS_CFG_WRITE_VERIFY TRUE
```

Verify written data.

7.2.10.3.2.13 MFS_CFG_STRONG_CHECKING

```
#define MFS_CFG_STRONG_CHECKING TRUE
```

Enables a stronger and slower check procedure on mount.

Strong checking requires reading of the whole written data and this can be slow, normal checking only checks integrity of metadata, data errors would be detected on read.

7.2.10.3.2.14 MFS_CFG_BUFFER_SIZE

```
#define MFS_CFG_BUFFER_SIZE 32
```

Size of the buffer used for data copying.

Note

The buffer size must be a power of two and not smaller than 16 bytes.

Larger buffers improve performance, buffers with size multiple of the flash program page size work better.

7.2.10.3.2.15 MFS_CFG_MEMORY_ALIGNMENT

```
#define MFS_CFG_MEMORY_ALIGNMENT 2
```

Enforced memory alignment.

This value must be a power of two, it enforces a memory alignment for records in the flash array. This is required when alignment constraints exist, for example when using a DTR mode on OSPI devices.

7.2.10.3.2.16 MFS_CFG_TRANSACTION_MAX

```
#define MFS_CFG_TRANSACTION_MAX 16
```

Maximum number of objects writable in a single transaction.

7.2.10.3.2.17 MFS_IS_ERROR

```
#define MFS_IS_ERROR(  
    err)
```

Value:

```
((err) < MFS_NO_ERROR)
```

7.2.10.3.2.18 MFS_IS_WARNING

```
#define MFS_IS_WARNING(  
    err)
```

Value:

```
((err) > MFS_NO_ERROR)
```

7.2.10.3.2.19 MFS_ALIGN_MASK

```
#define MFS_ALIGN_MASK ((uint32_t)MFS_CFG_MEMORY_ALIGNMENT - 1U)
```

7.2.10.3.2.20 MFS_IS_ALIGNED

```
#define MFS_IS_ALIGNED(  
    v)
```

Value:

```
((uint32_t)(v) & MFS_ALIGN_MASK) == 0U)
```

7.2.10.3.2.21 MFS_ALIGN_PREV

```
#define MFS_ALIGN_PREV(  
    v)
```

Value:

```
((uint32_t)(v) & ~MFS_ALIGN_MASK)
```

7.2.10.3.2.22 MFS_ALIGN_NEXT

```
#define MFS_ALIGN_NEXT(  
    v)
```

Value:

```
(MFS_ALIGN_PREV(((uint32_t)(v) - 1U)) +  
MFS_CFG_MEMORY_ALIGNMENT)
```


7.2.10.3.3 Typedef Documentation

7.2.10.3.3.1 mfs_id_t

```
typedef uint32_t mfs_id_t
```

Type of a record identifier.

7.2.10.3.3.2 mfs_nocache_buffer_t

```
typedef union mfs_nocache_buffer mfs_nocache_buffer_t
```

Type of an non-cacheable MFS buffer.

7.2.10.3.4 Enumeration Type Documentation

7.2.10.3.4.1 mfs_bank_t

```
enum mfs_bank_t
```

Type of a flash bank.

Enumerator

MFS_BANK↔ _0	
MFS_BANK↔ _1	

7.2.10.3.4.2 mfs_state_t

```
enum mfs_state_t
```

Type of driver state machine states.

Enumerator

MFS_UNINIT	
MFS_STOP	
MFS_READY	
MFS_TRANSACTION	
MFS_ERROR	

7.2.10.3.4.3 mfs_error_t

```
enum mfs_error_t
```

Type of an MFS error code.

Note

Errors are negative integers, informative warnings are positive integers.

Enumerator

MFS_NO_ERROR	
MFS_WARN_REPAIR	
MFS_WARN_GC	
MFS_ERR_INV_STATE	
MFS_ERR_INV_SIZE	
MFS_ERR_NOT_FOUND	
MFS_ERR_OUT_OF_MEM	
MFS_ERR_TRANSACTION_NUM	
MFS_ERR_TRANSACTION_SIZE	
MFS_ERR_NOT_ERASED	
MFS_ERR_FLASH_FAILURE	
MFS_ERR_INTERNAL	

7.2.10.3.4 mfs_bank_state_t

```
enum mfs_bank_state_t
```

Type of a bank state assessment.

Enumerator

MFS_BANK_ERASED	
MFS_BANK_OK	
MFS_BANK_GARBAGE	

7.2.10.3.5 Function Documentation

7.2.10.3.5.1 crc16()

```
uint16_t crc16 (
    uint16_t crc,
    const uint8_t * data,
    size_t n)
```

7.2.10.3.5.2 mfs_state_reset()

```
void mfs_state_reset (
    MFSDriver * mfsp) [static]
```

7.2.10.3.5.3 mfs_flash_get_bank_offset()

```
flash_offset_t mfs_flash_get_bank_offset (
    MFSDriver * mfsp,
    mfs_bank_t bank) [static]
```

Here is the call graph for this function:



7.2.10.3.5.4 mfs_flash_read()

```
mfs_error_t mfs_flash_read (
    MFSDriver * mfsp,
    flash_offset_t offset,
    size_t n,
    uint8_t * rp) [static]
```

Flash read.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>offset</i>	flash offset
in	<i>n</i>	number of bytes to be read
out	<i>rp</i>	pointer to the data buffer

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

7.2.10.3.5.5 mfs_flash_write()

```
mfs_error_t mfs_flash_write (
    MFSDriver * mfsp,
    flash_offset_t offset,
    size_t n,
    const uint8_t * wp) [static]
```

Flash write.

Note

If the option MFS_CFG_WRITE_VERIFY is enabled then the flash is also read back for verification.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>offset</i>	flash offset
in	<i>n</i>	number of bytes to be written
in	<i>wp</i>	pointer to the data buffer

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.3.5.6 mfs_flash_copy()**

```
mfs_error_t mfs_flash_copy (  
    MFSDriver * mfsp,  
    flash_offset_t doffset,  
    flash_offset_t soffset,  
    uint32_t n) [static]
```

Flash copy.

Note

If the option MFS_CFG_WRITE_VERIFY is enabled then the flash is also read back for verification.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>doffset</i>	destination flash offset
in	<i>soffset</i>	source flash offset
in	<i>n</i>	number of bytes to be copied

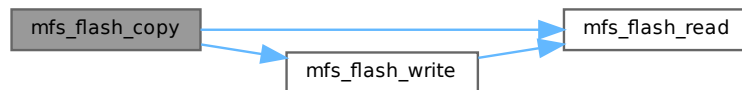
Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.3.5.7 mfs_bank_erase()**

```

mfs_error_t mfs_bank_erase (
    MFSDriver * mfsp,
    mfs_bank_t bank) [static]
  
```

Erases and verifies all sectors belonging to a bank.

Parameters

in	<i>mfsp</i>	pointer to the <code>MFSDriver</code> object
in	<i>bank</i>	bank to be erased

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.3.5.8 mfs_bank_verify_erase()

```
mfs_error_t mfs_bank_verify_erase (  
    MFSDriver * mfsp,  
    mfs_bank_t bank) [static]
```

Erases and verifies all sectors belonging to a bank.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>bank</i>	bank to be verified

Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

7.2.10.3.5.9 mfs_bank_write_header()

```
mfs_error_t mfs_bank_write_header (  
    MFSDriver * mfsp,  
    mfs_bank_t bank,  
    uint32_t cnt) [static]
```

Writes the validation header in a bank.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>bank</i>	bank to be validated
in	<i>cnt</i>	value for the flash usage counter

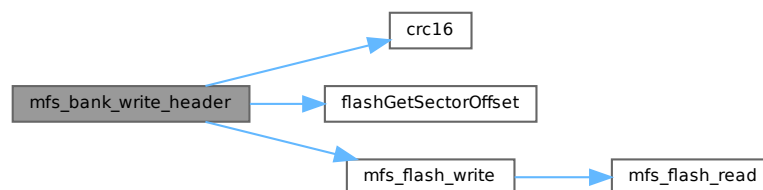
Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.3.5.10 mfs_bank_check_header()

```
mfs_bank_state_t mfs_bank_check_header (  
    MFSDriver * mfsp) [static]
```

Checks integrity of the header in the shared buffer.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object

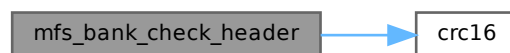
Returns

The header state.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.3.5.11 mfs_bank_scan_records()

```
mfs_error_t mfs_bank_scan_records (  
    MFSDriver * mfsp,  
    mfs_bank_t bank,  
    bool * wflagp) [static]
```

Scans blocks searching for records.

Note

The block integrity is strongly checked.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>bank</i>	the bank identifier
out	<i>wflagp</i>	warning flag on anomalies

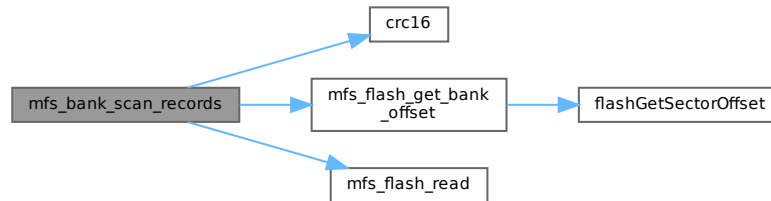
Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.3.5.12 mfs_bank_get_state()**

```

mfs_error_t mfs_bank_get_state (
    MFSDriver * mfsp,
    mfs_bank_t bank,
    mfs_bank_state_t * statep,
    uint32_t * cntp) [static]
  
```

Determines the state of a bank.

Note

This function does not test the bank integrity by scanning the data area, it just checks the header.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>bank</i>	bank to be checked
out	<i>statep</i>	bank state, it can be: • MFS_BANK_ERASED • MFS_BANK_GARBAGE • MFS_BANK_OK
out	<i>cntp</i>	bank counter

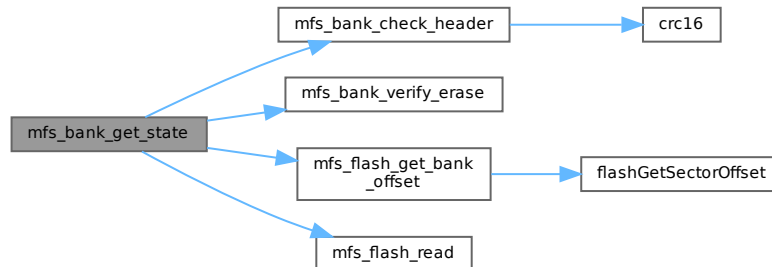
Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.3.5.13 mfs_garbage_collect()**

```

mfs_error_t mfs_garbage_collect (
    MFSDriver * mfsp) [static]
  
```

Enforces a garbage collection.

Storage data is compacted into a single bank.

Parameters

out	<i>mfsp</i>	pointer to the MFSDriver object

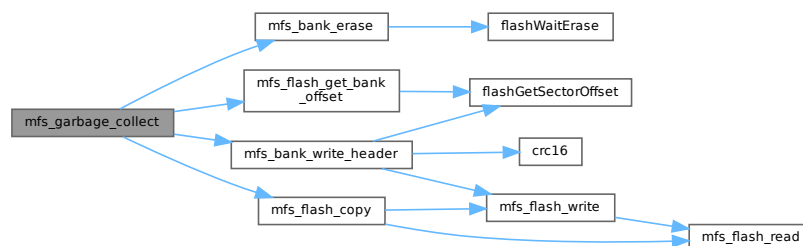
Returns

The operation status.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



Parameters

in	<i>mfs</i>	pointer to the MFSDriver object

Returns

The operation status.

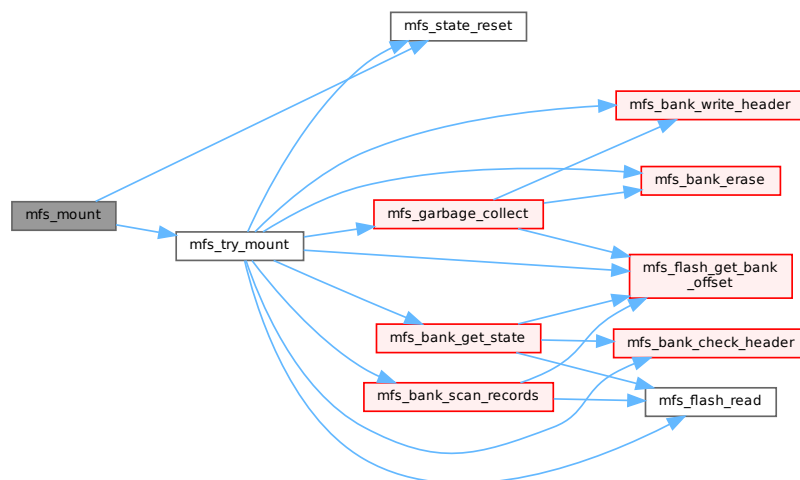
Return values

<i>MFS_NO_ERROR</i>	if the operation has been successfully completed.
<i>MFS_WARN_GC</i>	if the operation triggered a garbage collection.
<i>MFS_ERR_FLASH_FAILURE</i>	if the flash memory is unusable because HW failures. Makes the driver enter the <i>MFS_ERROR</i> state.
<i>MFS_ERR_INTERNAL</i>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.3.5.16 mfsObjectInit()

```

void mfsObjectInit (
    MFSDriver * mfs,
    mfs_nocache_buffer_t * ncbuf)

```

Initializes an instance.

Parameters

out	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>ncbuf</i>	pointer to an mfs_nocache_buffer_t buffer structure

Function Class:

Object or module initializer function.

7.2.10.3.5.17 mfsStart()

```
mfs_error_t mfsStart (
    MFSDriver * mfsp,
    const MFSCfg * config)
```

Configures and activates a MFS driver.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>config</i>	pointer to the configuration

Returns

The operation status.

Return values

<i>MFS_NO_ERROR</i>	if the operation has been completed.
<i>MFS_WARN_GC</i>	if the operation triggered a garbage collection.
<i>MFS_ERR_FLASH_FAILURE</i>	if the flash memory is unusable because HW failures. Makes the driver enter the <i>MFS_ERROR</i> state.
<i>MFS_ERR_INTERNAL</i>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.3.5.18 mfsStop()

```
void mfsStop (
    MFSDriver * mfsp)
```

Deactivates a MFS driver.

Parameters

in	<i>mfs</i>	pointer to the MFSDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.3.5.19 mfsErase()

```
mfs_error_t mfsErase (
    MFSDriver * mfsp)
```

Destroys the state of the managed storage by erasing the flash.

Parameters

in	<i>mfs</i>	pointer to the MFSDriver object

Returns

The operation status.

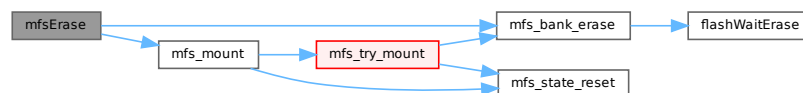
Return values

<i>MFS_ERR_INV_STATE</i>	if the driver is in not in <i>MFS_READY</i> state.
<i>MFS_NO_ERROR</i>	if the operation has been successfully completed.
<i>MFS_ERR_FLASH_FAILURE</i>	if the flash memory is unusable because HW failures. Makes the driver enter the <i>MFS_ERROR</i> state.
<i>MFS_ERR_INTERNAL</i>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.3.5.20 mfsReadRecord()

```
mfs_error_t mfsReadRecord (
    MFSDriver * mfsp,
    mfs_id_t id,
    size_t * np,
    uint8_t * buffer)
```

Retrieves and reads a data record.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object
in	<i>id</i>	record numeric identifier, the valid range is between 1 and <code>MFS_CFG_MAX_RECORDS</code>
in, out	<i>np</i>	on input is the maximum buffer size, on return it is the size of the data copied into the buffer
out	<i>buffer</i>	pointer to a buffer for record data

Returns

The operation status.

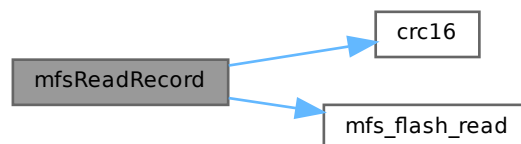
Return values

<code>MFS_NO_ERROR</code>	if the operation has been successfully completed.
<code>MFS_ERR_INV_STATE</code>	if the driver is in not in <code>MFS_READY</code> state.
<code>MFS_ERR_INV_SIZE</code>	if the passed buffer is not large enough to contain the record data.
<code>MFS_ERR_NOT_FOUND</code>	if the specified id does not exists.
<code>MFS_ERR_FLASH_FAILURE</code>	if the flash memory is unusable because HW failures. Makes the driver enter the <code>MFS_ERROR</code> state.
<code>MFS_ERR_INTERNAL</code>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

7.2.10.3.5.21 `mfsWriteRecord()`

```

mfs_error_t mfsWriteRecord (
    MFSDriver * mfsp,
    mfs_id_t id,
    size_t n,
    const uint8_t * buffer)
  
```

Creates or updates a data record.

Parameters

in	<i>mfs</i>	pointer to the MFSDriver object
in	<i>id</i>	record numeric identifier, the valid range is between 1 and <code>MFS_CFG_MAX_RECORDS</code>
in	<i>n</i>	size of data to be written, it cannot be zero
in	<i>buffer</i>	pointer to a buffer for record data

Returns

The operation status.

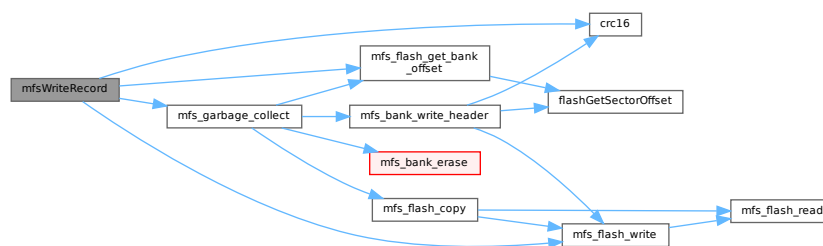
Return values

<code>MFS_NO_ERROR</code>	if the operation has been successfully completed.
<code>MFS_WARN_GC</code>	if the operation triggered a garbage collection.
<code>MFS_ERR_INV_STATE</code>	if the driver is in not in <code>MFS_READY</code> state.
<code>MFS_ERR_OUT_OF_MEM</code>	if there is not enough flash space for the operation.
<code>MFS_ERR_TRANSACTION_NUM</code>	if the transaction operations buffer space has been exceeded.
<code>MFS_ERR_TRANSACTION_SIZE</code>	if the transaction allocated space has been exceeded.
<code>MFS_ERR_FLASH_FAILURE</code>	if the flash memory is unusable because HW failures. Makes the driver enter the <code>MFS_ERROR</code> state.
<code>MFS_ERR_INTERNAL</code>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.3.5.22 mfsEraseRecord()

```

mfs_error_t mfsEraseRecord (
    MFSDriver * mfs,
    mfs_id_t id)

```

Erases a data record.

Parameters

in	<i>mfs</i>	pointer to the MFSDriver object
in	<i>id</i>	record numeric identifier, the valid range is between 1 and <code>MFS_CFG_MAX_RECORDS</code>

Returns

The operation status.

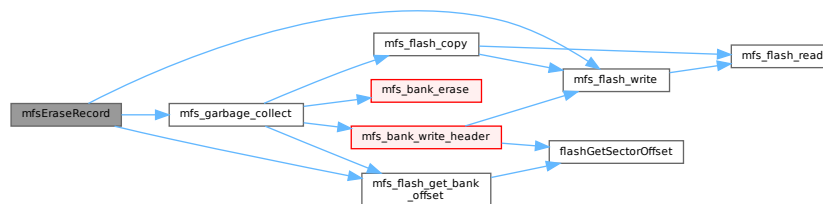
Return values

<code>MFS_NO_ERROR</code>	if the operation has been successfully completed.
<code>MFS_WARN_GC</code>	if the operation triggered a garbage collection.
<code>MFS_ERR_INV_STATE</code>	if the driver is in not in <code>MFS_READY</code> state.
<code>MFS_ERR_OUT_OF_MEM</code>	if there is not enough flash space for the operation.
<code>MFS_ERR_TRANSACTION_NUM</code>	if the transaction operations buffer space has been exceeded.
<code>MFS_ERR_TRANSACTION_SIZE</code>	if the transaction allocated space has been exceeded.
<code>MFS_ERR_FLASH_FAILURE</code>	if the flash memory is unusable because HW failures. Makes the driver enter the <code>MFS_ERROR</code> state.
<code>MFS_ERR_INTERNAL</code>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.3.5.23 mfsPerformGarbageCollection()

```

mfs_error_t mfsPerformGarbageCollection (
    MFSDriver * mfs)

```

Enforces a garbage collection operation.

Garbage collection involves: integrity check, optionally repairs, obsolete data removal, data compaction and a flash bank swap.

Parameters

in	<i>mfsp</i>	pointer to the MFSDriver object

Returns

The operation status.

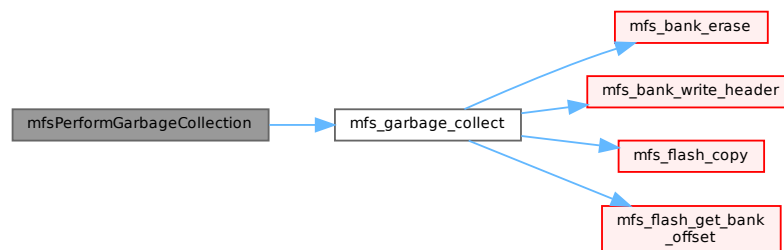
Return values

<i>MFS_NO_ERROR</i>	if the operation has been successfully completed.
<i>MFS_ERR_INV_STATE</i>	if the driver is in not in <i>MFS_READY</i> state.
<i>MFS_ERR_FLASH_FAILURE</i>	if the flash memory is unusable because HW failures. Makes the driver enter the <i>MFS_ERROR</i> state.
<i>MFS_ERR_INTERNAL</i>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.3.5.24 mfsStartTransaction()

```

mfs_error_t mfsStartTransaction (
    MFSDriver * mfsp,
    size_t size)

```

Puts the driver in transaction mode.

Note

The parameters `n` and `size` are used to make an estimation of the space required for the transaction to succeed. Note that the estimated size must include also the extra space required by alignment enforcement option. If the estimated size is wrong (by defect) what could happen is that there is a failure in the middle of a transaction and a roll-back would be required.

The conditions for starting a transaction are:

- The driver must be started.
- There must be enough compacted storage to accommodate the whole transaction. If the required space is available but it is not compacted then a garbage collect operation is performed.

Parameters

in	<i>mfs</i>	pointer to the MFSDriver object
in	<i>size</i>	estimated total size of written records in transaction, this includes, data, headers and alignment gaps

Returns

The operation status.

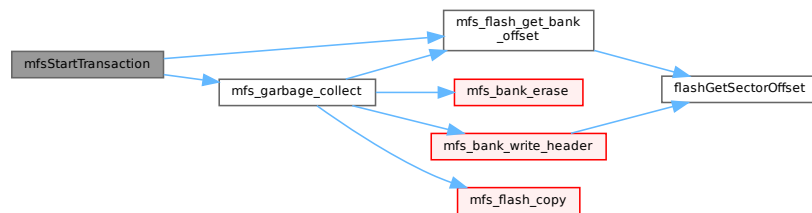
Return values

<i>MFS_NO_ERROR</i>	if the operation has been successfully completed.
<i>MFS_ERR_INV_STATE</i>	if the driver is in not in <i>MFS_READY</i> state.
<i>MFS_ERR_FLASH_FAILURE</i>	if the flash memory is unusable because HW failures. Makes the driver enter the <i>MFS_ERROR</i> state.
<i>MFS_ERR_INTERNAL</i>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.3.5.25 mfsCommitTransaction()**

```

mfs_error_t mfsCommitTransaction (
    MFSDriver * mfsp)

```

A transaction is committed and finalized atomically.

Parameters

in	<i>mfs</i>	pointer to the MFSDriver object

Returns

The operation status.

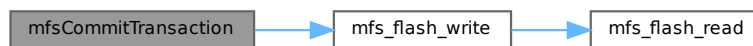
Return values

<i>MFS_NO_ERROR</i>	if the operation has been successfully completed.
<i>MFS_ERR_INV_STATE</i>	if the driver is in not in <i>MFS_TRANSACTION</i> state.
<i>MFS_ERR_FLASH_FAILURE</i>	if the flash memory is unusable because HW failures. Makes the driver enter the <i>MFS_ERROR</i> state.
<i>MFS_ERR_INTERNAL</i>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.3.5.26 mfsRollbackTransaction()

```

mfs_error_t mfsRollbackTransaction (
    MFSDriver * mfsp)
  
```

A transaction is rolled back atomically.

This function performs a garbage collection in order to discard all written data that has not been finalized.

Parameters

in	<i>mfsp</i>	pointer to the <i>MFSDriver</i> object

Returns

The operation status.

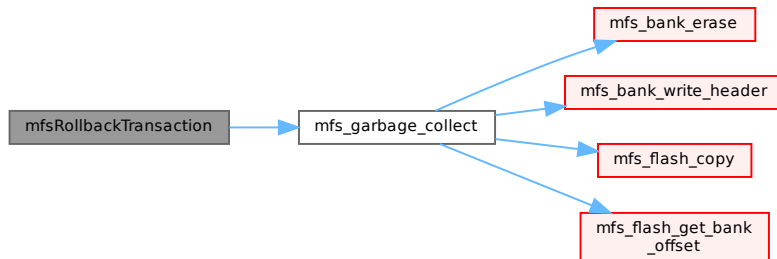
Return values

<i>MFS_NO_ERROR</i>	if the operation has been successfully completed.
<i>MFS_ERR_INV_STATE</i>	if the driver is in not in <i>MFS_TRANSACTION</i> state.
<i>MFS_ERR_FLASH_FAILURE</i>	if the flash memory is unusable because HW failures. Makes the driver enter the <i>MFS_ERROR</i> state.
<i>MFS_ERR_INTERNAL</i>	if an internal logic failure is detected.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.3.6 Variable Documentation****7.2.10.3.6.1 crc16_table**

```
const uint16_t crc16_table[256] [static]
```

7.2.10.4 MMC over SPI Driver

Generic MMC driver.

7.2.10.4.1 Detailed Description

Generic MMC driver.

This module implements a portable MMC/SD driver that uses a SPI driver as physical layer. Hot plugging and removal are supported through kernel events.

Precondition

In order to use the MMC_SPI driver the `HAL_USE_MMC_SPI` and `HAL_USE_SPI` options must be enabled in `halconf.h`.

7.2.10.4.2 Driver State Machine

This driver implements a state machine internally, see the [Abstract I/O Block Device](#) module documentation for details.

7.2.10.4.3 Driver Operations

This driver allows to read or write single or multiple 512 bytes blocks on a SD Card.

MMC_SPI configuration options

- #define `MMC_IDLE_TIMEOUT_MS` 1000
Timeout before assuming a failure while waiting for card idle.
- #define `MMC_USE_MUTUAL_EXCLUSION` TRUE
Mutual exclusion on the SPI bus.

Macro Functions

- #define `mmclsCardInserted`(mmcp)
Returns the card insertion status.
- #define `mmclsWriteProtected`(mmcp)
Returns the write protect status.

Data Structures

- struct `mmc_spi_config_t`
Type of a MMC/SD over SPI driver configuration structure.
- struct `mmc_spi_driver_vmt`
MMCDriver virtual methods table.
- struct `mmc_spi_driver_t`
Structure representing a MMC/SD over SPI driver.

Macros

- #define `MMC_CMD0_RETRY` 10U
- #define `MMC_CMD1_RETRY` 100U
- #define `MMC_ACMD41_RETRY` 100U
- #define `MMC_WAIT_DATA` 10000U
- #define `MMC_BUFFER_SIZE` 16U
Size of the buffer to be supplied to the driver.
- #define `__mmc_driver_methods` `_mmcsd_block_device_methods`
MMCDriver specific methods.

Typedefs

- typedef `mmc_spi_config_t` `MMConfig`
Legacy name for compatibility.
- typedef `mmc_spi_driver_t` `MMCDriver`
Legacy name for compatibility.

Functions

- static bool [mmc_is_card_inserted](#) (void *instance)
- static bool [mmc_is_write_protected](#) (void *instance)
- static bool [mmc_connect](#) (void *instance)
- static bool [mmc_disconnect](#) (void *instance)
- static bool [mmc_read](#) (void *instance, uint32_t startblk, uint8_t *buffer, uint32_t n)
- static bool [mmc_write](#) (void *instance, uint32_t startblk, const uint8_t *buffer, uint32_t n)
- static bool [mmc_sync](#) (void *instance)
- static bool [mmc_get_info](#) (void *instance, [BlockDeviceInfo](#) *bdip)
- static uint8_t [mmc_crc7](#) (uint8_t crc, const uint8_t *buffer, size_t len)
Calculate the MMC standard CRC-7 based on a lookup table.
- static bool [mmc_wait_idle](#) ([MMCDriver](#) *mmcp)
Waits an idle condition.
- static bool [mmc_send_hdr](#) ([MMCDriver](#) *mmcp, uint8_t cmd, uint32_t arg)
Sends a command header.
- static bool [mmc_recvr1](#) ([MMCDriver](#) *mmcp, uint8_t *r1p)
Receives a single byte response.
- static bool [mmc_recvr3](#) ([MMCDriver](#) *mmcp, uint8_t *r1p)
Receives a three byte response.
- static bool [mmc_send_command_R1](#) ([MMCDriver](#) *mmcp, uint8_t cmd, uint32_t arg, uint8_t *r1p)
Sends a command and returns a single byte response.
- static bool [mmc_send_command_R3](#) ([MMCDriver](#) *mmcp, uint8_t cmd, uint32_t arg, uint8_t *r1p)
Sends a command which returns a five bytes response (R3).
- static bool [mmc_read_CxD](#) ([MMCDriver](#) *mmcp, uint8_t cmd, uint32_t cxd[4])
Reads the CSD.
- void [mmcInit](#) (void)
MMC over SPI driver initialization.
- void [mmcObjectInit](#) ([MMCDriver](#) *mmcp, uint8_t *buffer)
Initializes an instance.
- [msg_t](#) [mmcStart](#) ([MMCDriver](#) *mmcp, const [MMCCfg](#) *config)
Configures and activates the MMC peripheral.
- void [mmcStop](#) ([MMCDriver](#) *mmcp)
Disables the MMC peripheral.
- bool [mmcConnect](#) ([MMCDriver](#) *mmcp)
Performs the initialization procedure on the inserted card.
- bool [mmcDisconnect](#) ([MMCDriver](#) *mmcp)
Brings the driver in a state safe for card removal.
- bool [mmcStartSequentialRead](#) ([MMCDriver](#) *mmcp, uint32_t startblk)
Starts a sequential read.
- bool [mmcSequentialRead](#) ([MMCDriver](#) *mmcp, uint8_t *buffer)
Reads a block within a sequential read operation.
- bool [mmcStopSequentialRead](#) ([MMCDriver](#) *mmcp)
Stops a sequential read gracefully.
- bool [mmcStartSequentialWrite](#) ([MMCDriver](#) *mmcp, uint32_t startblk)
Starts a sequential write.
- bool [mmcSequentialWrite](#) ([MMCDriver](#) *mmcp, const uint8_t *buffer)
Writes a block within a sequential write operation.
- bool [mmcStopSequentialWrite](#) ([MMCDriver](#) *mmcp)
Stops a sequential write gracefully.
- bool [mmcSync](#) ([MMCDriver](#) *mmcp)

Waits for card idle condition.

- bool `mmcGetInfo` (`MMCDriver` *mmcp, `BlockDeviceInfo` *bdip)

Returns the media info.

- bool `mmcErase` (`MMCDriver` *mmcp, uint32_t startblk, uint32_t endblk)

Erases blocks.

- bool `mmc_lld_is_card_inserted` (`MMCDriver` *mmcp)
- bool `mmc_lld_is_write_protected` (`MMCDriver` *mmcp)

Variables

- static const struct `mmc_spi_driver_vmt` `mmc_vmt`

Virtual methods table.

- static const uint8_t `mmc_crc7_lookup_table` [256]

Lookup table for CRC-7 (based on polynomial $x^7 + x^3 + 1$).

7.2.10.4.4 Macro Definition Documentation

7.2.10.4.4.1 MMC_CMD0_RETRY

```
#define MMC_CMD0_RETRY 10U
```

7.2.10.4.4.2 MMC_CMD1_RETRY

```
#define MMC_CMD1_RETRY 100U
```

7.2.10.4.4.3 MMC_ACMD41_RETRY

```
#define MMC_ACMD41_RETRY 100U
```

7.2.10.4.4.4 MMC_WAIT_DATA

```
#define MMC_WAIT_DATA 10000U
```

7.2.10.4.4.5 MMC_BUFFER_SIZE

```
#define MMC_BUFFER_SIZE 16U
```

Size of the buffer to be supplied to the driver.

Note

The buffer is meant to be non-cacheable on platforms with data cache.

7.2.10.4.4.6 MMC_IDLE_TIMEOUT_MS

```
#define MMC_IDLE_TIMEOUT_MS 1000
```

Timeout before assuming a failure while waiting for card idle.

Note

Time is in milliseconds.

7.2.10.4.4.7 MMC_USE_MUTUAL_EXCLUSION

```
#define MMC_USE_MUTUAL_EXCLUSION TRUE
```

Mutual exclusion on the SPI bus.

7.2.10.4.4.8 __mmc_driver_methods

```
#define __mmc_driver_methods _mmc_sdb_block_device_methods
```

[MMCDriver](#) specific methods.

7.2.10.4.4.9 mmclsCardInserted

```
#define mmclsCardInserted(  
    mmcp)
```

Value:

[mmc_lld_is_card_inserted](#)(*mmcp*)

Returns the card insertion status.

Note

This macro wraps a low level function named [sdc_lld_is_card_inserted\(\)](#), this function must be provided by the application because it is not part of the SDC driver.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object

Returns

The card state.

Return values

<i>false</i>	card not inserted.
<i>true</i>	card inserted.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.4.4.10 mmclsWriteProtected

```
#define mmcIsWriteProtected(  
    mmcp)
```

Value:

```
mmc_lld_is_write_protected(mmcp)
```

Returns the write protect status.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object

Returns

The card state.

Return values

<i>false</i>	card not inserted.
<i>true</i>	card inserted.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.4.5 Typedef Documentation

7.2.10.4.5.1 MMCConfig

```
typedef mmc\_spi\_config\_t MMCConfig
```

Legacy name for compatibility.

Deprecated

7.2.10.4.5.2 MMCDriver

```
typedef mmc\_spi\_driver\_t MMCDriver
```

Legacy name for compatibility.

Deprecated

7.2.10.4.6 Function Documentation

7.2.10.4.6.1 mmc_is_card_inserted()

```
bool mmc_is_card_inserted (
    void * instance) [static]
```

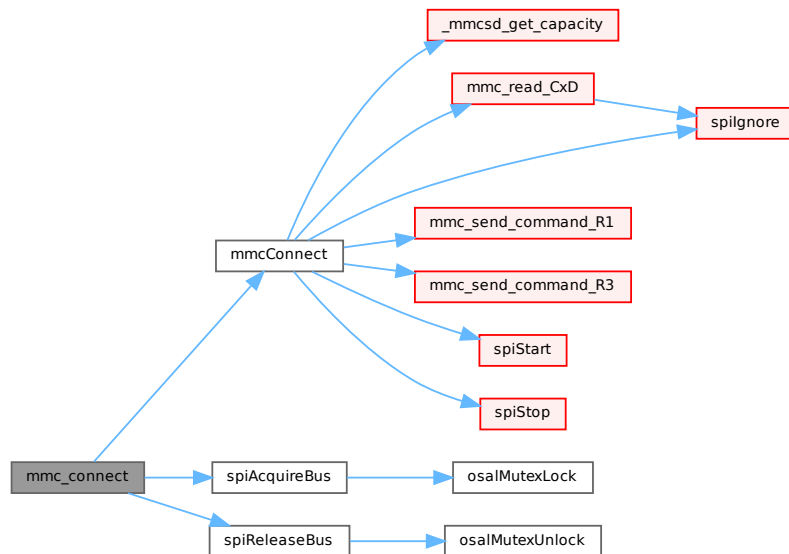
7.2.10.4.6.2 mmc_is_write_protected()

```
bool mmc_is_write_protected (
    void * instance) [static]
```

7.2.10.4.6.3 mmc_connect()

```
bool mmc_connect (
    void * instance) [static]
```

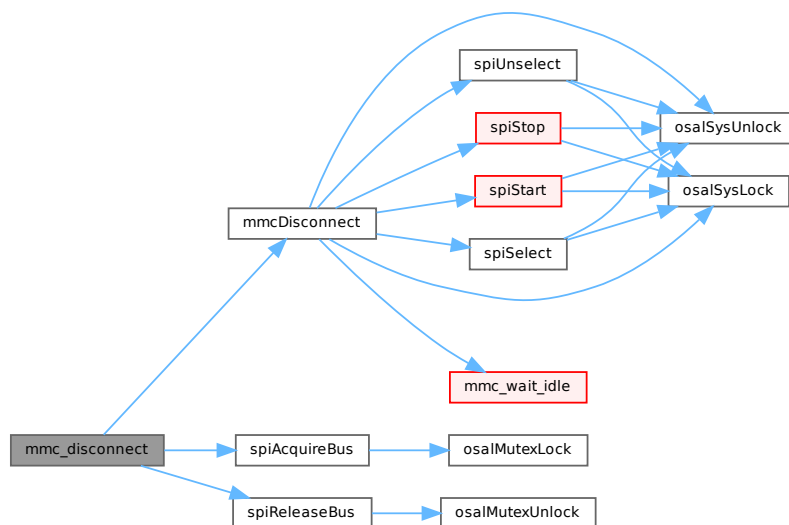
Here is the call graph for this function:



7.2.10.4.6.4 mmc_disconnect()

```
bool mmc_disconnect (
    void * instance) [static]
```

Here is the call graph for this function:

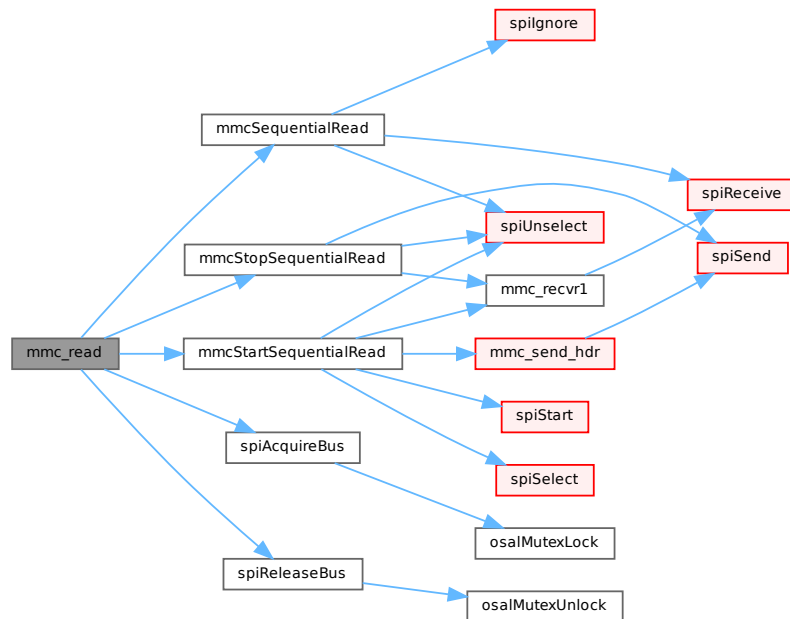


7.2.10.4.6.5 mmc_read()

```

bool mmc_read (
    void * instance,
    uint32_t startblk,
    uint8_t * buffer,
    uint32_t n) [static]
  
```

Here is the call graph for this function:



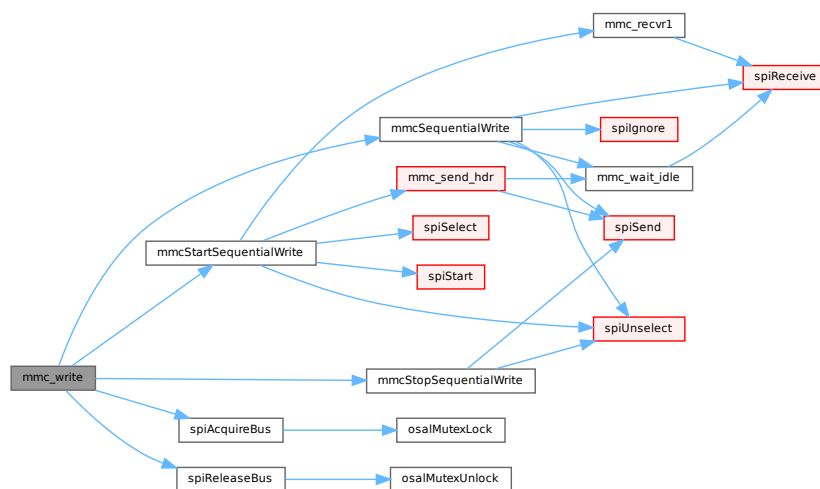
7.2.10.4.6.6 mmc_write()

```

bool mmc_write (
    void * instance,
    uint32_t startblk,
    const uint8_t * buffer,
    uint32_t n) [static]

```

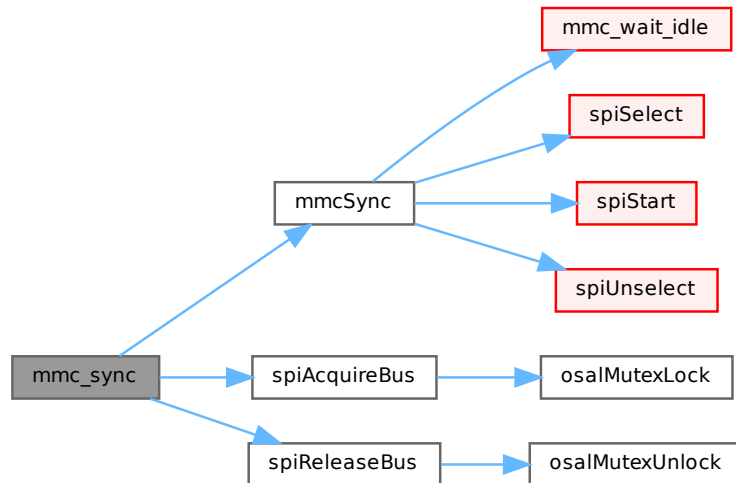
Here is the call graph for this function:



7.2.10.4.6.7 mmc_sync()

```
bool mmc_sync (
    void * instance) [static]
```

Here is the call graph for this function:



7.2.10.4.6.8 mmc_get_info()

```
bool mmc_get_info (
    void * instance,
    BlockDeviceInfo * bdip) [static]
```

Here is the call graph for this function:



7.2.10.4.6.9 mmc_crc7()

```
uint8_t mmc_crc7 (
    uint8_t crc,
    const uint8_t * buffer,
    size_t len) [static]
```

Calculate the MMC standard CRC-7 based on a lookup table.

Parameters

in	<i>crc</i>	start value for CRC
in	<i>buffer</i>	pointer to data buffer
in	<i>len</i>	length of data

Returns

Calculated CRC

7.2.10.4.6.10 mmc_wait_idle()

```
bool mmc_wait_idle (  
    MMCDriver * mmcp) [static]
```

Waits an idle condition.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object

Returns

The operation status.

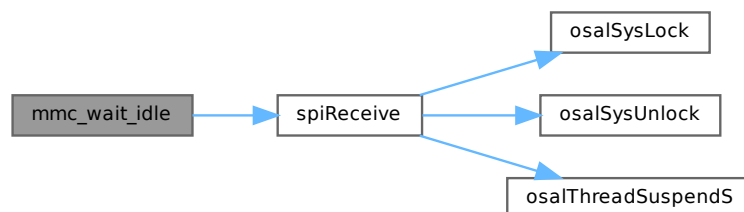
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.4.6.11 mmc_send_hdr()

```
bool mmc_send_hdr (
    MMCDriver * mmcp,
    uint8_t cmd,
    uint32_t arg) [static]
```

Sends a command header.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object
in	<i>cmd</i>	the command id
in	<i>arg</i>	the command argument

Returns

The operation status.

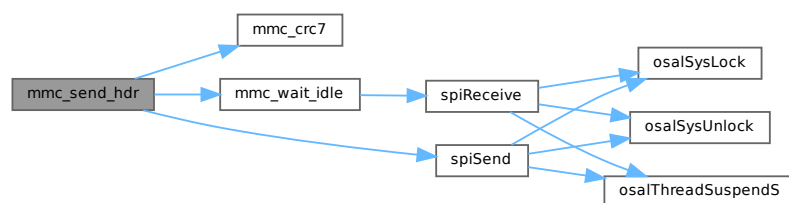
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.4.6.12 mmc_rcvr1()**

```
bool mmc_rcvr1 (
    MMCDriver * mmcp,
    uint8_t * r1p) [static]
```

Receives a single byte response.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object
out	<i>r1p</i>	pointer to buffer for R1 answer

Returns

Operation status.

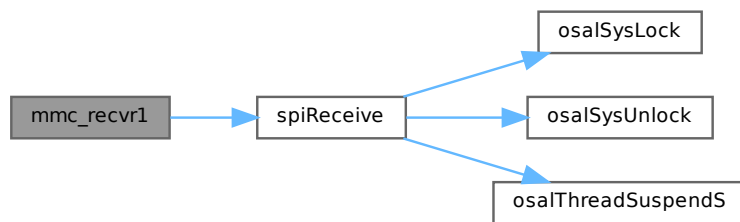
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.4.6.13 mmc_recvr3()**

```
bool mmc_recvr3 (  
    MMCDriver * mmcp,  
    uint8_t * r1p) [static]
```

Receives a three byte response.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object
out	<i>r1p</i>	pointer to buffer for R1 answer

Returns

Operation status.

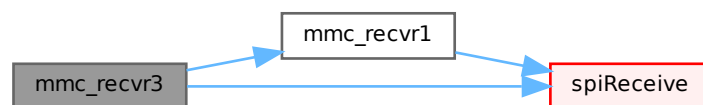
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.4.6.14 mmc_send_command_R1()

```
bool mmc_send_command_R1 (  
    MMCDriver * mmcp,  
    uint8_t cmd,  
    uint32_t arg,  
    uint8_t * r1p) [static]
```

Sends a command and returns a single byte response.

Parameters

in	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object
in	<i>cmd</i>	the command id
in	<i>arg</i>	the command argument
out	<i>r1p</i>	pointer to buffer for R1 answer

Returns

Operation status.

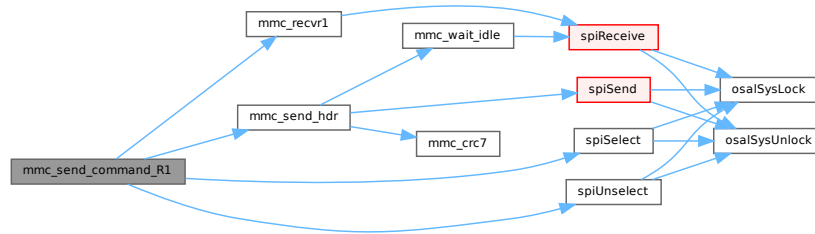
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.4.6.15 mmc_send_command_R3()**

```

bool mmc_send_command_R3 (
    MMCDriver * mmcp,
    uint8_t cmd,
    uint32_t arg,
    uint8_t * r1p) [static]

```

Sends a command which returns a five bytes response (R3).

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object
in	<i>cmd</i>	the command id
in	<i>arg</i>	the command argument
out	<i>r1p</i>	pointer to buffer for R1 answer

Returns

The first byte of the response (R1) as an `uint8_t` value.

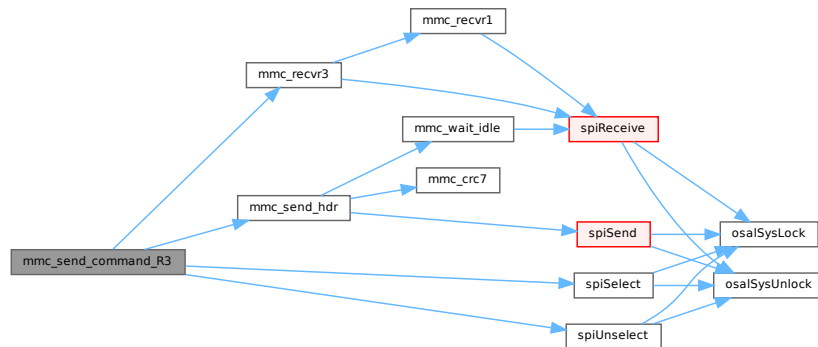
Return values

<code>0xFF</code>	timed out.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.4.6.16 mmc_read_CxD()**

```

bool mmc_read_CxD (
    MMCDriver * mmcp,
    uint8_t cmd,
    uint32_t cxd[4]) [static]
  
```

Reads the CSD.

Parameters

in	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object
out	<i>cmd</i>	command
out	<i>cxd</i>	pointer to the CSD/CID buffer

Returns

The operation status.

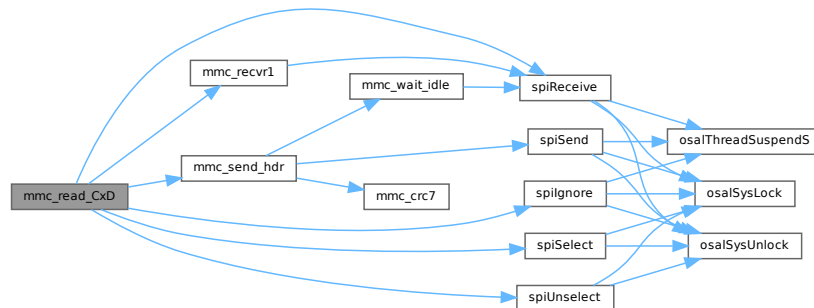
Return values

<code>HAL_SUCCESS</code>	if the operation succeeded.
<code>HAL_FAILED</code>	if the operation failed.

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.4.6.17 mmcInit()**

```
void mmcInit (
    void )
```

MMC over SPI driver initialization.

Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

7.2.10.4.6.18 mmcObjectInit()

```
void mmcObjectInit (
    MMCDriver * mmcp,
    uint8_t * buffer)
```

Initializes an instance.

Parameters

out	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object
in	<i>buffer</i>	operations buffer

Function Class:

Object or module initializer function.

7.2.10.4.6.19 mmcStart()

```
msg_t mmcStart (  
    MMCDriver * mmcp,  
    const MMCConfig * config)
```

Configures and activates the MMC peripheral.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object
in	<i>config</i>	pointer to the MMCConfig object.

Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.4.6.20 mmcStop()

```
void mmcStop (
    MMCDriver * mmcp)
```

Disables the MMC peripheral.

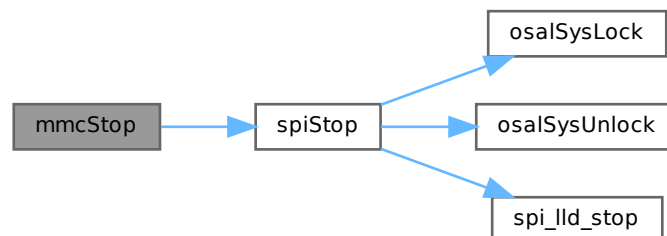
Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.4.6.21 mmcConnect()**

```
bool mmcConnect (
    MMCDriver * mmcp)
```

Performs the initialization procedure on the inserted card.

This function should be invoked when a card is inserted and brings the driver in the `MMC_READY` state where it is possible to perform read and write operations.

Note

It is possible to invoke this function from the insertion event handler.

Parameters

in	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object

Returns

The operation status.

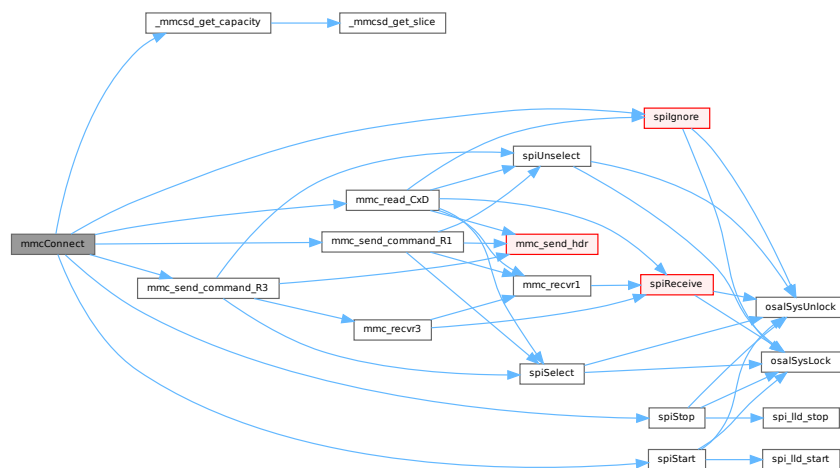
Return values

<code>HAL_SUCCESS</code>	if the operation succeeded.
<code>HAL_FAILED</code>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.4.6.22 mmcDisconnect()**

```
bool mmcDisconnect (
    MMCDriver * mmcp)
```

Brings the driver in a state safe for card removal.

Parameters

in	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object

Returns

The operation status.

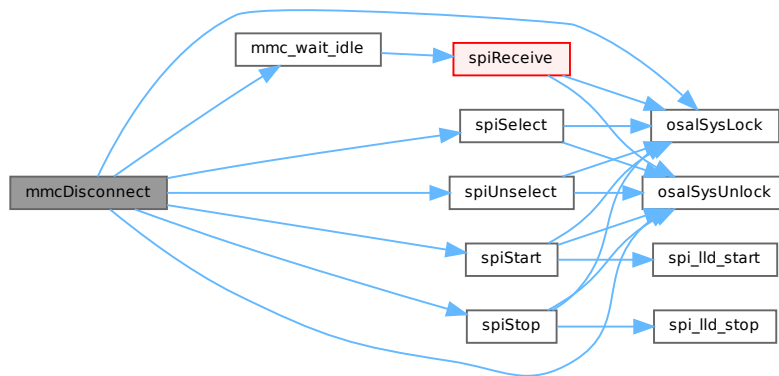
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.4.6.23 mmcStartSequentialRead()

```

bool mmcStartSequentialRead (
    MMCDriver * mmcp,
    uint32_t startblk)

```

Starts a sequential read.

Parameters

in	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object
in	<i>startblk</i>	first block to read

Returns

The operation status.

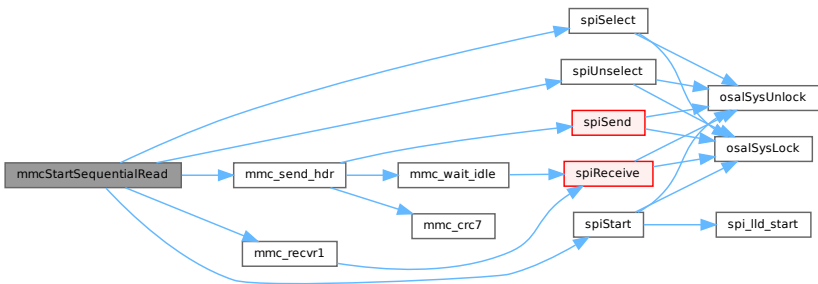
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.4.6.24 mmcSequentialRead()

```
bool mmcSequentialRead (  
    MMCDriver * mmcp,  
    uint8_t * buffer)
```

Reads a block within a sequential read operation.

Parameters

in	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object
out	<i>buffer</i>	pointer to the read buffer

Returns

The operation status.

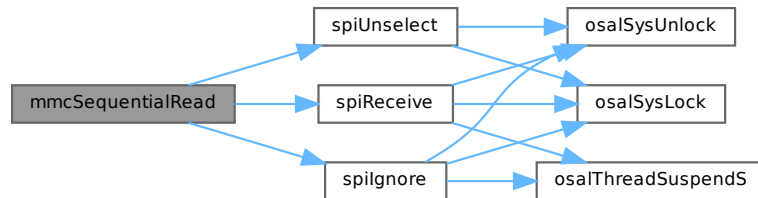
Return values

<code>HAL_SUCCESS</code>	if the operation succeeded.
<code>HAL_FAILED</code>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.4.6.25 mmcStopSequentialRead()**

```
bool mmcStopSequentialRead (
    MMCDriver * mmcp)
```

Stops a sequential read gracefully.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object

Returns

The operation status.

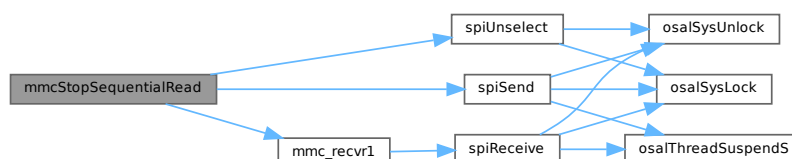
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.4.6.26 mmcStartSequentialWrite()

```
bool mmcStartSequentialWrite (
    MMCDriver * mmcp,
    uint32_t startblk)
```

Starts a sequential write.

Parameters

in	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object
in	<i>startblk</i>	first block to write

Returns

The operation status.

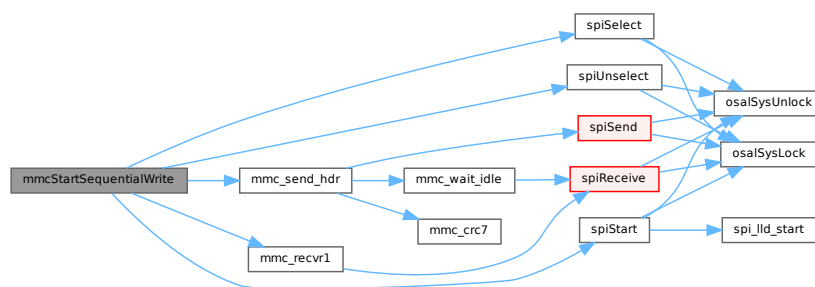
Return values

<code>HAL_SUCCESS</code>	if the operation succeeded.
<code>HAL_FAILED</code>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.4.6.27 mmcSequentialWrite()

```
bool mmcSequentialWrite (
    MMCDriver * mmcp,
    const uint8_t * buffer)
```

Writes a block within a sequential write operation.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object
out	<i>buffer</i>	pointer to the write buffer

Returns

The operation status.

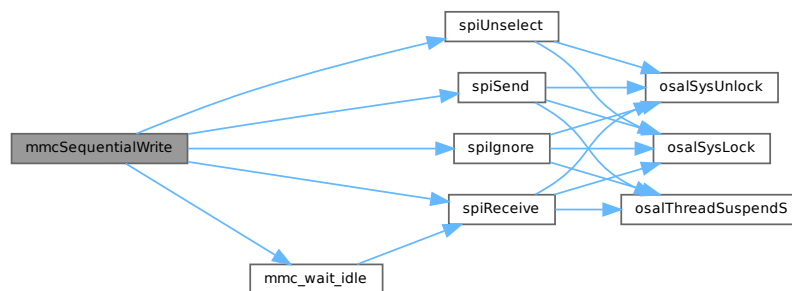
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.4.6.28 mmcStopSequentialWrite()**

```
bool mmcStopSequentialWrite (
    MMCDriver * mmcp)
```

Stops a sequential write gracefully.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object

Returns

The operation status.

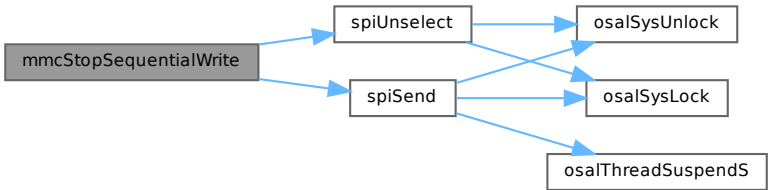
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.4.6.29 mmcSync()

```
bool mmcSync (  
    MMCDriver * mmcp)
```

Waits for card idle condition.

Parameters

in	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object

Returns

The operation status.

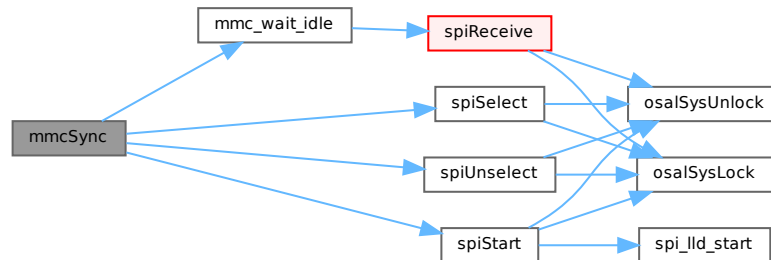
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.4.6.30 mmcGetInfo()**

```
bool mmcGetInfo (
    MMCDriver * mmcp,
    BlockDeviceInfo * bdip)
```

Returns the media info.

Parameters

in	<i>mmcp</i>	pointer to the <code>MMCDriver</code> object
out	<i>bdip</i>	pointer to a <code>BlockDeviceInfo</code> structure

Returns

The operation status.

Return values

<code>HAL_SUCCESS</code>	if the operation succeeded.
<code>HAL_FAILED</code>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.10.4.6.31 mmcErase()

```
bool mmcErase (
    MMCDriver * mmcp,
    uint32_t startblk,
    uint32_t endblk)
```

Erases blocks.

Parameters

in	<i>mmcp</i>	pointer to the MMCDriver object
in	<i>startblk</i>	starting block number
in	<i>endblk</i>	ending block number

Returns

The operation status.

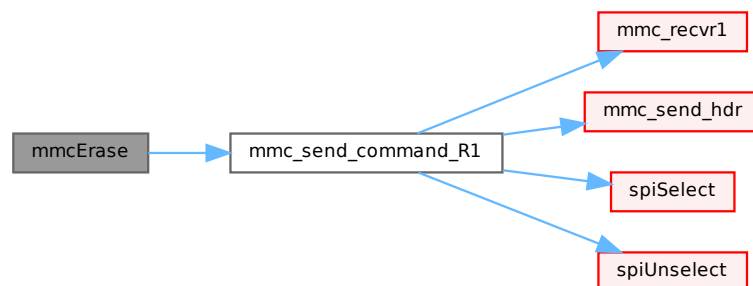
Return values

<i>HAL_SUCCESS</i>	if the operation succeeded.
<i>HAL_FAILED</i>	if the operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.4.6.32 mmc_lld_is_card_inserted()**

```
bool mmc_lld_is_card_inserted (  
    MMCDriver * mmcp)
```

7.2.10.4.6.33 mmc_lld_is_write_protected()

```
bool mmc_lld_is_write_protected (  
    MMCDriver * mmcp)
```

7.2.10.4.7 Variable Documentation

7.2.10.4.7.1 mmc_vmt

```
const struct mmc_spi_driver_vmt mmc_vmt [static]
```

Initial value:

```
= {
    (size_t)0,
    mmc_is_card_inserted,
    mmc_is_write_protected,
    mmc_connect,
    mmc_disconnect,
    mmc_read,
    mmc_write,
    mmc_sync,
    mmc_get_info
}
```

Virtual methods table.

7.2.10.4.7.2 mmc_crc7_lookup_table

```
const uint8_t mmc_crc7_lookup_table[256] [static]
```

Initial value:

```
= {
    0x00, 0x09, 0x12, 0x1b, 0x24, 0x2d, 0x36, 0x3f, 0x48, 0x41, 0x5a, 0x53,
    0x6c, 0x65, 0x7e, 0x77, 0x19, 0x10, 0x0b, 0x02, 0x3d, 0x34, 0x2f, 0x26,
    0x51, 0x58, 0x43, 0x4a, 0x75, 0x7c, 0x67, 0x6e, 0x32, 0x3b, 0x20, 0x29,
    0x16, 0x1f, 0x04, 0x0d, 0x7a, 0x73, 0x68, 0x61, 0x5e, 0x57, 0x4c, 0x45,
    0x2b, 0x22, 0x39, 0x30, 0x0f, 0x06, 0x1d, 0x14, 0x63, 0x6a, 0x71, 0x78,
    0x47, 0x4e, 0x55, 0x5c, 0x64, 0x6d, 0x76, 0x7f, 0x40, 0x49, 0x52, 0x5b,
    0x2c, 0x25, 0x3e, 0x37, 0x08, 0x01, 0x1a, 0x13, 0x7d, 0x74, 0x6f, 0x66,
    0x59, 0x50, 0x4b, 0x42, 0x35, 0x3c, 0x27, 0x2e, 0x11, 0x18, 0x03, 0x0a,
    0x56, 0x5f, 0x44, 0x4d, 0x72, 0x7b, 0x60, 0x69, 0x1e, 0x17, 0x0c, 0x05,
    0x3a, 0x33, 0x28, 0x21, 0x4f, 0x46, 0x5d, 0x54, 0x6b, 0x62, 0x79, 0x70,
    0x07, 0x0e, 0x15, 0x1c, 0x23, 0x2a, 0x31, 0x38, 0x41, 0x48, 0x53, 0x5a,
    0x65, 0x6c, 0x77, 0x7e, 0x09, 0x00, 0x1b, 0x12, 0x2d, 0x24, 0x3f, 0x36,
    0x58, 0x51, 0x4a, 0x43, 0x7c, 0x75, 0x6e, 0x67, 0x10, 0x19, 0x02, 0x0b,
    0x34, 0x3d, 0x26, 0x2f, 0x73, 0x7a, 0x61, 0x68, 0x57, 0x5e, 0x45, 0x4c,
    0x3b, 0x32, 0x29, 0x20, 0x1f, 0x16, 0x0d, 0x04, 0x6a, 0x63, 0x78, 0x71,
    0x4e, 0x47, 0x5c, 0x55, 0x22, 0x2b, 0x30, 0x39, 0x06, 0x0f, 0x14, 0x1d,
    0x25, 0x2c, 0x37, 0x3e, 0x01, 0x08, 0x13, 0x1a, 0x6d, 0x64, 0x7f, 0x76,
    0x49, 0x40, 0x5b, 0x52, 0x3c, 0x35, 0x2e, 0x27, 0x18, 0x11, 0x0a, 0x03,
    0x74, 0x7d, 0x66, 0x6f, 0x50, 0x59, 0x42, 0x4b, 0x17, 0x1e, 0x05, 0x0c,
    0x33, 0x3a, 0x21, 0x28, 0x5f, 0x56, 0x4d, 0x44, 0x7b, 0x72, 0x69, 0x60,
    0x0e, 0x07, 0x1c, 0x15, 0x2a, 0x23, 0x38, 0x31, 0x46, 0x4f, 0x54, 0x5d,
    0x62, 0x6b, 0x70, 0x79
}
```

Lookup table for CRC-7 (based on polynomial $x^7 + x^3 + 1$).

7.2.10.5 Serial NOR Flash Driver

Serial NOR Flash driver.

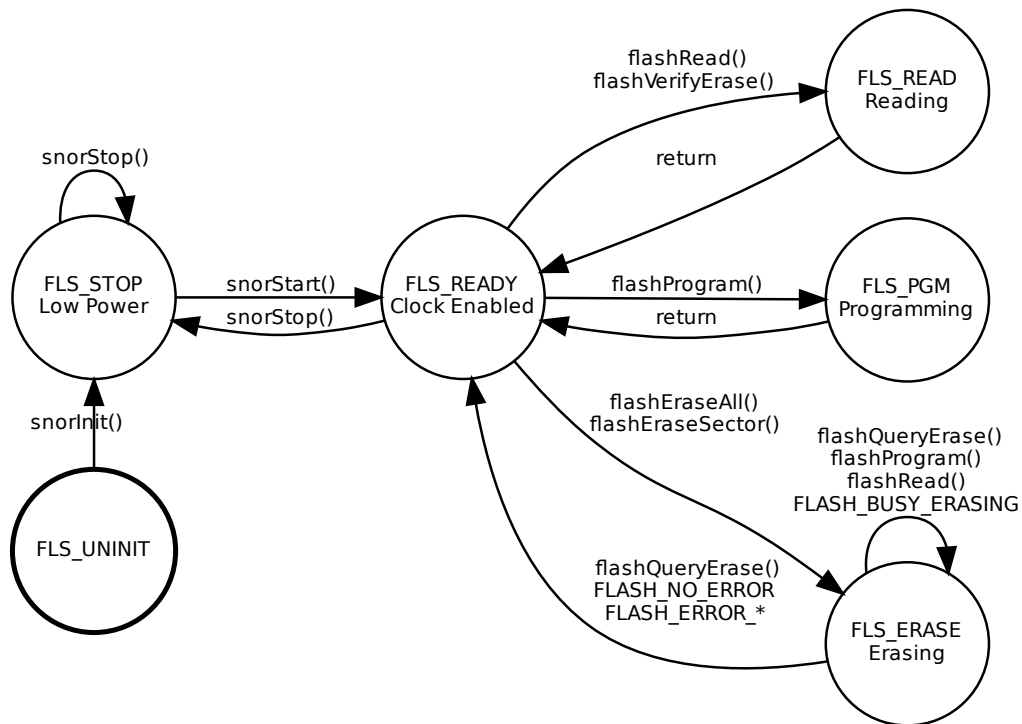
7.2.10.5.1 Detailed Description

Serial NOR Flash driver.

This module implements a generic driver for serial NOR Flash devices.

7.2.10.5.2 Driver State Machine

The flash driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



Bus interface modes.

- `#define SNOR_BUS_DRIVER_SPI 0U`
- `#define SNOR_BUS_DRIVER_WSPI 1U`

Configuration options

- `#define SNOR_BUS_DRIVER SNOR_BUS_DRIVER_WSPI`
Physical transport interface.
- `#define SNOR_SHARED_BUS TRUE`
Shared bus switch.
- `#define SNOR_USE_MUTUAL_EXCLUSION TRUE`
Exclusive access control.
- `#define SNOR_SPI_4BYTES_ADDRESS FALSE`
SPI 4-bytes address switch.

Data Structures

- struct [SNORConfig](#)
Type of a SNOR configuration structure.
- struct [SNORDriverVMT](#)
SNOR virtual methods table.
- struct [snor_nocache_buffer](#)
- struct [SNORDriver](#)
Type of SNOR flash class.

Macros

- #define [SNOR_BUFFER_SIZE](#) 32
Size of the buffer for internal operations.
- #define [BUSConfig](#) [SPICongig](#)
- #define [BUSDriver](#) [SPIDriver](#)
- #define [_snor_flash_methods_alone](#)
SNORDriver specific methods.
- #define [_snor_flash_methods](#)
SNORDriver specific methods with inherited ones.
- #define [bus_acquire](#)(busp, config)
- #define [bus_release](#)(busp)

Typedefs

- typedef struct [snor_nocache_buffer](#) [snor_nocache_buffer_t](#)

Functions

- static const [flash_descriptor_t](#) * [snor_get_descriptor](#) (void *instance)
Returns a pointer to the device descriptor.
- static [flash_error_t](#) [snor_read](#) (void *instance, [flash_offset_t](#) offset, size_t n, uint8_t *rp)
- static [flash_error_t](#) [snor_program](#) (void *instance, [flash_offset_t](#) offset, size_t n, const uint8_t *pp)
- static [flash_error_t](#) [snor_start_erase_all](#) (void *instance)
- static [flash_error_t](#) [snor_start_erase_sector](#) (void *instance, [flash_sector_t](#) sector)
- static [flash_error_t](#) [snor_verify_erase](#) (void *instance, [flash_sector_t](#) sector)
- static [flash_error_t](#) [snor_query_erase](#) (void *instance, uint32_t *msec)
- static [flash_error_t](#) [snor_acquire_exclusive](#) (void *instance)
- static [flash_error_t](#) [snor_release_exclusive](#) (void *instance)
- static [flash_error_t](#) [snor_read_sfdp](#) (void *instance, [flash_offset_t](#) offset, size_t n, uint8_t *rp)
- void [snor_spi_cmd_addr](#) ([BUSDriver](#) *busp, uint32_t cmd, [flash_offset_t](#) offset)
- void [bus_acquire](#) ([BUSDriver](#) *busp, const [BUSConfig](#) *config)
Bus acquisition and lock.
- void [bus_release](#) ([BUSDriver](#) *busp)
Bus release.
- void [bus_stop](#) ([BUSDriver](#) *busp)
Stops the underlying bus driver.
- void [bus_cmd](#) ([BUSDriver](#) *busp, uint32_t cmd)
Sends a naked command.
- void [bus_cmd_send](#) ([BUSDriver](#) *busp, uint32_t cmd, size_t n, const uint8_t *p)

- Sends a command followed by a data transmit phase.*
- void `bus_cmd_receive` (`BUSDriver *busp`, `uint32_t cmd`, `size_t n`, `uint8_t *p`)
- Sends a command followed by a data receive phase.*
- void `bus_cmd_addr` (`BUSDriver *busp`, `uint32_t cmd`, `flash_offset_t offset`)
- Sends a command followed by a flash address.*
- void `bus_cmd_addr_send` (`BUSDriver *busp`, `uint32_t cmd`, `flash_offset_t offset`, `size_t n`, `const uint8_t *p`)
- Sends a command followed by a flash address and a data transmit phase.*
- void `bus_cmd_addr_receive` (`BUSDriver *busp`, `uint32_t cmd`, `flash_offset_t offset`, `size_t n`, `uint8_t *p`)
- Sends a command followed by a flash address and a data receive phase.*
- void `bus_cmd_dummy_receive` (`BUSDriver *busp`, `uint32_t cmd`, `uint32_t dummy`, `size_t n`, `uint8_t *p`)
- Sends a command followed by dummy cycles and a data receive phase.*
- void `bus_cmd_addr_dummy_receive` (`BUSDriver *busp`, `uint32_t cmd`, `flash_offset_t offset`, `uint32_t dummy`, `size_t n`, `uint8_t *p`)
- Sends a command followed by a flash address, dummy cycles and a data receive phase.*
- void `snorObjectInit` (`SNORDriver *devp`, `snor_nocache_buffer_t *nocache`)
- Initializes an instance.*
- void `snorStart` (`SNORDriver *devp`, `const SNORConfig *config`)
- Configures and activates SNOR driver.*
- void `snorStop` (`SNORDriver *devp`)
- Deactivates the SNOR driver.*
- void `snorMemoryMap` (`SNORDriver *devp`, `uint8_t **addrp`)
- Enters the memory Mapping mode.*
- void `snorMemoryUnmap` (`SNORDriver *devp`)
- Leaves the memory Mapping mode.*

Variables

- static const struct `SNORDriverVMT snor_vmt`
- Virtual methods table.*

7.2.10.5.3 Macro Definition Documentation

7.2.10.5.3.1 SNOR_BUS_DRIVER_SPI

```
#define SNOR_BUS_DRIVER_SPI 0U
```

7.2.10.5.3.2 SNOR_BUS_DRIVER_WSPI

```
#define SNOR_BUS_DRIVER_WSPI 1U
```

7.2.10.5.3.3 SNOR_BUFFER_SIZE

```
#define SNOR_BUFFER_SIZE 32
```

Size of the buffer for internal operations.

7.2.10.5.3.4 SNOR_BUS_DRIVER

```
#define SNOR_BUS_DRIVER SNOR_BUS_DRIVER_WSPI
```

Physical transport interface.

7.2.10.5.3.5 SNOR_SHARED_BUS

```
#define SNOR_SHARED_BUS TRUE
```

Shared bus switch.

If set to `TRUE` the device acquires bus ownership on each transaction.

Note

Requires `SPI_USE_MUTUAL_EXCLUSION` or `WSPI_USE_MUTUAL_EXCLUSION` depending on mode selected with `SNOR_BUS_MODE`.

7.2.10.5.3.6 SNOR_USE_MUTUAL_EXCLUSION

```
#define SNOR_USE_MUTUAL_EXCLUSION TRUE
```

Exclusive access control.

Note

Disabling this option saves both code and data space.

7.2.10.5.3.7 SNOR_SPI_4BYTES_ADDRESS

```
#define SNOR_SPI_4BYTES_ADDRESS FALSE
```

SPI 4-bytes address switch.

If set to `TRUE` the device will use 4-bytes address in SPI bus, only relevant if SPI is used

7.2.10.5.3.8 BUSConfig

```
#define BUSConfig SPICongig
```

7.2.10.5.3.9 BUSDriver

```
#define BUSDriver SPIDriver
```

7.2.10.5.3.10 _snor_flash_methods_alone

```
#define _snor_flash_methods_alone
```

Value:

```
/* Read SFDP.*/
flash_error_t (*read_sfdp)(void *instance,
                           flash_offset_t offset,
                           size_t n,
                           uint8_t *rp);
```

SNORDriver specific methods.

7.2.10.5.3.11 _snor_flash_methods

```
#define _snor_flash_methods
```

Value:

```
_base_flash_methods
_snor_flash_methods_alone
```

SNORDriver specific methods with inherited ones.

7.2.10.5.3.12 bus_acquire

```
#define bus_acquire(
    busp,
    config)
```

7.2.10.5.3.13 bus_release

```
#define bus_release(
    busp)
```

7.2.10.5.4 Typedef Documentation**7.2.10.5.4.1 snor_nocache_buffer_t**

```
typedef struct snor_nocache_buffer snor_nocache_buffer_t
```

7.2.10.5.5 Function Documentation**7.2.10.5.5.1 snor_get_descriptor()**

```
const flash_descriptor_t * snor_get_descriptor (
    void * instance) [static]
```

Returns a pointer to the device descriptor.

Parameters

in	<i>instance</i>	instance pointer

Returns

Pointer to a static descriptor structure.

7.2.10.5.5.2 snor_read()

```
flash_error_t snor_read (  
    void * instance,  
    flash_offset_t offset,  
    size_t n,  
    uint8_t * rp) [static]
```

7.2.10.5.5.3 snor_program()

```
flash_error_t snor_program (  
    void * instance,  
    flash_offset_t offset,  
    size_t n,  
    const uint8_t * pp) [static]
```

7.2.10.5.5.4 snor_start_erase_all()

```
flash_error_t snor_start_erase_all (  
    void * instance) [static]
```

7.2.10.5.5.5 snor_start_erase_sector()

```
flash_error_t snor_start_erase_sector (  
    void * instance,  
    flash_sector_t sector) [static]
```

7.2.10.5.5.6 snor_verify_erase()

```
flash_error_t snor_verify_erase (  
    void * instance,  
    flash_sector_t sector) [static]
```

7.2.10.5.5.7 snor_query_erase()

```
flash_error_t snor_query_erase (  
    void * instance,  
    uint32_t * msec) [static]
```

7.2.10.5.5.8 snor_acquire_exclusive()

```
flash_error_t snor_acquire_exclusive (  
    void * instance) [static]
```

Here is the call graph for this function:



7.2.10.5.5.9 snor_release_exclusive()

```
flash_error_t snor_release_exclusive (  
    void * instance) [static]
```

Here is the call graph for this function:



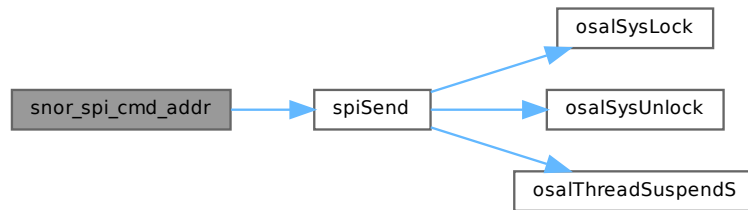
7.2.10.5.5.10 snor_read_sfdp()

```
flash_error_t snor_read_sfdp (  
    void * instance,  
    flash_offset_t offset,  
    size_t n,  
    uint8_t * rp) [static]
```

7.2.10.5.5.11 snor_spi_cmd_addr()

```
void snor_spi_cmd_addr (  
    BUSDriver * busp,  
    uint32_t cmd,  
    flash_offset_t offset)
```

Here is the call graph for this function:



7.2.10.5.5.12 bus_acquire()

```
void bus_acquire (
    BUSDriver * busp,
    const BUSConfig * config)
```

Bus acquisition and lock.

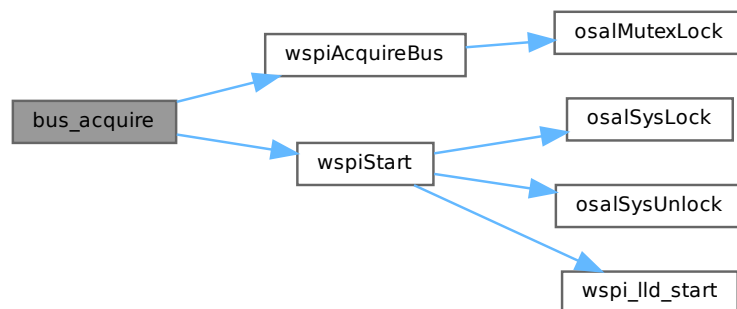
Parameters

in	<i>busp</i>	pointer to the bus driver
in	<i>config</i>	bus configuration

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.5.5.13 bus_release()

```
void bus_release (  
    BUSDriver * busp)
```

Bus release.

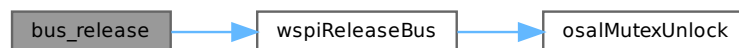
Parameters

in	<i>busp</i>	pointer to the bus driver

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.5.5.14 bus_stop()

```
void bus_stop (  
    BUSDriver * busp)
```

Stops the underlying bus driver.

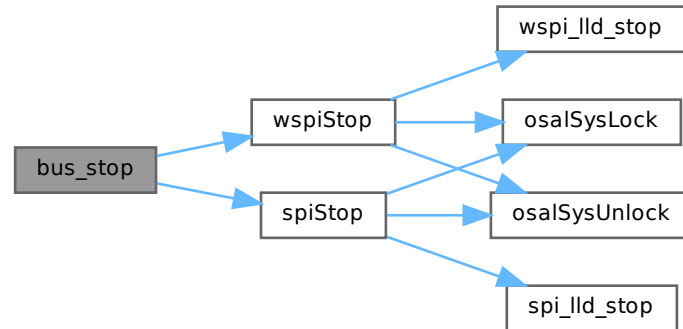
Parameters

in	<i>busp</i>	pointer to the bus driver

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.5.5.15 bus_cmd()**

```
void bus_cmd (
    BUSDriver * busp,
    uint32_t cmd)
```

Sends a naked command.

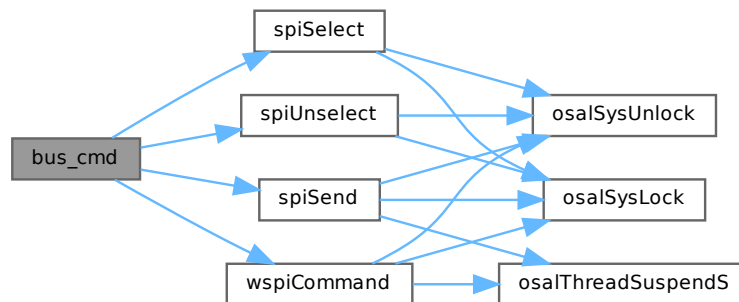
Parameters

in	<i>busp</i>	pointer to the bus driver
in	<i>cmd</i>	instruction code

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.5.5.16 bus_cmd_send()**

```

void bus_cmd_send (
    BUSDriver * busp,
    uint32_t cmd,
    size_t n,
    const uint8_t * p)

```

Sends a command followed by a data transmit phase.

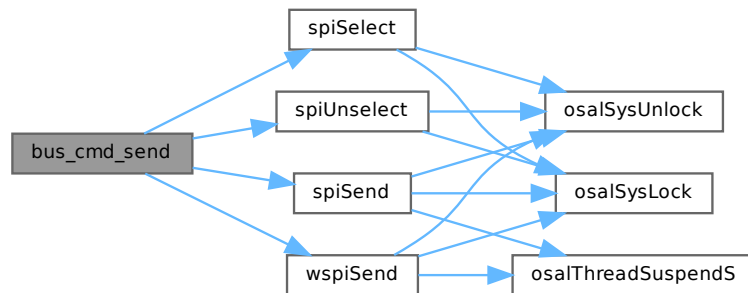
Parameters

in	<i>busp</i>	pointer to the bus driver
in	<i>cmd</i>	instruction code
in	<i>n</i>	number of bytes to transmit
in	<i>p</i>	data buffer

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.5.5.17 bus_cmd_receive()**

```

void bus_cmd_receive (
    BUSDriver * busp,
    uint32_t cmd,
    size_t n,
    uint8_t * p)

```

Sends a command followed by a data receive phase.

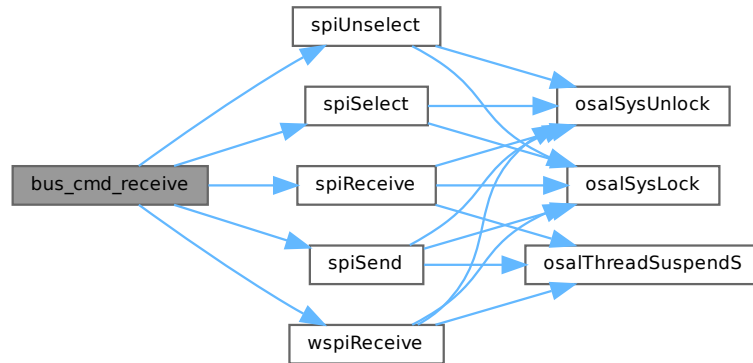
Parameters

in	<i>busp</i>	pointer to the bus driver
in	<i>cmd</i>	instruction code
in	<i>n</i>	number of bytes to receive
out	<i>p</i>	data buffer

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.5.5.18 bus_cmd_addr()**

```

void bus_cmd_addr (
    BUSDriver * busp,
    uint32_t cmd,
    flash_offset_t offset)
  
```

Sends a command followed by a flash address.

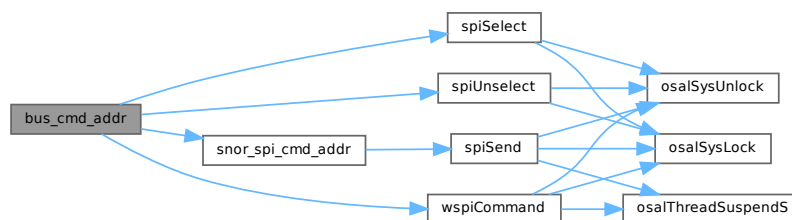
Parameters

in	<i>busp</i>	pointer to the bus driver
in	<i>cmd</i>	instruction code
in	<i>offset</i>	flash offset

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.5.5.19 bus_cmd_addr_send()

```
void bus_cmd_addr_send (
    BUSDriver * busp,
    uint32_t cmd,
    flash_offset_t offset,
    size_t n,
    const uint8_t * p)
```

Sends a command followed by a flash address and a data transmit phase.

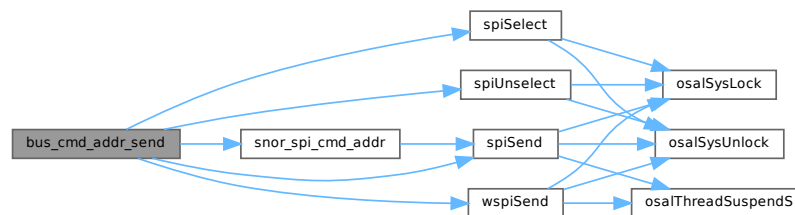
Parameters

in	<i>busp</i>	pointer to the bus driver
in	<i>cmd</i>	instruction code
in	<i>offset</i>	flash offset
in	<i>n</i>	number of bytes to receive
in	<i>p</i>	data buffer

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.5.5.20 bus_cmd_addr_receive()

```
void bus_cmd_addr_receive (
    BUSDriver * busp,
    uint32_t cmd,
    flash_offset_t offset,
    size_t n,
    uint8_t * p)
```

Sends a command followed by a flash address and a data receive phase.

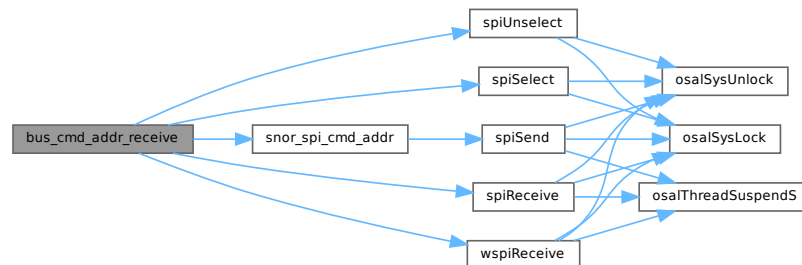
Parameters

in	<i>busp</i>	pointer to the bus driver
in	<i>cmd</i>	instruction code
in	<i>offset</i>	flash offset
in	<i>n</i>	number of bytes to receive
out	<i>p</i>	data buffer

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.5.5.21 bus_cmd_dummy_receive()**

```

void bus_cmd_dummy_receive (
    BUSDriver * busp,
    uint32_t cmd,
    uint32_t dummy,
    size_t n,
    uint8_t * p)
  
```

Sends a command followed by dummy cycles and a data receive phase.

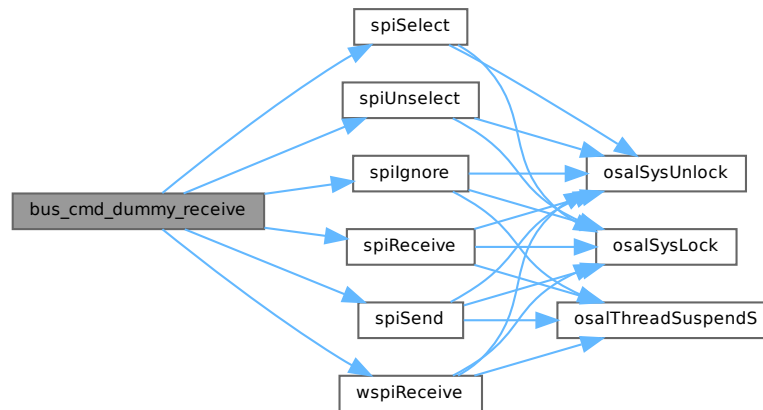
Parameters

in	<i>busp</i>	pointer to the bus driver
in	<i>cmd</i>	instruction code
in	<i>dummy</i>	number of dummy cycles
in	<i>n</i>	number of bytes to receive
out	<i>p</i>	data buffer

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.10.5.5.22 bus_cmd_addr_dummy_receive()

```

void bus_cmd_addr_dummy_receive (
    BUSDriver * busp,
    uint32_t cmd,
    flash_offset_t offset,
    uint32_t dummy,
    size_t n,
    uint8_t * p)
  
```

Sends a command followed by a flash address, dummy cycles and a data receive phase.

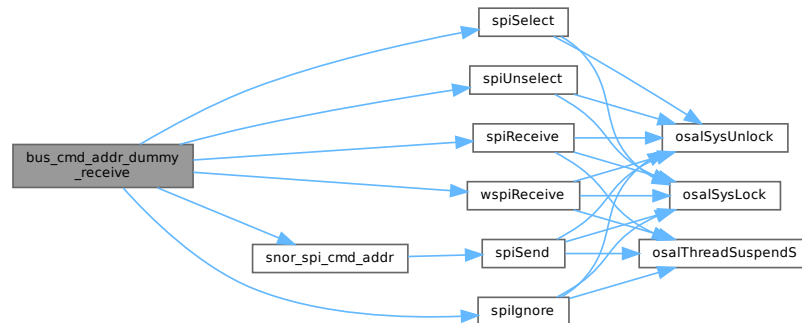
Parameters

in	<i>busp</i>	pointer to the bus driver
in	<i>cmd</i>	instruction code
in	<i>offset</i>	flash offset
in	<i>dummy</i>	number of dummy cycles
in	<i>n</i>	number of bytes to receive
out	<i>p</i>	data buffer

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.10.5.5.23 snorObjectInit()**

```
void snorObjectInit (
    SNORDriver * devp,
    snor_nocache_buffer_t * nocache)
```

Initializes an instance.

Parameters

out	<i>devp</i>	pointer to the <code>SNORDriver</code> object
in	<i>nocache</i>	pointer to the non-cacheable buffers

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.10.5.5.24 snorStart()**

```
void snorStart (
    SNORDriver * devp,
    const SNORConfig * config)
```

Configures and activates SNOR driver.

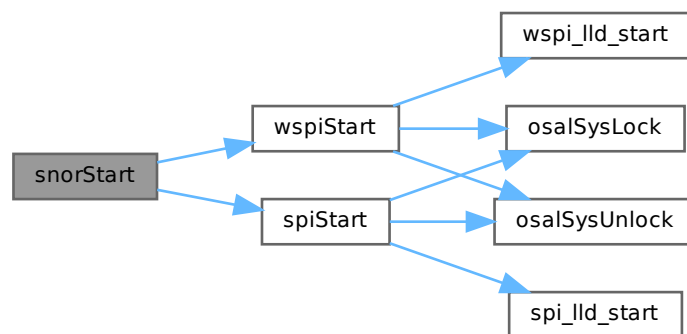
Parameters

in	<i>devp</i>	pointer to the SNORDriver object
in	<i>config</i>	pointer to the configuration

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.5.5.25 snorStop()**

```
void snorStop (  
    SNORDriver * devp)
```

Deactivates the SNOR driver.

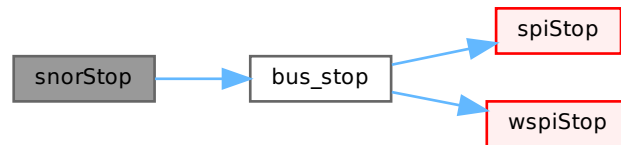
Parameters

in	<i>devp</i>	pointer to the SNORDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.5.5.26 snorMemoryMap()**

```
void snorMemoryMap (
    SNORDriver * devp,
    uint8_t ** addrp)
```

Enters the memory Mapping mode.

The memory mapping mode is only available when the WSPI mode is selected and the underlying WSPI controller supports the feature.

Parameters

in	<i>devp</i>	pointer to the SNORDriver object
out	<i>addrp</i>	pointer to the memory start address of the mapped flash or <code>NULL</code>

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.5.5.27 snorMemoryUnmap()

```
void snorMemoryUnmap (  
    SNORDriver * devp)
```

Leaves the memory Mapping mode.

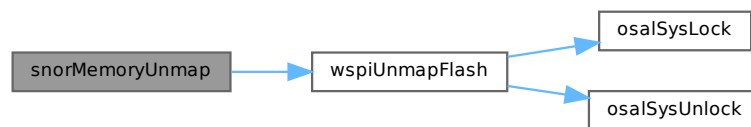
Parameters

in	devp	pointer to the SNORDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.5.6 Variable Documentation****7.2.10.5.6.1 snor_vmt**

```
const struct SNORDriverVMT snor_vmt [static]
```

Initial value:

```
= {
    (size_t)0,
    snor\_get\_descriptor, snor\_read, snor\_program,
    snor\_start\_erase\_all, snor\_start\_erase\_sector,
    snor\_query\_erase, snor\_verify\_erase, snor\_acquire\_exclusive,
    snor\_release\_exclusive, snor\_read\_sfdp,
}
```

Virtual methods table.

7.2.10.6 Serial over USB Driver

Serial over USB Driver.

7.2.10.6.1 Detailed Description

Serial over USB Driver.

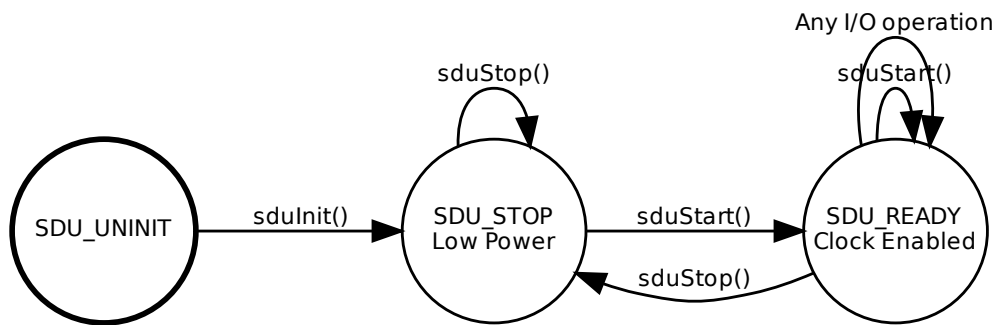
This module implements an USB Communication Device Class (CDC) as a normal serial communication port accessible from the device application.

Precondition

In order to use the USB over Serial driver the `HAL_USE_SERIAL_USB` option must be enabled in [halconf.h](#).

7.2.10.6.2 Driver State Machine

The driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



SERIAL_USB configuration options

- `#define SERIAL_USB_BUFFERS_SIZE 256`
Serial over USB buffers size.
- `#define SERIAL_USB_BUFFERS_NUMBER 2`
Serial over USB number of buffers.

Data Structures

- struct `SerialUSBConfig`
Serial over USB Driver configuration structure.
- struct `SerialUSBDriverVMT`
SerialDriver virtual methods table.
- struct `SerialUSBDriver`
Full duplex serial driver class.

Macros

- `#define _serial_usb_driver_data`
SerialDriver specific data.
- `#define _serial_usb_driver_methods _base_asynchronous_channel_methods`
SerialUSBDriver specific methods.

Typedefs

- typedef struct SerialUSBDriver [SerialUSBDriver](#)
Structure representing a serial over USB driver.

Enumerations

- enum [sdustate_t](#) { [SDU_UNINIT](#) = 0 , [SDU_STOP](#) = 1 , [SDU_READY](#) = 2 }
- Driver state machine possible states.*

Functions

- static bool [sdu_start_receive](#) ([SerialUSBDriver](#) *sdup)
- static size_t [_write](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [_read](#) (void *ip, uint8_t *bp, size_t n)
- static [msg_t](#) [_put](#) (void *ip, uint8_t b)
- static [msg_t](#) [_get](#) (void *ip)
- static [msg_t](#) [_putt](#) (void *ip, uint8_t b, [sysinterval_t](#) timeout)
- static [msg_t](#) [_gett](#) (void *ip, [sysinterval_t](#) timeout)
- static size_t [_writet](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static size_t [_readt](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static [msg_t](#) [_ctl](#) (void *ip, unsigned int operation, void *arg)
- static void [ibnotify](#) ([io_buffers_queue_t](#) *bqp)
Notification of empty buffer released into the input buffers queue.
- static void [obnotify](#) ([io_buffers_queue_t](#) *bqp)
Notification of filled buffer inserted into the output buffers queue.
- void [sduInit](#) (void)
Serial Driver initialization.
- void [sduObjectInit](#) ([SerialUSBDriver](#) *sdup)
Initializes a generic full duplex driver object.
- [msg_t](#) [sduStart](#) ([SerialUSBDriver](#) *sdup, const [SerialUSBConfig](#) *config)
Configures and starts the driver.
- void [sduStop](#) ([SerialUSBDriver](#) *sdup)
Stops the driver.
- void [sduSuspendHookI](#) ([SerialUSBDriver](#) *sdup)
USB device suspend handler.
- void [sduWakeupHookI](#) ([SerialUSBDriver](#) *sdup)
USB device wakeup handler.
- void [sduConfigureHookI](#) ([SerialUSBDriver](#) *sdup)
USB device configured handler.
- bool [sduRequestsHook](#) ([USBDriver](#) *usbp)
Default requests hook.
- void [sduSOFHookI](#) ([SerialUSBDriver](#) *sdup)
SOF handler.
- void [sduDataTransmitted](#) ([USBDriver](#) *usbp, [usbep_t](#) ep)
Default data transmitted callback.
- void [sduDataReceived](#) ([USBDriver](#) *usbp, [usbep_t](#) ep)
Default data received callback.
- void [sduInterruptTransmitted](#) ([USBDriver](#) *usbp, [usbep_t](#) ep)
Default data received callback.
- [msg_t](#) [sduControl](#) ([USBDriver](#) *usbp, unsigned int operation, void *arg)
Control operation on a serial USB port.

Variables

- static `cdc_linecoding_t` `linecoding`
- static const struct `SerialUSBDriverVMT` `vmt`

7.2.10.6.3 Macro Definition Documentation

7.2.10.6.3.1 SERIAL_USB_BUFFERS_SIZE

```
#define SERIAL_USB_BUFFERS_SIZE 256
```

Serial over USB buffers size.

Configuration parameter, the buffer size must be a multiple of the USB data endpoint maximum packet size.

Note

The default is 256 bytes for both the transmission and receive buffers.

7.2.10.6.3.2 SERIAL_USB_BUFFERS_NUMBER

```
#define SERIAL_USB_BUFFERS_NUMBER 2
```

Serial over USB number of buffers.

Note

The default is 2 buffers.

7.2.10.6.3.3 _serial_usb_driver_data

```
#define _serial_usb_driver_data
```

Value:

```
_base_asynchronous_channel_data
/* Driver state.*/
sdustate_t          state;
/* Input buffers queue.*/
input_buffers_queue_t  ibqueue;
/* Output queue.*/
output_buffers_queue_t obqueue;
/* Input buffer.*/
uint8_t             ib[BQ_BUFFER_SIZE(SERIAL_USB_BUFFERS_NUMBER,
                                      SERIAL_USB_BUFFERS_SIZE)];
/* Output buffer.*/
uint8_t             ob[BQ_BUFFER_SIZE(SERIAL_USB_BUFFERS_NUMBER,
                                      SERIAL_USB_BUFFERS_SIZE)];
/* End of the mandatory fields.*/
/* Current configuration data.*/
const SerialUSBConfig *config;
```

`SerialDriver` specific data.

7.2.10.6.3.4 _serial_usb_driver_methods

```
#define _serial_usb_driver_methods _base_asynchronous_channel_methods
```

`SerialUSBDriver` specific methods.

7.2.10.6.4 Typedef Documentation

7.2.10.6.4.1 SerialUSBDriver

```
typedef struct SerialUSBDriver SerialUSBDriver
```

Structure representing a serial over USB driver.

7.2.10.6.5 Enumeration Type Documentation

7.2.10.6.5.1 sdustate_t

```
enum sdustate_t
```

Driver state machine possible states.

Enumerator

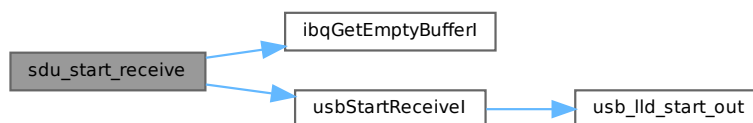
SDU_UNINIT	Not initialized.
SDU_STOP	Stopped.
SDU_READY	Ready.

7.2.10.6.6 Function Documentation

7.2.10.6.6.1 sdu_start_receive()

```
bool sdu_start_receive (
    SerialUSBDriver * sdup) [static]
```

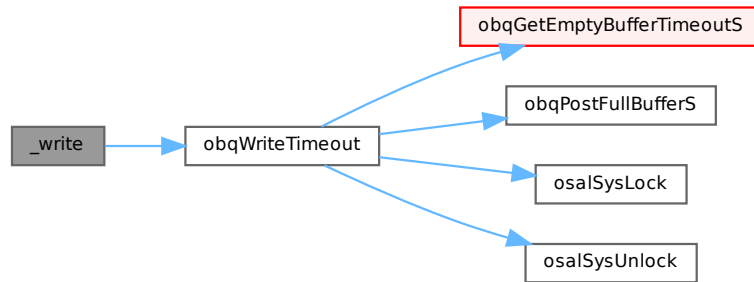
Here is the call graph for this function:



7.2.10.6.6.2 _write()

```
size_t _write (  
    void * ip,  
    const uint8_t * bp,  
    size_t n) [static]
```

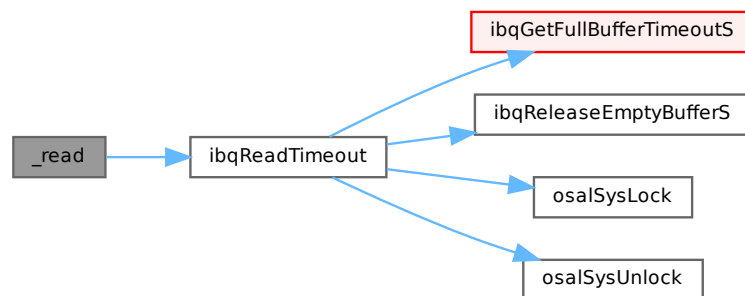
Here is the call graph for this function:



7.2.10.6.6.3 _read()

```
size_t _read (  
    void * ip,  
    uint8_t * bp,  
    size_t n) [static]
```

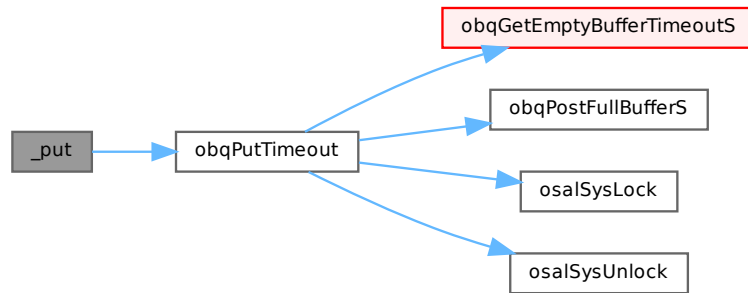
Here is the call graph for this function:



7.2.10.6.6.4 _put()

```
msg_t _put (  
    void * ip,  
    uint8_t b) [static]
```

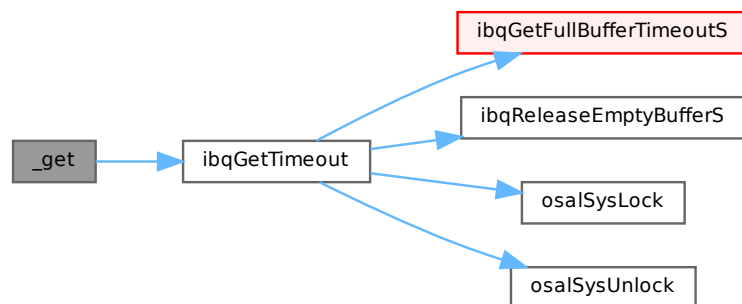
Here is the call graph for this function:



7.2.10.6.6.5 _get()

```
msg_t _get (  
    void * ip) [static]
```

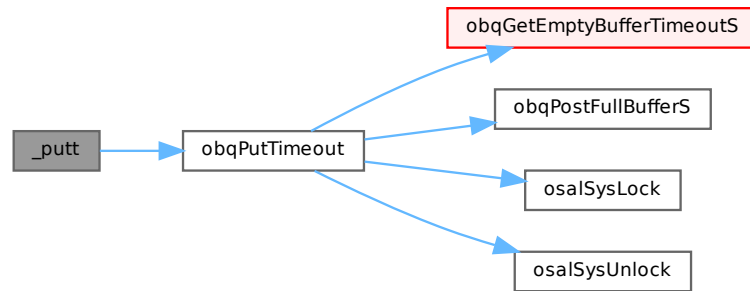
Here is the call graph for this function:



7.2.10.6.6.6 _putt()

```
msg_t _putt (
    void * ip,
    uint8_t b,
    sysinterval_t timeout) [static]
```

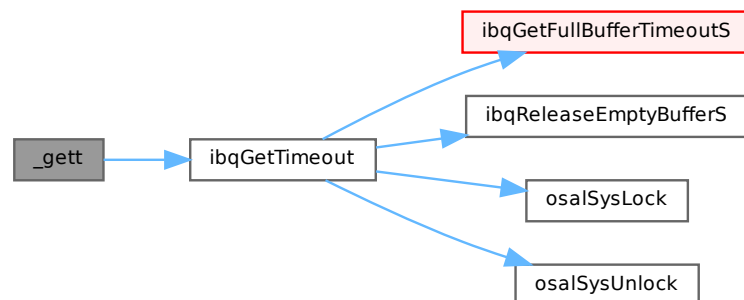
Here is the call graph for this function:



7.2.10.6.6.7 _gett()

```
msg_t _gett (
    void * ip,
    sysinterval_t timeout) [static]
```

Here is the call graph for this function:



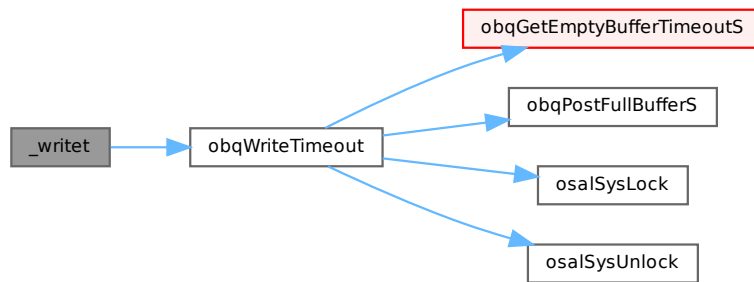
7.2.10.6.6.8 _writet()

```

size_t _writet (
    void * ip,
    const uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]

```

Here is the call graph for this function:



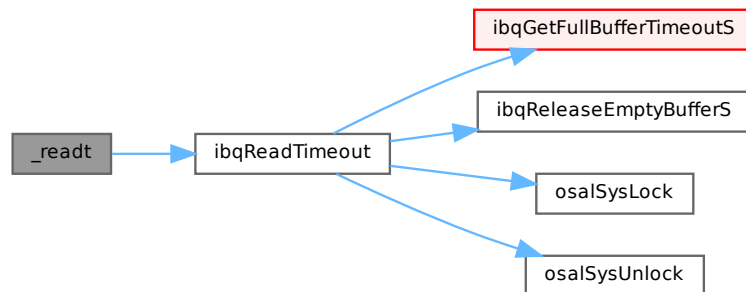
7.2.10.6.6.9 _readt()

```

size_t _readt (
    void * ip,
    uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [static]

```

Here is the call graph for this function:



7.2.10.6.6.10 _ctl()

```
msg_t _ctl (
    void * ip,
    unsigned int operation,
    void * arg) [static]
```

7.2.10.6.6.11 ibnotify()

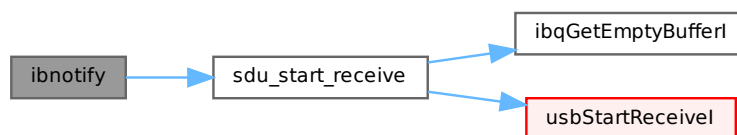
```
void ibnotify (
    io_buffers_queue_t * bqp) [static]
```

Notification of empty buffer released into the input buffers queue.

Parameters

in	<i>bqp</i>	the buffers queue pointer.

Here is the call graph for this function:



7.2.10.6.6.12 obnotify()

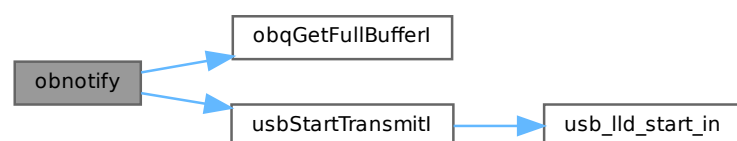
```
void obnotify (
    io_buffers_queue_t * bqp) [static]
```

Notification of filled buffer inserted into the output buffers queue.

Parameters

in	<i>bqp</i>	the buffers queue pointer.

Here is the call graph for this function:



7.2.10.6.6.13 sduInit()

```
void sduInit (
    void )
```

Serial Driver initialization.

Note

This function is implicitly invoked by `halInit()`, there is no need to explicitly initialize the driver.

Function Class:

Object or module initializer function.

7.2.10.6.6.14 sduObjectInit()

```
void sduObjectInit (
    SerialUSBDriver * sdup)
```

Initializes a generic full duplex driver object.

The HW dependent part of the initialization has to be performed outside, usually in the hardware initialization code.

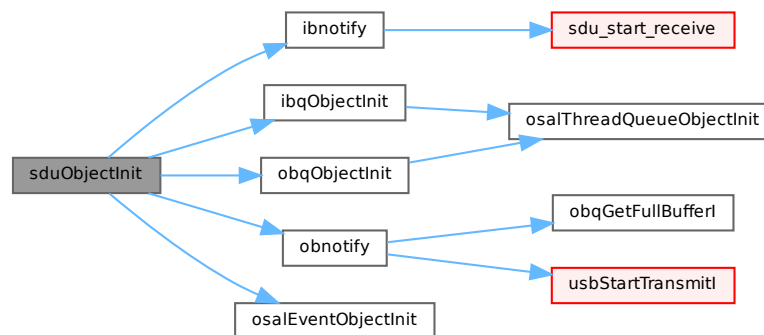
Parameters

out	<i>sdup</i>	pointer to a <code>SerialUSBDriver</code> structure

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.10.6.6.15 sduStart()**

```
msg_t sduStart (
    SerialUSBDriver * sdup,
    const SerialUSBConfig * config)
```

Configures and starts the driver.

Parameters

in	<i>sdup</i>	pointer to a SerialUSBDriver object
in	<i>config</i>	the serial over USB driver configuration

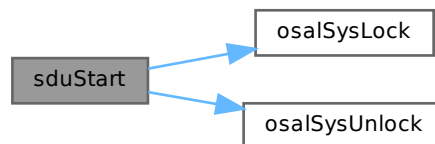
Returns

The operation status.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.6.6.16 sduStop()**

```
void sduStop (  
    SerialUSBDriver * sdup)
```

Stops the driver.

Any thread waiting on the driver's queues will be awakened with the message `MSG_RESET`.

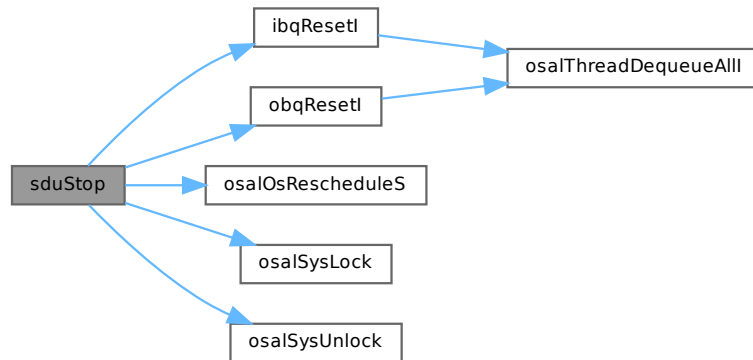
Parameters

in	<i>sdup</i>	pointer to a SerialUSBDriver object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.10.6.6.17 sduSuspendHookI()**

```
void sduSuspendHookI (
    SerialUSBDriver * sdup)
```

USB device suspend handler.

Generates a `CHN_DISCONNECT` event and puts queues in non-blocking mode, this way the application cannot get stuck in the middle of an I/O operations.

Note

If this function is not called from an ISR then an explicit call to `osalOsRescheduleS()` is necessary afterward.

Parameters

in	<code>sdup</code>	pointer to a <code>SerialUSBDriver</code> object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.10.6.6.18 sduWakeupHookI()

```
void sduWakeupHookI (
    SerialUSBDriver * sdup)
```

USB device wakeup handler.

Generates a CHN_CONNECT event and resumes normal queues operations.

Note

If this function is not called from an ISR then an explicit call to `osalOsRescheduleS()` is necessary afterward.

Parameters

in	<i>sdup</i>	pointer to a <code>SerialUSBDriver</code> object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.10.6.6.19 sduConfigureHookI()

```
void sduConfigureHookI (
    SerialUSBDriver * sdup)
```

USB device configured handler.

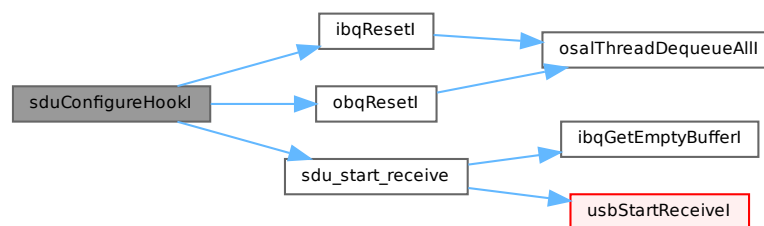
Parameters

in	<i>sdup</i>	pointer to a <code>SerialUSBDriver</code> object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.10.6.6.20 sduRequestsHook()

```
bool sduRequestsHook (  
    USBDriver * usbp)
```

Default requests hook.

Applications wanting to use the Serial over USB driver can use this function as requests hook in the USB configuration. The following requests are emulated:

- CDC_GET_LINE_CODING.
- CDC_SET_LINE_CODING.
- CDC_SET_CONTROL_LINE_STATE.

Parameters

in	<i>usbp</i>	pointer to the USBDriver object

Returns

The hook status.

Return values

<i>true</i>	Message handled internally.
<i>false</i>	Message not handled.

7.2.10.6.6.21 sduSOFHookI()

```
void sduSOFHookI (  
    SerialUSBDriver * sdup)
```

SOF handler.

The SOF interrupt is used for automatic flushing of incomplete buffers pending in the output queue.

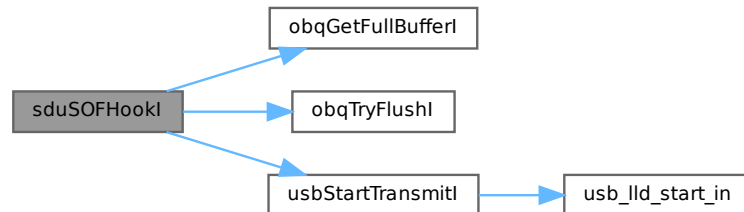
Parameters

in	<i>sdup</i>	pointer to a SerialUSBDriver object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.10.6.6.22 sduDataTransmitted()**

```
void sduDataTransmitted (
    USBDriver * usbp,
    usbep_t ep)
```

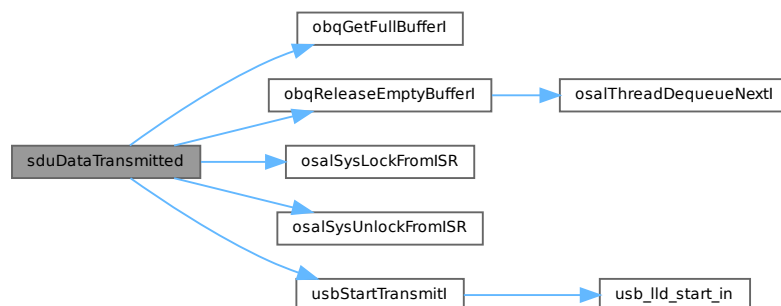
Default data transmitted callback.

The application must use this function as callback for the IN data endpoint.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	IN endpoint number

Here is the call graph for this function:



7.2.10.6.6.23 sduDataReceived()

```
void sduDataReceived (
    USBDriver * usbp,
    usbep_t ep)
```

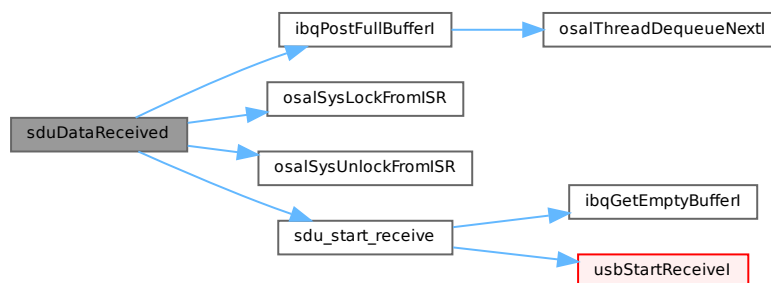
Default data received callback.

The application must use this function as callback for the OUT data endpoint.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	OUT endpoint number

Here is the call graph for this function:



7.2.10.6.6.24 sduInterruptTransmitted()

```
void sduInterruptTransmitted (
    USBDriver * usbp,
    usbep_t ep)
```

Default data received callback.

The application must use this function as callback for the IN interrupt endpoint.

Parameters

in	<i>usbp</i>	pointer to the <code>USBDriver</code> object
in	<i>ep</i>	endpoint number

7.2.10.6.6.25 sduControl()

```
msg_t sduControl (
    USBDriver * usbp,
    unsigned int operation,
    void * arg)
```

Control operation on a serial USB port.

Parameters

in	<i>usbp</i>	pointer to a USBDriver object
in	<i>operation</i>	control operation code
in, out	<i>arg</i>	operation argument

Returns

The control operation status.

Return values

<i>MSG_OK</i>	in case of success.
<i>MSG_TIMEOUT</i>	in case of operation timeout.
<i>MSG_RESET</i>	in case of operation reset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.10.6.7 Variable Documentation

7.2.10.6.7.1 linecoding

```
cdc_linecoding_t linecoding [static]
```

Initial value:

```
= {
    {0x00, 0x96, 0x00, 0x00},
    LC_STOP_1, LC_PARITY_NONE, 8
}
```

7.2.10.6.7.2 vmt

```
const struct SerialUSBDriverVMT vmt [static]
```

Initial value:

```
= {
    (size_t)0,
    _write, _read, _put, _get,
    _putt, _gett, _writet, _readt,
    _ctl
}
```

7.2.10.7 Improved Serial NOR Driver

Serial NOR Flash memories framework. This module provides support for a variety of SNOR devices. Features:

7.2.10.7.1 Detailed Description

Serial NOR Flash memories framework. This module provides support for a variety of SNOR devices. Features:

- Object-oriented, each driver is a class, all drivers inherit from a common [hal_xsnor_base_c](#) base class.
- Supports SPI and/os WSPI HAL drivers as busses.
- Device auto-detection.
- Support for mixed device.

Topics

- [Base Serial NOR Driver class](#)
Base Serial NOR Driver class.
- [Base Flash Interface](#)
Base Flash Interface.
- [Macronix MX25 SNOR Driver](#)
Macronix MX25 Driver.
- [Micron N25Q Driver](#)
Micron N25Q Driver.

7.2.10.7.2 Base Serial NOR Driver class

Base Serial NOR Driver class.

7.2.10.7.2.1 Detailed Description

Base Serial NOR Driver class.

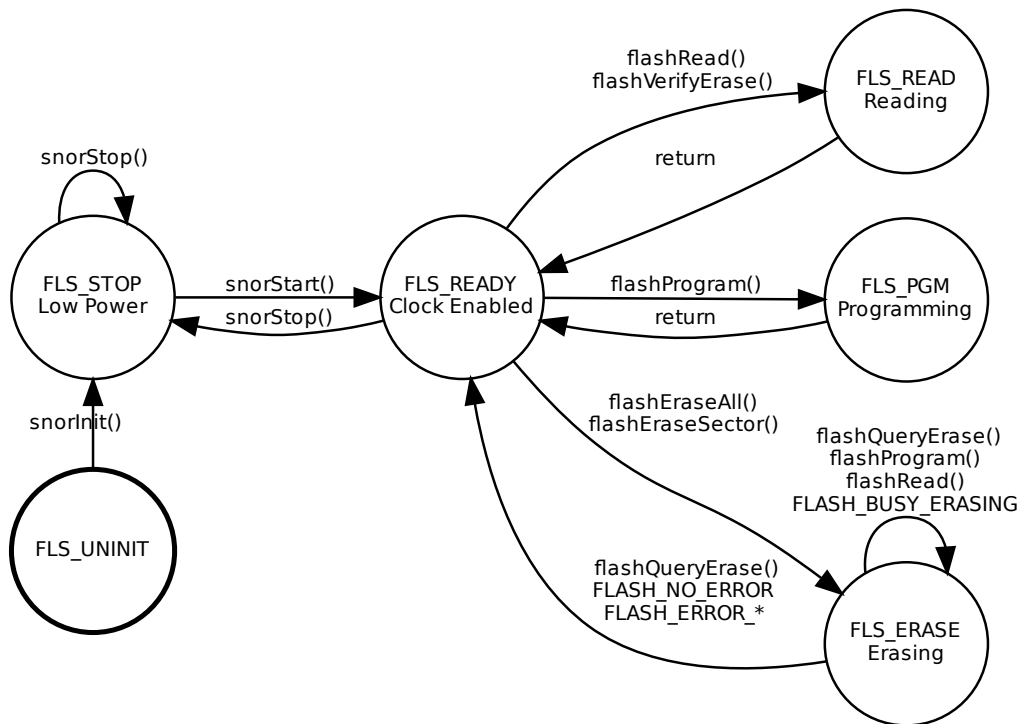
SNOR abstract driver.

This module implements a base class for Serial NOR Flash devices. This base class implements:

- Access API by implementing a Flash Interface.
- Driver state machine handling.
- Communication via SPI and/or WSPI drivers.

7.2.10.7.2.2 Driver State Machine

The flash driver implements a state machine internally, not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



Base class for SNOR flash devices.

Class @p hal_xsnor_base_c structures

- typedef struct [hal_xsnor_base](#) [hal_xsnor_base_c](#)
Type of a XSNOR base driver class.

Methods implementations of hal_xsnor_base_c

- void * [__xsnor_objinit_impl](#) (void *ip, const void *vmt)
Implementation of object creation.
- void [__xsnor_dispose_impl](#) (void *ip)
Implementation of object finalization.

Regular methods of `hal_xsnor_base_c`

- void `__xsnor_spi_cmd_addr` (void *ip, uint32_t cmd, `flash_offset_t` offset)
Sends command and address over SPI.
- void `__xsnor_bus_acquire` (void *ip)
Bus acquisition and lock.
- void `__xsnor_bus_release` (void *ip)
Bus release and unlock.
- void `__xsnor_bus_cmd` (void *ip, uint32_t cmd)
Sends a naked command.
- void `__xsnor_bus_cmd_send` (void *ip, uint32_t cmd, size_t n, const uint8_t *p)
Sends a command followed by a data transmit phase.
- void `__xsnor_bus_cmd_receive` (void *ip, uint32_t cmd, size_t n, uint8_t *p)
Sends a command followed by a data receive phase.
- void `__xsnor_bus_cmd_addr` (void *ip, uint32_t cmd, `flash_offset_t` offset)
Sends a command followed by a flash address.
- void `__xsnor_bus_cmd_addr_send` (void *ip, uint32_t cmd, `flash_offset_t` offset, size_t n, const uint8_t *p)
Sends a command followed by a flash address and a data transmit phase.
- void `__xsnor_bus_cmd_addr_receive` (void *ip, uint32_t cmd, `flash_offset_t` offset, size_t n, uint8_t *p)
Sends a command followed by a flash address and a data receive phase.
- void `__xsnor_bus_cmd_dummy_receive` (void *ip, uint32_t cmd, uint32_t dummy, size_t n, uint8_t *p)
Sends a command followed by dummy cycles and a data receive phase.
- void `__xsnor_bus_cmd_addr_dummy_receive` (void *ip, uint32_t cmd, `flash_offset_t` offset, uint32_t dummy, size_t n, uint8_t *p)
Sends a complete header followed by a data receive phase.
- `flash_error_t` `xsnorStart` (void *ip, const `xsnor_config_t` *config)
Configures and activates a SNOR driver.
- void `xsnorStop` (void *ip)
Deactivates a SNOR driver.
- `flash_error_t` `xsnorMemoryMap` (void *ip, uint8_t **addrp)
Enters the memory mapped mode.
- void `xsnorMemoryUnmap` (void *ip)
Leaves the memory mapped mode.

Virtual methods of `hal_xsnor_base_c`

- static `CC_FORCE_INLINE flash_error_t` `xsnor_device_init` (void *ip)
- static `CC_FORCE_INLINE flash_error_t` `xsnor_device_read` (void *ip, `flash_offset_t` offset, size_t n, uint8_t *rp)
Read operation.
- static `CC_FORCE_INLINE flash_error_t` `xsnor_device_program` (void *ip, `flash_offset_t` offset, size_t n, const uint8_t *pp)
Program operation.
- static `CC_FORCE_INLINE flash_error_t` `xsnor_device_start_erase_all` (void *ip)
Starts a whole-device erase operation.
- static `CC_FORCE_INLINE flash_error_t` `xsnor_device_start_erase_sector` (void *ip, `flash_sector_t` sector)
Starts an sector erase operation.
- static `CC_FORCE_INLINE flash_error_t` `xsnor_device_query_erase` (void *ip, unsigned *msec)
Queries the driver for erase operation progress.
- static `CC_FORCE_INLINE flash_error_t` `xsnor_device_verify_erase` (void *ip, `flash_sector_t` sector)
Returns the erase state of a sector.
- static `CC_FORCE_INLINE flash_error_t` `xsnor_device_mmap_on` (void *ip, uint8_t **addrp)
- static `CC_FORCE_INLINE void` `xsnor_device_mmap_off` (void *ip)

Interfaces implementation of hal_xsnor_base_c

- static const `flash_descriptor_t * __xsnor_fls_get_descriptor_impl` (void *ip)
Implementation of interface method `flsGetDescriptor()`.
- static `flash_error_t __xsnor_fls_read_impl` (void *ip, `flash_offset_t` offset, `size_t` n, `uint8_t *rp`)
Implementation of interface method `flsRead()`.
- static `flash_error_t __xsnor_fls_program_impl` (void *ip, `flash_offset_t` offset, `size_t` n, const `uint8_t *pp`)
Implementation of interface method `flsProgram()`.
- static `flash_error_t __xsnor_fls_start_erase_all_impl` (void *ip)
Implementation of interface method `flsStartEraseAll()`.
- static `flash_error_t __xsnor_fls_start_erase_sector_impl` (void *ip, `flash_sector_t` sector)
Implementation of interface method `flsStartEraseSector()`.
- static `flash_error_t __xsnor_fls_query_erase_impl` (void *ip, unsigned *msec)
Implementation of interface method `flsQueryErase()`.
- static `flash_error_t __xsnor_fls_verify_erase_impl` (void *ip, `flash_sector_t` sector)
Implementation of interface method `flsVerifyErase()`.
- static `flash_error_t __xsnor_fls_acquire_exclusive_impl` (void *ip)
Implementation of interface method `flsAcquireExclusive()`.
- static `flash_error_t __xsnor_fls_release_exclusive_impl` (void *ip)
Implementation of interface method `flsReleaseExclusive()`.

Bus type and width options

- #define `XSNOR_BUS_MODE_SPI` 0U
- #define `XSNOR_BUS_MODE_WSPI_1LINE` 1U
- #define `XSNOR_BUS_MODE_WSPI_2LINES` 2U
- #define `XSNOR_BUS_MODE_WSPI_4LINES` 3U
- #define `XSNOR_BUS_MODE_WSPI_8LINES` 4U

Configuration options

- #define `XSNOR_BUFFER_SIZE` 32
Non-cacheable operations buffer.
- #define `XSNOR_USE_SPI` TRUE
SPI support enable switch.
- #define `XSNOR_USE_WSPI` TRUE
WSPI support enable switch.
- #define `XSNOR_SHARED_BUS` TRUE
Bus share support enable switch.

Bus mutex macros when sharing is disabled

- #define `__xsnor_bus_acquire`(self)
- #define `__xsnor_bus_release`(self)

Data Structures

- struct `xsnor_buffers`
SNOR driver configuration.
- struct `xsnor_commands`
SNOR command configuration structure.
- struct `xsnor_bus_wspi`
WSPI-specific configuration fields.
- struct `xsnor_bus_spi`
SPI-specific configuration fields.
- union `xsnor_bus_configs`
Union of possible bus configurations.
- struct `xsnor_config`
SNOR driver configuration.
- struct `hal_xsnor_base_vmt`
Class `hal_xsnor_base_c` virtual methods table.
- struct `hal_xsnor_base`
Structure representing a XSNOR base driver class.
- class `hal_xsnor_base_c`
Base class of all SNOR drivers.

Macros

- #define `FLASH_ATTR_SPI_4BYTES_ADDR_HINT` 0x00008000U
Hint to use 4 bytes addresses in SPI protocol.
- #define `XSNOR_USE_BOTH` ((XSNOR_USE_SPI == TRUE) && (XSNOR_USE_WSPI == TRUE))
This switch is TRUE if both SPI and WSPI are in use.

Typedefs

- typedef struct `xsnor_buffers` `xsnor_buffers_t`
Type of a non-cacheable buffer.
- typedef struct `xsnor_config` `xsnor_config_t`
Type of a SNOR configuration structure.
- typedef struct `xsnor_commands` `xsnor_commands_t`
Type of a commands configuration structure.

7.2.10.7.2.3 Macro Definition Documentation

XSNOR_BUS_MODE_SPI

```
#define XSNOR_BUS_MODE_SPI 0U
```

XSNOR_BUS_MODE_WSPI_1LINE

```
#define XSNOR_BUS_MODE_WSPI_1LINE 1U
```

XSNOR_BUS_MODE_WSPI_2LINES

```
#define XSNOR_BUS_MODE_WSPI_2LINES 2U
```

XSNOR_BUS_MODE_WSPI_4LINES

```
#define XSNOR_BUS_MODE_WSPI_4LINES 3U
```

XSNOR_BUS_MODE_WSPI_8LINES

```
#define XSNOR_BUS_MODE_WSPI_8LINES 4U
```

FLASH_ATTR_SPI_4BYTES_ADDR_HINT

```
#define FLASH_ATTR_SPI_4BYTES_ADDR_HINT 0x00008000U
```

Hint to use 4 bytes addresses in SPI protocol.

Note

TODO: To be moved into the flash interface module.

XSNOR_BUFFER_SIZE

```
#define XSNOR_BUFFER_SIZE 32
```

Non-cacheable operations buffer.

XSNOR_USE_SPI

```
#define XSNOR_USE_SPI TRUE
```

SPI support enable switch.

XSNOR_USE_WSPI

```
#define XSNOR_USE_WSPI TRUE
```

WSPI support enable switch.

XSNOR_SHARED_BUS

```
#define XSNOR_SHARED_BUS TRUE
```

Bus share support enable switch.

XSNOR_USE_BOTH

```
#define XSNOR_USE_BOTH ((XSNOR_USE_SPI == TRUE) && (XSNOR_USE_WSPI == TRUE))
```

This switch is TRUE if both SPI and WSPI are in use.

__xsnor_bus_acquire

```
#define __xsnor_bus_acquire(  
    self)
```

__xsnor_bus_release

```
#define __xsnor_bus_release(  
    self)
```

7.2.10.7.2.4 Typedef Documentation**xsnor_buffers_t**

```
typedef struct xsnor_buffers xsnor_buffers_t
```

Type of a non-cacheable buffer.

xsnor_config_t

```
typedef struct xsnor_config xsnor_config_t
```

Type of a SNOR configuration structure.

xsnor_commands_t

```
typedef struct xsnor_commands xsnor_commands_t
```

Type of a commands configuration structure.

hal_xsnor_base_c

```
typedef struct hal_xsnor_base hal_xsnor_base_c
```

Type of a XSNOR base driver class.

7.2.10.7.2.5 Function Documentation**__xsnor_objinit_impl()**

```
void * __xsnor_objinit_impl (  
    void * ip,  
    const void * vmt)
```

Implementation of object creation.

Note

This function is meant to be used by derived classes.

Parameters

out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance to be initialized.
in	<i>vmt</i>	VMT pointer for the new object.

Returns

A new reference to the object.

Here is the call graph for this function:

**__xsnor_dispose_impl()**

```
void __xsnor_dispose_impl (
    void * ip)
```

Implementation of object finalization.

Note

This function is meant to be used by derived classes.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance to be disposed.

__xsnor_spi_cmd_addr()

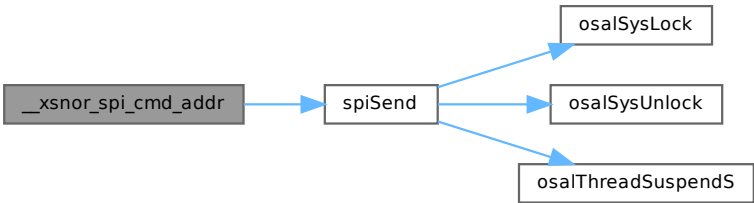
```
void __xsnor_spi_cmd_addr (  
    void * ip,  
    uint32_t cmd,  
    flash_offset_t offset)
```

Sends command and address over SPI.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>cmd</i>	Instruction code.
in	<i>offset</i>	Flash offset.

Here is the call graph for this function:



__xsnor_bus_acquire()

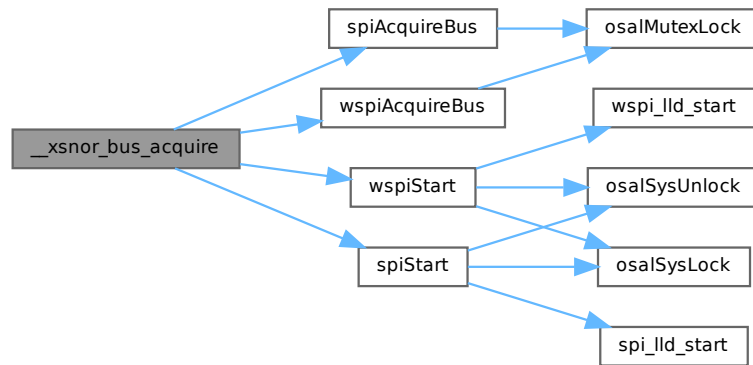
```
void __xsnor_bus_acquire (  
    void * ip)
```

Bus acquisition and lock.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.

Here is the call graph for this function:



`__xsnor_bus_release()`

```
void __xsnor_bus_release (
    void * ip)
```

Bus release and unlock.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.

Here is the call graph for this function:



`__xsnor_bus_cmd()`

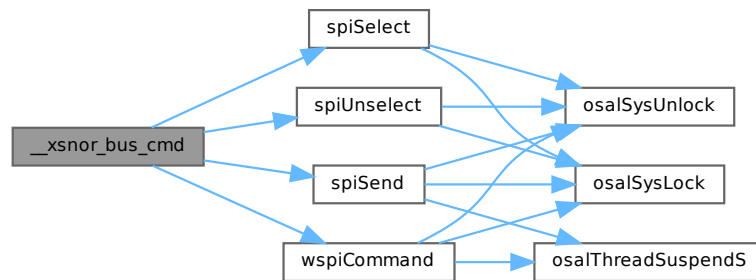
```
void __xsnor_bus_cmd (
    void * ip,
    uint32_t cmd)
```

Sends a naked command.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>cmd</i>	Instruction code.

Here is the call graph for this function:

**`__xsnor_bus_cmd_send()`**

```

void __xsnor_bus_cmd_send (
    void * ip,
    uint32_t cmd,
    size_t n,
    const uint8_t * p)

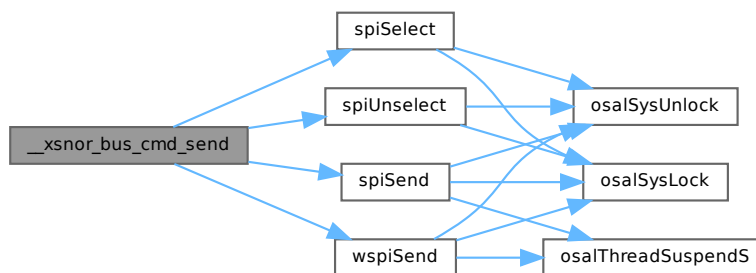
```

Sends a command followed by a data transmit phase.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>cmd</i>	Instruction code.
in	<i>n</i>	Number of bytes to transmit.
in	<i>p</i>	Data buffer.

Here is the call graph for this function:



__xsnor_bus_cmd_receive()

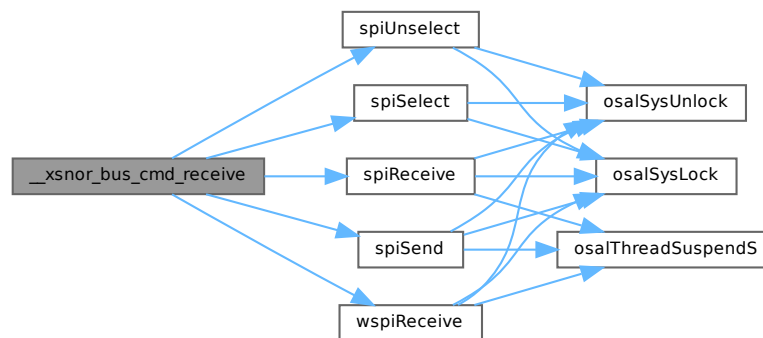
```
void __xsnor_bus_cmd_receive (
    void * ip,
    uint32_t cmd,
    size_t n,
    uint8_t * p)
```

Sends a command followed by a data receive phase.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>cmd</i>	Instruction code.
in	<i>n</i>	Number of bytes to receive.
out	<i>p</i>	Data buffer.

Here is the call graph for this function:

**__xsnor_bus_cmd_addr()**

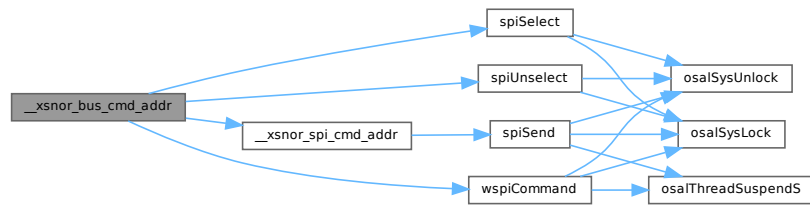
```
void __xsnor_bus_cmd_addr (
    void * ip,
    uint32_t cmd,
    flash_offset_t offset)
```

Sends a command followed by a flash address.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>cmd</i>	Instruction code.
in	<i>offset</i>	Flash offset.

Here is the call graph for this function:



__xsnor_bus_cmd_addr_send()

```

void __xsnor_bus_cmd_addr_send (
    void * ip,
    uint32_t cmd,
    flash_offset_t offset,
    size_t n,
    const uint8_t * p)

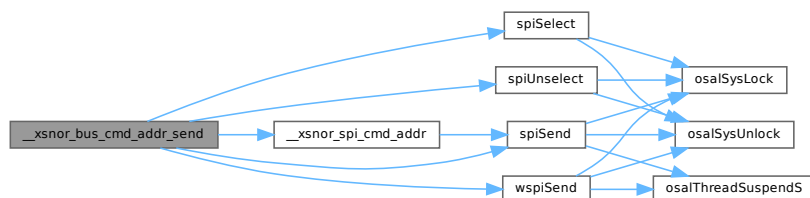
```

Sends a command followed by a flash address and a data transmit phase.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>cmd</i>	Instruction code.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to transmit.
in	<i>p</i>	Data buffer.

Here is the call graph for this function:



__xsnor_bus_cmd_addr_receive()

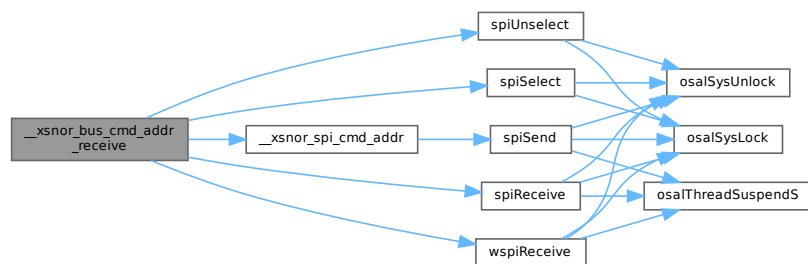
```
void __xsnor_bus_cmd_addr_receive (
    void * ip,
    uint32_t cmd,
    flash_offset_t offset,
    size_t n,
    uint8_t * p)
```

Sends a command followed by a flash address and a data receive phase.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>cmd</i>	Instruction code.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to receive.
in	<i>p</i>	Data buffer.

Here is the call graph for this function:

**__xsnor_bus_cmd_dummy_receive()**

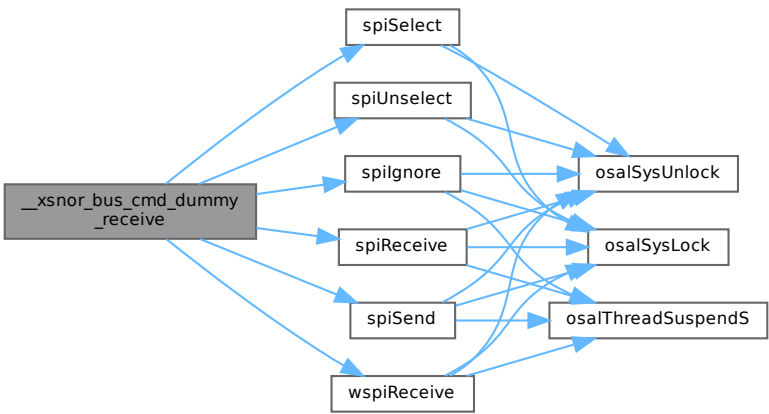
```
void __xsnor_bus_cmd_dummy_receive (
    void * ip,
    uint32_t cmd,
    uint32_t dummy,
    size_t n,
    uint8_t * p)
```

Sends a command followed by dummy cycles and a data receive phase.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>cmd</i>	Instruction code.
in	<i>dummy</i>	Number of dummy cycles.
in	<i>n</i>	Number of bytes to receive.
out	<i>p</i>	Data buffer.

Here is the call graph for this function:



__xsnor_bus_cmd_addr_dummy_receive()

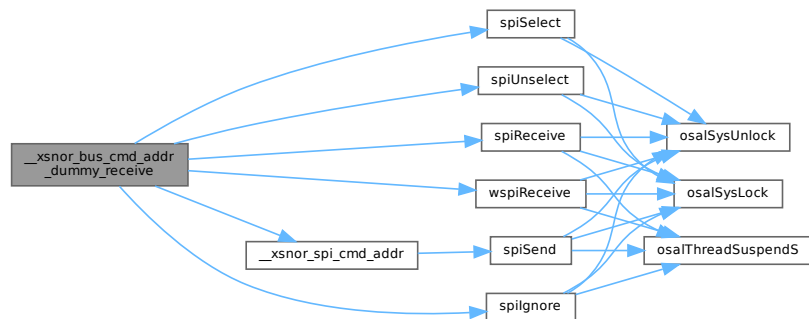
```
void __xsnor_bus_cmd_addr_dummy_receive (
    void * ip,
    uint32_t cmd,
    flash_offset_t offset,
    uint32_t dummy,
    size_t n,
    uint8_t * p)
```

Sends a complete header followed by a data receive phase.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>cmd</i>	Instruction code.
in	<i>offset</i>	Flash offset.
in	<i>dummy</i>	Number of dummy cycles.
in	<i>n</i>	Number of bytes to receive.
out	<i>p</i>	Data buffer.

Here is the call graph for this function:



xsnorStart()

```
flash_error_t xsnorStart (
    void * ip,
    const xsnor_config_t * config)
```

Configures and activates a SNOR driver.

Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_base_c</code> instance.
in	<i>config</i>	Pointer to the configuration.

Returns

An error code.

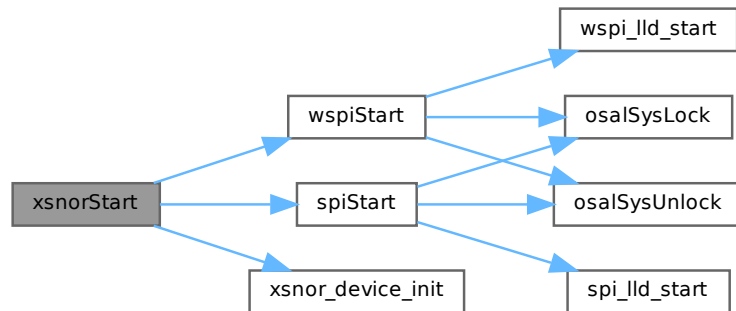
Return values

<code>FLASH_NO_ERROR</code>	Operation successful.
<code>FLASH_ERROR_HW_FAILURE</code>	If initialization failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**xsnorStop()**

```
void xsnorStop (
    void * ip)
```

Deactivates a SNOR driver.

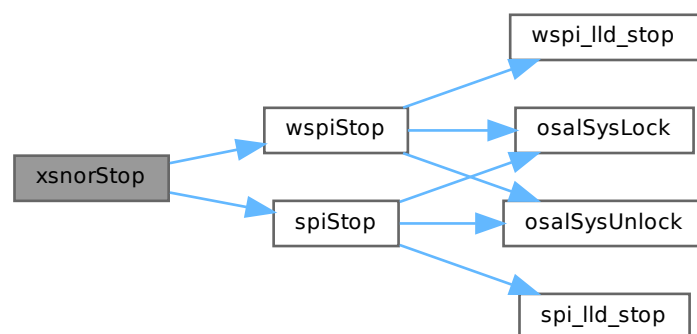
Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



xsnorMemoryMap()

```
flash_error_t xsnorMemoryMap (  
    void * ip,  
    uint8_t ** addrp)
```

Enters the memory mapped mode.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
out	<i>addrp</i>	Pointer to the memory mapped memory or NULL

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	Operation successful.
<i>FLASH_ERROR_HW_FAILURE</i>	If memory mapped mode failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



xsnorMemoryUnmap()

```
void xsnorMemoryUnmap (  
    void * ip)
```

Leaves the memory mapped mode.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**xsnor_device_init()**

```
CC_FORCE_INLINE flash_error_t xsnor_device_init (
    void * ip) [inline], [static]
```

xsnor_device_read()

```
CC_FORCE_INLINE flash_error_t xsnor_device_read (
    void * ip,
    flash_offset_t offset,
    size_t n,
    uint8_t * rp) [inline], [static]
```

Read operation.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be read.
out	<i>rp</i>	Pointer to the data buffer.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	Operation successful.
<i>FLASH_ERROR_READ</i>	If the read operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

xsnor_device_program()

```
CC_FORCE_INLINE flash_error_t xsnor_device_program (  
    void * ip,  
    flash_offset_t offset,  
    size_t n,  
    const uint8_t * pp) [inline], [static]
```

Program operation.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be programmed.
in	<i>pp</i>	Pointer to the data buffer.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	Operation successful.
<i>FLASH_ERROR_PROGRAM</i>	If the program operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

xsnor_device_start_erase_all()

```
CC_FORCE_INLINE flash_error_t xsnor_device_start_erase_all (  
    void * ip) [inline], [static]
```

Starts a whole-device erase operation.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	Operation successful.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

xsnor_device_start_erase_sector()

```
CC_FORCE_INLINE flash_error_t xsnor_device_start_erase_sector (
    void * ip,
    flash_sector_t sector) [inline], [static]
```

Starts an sector erase operation.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>sector</i>	Sector to be erased.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	Operation successful.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

xsnor_device_query_erase()

```
CC_FORCE_INLINE flash_error_t xsnor_device_query_erase (
    void * ip,
    unsigned * msec) [inline], [static]
```

Queries the driver for erase operation progress.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
out	<i>msec</i>	Recommended time, in milliseconds, that should be spent before calling this function again, can be NULL

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	If there is no erase operation in progress.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

xsnor_device_verify_erase()

```
CC_FORCE_INLINE flash_error_t xsnor_device_verify_erase (
    void * ip,
    flash_sector_t sector) [inline], [static]
```

Returns the erase state of a sector.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_base_c instance.
in	<i>sector</i>	Sector to be verified.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	Operation successful.
<i>FLASH_ERROR_VERIFY</i>	If the verify operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

Function Class:

Not an API, this function is for internal use only.

xsnor_device_mmap_on()

```
CC_FORCE_INLINE flash_error_t xsnor_device_mmap_on (
    void * ip,
    uint8_t ** addrp) [inline], [static]
```

xsnor_device_mmap_off()

```
CC_FORCE_INLINE void xsnor_device_mmap_off (
    void * ip) [inline], [static]
```

__xsnor_fls_get_descriptor_impl()

```
const flash_descriptor_t * __xsnor_fls_get_descriptor_impl (
    void * ip) [static]
```

Implementation of interface method [flsGetDescriptor\(\)](#).

Parameters

in, out	<i>ip</i>	Pointer to the flash_interface_i class interface.

Returns

A flash device descriptor.

__xsnor_fls_read_impl()

```
flash_error_t __xsnor_fls_read_impl (
    void * ip,
    flash_offset_t offset,
    size_t n,
    uint8_t * rp) [static]
```

Implementation of interface method `flsRead()`.

Parameters

in, out	<i>ip</i>	Pointer to the <code>flash_interface_i</code> class interface.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be read.
out	<i>rp</i>	Pointer to the data buffer.

Returns

An error code.

Here is the call graph for this function:

**__xsnor_fls_program_impl()**

```
flash_error_t __xsnor_fls_program_impl (
    void * ip,
    flash_offset_t offset,
    size_t n,
    const uint8_t * pp) [static]
```

Implementation of interface method `flsProgram()`.

Parameters

in, out	<i>ip</i>	Pointer to the <code>flash_interface_i</code> class interface.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be programmed.
in	<i>pp</i>	Pointer to the data buffer.

Returns

An error code.

Here is the call graph for this function:

**__xsnor_fls_start_erase_all_impl()**

```
flash_error_t __xsnor_fls_start_erase_all_impl (
    void * ip) [static]
```

Implementation of interface method `flsStartEraseAll()`.

Parameters

in, out	ip	Pointer to the <code>flash_interface_i</code> class interface.

Returns

An error code.

Here is the call graph for this function:

**__xsnor_fls_start_erase_sector_impl()**

```
flash_error_t __xsnor_fls_start_erase_sector_impl (
    void * ip,
    flash_sector_t sector) [static]
```

Implementation of interface method `flsStartEraseSector()`.

Parameters

in, out	<i>ip</i>	Pointer to the flash_interface_i class interface.
in	<i>sector</i>	Sector to be erased.

Returns

An error code.

Here is the call graph for this function:

**`__xsnor_fls_query_erase_impl()`**

```
flash_error_t __xsnor_fls_query_erase_impl (  
    void * ip,  
    unsigned * msec) [static]
```

Implementation of interface method `flsQueryErase()`.

Parameters

in, out	<i>ip</i>	Pointer to the flash_interface_i class interface.
out	<i>msec</i>	Recommended time, in milliseconds, that should be spent before calling this function again, can be NULL

Returns

An error code.

Here is the call graph for this function:



__xsnor_fls_verify_erase_impl()

```
flash_error_t __xsnor_fls_verify_erase_impl (
    void * ip,
    flash_sector_t sector) [static]
```

Implementation of interface method `flsVerifyErase()`.

Parameters

in, out	<i>ip</i>	Pointer to the <code>flash_interface_i</code> class interface.
in	<i>sector</i>	Sector to be verified.

Returns

An error code.

Here is the call graph for this function:

**__xsnor_fls_acquire_exclusive_impl()**

```
flash_error_t __xsnor_fls_acquire_exclusive_impl (
    void * ip) [static]
```

Implementation of interface method `flsAcquireExclusive()`.

Parameters

in, out	<i>ip</i>	Pointer to the <code>flash_interface_i</code> class interface.

Returns

An error code.

Here is the call graph for this function:



__xsnor_fls_release_exclusive_impl()

```
flash_error_t __xsnor_fls_release_exclusive_impl (
    void * ip) [static]
```

Implementation of interface method `flsReleaseExclusive()`.

Parameters

in, out	ip	Pointer to the <code>flash_interface_i</code> class interface.

Returns

An error code.

Here is the call graph for this function:

**7.2.10.7.3 Base Flash Interface**

Base Flash Interface.

7.2.10.7.3.1 Detailed Description

Base Flash Interface.

Flash Interface.

This interface defines a general API for accessing Flash devices.

Base interface for flash devices.

Interface @p flash_interface_i structures

- typedef struct `flash_interface` `flash_interface_i`
Type of a flash interface interface.

Virtual methods of `flash_interface_i`

- static `CC_FORCE_INLINE` const `flash_descriptor_t` * `flsGetDescriptor` (void *ip)
Gets the flash descriptor structure.
- static `CC_FORCE_INLINE` `flash_error_t` `flsRead` (void *ip, `flash_offset_t` offset, `size_t` n, `uint8_t` *rp)
Read operation.
- static `CC_FORCE_INLINE` `flash_error_t` `flsProgram` (void *ip, `flash_offset_t` offset, `size_t` n, const `uint8_t` *pp)
Program operation.
- static `CC_FORCE_INLINE` `flash_error_t` `flsStartEraseAll` (void *ip)
Starts a whole-device erase operation.
- static `CC_FORCE_INLINE` `flash_error_t` `flsStartEraseSector` (void *ip, `flash_sector_t` sector)
Starts an sector erase operation.
- static `CC_FORCE_INLINE` `flash_error_t` `flsQueryErase` (void *ip, unsigned *msec)
Queries the driver for erase operation progress.
- static `CC_FORCE_INLINE` `flash_error_t` `flsVerifyErase` (void *ip, `flash_sector_t` sector)
Returns the erase state of a sector.
- static `CC_FORCE_INLINE` `flash_error_t` `flsAcquireExclusive` (void *ip)
Acquires exclusive access to flash.
- static `CC_FORCE_INLINE` `flash_error_t` `flsReleaseExclusive` (void *ip)
Releases exclusive access to flash.

Data Structures

- struct `flash_interface_vmt`
Interface `flash_interface_i` virtual methods table.
- struct `flash_interface`
Structure representing a flash interface interface.
- interface `flash_interface_i`
Generic flash interface.

7.2.10.7.3.2 Typedef Documentation

`flash_interface_i`

```
typedef struct flash_interface flash_interface_i
```

Type of a flash interface interface.

7.2.10.7.3.3 Function Documentation

`flsGetDescriptor()`

```
CC_FORCE_INLINE const flash_descriptor_t * flsGetDescriptor (  
    void * ip)    [inline], [static]
```

Gets the flash descriptor structure.

Parameters

in, out	<i>ip</i>	Pointer to a flash_interface_i instance.

Returns

A flash device descriptor.

flsRead()

```
CC_FORCE_INLINE flash_error_t flsRead (
    void * ip,
    flash_offset_t offset,
    size_t n,
    uint8_t * rp) [inline], [static]
```

Read operation.

Parameters

in, out	<i>ip</i>	Pointer to a flash_interface_i instance.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be read.
out	<i>rp</i>	Pointer to the data buffer.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	If there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	If there is an erase operation in progress.
<i>FLASH_ERROR_READ</i>	If the read operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

flsProgram()

```
CC_FORCE_INLINE flash_error_t flsProgram (
    void * ip,
    flash_offset_t offset,
    size_t n,
    const uint8_t * pp) [inline], [static]
```

Program operation.

Parameters

in, out	<i>ip</i>	Pointer to a flash_interface_i instance.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be programmed.
in	<i>pp</i>	Pointer to the data buffer.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	If there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	If there is an erase operation in progress.
<i>FLASH_ERROR_PROGRAM</i>	If the program operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

flsStartEraseAll()

```
CC_FORCE_INLINE flash_error_t flsStartEraseAll (
    void * ip) [inline], [static]
```

Starts a whole-device erase operation.

Parameters

in, out	<i>ip</i>	Pointer to a flash_interface_i instance.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	If there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	If there is an erase operation in progress.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

flsStartEraseSector()

```
CC_FORCE_INLINE flash_error_t flsStartEraseSector (
    void * ip,
    flash_sector_t sector) [inline], [static]
```

Starts an sector erase operation.

Parameters

in, out	<i>ip</i>	Pointer to a flash_interface_i instance.
in	<i>sector</i>	Sector to be erased.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	If there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	If there is an erase operation in progress.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

flsQueryErase()

```
CC_FORCE_INLINE flash_error_t flsQueryErase (
    void * ip,
    unsigned * msec) [inline], [static]
```

Queries the driver for erase operation progress.

Parameters

in, out	<i>ip</i>	Pointer to a flash_interface_i instance.
out	<i>msec</i>	Recommended time, in milliseconds, that should be spent before calling this function again, can be NULL

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	If there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	If there is an erase operation in progress.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

flsVerifyErase()

```
CC_FORCE_INLINE flash_error_t flsVerifyErase (
    void * ip,
    flash_sector_t sector) [inline], [static]
```

Returns the erase state of a sector.

Parameters

in, out	<i>ip</i>	Pointer to a flash_interface_i instance.
in	<i>sector</i>	Sector to be verified.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	If there is no erase operation in progress.
<i>FLASH_BUSY_ERASING</i>	If there is an erase operation in progress.
<i>FLASH_ERROR_VERIFY</i>	If the verify operation failed.
<i>FLASH_ERROR_HW_FAILURE</i>	If access to the memory failed.

flsAcquireExclusive()

```
CC_FORCE_INLINE flash_error_t flsAcquireExclusive (
    void * ip) [inline], [static]
```

Acquires exclusive access to flash.

Parameters

in, out	<i>ip</i>	Pointer to a flash_interface_i instance.

Returns

An error code.

Return values

<i>FLASH_NO_ERROR</i>	If there is no erase operation in progress.
<i>FLASH_ERROR_UNIMPLEMENTED</i>	UNIMPLEMENTED if exclusive access not enabled.

flsReleaseExclusive()

```
CC_FORCE_INLINE flash_error_t flsReleaseExclusive (
    void * ip) [inline], [static]
```

Releases exclusive access to flash.

Parameters

in, out	<i>ip</i>	Pointer to a flash_interface_i instance.

Returns

An error code.

Return values

<code>FLASH_NO_ERROR</code>	If there is no erase operation in progress.
<code>FLASH_ERROR_UNIMPLEMENTED</code>	UNIMPLEMENTED if exclusive access not enabled.

7.2.10.7.4 Macronix MX25 SNOR Driver

Macronix MX25 Driver.

7.2.10.7.4.1 Detailed Description

Macronix MX25 Driver.

SNOR Macronix MX25 driver.

This module implements a Macronix MX25 SNOR driver class inheriting from `hal_xsnor_base_c` class.

Module for SNOR Macronix MX25 flash devices.

Class @p hal_xsnor_macronix_mx25_c structures

- typedef struct `hal_xsnor_macronix_mx25` `hal_xsnor_macronix_mx25_c`
Type of a SNOR Macronix MX25 driver class.

Methods implementations of hal_xsnor_macronix_mx25_c

- void * `__mx25_objinit_impl` (void *ip, const void *vmt)
Implementation of object creation.
- void `__mx25_dispose_impl` (void *ip)
Implementation of object finalization.
- `flash_error_t __mx25_init_impl` (void *ip)
Override of method `xsnor_device_init()`.
- `flash_error_t __mx25_read_impl` (void *ip, `flash_offset_t` offset, `size_t` n, `uint8_t` *rp)
Override of method `xsnor_device_read()`.
- `flash_error_t __mx25_program_impl` (void *ip, `flash_offset_t` offset, `size_t` n, const `uint8_t` *pp)
Override of method `xsnor_device_program()`.
- `flash_error_t __mx25_start_erase_all_impl` (void *ip)
Override of method `xsnor_device_start_erase_all()`.
- `flash_error_t __mx25_start_erase_sector_impl` (void *ip, `flash_sector_t` sector)
Override of method `xsnor_device_start_erase_sector()`.
- `flash_error_t __mx25_query_erase_impl` (void *ip, unsigned *msec)
Override of method `xsnor_device_query_erase()`.
- `flash_error_t __mx25_verify_erase_impl` (void *ip, `flash_sector_t` sector)
Override of method `xsnor_device_verify_erase()`.
- `flash_error_t __mx25_mmap_on_impl` (void *ip, `uint8_t` **addrp)
Override of method `xsnor_device_mmap_on()`.
- void `__mx25_mmap_off_impl` (void *ip)
Override of method `xsnor_device_mmap_off()`.

Default constructor of `hal_xsnor_macronix_mx25_c`

- static `CC_FORCE_INLINE hal_xsnor_macronix_mx25_c * mx25ObjectInit (hal_xsnor_macronix_mx25_c *self)`

Default initialization function of `hal_xsnor_macronix_mx25_c`.

Device options

- `#define MX25_OPT_DUMMY_CYCLES_MASK (15U << 0)`
Mask of the dummy cycles field.
- `#define MX25_OPT_DUMMY_CYCLES(n)`
Number of dummy cycles.
- `#define MX25_OPT_NO_WIDTH_SWITCH (1U << 4)`
Switch bus width on initialization.
- `#define MX25_OPT_USE_SUBSECTORS (1U << 5)`
Use 4kB sub-sectors rather than 64kB sectors.
- `#define MX25_OPT_NICE_WAITING (1U << 6)`
Delays insertion.
- `#define MX25_OPT_USE_DTR (1U << 7)`
Enables DTR in 8 lines mode.

Command codes, SPI mode

- `#define CMD_SPI_READ3B 0x03U`
- `#define CMD_SPI_FAST_READ3B 0x0BU`
- `#define CMD_SPI_PP3B 0x02U`
- `#define CMD_SPI_SE3B 0x20U`
- `#define CMD_SPI_BE3B 0xD8U`
- `#define CMD_SPI_CE 0xC7U`
- `#define CMD_SPI_READ4B 0x13U`
- `#define CMD_SPI_FAST_READ4B 0x0CU`
- `#define CMD_SPI_PP4B 0x12U`
- `#define CMD_SPI_SE4B 0x21U`
- `#define CMD_SPI_BE4B 0xDCU`
- `#define CMD_SPI_WREN 0x06U`
- `#define CMD_SPI_WRDI 0x04U`
- `#define CMD_SPI_PE_SUSPEND 0xB0U`
- `#define CMD_SPI_PE_RESUME 0x30U`
- `#define CMD_SPI_DP 0xB9U`
- `#define CMD_SPI_SBL 0xC0U`
- `#define CMD_SPI_ENSO 0xB1U`
- `#define CMD_SPI_EXSO 0xC1U`
- `#define CMD_SPI_NOP 0x00U`
- `#define CMD_SPI_RSTEN 0x66U`
- `#define CMD_SPI_RST 0x99U`
- `#define CMD_SPI_RDID 0x9FU`
- `#define CMD_SPI_RDSFDP 0x5AU`
- `#define CMD_SPI_RDSR 0x05U`
- `#define CMD_SPI_RDCCR 0x15U`
- `#define CMD_SPI_WRSR 0x01U`
- `#define CMD_SPI_RDCCR2 0x71U`
- `#define CMD_SPI_WRCR2 0x72U`

- #define `CMD_SPI_RDFBR` 0x16U
- #define `CMD_SPI_WRFBR` 0x17U
- #define `CMD_SPI_ESFBR` 0x18U
- #define `CMD_SPI_RDSCUR` 0x2BU
- #define `CMD_SPI_WRSCUR` 0x2FU
- #define `CMD_SPI_WRLR` 0x2CU
- #define `CMD_SPI_RDLR` 0x2DU
- #define `CMD_SPI_WRSPB` 0xE3U
- #define `CMD_SPI_ESSPB` 0xE4U
- #define `CMD_SPI_RDSPB` 0xE2U
- #define `CMD_SPI_WRDPB` 0xE1U
- #define `CMD_SPI_RDDPB` 0xE0U
- #define `CMD_SPI_WPSEL` 0x68U
- #define `CMD_SPI_GBLK` 0x7EU
- #define `CMD_SPI_GBULK` 0x98U
- #define `CMD_SPI_RDPASS` 0x27U
- #define `CMD_SPI_WRPASS` 0x28U
- #define `CMD_SPI_PASSULK` 0x29U

Command codes, OPI mode

- #define `CMD_OPI_8READ` 0xEC13U
- #define `CMD_OPI_8DTRD` 0xEE11U
- #define `CMD_OPI_RDID` 0x9F60U
- #define `CMD_OPI_RDSFDP` 0x5AA5U
- #define `CMD_OPI_PP` 0x12EDU
- #define `CMD_OPI_SE` 0x21DEU
- #define `CMD_OPI_BE` 0xDC23U
- #define `CMD_OPI_CE` 0xC738U
- #define `CMD_OPI_WREN` 0x06F9U
- #define `CMD_OPI_WRDI` 0x04FBU
- #define `CMD_OPI_PE_SUSPEND` 0xB04FU
- #define `CMD_OPI_PE_RESUME` 0x30CFU
- #define `CMD_OPI_DP` 0xB946U
- #define `CMD_OPI_SBL` 0xC03FU
- #define `CMD_OPI_ENSO` 0xB14EU
- #define `CMD_OPI_EXSO` 0xC13EU
- #define `CMD_OPI_NOP` 0x00FFU
- #define `CMD_OPI_RSTEN` 0x6699U
- #define `CMD_OPI_RST` 0x9966U
- #define `CMD_OPI_RDSR` 0x05FAU
- #define `CMD_OPI_RDCCR` 0x15EAU
- #define `CMD_OPI_WRSR` 0x01FEU
- #define `CMD_OPI_WRCR` 0x01FEU
- #define `CMD_OPI_RDCCR2` 0x718EU
- #define `CMD_OPI_WRCR2` 0x728DU
- #define `CMD_OPI_RDFBR` 0x16E9U
- #define `CMD_OPI_WRFBR` 0x17E8U
- #define `CMD_OPI_ESFBR` 0x18E7U
- #define `CMD_OPI_RDSCUR` 0x2BD4U
- #define `CMD_OPI_WRSCUR` 0x2FD0U
- #define `CMD_OPI_WRLR` 0x2CD3U
- #define `CMD_OPI_RDLR` 0x2DD2U
- #define `CMD_OPI_WRSPB` 0xE31CU

- `#define CMD_OPI_ESSPB 0xE41BU`
- `#define CMD_OPI_RDSPB 0xE21DU`
- `#define CMD_OPI_WRDPB 0xE11EU`
- `#define CMD_OPI_RDDPB 0xE01FU`
- `#define CMD_OPI_WPSEL 0x6897U`
- `#define CMD_OPI_GBLK 0x7E81U`
- `#define CMD_OPI_GBULK 0x9867U`
- `#define CMD_OPI_RDPASS 0x27D8U`
- `#define CMD_OPI_WRPASS 0x28D7U`
- `#define CMD_OPI_PASSULK 0x29D6U`

Flags status register bits

- `#define FLAGS_WPSEL 0x80U`
- `#define FLAGS_E_FAIL 0x40U`
- `#define FLAGS_P_FAIL 0x20U`
- `#define FLAGS_ESB 0x08U`
- `#define FLAGS_PSB 0x04U`
- `#define FLAGS_LDSO 0x02U`
- `#define FLAGS_SECURED_OTP 0x01U`
- `#define FLAGS_ALL_ERRORS (FLAGS_E_FAIL | FLAGS_P_FAIL)`

MX25-required transfer modes

- `#define CFG_C8_SPI`
- `#define CFG_C16_8STR`
- `#define CFG_C16_8DTR`
- `#define CFG_C8_A32_SPI`
- `#define CFG_C16_A32_8STR`
- `#define CFG_C16_A32_8DTR`
- `#define CFG_C8_DATA_SPI`
- `#define CFG_C16_DATA_8STR`
- `#define CFG_C16_DATA_8DTR`
- `#define CFG_C8_A32_DATA_SPI`
- `#define CFG_C16_A32_DATA_8STR`
- `#define CFG_C16_A32_DATA_8DTR`

Data Structures

- struct `hal_xsnor_macronix_mx25_vmt`
Class `hal_xsnor_macronix_mx25_c` virtual methods table.
- struct `hal_xsnor_macronix_mx25`
Structure representing a SNOR Macronix MX25 driver class.
- class `hal_xsnor_macronix_mx25_c`

Macros

- `#define PAGE_SIZE 256U`
- `#define PAGE_MASK (PAGE_SIZE - 1U)`

Functions

- static bool `mx25_find_id` (const uint8_t *set, size_t size, uint8_t element)
- static `flash_error_t mx25_poll_status` (hal_xsnor_macronix_mx25_c *self)
- static void `mx25_reset_memory` (hal_xsnor_macronix_mx25_c *self)
- static void `mx25_read_id` (hal_xsnor_macronix_mx25_c *self, uint8_t *buf)
- static void `mx25_write_cr2` (hal_xsnor_macronix_mx25_c *self, uint32_t addr, const uint8_t *value)

Variables

- static const `xsnor_commands_t cmd_spi`
- static const `xsnor_commands_t cmd_8str`
- static const `xsnor_commands_t cmd_8dtr`
- static const `wspi_command_t cmd_reset_enable_spi`
- static const `wspi_command_t cmd_reset_memory_spi`
- static const `wspi_command_t cmd_reset_enable_8str`
- static const `wspi_command_t cmd_reset_memory_8str`
- static const `wspi_command_t cmd_reset_enable_8dtr`
- static const `wspi_command_t cmd_reset_memory_8dtr`
- static const uint8_t `mx25_manufacturer_ids` [] = {0xC2}
- static const uint8_t `mx25_memory_type_ids` [] = {0x85}
- const struct `hal_xsnor_macronix_mx25_vmt __hal_xsnor_macronix_mx25_vmt`

VMT structure of SNOR Macronix MX25 driver class.

7.2.10.7.4.2 Macro Definition Documentation

MX25_OPT_DUMMY_CYCLES_MASK

```
#define MX25_OPT_DUMMY_CYCLES_MASK (15U << 0)
```

Mask of the dummy cycles field.

MX25_OPT_DUMMY_CYCLES

```
#define MX25_OPT_DUMMY_CYCLES(  
    n)
```

Value:

```
((n) << 0)
```

Number of dummy cycles.

Parameters

<i>n</i>	Number of dummy cycles (2..15)

MX25_OPT_NO_WIDTH_SWITCH

```
#define MX25_OPT_NO_WIDTH_SWITCH (1U << 4)
```

Switch bus width on initialization.

If `MX25_OPT_NO_WIDTH_SWITCH` is specified then this is the bus mode that the device is expected to be using else this is the bus mode that the device will be switched in.

Note

This option is only valid in WSPI bus mode.

MX25_OPT_USE_SUBSECTORS

```
#define MX25_OPT_USE_SUBSECTORS (1U << 5)
```

Use 4kB sub-sectors rather than 64kB sectors.

MX25_OPT_NICE_WAITING

```
#define MX25_OPT_NICE_WAITING (1U << 6)
```

Delays insertion.

If enabled this options inserts delays into the flash waiting routines releasing some extra CPU time for threads with lower priority, this may slow down the driver a bit however.

MX25_OPT_USE_DTR

```
#define MX25_OPT_USE_DTR (1U << 7)
```

Enables DTR in 8 lines mode.

PAGE_SIZE

```
#define PAGE_SIZE 256U
```

PAGE_MASK

```
#define PAGE_MASK (PAGE_SIZE - 1U)
```

CMD_SPI_READ3B

```
#define CMD_SPI_READ3B 0x03U
```

CMD_SPI_FAST_READ3B

```
#define CMD_SPI_FAST_READ3B 0x0BU
```

CMD_SPI_PP3B

```
#define CMD_SPI_PP3B 0x02U
```

CMD_SPI_SE3B

```
#define CMD_SPI_SE3B 0x20U
```

CMD_SPI_BE3B

```
#define CMD_SPI_BE3B 0xD8U
```

CMD_SPI_CE

```
#define CMD_SPI_CE 0xC7U
```

CMD_SPI_READ4B

```
#define CMD_SPI_READ4B 0x13U
```

CMD_SPI_FAST_READ4B

```
#define CMD_SPI_FAST_READ4B 0x0CU
```

CMD_SPI_PP4B

```
#define CMD_SPI_PP4B 0x12U
```

CMD_SPI_SE4B

```
#define CMD_SPI_SE4B 0x21U
```

CMD_SPI_BE4B

```
#define CMD_SPI_BE4B 0xDCU
```

CMD_SPI_WREN

```
#define CMD_SPI_WREN 0x06U
```

CMD_SPI_WRDI

```
#define CMD_SPI_WRDI 0x04U
```

CMD_SPI_PE_SUSPEND

```
#define CMD_SPI_PE_SUSPEND 0xB0U
```

CMD_SPI_PE_RESUME

```
#define CMD_SPI_PE_RESUME 0x30U
```

CMD_SPI_DP

```
#define CMD_SPI_DP 0xB9U
```

CMD_SPI_SBL

```
#define CMD_SPI_SBL 0xC0U
```

CMD_SPI_ENSO

```
#define CMD_SPI_ENSO 0xB1U
```

CMD_SPI_EXSO

```
#define CMD_SPI_EXSO 0xC1U
```

CMD_SPI_NOP

```
#define CMD_SPI_NOP 0x00U
```

CMD_SPI_RSTEN

```
#define CMD_SPI_RSTEN 0x66U
```

CMD_SPI_RST

```
#define CMD_SPI_RST 0x99U
```

CMD_SPI_RDID

```
#define CMD_SPI_RDID 0x9FU
```

CMD_SPI_RDSFDP

```
#define CMD_SPI_RDSFDP 0x5AU
```

CMD_SPI_RDSR

```
#define CMD_SPI_RDSR 0x05U
```

CMD_SPI_RDCCR

```
#define CMD_SPI_RDCCR 0x15U
```

CMD_SPI_WRSR

```
#define CMD_SPI_WRSR 0x01U
```

CMD_SPI_RDCCR2

```
#define CMD_SPI_RDCCR2 0x71U
```

CMD_SPI_WRCR2

```
#define CMD_SPI_WRCR2 0x72U
```

CMD_SPI_RDFBR

```
#define CMD_SPI_RDFBR 0x16U
```

CMD_SPI_WRFBR

```
#define CMD_SPI_WRFBR 0x17U
```

CMD_SPI_ESFBR

```
#define CMD_SPI_ESFBR 0x18U
```

CMD_SPI_RDSCUR

```
#define CMD_SPI_RDSCUR 0x2BU
```

CMD_SPI_WRSCUR

```
#define CMD_SPI_WRSCUR 0x2FU
```

CMD_SPI_WRLR

```
#define CMD_SPI_WRLR 0x2CU
```

CMD_SPI_RDLR

```
#define CMD_SPI_RDLR 0x2DU
```

CMD_SPI_WRSPB

```
#define CMD_SPI_WRSPB 0xE3U
```

CMD_SPI_ESSPB

```
#define CMD_SPI_ESSPB 0xE4U
```

CMD_SPI_RDSPB

```
#define CMD_SPI_RDSPB 0xE2U
```

CMD_SPI_WRDPB

```
#define CMD_SPI_WRDPB 0xE1U
```

CMD_SPI_RDDPB

```
#define CMD_SPI_RDDPB 0xE0U
```


CMD_SPI_WPSEL

```
#define CMD_SPI_WPSEL 0x68U
```

CMD_SPI_GBLK

```
#define CMD_SPI_GBLK 0x7EU
```

CMD_SPI_GBULK

```
#define CMD_SPI_GBULK 0x98U
```

CMD_SPI_RDPASS

```
#define CMD_SPI_RDPASS 0x27U
```

CMD_SPI_WRPASS

```
#define CMD_SPI_WRPASS 0x28U
```

CMD_SPI_PASSULK

```
#define CMD_SPI_PASSULK 0x29U
```

CMD_OPI_8READ

```
#define CMD_OPI_8READ 0xEC13U
```

CMD_OPI_8DTRD

```
#define CMD_OPI_8DTRD 0xEE11U
```

CMD_OPI_RDID

```
#define CMD_OPI_RDID 0x9F60U
```

CMD_OPI_RDSFDP

```
#define CMD_OPI_RDSFDP 0x5AA5U
```

CMD_OPI_PP

```
#define CMD_OPI_PP 0x12EDU
```

CMD_OPI_SE

```
#define CMD_OPI_SE 0x21DEU
```

CMD_OPI_BE

```
#define CMD_OPI_BE 0xDC23U
```

CMD_OPI_CE

```
#define CMD_OPI_CE 0xC738U
```

CMD_OPI_WREN

```
#define CMD_OPI_WREN 0x06F9U
```

CMD_OPI_WRDI

```
#define CMD_OPI_WRDI 0x04FBU
```

CMD_OPI_PE_SUSPEND

```
#define CMD_OPI_PE_SUSPEND 0xB04FU
```

CMD_OPI_PE_RESUME

```
#define CMD_OPI_PE_RESUME 0x30CFU
```

CMD_OPI_DP

```
#define CMD_OPI_DP 0xB946U
```

CMD_OPI_SBL

```
#define CMD_OPI_SBL 0xC03FU
```

CMD_OPI_ENSO

```
#define CMD_OPI_ENSO 0xB14EU
```

CMD_OPI_EXSO

```
#define CMD_OPI_EXSO 0xC13EU
```

CMD_OPI_NOP

```
#define CMD_OPI_NOP 0x00FFU
```

CMD_OPI_RSTEN

```
#define CMD_OPI_RSTEN 0x6699U
```

CMD_OPI_RST

```
#define CMD_OPI_RST 0x9966U
```

CMD_OPI_RDSR

```
#define CMD_OPI_RDSR 0x05FAU
```

CMD_OPI_RDCR

```
#define CMD_OPI_RDCR 0x15EAU
```

CMD_OPI_WRSR

```
#define CMD_OPI_WRSR 0x01FEU
```

CMD_OPI_WRCR

```
#define CMD_OPI_WRCR 0x01FEU
```

CMD_OPI_RDCR2

```
#define CMD_OPI_RDCR2 0x718EU
```

CMD_OPI_WRCR2

```
#define CMD_OPI_WRCR2 0x728DU
```

CMD_OPI_RDFBR

```
#define CMD_OPI_RDFBR 0x16E9U
```

CMD_OPI_WRFBR

```
#define CMD_OPI_WRFBR 0x17E8U
```

CMD_OPI_ESFBR

```
#define CMD_OPI_ESFBR 0x18E7U
```

CMD_OPI_RDSCUR

```
#define CMD_OPI_RDSCUR 0x2BD4U
```

CMD_OPI_WRSCUR

```
#define CMD_OPI_WRSCUR 0x2FD0U
```

CMD_OPI_WRLR

```
#define CMD_OPI_WRLR 0x2CD3U
```

CMD_OPI_RDLR

```
#define CMD_OPI_RDLR 0x2DD2U
```

CMD_OPI_WRSPB

```
#define CMD_OPI_WRSPB 0xE31CU
```

CMD_OPI_ESSPB

```
#define CMD_OPI_ESSPB 0xE41BU
```

CMD_OPI_RDSPB

```
#define CMD_OPI_RDSPB 0xE21DU
```

CMD_OPI_WRDPB

```
#define CMD_OPI_WRDPB 0xE11EU
```

CMD_OPI_RDDPB

```
#define CMD_OPI_RDDPB 0xE01FU
```

CMD_OPI_WPSEL

```
#define CMD_OPI_WPSEL 0x6897U
```

CMD_OPI_GBLK

```
#define CMD_OPI_GBLK 0x7E81U
```

CMD_OPI_GBULK

```
#define CMD_OPI_GBULK 0x9867U
```

CMD_OPI_RDPASS

```
#define CMD_OPI_RDPASS 0x27D8U
```

CMD_OPI_WRPASS

```
#define CMD_OPI_WRPASS 0x28D7U
```

CMD_OPI_PASSULK

```
#define CMD_OPI_PASSULK 0x29D6U
```

FLAGS_WPSEL

```
#define FLAGS_WPSEL 0x80U
```

FLAGS_E_FAIL

```
#define FLAGS_E_FAIL 0x40U
```

FLAGS_P_FAIL

```
#define FLAGS_P_FAIL 0x20U
```

FLAGS_ESB

```
#define FLAGS_ESB 0x08U
```

FLAGS_PSB

```
#define FLAGS_PSB 0x04U
```

FLAGS_LDSO

```
#define FLAGS_LDSO 0x02U
```

FLAGS_SECURED_OTP

```
#define FLAGS_SECURED_OTP 0x01U
```

FLAGS_ALL_ERRORS

```
#define FLAGS_ALL_ERRORS (FLAGS_E_FAIL | FLAGS_P_FAIL)
```

CFG_C8_SPI

```
#define CFG_C8_SPI
```

Value:

```
(WSPI_CFG_CMD_SIZE_8      | \
WSPI_CFG_CMD_MODE_ONE_LINE | \
WSPI_CFG_ADDR_MODE_NONE   | \
WSPI_CFG_ALT_MODE_NONE     | \
WSPI_CFG_DATA_MODE_NONE)
```

CFG_C16_8STR

```
#define CFG_C16_8STR
```

Value:

```
(WSPI_CFG_CMD_SIZE_16      | \
WSPI_CFG_CMD_MODE_EIGHT_LINES | \
WSPI_CFG_ADDR_MODE_NONE   | \
WSPI_CFG_ALT_MODE_NONE     | \
WSPI_CFG_DATA_MODE_NONE)
```

CFG_C16_8DTR

```
#define CFG_C16_8DTR
```

Value:

```
(WSPI_CFG_CMD_SIZE_16      | | \
WSPI_CFG_CMD_MODE_EIGHT_LINES | \
WSPI_CFG_ADDR_MODE_NONE    | | \
WSPI_CFG_ALT_MODE_NONE     | | \
WSPI_CFG_DATA_MODE_NONE    | | \
WSPI_CFG_ALL_DTR)
```

CFG_C8_A32_SPI

```
#define CFG_C8_A32_SPI
```

Value:

```
(WSPI_CFG_CMD_SIZE_8      | | \
WSPI_CFG_CMD_MODE_ONE_LINE | \
WSPI_CFG_ADDR_MODE_ONE_LINE | \
WSPI_CFG_ADDR_SIZE_32     | | \
WSPI_CFG_ALT_MODE_NONE     | | \
WSPI_CFG_DATA_MODE_NONE)
```

CFG_C16_A32_8STR

```
#define CFG_C16_A32_8STR
```

Value:

```
(WSPI_CFG_CMD_SIZE_16      | | \
WSPI_CFG_CMD_MODE_EIGHT_LINES | \
WSPI_CFG_ADDR_MODE_EIGHT_LINES | \
WSPI_CFG_ADDR_SIZE_32     | | \
WSPI_CFG_ALT_MODE_NONE     | | \
WSPI_CFG_DATA_MODE_NONE)
```

CFG_C16_A32_8DTR

```
#define CFG_C16_A32_8DTR
```

Value:

```
(WSPI_CFG_CMD_SIZE_16      | | \
WSPI_CFG_CMD_MODE_EIGHT_LINES | \
WSPI_CFG_ADDR_MODE_EIGHT_LINES | \
WSPI_CFG_ADDR_SIZE_32     | | \
WSPI_CFG_ALT_MODE_NONE     | | \
WSPI_CFG_DATA_MODE_NONE    | | \
WSPI_CFG_ALL_DTR)
```

CFG_C8_DATA_SPI

```
#define CFG_C8_DATA_SPI
```

Value:

```
(WSPI_CFG_CMD_SIZE_8      | | \
WSPI_CFG_CMD_MODE_ONE_LINE | \
WSPI_CFG_ADDR_MODE_NONE    | | \
WSPI_CFG_ALT_MODE_NONE     | | \
WSPI_CFG_DATA_MODE_ONE_LINE)
```

CFG_C16_DATA_8STR

```
#define CFG_C16_DATA_8STR
```

Value:

```
(WSPI_CFG_CMD_SIZE_16          |  \
WSPI_CFG_CMD_MODE_EIGHT_LINES |  \
WSPI_CFG_ADDR_MODE_NONE       |  \
WSPI_CFG_ALT_MODE_NONE        |  \
WSPI_CFG_DATA_MODE_EIGHT_LINES)
```

CFG_C16_DATA_8DTR

```
#define CFG_C16_DATA_8DTR
```

Value:

```
(WSPI_CFG_CMD_SIZE_16          |  \
WSPI_CFG_CMD_MODE_EIGHT_LINES |  \
WSPI_CFG_ADDR_MODE_NONE       |  \
WSPI_CFG_ALT_MODE_NONE        |  \
WSPI_CFG_DATA_MODE_EIGHT_LINES |  \
WSPI_CFG_ALL_DTR              |  \
WSPI_CFG_DQS_ENABLE)
```

CFG_C8_A32_DATA_SPI

```
#define CFG_C8_A32_DATA_SPI
```

Value:

```
(WSPI_CFG_CMD_SIZE_8          |  \
WSPI_CFG_CMD_MODE_ONE_LINE   |  \
WSPI_CFG_ADDR_MODE_ONE_LINE  |  \
WSPI_CFG_ADDR_SIZE_32        |  \
WSPI_CFG_ALT_MODE_NONE       |  \
WSPI_CFG_DATA_MODE_ONE_LINE)
```

CFG_C16_A32_DATA_8STR

```
#define CFG_C16_A32_DATA_8STR
```

Value:

```
(WSPI_CFG_CMD_SIZE_16          |  \
WSPI_CFG_CMD_MODE_EIGHT_LINES |  \
WSPI_CFG_ADDR_MODE_EIGHT_LINES |  \
WSPI_CFG_ADDR_SIZE_32         |  \
WSPI_CFG_ALT_MODE_NONE        |  \
WSPI_CFG_DATA_MODE_EIGHT_LINES)
```

CFG_C16_A32_DATA_8DTR

```
#define CFG_C16_A32_DATA_8DTR
```

Value:

```
(WSPI_CFG_CMD_SIZE_16          |  \
WSPI_CFG_CMD_MODE_EIGHT_LINES |  \
WSPI_CFG_ADDR_MODE_EIGHT_LINES |  \
WSPI_CFG_ADDR_SIZE_32         |  \
WSPI_CFG_ALT_MODE_NONE        |  \
WSPI_CFG_DATA_MODE_EIGHT_LINES |  \
WSPI_CFG_ALL_DTR              |  \
WSPI_CFG_DQS_ENABLE)
```


7.2.10.7.4.3 Typedef Documentation

hal_xsnor_macronix_mx25_c

```
typedef struct hal\_xsnor\_macronix\_mx25 hal_xsnor_macronix_mx25_c
```

Type of a SNOR Macronix MX25 driver class.

7.2.10.7.4.4 Function Documentation

__mx25_objinit_impl()

```
void * __mx25_objinit_impl (  
    void * ip,  
    const void * vmt)
```

Implementation of object creation.

Note

This function is meant to be used by derived classes.

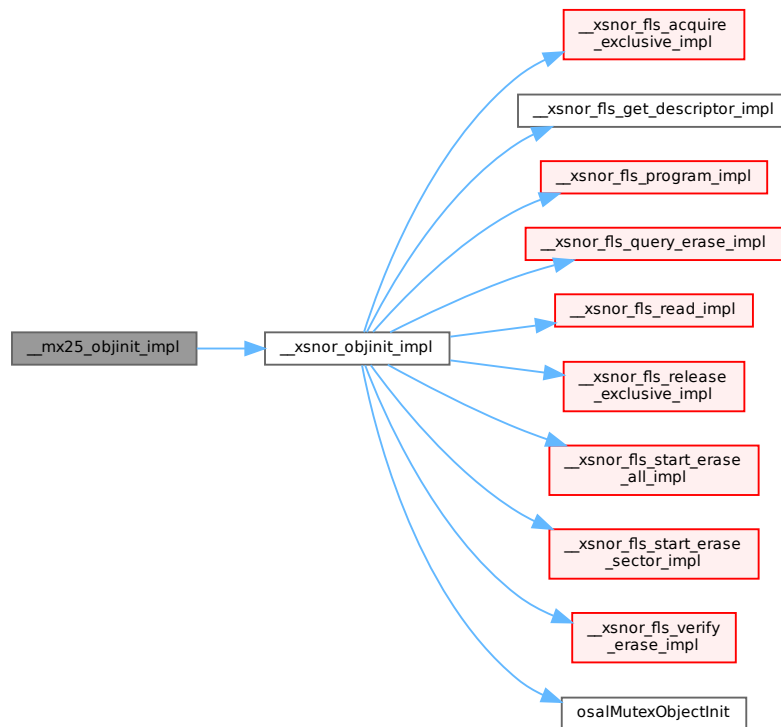
Parameters

out	<i>ip</i>	Pointer to a hal_xsnor_macronix_mx25_c instance to be initialized.
in	<i>vmt</i>	VMT pointer for the new object.

Returns

A new reference to the object.

Here is the call graph for this function:

**__mx25_dispose_impl()**

```
void __mx25_dispose_impl (
    void * ip)
```

Implementation of object finalization.

Note

This function is meant to be used by derived classes.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_macronix_mx25_c instance to be disposed.

Here is the call graph for this function:



`__mx25_init_impl()`

```
flash_error_t __mx25_init_impl (
    void * ip)
```

Override of method `xsnor_device_init()`.

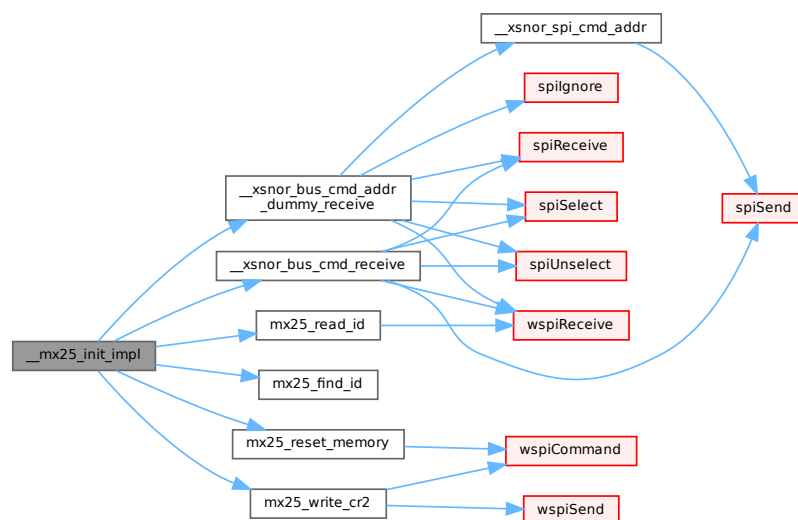
Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_macronix_mx25_c</code> instance.

Returns

An error code.

Here is the call graph for this function:



__mx25_read_impl()

```
flash_error_t __mx25_read_impl (
    void * ip,
    flash_offset_t offset,
    size_t n,
    uint8_t * rp)
```

Override of method `xsnor_device_read()`.

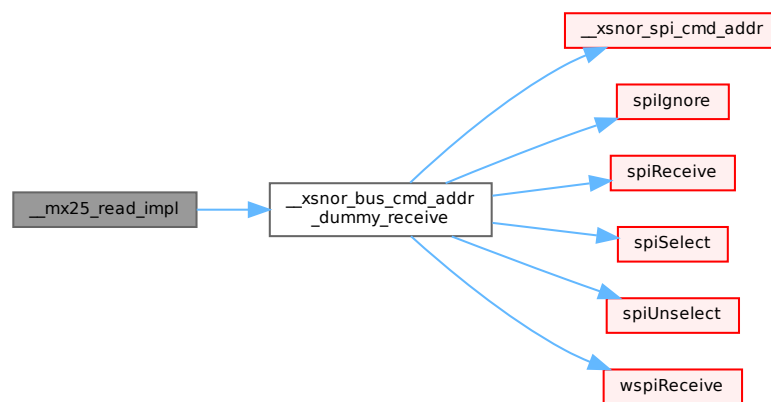
Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_macronix_mx25_c</code> instance.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be read.
out	<i>rp</i>	Pointer to the data buffer.

Returns

An error code.

Here is the call graph for this function:

**__mx25_program_impl()**

```
flash_error_t __mx25_program_impl (
    void * ip,
    flash_offset_t offset,
    size_t n,
    const uint8_t * pp)
```

Override of method `xsnor_device_program()`.

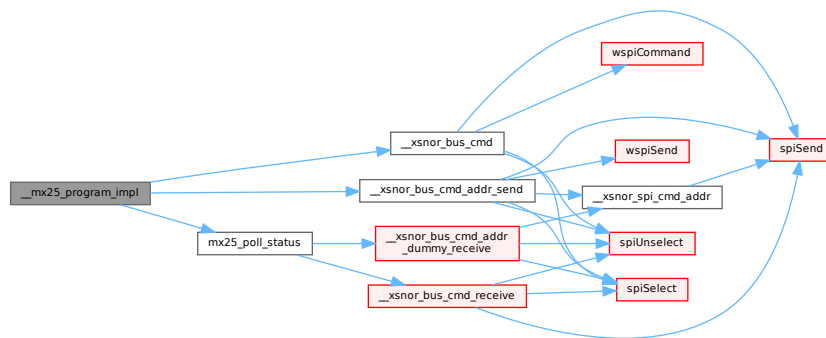
Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_macronix_mx25_c instance.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be programmed.
in	<i>pp</i>	Pointer to the data buffer.

Returns

An error code.

Here is the call graph for this function:

**__mx25_start_erase_all_impl()**

```
flash_error_t __mx25_start_erase_all_impl (
    void * ip)
```

Override of method [xsnor_device_start_erase_all\(\)](#).

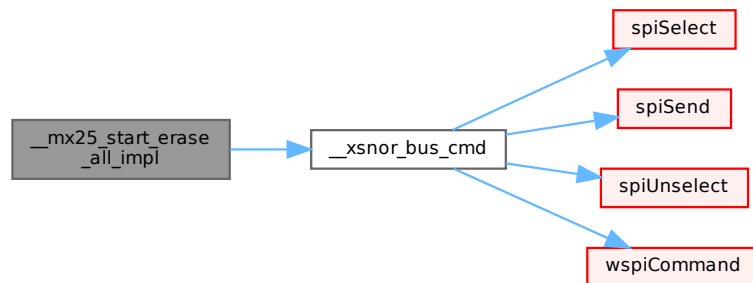
Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_macronix_mx25_c instance.

Returns

An error code.

Here is the call graph for this function:

**__mx25_start_erase_sector_impl()**

```
flash_error_t __mx25_start_erase_sector_impl (
    void * ip,
    flash_sector_t sector)
```

Override of method `xsnor_device_start_erase_sector()`.

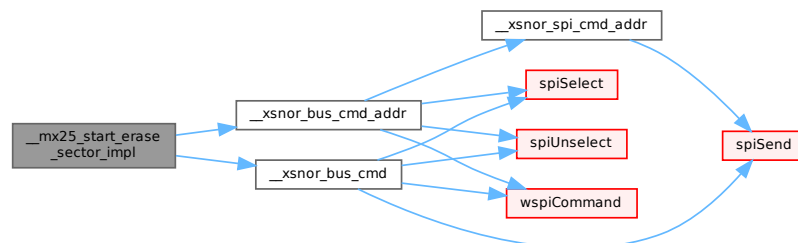
Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_macronix_mx25_c</code> instance.
in	<i>sector</i>	Sector to be erased.

Returns

An error code.

Here is the call graph for this function:



__mx25_query_erase_impl()

```
flash_error_t __mx25_query_erase_impl (
    void * ip,
    unsigned * msec)
```

Override of method `xsnor_device_query_erase()`.

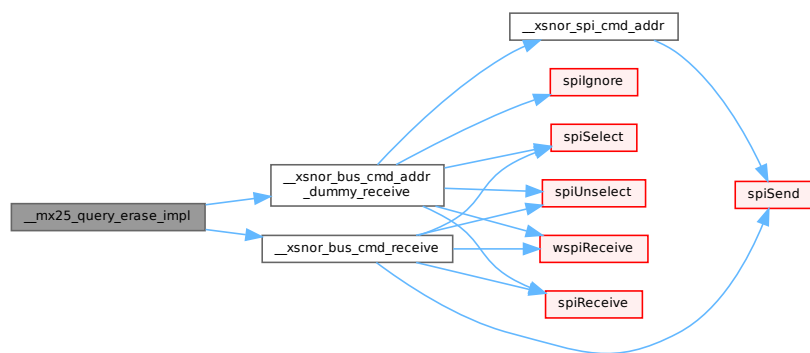
Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_macronix_mx25_c</code> instance.
out	<i>msec</i>	Recommended time, in milliseconds, that should be spent before calling this function again, can be NULL

Returns

An error code.

Here is the call graph for this function:

**__mx25_verify_erase_impl()**

```
flash_error_t __mx25_verify_erase_impl (
    void * ip,
    flash_sector_t sector)
```

Override of method `xsnor_device_verify_erase()`.

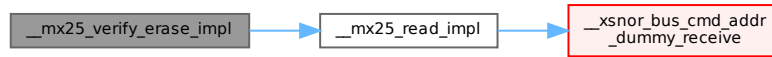
Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_macronix_mx25_c</code> instance.
in	<i>sector</i>	Sector to be verified.

Returns

An error code.

Here is the call graph for this function:

**__mx25_mmap_on_impl()**

```
flash_error_t __mx25_mmap_on_impl (
    void * ip,
    uint8_t ** addrp)
```

Override of method `xsnor_device_mmap_on()`.

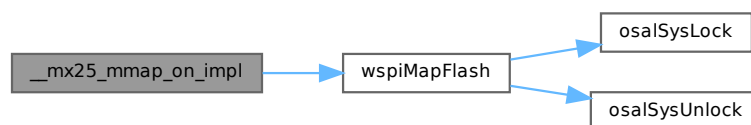
Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_macronix_mx25_c instance.
out	<i>addrp</i>	Pointer to the memory mapped memory or NULL

Returns

An error code.

Here is the call graph for this function:

**__mx25_mmap_off_impl()**

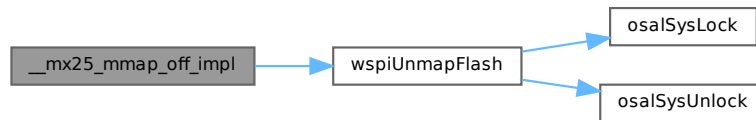
```
void __mx25_mmap_off_impl (
    void * ip)
```

Override of method `xsnor_device_mmap_off()`.

Parameters

<i>in, out</i>	<i>ip</i>	Pointer to a hal_xsnor_macronix_mx25_c instance.

Here is the call graph for this function:

**mx25ObjectInit()**

```
CC_FORCE_INLINE hal_xsnor_macronix_mx25_c * mx25ObjectInit (
    hal_xsnor_macronix_mx25_c * self) [inline], [static]
```

Default initialization function of [hal_xsnor_macronix_mx25_c](#).

Parameters

<i>out</i>	<i>self</i>	Pointer to a hal_xsnor_macronix_mx25_c instance to be initialized.

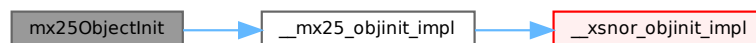
Returns

Pointer to the initialized object.

Function Class:

Object instance constructor.

Here is the call graph for this function:

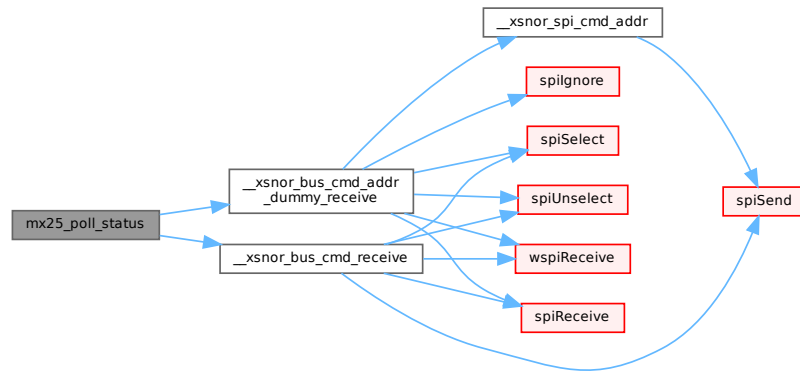
**mx25_find_id()**

```
bool mx25_find_id (
    const uint8_t * set,
    size_t size,
    uint8_t element) [static]
```

mx25_poll_status()

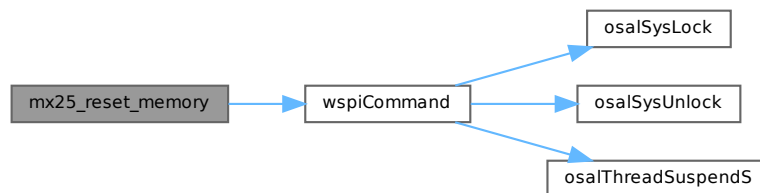
```
flash_error_t mx25_poll_status (
    hal_xsnor_macronix_mx25_c * self) [static]
```

Here is the call graph for this function:

**mx25_reset_memory()**

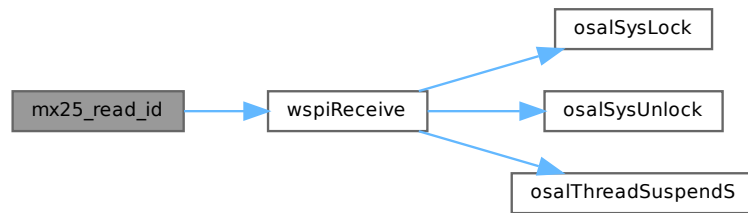
```
void mx25_reset_memory (
    hal_xsnor_macronix_mx25_c * self) [static]
```

Here is the call graph for this function:

**mx25_read_id()**

```
void mx25_read_id (
    hal_xsnor_macronix_mx25_c * self,
    uint8_t * buf) [static]
```

Here is the call graph for this function:

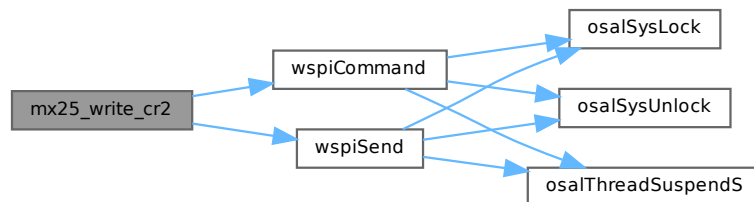


mx25_write_cr2()

```

void mx25_write_cr2 (
    hal_xsnor_macronix_mx25_c * self,
    uint32_t addr,
    const uint8_t * value) [static]
  
```

Here is the call graph for this function:



7.2.10.7.4.5 Variable Documentation

cmd_spi

```
const xsnor_commands_t cmd_spi [static]
```

Initial value:

```

= {
    .cmd          = CFG_C8_SPI,
    .cmd_addr     = CFG_C8_A32_SPI,
    .cmd_data     = CFG_C8_DATA_SPI,
    .cmd_addr_data = CFG_C8_A32_DATA_SPI
}
  
```

cmd_8str

```
const xsnor_commands_t cmd_8str [static]
```

Initial value:

```
= {
    .cmd           = CFG_C16_8STR,
    .cmd_addr      = CFG_C16_A32_8STR,
    .cmd_data      = CFG_C16_DATA_8STR,
    .cmd_addr_data = CFG_C16_A32_DATA_8STR
}
```

cmd_8dtr

```
const xsnor_commands_t cmd_8dtr [static]
```

Initial value:

```
= {
    .cmd           = CFG_C16_8DTR,
    .cmd_addr      = CFG_C16_A32_8DTR,
    .cmd_data      = CFG_C16_DATA_8DTR,
    .cmd_addr_data = CFG_C16_A32_DATA_8DTR
}
```

cmd_reset_enable_spi

```
const wspi_command_t cmd_reset_enable_spi [static]
```

Initial value:

```
= {
    .cmd      = CMD_SPI_RSTEN,
    .cfg      = CFG_C8_SPI,
    .addr     = 0,
    .alt      = 0,
    .dummy    = 0
}
```

cmd_reset_memory_spi

```
const wspi_command_t cmd_reset_memory_spi [static]
```

Initial value:

```
= {
    .cmd      = CMD_SPI_RST,
    .cfg      = CFG_C8_SPI,
    .addr     = 0,
    .alt      = 0,
    .dummy    = 0
}
```

cmd_reset_enable_8str

```
const wspi_command_t cmd_reset_enable_8str [static]
```

Initial value:

```
= {
    .cmd      = CMD_OPI_RSTEN,
    .cfg      = CFG_C16_8STR,
    .addr     = 0,
    .alt      = 0,
    .dummy    = 0
}
```

cmd_reset_memory_8str

```
const wspi_command_t cmd_reset_memory_8str [static]
```

Initial value:

```
= {  
    .cmd      = CMD_OPI_RST,  
    .cfg      = CFG_C16_8STR,  
    .addr     = 0,  
    .alt      = 0,  
    .dummy    = 0  
}
```

cmd_reset_enable_8dtr

```
const wspi_command_t cmd_reset_enable_8dtr [static]
```

Initial value:

```
= {  
    .cmd      = CMD_OPI_RSTEN,  
    .cfg      = CFG_C16_8DTR,  
    .addr     = 0,  
    .alt      = 0,  
    .dummy    = 0  
}
```

cmd_reset_memory_8dtr

```
const wspi_command_t cmd_reset_memory_8dtr [static]
```

Initial value:

```
= {  
    .cmd      = CMD_OPI_RST,  
    .cfg      = CFG_C16_8DTR,  
    .addr     = 0,  
    .alt      = 0,  
    .dummy    = 0  
}
```

mx25_manufacturer_ids

```
const uint8_t mx25_manufacturer_ids[] = {0xC2} [static]
```

mx25_memory_type_ids

```
const uint8_t mx25_memory_type_ids[] = {0x85} [static]
```

`__hal_xsnor_macronix_mx25_vmt`

```
const struct hal\_xsnor\_macronix\_mx25\_vmt __hal_xsnor_macronix_mx25_vmt
```

Initial value:

```
= {
    .dispose           = __mx25_dispose_impl,
    .init              = __mx25_init_impl,
    .read              = __mx25_read_impl,
    .program           = __mx25_program_impl,
    .start_erase_all   = __mx25_start_erase_all_impl,
    .start_erase_sector = __mx25_start_erase_sector_impl,
    .query_erase       = __mx25_query_erase_impl,
    .verify_erase      = __mx25_verify_erase_impl,
    .mmap_on           = __mx25_mmap_on_impl,
    .mmap_off          = __mx25_mmap_off_impl
}
```

VMT structure of SNOR Macronix MX25 driver class.

Note

It is public because accessed by the inlined constructor.

7.2.10.7.5 Micron N25Q Driver

Micron N25Q Driver.

7.2.10.7.5.1 Detailed Description

Micron N25Q Driver.

SNOR Micron N25Q driver.

This module implements a Micron N25Q SNOR driver class inheriting from [hal_xsnor_base_c](#) class.

Module for SNOR Micron N25Q flash devices.

Class @p `hal_xsnor_micron_n25q_c` structures

- typedef struct [hal_xsnor_micron_n25q](#) [hal_xsnor_micron_n25q_c](#)
Type of a SNOR Micron N25Q driver class.

Methods implementations of `hal_xsnor_micron_n25q_c`

- `void * __n25q_objinit_impl (void *ip, const void *vmt)`
Implementation of object creation.
- `void __n25q_dispose_impl (void *ip)`
Implementation of object finalization.
- `flash_error_t __n25q_init_impl (void *ip)`
Override of method `xsnor_device_init()`.
- `flash_error_t __n25q_read_impl (void *ip, flash_offset_t offset, size_t n, uint8_t *rp)`
Override of method `xsnor_device_read()`.
- `flash_error_t __n25q_program_impl (void *ip, flash_offset_t offset, size_t n, const uint8_t *pp)`
Override of method `xsnor_device_program()`.
- `flash_error_t __n25q_start_erase_all_impl (void *ip)`
Override of method `xsnor_device_start_erase_all()`.
- `flash_error_t __n25q_start_erase_sector_impl (void *ip, flash_sector_t sector)`
Override of method `xsnor_device_start_erase_sector()`.
- `flash_error_t __n25q_query_erase_impl (void *ip, unsigned *msec)`
Override of method `xsnor_device_query_erase()`.
- `flash_error_t __n25q_verify_erase_impl (void *ip, flash_sector_t sector)`
Override of method `xsnor_device_verify_erase()`.
- `flash_error_t __n25q_mmap_on_impl (void *ip, uint8_t **addrp)`
Override of method `xsnor_device_mmap_on()`.
- `void __n25q_mmap_off_impl (void *ip)`
Override of method `xsnor_device_mmap_off()`.

Default constructor of `hal_xsnor_micron_n25q_c`

- `static CC_FORCE_INLINE hal_xsnor_micron_n25q_c * n25qObjectInit (hal_xsnor_micron_n25q_c *self)`
Default initialization function of `hal_xsnor_micron_n25q_c`.

Device options

- `#define N25Q_OPT_DUMMY_CYCLES_MASK (15U << 0)`
Mask of the dummy cycles field.
- `#define N25Q_OPT_DUMMY_CYCLES(n)`
Number of dummy cycles.
- `#define N25Q_OPT_NO_WIDTH_SWITCH (1U << 4)`
Switch bus width on initialization.
- `#define N25Q_OPT_USE_SUBSECTORS (1U << 5)`
Use 4kB sub-sectors rather than 64kB sectors.
- `#define N25Q_OPT_NICE_WAITING (1U << 6)`
Delays insertion.

Command codes

- `#define CMD_RESET_ENABLE 0x66`
- `#define CMD_RESET_MEMORY 0x99`
- `#define CMD_READ_ID 0x9F`
- `#define CMD_MULTIPLE_IO_READ_ID 0xAF`
- `#define CMD_READ_DISCOVERY_PARAMETER 0x5A`
- `#define CMD_READ 0x03`
- `#define CMD_FAST_READ 0x0B`
- `#define CMD_WRITE_ENABLE 0x06`
- `#define CMD_WRITE_DISABLE 0x04`
- `#define CMD_READ_STATUS_REGISTER 0x05`
- `#define CMD_WRITE_STATUS_REGISTER 0x01`
- `#define CMD_READ_LOCK_REGISTER 0xE8`
- `#define CMD_WRITE_LOCK_REGISTER 0xE5`
- `#define CMD_READ_FLAG_STATUS_REGISTER 0x70`
- `#define CMD_CLEAR_FLAG_STATUS_REGISTER 0x50`
- `#define CMD_READ_NV_CONFIGURATION_REGISTER 0xB5`
- `#define CMD_WRITE_NV_CONFIGURATION_REGISTER 0xB1`
- `#define CMD_READ_V_CONF_REGISTER 0x85`
- `#define CMD_WRITE_V_CONF_REGISTER 0x81`
- `#define CMD_READ_ENHANCED_V_CONF_REGISTER 0x65`
- `#define CMD_WRITE_ENHANCED_V_CONF_REGISTER 0x61`
- `#define CMD_PAGE_PROGRAM 0x02`
- `#define CMD_SUBSECTOR_ERASE 0x20`
- `#define CMD_SECTOR_ERASE 0xD8`
- `#define CMD_BULK_ERASE 0xC7`
- `#define CMD_PROGRAM_ERASE_RESUME 0x7A`
- `#define CMD_PROGRAM_ERASE_SUSPEND 0x75`
- `#define CMD_READ_OTP_ARRAY 0x4B`
- `#define CMD_PROGRAM_OTP_ARRAY 0x42`

Flags status register bits

- `#define FLAGS_PROGRAM_ERASE 0x80U`
- `#define FLAGS_ERASE_SUSPEND 0x40U`
- `#define FLAGS_ERASE_ERROR 0x20U`
- `#define FLAGS_PROGRAM_ERROR 0x10U`
- `#define FLAGS_VPP_ERROR 0x08U`
- `#define FLAGS_PROGRAM_SUSPEND 0x04U`
- `#define FLAGS_PROTECTION_ERROR 0x02U`
- `#define FLAGS_RESERVED 0x01U`
- `#define FLAGS_ALL_ERRORS`

Data Structures

- struct `hal_xsnor_micron_n25q_vmt`
 Class `hal_xsnor_micron_n25q_c` virtual methods table.
- struct `hal_xsnor_micron_n25q`
 Structure representing a SNOR Micron N25Q driver class.
- class `hal_xsnor_micron_n25q_c`

Macros

- `#define PAGE_SIZE 256U`
- `#define PAGE_MASK (PAGE_SIZE - 1U)`

Functions

- static bool `n25q_find_id` (const uint8_t *set, size_t size, uint8_t element)
- static flash_error_t `n25q_poll_status` (hal_xsnor_micron_n25q_c *self)
- static void `n25q_activate_xip` (hal_xsnor_micron_n25q_c *self)
- static void `n25q_reset_xip` (hal_xsnor_micron_n25q_c *self)
- static void `n25q_reset_memory` (hal_xsnor_micron_n25q_c *self)
- static void `n25q_read_id` (hal_xsnor_micron_n25q_c *self, uint8_t *buf)

Variables

- static const xsnor_commands_t `cmd1l`
- static const xsnor_commands_t `cmd2l`
- static const xsnor_commands_t `cmd4l`
- static const wspi_command_t `cmd_reset_enable_1`
- static const wspi_command_t `cmd_reset_memory_1`
- static const wspi_command_t `cmd_reset_enable_2`
- static const wspi_command_t `cmd_reset_memory_2`
- static const wspi_command_t `cmd_reset_enable_4`
- static const wspi_command_t `cmd_reset_memory_4`
- static const uint8_t `n25q_manufacturer_ids` [] = {0x20}
- static const uint8_t `n25q_memory_type_ids` [] = {0xBA, 0xBB}
- const struct hal_xsnor_micron_n25q_vmt `__hal_xsnor_micron_n25q_vmt`

VMT structure of SNOR Micron N25Q driver class.

7.2.10.7.5.2 Macro Definition Documentation

N25Q_OPT_DUMMY_CYCLES_MASK

```
#define N25Q_OPT_DUMMY_CYCLES_MASK (15U << 0)
```

Mask of the dummy cycles field.

N25Q_OPT_DUMMY_CYCLES

```
#define N25Q_OPT_DUMMY_CYCLES(  
    n)
```

Value:

```
((n) << 0)
```

Number of dummy cycles.

Parameters

<i>n</i>	Number of dummy cycles (2..15)

N25Q_OPT_NO_WIDTH_SWITCH

```
#define N25Q_OPT_NO_WIDTH_SWITCH (1U << 4)
```

Switch bus width on initialization.

If N25Q_OPT_NO_WIDTH_SWITCH is specified then this is the bus mode that the device is expected to be using else this is the bus mode that the device will be switched in.

Note

This option is only valid in WSPI bus mode.

N25Q_OPT_USE_SUBSECTORS

```
#define N25Q_OPT_USE_SUBSECTORS (1U << 5)
```

Use 4kB sub-sectors rather than 64kB sectors.

N25Q_OPT_NICE_WAITING

```
#define N25Q_OPT_NICE_WAITING (1U << 6)
```

Delays insertion.

If enabled this options inserts delays into the flash waiting routines releasing some extra CPU time for threads with lower priority, this may slow down the driver a bit however.

PAGE_SIZE

```
#define PAGE_SIZE 256U
```

PAGE_MASK

```
#define PAGE_MASK (PAGE_SIZE - 1U)
```

CMD_RESET_ENABLE

```
#define CMD_RESET_ENABLE 0x66
```

CMD_RESET_MEMORY

```
#define CMD_RESET_MEMORY 0x99
```

CMD_READ_ID

```
#define CMD_READ_ID 0x9F
```

CMD_MULTIPLE_IO_READ_ID

```
#define CMD_MULTIPLE_IO_READ_ID 0xAF
```

CMD_READ_DISCOVERY_PARAMETER

```
#define CMD_READ_DISCOVERY_PARAMETER 0x5A
```

CMD_READ

```
#define CMD_READ 0x03
```

CMD_FAST_READ

```
#define CMD_FAST_READ 0x0B
```

CMD_WRITE_ENABLE

```
#define CMD_WRITE_ENABLE 0x06
```

CMD_WRITE_DISABLE

```
#define CMD_WRITE_DISABLE 0x04
```

CMD_READ_STATUS_REGISTER

```
#define CMD_READ_STATUS_REGISTER 0x05
```

CMD_WRITE_STATUS_REGISTER

```
#define CMD_WRITE_STATUS_REGISTER 0x01
```

CMD_READ_LOCK_REGISTER

```
#define CMD_READ_LOCK_REGISTER 0xE8
```

CMD_WRITE_LOCK_REGISTER

```
#define CMD_WRITE_LOCK_REGISTER 0xE5
```

CMD_READ_FLAG_STATUS_REGISTER

```
#define CMD_READ_FLAG_STATUS_REGISTER 0x70
```

CMD_CLEAR_FLAG_STATUS_REGISTER

```
#define CMD_CLEAR_FLAG_STATUS_REGISTER 0x50
```

CMD_READ_NV_CONFIGURATION_REGISTER

```
#define CMD_READ_NV_CONFIGURATION_REGISTER 0xB5
```

CMD_WRITE_NV_CONFIGURATION_REGISTER

```
#define CMD_WRITE_NV_CONFIGURATION_REGISTER 0xB1
```

CMD_READ_V_CONF_REGISTER

```
#define CMD_READ_V_CONF_REGISTER 0x85
```

CMD_WRITE_V_CONF_REGISTER

```
#define CMD_WRITE_V_CONF_REGISTER 0x81
```

CMD_READ_ENHANCED_V_CONF_REGISTER

```
#define CMD_READ_ENHANCED_V_CONF_REGISTER 0x65
```

CMD_WRITE_ENHANCED_V_CONF_REGISTER

```
#define CMD_WRITE_ENHANCED_V_CONF_REGISTER 0x61
```

CMD_PAGE_PROGRAM

```
#define CMD_PAGE_PROGRAM 0x02
```

CMD_SUBSECTOR_ERASE

```
#define CMD_SUBSECTOR_ERASE 0x20
```

CMD_SECTOR_ERASE

```
#define CMD_SECTOR_ERASE 0xD8
```

CMD_BULK_ERASE

```
#define CMD_BULK_ERASE 0xC7
```

CMD_PROGRAM_ERASE_RESUME

```
#define CMD_PROGRAM_ERASE_RESUME 0x7A
```

CMD_PROGRAM_ERASE_SUSPEND

```
#define CMD_PROGRAM_ERASE_SUSPEND 0x75
```

CMD_READ_OTP_ARRAY

```
#define CMD_READ_OTP_ARRAY 0x4B
```

CMD_PROGRAM_OTP_ARRAY

```
#define CMD_PROGRAM_OTP_ARRAY 0x42
```

FLAGS_PROGRAM_ERASE

```
#define FLAGS_PROGRAM_ERASE 0x80U
```

FLAGS_ERASE_SUSPEND

```
#define FLAGS_ERASE_SUSPEND 0x40U
```

FLAGS_ERASE_ERROR

```
#define FLAGS_ERASE_ERROR 0x20U
```

FLAGS_PROGRAM_ERROR

```
#define FLAGS_PROGRAM_ERROR 0x10U
```

FLAGS_VPP_ERROR

```
#define FLAGS_VPP_ERROR 0x08U
```

FLAGS_PROGRAM_SUSPEND

```
#define FLAGS_PROGRAM_SUSPEND 0x04U
```

FLAGS_PROTECTION_ERROR

```
#define FLAGS_PROTECTION_ERROR 0x02U
```

FLAGS_RESERVED

```
#define FLAGS_RESERVED 0x01U
```

FLAGS_ALL_ERRORS

```
#define FLAGS_ALL_ERRORS
```

Value:

```
(FLAGS_ERASE_ERROR | \
  FLAGS_PROGRAM_ERROR | \
  FLAGS_VPP_ERROR | \
  FLAGS_PROTECTION_ERROR)
```

7.2.10.7.5.3 Typedef Documentation**hal_xsnor_micron_n25q_c**

```
typedef struct hal_xsnor_micron_n25q hal_xsnor_micron_n25q_c
```

Type of a SNOR Micron N25Q driver class.

7.2.10.7.5.4 Function Documentation**__n25q_objinit_impl()**

```
void * __n25q_objinit_impl (
    void * ip,
    const void * vmt)
```

Implementation of object creation.

Note

This function is meant to be used by derived classes.

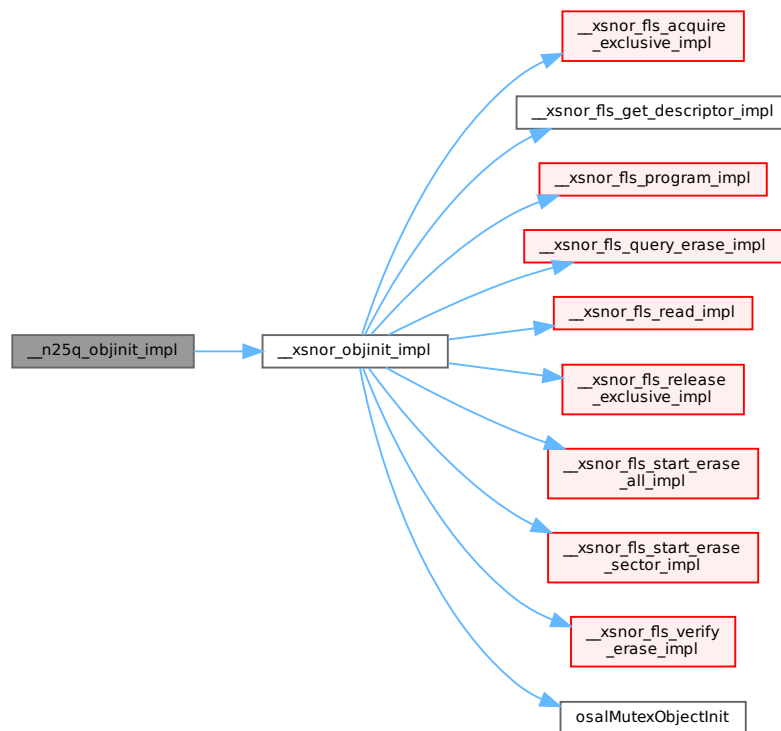
Parameters

out	<i>ip</i>	Pointer to a hal_xsnor_micron_n25q_c instance to be initialized.
in	<i>vmt</i>	VMT pointer for the new object.

Returns

A new reference to the object.

Here is the call graph for this function:

`__n25q_dispose_impl()`

```
void __n25q_dispose_impl (
    void * ip)
```

Implementation of object finalization.

Note

This function is meant to be used by derived classes.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_micron_n25q_c instance to be disposed.

Here is the call graph for this function:

**__n25q_init_impl()**

```
flash_error_t __n25q_init_impl (
    void * ip)
```

Override of method `xsnor_device_init()`.

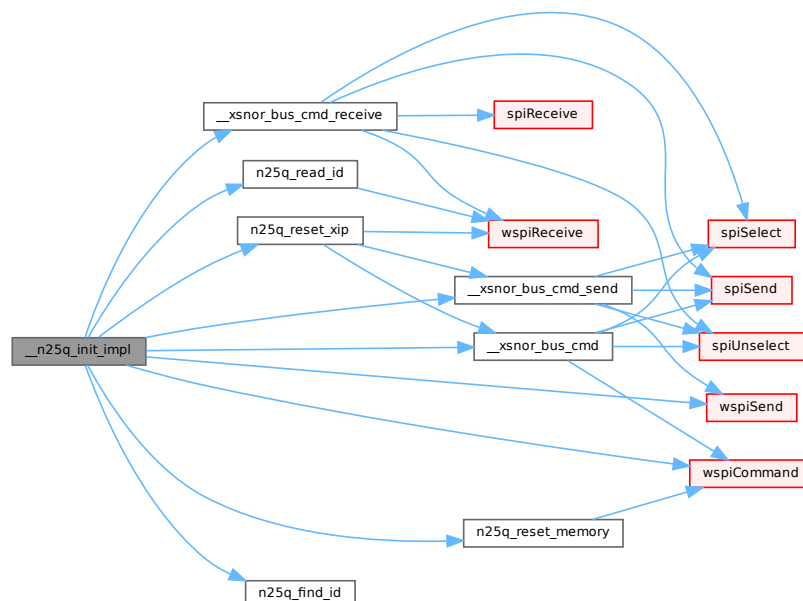
Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_micron_n25q_c instance.

Returns

An error code.

Here is the call graph for this function:



__n25q_read_impl()

```
flash_error_t __n25q_read_impl (
    void * ip,
    flash_offset_t offset,
    size_t n,
    uint8_t * rp)
```

Override of method `xsnor_device_read()`.

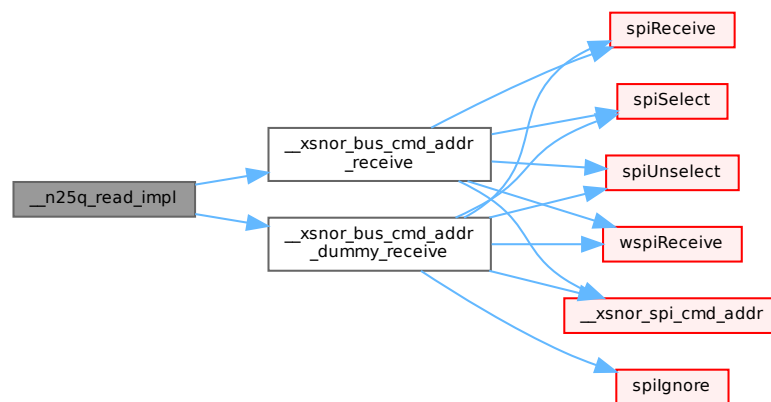
Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_micron_n25q_c</code> instance.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be read.
out	<i>rp</i>	Pointer to the data buffer.

Returns

An error code.

Here is the call graph for this function:

**__n25q_program_impl()**

```
flash_error_t __n25q_program_impl (
    void * ip,
    flash_offset_t offset,
    size_t n,
    const uint8_t * pp)
```

Override of method `xsnor_device_program()`.

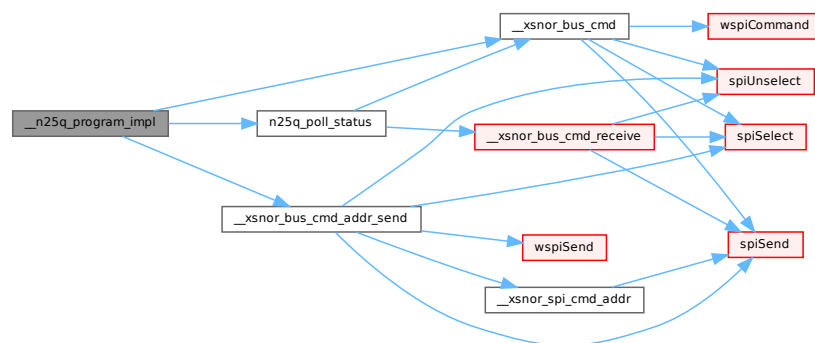
Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_micron_n25q_c instance.
in	<i>offset</i>	Flash offset.
in	<i>n</i>	Number of bytes to be programmed.
in	<i>pp</i>	Pointer to the data buffer.

Returns

An error code.

Here is the call graph for this function:

`__n25q_start_erase_all_impl()`

```
flash_error_t __n25q_start_erase_all_impl (
    void * ip)
```

Override of method `xsnor_device_start_erase_all()`.

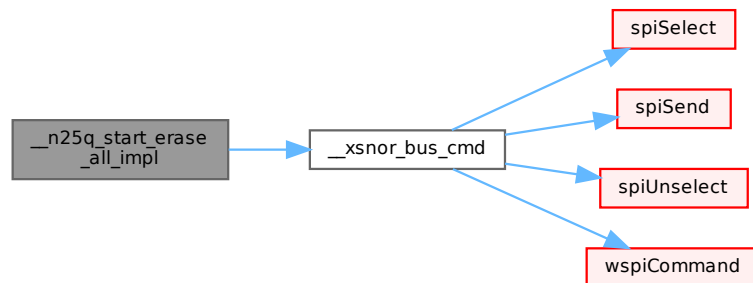
Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_micron_n25q_c instance.

Returns

An error code.

Here is the call graph for this function:

**__n25q_start_erase_sector_impl()**

```
flash_error_t __n25q_start_erase_sector_impl (
    void * ip,
    flash_sector_t sector)
```

Override of method `xsnor_device_start_erase_sector()`.

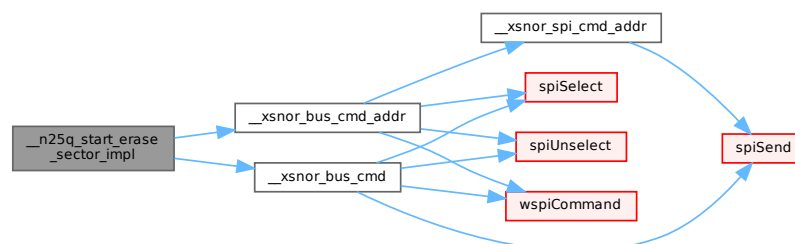
Parameters

in, out	<code>ip</code>	Pointer to a <code>hal_xsnor_micron_n25q_c</code> instance.
in	<code>sector</code>	Sector to be erased.

Returns

An error code.

Here is the call graph for this function:



__n25q_query_erase_impl()

```
flash_error_t __n25q_query_erase_impl (
    void * ip,
    unsigned * msec)
```

Override of method `xsnor_device_query_erase()`.

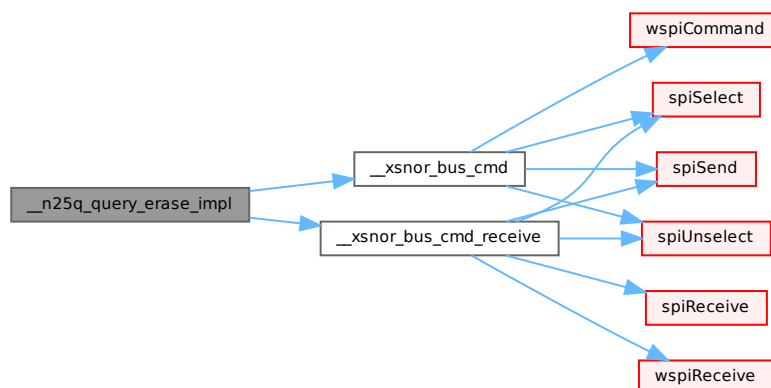
Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_micron_n25q_c</code> instance.
out	<i>msec</i>	Recommended time, in milliseconds, that should be spent before calling this function again, can be NULL

Returns

An error code.

Here is the call graph for this function:

**__n25q_verify_erase_impl()**

```
flash_error_t __n25q_verify_erase_impl (
    void * ip,
    flash_sector_t sector)
```

Override of method `xsnor_device_verify_erase()`.

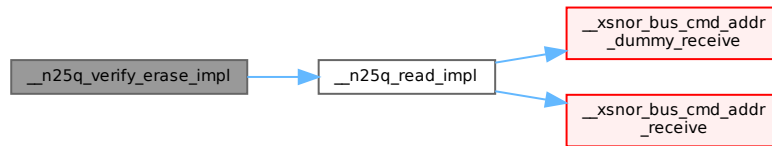
Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_micron_n25q_c</code> instance.
in	<i>sector</i>	Sector to be verified.

Returns

An error code.

Here is the call graph for this function:

**__n25q_mmap_on_impl()**

```
flash_error_t __n25q_mmap_on_impl (
    void * ip,
    uint8_t ** addrp)
```

Override of method `xsnor_device_mmap_on()`.

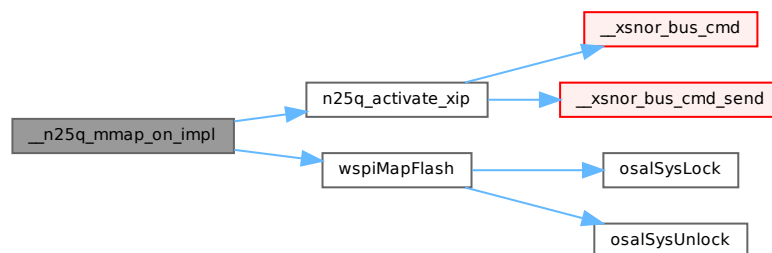
Parameters

in, out	<i>ip</i>	Pointer to a <code>hal_xsnor_micron_n25q_c</code> instance.
out	<i>addrp</i>	Pointer to the memory mapped memory or NULL

Returns

An error code.

Here is the call graph for this function:

**__n25q_mmap_off_impl()**

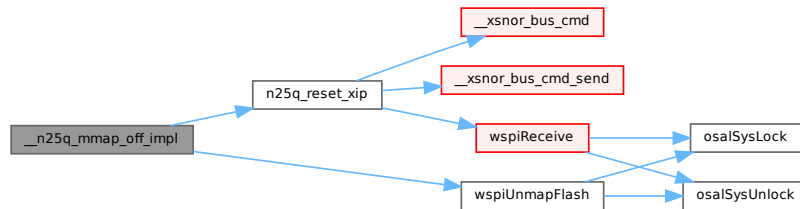
```
void __n25q_mmap_off_impl (
    void * ip)
```

Override of method `xsnor_device_mmap_off()`.

Parameters

in, out	<i>ip</i>	Pointer to a hal_xsnor_micron_n25q_c instance.

Here is the call graph for this function:

**n25qObjectInit()**

```
CC_FORCE_INLINE hal_xsnor_micron_n25q_c * n25qObjectInit (
    hal_xsnor_micron_n25q_c * self) [inline], [static]
```

Default initialization function of [hal_xsnor_micron_n25q_c](#).

Parameters

out	<i>self</i>	Pointer to a hal_xsnor_micron_n25q_c instance to be initialized.

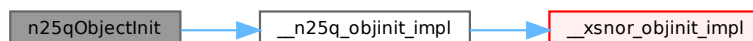
Returns

Pointer to the initialized object.

Function Class:

Object instance constructor.

Here is the call graph for this function:



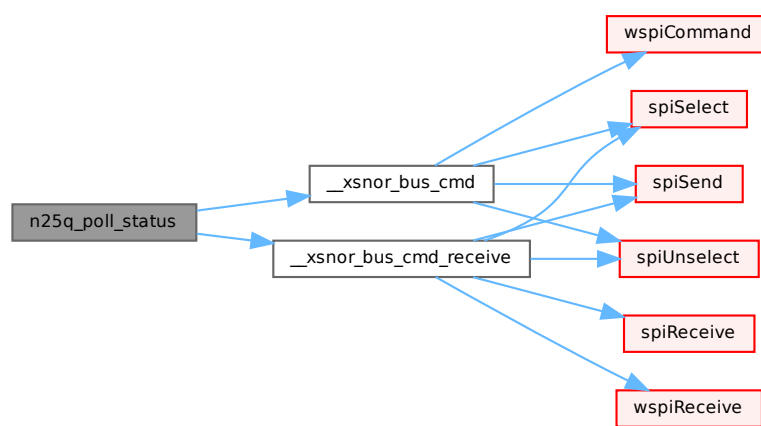
n25q_find_id()

```
bool n25q_find_id (
    const uint8_t * set,
    size_t size,
    uint8_t element) [static]
```

n25q_poll_status()

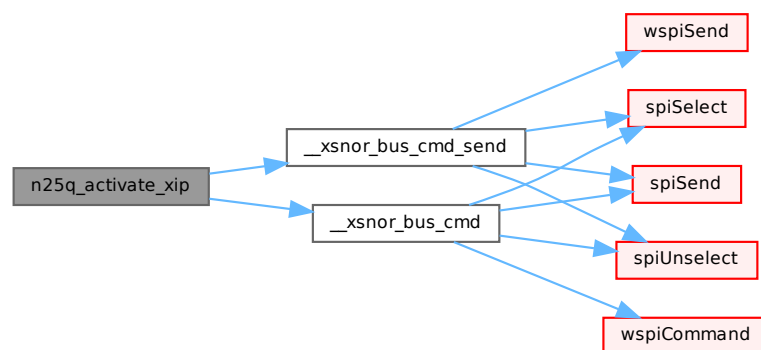
```
flash_error_t n25q_poll_status (
    hal_xsnor_micron_n25q_c * self) [static]
```

Here is the call graph for this function:

**n25q_activate_xip()**

```
void n25q_activate_xip (
    hal_xsnor_micron_n25q_c * self) [static]
```

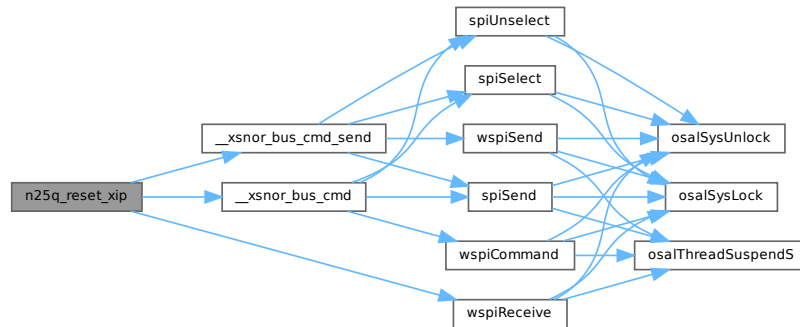
Here is the call graph for this function:



n25q_reset_xip()

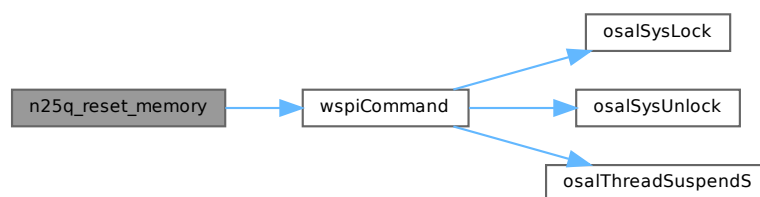
```
void n25q_reset_xip (
    hal_xsnor_micron_n25q_c * self) [static]
```

Here is the call graph for this function:

**n25q_reset_memory()**

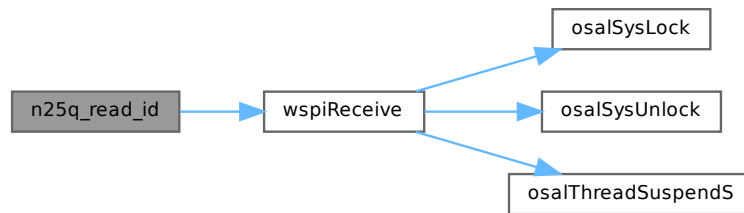
```
void n25q_reset_memory (
    hal_xsnor_micron_n25q_c * self) [static]
```

Here is the call graph for this function:

**n25q_read_id()**

```
void n25q_read_id (
    hal_xsnor_micron_n25q_c * self,
    uint8_t * buf) [static]
```


Here is the call graph for this function:



7.2.10.7.5.5 Variable Documentation

cmd1l

```
const xsnor_commands_t cmd1l [static]
```

Initial value:

```

= {
    .cmd = (WSPI_CFG_CMD_MODE_ONE_LINE |
            WSPI_CFG_ADDR_MODE_NONE |
            WSPI_CFG_ALT_MODE_NONE |
            WSPI_CFG_DATA_MODE_NONE |
            WSPI_CFG_CMD_SIZE_8 |
            WSPI_CFG_ADDR_SIZE_24),
    .cmd_addr = (WSPI_CFG_CMD_MODE_ONE_LINE |
                 WSPI_CFG_ADDR_MODE_ONE_LINE |
                 WSPI_CFG_ALT_MODE_NONE |
                 WSPI_CFG_DATA_MODE_NONE |
                 WSPI_CFG_CMD_SIZE_8 |
                 WSPI_CFG_ADDR_SIZE_24),
    .cmd_data = (WSPI_CFG_CMD_MODE_ONE_LINE |
                 WSPI_CFG_ADDR_MODE_NONE |
                 WSPI_CFG_ALT_MODE_NONE |
                 WSPI_CFG_DATA_MODE_ONE_LINE |
                 WSPI_CFG_CMD_SIZE_8 |
                 WSPI_CFG_ADDR_SIZE_24),
    .cmd_addr_data = (WSPI_CFG_CMD_MODE_ONE_LINE |
                      WSPI_CFG_ADDR_MODE_ONE_LINE |
                      WSPI_CFG_ALT_MODE_NONE |
                      WSPI_CFG_DATA_MODE_ONE_LINE |
                      WSPI_CFG_CMD_SIZE_8 |
                      WSPI_CFG_ADDR_SIZE_24)
}

```

cmd2l

```
const xsnor_commands_t cmd2l [static]
```

Initial value:

```

= {
    .cmd = (WSPI_CFG_CMD_MODE_TWO_LINES |
            WSPI_CFG_ADDR_MODE_NONE |
            WSPI_CFG_ALT_MODE_NONE |
            WSPI_CFG_DATA_MODE_NONE |
            WSPI_CFG_CMD_SIZE_8 |
            WSPI_CFG_ADDR_SIZE_24),
    .cmd_addr = (WSPI_CFG_CMD_MODE_TWO_LINES |
                 WSPI_CFG_ADDR_MODE_TWO_LINES |

```

```

        WSPI_CFG_ALT_MODE_NONE |
        WSPI_CFG_DATA_MODE_NONE |
        WSPI_CFG_CMD_SIZE_8 |
        WSPI_CFG_ADDR_SIZE_24), |
    .cmd_data = (WSPI_CFG_CMD_MODE_TWO_LINES |
        WSPI_CFG_ADDR_MODE_NONE |
        WSPI_CFG_ALT_MODE_NONE |
        WSPI_CFG_DATA_MODE_TWO_LINES |
        WSPI_CFG_CMD_SIZE_8 |
        WSPI_CFG_ADDR_SIZE_24), |
    .cmd_addr_data = (WSPI_CFG_CMD_MODE_TWO_LINES |
        WSPI_CFG_ADDR_MODE_TWO_LINES |
        WSPI_CFG_ALT_MODE_NONE |
        WSPI_CFG_DATA_MODE_TWO_LINES |
        WSPI_CFG_CMD_SIZE_8 |
        WSPI_CFG_ADDR_SIZE_24) |
}

```

cmd4l

```
const xsnor_commands_t cmd4l [static]
```

Initial value:

```

= {
    .cmd = (WSPI_CFG_CMD_MODE_FOUR_LINES |
        WSPI_CFG_ADDR_MODE_NONE |
        WSPI_CFG_ALT_MODE_NONE |
        WSPI_CFG_DATA_MODE_NONE |
        WSPI_CFG_CMD_SIZE_8 |
        WSPI_CFG_ADDR_SIZE_24), |
    .cmd_addr = (WSPI_CFG_CMD_MODE_FOUR_LINES |
        WSPI_CFG_ADDR_MODE_FOUR_LINES |
        WSPI_CFG_ALT_MODE_NONE |
        WSPI_CFG_DATA_MODE_NONE |
        WSPI_CFG_CMD_SIZE_8 |
        WSPI_CFG_ADDR_SIZE_24), |
    .cmd_data = (WSPI_CFG_CMD_MODE_FOUR_LINES |
        WSPI_CFG_ADDR_MODE_NONE |
        WSPI_CFG_ALT_MODE_NONE |
        WSPI_CFG_DATA_MODE_FOUR_LINES |
        WSPI_CFG_CMD_SIZE_8 |
        WSPI_CFG_ADDR_SIZE_24), |
    .cmd_addr_data = (WSPI_CFG_CMD_MODE_FOUR_LINES |
        WSPI_CFG_ADDR_MODE_FOUR_LINES |
        WSPI_CFG_ALT_MODE_NONE |
        WSPI_CFG_DATA_MODE_FOUR_LINES |
        WSPI_CFG_CMD_SIZE_8 |
        WSPI_CFG_ADDR_SIZE_24) |
}

```

cmd_reset_enable_1

```
const wspi_command_t cmd_reset_enable_1 [static]
```

Initial value:

```

= {
    .cmd = CMD_RESET_ENABLE,
    .cfg = WSPI_CFG_CMD_MODE_ONE_LINE,
    .addr = 0,
    .alt = 0,
    .dummy = 0
}

```

cmd_reset_memory_1

```
const wspi_command_t cmd_reset_memory_1 [static]
```

Initial value:

```

= {
    .cmd = CMD_RESET_MEMORY,
    .cfg = WSPI_CFG_CMD_MODE_ONE_LINE,
    .addr = 0,
    .alt = 0,
    .dummy = 0
}

```

cmd_reset_enable_2

```
const wspi_command_t cmd_reset_enable_2 [static]
```

Initial value:

```
= {  
    .cmd      = CMD_RESET_ENABLE,  
    .cfg      = WSPI_CFG_CMD_MODE_TWO_LINES,  
    .addr     = 0,  
    .alt      = 0,  
    .dummy    = 0  
}
```

cmd_reset_memory_2

```
const wspi_command_t cmd_reset_memory_2 [static]
```

Initial value:

```
= {  
    .cmd      = CMD_RESET_MEMORY,  
    .cfg      = WSPI_CFG_CMD_MODE_TWO_LINES,  
    .addr     = 0,  
    .alt      = 0,  
    .dummy    = 0  
}
```

cmd_reset_enable_4

```
const wspi_command_t cmd_reset_enable_4 [static]
```

Initial value:

```
= {  
    .cmd      = CMD_RESET_ENABLE,  
    .cfg      = WSPI_CFG_CMD_MODE_FOUR_LINES,  
    .addr     = 0,  
    .alt      = 0,  
    .dummy    = 0  
}
```

cmd_reset_memory_4

```
const wspi_command_t cmd_reset_memory_4 [static]
```

Initial value:

```
= {  
    .cmd      = CMD_RESET_MEMORY,  
    .cfg      = WSPI_CFG_CMD_MODE_FOUR_LINES,  
    .addr     = 0,  
    .alt      = 0,  
    .dummy    = 0  
}
```

n25q_manufacturer_ids

```
const uint8_t n25q_manufacturer_ids[] = {0x20} [static]
```

n25q_memory_type_ids

```
const uint8_t n25q_memory_type_ids[] = {0xBA, 0xBB} [static]
```

__hal_xsnor_micron_n25q_vmt

```
const struct hal_xsnor_micron_n25q_vmt __hal_xsnor_micron_n25q_vmt
```

Initial value:

```
= {
    .dispose           = __n25q_dispose_impl,
    .init              = __n25q_init_impl,
    .read              = __n25q_read_impl,
    .program           = __n25q_program_impl,
    .start_erase_all   = __n25q_start_erase_all_impl,
    .start_erase_sector = __n25q_start_erase_sector_impl,
    .query_erase       = __n25q_query_erase_impl,
    .verify_erase      = __n25q_verify_erase_impl,
    .mmap_on           = __n25q_mmap_on_impl,
    .mmap_off          = __n25q_mmap_off_impl
}
```

VMT structure of SNOR Micron N25Q driver class.

Note

It is public because accessed by the inlined constructor.

7.2.11 Interfaces and Classes

HAL Interfaces and Classes.

7.2.11.1 Detailed Description

HAL Interfaces and Classes.

Topics

- [Blocks](#)
- [Streams](#)
- [Base Object](#)

7.2.11.2 Blocks

7.2.11.2.1 Detailed Description

Topics

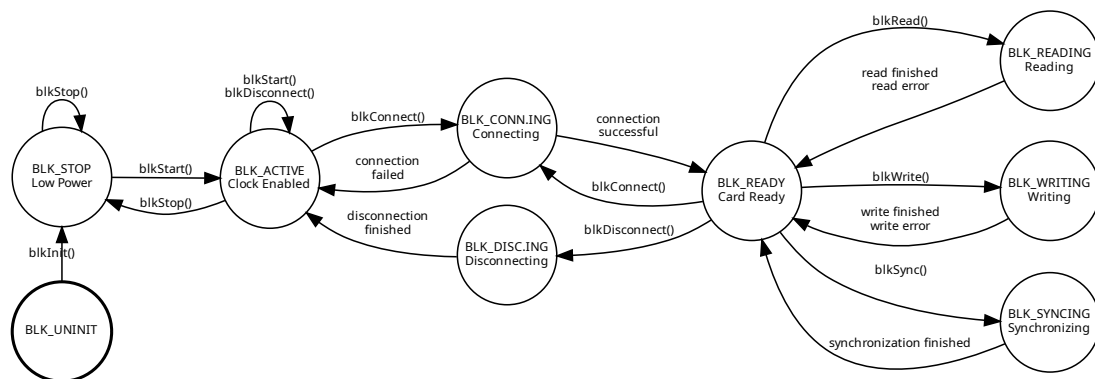
- [Abstract I/O Block Device](#)

7.2.11.2.2 Abstract I/O Block Device

7.2.11.2.2.1 Detailed Description

7.2.11.2.2.2 Driver State Machine

The drivers implementing this interface shall implement the following state machine internally. Not all the driver functionalities can be used in any moment, any transition not explicitly shown in the following diagram has to be considered an error and shall be captured by an assertion (if enabled).



This module defines an abstract interface for accessing generic block devices.

Note that no code is present, just abstract interfaces-like structures, you should look at the system as to a set of abstract C++ classes (even if written in C). This system has then advantage to make the access to block devices independent from the implementation logic.

Macro Functions (BaseBlockDevice)

- `#define blkGetDriverState(ip)`
Returns the driver state.
- `#define blkIsTransferring(ip)`
Determines if the device is transferring data.
- `#define blkIsInserted(ip)`
Returns the media insertion status.
- `#define blkIsWriteProtected(ip)`
Returns the media write protection status.
- `#define blkConnect(ip)`
Performs the initialization procedure on the block device.
- `#define blkDisconnect(ip)`
Terminates operations on the block device.
- `#define blkRead(ip, startblk, buf, n)`
Reads one or more blocks.
- `#define blkWrite(ip, startblk, buf, n)`
Writes one or more blocks.
- `#define blkSync(ip)`
Ensures write synchronization.
- `#define blkGetInfo(ip, bdip)`
Returns a media information structure.

Data Structures

- struct [BlockDeviceInfo](#)
Block device info.
- struct [BaseBlockDeviceVMT](#)
BaseBlockDevice virtual methods table.
- struct [BaseBlockDevice](#)
Base block device class.

Macros

- #define [_base_block_device_methods](#)
BaseBlockDevice specific methods.
- #define [_base_block_device_data](#)
BaseBlockDevice specific data.

Enumerations

- enum [blkstate_t](#) {
[BLK_UNINIT](#) = 0 , [BLK_STOP](#) = 1 , [BLK_ACTIVE](#) = 2 , [BLK_CONNECTING](#) = 3 ,
[BLK_DISCONNECTING](#) = 4 , [BLK_READY](#) = 5 , [BLK_READING](#) = 6 , [BLK_WRITING](#) = 7 ,
[BLK_SYNCING](#) = 8 }
Driver state machine possible states.

7.2.11.2.2.3 Macro Definition Documentation

[_base_block_device_methods](#)

```
#define _base_block_device_methods
```

Value:

```
\_base\_object\_methods
/* Removable media detection.*/
bool (*is_inserted)(void *instance);
/* Removable write protection detection.*/
bool (*is_protected)(void *instance);
/* Connection to the block device.*/
bool (*connect)(void *instance);
/* Disconnection from the block device.*/
bool (*disconnect)(void *instance);
/* Reads one or more blocks.*/
bool (*read)(void *instance, uint32_t startblk,
             uint8_t *buffer, uint32_t n);
/* Writes one or more blocks.*/
bool (*write)(void *instance, uint32_t startblk,
             const uint8_t *buffer, uint32_t n);
/* Write operations synchronization.*/
bool (*sync)(void *instance);
/* Obtains info about the media.*/
bool (*get_info)(void *instance, BlockDeviceInfo *bdip);
```

[BaseBlockDevice](#) specific methods.

_base_block_device_data

```
#define _base_block_device_data
```

Value:

```
_base_object_data          \
/* Driver state.*/         \
blkstate_t                  state;
```

[BaseBlockDevice](#) specific data.

blkGetDriverState

```
#define blkGetDriverState(
    ip)
```

Value:

```
((ip)->state)
```

Returns the driver state.

Note

Can be called in ISR context.

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class

Returns

The driver state.

Function Class:

Special function, this function has special requirements see the notes.

blkIsTransferring

```
#define blkIsTransferring(
    ip)
```

Value:

```
((ip)->state) == BLK_CONNECTING) || \
((ip)->state) == BLK_DISCONNECTING) || \
((ip)->state) == BLK_READING) || \
((ip)->state) == BLK_WRITING))
```

Determines if the device is transferring data.

Note

Can be called in ISR context.

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class

Returns

The driver state.

Return values

<i>false</i>	the device is not transferring data.
<i>true</i>	the device not transferring data.

Function Class:

Special function, this function has special requirements see the notes.

blkIsInserted

```
#define blkIsInserted(  
    ip)
```

Value:

```
((ip)->vmt->is_inserted(ip))
```

Returns the media insertion status.

Note

On some implementations this function can only be called if the device is not transferring data. The function [blkIsTransferring\(\)](#) should be used before calling this function.

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class

Returns

The media state.

Return values

<i>false</i>	media not inserted.
<i>true</i>	media inserted.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

blkIsWriteProtected

```
#define blkIsWriteProtected(  
    ip)
```

Value:

```
((ip)->vmt->is_protected(ip))
```

Returns the media write protection status.

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class

Returns

The media state.

Return values

<i>false</i>	writable media.
<i>true</i>	non writable media.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

blkConnect

```
#define blkConnect(  
    ip)
```

Value:

```
((ip)->vmt->connect(ip))
```

Performs the initialization procedure on the block device.

This function should be performed before I/O operations can be attempted on the block device and after insertion has been confirmed using [blkIsInserted\(\)](#).

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

blkDisconnect

```
#define blkDisconnect(  
    ip)
```

Value:

```
((ip)->vmt->disconnect(ip))
```

Terminates operations on the block device.

This operation safely terminates operations on the block device.

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

blkRead

```
#define blkRead(  
    ip,  
    startblk,  
    buf,  
    n)
```

Value:

```
((ip)->vmt->read(ip, startblk, buf, n))
```

Reads one or more blocks.

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class
in	<i>startblk</i>	first block to read
out	<i>buf</i>	pointer to the read buffer
in	<i>n</i>	number of blocks to read

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

blkWrite

```
#define blkWrite(
    ip,
    startblk,
    buf,
    n)
```

Value:

```
((ip)->vmt->write(ip, startblk, buf, n))
```

Writes one or more blocks.

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class
in	<i>startblk</i>	first block to write
out	<i>buf</i>	pointer to the write buffer
in	<i>n</i>	number of blocks to write

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

blkSync

```
#define blkSync(
    ip)
```

Value:

```
((ip)->vmt->sync(ip))
```

Ensures write synchronization.

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

blkGetInfo

```
#define blkGetInfo(  
    ip,  
    bdip)
```

Value:

```
((ip)->vmt->get_info(ip, bdip))
```

Returns a media information structure.

Parameters

in	<i>ip</i>	pointer to a BaseBlockDevice or derived class
out	<i>bdip</i>	pointer to a BlockDeviceInfo structure

Returns

The operation status.

Return values

<i>HAL_SUCCESS</i>	operation succeeded.
<i>HAL_FAILED</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.11.2.2.4 Enumeration Type Documentation**blkstate_t**

```
enum blkstate_t
```

Driver state machine possible states.

Enumerator

BLK_UNINIT	Not initialized.
BLK_STOP	Stopped.
BLK_ACTIVE	Interface active.
BLK_CONNECTING	Connection in progress.
BLK_DISCONNECTING	Disconnection in progress.
BLK_READY	Device ready.
BLK_READING	Read operation in progress.
BLK_WRITING	Write operation in progress.
BLK_SYNCING	Sync. operation in progress.

7.2.11.3 Streams

7.2.11.3.1 Detailed Description

Topics

- [Abstract Streams Interface](#)
- [Abstract I/O Channel Interface](#)
- [Abstract Files Interface](#)
- [Abstract Persistent Storage Interface](#)
- [Memory Streams Class](#)
- [Null Streams Class](#)
- [Buffered Serial Class](#)
- [Buffered Streams Class](#)
- [Output Formatter Utility](#)
- [Input Formatter Utility](#)

7.2.11.3.2 Abstract Streams Interface

7.2.11.3.2.1 Detailed Description

This module define an abstract interface for generic data streams. Note that no code is present, just abstract interfaces-like structures, you should look at the system as to a set of abstract C++ classes (even if written in C). This system has then advantage to make the access to data streams independent from the implementation logic. The stream interface can be used as base class for high level object types such as files, sockets, serial ports, pipes etc.

Streams return codes

- `#define STM_OK MSG_OK`
- `#define STM_TIMEOUT MSG_TIMEOUT`
- `#define STM_RESET MSG_RESET`

Macro Functions (BaseSequentialStream)

- #define `streamWrite(ip, bp, n)`
Sequential Stream write.
- #define `streamRead(ip, bp, n)`
Sequential Stream read.
- #define `streamPut(ip, b)`
Sequential Stream blocking byte write.
- #define `streamGet(ip)`
Sequential Stream blocking byte read.

Macro Functions (BaseBufferedStream)

- #define `streamUnget(ip, b)`
Buffered Stream unget.

Data Structures

- struct `BaseSequentialStreamVMT`
BaseSequentialStream virtual methods table.
- struct `BaseSequentialStream`
Base stream class.
- struct `BaseBufferedStreamVMT`
BaseBufferedStream virtual methods table.
- struct `BaseBufferedStream`
Buffered stream class.

Macros

- #define `_base_sequential_stream_methods`
BaseSequentialStream specific methods.
- #define `_base_sequential_stream_data _base_object_data`
BaseSequentialStream specific data.
- #define `_base_buffered_stream_methods`
BaseBufferedStream specific methods.
- #define `_base_buffered_stream_data _base_sequential_stream_data`
BaseBufferedStream specific data.

7.2.11.3.2.2 Macro Definition Documentation

STM_OK

```
#define STM_OK MSG_OK
```

STM_TIMEOUT

```
#define STM_TIMEOUT MSG_TIMEOUT
```

STM_RESET

```
#define STM_RESET MSG_RESET
```

_base_sequential_stream_methods

```
#define _base_sequential_stream_methods
```

Value:

```
_base_object_methods
/* Stream write buffer method.*/
size_t (*write)(void *instance, const uint8_t *bp, size_t n);
/* Stream read buffer method.*/
size_t (*read)(void *instance, uint8_t *bp, size_t n);
/* Channel put method, blocking.*/
msg_t (*put)(void *instance, uint8_t b);
/* Channel get method, blocking.*/
msg_t (*get)(void *instance);
```

[BaseSequentialStream](#) specific methods.

_base_sequential_stream_data

```
#define _base_sequential_stream_data _base_object_data
```

[BaseSequentialStream](#) specific data.

Note

It is empty because [BaseSequentialStream](#) is only an interface without implementation.

streamWrite

```
#define streamWrite(
    ip,
    bp,
    n)
```

Value:

```
((ip)->vmt->write(ip, bp, n))
```

Sequential Stream write.

The function writes data from a buffer to a stream.

Parameters

in	<i>ip</i>	pointer to a BaseSequentialStream or derived class
in	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred

Returns

The number of bytes transferred. The return value can be less than the specified number of bytes if an end-of-file condition has been met.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

streamRead

```
#define streamRead(  
    ip,  
    bp,  
    n)
```

Value:

```
((ip)->vmt->read(ip, bp, n))
```

Sequential Stream read.

The function reads data from a stream into a buffer.

Parameters

in	<i>ip</i>	pointer to a BaseSequentialStream or derived class
out	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred

Returns

The number of bytes transferred. The return value can be less than the specified number of bytes if an end-of-file condition has been met.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

streamPut

```
#define streamPut(  
    ip,  
    b)
```

Value:

```
((ip)->vmt->put(ip, b))
```

Sequential Stream blocking byte write.

This function writes a byte value to a channel. If the channel is not ready to accept data then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a BaseChannel or derived class
in	<i>b</i>	the byte value to be written to the channel

Returns

The operation status.

Return values

<i>STM_OK</i>	if the operation succeeded.
<i>STM_RESET</i>	if an end-of-file condition has been met.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

streamGet

```
#define streamGet(  
    ip)
```

Value:

```
((ip)->vmt->get(ip))
```

Sequential Stream blocking byte read.

This function reads a byte value from a channel. If the data is not available then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a BaseChannel or derived class

Returns

A byte value from the queue.

Return values

<i>STM_RESET</i>	if an end-of-file condition has been met.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

_base_buffered_stream_methods

```
#define _base_buffered_stream_methods
```

Value:

```
_base_sequential_stream_methods  
/* Channel unget method */  
msg_t (*unget)(void *instance, uint8_t b);
```

[BaseBufferedStream](#) specific methods.

`_base_buffered_stream_data`

```
#define _base_buffered_stream_data _base_sequential_stream_data
```

`BaseBufferedStream` specific data.

Note

It is empty because `BaseBufferedStream` is only an interface without implementation.

`streamUnget`

```
#define streamUnget(  
    ip,  
    b)
```

Value:

```
((ip)->vmt->unget(ip, b))
```

Buffered Stream unget.

This function replaces a byte value to a stream. `streamUnget` only guarantees a single byte can be replaced, and multiple calls without intervening calls to `streamGet` or `streamRead` may fail

Parameters

in	<i>ip</i>	pointer to a <code>BaseBufferedStream</code> or derived class
in	<i>b</i>	the byte value to be written to the channel

Postcondition

Returns

The operation status.

Return values

<i>STM_OK</i>	if the operation succeeded.
<i>STM_RESET</i>	if the operation failed

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.11.3.3 Abstract I/O Channel Interface

7.2.11.3.3.1 Detailed Description

This module defines an abstract interface for I/O channels by extending the [BaseSequentialStream](#) interface. Note that no code is present, I/O channels are just abstract interface like structures, you should look at the systems as to a set of abstract C++ classes (even if written in C). Specific device drivers can use/extend the interface and implement them.

This system has the advantage to make the access to channels independent from the implementation logic.

Default control operation codes.

- #define [CHN_CTL_INVALID](#) 0
Invalid operation code.
- #define [CHN_CTL_NOP](#) 1
Does nothing.
- #define [CHN_CTL_TX_WAIT](#) 2
Wait for TX completion.

Macro Functions (BaseChannel)

- #define [chnPutTimeout](#)(ip, b, time)
Channel blocking byte write with timeout.
- #define [chnGetTimeout](#)(ip, time)
Channel blocking byte read with timeout.
- #define [chnWrite](#)(ip, bp, n)
Channel blocking write.
- #define [chnWriteTimeout](#)(ip, bp, n, time)
Channel blocking write with timeout.
- #define [chnRead](#)(ip, bp, n)
Channel blocking read.
- #define [chnReadTimeout](#)(ip, bp, n, time)
Channel blocking read with timeout.
- #define [chnControl](#)(ip, operation, arg)
Control operation on a channel.

I/O status flags added to the event listener

- #define [CHN_NO_ERROR](#) ([eventflags_t](#))0
No pending conditions.
- #define [CHN_CONNECTED](#) ([eventflags_t](#))1
Connection happened.
- #define [CHN_DISCONNECTED](#) ([eventflags_t](#))2
Disconnection happened.
- #define [CHN_INPUT_AVAILABLE](#) ([eventflags_t](#))4
Data available in the input queue.
- #define [CHN_OUTPUT_EMPTY](#) ([eventflags_t](#))8
Output queue empty.
- #define [CHN_TRANSMISSION_END](#) ([eventflags_t](#))16

- *Transmission end.*
- #define `CHN_PARITY_ERROR` (`eventflags_t`)32
- *Parity error.*
- #define `CHN_FRAMING_ERROR` (`eventflags_t`)64
- *Framing error.*
- #define `CHN_NOISE_ERROR` (`eventflags_t`)128
- *Line noise error.*
- #define `CHN_OVERRUN_ERROR` (`eventflags_t`)256
- *Overflow error.*
- #define `CHN_IDLE_DETECTED` (`eventflags_t`)512
- *RX line idle.*
- #define `CHN_BREAK_DETECTED` (`eventflags_t`)1024
- *LIN Break.*
- #define `CHN_BUFFER_FULL_ERROR` (`eventflags_t`)2048

Macro Functions (BaseAsynchronousChannel)

- #define `chnGetEventSource`(ip)
- *Returns the I/O condition event source.*
- #define `chnAddFlagsI`(ip, flags)
- *Adds status flags to the listeners's flags mask.*

Data Structures

- struct `BaseChannelVMT`
- *BaseChannel virtual methods table.*
- struct `BaseChannel`
- *Base channel class.*
- struct `BaseAsynchronousChannelVMT`
- *BaseAsynchronousChannel virtual methods table.*
- struct `BaseAsynchronousChannel`
- *Base asynchronous channel class.*

Macros

- #define `_base_channel_methods`
- *BaseChannel specific methods.*
- #define `_base_channel_data _base_sequential_stream_data`
- *BaseChannel specific data.*
- #define `_base_asynchronous_channel_methods _base_channel_methods \`
- *BaseAsynchronousChannel specific methods.*
- #define `_base_asynchronous_channel_data`
- *BaseAsynchronousChannel specific data.*

7.2.11.3.3.2 Macro Definition Documentation

CHN_CTL_INVALID

```
#define CHN_CTL_INVALID 0
```

Invalid operation code.

CHN_CTL_NOP

```
#define CHN_CTL_NOP 1
```

Does nothing.

CHN_CTL_TX_WAIT

```
#define CHN_CTL_TX_WAIT 2
```

Wait for TX completion.

_base_channel_methods

```
#define _base_channel_methods
```

Value:

```
_base_sequential_stream_methods
/* Channel put method with timeout specification.*/
msg_t (*putt)(void *instance, uint8_t b, sysinterval_t time);
/* Channel get method with timeout specification.*/
msg_t (*gett)(void *instance, sysinterval_t time);
/* Channel write method with timeout specification.*/
size_t (*writet)(void *instance, const uint8_t *bp,
                 size_t n, sysinterval_t time);
/* Channel read method with timeout specification.*/
size_t (*readt)(void *instance, uint8_t *bp, size_t n,
                sysinterval_t time);
/* Channel control method.*/
msg_t (*ctl)(void *instance, unsigned int operation, void *arg);
```

[BaseChannel](#) specific methods.

_base_channel_data

```
#define _base_channel_data _base_sequential_stream_data
```

[BaseChannel](#) specific data.

Note

It is empty because [BaseChannel](#) is only an interface without implementation.

chnPutTimeout

```
#define chnPutTimeout(
    ip,
    b,
    time)
```

Value:

```
((ip)->vmt->putt(ip, b, time))
```

Channel blocking byte write with timeout.

This function writes a byte value to a channel. If the channel is not ready to accept data then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a BaseChannel or derived class
in	<i>b</i>	the byte value to be written to the channel
in	<i>time</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

Return values

<i>STM_OK</i>	if the operation succeeded.
<i>STM_TIMEOUT</i>	if the specified time expired.
<i>STM_RESET</i>	if the channel associated queue (if any) was reset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

chnGetTimeout

```
#define chnGetTimeout(
    ip,
    time)
```

Value:

```
((ip)->vmt->gett(ip, time))
```

Channel blocking byte read with timeout.

This function reads a byte value from a channel. If the data is not available then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a BaseChannel or derived class
in	<i>time</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

A byte value from the queue.

Return values

<code>STM_TIMEOUT</code>	if the specified time expired.
<code>STM_RESET</code>	if the channel associated queue (if any) has been reset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

chnWrite

```
#define chnWrite(  
    ip,  
    bp,  
    n)
```

Value:

```
streamWrite(ip, bp, n)
```

Channel blocking write.

The function writes data from a buffer to a channel. If the channel is not ready to accept data then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a BaseChannel or derived class
out	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred

Returns

The number of bytes transferred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

chnWriteTimeout

```
#define chnWriteTimeout(  
    ip,  
    bp,  
    n,  
    time)
```

Value:

```
((ip)->vmt->writet(ip, bp, n, time))
```

Channel blocking write with timeout.

The function writes data from a buffer to a channel. If the channel is not ready to accept data then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a BaseChannel or derived class
out	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred
in	<i>time</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The number of bytes transferred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

chnRead

```
#define chnRead(
    ip,
    bp,
    n)
```

Value:

```
streamRead(ip, bp, n)
```

Channel blocking read.

The function reads data from a channel into a buffer. If the data is not available then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a BaseChannel or derived class
in	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred

Returns

The number of bytes transferred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

chnReadTimeout

```
#define chnReadTimeout(  
    ip,  
    bp,  
    n,  
    time)
```

Value:

```
((ip)->vmt->readt(ip, bp, n, time))
```

Channel blocking read with timeout.

The function reads data from a channel into a buffer. If the data is not available then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a BaseChannel or derived class
in	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred
in	<i>time</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The number of bytes transferred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

chnControl

```
#define chnControl(
    ip,
    operation,
    arg)
```

Value:

```
((ip)->vmt->ctl(ip, operation, arg))
```

Control operation on a channel.

Parameters

in	<i>ip</i>	pointer to a BaseChannel or derived class
in	<i>operation</i>	control operation code
in, out	<i>arg</i>	operation argument

Returns

The control operation status.

Return values

<i>MSG_OK</i>	in case of success.
<i>MSG_TIMEOUT</i>	in case of operation timeout.
<i>MSG_RESET</i>	in case of operation reset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

CHN_NO_ERROR

```
#define CHN_NO_ERROR (eventflags_t)0
```

No pending conditions.

CHN_CONNECTED

```
#define CHN_CONNECTED (eventflags_t)1
```

Connection happened.

CHN_DISCONNECTED

```
#define CHN_DISCONNECTED (eventflags_t)2
```

Disconnection happened.

CHN_INPUT_AVAILABLE

```
#define CHN_INPUT_AVAILABLE (eventflags_t)4
```

Data available in the input queue.

CHN_OUTPUT_EMPTY

```
#define CHN_OUTPUT_EMPTY (eventflags_t)8
```

Output queue empty.

CHN_TRANSMISSION_END

```
#define CHN_TRANSMISSION_END (eventflags_t)16
```

Transmission end.

CHN_PARITY_ERROR

```
#define CHN_PARITY_ERROR (eventflags_t)32
```

Parity error.

CHN_FRAMING_ERROR

```
#define CHN_FRAMING_ERROR (eventflags_t)64
```

Framing error.

CHN_NOISE_ERROR

```
#define CHN_NOISE_ERROR (eventflags_t)128
```

Line noise error.

CHN_OVERRUN_ERROR

```
#define CHN_OVERRUN_ERROR (eventflags_t)256
```

Overflow error.

CHN_IDLE_DETECTED

```
#define CHN_IDLE_DETECTED (eventflags_t)512
```

RX line idle.

CHN_BREAK_DETECTED

```
#define CHN_BREAK_DETECTED (eventflags_t)1024
```

LIN Break.

RX buffer full.

CHN_BUFFER_FULL_ERROR

```
#define CHN_BUFFER_FULL_ERROR (eventflags_t)2048
```

_base_asynchronous_channel_methods

```
#define _base_asynchronous_channel_methods _base_channel_methods \
```

[BaseAsynchronousChannel](#) specific methods.

_base_asynchronous_channel_data

```
#define _base_asynchronous_channel_data
```

Value:

```
_base_channel_data          \
/* I/O condition event source.*/ \
event_source_t              event;
```

[BaseAsynchronousChannel](#) specific data.

chnGetEventSource

```
#define chnGetEventSource(
    ip)
```

Value:

```
(&((ip)->event))
```

Returns the I/O condition event source.

The event source is broadcasted when an I/O condition happens.

Parameters

in	<i>ip</i>	pointer to a BaseAsynchronousChannel or derived class

Returns

A pointer to an `EventSource` object.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

chnAddFlagsI

```
#define chnAddFlagsI(
    ip,
    flags)
```

Value:

```
{
    osalEventBroadcastFlagsI(&(ip)->event, flags);
}
```

Adds status flags to the listeners's flags mask.

This function is usually called from the I/O ISRs in order to notify I/O conditions such as data events, errors, signal changes etc.

Parameters

in	<i>ip</i>	pointer to a BaseAsynchronousChannel or derived class
in	<i>flags</i>	condition flags to be added to the listener flags mask

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.11.3.4 Abstract Files Interface**7.2.11.3.4.1 Detailed Description**

This module define an abstract interface for generic data files by extending the [BaseSequentialStream](#) interface. Note that no code is present, data files are just abstract interface-like structures, you should look at the systems as to a set of abstract C++ classes (even if written in C). This system has the advantage to make the access to streams independent from the implementation logic.

The data files interface can be used as base class for high level object types such as an API for a File System implementation.

Files return codes

- #define `FILE_OK STM_OK`
No error return code.
- #define `FILE_ERROR STM_TIMEOUT`
Error code from the file stream methods.
- #define `FILE_EOF STM_RESET`
End-of-file condition for file get/put methods.

Macro Functions (FileStream)

- #define `fileStreamWrite(ip, bp, n)`
File stream write.
- #define `fileStreamRead(ip, bp, n)`
File stream read.
- #define `fileStreamPut(ip, b)`
File stream blocking byte write.
- #define `fileStreamGet(ip)`
File stream blocking byte read.
- #define `fileStreamClose(ip)`
File Stream close.
- #define `fileStreamGetError(ip)`
Returns an implementation dependent error code.
- #define `fileStreamGetSize(ip, offset)`
Returns the current file size.
- #define `fileStreamGetPosition(ip, offset)`
Returns the current file pointer position.
- #define `fileStreamSetPosition(ip, offset)`
Moves the file current pointer to an absolute position.

Data Structures

- struct `FileStreamVMT`
FileStream virtual methods table.
- struct `FileStream`
Base file stream class.

Macros

- #define `_file_stream_methods`
FileStream specific methods.
- #define `_file_stream_data _base_sequential_stream_data`
FileStream specific data.

Typedefs

- typedef uint32_t `fileoffset_t`
File offset type.

7.2.11.3.4.2 Macro Definition Documentation

FILE_OK

```
#define FILE_OK STM_OK
```

No error return code.

FILE_ERROR

```
#define FILE_ERROR STM_TIMEOUT
```

Error code from the file stream methods.

FILE_EOF

```
#define FILE_EOF STM_RESET
```

End-of-file condition for file get/put methods.

_file_stream_methods

```
#define _file_stream_methods
```

Value:

```
_base_sequential_stream_methods
/* File close method.*/
msg_t (*close)(void *instance);
/* Get last error code method.*/
msg_t (*geterror)(void *instance);
/* File get size method.*/
msg_t (*getsize)(void *instance, fileoffset_t *offset);
/* File get current position method.*/
msg_t (*getposition)(void *instance, fileoffset_t *offset);
/* File set current position method.*/
msg_t (*setposition)(void *instance, fileoffset_t offset);
```

[FileStream](#) specific methods.

_file_stream_data

```
#define _file_stream_data _base_sequential_stream_data
```

[FileStream](#) specific data.

Note

It is empty because [FileStream](#) is only an interface without implementation.

fileStreamWrite

```
#define fileStreamWrite(
    ip,
    bp,
    n)
```

Value:

```
streamWrite(ip, bp, n)
```

File stream write.

The function writes data from a buffer to a file stream.

Parameters

in	<i>ip</i>	pointer to a FileStream or derived class
in	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred

Returns

The number of bytes transferred. The return value can be less than the specified number of bytes if an end-of-file condition has been met.

Return values

<i>FILE_ERROR</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

fileStreamRead

```
#define fileStreamRead(  
    ip,  
    bp,  
    n)
```

Value:

[streamRead](#)(*ip*, *bp*, *n*)

File stream read.

The function reads data from a file stream into a buffer.

Parameters

in	<i>ip</i>	pointer to a FileStream or derived class
out	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred

Returns

The number of bytes transferred. The return value can be less than the specified number of bytes if an end-of-file condition has been met.

Return values

<i>FILE_ERROR</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

fileStreamPut

```
#define fileStreamPut(  
    ip,  
    b)
```

Value:

`streamPut(ip, b)`

File stream blocking byte write.

This function writes a byte value to a channel. If the channel is not ready to accept data then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a FileStream or derived class
in	<i>b</i>	the byte value to be written to the channel

Returns

The operation status.

Return values

<i>FILE_OK</i>	if the operation succeeded.
<i>FILE_ERROR</i>	operation failed.
<i>FILE_EOF</i>	if an end-of-file condition has been met.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

fileStreamGet

```
#define fileStreamGet(  
    ip)
```

Value:

`streamGet(ip)`

File stream blocking byte read.

This function reads a byte value from a channel. If the data is not available then the calling thread is suspended.

Parameters

in	<i>ip</i>	pointer to a FileStream or derived class

Returns

A byte value from the queue.

Return values

<i>FILE_ERROR</i>	operation failed.
<i>FILE_EOF</i>	if an end-of-file condition has been met.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

fileStreamClose

```
#define fileStreamClose(  
    ip)
```

Value:

```
((ip)->vmt->close(ip))
```

File Stream close.

The function closes a file stream.

Parameters

in	<i>ip</i>	pointer to a FileStream or derived class

Returns

The operation status.

Return values

<i>FILE_OK</i>	no error.
<i>FILE_ERROR</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

fileStreamGetError

```
#define fileStreamGetError(  
    ip)
```

Value:

```
((ip)->vmt->geterror(ip))
```

Returns an implementation dependent error code.

Precondition

The previously called function must have returned `FILE_ERROR`.

Parameters

in	<i>ip</i>	pointer to a FileStream or derived class

Returns

Implementation dependent error code.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

fileStreamGetSize

```
#define fileStreamGetSize(  
    ip,  
    offset)
```

Value:

```
((ip)->vmt->getsize(ip, offset))
```

Returns the current file size.

Parameters

in	<i>ip</i>	pointer to a FileStream or derived class
out	<i>offset</i>	current size of the file

Returns

The file size.

Return values

<i>FILE_ERROR</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

fileStreamGetPosition

```
#define fileStreamGetPosition(  
    ip,  
    offset)
```

Value:

```
((ip)->vmt->getposition(ip, offset))
```

Returns the current file pointer position.

Parameters

in	<i>ip</i>	pointer to a FileStream or derived class
out	<i>offset</i>	current position in the file

Returns

The current position inside the file.

Return values

<i>FILE_ERROR</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

fileStreamSetPosition

```
#define fileStreamSetPosition(  
    ip,  
    offset)
```

Value:

```
((ip)->vmt->setposition(ip, offset))
```

Moves the file current pointer to an absolute position.

Parameters

in	<i>ip</i>	pointer to a FileStream or derived class
in	<i>offset</i>	new absolute position

Returns

The operation status.

Return values

<i>FILE_OK</i>	no error.
<i>FILE_ERROR</i>	operation failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.11.3.4.3 Typedef Documentation

fileoffset_t

```
typedef uint32_t fileoffset_t
```

File offset type.

7.2.11.3.5 Abstract Persistent Storage Interface

7.2.11.3.5.1 Detailed Description

This module define an abstract interface for generic persistent storage. Such storage has a fixed size and can be read and written.

Macro Functions (BasePersistentStorage)

- #define `getBasePersistentStorage(ip)`
Instance getter.
- #define `psGetStorageSize(ip)`
Get storage size.
- #define `psRead(ip, offset, n, rp)`
Read operation.
- #define `psWrite(ip, offset, n, wp)`
Write operation.

Data Structures

- struct `BasePersistentStorageVMT`
BasePersistentStorage virtual methods table.
- struct `BasePersistentStorage`
Base persistent storage class.

Macros

- #define `_base_pers_storage_methods_alone`
BasePersistentStorage specific methods.
- #define `_base_pers_storage_methods`
BasePersistentStorage specific methods with inherited ones.
- #define `_base_persistent_storage_data _base_object_data`
BasePersistentStorage specific data.

Typedefs

- typedef uint32_t `ps_offset_t`
Type of a persistent storage offset.

Enumerations

- enum `ps_error_t` {
`PS_NO_ERROR` = 0 , `PS_ERROR_READ` = 2 , `PS_ERROR_WRITE` = 3 , `PS_ERROR_VERIFY` = 5 ,
`PS_ERROR_HW_FAILURE` = 6 }

Type of a persistent storage error code.

7.2.11.3.5.2 Macro Definition Documentation

`_base_pers_storage_methods_alone`

```
#define _base_pers_storage_methods_alone
```

Value:

```
/* Storage size.*/
size_t (*getsize)(void *instance);
/* Read operation.*/
ps_error_t (*read)(void *instance, ps_offset_t offset,
                    size_t n, uint8_t *rp);
/* Write operation.*/
ps_error_t (*write)(void *instance, ps_offset_t offset,
                    size_t n, const uint8_t *wp);
```

`BasePersistentStorage` specific methods.

`_base_pers_storage_methods`

```
#define _base_pers_storage_methods
```

Value:

```
_base_object_methods
_base_pers_storage_methods_alone
```

`BasePersistentStorage` specific methods with inherited ones.

`_base_persistent_storage_data`

```
#define _base_persistent_storage_data _base_object_data
```

`BasePersistentStorage` specific data.

`getBasePersistentStorage`

```
#define getBasePersistentStorage(  
    ip)
```

Value:

```
((BasePersistentStorage *) &(ip)->vmt)
```

Instance getter.

This special method is used to get the instance of this class object from a derived class.

`psGetStorageSize`

```
#define psGetStorageSize(  
    ip)
```

Value:

```
(ip)->vmt->getsize(ip)
```

Get storage size.

Parameters

in	<i>ip</i>	pointer to a BasePersistentStorage or derived class

Returns

The storage size in bytes.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

psRead

```
#define psRead(
    ip,
    offset,
    n,
    rp)
```

Value:

```
(ip)->vmt->read(ip, offset, n, rp)
```

Read operation.

Parameters

in	<i>ip</i>	pointer to a BasePersistentStorage or derived class
in	<i>offset</i>	persistent storage offset
in	<i>n</i>	number of bytes to be read
out	<i>rp</i>	pointer to the data buffer

Returns

An error code.

Return values

<i>PS_NO_ERROR</i>	if there is no erase operation in progress.
<i>PS_ERROR_READ</i>	if the read operation failed.
<i>PS_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

psWrite

```
#define psWrite(
    ip,
    offset,
    n,
    wp)
```

Value:

```
(ip)->vmt->write(ip, offset, n, wp)
```

Write operation.

Parameters

in	<i>ip</i>	pointer to a BasePersistentStorage or derived class
in	<i>offset</i>	persistent storage offset
in	<i>n</i>	number of bytes to be written
in	<i>wp</i>	pointer to the data buffer

Returns

An error code.

Return values

<i>PS_NO_ERROR</i>	if there is no erase operation in progress.
<i>PS_ERROR_WRITE</i>	if the write operation failed.
<i>PS_ERROR_HW_FAILURE</i>	if access to the memory failed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.11.3.5.3 Typedef Documentation**ps_offset_t**

```
typedef uint32_t ps_offset_t
```

Type of a persistent storage offset.

7.2.11.3.5.4 Enumeration Type Documentation**ps_error_t**

```
enum ps_error_t
```

Type of a persistent storage error code.

Note

Code values are kept equal to the equivalent codes in the flash interface, this is intentional.

Enumerator

<i>PS_NO_ERROR</i>	
<i>PS_ERROR_READ</i>	
<i>PS_ERROR_WRITE</i>	
<i>PS_ERROR_VERIFY</i>	
<i>PS_ERROR_HW_FAILURE</i>	

7.2.11.3.6 Memory Streams Class

7.2.11.3.6.1 Detailed Description

Memory buffers handled as streams.

Data Structures

- struct [MemStreamVMT](#)
MemStream virtual methods table.
- struct [MemoryStream](#)
Memory stream object.

Macros

- `#define _memory_stream_data`
MemStream specific data.

Functions

- static `size_t _writes` (void *ip, const uint8_t *bp, size_t n)
- static `size_t _reads` (void *ip, uint8_t *bp, size_t n)
- static `msg_t _put` (void *ip, uint8_t b)
- static `msg_t _get` (void *ip)
- static `msg_t _unget` (void *ip, uint8_t b)
- void `msObjectInit` ([MemoryStream](#) *msp, uint8_t *buffer, size_t size, size_t eos)
Memory stream object initialization.

Variables

- static const struct [MemStreamVMT](#) vmt = {(size_t)0, `_writes`, `_reads`, `_put`, `_get`, `_unget`}

7.2.11.3.6.2 Macro Definition Documentation

`_memory_stream_data`

```
#define _memory_stream_data
```

Value:

```
_base_buffered_stream_data
/* Pointer to the stream buffer.*/
uint8_t      *buffer;
/* Size of the stream.*/
size_t       size;
/* Current end of stream.*/
size_t       eos;
/* Current read offset.*/
size_t       offset;
```

```

/
/
/
/
/
/
/

```

MemStream specific data.

7.2.11.3.6.3 Function Documentation

_writes()

```
size_t _writes (  
    void * ip,  
    const uint8_t * bp,  
    size_t n) [static]
```

_reads()

```
size_t _reads (  
    void * ip,  
    uint8_t * bp,  
    size_t n) [static]
```

_put()

```
msg_t _put (  
    void * ip,  
    uint8_t b) [static]
```

_get()

```
msg_t _get (  
    void * ip) [static]
```

_unget()

```
msg_t _unget (  
    void * ip,  
    uint8_t b) [static]
```

msObjectInit()

```
void msObjectInit (  
    MemoryStream * msp,  
    uint8_t * buffer,  
    size_t size,  
    size_t eos)
```

Memory stream object initialization.

Parameters

out	<i>msp</i>	pointer to the MemoryStream object to be initialized
in	<i>buffer</i>	pointer to the memory buffer for the memory stream
in	<i>size</i>	total size of the memory stream buffer
in	<i>eos</i>	initial End Of Stream offset. Normally you need to put this to zero for RAM buffers or equal to <i>size</i> for ROM streams.

7.2.11.3.6.4 Variable Documentation

vmt

```
const struct MemStreamVMT vmt = {(size_t)0, _writes, _reads, _put, _get, _unget} [static]
```

7.2.11.3.7 Null Streams Class

7.2.11.3.7.1 Detailed Description

A null streams.

Data Structures

- struct [NullStreamVMT](#)
NullStream virtual methods table.
- struct [NullStream](#)
Null stream object.

Macros

- `#define _null_stream_data _base_buffered_stream_data`
NullStream specific data.

Functions

- static `size_t writes` (void *ip, const uint8_t *bp, size_t n)
- static `size_t reads` (void *ip, uint8_t *bp, size_t n)
- static `msg_t put` (void *ip, uint8_t b)
- static `msg_t get` (void *ip)
- static `msg_t unget` (void *ip, uint8_t b)
- void `nullObjectInit` ([NullStream](#) *nsp)
Null stream object initialization.

Variables

- static const struct [NullStreamVMT](#) vmt = {(size_t)0, [writes](#), [reads](#), [put](#), [get](#), [unget](#)}

7.2.11.3.7.2 Macro Definition Documentation

`_null_stream_data`

```
#define _null_stream_data _base_buffered_stream_data
```

[NullStream](#) specific data.

7.2.11.3.7.3 Function Documentation

writes()

```
size_t writes (  
    void * ip,  
    const uint8_t * bp,  
    size_t n) [static]
```

reads()

```
size_t reads (  
    void * ip,  
    uint8_t * bp,  
    size_t n) [static]
```

put()

```
msg_t put (  
    void * ip,  
    uint8_t b) [static]
```

get()

```
msg_t get (  
    void * ip) [static]
```

unget()

```
msg_t unget (  
    void * ip,  
    uint8_t b) [static]
```

nullObjectInit()

```
void nullObjectInit (  
    NullStream * nsp)
```

Null stream object initialization.

Parameters

out	<i>nsp</i>	pointer to the NullStream object to be initialized

7.2.11.3.7.4 Variable Documentation

vmt

```
const struct NullStreamVMT vmt = {(size_t)0, writes, reads, put, get, unget} [static]
```

7.2.11.3.8 Buffered Serial Class

7.2.11.3.8.1 Detailed Description

Methods implementations

- static [CC_FORCE_INLINE](#) void [__buffered_serial_objinit_impl](#) (void *ip, const void *vmt, uint8_t *ib, size_t ibsize, [qnotify_t](#) inotify, void *iarg, uint8_t *ob, size_t obsize, [qnotify_t](#) onotify, void *oarg)
Object initialization implementation.
- static [CC_FORCE_INLINE](#) size_t [__buffered_serial_write_impl](#) (void *ip, const uint8_t *bp, size_t n)
- static [CC_FORCE_INLINE](#) size_t [__buffered_serial_read_impl](#) (void *ip, uint8_t *bp, size_t n)
- static [CC_FORCE_INLINE](#) msg_t [__buffered_serial_put_impl](#) (void *ip, uint8_t b)
- static [CC_FORCE_INLINE](#) msg_t [__buffered_serial_get_impl](#) (void *ip)
- static [CC_FORCE_INLINE](#) msg_t [__buffered_serial_put_timeout_impl](#) (void *ip, uint8_t b, [sysinterval_t](#) timeout)
- static [CC_FORCE_INLINE](#) msg_t [__buffered_serial_get_timeout_impl](#) (void *ip, [sysinterval_t](#) timeout)
- static [CC_FORCE_INLINE](#) size_t [__buffered_serial_write_timeout_impl](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static [CC_FORCE_INLINE](#) size_t [__buffered_serial_read_timeout_impl](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static [CC_FORCE_INLINE](#) msg_t [__buffered_serial_ctl_impl](#) (void *ip, unsigned int operation, void *arg)

Data Structures

- struct [BufferedSerialVMT](#)
[BufferedSerial](#) virtual methods table.
- struct [hal_buffered_serial](#)
Buffered serial channel class.

Macros

- [#define](#) [__buffered_serial_methods_base_asynchronous_channel_methods](#)
[BufferedSerial](#) specific methods.
- [#define](#) [__buffered_serial_data](#)
[BufferedSerial](#) specific data.

Typedefs

- typedef struct [hal_buffered_serial](#) [BufferedSerial](#)
Structure representing a buffered serial class.

Enumerations

- enum `bsstate_t` { `BS_UNINIT` = 0 , `BS_STOP` = 1 , `BS_READY` = 2 }
- BufferedSerial* state machine states.

Functions

- void `bsIncomingData` (`BufferedSerial` *bsp, uint8_t b)
Handles incoming data.
- `msg_t` `bsRequestData` (`BufferedSerial` *bsp)
Handles outgoing data.

7.2.11.3.8.2 Macro Definition Documentation

`__buffered_serial_methods`

```
#define __buffered_serial_methods _base_asynchronous_channel_methods
```

BufferedSerial specific methods.

`__buffered_serial_data`

```
#define __buffered_serial_data
```

Value:

```
_base_asynchronous_channel_data
/* Driver state.*/
bsstate_t                state;
/* Input queue.*/
input_queue_t            iqueue;
/* Output queue.*/
output_queue_t           oqueue;
```

```

/
/
/
/
/

```

BufferedSerial specific data.

7.2.11.3.8.3 Typedef Documentation

`BufferedSerial`

```
typedef struct hal_buffered_serial BufferedSerial
```

Structure representing a buffered serial class.

7.2.11.3.8.4 Enumeration Type Documentation

`bsstate_t`

```
enum bsstate_t
```

BufferedSerial state machine states.

Enumerator

BS_UNINIT	Not initialized.
BS_STOP	Stopped.
BS_READY	Ready.

7.2.11.3.8.5 Function Documentation

bsIncomingDataI()

```
void bsIncomingDataI (
    BufferedSerial * bsp,
    uint8_t b)
```

Handles incoming data.

This function must be called from the input interrupt service routine in order to enqueue incoming data and generate the related events.

Note

The incoming data event is only generated when the input queue becomes non-empty.

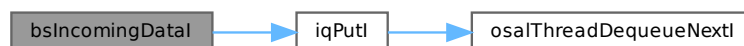
Parameters

in	<i>bsp</i>	pointer to a <code>BufferedSerial</code> structure
in	<i>b</i>	the byte to be written in the driver's Input Queue

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**bsRequestDataI()**

```
msg_t bsRequestDataI (
    BufferedSerial * bsp)
```

Handles outgoing data.

Must be called from the output interrupt service routine in order to get the next byte to be transmitted.

Note

In order to gain some performance it is suggested to not use this function directly but copy this code directly into the interrupt service routine.

Parameters

in	<i>bsp</i>	pointer to a BufferedSerial structure

Returns

The byte value read from the driver's output queue.

Return values

<i>MSG_TIMEOUT</i>	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**__buffered_serial_objinit_impl()**

```

CC_FORCE_INLINE void __buffered_serial_objinit_impl (
    void * ip,
    const void * vmt,
    uint8_t * ib,
    size_t ibsize,
    qnotify_t inotify,
    void * iarg,
    uint8_t * ob,
    size_t obsize,
    qnotify_t onotify,
    void * oarg) [inline], [static]
  
```

Object initialization implementation.

Parameters

in	<i>ip</i>	Pointer to a BufferedSerial structure to be initialized.
in	<i>vmt</i>	VMT pointer for the new object.
in	<i>ib</i>	pointer to the input buffer
in	<i>ibsize</i>	size of the input buffer

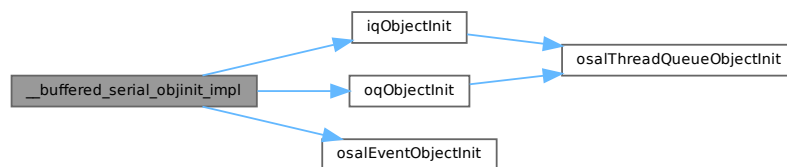
Parameters

in	<i>inotify</i>	pointer to a callback function that is invoked when some data is read from the input queue. The value can be <code>NULL</code> .
in	<i>iarg</i>	parameter for the input notification callback
in	<i>ob</i>	pointer to the output buffer
in	<i>obsz</i>	size of the output buffer
in	<i>onotify</i>	pointer to a callback function that is invoked when some data is written in the output queue. The value can be <code>NULL</code> .
in	<i>oarg</i>	parameter for the output notification callback

Function Class:

Object or module initializer function.

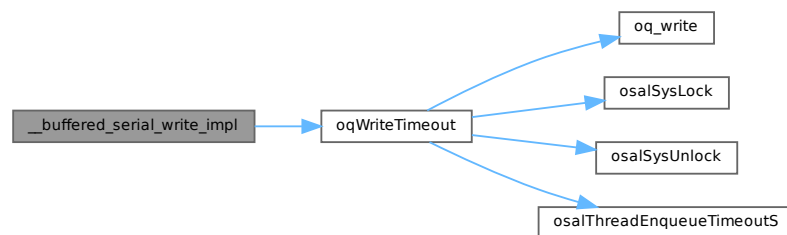
Here is the call graph for this function:

**__buffered_serial_write_impl()**

```

CC_FORCE_INLINE size_t __buffered_serial_write_impl (
    void * ip,
    const uint8_t * bp,
    size_t n) [inline], [static]
  
```

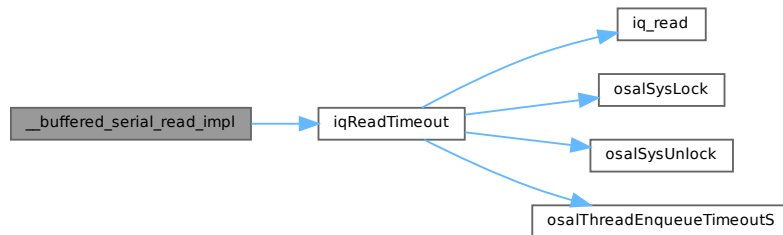
Here is the call graph for this function:



__buffered_serial_read_impl()

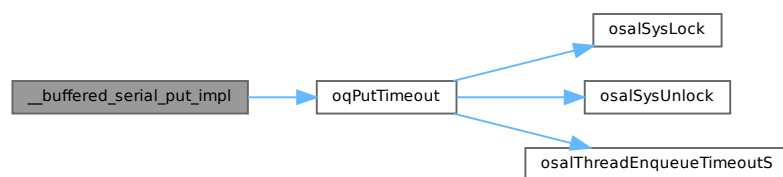
```
CC_FORCE_INLINE size_t __buffered_serial_read_impl (
    void * ip,
    uint8_t * bp,
    size_t n) [inline], [static]
```

Here is the call graph for this function:

**__buffered_serial_put_impl()**

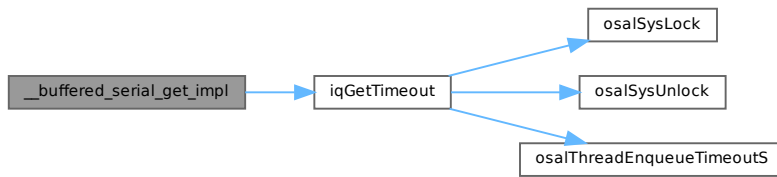
```
CC_FORCE_INLINE msg_t __buffered_serial_put_impl (
    void * ip,
    uint8_t b) [inline], [static]
```

Here is the call graph for this function:

**__buffered_serial_get_impl()**

```
CC_FORCE_INLINE msg_t __buffered_serial_get_impl (
    void * ip) [inline], [static]
```

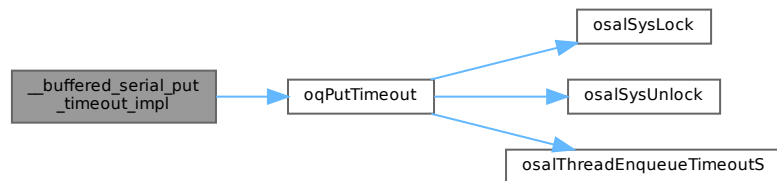
Here is the call graph for this function:



`__buffered_serial_put_timeout_impl()`

```
CC_FORCE_INLINE msg_t __buffered_serial_put_timeout_impl (
    void * ip,
    uint8_t b,
    sysinterval_t timeout) [inline], [static]
```

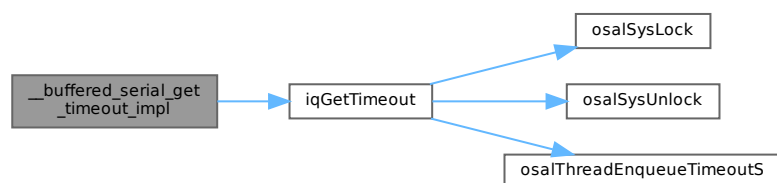
Here is the call graph for this function:



`__buffered_serial_get_timeout_impl()`

```
CC_FORCE_INLINE msg_t __buffered_serial_get_timeout_impl (
    void * ip,
    sysinterval_t timeout) [inline], [static]
```

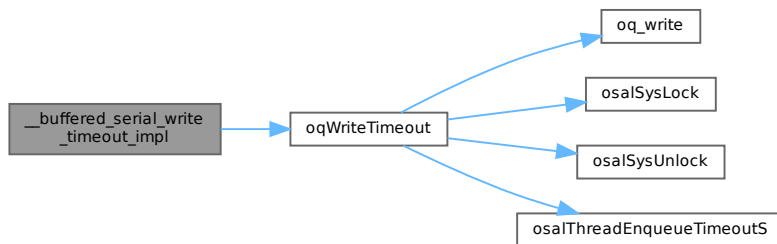
Here is the call graph for this function:



__buffered_serial_write_timeout_impl()

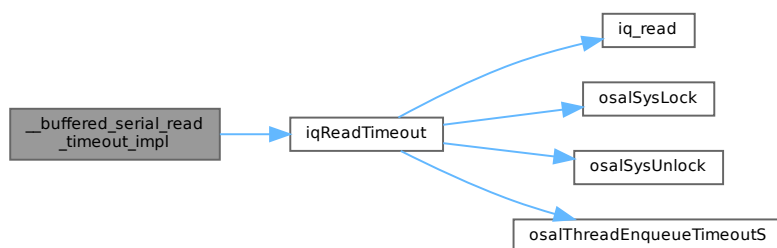
```
CC_FORCE_INLINE size_t __buffered_serial_write_timeout_impl (
    void * ip,
    const uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [inline], [static]
```

Here is the call graph for this function:

**__buffered_serial_read_timeout_impl()**

```
CC_FORCE_INLINE size_t __buffered_serial_read_timeout_impl (
    void * ip,
    uint8_t * bp,
    size_t n,
    sysinterval_t timeout) [inline], [static]
```

Here is the call graph for this function:

**__buffered_serial_ctl_impl()**

```
CC_FORCE_INLINE msg_t __buffered_serial_ctl_impl (
    void * ip,
    unsigned int operation,
    void * arg) [inline], [static]
```

7.2.11.3.9 Buffered Streams Class

7.2.11.3.9.1 Detailed Description

Wrapper for BaseSequentialStreams that allows some ungetting.

Data Structures

- struct [BufferedStreamAdapterVMT](#)
BufferedStreamAdapter virtual methods table.
- struct [BufferedStreamAdapter](#)
Buffered stream adapter object.

Macros

- #define [BUFSTREAM_BUFFER_SIZE](#) 1
Buffer size for unget.
- #define [_buffered_stream_adapter_data](#)
BufferedStreamAdapter specific data.

Functions

- static size_t [_writes](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [_reads](#) (void *ip, uint8_t *bp, size_t n)
- static msg_t [_put](#) (void *ip, uint8_t b)
- static msg_t [_get](#) (void *ip)
- static msg_t [_unget](#) (void *ip, uint8_t b)
- void [bsaObjectInit](#) ([BufferedStreamAdapter](#) *bsap, [BaseSequentialStream](#) *bssp)
Buffered stream adapter object initialization.

Variables

- static const struct [BufferedStreamAdapterVMT](#) vmt

7.2.11.3.9.2 Macro Definition Documentation

BUFSTREAM_BUFFER_SIZE

```
#define BUFSTREAM_BUFFER_SIZE 1
```

Buffer size for unget.

_buffered_stream_adapter_data

```
#define _buffered_stream_adapter_data
```

Value:

```
_base_buffered_stream_data
/* Pointer to a wrapped BaseSequentialStream object */
BaseSequentialStream* bssp;
/* Stream buffer */
uint8_t                buffer[BUFSTREAM_BUFFER_SIZE];
/* Index in the stream buffer */
size_t                ndx;
```

[BufferedStreamAdapter](#) specific data.

7.2.11.3.9.3 Function Documentation**_writes()**

```
size_t _writes (
    void * ip,
    const uint8_t * bp,
    size_t n) [static]
```

_reads()

```
size_t _reads (
    void * ip,
    uint8_t * bp,
    size_t n) [static]
```

_put()

```
msg_t _put (
    void * ip,
    uint8_t b) [static]
```

_get()

```
msg_t _get (
    void * ip) [static]
```

_unget()

```
msg_t _unget (
    void * ip,
    uint8_t b) [static]
```

bsaObjectInit()

```
void bsaObjectInit (
    BufferedStreamAdapter * bsap,
    BaseSequentialStream * bssp)
```

Buffered stream adapter object initialization.

Parameters

out	<i>bsap</i>	pointer to the <code>BufferedStream</code> object to be initialized
in	<i>bssp</i>	pointer to a <code>BaseSequentialStream</code> fulfilling object to be wrapped

7.2.11.3.9.4 Variable Documentation

vmt

```
const struct BufferedStreamAdapterVMT vmt [static]
```

Initial value:

```
= {
    (size_t)0, _writes, _reads, _put, _get, _unget
}
```

7.2.11.3.10 Output Formatter Utility

7.2.11.3.10.1 Detailed Description

Mini printf-like functionality.

Macros

- `#define MAX_FILLER 11`
- `#define FLOAT_PRECISION 9`
- `#define CHPRINTF_USE_FLOAT FALSE`
Float type support.

Functions

- static char * `long_to_string_with_divisor` (char *p, long num, unsigned radix, long divisor)
- static char * `ch_ltoa` (char *p, long num, unsigned radix)
- int `chvprintf` (`BaseSequentialStream` *chp, const char *fmt, va_list ap)
System formatted output function.
- int `chprintf` (`BaseSequentialStream` *chp, const char *fmt,...)
System formatted output function.
- int `chsnprintf` (char *str, size_t size, const char *fmt,...)
System formatted output function.
- int `chvsnprintf` (char *str, size_t size, const char *fmt, va_list ap)
System formatted output function.

7.2.11.3.10.2 Macro Definition Documentation

MAX_FILLER

```
#define MAX_FILLER 11
```

FLOAT_PRECISION

```
#define FLOAT_PRECISION 9
```

CHPRINTF_USE_FLOAT

```
#define CHPRINTF_USE_FLOAT FALSE
```

Float type support.

7.2.11.3.10.3 Function Documentation

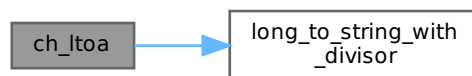
long_to_string_with_divisor()

```
char * long_to_string_with_divisor (  
    char * p,  
    long num,  
    unsigned radix,  
    long divisor) [static]
```

ch_ltoa()

```
char * ch_ltoa (  
    char * p,  
    long num,  
    unsigned radix) [static]
```

Here is the call graph for this function:



chvprintf()

```
int chvprintf (
    BaseSequentialStream * chp,
    const char * fmt,
    va_list ap)
```

System formatted output function.

This function implements a minimal `vprintf()`-like functionality with output on a `BaseSequentialStream`. The general parameters format is: `%[-][width]*[.precision]*[I|L]p`. The following parameter types (p) are supported:

- **x** hexadecimal integer.
- **X** hexadecimal long.
- **o** octal integer.
- **O** octal long.
- **d** decimal signed integer.
- **D** decimal signed long.
- **u** decimal unsigned integer.
- **U** decimal unsigned long.
- **c** character.
- **s** string.

Parameters

in	<i>chp</i>	pointer to a <code>BaseSequentialStream</code> implementing object
in	<i>fmt</i>	formatting string
in	<i>ap</i>	list of parameters

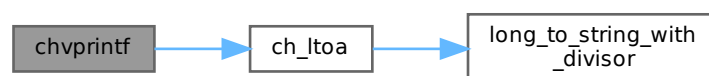
Returns

The number of bytes that would have been written to `chp` if no stream error occurs

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



chprintf()

```
int chprintf (
    BaseSequentialStream * chp,
    const char * fmt,
    ...)
```

System formatted output function.

This function implements a minimal `printf()` like functionality with output on a [BaseSequentialStream](#). The general parameters format is: `%[-][width|*][.precision|*][l|L]p`. The following parameter types (p) are supported:

- **x** hexadecimal integer.
- **X** hexadecimal long.
- **o** octal integer.
- **O** octal long.
- **d** decimal signed integer.
- **D** decimal signed long.
- **u** decimal unsigned integer.
- **U** decimal unsigned long.
- **c** character.
- **s** string.

Parameters

in	<i>chp</i>	pointer to a BaseSequentialStream implementing object
in	<i>fmt</i>	formatting string

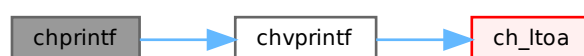
Returns

The number of bytes that would have been written to `chp` if no stream error occurs

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



chsnprintf()

```
int chsnprintf (
    char * str,
    size_t size,
    const char * fmt,
    ...)
```

System formatted output function.

This function implements a minimal `snprintf()`-like functionality. The general parameters format is: `%[-][width|*][.precision|*][|L]p`. The following parameter types (p) are supported:

- **x** hexadecimal integer.
- **X** hexadecimal long.
- **o** octal integer.
- **O** octal long.
- **d** decimal signed integer.
- **D** decimal signed long.
- **u** decimal unsigned integer.
- **U** decimal unsigned long.
- **c** character.
- **s** string.

Postcondition

`str` is NUL-terminated, unless `size` is 0.

Parameters

in	<i>str</i>	pointer to a buffer
in	<i>size</i>	maximum size of the buffer
in	<i>fmt</i>	formatting string

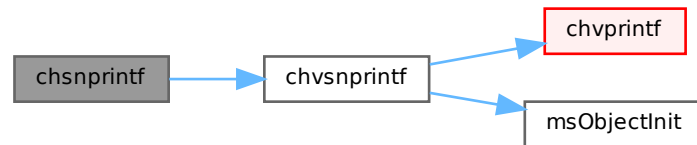
Returns

The number of characters (excluding the terminating NUL byte) that would have been stored in `str` if there was room.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**chvsprintf()**

```

int chvsprintf (
    char * str,
    size_t size,
    const char * fmt,
    va_list ap)

```

System formatted output function.

This function implements a minimal `vsnprintf()`-like functionality. The general parameters format is: `%[-][width|*][.precision|*][I|L]p`. The following parameter types (p) are supported:

- **x** hexadecimal integer.
- **X** hexadecimal long.
- **o** octal integer.
- **O** octal long.
- **d** decimal signed integer.
- **D** decimal signed long.
- **u** decimal unsigned integer.
- **U** decimal unsigned long.
- **c** character.
- **s** string.

Postcondition

`str` is NUL-terminated, unless `size` is 0.

Parameters

in	<i>str</i>	pointer to a buffer
in	<i>size</i>	maximum size of the buffer
in	<i>fmt</i>	formatting string
in	<i>ap</i>	list of parameters

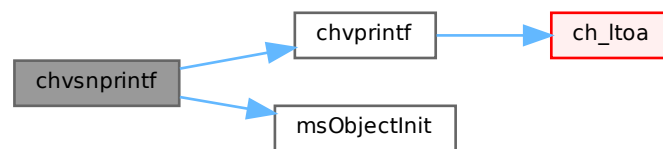
Returns

The number of characters (excluding the terminating NUL byte) that would have been stored in `str` if there was room.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.11.3.11 Input Formatter Utility****7.2.11.3.11.1 Detailed Description**

Mini scanf-like functionality.

Macros

- `#define CHSCANF_USE_FLOAT FALSE`
Float type support.

Functions

- static long `sym_to_val` (char sym, int base)
- int `chvscanf` (`BaseBufferedStream` *chp, const char *fmt, va_list ap)
System formatted input function.
- int `chscanf` (`BaseBufferedStream` *chp, const char *fmt,...)
System formatted input function.
- int `chsnscanf` (char *str, size_t size, const char *fmt,...)
System formatted input function.
- int `chvsnscanf` (char *str, size_t size, const char *fmt, va_list ap)
System formatted input function.

7.2.11.3.11.2 Macro Definition Documentation

CHSCANF_USE_FLOAT

```
#define CHSCANF_USE_FLOAT FALSE
```

Float type support.

7.2.11.3.11.3 Function Documentation

sym_to_val()

```
long sym_to_val (
    char sym,
    int base) [static]
```

chvscanf()

```
int chvscanf (
    BaseBufferedStream * chp,
    const char * fmt,
    va_list ap)
```

System formatted input function.

This function implements a minimal `vscanf()`-like functionality with input on a [BaseSequentialStream](#). The general parameters format is: `%[*][width][!|L]p` The following parameter types (p) are supported:

- **x** hexadecimal integer.
- **X** hexadecimal long.
- **o** octal integer.
- **O** octal long.
- **d** decimal signed integer.
- **D** decimal signed long.
- **u** decimal unsigned integer.
- **U** decimal unsigned long.
- **c** character.
- **s** string.

Parameters

in	<i>chp</i>	pointer to a <code>BufferedStream</code> implementing object
in	<i>fmt</i>	formatting string
in	<i>ap</i>	list of parameters

Returns

The number parameters in `ap` that have been successfully filled. This does not conform to the standard in that if a failure (either matching or input) occurs before any parameters are assigned, the function will return 0.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



chscanf()

```
int chscanf (  
    BaseBufferedStream * chp,  
    const char * fmt,  
    ...)
```

System formatted input function.

This function implements a minimal `scanf()` like functionality with input from a `BufferedStream`. The general parameters format is: `%[*][width][i|L]p` The following parameter types (`p`) are supported:

- **x** hexadecimal integer.
- **X** hexadecimal long.
- **o** octal integer.
- **O** octal long.
- **d** decimal signed integer.
- **D** decimal signed long.
- **u** decimal unsigned integer.
- **U** decimal unsigned long.
- **c** character.
- **s** string.

Parameters

in	<i>chp</i>	pointer to a <code>BufferedStream</code> implementing object
in	<i>fmt</i>	formatting string

Returns

The number parameters in `ap` that have been successfully filled. This does not conform to the standard in that if a failure (either matching or input) occurs before any parameters are assigned, the function will return 0.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**chscanf()**

```

int chscanf (
    char * str,
    size_t size,
    const char * fmt,
    ...)
  
```

System formatted input function.

This function implements a minimal `scanf()`-like functionality. The general parameters format is `%[*][width][I|L]p`. The following parameter types (`p`) are supported:

- **x** hexadecimal integer.
- **X** hexadecimal long.
- **o** octal integer.
- **O** octal long.
- **d** decimal signed integer.
- **D** decimal signed long.
- **u** decimal unsigned integer.
- **U** decimal unsigned long.
- **c** character.
- **s** string.

Parameters

in	<i>str</i>	pointer to a buffer
in	<i>size</i>	size of the buffer
in	<i>fmt</i>	formatting string

Returns

The number parameters in *ap* that have been successfully filled. This does not conform to the standard in that if a failure (either matching or input) occurs before any parameters are assigned, the function will return 0.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**chvnsscanf()**

```

int chvnsscanf (
    char * str,
    size_t size,
    const char * fmt,
    va_list ap)

```

System formatted input function.

This function implements a minimal `vsnsscanf()`-like functionality. The general parameters format is `%[*][width][I|L]p`. The following parameter types (p) are supported:

- **x** hexadecimal integer.
- **X** hexadecimal long.
- **o** octal integer.
- **O** octal long.
- **d** decimal signed integer.
- **D** decimal signed long.
- **u** decimal unsigned integer.
- **U** decimal unsigned long.
- **c** character.
- **s** string.

Parameters

in	<i>str</i>	pointer to a buffer
in	<i>size</i>	size of the buffer
in	<i>fmt</i>	formatting string
in	<i>ap</i>	list of parameters

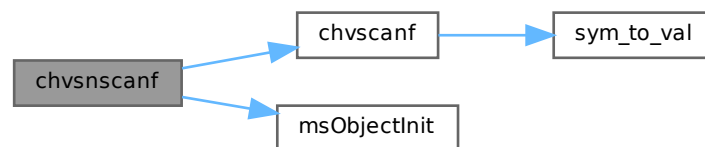
Returns

The number parameters in *ap* that have been successfully filled. This does not conform to the standard in that if a failure (either matching or input) occurs before any parameters are assigned, the function will return 0.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.11.4 Base Object****7.2.11.4.1 Detailed Description**

HAL uses concepts of Object Oriented Programming even if it is written in C. Things like simple inheritance, multiple inheritance and interfaces are used through the system. This module defines a "base object" that is the ancestor of all classes in the system.

Macro Functions (BaseObject)

- `#define objGetInstance(type, ip)`
Returns the instance pointer starting from an interface pointer.

Data Structures

- struct `BaseObjectVMT`
BaseObject virtual methods table.
- struct `BaseObject`
Base object class.

Macros

- `#define _base_object_methods`
BaseObject specific methods.
- `#define _base_object_data`
BaseObject specific data.

7.2.11.4.2 Macro Definition Documentation

7.2.11.4.2.1 _base_object_methods

```
#define _base_object_methods
```

Value:

```
/* Instance offset, used for multiple inheritance, normally zero. It
   represents the offset between the current object and the container
   object*/
size_t instance_offset;
```

BaseObject specific methods.

Note

This object defines no methods.

7.2.11.4.2.2 _base_object_data

```
#define _base_object_data
```

BaseObject specific data.

Note

This object defines no data.

7.2.11.4.2.3 objGetInstance

```
#define objGetInstance(
    type,
    ip)
```

Value:

```
(type)((size_t)(ip)) - (ip)->vmt->instance_offset)
```

Returns the instance pointer starting from an interface pointer.

Parameters

in	<i>type</i>	the type of the instance pointer, it is used for casting
in	<i>ip</i>	the interface pointer

Returns

A pointer to the object implementing the interface

7.2.12 Inner Code

HAL Inner Code.

7.2.12.1 Detailed Description

HAL Inner Code.

Topics

- [I/O Buffers Queues](#)
- [MMC/SD Block Device](#)
- [I/O Bytes Queues](#)

7.2.12.2 I/O Buffers Queues

7.2.12.2.1 Detailed Description

Buffers Queues are used when there is the need to exchange fixed-length data buffers between ISRs and threads. On the ISR side data can be exchanged only using buffers, on the thread side data can be exchanged both using buffers and/or using an emulation of regular byte queues. There are several kind of buffers queues:

- **Input queue**, unidirectional queue where the writer is the ISR side and the reader is the thread side.
- **Output queue**, unidirectional queue where the writer is the thread side and the reader is the ISR side.
- **Full duplex queue**, bidirectional queue. Full duplex queues are implemented by pairing an input queue and an output queue together.

Macro Functions

- `#define bqSizeX(bqp)`
Returns the queue's number of buffers.
- `#define bqSpacel(bqp)`
Return the ready buffers number.
- `#define bqGetLinkX(bqp)`
Returns the queue application-defined link.
- `#define bqSetLinkX(bqp, lk)`
Sets the queue application-defined link.
- `#define bqIsSuspendedX(bqp)`
Return the suspended state of the queue.
- `#define bqSuspendI(bqp)`
Puts the queue in suspended state.
- `#define bqResumeX(bqp)`
Resumes normal queue operations.
- `#define ibqlsEmptyI(ibqp)`
Evaluates to `true` if the specified input buffers queue is empty.
- `#define ibqlsFullI(ibqp)`
Evaluates to `true` if the specified input buffers queue is full.
- `#define obqlsEmptyI(obqp)`
Evaluates to `true` if the specified output buffers queue is empty.
- `#define obqlsFullI(obqp)`
Evaluates to `true` if the specified output buffers queue is full.

Data Structures

- struct [io_buffers_queue](#)
Structure of a generic buffers queue.

Macros

- #define [BUFFERS_CHUNKS_SIZE](#) 64
Maximum size of blocks copied in critical sections.
- #define [BQ_BUFFER_SIZE](#)(n, size)
Computes the size of a buffers queue buffer size.

Typedefs

- typedef struct [io_buffers_queue](#) [io_buffers_queue_t](#)
Type of a generic queue of buffers.
- typedef void(* [bqnotify_t](#)) ([io_buffers_queue_t](#) *bqp)
Double buffer notification callback type.
- typedef [io_buffers_queue_t](#) [input_buffers_queue_t](#)
Type of an input buffers queue.
- typedef [io_buffers_queue_t](#) [output_buffers_queue_t](#)
Type of an output buffers queue.

Functions

- void [ibqObjectInit](#) ([input_buffers_queue_t](#) *ibqp, bool suspended, uint8_t *bp, size_t size, size_t n, [bqnotify_t](#) infy, void *link)
Initializes an input buffers queue object.
- void [ibqResetI](#) ([input_buffers_queue_t](#) *ibqp)
Resets an input buffers queue.
- uint8_t * [ibqGetEmptyBufferI](#) ([input_buffers_queue_t](#) *ibqp)
Gets the next empty buffer from the queue.
- void [ibqPostFullBufferI](#) ([input_buffers_queue_t](#) *ibqp, size_t size)
Posts a new filled buffer to the queue.
- [msg_t](#) [ibqGetFullBufferTimeout](#) ([input_buffers_queue_t](#) *ibqp, [sysinterval_t](#) timeout)
Gets the next filled buffer from the queue.
- [msg_t](#) [ibqGetFullBufferTimeoutS](#) ([input_buffers_queue_t](#) *ibqp, [sysinterval_t](#) timeout)
Gets the next filled buffer from the queue.
- void [ibqReleaseEmptyBuffer](#) ([input_buffers_queue_t](#) *ibqp)
Releases the buffer back in the queue.
- void [ibqReleaseEmptyBufferS](#) ([input_buffers_queue_t](#) *ibqp)
Releases the buffer back in the queue.
- [msg_t](#) [ibqGetTimeout](#) ([input_buffers_queue_t](#) *ibqp, [sysinterval_t](#) timeout)
Input queue read with timeout.
- size_t [ibqReadTimeout](#) ([input_buffers_queue_t](#) *ibqp, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
Input queue read with timeout.
- void [obqObjectInit](#) ([output_buffers_queue_t](#) *obqp, bool suspended, uint8_t *bp, size_t size, size_t n, [bqnotify_t](#) onfy, void *link)
Initializes an output buffers queue object.

- void `obqResetI` (`output_buffers_queue_t` *obqp)
Resets an output buffers queue.
- uint8_t * `obqGetFullBufferI` (`output_buffers_queue_t` *obqp, size_t *sizep)
Gets the next filled buffer from the queue.
- void `obqReleaseEmptyBufferI` (`output_buffers_queue_t` *obqp)
Releases the next filled buffer back in the queue.
- msg_t `obqGetEmptyBufferTimeout` (`output_buffers_queue_t` *obqp, sysinterval_t timeout)
Gets the next empty buffer from the queue.
- msg_t `obqGetEmptyBufferTimeoutS` (`output_buffers_queue_t` *obqp, sysinterval_t timeout)
Gets the next empty buffer from the queue.
- void `obqPostFullBuffer` (`output_buffers_queue_t` *obqp, size_t size)
Posts a new filled buffer to the queue.
- void `obqPostFullBufferS` (`output_buffers_queue_t` *obqp, size_t size)
Posts a new filled buffer to the queue.
- msg_t `obqPutTimeout` (`output_buffers_queue_t` *obqp, uint8_t b, sysinterval_t timeout)
Output queue write with timeout.
- size_t `obqWriteTimeout` (`output_buffers_queue_t` *obqp, const uint8_t *bp, size_t n, sysinterval_t timeout)
Output queue write with timeout.
- bool `obqTryFlushI` (`output_buffers_queue_t` *obqp)
Flushes the current, partially filled, buffer to the queue.
- void `obqFlush` (`output_buffers_queue_t` *obqp)
Flushes the current, partially filled, buffer to the queue.

7.2.12.2.2 Macro Definition Documentation

7.2.12.2.2.1 BUFFERS_CHUNKS_SIZE

```
#define BUFFERS_CHUNKS_SIZE 64
```

Maximum size of blocks copied in critical sections.

Note

Increasing this value increases performance at expense of IRQ servicing efficiency.

It must be a power of two.

7.2.12.2.2.2 BQ_BUFFER_SIZE

```
#define BQ_BUFFER_SIZE(  
    n,  
    size)
```

Value:

```
((size_t)(size) + sizeof (size_t)) * (size_t)(n)
```

Computes the size of a buffers queue buffer size.

Parameters

in	<i>n</i>	number of buffers in the queue
in	<i>size</i>	size of the buffers

7.2.12.2.2.3 bqSizeX

```
#define bqSizeX(  
    bqp)
```

Value:

```
((bqp)->bn)
```

Returns the queue's number of buffers.

Parameters

in	<i>bqp</i>	pointer to an io_buffers_queue_t structure

Returns

The number of buffers.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.12.2.2.4 bqSpaceI

```
#define bqSpaceI(  
    bqp)
```

Value:

```
((bqp)->bcounter)
```

Return the ready buffers number.

Returns the number of filled buffers if used on an input queue or the number of empty buffers if used on an output queue.

Parameters

in	<i>bqp</i>	pointer to an io_buffers_queue_t structure

Returns

The number of ready buffers.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.2.2.5 bqGetLinkX

```
#define bqGetLinkX(  
    bqp)
```

Value:

```
((bqp)->link)
```

Returns the queue application-defined link.

Parameters

in	<i>bqp</i>	pointer to an io_buffers_queue_t structure

Returns

The application-defined link.

Function Class:

Special function, this function has special requirements see the notes.

7.2.12.2.2.6 bqSetLinkX

```
#define bqSetLinkX(  
    bqp,  
    lk)
```

Value:

```
((bqp)->link = lk)
```

Sets the queue application-defined link.

Parameters

in	<i>bqp</i>	pointer to an io_buffers_queue_t structure
in	<i>lk</i>	The application-defined link.

Function Class:

Special function, this function has special requirements see the notes.

7.2.12.2.2.7 bqIsSuspendedX

```
#define bqIsSuspendedX(  
    bqp)
```

Value:

```
((bqp)->suspended)
```

Return the suspended state of the queue.

Parameters

in	<i>bqp</i>	pointer to an <code>io_buffers_queue_t</code> structure

Returns

The suspended state.

Return values

<i>false</i>	if blocking access to the queue is enabled.
<i>true</i>	if blocking access to the queue is suspended.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.12.2.2.8 bqSuspendI

```
#define bqSuspendI(
    bqp)
```

Value:

```
{
    (bqp)->suspended = true;
    osalThreadDequeueAllI(&(bqp)->waiting, MSG_RESET);
}
```

Puts the queue in suspended state.

When the queue is put in suspended state all waiting threads are woken with message `MSG_RESET` and subsequent attempt at waiting on the queue will result in an immediate return with `MSG_RESET` message.

Note

The content of the queue is not altered, queues can be accessed is suspended state until a blocking operation is met then a `MSG_RESET` occurs.

Parameters

in	<i>bqp</i>	pointer to an <code>io_buffers_queue_t</code> structure

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.2.2.9 bqResumeX

```
#define bqResumeX(
    bqp)
```

Value:

```
{
    (bqp)->suspended = false;
}
```

Resumes normal queue operations.

Parameters

in	<i>bqp</i>	pointer to an <code>io_buffers_queue_t</code> structure

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.12.2.2.10 ibqIsEmptyI

```
#define ibqIsEmptyI(  
    ibqp)
```

Value:

```
((bool) (bqSpaceI(ibqp) == 0U))
```

Evaluates to `true` if the specified input buffers queue is empty.

Parameters

in	<i>ibqp</i>	pointer to an <code>input_buffers_queue_t</code> structure

Returns

The queue status.

Return values

<i>false</i>	if the queue is not empty.
<i>true</i>	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.2.2.11 ibqIsFullI

```
#define ibqIsFullI(  
    ibqp)
```

Value:

```
/*lint -save -e9007 [13.5] No side effects, a pointer is passed.*/      \  
((bool) (((ibqp)->bwrptr == (ibqp)->brdptr) && ((ibqp)->bcounter != 0U))) \  
/*lint -restore*/
```

Evaluates to `true` if the specified input buffers queue is full.

Parameters

in	<i>ibqp</i>	pointer to an input_buffers_queue_t structure

Returns

The queue status.

Return values

<i>false</i>	if the queue is not full.
<i>true</i>	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.2.2.12 obqIsEmptyI

```
#define obqIsEmptyI(
    obqp)
```

Value:

```
/*lint -save -e9007 [13.5] No side effects, a pointer is passed.*/ \
((bool)((obqp)->bwrptr == (obqp)->brdptr) && ((obqp)->bcounter != 0U)) \
/*lint -restore*/
```

Evaluates to `true` if the specified output buffers queue is empty.

Parameters

in	<i>obqp</i>	pointer to an output_buffers_queue_t structure

Returns

The queue status.

Return values

<i>false</i>	if the queue is not empty.
<i>true</i>	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.2.2.13 obqIsFullI

```
#define obqIsFullI(
    obqp)
```

Value:

```
((bool)(bqSpaceI(obqp) == 0U))
```

Evaluates to `true` if the specified output buffers queue is full.

Parameters

in	<i>obqp</i>	pointer to an output_buffers_queue_t structure

Returns

The queue status.

Return values

<i>false</i>	if the queue is not full.
<i>true</i>	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.2.3 Typedef Documentation**7.2.12.2.3.1 io_buffers_queue_t**

```
typedef struct io_buffers_queue io_buffers_queue_t
```

Type of a generic queue of buffers.

7.2.12.2.3.2 bqnotify_t

```
typedef void(* bqnotify_t) (io_buffers_queue_t *bqp)
```

Double buffer notification callback type.

Parameters

in	<i>bqp</i>	the buffers queue pointer

7.2.12.2.3.3 input_buffers_queue_t

```
typedef io_buffers_queue_t input_buffers_queue_t
```

Type of an input buffers queue.

7.2.12.2.3.4 output_buffers_queue_t

```
typedef io_buffers_queue_t output_buffers_queue_t
```

Type of an output buffers queue.

7.2.12.2.4 Function Documentation

7.2.12.2.4.1 ibqObjectInit()

```
void ibqObjectInit (
    input_buffers_queue_t * ibqp,
    bool suspended,
    uint8_t * bp,
    size_t size,
    size_t n,
    bqnotify_t infy,
    void * link)
```

Initializes an input buffers queue object.

Parameters

out	<i>ibqp</i>	pointer to the input_buffers_queue_t object
in	<i>suspended</i>	initial state of the queue
in	<i>bp</i>	pointer to a memory area allocated for buffers
in	<i>size</i>	buffers size
in	<i>n</i>	number of buffers
in	<i>infy</i>	callback called when a buffer is returned to the queue
in	<i>link</i>	application defined pointer

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.12.2.4.2 ibqResetI()

```
void ibqResetI (
    input_buffers_queue_t * ibqp)
```

Resets an input buffers queue.

All the data in the input buffers queue is erased and lost, any waiting thread is resumed with status `MSG_RESET`.

Note

A reset operation can be used by a low level driver in order to obtain immediate attention from the high level layers.

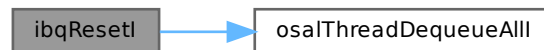
Parameters

in	<i>ibqp</i>	pointer to the <code>input_buffers_queue_t</code> object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.12.2.4.3 ibqGetEmptyBufferI()**

```
uint8_t * ibqGetEmptyBufferI (
    input_buffers_queue_t * ibqp)
```

Gets the next empty buffer from the queue.

Note

The function always returns the same buffer if called repeatedly.

Parameters

in	<i>ibqp</i>	pointer to the <code>input_buffers_queue_t</code> object

Returns

A pointer to the next buffer to be filled.

Return values

<i>NULL</i>	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.2.4.4 ibqPostFullBufferI()

```
void ibqPostFullBufferI (
    input_buffers_queue_t * ibqp,
    size_t size)
```

Posts a new filled buffer to the queue.

Parameters

in	<i>ibqp</i>	pointer to the input_buffers_queue_t object
in	<i>size</i>	used size of the buffer, cannot be zero

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.12.2.4.5 ibqGetFullBufferTimeout()

```

msg_t ibqGetFullBufferTimeout (
    input_buffers_queue_t * ibqp,
    sysinterval_t timeout)
  
```

Gets the next filled buffer from the queue.

Note

The function always acquires the same buffer if called repeatedly.

Postcondition

After calling the function the fields `ptr` and `top` are set at beginning and end of the buffer data or `NULL` if the queue is empty.

Parameters

in	<i>ibqp</i>	pointer to the input_buffers_queue_t object
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <code>TIME_IMMEDIATE</code> immediate timeout. • <code>TIME_INFINITE</code> no timeout.

Returns

The operation status.

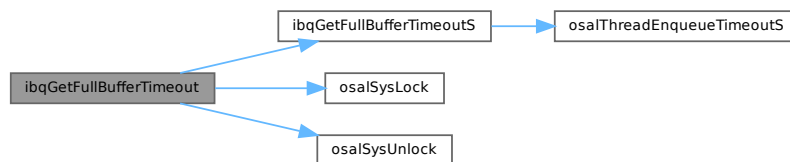
Return values

<i>MSG_OK</i>	if a buffer has been acquired.
<i>MSG_TIMEOUT</i>	if the specified time expired.
<i>MSG_RESET</i>	if the queue has been reset or has been put in suspended state.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.2.4.6 ibqGetFullBufferTimeoutS()

```

msg_t ibqGetFullBufferTimeoutS (
    input_buffers_queue_t * ibqp,
    sysinterval_t timeout)
  
```

Gets the next filled buffer from the queue.

Note

The function always acquires the same buffer if called repeatedly.

Postcondition

After calling the function the fields `ptr` and `top` are set at beginning and end of the buffer data or `NULL` if the queue is empty.

Parameters

in	<i>ibqp</i>	pointer to the <code>input_buffers_queue_t</code> object
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

Return values

<i>MSG_OK</i>	if a buffer has been acquired.
<i>MSG_TIMEOUT</i>	if the specified time expired.
<i>MSG_RESET</i>	if the queue has been reset or has been put in suspended state.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

Here is the call graph for this function:



7.2.12.2.4.7 ibqReleaseEmptyBuffer()

```
void ibqReleaseEmptyBuffer (
    input_buffers_queue_t * ibqp)
```

Releases the buffer back in the queue.

Note

The object callback is called after releasing the buffer.

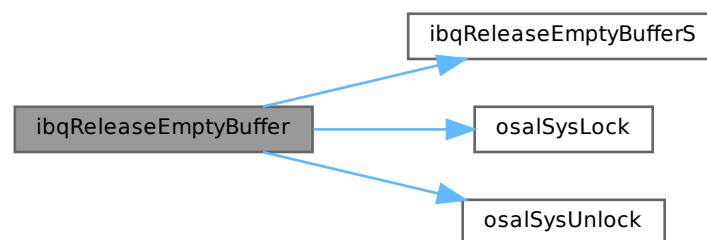
Parameters

in	<i>ibqp</i>	pointer to the <code>input_buffers_queue_t</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.2.4.8 **ibqReleaseEmptyBufferS()**

```
void ibqReleaseEmptyBufferS (  
    input_buffers_queue_t * ibqp)
```

Releases the buffer back in the queue.

Note

The object callback is called after releasing the buffer.

Parameters

in	<i>ibqp</i>	pointer to the <code>input_buffers_queue_t</code> object

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

7.2.12.2.4.9 **ibqGetTimeout()**

```
msg_t ibqGetTimeout (  
    input_buffers_queue_t * ibqp,  
    sysinterval_t timeout)
```

Input queue read with timeout.

This function reads a byte value from an input queue. If the queue is empty then the calling thread is suspended until a new buffer arrives in the queue or a timeout occurs.

Parameters

in	<i>ibqp</i>	pointer to the <code>input_buffers_queue_t</code> object
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <code>TIME_IMMEDIATE</code> immediate timeout. • <code>TIME_INFINITE</code> no timeout.

Returns

A byte value from the queue.

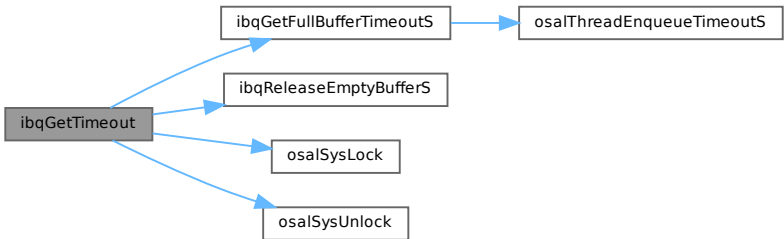
Return values

<code>MSG_TIMEOUT</code>	if the specified time expired.
<code>MSG_RESET</code>	if the queue has been reset or has been put in suspended state.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.2.4.10 `ibqReadTimeout()`

```
size_t ibqReadTimeout (
    input_buffers_queue_t * ibqp,
    uint8_t * bp,
    size_t n,
    sysinterval_t timeout)
```

Input queue read with timeout.

The function reads data from an input queue into a buffer. The operation completes when the specified amount of data has been transferred or after the specified timeout or if the queue has been reset.

Parameters

in	<i>ibqp</i>	pointer to the <code>input_buffers_queue_t</code> object
out	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <code>TIME_IMMEDIATE</code> immediate timeout. • <code>TIME_INFINITE</code> no timeout.

Returns

The number of bytes effectively transferred.

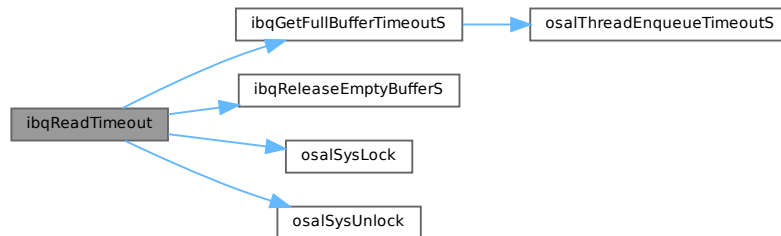
Return values

0	if a timeout occurred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.12.2.4.11 obqObjectInit()**

```

void obqObjectInit (
    output_buffers_queue_t * obqp,
    bool suspended,
    uint8_t * bp,
    size_t size,
    size_t n,
    bqnotify_t onfy,
    void * link)
  
```

Initializes an output buffers queue object.

Parameters

out	<i>obqp</i>	pointer to the <code>output_buffers_queue_t</code> object
in	<i>suspended</i>	initial state of the queue
in	<i>bp</i>	pointer to a memory area allocated for buffers
in	<i>size</i>	buffers size
in	<i>n</i>	number of buffers
in	<i>onfy</i>	callback called when a buffer is posted in the queue
in	<i>link</i>	application defined pointer

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.12.2.4.12 obqResetI()

```
void obqResetI (  
    output_buffers_queue_t * obqp)
```

Resets an output buffers queue.

All the data in the output buffers queue is erased and lost, any waiting thread is resumed with status MSG_RESET.

Note

A reset operation can be used by a low level driver in order to obtain immediate attention from the high level layers.

Parameters

in	<i>obqp</i>	pointer to the <code>output_buffers_queue_t</code> object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.12.2.4.13 obqGetFullBufferI()

```
uint8_t * obqGetFullBufferI (  
    output_buffers_queue_t * obqp,  
    size_t * sizep)
```

Gets the next filled buffer from the queue.

Note

The function always returns the same buffer if called repeatedly.

Parameters

in	<i>obqp</i>	pointer to the <code>output_buffers_queue_t</code> object
out	<i>sizep</i>	pointer to the filled buffer size

Returns

A pointer to the filled buffer.

Return values

<code>NULL</code>	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.2.4.14 obqReleaseEmptyBufferI()

```
void obqReleaseEmptyBufferI (
    output_buffers_queue_t * obqp)
```

Releases the next filled buffer back in the queue.

Parameters

in	<code>obqp</code>	pointer to the <code>output_buffers_queue_t</code> object

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.12.2.4.15 obqGetEmptyBufferTimeout()**

```
msg_t obqGetEmptyBufferTimeout (
    output_buffers_queue_t * obqp,
    sysinterval_t timeout)
```

Gets the next empty buffer from the queue.

Note

The function always acquires the same buffer if called repeatedly.

Postcondition

After calling the function the fields `ptr` and `top` are set at beginning and end of the buffer data or `NULL` if the queue is empty.

Parameters

in	<i>obqp</i>	pointer to the output_buffers_queue_t object
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

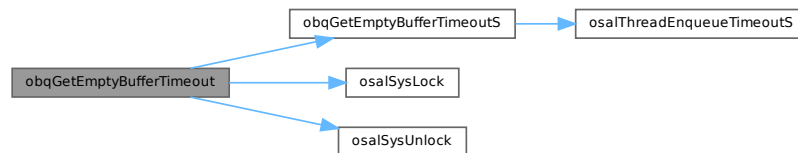
Return values

<i>MSG_OK</i>	if a buffer has been acquired.
<i>MSG_TIMEOUT</i>	if the specified time expired.
<i>MSG_RESET</i>	if the queue has been reset or has been put in suspended state.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.2.4.16 obqGetEmptyBufferTimeoutS()

```

msg_t obqGetEmptyBufferTimeoutS (
    output_buffers_queue_t * obqp,
    sysinterval_t timeout)

```

Gets the next empty buffer from the queue.

Note

The function always acquires the same buffer if called repeatedly.

Postcondition

After calling the function the fields `ptr` and `top` are set at beginning and end of the buffer data or `NULL` if the queue is empty.

Parameters

in	<i>obqp</i>	pointer to the output_buffers_queue_t object
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

Return values

<i>MSG_OK</i>	if a buffer has been acquired.
<i>MSG_TIMEOUT</i>	if the specified time expired.
<i>MSG_RESET</i>	if the queue has been reset or has been put in suspended state.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

Here is the call graph for this function:

**7.2.12.2.4.17 obqPostFullBuffer()**

```
void obqPostFullBuffer (  
    output\_buffers\_queue\_t * obqp,  
    size_t size)
```

Posts a new filled buffer to the queue.

Note

The object callback is called after releasing the buffer.

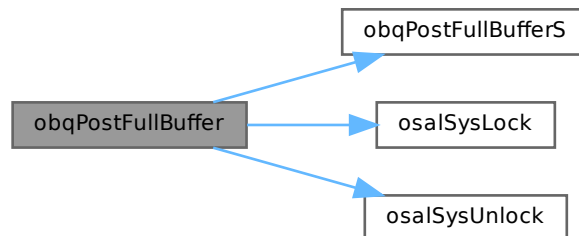
Parameters

in	<i>obqp</i>	pointer to the output_buffers_queue_t object
in	<i>size</i>	used size of the buffer, cannot be zero

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:

**7.2.12.2.4.18 obqPostFullBufferS()**

```
void obqPostFullBufferS (
    output_buffers_queue_t * obqp,
    size_t size)
```

Posts a new filled buffer to the queue.

Note

The object callback is called after releasing the buffer.

Parameters

in	<i>obqp</i>	pointer to the <code>output_buffers_queue_t</code> object
in	<i>size</i>	used size of the buffer, cannot be zero

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

7.2.12.2.4.19 obqPutTimeout()

```
msg_t obqPutTimeout (
    output_buffers_queue_t * obqp,
    uint8_t b,
    sysinterval_t timeout)
```

Output queue write with timeout.

This function writes a byte value to an output queue. If the queue is full then the calling thread is suspended until a new buffer is freed in the queue or a timeout occurs.

Parameters

in	<i>obqp</i>	pointer to the output_buffers_queue_t object
in	<i>b</i>	byte value to be transferred
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

A byte value from the queue.

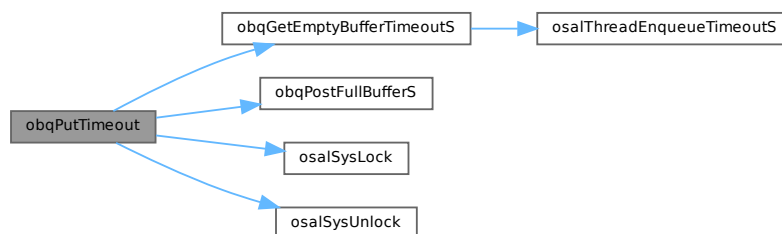
Return values

<i>MSG_TIMEOUT</i>	if the specified time expired.
<i>MSG_RESET</i>	if the queue has been reset or has been put in suspended state.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.2.4.20 obqWriteTimeout()

```

size_t obqWriteTimeout (
    output_buffers_queue_t * obqp,
    const uint8_t * bp,
    size_t n,
    sysinterval_t timeout)

```

Output queue write with timeout.

The function writes data from a buffer to an output queue. The operation completes when the specified amount of data has been transferred or after the specified timeout or if the queue has been reset.

Parameters

in	<i>obqp</i>	pointer to the output_buffers_queue_t object
in	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The number of bytes effectively transferred.

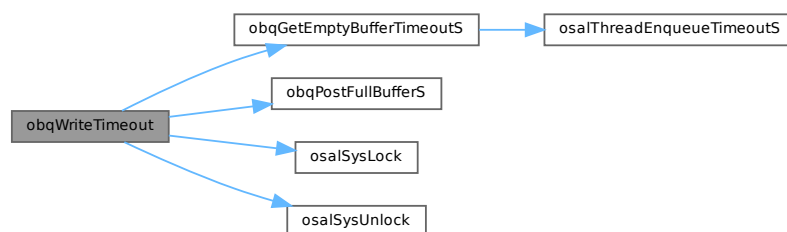
Return values

0	if a timeout occurred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.2.4.21 obqTryFlushI()

```
bool obqTryFlushI (
    output_buffers_queue_t * obqp)
```

Flushes the current, partially filled, buffer to the queue.

Note

The notification callback is not invoked because the function is meant to be called from ISR context. An operation status is returned instead.

Parameters

in	<i>obqp</i>	pointer to the output_buffers_queue_t object

Returns

The operation status.

Return values

<i>false</i>	if no new filled buffer has been posted to the queue.
<i>true</i>	if a new filled buffer has been posted to the queue.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.2.4.22 obqFlush()

```
void obqFlush (  
    output\_buffers\_queue\_t * obqp)
```

Flushes the current, partially filled, buffer to the queue.

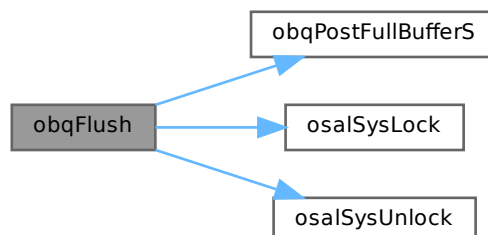
Parameters

in	<i>obqp</i>	pointer to the output_buffers_queue_t object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.3 MMC/SD Block Device

7.2.12.3.1 Detailed Description

This module implements a common ancestor for all device drivers accessing MMC or SD cards. This interface inherits the state machine and the interface from the [Abstract I/O Block Device](#) module.

SD/MMC status conditions

- #define [MMCSD_STS_IDLE](#) 0U
- #define [MMCSD_STS_READY](#) 1U
- #define [MMCSD_STS_IDENT](#) 2U
- #define [MMCSD_STS_STBY](#) 3U
- #define [MMCSD_STS_TRAN](#) 4U
- #define [MMCSD_STS_DATA](#) 5U
- #define [MMCSD_STS_RCV](#) 6U
- #define [MMCSD_STS_PRG](#) 7U
- #define [MMCSD_STS_DIS](#) 8U

SD/MMC commands

- #define [MMCSD_CMD_GO_IDLE_STATE](#) 0U
- #define [MMCSD_CMD_INIT](#) 1U
- #define [MMCSD_CMD_ALL_SEND_CID](#) 2U
- #define [MMCSD_CMD_SEND_RELATIVE_ADDR](#) 3U
- #define [MMCSD_CMD_SET_RELATIVE_ADDR](#) [MMCSD_CMD_SEND_RELATIVE_ADDR](#)
- #define [MMCSD_CMD_SET_BUS_WIDTH](#) 6U
- #define [MMCSD_CMD_SWITCH](#) [MMCSD_CMD_SET_BUS_WIDTH](#)
- #define [MMCSD_CMD_SEL_DESEL_CARD](#) 7U
- #define [MMCSD_CMD_SEND_IF_COND](#) 8U
- #define [MMCSD_CMD_SEND_EXT_CSD](#) [MMCSD_CMD_SEND_IF_COND](#)
- #define [MMCSD_CMD_SEND_CSD](#) 9U
- #define [MMCSD_CMD_SEND_CID](#) 10U
- #define [MMCSD_CMD_STOP_TRANSMISSION](#) 12U
- #define [MMCSD_CMD_SEND_STATUS](#) 13U
- #define [MMCSD_CMD_SET_BLOCKLEN](#) 16U
- #define [MMCSD_CMD_READ_SINGLE_BLOCK](#) 17U
- #define [MMCSD_CMD_READ_MULTIPLE_BLOCK](#) 18U
- #define [MMCSD_CMD_SET_BLOCK_COUNT](#) 23U
- #define [MMCSD_CMD_WRITE_BLOCK](#) 24U
- #define [MMCSD_CMD_WRITE_MULTIPLE_BLOCK](#) 25U
- #define [MMCSD_CMD_ERASE_RW_BLK_START](#) 32U
- #define [MMCSD_CMD_ERASE_RW_BLK_END](#) 33U
- #define [MMCSD_CMD_ERASE](#) 38U
- #define [MMCSD_CMD_APP_OP_COND](#) 41U
- #define [MMCSD_CMD_LOCK_UNLOCK](#) 42U
- #define [MMCSD_CMD_APP_CMD](#) 55U
- #define [MMCSD_CMD_READ_OCR](#) 58U

CSD record offsets

- #define [MMCSD_CSD_MMC_CSD_STRUCTURE_SLICE](#) 127U, 126U
- #define [MMCSD_CSD_MMC_SPEC_VERS_SLICE](#) 125U, 122U
- #define [MMCSD_CSD_MMC_TAAC_SLICE](#) 119U, 112U
- #define [MMCSD_CSD_MMC_NSAC_SLICE](#) 111U, 104U
- #define [MMCSD_CSD_MMC_TRAN_SPEED_SLICE](#) 103U, 96U
- #define [MMCSD_CSD_MMC_CCC_SLICE](#) 95U, 84U
- #define [MMCSD_CSD_MMC_READ_BL_LEN_SLICE](#) 83U, 80U
- #define [MMCSD_CSD_MMC_READ_BL_PARTIAL_SLICE](#) 79U, 79U
- #define [MMCSD_CSD_MMC_WRITE_BLK_MISALIGN_SLICE](#) 78U, 78U
- #define [MMCSD_CSD_MMC_READ_BLK_MISALIGN_SLICE](#) 77U, 77U
- #define [MMCSD_CSD_MMC_DSR_IMP_SLICE](#) 76U, 76U
- #define [MMCSD_CSD_MMC_C_SIZE_SLICE](#) 73U, 62U
- #define [MMCSD_CSD_MMC_VDD_R_CURR_MIN_SLICE](#) 61U, 59U
- #define [MMCSD_CSD_MMC_VDD_R_CURR_MAX_SLICE](#) 58U, 56U
- #define [MMCSD_CSD_MMC_VDD_W_CURR_MIN_SLICE](#) 55U, 53U
- #define [MMCSD_CSD_MMC_VDD_W_CURR_MAX_SLICE](#) 52U, 50U
- #define [MMCSD_CSD_MMC_C_SIZE_MULT_SLICE](#) 49U, 47U
- #define [MMCSD_CSD_MMC_ERASE_GRP_SIZE_SLICE](#) 46U, 42U
- #define [MMCSD_CSD_MMC_ERASE_GRP_MULT_SLICE](#) 41U, 37U
- #define [MMCSD_CSD_MMC_WP_GRP_SIZE_SLICE](#) 36U, 32U
- #define [MMCSD_CSD_MMC_WP_GRP_ENABLE_SLICE](#) 31U, 31U
- #define [MMCSD_CSD_MMC_DEFAULT_ECC_SLICE](#) 30U, 29U
- #define [MMCSD_CSD_MMC_R2W_FACTOR_SLICE](#) 28U, 26U
- #define [MMCSD_CSD_MMC_WRITE_BL_LEN_SLICE](#) 25U, 22U
- #define [MMCSD_CSD_MMC_WRITE_BL_PARTIAL_SLICE](#) 21U, 21U
- #define [MMCSD_CSD_MMC_CONTENT_PROT_APP_SLICE](#) 16U, 16U
- #define [MMCSD_CSD_MMC_FILE_FORMAT_GRP_SLICE](#) 15U, 15U
- #define [MMCSD_CSD_MMC_COPY_SLICE](#) 14U, 14U
- #define [MMCSD_CSD_MMC_PERM_WRITE_PROTECT_SLICE](#) 13U, 13U
- #define [MMCSD_CSD_MMC_TMP_WRITE_PROTECT_SLICE](#) 12U, 12U
- #define [MMCSD_CSD_MMC_FILE_FORMAT_SLICE](#) 11U, 10U
- #define [MMCSD_CSD_MMC_ECC_SLICE](#) 9U, 8U
- #define [MMCSD_CSD_MMC_CRC_SLICE](#) 7U, 1U
- #define [MMCSD_CSD_20_CRC_SLICE](#) 7U, 1U
- #define [MMCSD_CSD_20_FILE_FORMAT_SLICE](#) 11U, 10U
- #define [MMCSD_CSD_20_TMP_WRITE_PROTECT_SLICE](#) 12U, 12U
- #define [MMCSD_CSD_20_PERM_WRITE_PROTECT_SLICE](#) 13U, 13U
- #define [MMCSD_CSD_20_COPY_SLICE](#) 14U, 14U
- #define [MMCSD_CSD_20_FILE_FORMAT_GRP_SLICE](#) 15U, 15U
- #define [MMCSD_CSD_20_WRITE_BL_PARTIAL_SLICE](#) 21U, 21U
- #define [MMCSD_CSD_20_WRITE_BL_LEN_SLICE](#) 25U, 12U
- #define [MMCSD_CSD_20_R2W_FACTOR_SLICE](#) 28U, 26U
- #define [MMCSD_CSD_20_WP_GRP_ENABLE_SLICE](#) 31U, 31U
- #define [MMCSD_CSD_20_WP_GRP_SIZE_SLICE](#) 38U, 32U
- #define [MMCSD_CSD_20_ERASE_SECTOR_SIZE_SLICE](#) 45U, 39U
- #define [MMCSD_CSD_20_ERASE_BLK_EN_SLICE](#) 46U, 46U
- #define [MMCSD_CSD_20_C_SIZE_SLICE](#) 69U, 48U
- #define [MMCSD_CSD_20_DSR_IMP_SLICE](#) 76U, 76U
- #define [MMCSD_CSD_20_READ_BLK_MISALIGN_SLICE](#) 77U, 77U
- #define [MMCSD_CSD_20_WRITE_BLK_MISALIGN_SLICE](#) 78U, 78U
- #define [MMCSD_CSD_20_READ_BL_PARTIAL_SLICE](#) 79U, 79U
- #define [MMCSD_CSD_20_READ_BL_LEN_SLICE](#) 83U, 80U
- #define [MMCSD_CSD_20_CCC_SLICE](#) 95U, 84U

- #define `MMCSD_CSD_20_TRANS_SPEED_SLICE` 103U, 96U
- #define `MMCSD_CSD_20_NSAC_SLICE` 111U, 104U
- #define `MMCSD_CSD_20_TAAC_SLICE` 119U, 112U
- #define `MMCSD_CSD_20_CSD_STRUCTURE_SLICE` 127U, 126U
- #define `MMCSD_CSD_10_CRC_SLICE` `MMCSD_CSD_20_CRC_SLICE`
- #define `MMCSD_CSD_10_FILE_FORMAT_SLICE` `MMCSD_CSD_20_FILE_FORMAT_SLICE`
- #define `MMCSD_CSD_10_TMP_WRITE_PROTECT_SLICE` `MMCSD_CSD_20_TMP_WRITE_PROTECT_SLICE`
- #define `MMCSD_CSD_10_PERM_WRITE_PROTECT_SLICE` `MMCSD_CSD_20_PERM_WRITE_PROTECT_SLICE`
- #define `MMCSD_CSD_10_COPY_SLICE` `MMCSD_CSD_20_COPY_SLICE`
- #define `MMCSD_CSD_10_FILE_FORMAT_GRP_SLICE` `MMCSD_CSD_20_FILE_FORMAT_GRP_SLICE`
- #define `MMCSD_CSD_10_WRITE_BL_PARTIAL_SLICE` `MMCSD_CSD_20_WRITE_BL_PARTIAL_SLICE`
- #define `MMCSD_CSD_10_WRITE_BL_LEN_SLICE` `MMCSD_CSD_20_WRITE_BL_LEN_SLICE`
- #define `MMCSD_CSD_10_R2W_FACTOR_SLICE` `MMCSD_CSD_20_R2W_FACTOR_SLICE`
- #define `MMCSD_CSD_10_WP_GRP_ENABLE_SLICE` `MMCSD_CSD_20_WP_GRP_ENABLE_SLICE`
- #define `MMCSD_CSD_10_WP_GRP_SIZE_SLICE` `MMCSD_CSD_20_WP_GRP_SIZE_SLICE`
- #define `MMCSD_CSD_10_ERASE_SECTOR_SIZE_SLICE` `MMCSD_CSD_20_ERASE_SECTOR_SIZE_SLICE`
- #define `MMCSD_CSD_10_ERASE_BLK_EN_SLICE` `MMCSD_CSD_20_ERASE_BLK_EN_SLICE`
- #define `MMCSD_CSD_10_C_SIZE_MULT_SLICE` 49U, 47U
- #define `MMCSD_CSD_10_VDD_W_CURR_MAX_SLICE` 52U, 50U
- #define `MMCSD_CSD_10_VDD_W_CURR_MIN_SLICE` 55U, 53U
- #define `MMCSD_CSD_10_VDD_R_CURR_MAX_SLICE` 58U, 56U
- #define `MMCSD_CSD_10_VDD_R_CURR_MIX_SLICE` 61U, 59U
- #define `MMCSD_CSD_10_C_SIZE_SLICE` 73U, 62U
- #define `MMCSD_CSD_10_DSR_IMP_SLICE` `MMCSD_CSD_20_DSR_IMP_SLICE`
- #define `MMCSD_CSD_10_READ_BLK_MISALIGN_SLICE` `MMCSD_CSD_20_READ_BLK_MISALIGN_SLICE`
- #define `MMCSD_CSD_10_WRITE_BLK_MISALIGN_SLICE` `MMCSD_CSD_20_WRITE_BLK_MISALIGN_SLICE`
- #define `MMCSD_CSD_10_READ_BL_PARTIAL_SLICE` `MMCSD_CSD_20_READ_BL_PARTIAL_SLICE`
- #define `MMCSD_CSD_10_READ_BL_LEN_SLICE` 83U, 80U
- #define `MMCSD_CSD_10_CCC_SLICE` `MMCSD_CSD_20_CCC_SLICE`
- #define `MMCSD_CSD_10_TRANS_SPEED_SLICE` `MMCSD_CSD_20_TRANS_SPEED_SLICE`
- #define `MMCSD_CSD_10_NSAC_SLICE` `MMCSD_CSD_20_NSAC_SLICE`
- #define `MMCSD_CSD_10_TAAC_SLICE` `MMCSD_CSD_20_TAAC_SLICE`
- #define `MMCSD_CSD_10_CSD_STRUCTURE_SLICE` `MMCSD_CSD_20_CSD_STRUCTURE_SLICE`

CID record offsets

- #define `MMCSD_CID_SDC_CRC_SLICE` 7U, 1U
- #define `MMCSD_CID_SDC_MDT_M_SLICE` 11U, 8U
- #define `MMCSD_CID_SDC_MDT_Y_SLICE` 19U, 12U
- #define `MMCSD_CID_SDC_PSN_SLICE` 55U, 24U
- #define `MMCSD_CID_SDC_PRV_M_SLICE` 59U, 56U
- #define `MMCSD_CID_SDC_PRV_N_SLICE` 63U, 60U
- #define `MMCSD_CID_SDC_PNM0_SLICE` 71U, 64U
- #define `MMCSD_CID_SDC_PNM1_SLICE` 79U, 72U
- #define `MMCSD_CID_SDC_PNM2_SLICE` 87U, 80U
- #define `MMCSD_CID_SDC_PNM3_SLICE` 95U, 88U
- #define `MMCSD_CID_SDC_PNM4_SLICE` 103U, 96U
- #define `MMCSD_CID_SDC_OID_SLICE` 119U, 104U
- #define `MMCSD_CID_SDC_MID_SLICE` 127U, 120U
- #define `MMCSD_CID_MMC_CRC_SLICE` 7U, 1U
- #define `MMCSD_CID_MMC_MDT_Y_SLICE` 11U, 8U
- #define `MMCSD_CID_MMC_MDT_M_SLICE` 15U, 12U
- #define `MMCSD_CID_MMC_PSN_SLICE` 47U, 16U
- #define `MMCSD_CID_MMC_PRV_M_SLICE` 51U, 48U

- #define [MMCSD_CID_MMC_PRV_N_SLICE](#) 55U, 52U
- #define [MMCSD_CID_MMC_PNM0_SLICE](#) 63U, 56U
- #define [MMCSD_CID_MMC_PNM1_SLICE](#) 71U, 64U
- #define [MMCSD_CID_MMC_PNM2_SLICE](#) 79U, 72U
- #define [MMCSD_CID_MMC_PNM3_SLICE](#) 87U, 80U
- #define [MMCSD_CID_MMC_PNM4_SLICE](#) 95U, 88U
- #define [MMCSD_CID_MMC_PNM5_SLICE](#) 103U, 96U
- #define [MMCSD_CID_MMC_OID_SLICE](#) 119U, 104U
- #define [MMCSD_CID_MMC_MID_SLICE](#) 127U, 120U

R1 response utilities

- #define [MMCSD_R1_ERROR](#)(r1)
Evaluates to `true` if the R1 response contains error flags.
- #define [MMCSD_R1_STS](#)(r1)
Returns the status field of an R1 response.
- #define [MMCSD_R1_IS_CARD_LOCKED](#)(r1)
Evaluates to `true` if the R1 response indicates a locked card.

Macro Functions

- #define [mmcscdGetCardCapacity](#)(ip)
Returns the card capacity in blocks.

Data Structures

- struct [MMCSDBlockDeviceVMT](#)
[MMCSDBlockDevice](#) virtual methods table.
- struct [MMCSDBlockDevice](#)
MCC/SD block device class.
- struct [unpacked_sdc_cid_t](#)
Unpacked CID register from SDC.
- struct [unpacked_mmc_cid_t](#)
Unpacked CID register from MMC.
- struct [unpacked_sdc_csd_10_t](#)
Unpacked CSD v1.0 register from SDC.
- struct [unpacked_sdc_csd_20_t](#)
Unpacked CSD v2.0 register from SDC.
- struct [unpacked_mmc_csd_t](#)
Unpacked CSD register from MMC.

Macros

- #define [MMCSD_BLOCK_SIZE](#) 512U
Fixed block size for MMC/SD block devices.
- #define [MMCSD_R1_ERROR_MASK](#) 0xFDFFE008U
Mask of error bits in R1 responses.
- #define [MMCSD_CMD8_PATTERN](#) 0x000001AAU
Fixed pattern for CMD8.
- #define [_mmcscd_block_device_methods](#) [_base_block_device_methods](#)
[MMCSDBlockDevice](#) specific methods.
- #define [_mmcscd_block_device_data](#)
[MMCSDBlockDevice](#) specific data.

Functions

- `uint32_t _mmcscd_get_slice` (const `uint32_t` *data, `uint32_t` end, `uint32_t` start)
Gets a bit field from a words array.
- `uint32_t _mmcscd_get_capacity` (const `uint32_t` *csd)
Extract card capacity from a CSD.
- `uint32_t _mmcscd_get_capacity_ext` (const `uint8_t` *ext_csd)
Extract MMC card capacity from EXT_CSD.
- `void _mmcscd_unpack_sdc_cid` (const `MMCSDBlockDevice` *sdcp, `unpacked_sdc_cid_t` *cid_sdc)
Unpacks SDC CID array in structure.
- `void _mmcscd_unpack_mmc_cid` (const `MMCSDBlockDevice` *sdcp, `unpacked_mmc_cid_t` *cid_mmc)
Unpacks MMC CID array in structure.
- `void _mmcscd_unpack_csd_mmc` (const `MMCSDBlockDevice` *sdcp, `unpacked_mmc_csd_t` *csd_mmc)
Unpacks MMC CSD array in structure.
- `void _mmcscd_unpack_csd_v10` (const `MMCSDBlockDevice` *sdcp, `unpacked_sdc_csd_10_t` *csd10)
Unpacks SDC CSD v1.0 array in structure.
- `void _mmcscd_unpack_csd_v20` (const `MMCSDBlockDevice` *sdcp, `unpacked_sdc_csd_20_t` *csd20)
Unpacks SDC CSD v2.0 array in structure.

7.2.12.3.2 Macro Definition Documentation

7.2.12.3.2.1 MMCSD_BLOCK_SIZE

```
#define MMCSD_BLOCK_SIZE 512U
```

Fixed block size for MMC/SD block devices.

7.2.12.3.2.2 MMCSD_R1_ERROR_MASK

```
#define MMCSD_R1_ERROR_MASK 0xFDFFE008U
```

Mask of error bits in R1 responses.

7.2.12.3.2.3 MMCSD_CMD8_PATTERN

```
#define MMCSD_CMD8_PATTERN 0x000001AAU
```

Fixed pattern for CMD8.

7.2.12.3.2.4 MMCSD_STS_IDLE

```
#define MMCSD_STS_IDLE 0U
```

7.2.12.3.2.5 MMCSD_STS_READY

```
#define MMCSD_STS_READY 1U
```

7.2.12.3.2.6 MMCSD_STS_IDENT

```
#define MMCSD_STS_IDENT 2U
```

7.2.12.3.2.7 MMCSD_STS_STBY

```
#define MMCSD_STS_STBY 3U
```

7.2.12.3.2.8 MMCSD_STS_TRAN

```
#define MMCSD_STS_TRAN 4U
```

7.2.12.3.2.9 MMCSD_STS_DATA

```
#define MMCSD_STS_DATA 5U
```

7.2.12.3.2.10 MMCSD_STS_RCV

```
#define MMCSD_STS_RCV 6U
```

7.2.12.3.2.11 MMCSD_STS_PRG

```
#define MMCSD_STS_PRG 7U
```

7.2.12.3.2.12 MMCSD_STS_DIS

```
#define MMCSD_STS_DIS 8U
```

7.2.12.3.2.13 MMCSD_CMD_GO_IDLE_STATE

```
#define MMCSD_CMD_GO_IDLE_STATE 0U
```

7.2.12.3.2.14 MMCSD_CMD_INIT

```
#define MMCSD_CMD_INIT 1U
```

7.2.12.3.2.15 MMCSD_CMD_ALL_SEND_CID

```
#define MMCSD_CMD_ALL_SEND_CID 2U
```

7.2.12.3.2.16 MMCSD_CMD_SEND_RELATIVE_ADDR

```
#define MMCSD_CMD_SEND_RELATIVE_ADDR 3U
```

7.2.12.3.2.17 MMCSD_CMD_SET_RELATIVE_ADDR

```
#define MMCSD_CMD_SET_RELATIVE_ADDR MMCSD_CMD_SEND_RELATIVE_ADDR
```

7.2.12.3.2.18 MMCSD_CMD_SET_BUS_WIDTH

```
#define MMCSD_CMD_SET_BUS_WIDTH 6U
```

7.2.12.3.2.19 MMCSD_CMD_SWITCH

```
#define MMCSD_CMD_SWITCH MMCSD_CMD_SET_BUS_WIDTH
```

7.2.12.3.2.20 MMCSD_CMD_SEL_DESEL_CARD

```
#define MMCSD_CMD_SEL_DESEL_CARD 7U
```

7.2.12.3.2.21 MMCSD_CMD_SEND_IF_COND

```
#define MMCSD_CMD_SEND_IF_COND 8U
```

7.2.12.3.2.22 MMCSD_CMD_SEND_EXT_CSD

```
#define MMCSD_CMD_SEND_EXT_CSD MMCSD_CMD_SEND_IF_COND
```

7.2.12.3.2.23 MMCSD_CMD_SEND_CSD

```
#define MMCSD_CMD_SEND_CSD 9U
```

7.2.12.3.2.24 MMCSD_CMD_SEND_CID

```
#define MMCSD_CMD_SEND_CID 10U
```

7.2.12.3.2.25 MMCSD_CMD_STOP_TRANSMISSION

```
#define MMCSD_CMD_STOP_TRANSMISSION 12U
```

7.2.12.3.2.26 MMCSD_CMD_SEND_STATUS

```
#define MMCSD_CMD_SEND_STATUS 13U
```

7.2.12.3.2.27 MMCSD_CMD_SET_BLOCKLEN

```
#define MMCSD_CMD_SET_BLOCKLEN 16U
```

7.2.12.3.2.28 MMCSD_CMD_READ_SINGLE_BLOCK

```
#define MMCSD_CMD_READ_SINGLE_BLOCK 17U
```

7.2.12.3.2.29 MMCSD_CMD_READ_MULTIPLE_BLOCK

```
#define MMCSD_CMD_READ_MULTIPLE_BLOCK 18U
```

7.2.12.3.2.30 MMCSD_CMD_SET_BLOCK_COUNT

```
#define MMCSD_CMD_SET_BLOCK_COUNT 23U
```

7.2.12.3.2.31 MMCSD_CMD_WRITE_BLOCK

```
#define MMCSD_CMD_WRITE_BLOCK 24U
```

7.2.12.3.2.32 MMCSD_CMD_WRITE_MULTIPLE_BLOCK

```
#define MMCSD_CMD_WRITE_MULTIPLE_BLOCK 25U
```

7.2.12.3.2.33 MMCSD_CMD_ERASE_RW_BLK_START

```
#define MMCSD_CMD_ERASE_RW_BLK_START 32U
```

7.2.12.3.2.34 MMCSD_CMD_ERASE_RW_BLK_END

```
#define MMCSD_CMD_ERASE_RW_BLK_END 33U
```

7.2.12.3.2.35 MMCSD_CMD_ERASE

```
#define MMCSD_CMD_ERASE 38U
```

7.2.12.3.2.36 MMCSD_CMD_APP_OP_COND

```
#define MMCSD_CMD_APP_OP_COND 41U
```

7.2.12.3.2.37 MMCSD_CMD_LOCK_UNLOCK

```
#define MMCSD_CMD_LOCK_UNLOCK 42U
```

7.2.12.3.2.38 MMCSD_CMD_APP_CMD

```
#define MMCSD_CMD_APP_CMD 55U
```

7.2.12.3.2.39 MMCSD_CMD_READ_OCR

```
#define MMCSD_CMD_READ_OCR 58U
```

7.2.12.3.2.40 MMCSD_CSD_MMC_CSD_STRUCTURE_SLICE

```
#define MMCSD_CSD_MMC_CSD_STRUCTURE_SLICE 127U, 126U
```

7.2.12.3.2.41 MMCSD_CSD_MMC_SPEC_VERS_SLICE

```
#define MMCSD_CSD_MMC_SPEC_VERS_SLICE 125U, 122U
```

7.2.12.3.2.42 MMCSD_CSD_MMC_TAAC_SLICE

```
#define MMCSD_CSD_MMC_TAAC_SLICE 119U, 112U
```

7.2.12.3.2.43 MMCSD_CSD_MMC_NSAC_SLICE

```
#define MMCSD_CSD_MMC_NSAC_SLICE 111U, 104U
```

7.2.12.3.2.44 MMCSD_CSD_MMC_TRAN_SPEED_SLICE

```
#define MMCSD_CSD_MMC_TRAN_SPEED_SLICE 103U, 96U
```

7.2.12.3.2.45 MMCSD_CSD_MMC_CCC_SLICE

```
#define MMCSD_CSD_MMC_CCC_SLICE 95U, 84U
```

7.2.12.3.2.46 MMCSD_CSD_MMC_READ_BL_LEN_SLICE

```
#define MMCSD_CSD_MMC_READ_BL_LEN_SLICE 83U, 80U
```

7.2.12.3.2.47 MMCSD_CSD_MMC_READ_BL_PARTIAL_SLICE

```
#define MMCSD_CSD_MMC_READ_BL_PARTIAL_SLICE 79U, 79U
```

7.2.12.3.2.48 MMCSD_CSD_MMC_WRITE_BLK_MISALIGN_SLICE

```
#define MMCSD_CSD_MMC_WRITE_BLK_MISALIGN_SLICE 78U, 78U
```

7.2.12.3.2.49 MMCSD_CSD_MMC_READ_BLK_MISALIGN_SLICE

```
#define MMCSD_CSD_MMC_READ_BLK_MISALIGN_SLICE 77U, 77U
```

7.2.12.3.2.50 MMCSD_CSD_MMC_DSR_IMP_SLICE

```
#define MMCSD_CSD_MMC_DSR_IMP_SLICE 76U, 76U
```

7.2.12.3.2.51 MMCSD_CSD_MMC_C_SIZE_SLICE

```
#define MMCSD_CSD_MMC_C_SIZE_SLICE 73U, 62U
```

7.2.12.3.2.52 MMCSD_CSD_MMC_VDD_R_CURR_MIN_SLICE

```
#define MMCSD_CSD_MMC_VDD_R_CURR_MIN_SLICE 61U, 59U
```

7.2.12.3.2.53 MMCSD_CSD_MMC_VDD_R_CURR_MAX_SLICE

```
#define MMCSD_CSD_MMC_VDD_R_CURR_MAX_SLICE 58U, 56U
```

7.2.12.3.2.54 MMCSD_CSD_MMC_VDD_W_CURR_MIN_SLICE

```
#define MMCSD_CSD_MMC_VDD_W_CURR_MIN_SLICE 55U, 53U
```

7.2.12.3.2.55 MMCSD_CSD_MMC_VDD_W_CURR_MAX_SLICE

```
#define MMCSD_CSD_MMC_VDD_W_CURR_MAX_SLICE 52U, 50U
```

7.2.12.3.2.56 MMCSD_CSD_MMC_C_SIZE_MULT_SLICE

```
#define MMCSD_CSD_MMC_C_SIZE_MULT_SLICE 49U, 47U
```

7.2.12.3.2.57 MMCSD_CSD_MMC_ERASE_GRP_SIZE_SLICE

```
#define MMCSD_CSD_MMC_ERASE_GRP_SIZE_SLICE 46U, 42U
```

7.2.12.3.2.58 MMCSD_CSD_MMC_ERASE_GRP_MULT_SLICE

```
#define MMCSD_CSD_MMC_ERASE_GRP_MULT_SLICE 41U, 37U
```

7.2.12.3.2.59 MMCSD_CSD_MMC_WP_GRP_SIZE_SLICE

```
#define MMCSD_CSD_MMC_WP_GRP_SIZE_SLICE 36U, 32U
```

7.2.12.3.2.60 MMCSD_CSD_MMC_WP_GRP_ENABLE_SLICE

```
#define MMCSD_CSD_MMC_WP_GRP_ENABLE_SLICE 31U, 31U
```

7.2.12.3.2.61 MMCSD_CSD_MMC_DEFAULT_ECC_SLICE

```
#define MMCSD_CSD_MMC_DEFAULT_ECC_SLICE 30U, 29U
```

7.2.12.3.2.62 MMCSD_CSD_MMC_R2W_FACTOR_SLICE

```
#define MMCSD_CSD_MMC_R2W_FACTOR_SLICE 28U, 26U
```

7.2.12.3.2.63 MMCSD_CSD_MMC_WRITE_BL_LEN_SLICE

```
#define MMCSD_CSD_MMC_WRITE_BL_LEN_SLICE 25U, 22U
```

7.2.12.3.2.64 MMCSD_CSD_MMC_WRITE_BL_PARTIAL_SLICE

```
#define MMCSD_CSD_MMC_WRITE_BL_PARTIAL_SLICE 21U, 21U
```

7.2.12.3.2.65 MMCSD_CSD_MMC_CONTENT_PROT_APP_SLICE

```
#define MMCSD_CSD_MMC_CONTENT_PROT_APP_SLICE 16U, 16U
```

7.2.12.3.2.66 MMCSD_CSD_MMC_FILE_FORMAT_GRP_SLICE

```
#define MMCSD_CSD_MMC_FILE_FORMAT_GRP_SLICE 15U, 15U
```

7.2.12.3.2.67 MMCSD_CSD_MMC_COPY_SLICE

```
#define MMCSD_CSD_MMC_COPY_SLICE 14U, 14U
```

7.2.12.3.2.68 MMCSD_CSD_MMC_PERM_WRITE_PROTECT_SLICE

```
#define MMCSD_CSD_MMC_PERM_WRITE_PROTECT_SLICE 13U, 13U
```

7.2.12.3.2.69 MMCSD_CSD_MMC_TMP_WRITE_PROTECT_SLICE

```
#define MMCSD_CSD_MMC_TMP_WRITE_PROTECT_SLICE 12U, 12U
```

7.2.12.3.2.70 MMCSD_CSD_MMC_FILE_FORMAT_SLICE

```
#define MMCSD_CSD_MMC_FILE_FORMAT_SLICE 11U, 10U
```

7.2.12.3.2.71 MMCSD_CSD_MMC_ECC_SLICE

```
#define MMCSD_CSD_MMC_ECC_SLICE 9U, 8U
```

7.2.12.3.2.72 MMCSD_CSD_MMC_CRC_SLICE

```
#define MMCSD_CSD_MMC_CRC_SLICE 7U, 1U
```

7.2.12.3.2.73 MMCSD_CSD_20_CRC_SLICE

```
#define MMCSD_CSD_20_CRC_SLICE 7U, 1U
```

7.2.12.3.2.74 MMCSD_CSD_20_FILE_FORMAT_SLICE

```
#define MMCSD_CSD_20_FILE_FORMAT_SLICE 11U, 10U
```

7.2.12.3.2.75 MMCSD_CSD_20_TMP_WRITE_PROTECT_SLICE

```
#define MMCSD_CSD_20_TMP_WRITE_PROTECT_SLICE 12U, 12U
```


7.2.12.3.2.76 MMCSD_CSD_20_PERM_WRITE_PROTECT_SLICE

```
#define MMCSD_CSD_20_PERM_WRITE_PROTECT_SLICE 13U, 13U
```

7.2.12.3.2.77 MMCSD_CSD_20_COPY_SLICE

```
#define MMCSD_CSD_20_COPY_SLICE 14U, 14U
```

7.2.12.3.2.78 MMCSD_CSD_20_FILE_FORMAT_GRP_SLICE

```
#define MMCSD_CSD_20_FILE_FORMAT_GRP_SLICE 15U, 15U
```

7.2.12.3.2.79 MMCSD_CSD_20_WRITE_BL_PARTIAL_SLICE

```
#define MMCSD_CSD_20_WRITE_BL_PARTIAL_SLICE 21U, 21U
```

7.2.12.3.2.80 MMCSD_CSD_20_WRITE_BL_LEN_SLICE

```
#define MMCSD_CSD_20_WRITE_BL_LEN_SLICE 25U, 12U
```

7.2.12.3.2.81 MMCSD_CSD_20_R2W_FACTOR_SLICE

```
#define MMCSD_CSD_20_R2W_FACTOR_SLICE 28U, 26U
```

7.2.12.3.2.82 MMCSD_CSD_20_WP_GRP_ENABLE_SLICE

```
#define MMCSD_CSD_20_WP_GRP_ENABLE_SLICE 31U, 31U
```

7.2.12.3.2.83 MMCSD_CSD_20_WP_GRP_SIZE_SLICE

```
#define MMCSD_CSD_20_WP_GRP_SIZE_SLICE 38U, 32U
```

7.2.12.3.2.84 MMCSD_CSD_20_ERASE_SECTOR_SIZE_SLICE

```
#define MMCSD_CSD_20_ERASE_SECTOR_SIZE_SLICE 45U, 39U
```

7.2.12.3.2.85 MMCSD_CSD_20_ERASE_BLK_EN_SLICE

```
#define MMCSD_CSD_20_ERASE_BLK_EN_SLICE 46U, 46U
```

7.2.12.3.2.86 MMCSD_CSD_20_C_SIZE_SLICE

```
#define MMCSD_CSD_20_C_SIZE_SLICE 69U, 48U
```

7.2.12.3.2.87 MMCSD_CSD_20_DSR_IMP_SLICE

```
#define MMCSD_CSD_20_DSR_IMP_SLICE 76U, 76U
```

7.2.12.3.2.88 MMCSD_CSD_20_READ_BLK_MISALIGN_SLICE

```
#define MMCSD_CSD_20_READ_BLK_MISALIGN_SLICE 77U, 77U
```

7.2.12.3.2.89 MMCSD_CSD_20_WRITE_BLK_MISALIGN_SLICE

```
#define MMCSD_CSD_20_WRITE_BLK_MISALIGN_SLICE 78U, 78U
```

7.2.12.3.2.90 MMCSD_CSD_20_READ_BL_PARTIAL_SLICE

```
#define MMCSD_CSD_20_READ_BL_PARTIAL_SLICE 79U, 79U
```

7.2.12.3.2.91 MMCSD_CSD_20_READ_BL_LEN_SLICE

```
#define MMCSD_CSD_20_READ_BL_LEN_SLICE 83U, 80U
```

7.2.12.3.2.92 MMCSD_CSD_20_CCC_SLICE

```
#define MMCSD_CSD_20_CCC_SLICE 95U, 84U
```

7.2.12.3.2.93 MMCSD_CSD_20_TRANS_SPEED_SLICE

```
#define MMCSD_CSD_20_TRANS_SPEED_SLICE 103U, 96U
```

7.2.12.3.2.94 MMCSD_CSD_20_NSAC_SLICE

```
#define MMCSD_CSD_20_NSAC_SLICE 111U, 104U
```

7.2.12.3.2.95 MMCSD_CSD_20_TAAC_SLICE

```
#define MMCSD_CSD_20_TAAC_SLICE 119U, 112U
```

7.2.12.3.2.96 MMCSD_CSD_20_CSD_STRUCTURE_SLICE

```
#define MMCSD_CSD_20_CSD_STRUCTURE_SLICE 127U, 126U
```

7.2.12.3.2.97 MMCSD_CSD_10_CRC_SLICE

```
#define MMCSD_CSD_10_CRC_SLICE MMCSD_CSD_20_CRC_SLICE
```

7.2.12.3.2.98 MMCSD_CSD_10_FILE_FORMAT_SLICE

```
#define MMCSD_CSD_10_FILE_FORMAT_SLICE MMCSD_CSD_20_FILE_FORMAT_SLICE
```

7.2.12.3.2.99 MMCSD_CSD_10_TMP_WRITE_PROTECT_SLICE

```
#define MMCSD_CSD_10_TMP_WRITE_PROTECT_SLICE MMCSD_CSD_20_TMP_WRITE_PROTECT_SLICE
```

7.2.12.3.2.100 MMCSD_CSD_10_PERM_WRITE_PROTECT_SLICE

```
#define MMCSD_CSD_10_PERM_WRITE_PROTECT_SLICE MMCSD_CSD_20_PERM_WRITE_PROTECT_SLICE
```

7.2.12.3.2.101 MMCSD_CSD_10_COPY_SLICE

```
#define MMCSD_CSD_10_COPY_SLICE MMCSD_CSD_20_COPY_SLICE
```

7.2.12.3.2.102 MMCSD_CSD_10_FILE_FORMAT_GRP_SLICE

```
#define MMCSD_CSD_10_FILE_FORMAT_GRP_SLICE MMCSD_CSD_20_FILE_FORMAT_GRP_SLICE
```

7.2.12.3.2.103 MMCSD_CSD_10_WRITE_BL_PARTIAL_SLICE

```
#define MMCSD_CSD_10_WRITE_BL_PARTIAL_SLICE MMCSD_CSD_20_WRITE_BL_PARTIAL_SLICE
```

7.2.12.3.2.104 MMCSD_CSD_10_WRITE_BL_LEN_SLICE

```
#define MMCSD_CSD_10_WRITE_BL_LEN_SLICE MMCSD_CSD_20_WRITE_BL_LEN_SLICE
```

7.2.12.3.2.105 MMCSD_CSD_10_R2W_FACTOR_SLICE

```
#define MMCSD_CSD_10_R2W_FACTOR_SLICE MMCSD_CSD_20_R2W_FACTOR_SLICE
```

7.2.12.3.2.106 MMCSD_CSD_10_WP_GRP_ENABLE_SLICE

```
#define MMCSD_CSD_10_WP_GRP_ENABLE_SLICE MMCSD_CSD_20_WP_GRP_ENABLE_SLICE
```

7.2.12.3.2.107 MMCSD_CSD_10_WP_GRP_SIZE_SLICE

```
#define MMCSD_CSD_10_WP_GRP_SIZE_SLICE MMCSD_CSD_20_WP_GRP_SIZE_SLICE
```

7.2.12.3.2.108 MMCSD_CSD_10_ERASE_SECTOR_SIZE_SLICE

```
#define MMCSD_CSD_10_ERASE_SECTOR_SIZE_SLICE MMCSD_CSD_20_ERASE_SECTOR_SIZE_SLICE
```

7.2.12.3.2.109 MMCSD_CSD_10_ERASE_BLK_EN_SLICE

```
#define MMCSD_CSD_10_ERASE_BLK_EN_SLICE MMCSD_CSD_20_ERASE_BLK_EN_SLICE
```

7.2.12.3.2.110 MMCSD_CSD_10_C_SIZE_MULT_SLICE

```
#define MMCSD_CSD_10_C_SIZE_MULT_SLICE 49U, 47U
```

7.2.12.3.2.111 MMCSD_CSD_10_VDD_W_CURR_MAX_SLICE

```
#define MMCSD_CSD_10_VDD_W_CURR_MAX_SLICE 52U, 50U
```

7.2.12.3.2.112 MMCSD_CSD_10_VDD_W_CURR_MIN_SLICE

```
#define MMCSD_CSD_10_VDD_W_CURR_MIN_SLICE 55U, 53U
```

7.2.12.3.2.113 MMCSD_CSD_10_VDD_R_CURR_MAX_SLICE

```
#define MMCSD_CSD_10_VDD_R_CURR_MAX_SLICE 58U, 56U
```

7.2.12.3.2.114 MMCSD_CSD_10_VDD_R_CURR_MIX_SLICE

```
#define MMCSD_CSD_10_VDD_R_CURR_MIX_SLICE 61U, 59U
```

7.2.12.3.2.115 MMCSD_CSD_10_C_SIZE_SLICE

```
#define MMCSD_CSD_10_C_SIZE_SLICE 73U, 62U
```

7.2.12.3.2.116 MMCSD_CSD_10_DSR_IMP_SLICE

```
#define MMCSD_CSD_10_DSR_IMP_SLICE MMCSD_CSD_20_DSR_IMP_SLICE
```

7.2.12.3.2.117 MMCSD_CSD_10_READ_BLK_MISALIGN_SLICE

```
#define MMCSD_CSD_10_READ_BLK_MISALIGN_SLICE MMCSD_CSD_20_READ_BLK_MISALIGN_SLICE
```

7.2.12.3.2.118 MMCSD_CSD_10_WRITE_BLK_MISALIGN_SLICE

```
#define MMCSD_CSD_10_WRITE_BLK_MISALIGN_SLICE MMCSD_CSD_20_WRITE_BLK_MISALIGN_SLICE
```

7.2.12.3.2.119 MMCSD_CSD_10_READ_BL_PARTIAL_SLICE

```
#define MMCSD_CSD_10_READ_BL_PARTIAL_SLICE MMCSD_CSD_20_READ_BL_PARTIAL_SLICE
```

7.2.12.3.2.120 MMCSD_CSD_10_READ_BL_LEN_SLICE

```
#define MMCSD_CSD_10_READ_BL_LEN_SLICE 83U, 80U
```

7.2.12.3.2.121 MMCSD_CSD_10_CCC_SLICE

```
#define MMCSD_CSD_10_CCC_SLICE MMCSD_CSD_20_CCC_SLICE
```

7.2.12.3.2.122 MMCSD_CSD_10_TRANS_SPEED_SLICE

```
#define MMCSD_CSD_10_TRANS_SPEED_SLICE MMCSD_CSD_20_TRANS_SPEED_SLICE
```

7.2.12.3.2.123 MMCSD_CSD_10_NSAC_SLICE

```
#define MMCSD_CSD_10_NSAC_SLICE MMCSD_CSD_20_NSAC_SLICE
```

7.2.12.3.2.124 MMCSD_CSD_10_TAAC_SLICE

```
#define MMCSD_CSD_10_TAAC_SLICE MMCSD_CSD_20_TAAC_SLICE
```

7.2.12.3.2.125 MMCSD_CSD_10_CSD_STRUCTURE_SLICE

```
#define MMCSD_CSD_10_CSD_STRUCTURE_SLICE MMCSD_CSD_20_CSD_STRUCTURE_SLICE
```

7.2.12.3.2.126 MMCSD_CID_SDC_CRC_SLICE

```
#define MMCSD_CID_SDC_CRC_SLICE 7U, 1U
```

7.2.12.3.2.127 MMCSD_CID_SDC_MDT_M_SLICE

```
#define MMCSD_CID_SDC_MDT_M_SLICE 11U, 8U
```

7.2.12.3.2.128 MMCSD_CID_SDC_MDT_Y_SLICE

```
#define MMCSD_CID_SDC_MDT_Y_SLICE 19U, 12U
```

7.2.12.3.2.129 MMCSD_CID_SDC_PSN_SLICE

```
#define MMCSD_CID_SDC_PSN_SLICE 55U, 24U
```

7.2.12.3.2.130 MMCSD_CID_SDC_PRV_M_SLICE

```
#define MMCSD_CID_SDC_PRV_M_SLICE 59U, 56U
```

7.2.12.3.2.131 MMCSD_CID_SDC_PRV_N_SLICE

```
#define MMCSD_CID_SDC_PRV_N_SLICE 63U, 60U
```

7.2.12.3.2.132 MMCSD_CID_SDC_PNM0_SLICE

```
#define MMCSD_CID_SDC_PNM0_SLICE 71U, 64U
```

7.2.12.3.2.133 MMCSD_CID_SDC_PNM1_SLICE

```
#define MMCSD_CID_SDC_PNM1_SLICE 79U, 72U
```

7.2.12.3.2.134 MMCSD_CID_SDC_PNM2_SLICE

```
#define MMCSD_CID_SDC_PNM2_SLICE 87U, 80U
```

7.2.12.3.2.135 MMCSD_CID_SDC_PNM3_SLICE

```
#define MMCSD_CID_SDC_PNM3_SLICE 95U, 88U
```

7.2.12.3.2.136 MMCSD_CID_SDC_PNM4_SLICE

```
#define MMCSD_CID_SDC_PNM4_SLICE 103U, 96U
```

7.2.12.3.2.137 MMCSD_CID_SDC_OID_SLICE

```
#define MMCSD_CID_SDC_OID_SLICE 119U, 104U
```

7.2.12.3.2.138 MMCSD_CID_SDC_MID_SLICE

```
#define MMCSD_CID_SDC_MID_SLICE 127U, 120U
```

7.2.12.3.2.139 MMCSD_CID_MMC_CRC_SLICE

```
#define MMCSD_CID_MMC_CRC_SLICE 7U, 1U
```

7.2.12.3.2.140 MMCSD_CID_MMC_MDT_Y_SLICE

```
#define MMCSD_CID_MMC_MDT_Y_SLICE 11U, 8U
```

7.2.12.3.2.141 MMCSD_CID_MMC_MDT_M_SLICE

```
#define MMCSD_CID_MMC_MDT_M_SLICE 15U, 12U
```

7.2.12.3.2.142 MMCSD_CID_MMC_PSN_SLICE

```
#define MMCSD_CID_MMC_PSN_SLICE 47U, 16U
```

7.2.12.3.2.143 MMCSD_CID_MMC_PRV_M_SLICE

```
#define MMCSD_CID_MMC_PRV_M_SLICE 51U, 48U
```

7.2.12.3.2.144 MMCSD_CID_MMC_PRV_N_SLICE

```
#define MMCSD_CID_MMC_PRV_N_SLICE 55U, 52U
```

7.2.12.3.2.145 MMCSD_CID_MMC_PNM0_SLICE

```
#define MMCSD_CID_MMC_PNM0_SLICE 63U, 56U
```

7.2.12.3.2.146 MMCSD_CID_MMC_PNM1_SLICE

```
#define MMCSD_CID_MMC_PNM1_SLICE 71U, 64U
```

7.2.12.3.2.147 MMCSD_CID_MMC_PNM2_SLICE

```
#define MMCSD_CID_MMC_PNM2_SLICE 79U, 72U
```

7.2.12.3.2.148 MMCSD_CID_MMC_PNM3_SLICE

```
#define MMCSD_CID_MMC_PNM3_SLICE 87U, 80U
```

7.2.12.3.2.149 MMCSD_CID_MMC_PNM4_SLICE

```
#define MMCSD_CID_MMC_PNM4_SLICE 95U, 88U
```

7.2.12.3.2.150 MMCSD_CID_MMC_PNM5_SLICE

```
#define MMCSD_CID_MMC_PNM5_SLICE 103U, 96U
```

7.2.12.3.2.151 MMCSD_CID_MMC_OID_SLICE

```
#define MMCSD_CID_MMC_OID_SLICE 119U, 104U
```

7.2.12.3.2.152 MMCSD_CID_MMC_MID_SLICE

```
#define MMCSD_CID_MMC_MID_SLICE 127U, 120U
```

7.2.12.3.2.153 _mmcscd_block_device_methods

```
#define _mmcscd_block_device_methods _base_block_device_methods
```

[MMCSDBlockDevice](#) specific methods.

7.2.12.3.2.154 _mmcscd_block_device_data

```
#define _mmcscd_block_device_data
```

Value:

```
\_base\_block\_device\_data
/* Card CID.*/
uint32_t          cid[4];
/* Card CSD.*/
uint32_t          csd[4];
/* Total number of blocks in card.*/
uint32_t          capacity;
```

[MMCSDBlockDevice](#) specific data.

Note

It is empty because [MMCSDBlockDevice](#) is only an interface without implementation.

7.2.12.3.2.155 MMCSD_R1_ERROR

```
#define MMCSD_R1_ERROR(  
    r1)
```

Value:

```
((r1) & MMCSD_R1_ERROR_MASK) != 0U)
```

Evaluates to `true` if the R1 response contains error flags.

Parameters

in	<i>r1</i>	the r1 response

7.2.12.3.2.156 MMCSD_R1_STS

```
#define MMCSD_R1_STS(  
    r1)
```

Value:

```
((r1) » 9U) & 15U)
```

Returns the status field of an R1 response.

Parameters

in	<i>r1</i>	the r1 response

7.2.12.3.2.157 MMCSD_R1_IS_CARD_LOCKED

```
#define MMCSD_R1_IS_CARD_LOCKED(  
    r1)
```

Value:

```
((r1) » 21U) & 1U) != 0U)
```

Evaluates to `true` if the R1 response indicates a locked card.

Parameters

in	<i>r1</i>	the r1 response

7.2.12.3.2.158 mmcscGetCardCapacity

```
#define mmcscGetCardCapacity(  
    ip)
```

Value:

```
((ip)->capacity)
```

Returns the card capacity in blocks.

Parameters

in	<i>ip</i>	pointer to a MMCSDBlockDevice or derived class

Returns

The card capacity.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.12.3.3 Function Documentation**7.2.12.3.3.1 _mmcscd_get_slice()**

```
uint32_t _mmcscd_get_slice (  
    const uint32_t * data,  
    uint32_t end,  
    uint32_t start)
```

Gets a bit field from a words array.

Note

The bit zero is the LSb of the first word.

Parameters

in	<i>data</i>	pointer to the words array
in	<i>end</i>	bit offset of the last bit of the field, inclusive
in	<i>start</i>	bit offset of the first bit of the field, inclusive

Returns

The bits field value, left aligned.

Function Class:

Not an API, this function is for internal use only.

7.2.12.3.3.2 _mmcscd_get_capacity()

```
uint32_t _mmcscd_get_capacity (  
    const uint32_t * csd)
```

Extract card capacity from a CSD.

The capacity is returned as number of available blocks.

Parameters

in	<i>csd</i>	the CSD record

Returns

The card capacity.

Return values

0	CSD format error

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.12.3.3.3 _mmcscd_get_capacity_ext()**

```
uint32_t _mmcscd_get_capacity_ext (
    const uint8_t * ext_csd)
```

Extract MMC card capacity from EXT_CSD.

The capacity is returned as number of available blocks.

Parameters

in	<i>ext_csd</i>	the extended CSD record

Returns

The card capacity.

Function Class:

Not an API, this function is for internal use only.

7.2.12.3.3.4 _mmcscd_unpack_sdc_cid()

```
void _mmcscd_unpack_sdc_cid (
    const MMCSCDBlockDevice * sdc,
    unpacked_sdc_cid_t * cid_sdc)
```

Unpacks SDC CID array in structure.

Parameters

in	<i>sdcp</i>	pointer to the MMCSDBlockDevice object
out	<i>cidsdc</i>	pointer to the unpacked_sdc_cid_t object

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.12.3.3.5 _mmcsd_unpack_mmc_cid()**

```
void _mmcsd_unpack_mmc_cid (  
    const MMCSDBlockDevice * sdcp,  
    unpacked\_mmc\_cid\_t * cidmmc)
```

Unpacks MMC CID array in structure.

Parameters

in	<i>sdcp</i>	pointer to the MMCSDBlockDevice object
out	<i>cidmmc</i>	pointer to the unpacked_mmc_cid_t object

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.12.3.3.6 _mmcsd_unpack_csd_mmc()**

```
void _mmcsd_unpack_csd_mmc (  
    const MMCSDBlockDevice * sdcp,  
    unpacked\_mmc\_csd\_t * csdmmc)
```

Unpacks MMC CSD array in structure.

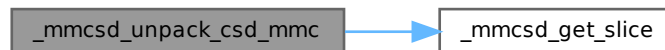
Parameters

in	<i>sdcp</i>	pointer to the MMCSDBlockDevice object
out	<i>csdmmc</i>	pointer to the unpacked_mmc_csd_t object

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.12.3.3.7 _mmcscd_unpack_csd_v10()**

```
void _mmcscd_unpack_csd_v10 (  
    const MMCSDBlockDevice * sdcp,  
    unpacked\_sdc\_csd\_10\_t * csd10)
```

Unpacks SDC CSD v1.0 array in structure.

Parameters

in	<i>sdcp</i>	pointer to the MMCSDBlockDevice object
out	<i>csd10</i>	pointer to the unpacked_sdc_csd_10_t object

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:

**7.2.12.3.3.8 _mmcscd_unpack_csd_v20()**

```
void _mmcscd_unpack_csd_v20 (  
    const MMCSDBlockDevice * sdcp,  
    unpacked\_sdc\_csd\_20\_t * csd20)
```

Unpacks SDC CSD v2.0 array in structure.

Parameters

in	<i>sdcp</i>	pointer to the MMCSDBlockDevice object
out	<i>csd20</i>	pointer to the unpacked_sdc_csd_20_t object

Function Class:

Not an API, this function is for internal use only.

Here is the call graph for this function:



7.2.12.4 I/O Bytes Queues

7.2.12.4.1 Detailed Description

Queues are mostly used in serial-like device drivers. Serial device drivers are usually designed to have a lower side (lower driver, it is usually an interrupt service routine) and an upper side (upper driver, accessed by the application threads).

There are several kind of queues:

- **Input queue**, unidirectional queue where the writer is the lower side and the reader is the upper side.
- **Output queue**, unidirectional queue where the writer is the upper side and the reader is the lower side.
- **Full duplex queue**, bidirectional queue. Full duplex queues are implemented by pairing an input queue and an output queue together.

Queue functions returned status value

- `#define Q_OK MSG_OK`
Operation successful.
- `#define Q_TIMEOUT MSG_TIMEOUT`
Timeout condition.
- `#define Q_RESET MSG_RESET`
Queue has been reset.
- `#define Q_EMPTY MSG_TIMEOUT`
Queue empty.
- `#define Q_FULL MSG_TIMEOUT`
Queue full,.

Macro Functions

- `#define qSizeX(qp)`
Returns the queue's buffer size.
- `#define qSpacel(qp)`
Queue space.
- `#define qGetLink(qp)`
Returns the queue application-defined link.
- `#define qSetLink(qp, lk)`
Sets the queue application-defined link.
- `#define iqGetFullI(iqp)`
Returns the filled space into an input queue.
- `#define iqGetEmptyI(iqp)`
Returns the empty space into an input queue.
- `#define iqIsEmptyI(iqp)`
Evaluates to `true` if the specified input queue is empty.
- `#define iqIsFullI(iqp)`
Evaluates to `true` if the specified input queue is full.
- `#define iqGet(iqp)`
Input queue read.
- `#define oqGetFullI(oqp)`
Returns the filled space into an output queue.
- `#define oqGetEmptyI(oqp)`
Returns the empty space into an output queue.
- `#define oqIsEmptyI(oqp)`
Evaluates to `true` if the specified output queue is empty.
- `#define oqIsFullI(oqp)`
Evaluates to `true` if the specified output queue is full.
- `#define oqPut(oqp, b)`
Output queue write.

Data Structures

- `struct io_queue`
Generic I/O queue structure.

Typedefs

- `typedef struct io_queue io_queue_t`
Type of a generic I/O queue structure.
- `typedef void(* qnotify_t) (io_queue_t *qp)`
Queue notification callback type.
- `typedef io_queue_t input_queue_t`
Type of an input queue structure.
- `typedef io_queue_t output_queue_t`
Type of an output queue structure.

Functions

- static size_t [iq_read](#) ([input_queue_t](#) *iqp, uint8_t *bp, size_t n)
Non-blocking input queue read.
- static size_t [oq_write](#) ([output_queue_t](#) *oqp, const uint8_t *bp, size_t n)
Non-blocking output queue write.
- void [iqObjectInit](#) ([input_queue_t](#) *iqp, uint8_t *bp, size_t size, [qnotify_t](#) infy, void *link)
Initializes an input queue.
- void [iqResetI](#) ([input_queue_t](#) *iqp)
Resets an input queue.
- [msg_t](#) [iqPutI](#) ([input_queue_t](#) *iqp, uint8_t b)
Input queue write.
- [msg_t](#) [iqGetI](#) ([input_queue_t](#) *iqp)
Input queue non-blocking read.
- [msg_t](#) [iqGetTimeout](#) ([input_queue_t](#) *iqp, [sysinterval_t](#) timeout)
Input queue read with timeout.
- size_t [iqReadI](#) ([input_queue_t](#) *iqp, uint8_t *bp, size_t n)
Input queue non-blocking read.
- size_t [iqReadTimeout](#) ([input_queue_t](#) *iqp, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
Input queue read with timeout.
- void [oqObjectInit](#) ([output_queue_t](#) *oqp, uint8_t *bp, size_t size, [qnotify_t](#) onfy, void *link)
Initializes an output queue.
- void [oqResetI](#) ([output_queue_t](#) *oqp)
Resets an output queue.
- [msg_t](#) [oqPutI](#) ([output_queue_t](#) *oqp, uint8_t b)
Output queue non-blocking write.
- [msg_t](#) [oqPutTimeout](#) ([output_queue_t](#) *oqp, uint8_t b, [sysinterval_t](#) timeout)
Output queue write with timeout.
- [msg_t](#) [oqGetI](#) ([output_queue_t](#) *oqp)
Output queue read.
- size_t [oqWriteI](#) ([output_queue_t](#) *oqp, const uint8_t *bp, size_t n)
Output queue non-blocking write.
- size_t [oqWriteTimeout](#) ([output_queue_t](#) *oqp, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
Output queue write with timeout.

7.2.12.4.2 Macro Definition Documentation

7.2.12.4.2.1 Q_OK

```
#define Q_OK MSG_OK
```

Operation successful.

7.2.12.4.2.2 Q_TIMEOUT

```
#define Q_TIMEOUT MSG_TIMEOUT
```

Timeout condition.

7.2.12.4.2.3 Q_RESET

```
#define Q_RESET MSG_RESET
```

Queue has been reset.

7.2.12.4.2.4 Q_EMPTY

```
#define Q_EMPTY MSG_TIMEOUT
```

Queue empty.

7.2.12.4.2.5 Q_FULL

```
#define Q_FULL MSG_TIMEOUT
```

Queue full,.

7.2.12.4.2.6 qSizeX

```
#define qSizeX(  
    qp)
```

Value:

```
/*lint -save -e9033 [10.8] The cast is safe.*/  
((size_t)((qp)->q_top - (qp)->q_buffer))  
/*lint -restore*/
```

\
\\

Returns the queue's buffer size.

Parameters

in	qp	pointer to a io_queue_t structure

Returns

The buffer size.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.12.4.2.7 qSpacel

```
#define qSpaceI(  
    qp)
```

Value:

```
((qp)->q_counter)
```

Queue space.

Returns the used space if used on an input queue or the empty space if used on an output queue.

Parameters

in	<i>qp</i>	pointer to a io_queue_t structure

Returns

The buffer space.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.2.8 qGetLink

```
#define qGetLink(  
    qp)
```

Value:

```
((qp)->q_link)
```

Returns the queue application-defined link.

Note

This function can be called in any context.

Parameters

in	<i>qp</i>	pointer to a io_queue_t structure

Returns

The application-defined link.

Function Class:

Special function, this function has special requirements see the notes.

7.2.12.4.2.9 qSetLink

```
#define qSetLink(  
    qp,  
    lk)
```

Value:

```
((qp)->q_link = lk)
```

Sets the queue application-defined link.

Note

This function can be called in any context.

Parameters

in	<i>qp</i>	pointer to a io_queue_t structure
in	<i>lk</i>	The application-defined link.

Function Class:

Special function, this function has special requirements see the notes.

7.2.12.4.2.10 iqGetFullI

```
#define iqGetFullI(  
    iqp)
```

Value:

[qSpaceI](#)(*iqp*)

Returns the filled space into an input queue.

Parameters

in	<i>iqp</i>	pointer to an input_queue_t structure

Returns

The number of full bytes in the queue.

Return values

0	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.2.11 iqGetEmptyI

```
#define iqGetEmptyI(  
    iqp)
```

Value:

([qSizeX](#)(*iqp*) - [qSpaceI](#)(*iqp*))

Returns the empty space into an input queue.

Parameters

in	<i>iqp</i>	pointer to an input_queue_t structure

Returns

The number of empty bytes in the queue.

Return values

0	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.2.12 iqIsEmptyI

```
#define iqIsEmptyI(  
    iqp)
```

Value:

```
((bool) (qSpaceI(iqp) == 0U))
```

Evaluates to `true` if the specified input queue is empty.

Parameters

in	<i>iqp</i>	pointer to an input_queue_t structure

Returns

The queue status.

Return values

<i>false</i>	if the queue is not empty.
<i>true</i>	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.2.13 iqIsFullI

```
#define iqIsFullI(  
    iqp)
```

Value:

```
/*lint -save -e9007 [13.5] No side effects, a pointer is passed.*/ \
((bool) (((iqp)->q_wrptr == (iqp)->q_rdptr) && ((iqp)->q_counter != 0U))) \
/*lint -restore*/
```

Evaluates to `true` if the specified input queue is full.

Parameters

in	<i>iqp</i>	pointer to an <code>input_queue_t</code> structure

Returns

The queue status.

Return values

<i>false</i>	if the queue is not full.
<i>true</i>	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.2.14 iqGet

```
#define iqGet(  
    iqp)
```

Value:

```
iqGetTimeout(iqp, TIME_INFINITE)
```

Input queue read.

This function reads a byte value from an input queue. If the queue is empty then the calling thread is suspended until a byte arrives in the queue.

Parameters

in	<i>iqp</i>	pointer to an <code>input_queue_t</code> structure

Returns

A byte value from the queue.

Return values

<code>MSG_RESET</code>	if the queue has been reset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.12.4.2.15 oqGetFullI

```
#define oqGetFullI(  
    oqp)
```

Value:

```
(qSizeX(oqp) - qSpaceI(oqp))
```

Returns the filled space into an output queue.

Parameters

in	<i>oqp</i>	pointer to an output_queue_t structure

Returns

The number of full bytes in the queue.

Return values

0	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.2.16 oqGetEmptyI

```
#define oqGetEmptyI(  
    oqp)
```

Value:

[qSpaceI](#)(*oqp*)

Returns the empty space into an output queue.

Parameters

in	<i>oqp</i>	pointer to an output_queue_t structure

Returns

The number of empty bytes in the queue.

Return values

0	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.2.17 oqIsEmptyI

```
#define oqIsEmptyI(  
    oqp)
```

Value:

```
/*lint -save -e9007 [13.5] No side effects, a pointer is passed.*/  
((bool)(((oqp)->q_wrptr == (oqp)->q_rdprr) && ((oqp)->q_counter != 0U)))  
/*lint -restore*/
```

Evaluates to `true` if the specified output queue is empty.

Parameters

<code>in</code>	<code>oqp</code>	pointer to an <code>output_queue_t</code> structure

Returns

The queue status.

Return values

<code>false</code>	if the queue is not empty.
<code>true</code>	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.2.18 oqlsFullI

```
#define oqlsFullI(  
    oqp)
```

Value:

```
((bool) (qSpaceI(oqp) == 0U))
```

Evaluates to `true` if the specified output queue is full.

Parameters

<code>in</code>	<code>oqp</code>	pointer to an <code>output_queue_t</code> structure

Returns

The queue status.

Return values

<code>false</code>	if the queue is not full.
<code>true</code>	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.2.19 oqPut

```
#define oqPut(  
    oqp,  
    b)
```

Value:

```
oqPutTimeout(oqp, b, TIME_INFINITE)
```

Output queue write.

This function writes a byte value to an output queue. If the queue is full then the calling thread is suspended until there is space in the queue.

Parameters

in	<i>oqp</i>	pointer to an output_queue_t structure
in	<i>b</i>	the byte value to be written in the queue

Returns

The operation status.

Return values

<i>MSG_OK</i>	if the operation succeeded.
<i>MSG_RESET</i>	if the queue has been reset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.12.4.3 Typedef Documentation**7.2.12.4.3.1 io_queue_t**

```
typedef struct io_queue io_queue_t
```

Type of a generic I/O queue structure.

7.2.12.4.3.2 qnotify_t

```
typedef void(* qnotify_t) (io_queue_t *qp)
```

Queue notification callback type.

Parameters

in	<i>qp</i>	the queue pointer

7.2.12.4.3.3 input_queue_t

```
typedef io_queue_t input_queue_t
```

Type of an input queue structure.

This structure represents a generic asymmetrical input queue. Writing to the queue is non-blocking and can be performed from interrupt handlers or from within a kernel lock zone. Reading the queue can be a blocking operation and is supposed to be performed by a system thread.

7.2.12.4.3.4 output_queue_t

```
typedef io_queue_t output_queue_t
```

Type of an output queue structure.

This structure represents a generic asymmetrical output queue. Reading from the queue is non-blocking and can be performed from interrupt handlers or from within a kernel lock zone. Writing the queue can be a blocking operation and is supposed to be performed by a system thread.

7.2.12.4.4 Function Documentation

7.2.12.4.4.1 iq_read()

```
size_t iq_read (  
    input_queue_t * iqp,  
    uint8_t * bp,  
    size_t n) [static]
```

Non-blocking input queue read.

The function reads data from an input queue into a buffer. The operation completes when the specified amount of data has been transferred or when the input queue has been emptied.

Parameters

in	<i>iqp</i>	pointer to an input_queue_t structure
out	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Returns

The number of bytes effectively transferred.

Function Class:

Not an API, this function is for internal use only.

7.2.12.4.4.2 oq_write()

```
size_t oq_write (  
    output_queue_t * oqp,  
    const uint8_t * bp,  
    size_t n) [static]
```

Non-blocking output queue write.

The function writes data from a buffer to an output queue. The operation completes when the specified amount of data has been transferred or when the output queue has been filled.

Parameters

in	<i>oqp</i>	pointer to an output_queue_t structure
in	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Returns

The number of bytes effectively transferred.

Function Class:

Not an API, this function is for internal use only.

7.2.12.4.4.3 iqObjectInit()

```
void iqObjectInit (
    input_queue_t * iqp,
    uint8_t * bp,
    size_t size,
    qnotify_t infy,
    void * link)
```

Initializes an input queue.

A Semaphore is internally initialized and works as a counter of the bytes contained in the queue.

Note

The callback is invoked from within the S-Locked system state.

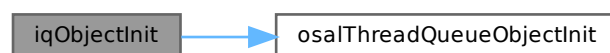
Parameters

out	<i>iqp</i>	pointer to an input_queue_t structure
in	<i>bp</i>	pointer to a memory area allocated as queue buffer
in	<i>size</i>	size of the queue buffer
in	<i>infy</i>	pointer to a callback function that is invoked when data is read from the queue. The value can be NULL.
in	<i>link</i>	application defined pointer

Function Class:

Object or module initializer function.

Here is the call graph for this function:



7.2.12.4.4.4 iqResetI()

```
void iqResetI (  
    input_queue_t * iqp)
```

Resets an input queue.

All the data in the input queue is erased and lost, any waiting thread is resumed with status MSG_RESET.

Note

A reset operation can be used by a low level driver in order to obtain immediate attention from the high level layers.

Parameters

in	<i>iqp</i>	pointer to an <code>input_queue_t</code> structure

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.12.4.4.5 iqPutI()

```
msg_t iqPutI (  
    input_queue_t * iqp,  
    uint8_t b)
```

Input queue write.

A byte value is written into the low end of an input queue. The operation completes immediately.

Parameters

in	<i>iqp</i>	pointer to an <code>input_queue_t</code> structure
in	<i>b</i>	the byte value to be written in the queue

Returns

The operation status.

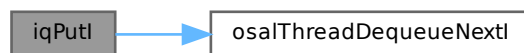
Return values

<i>MSG_OK</i>	if the operation has been completed with success.
<i>MSG_TIMEOUT</i>	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.12.4.4.6 iqGetI()**

```
msg_t iqGetI (  
    input_queue_t * iqp)
```

Input queue non-blocking read.

This function reads a byte value from an input queue. The operation completes immediately.

Note

The callback is invoked after removing a character from the queue.

Parameters

in	<i>iqp</i>	pointer to an <code>input_queue_t</code> structure

Returns

A byte value from the queue.

Return values

<i>MSG_TIMEOUT</i>	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.4.7 iqGetTimeout()

```
msg_t iqGetTimeout (  
    input_queue_t * iqp,  
    sysinterval_t timeout)
```

Input queue read with timeout.

This function reads a byte value from an input queue. If the queue is empty then the calling thread is suspended until a byte arrives in the queue or a timeout occurs.

Note

The callback is invoked after removing a character from the queue.

Parameters

in	<i>iqp</i>	pointer to an <code>input_queue_t</code> structure
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: <code>TIME_IMMEDIATE</code> immediate timeout. <code>TIME_INFINITE</code> no timeout.

Returns

A byte value from the queue.

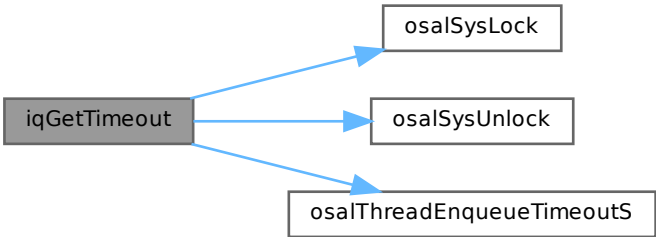
Return values

<code>MSG_TIMEOUT</code>	if the specified time expired.
<code>MSG_RESET</code>	if the queue has been reset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.4.4.8 iqReadI()

```
size_t iqReadI (
    input_queue_t * iqp,
    uint8_t * bp,
    size_t n)
```

Input queue non-blocking read.

The function reads data from an input queue into a buffer. The operation completes immediately.

Parameters

in	<i>iqp</i>	pointer to an <code>input_queue_t</code> structure
out	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Returns

The number of bytes effectively transferred.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.12.4.4.9 iqReadTimeout()

```
size_t iqReadTimeout (
    input_queue_t * iqp,
    uint8_t * bp,
    size_t n,
    sysinterval_t timeout)
```

Input queue read with timeout.

The function reads data from an input queue into a buffer. The operation completes when the specified amount of data has been transferred or after the specified timeout or if the queue has been reset.

Note

The function is not atomic, if you need atomicity it is suggested to use a semaphore or a mutex for mutual exclusion.

The callback is invoked after removing each character from the queue.

Parameters

in	<i>iqp</i>	pointer to an <code>input_queue_t</code> structure
out	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <code>TIME_IMMEDIATE</code> immediate timeout. • <code>TIME_INFINITE</code> no timeout.

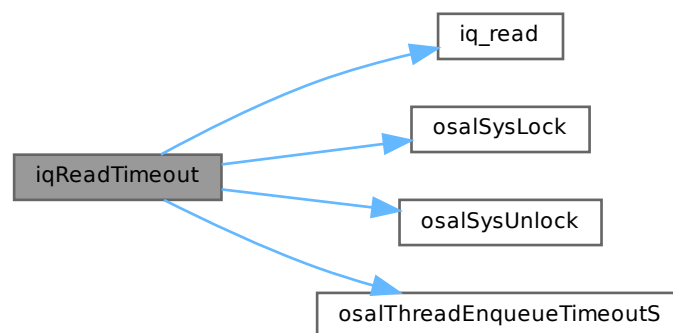
Returns

The number of bytes effectively transferred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.4.4.10 oqObjectInit()

```

void oqObjectInit (
    output_queue_t * oqp,
    uint8_t * bp,
    size_t size,
    qnotify_t onfy,
    void * link)
  
```

Initializes an output queue.

A Semaphore is internally initialized and works as a counter of the free bytes in the queue.

Note

The callback is invoked from within the S-Locked system state.

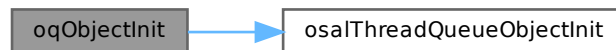
Parameters

out	<i>oqp</i>	pointer to an output_queue_t structure
in	<i>bp</i>	pointer to a memory area allocated as queue buffer
in	<i>size</i>	size of the queue buffer
in	<i>onfy</i>	pointer to a callback function that is invoked when data is written to the queue. The value can be NULL.
in	<i>link</i>	application defined pointer

Function Class:

Object or module initializer function.

Here is the call graph for this function:

**7.2.12.4.4.11 oqResetI()**

```
void oqResetI (  
    output\_queue\_t * oqp)
```

Resets an output queue.

All the data in the output queue is erased and lost, any waiting thread is resumed with status `MSG_RESET`.

Note

A reset operation can be used by a low level driver in order to obtain immediate attention from the high level layers.

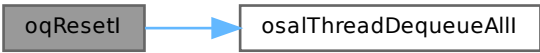
Parameters

in	<i>oqp</i>	pointer to an output_queue_t structure

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:



7.2.12.4.4.12 oqPutI()

```
msg_t oqPutI (
    output_queue_t * oqp,
    uint8_t b)
```

Output queue non-blocking write.

This function writes a byte value to an output queue. The operation completes immediately.

Parameters

in	<i>oqp</i>	pointer to an output_queue_t structure
in	<i>b</i>	the byte value to be written in the queue

Returns

The operation status.

Return values

<i>MSG_OK</i>	if the operation succeeded.
<i>MSG_TIMEOUT</i>	if the queue is full.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.12.4.4.13 oqPutTimeout()

```
msg_t oqPutTimeout (
    output_queue_t * oqp,
    uint8_t b,
    sysinterval_t timeout)
```

Output queue write with timeout.

This function writes a byte value to an output queue. If the queue is full then the calling thread is suspended until there is space in the queue or a timeout occurs.

Note

The callback is invoked after putting the character into the queue.

Parameters

in	<i>oqp</i>	pointer to an output_queue_t structure
in	<i>b</i>	the byte value to be written in the queue
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

Returns

The operation status.

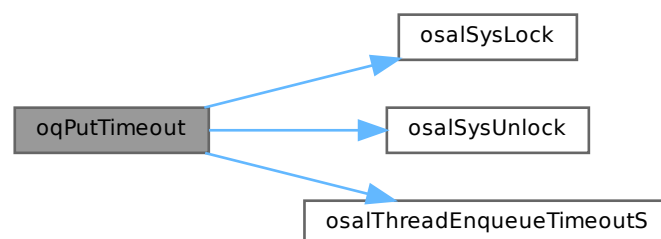
Return values

<i>MSG_OK</i>	if the operation succeeded.
<i>MSG_TIMEOUT</i>	if the specified time expired.
<i>MSG_RESET</i>	if the queue has been reset.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.12.4.4.14 oqGetI()

```
msg_t oqGetI (
    output_queue_t * oqp)
```

Output queue read.

A byte value is read from the low end of an output queue. The operation completes immediately.

Parameters

in	<i>oqp</i>	pointer to an <code>output_queue_t</code> structure

Returns

The byte value from the queue.

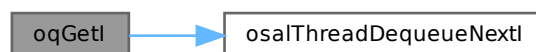
Return values

<code>MSG_TIMEOUT</code>	if the queue is empty.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.12.4.4.15 oqWriteI()**

```
size_t oqWriteI (
    output_queue_t * oqp,
    const uint8_t * bp,
    size_t n)
```

Output queue non-blocking write.

The function writes data from a buffer to an output queue. The operation completes immediately.

Parameters

in	<i>oqp</i>	pointer to an output_queue_t structure
in	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved

Returns

The number of bytes effectively transferred.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.12.4.4.16 oqWriteTimeout()**

```

size_t oqWriteTimeout (
    output_queue_t * oqp,
    const uint8_t * bp,
    size_t n,
    sysinterval_t timeout)
  
```

Output queue write with timeout.

The function writes data from a buffer to an output queue. The operation completes when the specified amount of data has been transferred or after the specified timeout or if the queue has been reset.

Note

The function is not atomic, if you need atomicity it is suggested to use a semaphore or a mutex for mutual exclusion.

The callback is invoked after putting each character into the queue.

Parameters

in	<i>oqp</i>	pointer to an output_queue_t structure
in	<i>bp</i>	pointer to the data buffer
in	<i>n</i>	the maximum amount of data to be transferred, the value 0 is reserved
in	<i>timeout</i>	the number of ticks before the operation timeouts, the following special values are allowed: • <i>TIME_IMMEDIATE</i> immediate timeout. • <i>TIME_INFINITE</i> no timeout.

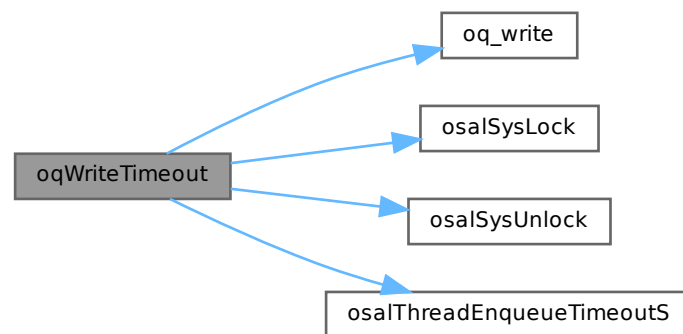
Returns

The number of bytes effectively transferred.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.13 Support Code

HAL Support Code.

7.2.13.1 Detailed Description

HAL Support Code.

Topics

- [MII/RMII Header](#)
MII/RMII Support Header.
- [USB CDC Header](#)
USB CDC Support Header.

7.2.13.2 MII/RMII Header

MII/RMII Support Header.

7.2.13.2.1 Detailed Description

MII/RMII Support Header.

This header contains definitions and types related to MII/RMII.

Generic MII registers

- #define [MII_BMCR](#) 0x00
- #define [MII_BMSR](#) 0x01
- #define [MII_PHYSID1](#) 0x02
- #define [MII_PHYSID2](#) 0x03
- #define [MII_ADVERTISE](#) 0x04
- #define [MII_LPA](#) 0x05
- #define [MII_EXPANSION](#) 0x06
- #define [MII_ANNPTR](#) 0x07
- #define [MII_CTRL1000](#) 0x09
- #define [MII_STAT1000](#) 0x0a
- #define [MII_ESTATUS](#) 0x0f
- #define [MII_PHYSTS](#) 0x10
- #define [MII_MICR](#) 0x11
- #define [MII_DCOUNTER](#) 0x12
- #define [MII_FCSCOUNTER](#) 0x13
- #define [MII_NWAYTEST](#) 0x14
- #define [MII_RERRCOUNTER](#) 0x15
- #define [MII_SREVISION](#) 0x16
- #define [MII_RESV1](#) 0x17
- #define [MII_LBRERROR](#) 0x18
- #define [MII_PHYADDR](#) 0x19
- #define [MII_RESV2](#) 0x1a
- #define [MII_TPISTATUS](#) 0x1b
- #define [MII_NCONFIG](#) 0x1c

Basic mode control register

- #define [BMCR_RESV](#) 0x007f
- #define [BMCR_CTST](#) 0x0080
- #define [BMCR_FULLDPLX](#) 0x0100
- #define [BMCR_ANRESTART](#) 0x0200
- #define [BMCR_ISOLATE](#) 0x0400
- #define [BMCR_PDOWN](#) 0x0800
- #define [BMCR_ANENABLE](#) 0x1000
- #define [BMCR_SPEED100](#) 0x2000
- #define [BMCR_LOOPBACK](#) 0x4000
- #define [BMCR_RESET](#) 0x8000

Basic mode status register

- #define [BMSR_ERCAP](#) 0x0001
- #define [BMSR_JCD](#) 0x0002
- #define [BMSR_LSTATUS](#) 0x0004
- #define [BMSR_ANEGCAPABLE](#) 0x0008
- #define [BMSR_RFAULT](#) 0x0010
- #define [BMSR_ANEGCOMPLETE](#) 0x0020
- #define [BMSR_MFPRESUPPCAP](#) 0x0040
- #define [BMSR_RESV](#) 0x0780
- #define [BMSR_10HALF](#) 0x0800
- #define [BMSR_10FULL](#) 0x1000
- #define [BMSR_100HALF](#) 0x2000
- #define [BMSR_100FULL](#) 0x4000
- #define [BMSR_100BASE4](#) 0x8000

Advertisement control register

- #define [ADVERTISE_SLCT](#) 0x001f
- #define [ADVERTISE_CSMA](#) 0x0001
- #define [ADVERTISE_10HALF](#) 0x0020
- #define [ADVERTISE_10FULL](#) 0x0040
- #define [ADVERTISE_100HALF](#) 0x0080
- #define [ADVERTISE_100FULL](#) 0x0100
- #define [ADVERTISE_100BASE4](#) 0x0200
- #define [ADVERTISE_PAUSE_CAP](#) 0x0400
- #define [ADVERTISE_PAUSE_ASYM](#) 0x0800
- #define [ADVERTISE_RESV](#) 0x1000
- #define [ADVERTISE_RFAULT](#) 0x2000
- #define [ADVERTISE_LPACK](#) 0x4000
- #define [ADVERTISE_NPAGE](#) 0x8000
- #define [ADVERTISE_FULL](#)
- #define [ADVERTISE_ALL](#)

Link partner ability register

- #define [LPA_SLCT](#) 0x001f
- #define [LPA_10HALF](#) 0x0020
- #define [LPA_10FULL](#) 0x0040
- #define [LPA_100HALF](#) 0x0080
- #define [LPA_100FULL](#) 0x0100
- #define [LPA_100BASE4](#) 0x0200
- #define [LPA_PAUSE_CAP](#) 0x0400
- #define [LPA_PAUSE_ASYM](#) 0x0800
- #define [LPA_RESV](#) 0x1000
- #define [LPA_RFAULT](#) 0x2000
- #define [LPA_LPACK](#) 0x4000
- #define [LPA_NPAGE](#) 0x8000
- #define [LPA_DUPLEX](#) ([LPA_10FULL](#) | [LPA_100FULL](#))
- #define [LPA_100](#) ([LPA_100FULL](#) | [LPA_100HALF](#) | [LPA_100BASE4](#))

Expansion register for auto-negotiation

- #define `EXPANSION_NWAY` 0x0001
- #define `EXPANSION_LCWP` 0x0002
- #define `EXPANSION_ENABLENPAGE` 0x0004
- #define `EXPANSION_NPCAPABLE` 0x0008
- #define `EXPANSION_MFAULTS` 0x0010
- #define `EXPANSION_RESV` 0xffe0

N-way test register

- #define `NWAYTEST_RESV1` 0x00ff
- #define `NWAYTEST_LOOPBACK` 0x0100
- #define `NWAYTEST_RESV2` 0xfe00

PHY identifiers

- #define `MII_DM9161_ID` 0x0181b8a0
- #define `MII_AM79C875_ID` 0x00225540
- #define `MII_KSZ8081_ID` 0x00221560
- #define `MII_KS8721_ID` 0x00221610
- #define `MII_STE101P_ID` 0x00061C50
- #define `MII_DP83848I_ID` 0x20005C90
- #define `MII_LAN8710A_ID` 0x0007C0F1
- #define `MII_LAN8720_ID` 0x0007C0F0
- #define `MII_LAN8742A_ID` 0x0007C130
- #define `MII_LAN8740A_ID` 0x0007C110

7.2.13.2.2 Macro Definition Documentation

7.2.13.2.2.1 MII_BMCR

```
#define MII_BMCR 0x00
```

Basic mode control register.

7.2.13.2.2.2 MII_BMSR

```
#define MII_BMSR 0x01
```

Basic mode status register.

7.2.13.2.2.3 MII_PHYSID1

```
#define MII_PHYSID1 0x02
```

PHYS ID 1.

7.2.13.2.2.4 MII_PHYSID2

```
#define MII_PHYSID2 0x03
```

PHYS ID 2.

7.2.13.2.2.5 MII_ADVERTISE

```
#define MII_ADVERTISE 0x04
```

Advertisement control reg.

7.2.13.2.2.6 MII_LPA

```
#define MII_LPA 0x05
```

Link partner ability reg.

7.2.13.2.2.7 MII_EXPANSION

```
#define MII_EXPANSION 0x06
```

Expansion register.

7.2.13.2.2.8 MII_ANNPTR

```
#define MII_ANNPTR 0x07
```

1000BASE-T control.

7.2.13.2.2.9 MII_CTRL1000

```
#define MII_CTRL1000 0x09
```

1000BASE-T control.

7.2.13.2.2.10 MII_STAT1000

```
#define MII_STAT1000 0x0a
```

1000BASE-T status.

7.2.13.2.2.11 MII_ESTATUS

```
#define MII_ESTATUS 0x0f
```

Extended Status.

7.2.13.2.2.12 MII_PHYSTS

```
#define MII_PHYSTS 0x10
```

PHY Status register.

7.2.13.2.2.13 MII_MICR

```
#define MII_MICR 0x11
```

MII Interrupt ctrl register.

7.2.13.2.2.14 MII_DCOUNTER

```
#define MII_DCOUNTER 0x12
```

Disconnect counter.

7.2.13.2.2.15 MII_FCSCOUNTER

```
#define MII_FCSCOUNTER 0x13
```

False carrier counter.

7.2.13.2.2.16 MII_NWAYTEST

```
#define MII_NWAYTEST 0x14
```

N-way auto-neg test reg.

7.2.13.2.2.17 MII_RERRCOUNTER

```
#define MII_RERRCOUNTER 0x15
```

Receive error counter.

7.2.13.2.2.18 MII_SREVISION

```
#define MII_SREVISION 0x16
```

Silicon revision.

7.2.13.2.2.19 MII_RESV1

```
#define MII_RESV1 0x17
```

Reserved.

7.2.13.2.2.20 MII_LBRERROR

```
#define MII_LBRERROR 0x18
```

Lpback, rx, bypass error.

7.2.13.2.2.21 MII_PHYADDR

```
#define MII_PHYADDR 0x19
```

PHY address.

7.2.13.2.2.22 MII_RESV2

```
#define MII_RESV2 0x1a
```

Reserved.

7.2.13.2.2.23 MII_TPISTATUS

```
#define MII_TPISTATUS 0x1b
```

TPI status for 10Mbps.

7.2.13.2.2.24 MII_NCONFIG

```
#define MII_NCONFIG 0x1c
```

Network interface config.

7.2.13.2.2.25 BMCR_RESV

```
#define BMCR_RESV 0x007f
```

Unused.

7.2.13.2.2.26 BMCR_CTST

```
#define BMCR_CTST 0x0080
```

Collision test.

7.2.13.2.2.27 BMCR_FULLDPLX

```
#define BMCR_FULLDPLX 0x0100
```

Full duplex.

7.2.13.2.2.28 BMCR_ANRESTART

```
#define BMCR_ANRESTART 0x0200
```

Auto negotiation restart.

7.2.13.2.2.29 BMCR_ISOLATE

```
#define BMCR_ISOLATE 0x0400
```

Disconnect DP83840 from MII.

7.2.13.2.2.30 BMCR_PDOWN

```
#define BMCR_PDOWN 0x0800
```

Powerdown.

7.2.13.2.2.31 BMCR_ANENABLE

```
#define BMCR_ANENABLE 0x1000
```

Enable auto negotiation.

7.2.13.2.2.32 BMCR_SPEED100

```
#define BMCR_SPEED100 0x2000
```

Select 100Mbps.

7.2.13.2.2.33 BMCR_LOOPBACK

```
#define BMCR_LOOPBACK 0x4000
```

TXD loopback bit.

7.2.13.2.2.34 BMCR_RESET

```
#define BMCR_RESET 0x8000
```

Reset.

7.2.13.2.2.35 BMSR_ERCAP

```
#define BMSR_ERCAP 0x0001
```

Ext-reg capability.

7.2.13.2.2.36 BMSR_JCD

```
#define BMSR_JCD 0x0002
```

Jabber detected.

7.2.13.2.2.37 BMSR_LSTATUS

```
#define BMSR_LSTATUS 0x0004
```

Link status.

7.2.13.2.2.38 BMSR_ANEGCAPABLE

```
#define BMSR_ANEGCAPABLE 0x0008
```

Able to do auto-negotiation.

7.2.13.2.2.39 BMSR_RFAULT

```
#define BMSR_RFAULT 0x0010
```

Remote fault detected.

7.2.13.2.2.40 BMSR_ANEGCOMPLETE

```
#define BMSR_ANEGCOMPLETE 0x0020
```

Auto-negotiation complete.

7.2.13.2.2.41 BMSR_MFPRESUPPCAP

```
#define BMSR_MFPRESUPPCAP 0x0040
```

Able to suppress preamble.

7.2.13.2.2.42 BMSR_RESV

```
#define BMSR_RESV 0x0780
```

Unused.

7.2.13.2.2.43 BMSR_10HALF

```
#define BMSR_10HALF 0x0800
```

Can do 10mbps, half-duplex.

7.2.13.2.2.44 BMSR_10FULL

```
#define BMSR_10FULL 0x1000
```

Can do 10mbps, full-duplex.

7.2.13.2.2.45 BMSR_100HALF

```
#define BMSR_100HALF 0x2000
```

Can do 100mbps, half-duplex.

7.2.13.2.2.46 BMSR_100FULL

```
#define BMSR_100FULL 0x4000
```

Can do 100mbps, full-duplex.

7.2.13.2.2.47 BMSR_100BASE4

```
#define BMSR_100BASE4 0x8000
```

Can do 100mbps, 4k packets.

7.2.13.2.2.48 ADVERTISE_SLCT

```
#define ADVERTISE_SLCT 0x001f
```

Selector bits.

7.2.13.2.2.49 ADVERTISE_CSMA

```
#define ADVERTISE_CSMA 0x0001
```

Only selector supported.

7.2.13.2.2.50 ADVERTISE_10HALF

```
#define ADVERTISE_10HALF 0x0020
```

Try for 10mbps half-duplex.

7.2.13.2.2.51 ADVERTISE_10FULL

```
#define ADVERTISE_10FULL 0x0040
```

Try for 10mbps full-duplex.

7.2.13.2.2.52 ADVERTISE_100HALF

```
#define ADVERTISE_100HALF 0x0080
```

Try for 100mbps half-duplex.

7.2.13.2.2.53 ADVERTISE_100FULL

```
#define ADVERTISE_100FULL 0x0100
```

Try for 100mbps full-duplex.

7.2.13.2.2.54 ADVERTISE_100BASE4

```
#define ADVERTISE_100BASE4 0x0200
```

Try for 100mbps 4k packets.

7.2.13.2.2.55 ADVERTISE_PAUSE_CAP

```
#define ADVERTISE_PAUSE_CAP 0x0400
```

Try for pause.

7.2.13.2.2.56 ADVERTISE_PAUSE_ASYM

```
#define ADVERTISE_PAUSE_ASYM 0x0800
```

Try for asymmetric pause.

7.2.13.2.2.57 ADVERTISE_RESV

```
#define ADVERTISE_RESV 0x1000
```

Unused.

7.2.13.2.2.58 ADVERTISE_RFAULT

```
#define ADVERTISE_RFAULT 0x2000
```

Say we can detect faults.

7.2.13.2.2.59 ADVERTISE_LPACK

```
#define ADVERTISE_LPACK 0x4000
```

Ack link partners response.

7.2.13.2.2.60 ADVERTISE_NPAGE

```
#define ADVERTISE_NPAGE 0x8000
```

Next page bit.

7.2.13.2.2.61 ADVERTISE_FULL

```
#define ADVERTISE_FULL
```

Value:

```
(ADVERTISE_100FULL | ADVERTISE_10FULL | \
ADVERTISE_CSMA)
```

7.2.13.2.2.62 ADVERTISE_ALL

```
#define ADVERTISE_ALL
```

Value:

```
(ADVERTISE_10HALF | ADVERTISE_10FULL | \
ADVERTISE_100HALF | ADVERTISE_100FULL)
```

7.2.13.2.2.63 LPA_SLCT

```
#define LPA_SLCT 0x001f
```

Same as advertise selector.

7.2.13.2.2.64 LPA_10HALF

```
#define LPA_10HALF 0x0020
```

Can do 10mbps half-duplex.

7.2.13.2.2.65 LPA_10FULL

```
#define LPA_10FULL 0x0040
```

Can do 10mbps full-duplex.

7.2.13.2.2.66 LPA_100HALF

```
#define LPA_100HALF 0x0080
```

Can do 100mbps half-duplex.

7.2.13.2.2.67 LPA_100FULL

```
#define LPA_100FULL 0x0100
```

Can do 100mbps full-duplex.

7.2.13.2.2.68 LPA_100BASE4

```
#define LPA_100BASE4 0x0200
```

Can do 100mbps 4k packets.

7.2.13.2.2.69 LPA_PAUSE_CAP

```
#define LPA_PAUSE_CAP 0x0400
```

Can pause.

7.2.13.2.2.70 LPA_PAUSE_ASYM

```
#define LPA_PAUSE_ASYM 0x0800
```

Can pause asymmetrically.

7.2.13.2.2.71 LPA_RESV

```
#define LPA_RESV 0x1000
```

Unused.

7.2.13.2.2.72 LPA_RFAULT

```
#define LPA_RFAULT 0x2000
```

Link partner faulted.

7.2.13.2.2.73 LPA_LPACK

```
#define LPA_LPACK 0x4000
```

Link partner acked us.

7.2.13.2.2.74 LPA_NPAGE

```
#define LPA_NPAGE 0x8000
```

Next page bit.

7.2.13.2.2.75 LPA_DUPLEX

```
#define LPA_DUPLEX (LPA_10FULL | LPA_100FULL)
```

7.2.13.2.2.76 LPA_100

```
#define LPA_100 (LPA_100FULL | LPA_100HALF | LPA_100BASE4)
```

7.2.13.2.2.77 EXPANSION_NWAY

```
#define EXPANSION_NWAY 0x0001
```

Can do N-way auto-nego.

7.2.13.2.2.78 EXPANSION_LCWP

```
#define EXPANSION_LCWP 0x0002
```

Got new RX page code word.

7.2.13.2.2.79 EXPANSION_ENABLENPAGE

```
#define EXPANSION_ENABLENPAGE 0x0004
```

This enables npage words.

7.2.13.2.2.80 EXPANSION_NPCAPABLE

```
#define EXPANSION_NPCAPABLE 0x0008
```

Link partner supports npage.

7.2.13.2.2.81 EXPANSION_MFAULTS

```
#define EXPANSION_MFAULTS 0x0010
```

Multiple faults detected.

7.2.13.2.2.82 EXPANSION_RESV

```
#define EXPANSION_RESV 0xffe0
```

Unused.

7.2.13.2.2.83 NWAYTEST_RESV1

```
#define NWAYTEST_RESV1 0x00ff
```

Unused.

7.2.13.2.2.84 NWAYTEST_LOOPBACK

```
#define NWAYTEST_LOOPBACK 0x0100
```

Enable loopback for N-way.

7.2.13.2.2.85 NWAYTEST_RESV2

```
#define NWAYTEST_RESV2 0xfe00
```

Unused.

7.2.13.2.2.86 MII_DM9161_ID

```
#define MII_DM9161_ID 0x0181b8a0
```

7.2.13.2.2.87 MII_AM79C875_ID

```
#define MII_AM79C875_ID 0x00225540
```

7.2.13.2.2.88 MII_KSZ8081_ID

```
#define MII_KSZ8081_ID 0x00221560
```

7.2.13.2.2.89 MII_KS8721_ID

```
#define MII_KS8721_ID 0x00221610
```

7.2.13.2.2.90 MII_STE101P_ID

```
#define MII_STE101P_ID 0x00061C50
```

7.2.13.2.2.91 MII_DP83848I_ID

```
#define MII_DP83848I_ID 0x20005C90
```

7.2.13.2.2.92 MII_LAN8710A_ID

```
#define MII_LAN8710A_ID 0x0007C0F1
```

7.2.13.2.2.93 MII_LAN8720_ID

```
#define MII_LAN8720_ID 0x0007C0F0
```

7.2.13.2.2.94 MII_LAN8742A_ID

```
#define MII_LAN8742A_ID 0x0007C130
```

7.2.13.2.2.95 MII_LAN8740A_ID

```
#define MII_LAN8740A_ID 0x0007C110
```

7.2.13.3 USB CDC Header

USB CDC Support Header.

7.2.13.3.1 Detailed Description

USB CDC Support Header.

This header contains definitions and types related to USB CDC.

CDC specific messages.

- `#define CDC_SEND_ENCAPSULATED_COMMAND 0x00U`
- `#define CDC_GET_ENCAPSULATED_RESPONSE 0x01U`
- `#define CDC_SET_COMM_FEATURE 0x02U`
- `#define CDC_GET_COMM_FEATURE 0x03U`
- `#define CDC_CLEAR_COMM_FEATURE 0x04U`
- `#define CDC_SET_AUX_LINE_STATE 0x10U`
- `#define CDC_SET_HOOK_STATE 0x11U`
- `#define CDC_PULSE_SETUP 0x12U`
- `#define CDC_SEND_PULSE 0x13U`
- `#define CDC_SET_PULSE_TIME 0x14U`
- `#define CDC_RING_AUX_JACK 0x15U`
- `#define CDC_SET_LINE_CODING 0x20U`
- `#define CDC_GET_LINE_CODING 0x21U`
- `#define CDC_SET_CONTROL_LINE_STATE 0x22U`
- `#define CDC_SEND_BREAK 0x23U`
- `#define CDC_SET_RINGER_PARMS 0x30U`
- `#define CDC_GET_RINGER_PARMS 0x31U`
- `#define CDC_SET_OPERATION_PARMS 0x32U`
- `#define CDC_GET_OPERATION_PARMS 0x33U`

CDC classes

- #define `CDC_COMMUNICATION_INTERFACE_CLASS` 0x02U
- #define `CDC_DATA_INTERFACE_CLASS` 0x0AU

CDC subclasses

- #define `CDC_ABSTRACT_CONTROL_MODEL` 0x02U

CDC descriptors

- #define `CDC_CS_INTERFACE` 0x24U

CDC subdescriptors

- #define `CDC_HEADER` 0x00U
- #define `CDC_CALL_MANAGEMENT` 0x01U
- #define `CDC_ABSTRACT_CONTROL_MANAGEMENT` 0x02U
- #define `CDC_UNION` 0x06U

Line Control bit definitions.

- #define `LC_STOP_1` 0U
- #define `LC_STOP_1P5` 1U
- #define `LC_STOP_2` 2U
- #define `LC_PARITY_NONE` 0U
- #define `LC_PARITY_ODD` 1U
- #define `LC_PARITY_EVEN` 2U
- #define `LC_PARITY_MARK` 3U
- #define `LC_PARITY_SPACE` 4U

Data Structures

- struct `cdc_linecoding_t`
Type of Line Coding structure.

7.2.13.3.2 Macro Definition Documentation

7.2.13.3.2.1 CDC_SEND_ENCAPSULATED_COMMAND

```
#define CDC_SEND_ENCAPSULATED_COMMAND 0x00U
```

7.2.13.3.2.2 CDC_GET_ENCAPSULATED_RESPONSE

```
#define CDC_GET_ENCAPSULATED_RESPONSE 0x01U
```

7.2.13.3.2.3 CDC_SET_COMM_FEATURE

```
#define CDC_SET_COMM_FEATURE 0x02U
```

7.2.13.3.2.4 CDC_GET_COMM_FEATURE

```
#define CDC_GET_COMM_FEATURE 0x03U
```

7.2.13.3.2.5 CDC_CLEAR_COMM_FEATURE

```
#define CDC_CLEAR_COMM_FEATURE 0x04U
```

7.2.13.3.2.6 CDC_SET_AUX_LINE_STATE

```
#define CDC_SET_AUX_LINE_STATE 0x10U
```

7.2.13.3.2.7 CDC_SET_HOOK_STATE

```
#define CDC_SET_HOOK_STATE 0x11U
```

7.2.13.3.2.8 CDC_PULSE_SETUP

```
#define CDC_PULSE_SETUP 0x12U
```

7.2.13.3.2.9 CDC_SEND_PULSE

```
#define CDC_SEND_PULSE 0x13U
```

7.2.13.3.2.10 CDC_SET_PULSE_TIME

```
#define CDC_SET_PULSE_TIME 0x14U
```

7.2.13.3.2.11 CDC_RING_AUX_JACK

```
#define CDC_RING_AUX_JACK 0x15U
```

7.2.13.3.2.12 CDC_SET_LINE_CODING

```
#define CDC_SET_LINE_CODING 0x20U
```

7.2.13.3.2.13 CDC_GET_LINE_CODING

```
#define CDC_GET_LINE_CODING 0x21U
```

7.2.13.3.2.14 CDC_SET_CONTROL_LINE_STATE

```
#define CDC_SET_CONTROL_LINE_STATE 0x22U
```

7.2.13.3.2.15 CDC_SEND_BREAK

```
#define CDC_SEND_BREAK 0x23U
```

7.2.13.3.2.16 CDC_SET_RINGER_PARMS

```
#define CDC_SET_RINGER_PARMS 0x30U
```

7.2.13.3.2.17 CDC_GET_RINGER_PARMS

```
#define CDC_GET_RINGER_PARMS 0x31U
```

7.2.13.3.2.18 CDC_SET_OPERATION_PARMS

```
#define CDC_SET_OPERATION_PARMS 0x32U
```

7.2.13.3.2.19 CDC_GET_OPERATION_PARMS

```
#define CDC_GET_OPERATION_PARMS 0x33U
```

7.2.13.3.2.20 CDC_COMMUNICATION_INTERFACE_CLASS

```
#define CDC_COMMUNICATION_INTERFACE_CLASS 0x02U
```

7.2.13.3.2.21 CDC_DATA_INTERFACE_CLASS

```
#define CDC_DATA_INTERFACE_CLASS 0x0AU
```

7.2.13.3.2.22 CDC_ABSTRACT_CONTROL_MODEL

```
#define CDC_ABSTRACT_CONTROL_MODEL 0x02U
```

7.2.13.3.2.23 CDC_CS_INTERFACE

```
#define CDC_CS_INTERFACE 0x24U
```

7.2.13.3.2.24 CDC_HEADER

```
#define CDC_HEADER 0x00U
```

7.2.13.3.2.25 CDC_CALL_MANAGEMENT

```
#define CDC_CALL_MANAGEMENT 0x01U
```

7.2.13.3.2.26 CDC_ABSTRACT_CONTROL_MANAGEMENT

```
#define CDC_ABSTRACT_CONTROL_MANAGEMENT 0x02U
```

7.2.13.3.2.27 CDC_UNION

```
#define CDC_UNION 0x06U
```

7.2.13.3.2.28 LC_STOP_1

```
#define LC_STOP_1 0U
```

7.2.13.3.2.29 LC_STOP_1P5

```
#define LC_STOP_1P5 1U
```

7.2.13.3.2.30 LC_STOP_2

```
#define LC_STOP_2 2U
```

7.2.13.3.2.31 LC_PARITY_NONE

```
#define LC_PARITY_NONE 0U
```

7.2.13.3.2.32 LC_PARITY_ODD

```
#define LC_PARITY_ODD 1U
```


7.2.13.3.2.33 LC_PARITY_EVEN

```
#define LC_PARITY_EVEN 2U
```

7.2.13.3.2.34 LC_PARITY_MARK

```
#define LC_PARITY_MARK 3U
```

7.2.13.3.2.35 LC_PARITY_SPACE

```
#define LC_PARITY_SPACE 4U
```

7.2.14 OSAL

Operating System Abstraction Layer.

7.2.14.1 Detailed Description

Operating System Abstraction Layer.

The OSAL

The OSAL is the link between ChibiOS/HAL and services provided by operating systems like:

- Critical Zones handling.
- Interrupts handling.
- Runtime Errors management.
- Inter-task synchronization.
- Task-ISR synchronization.
- Time management.
- Events.

ChibiOS/HAL is designed to tightly integrate with the underlying RTOS in order to provide the best experience to developers and minimize integration issues.

This section describes the API that OSALs are expected to expose to the HAL.

RTOS Requirements

The OSAL API closely resembles the ChibiOS/RT API, for obvious reasons, however an OSAL module can be implemented for any reasonably complete RTOS or even a RTOS-less bare metal machine, if required. In order to be able to support an HAL an RTOS should support the following minimal set of features:

- Task-level critical zones API.
- ISR-level critical zones API, only required on those CPU architectures supporting preemptable ISRs like Cortex-Mx cores.
- Ability to invoke API functions from inside a task critical zone. Functions that are required to support this feature are marked with an "I" or "S" letter at the end of the name.
- Ability to invoke API functions from inside an ISR critical zone. Functions that are required to support this feature are marked with an "I" letter at the end of the name.
- Tasks Queues or Counting Semaphores with Timeout capability.
- Ability to suspend a task and wakeup it from ISR with Timeout capability.
- Event flags, the mechanism can be simulated using callbacks in case the RTOS does not support it.
- Mutual Exclusion mechanism like Semaphores or Mutexes.

All the above requirements can be satisfied even on naked HW with a very thin SW layer. In case that the HAL is required to work without an RTOS.

Supported RTOSes

The RTOSes supported out of the box are:

- ChibiOS/RT
- ChibiOS/NIL

Implementations have also been successfully created on RTOSes not belonging to the ChibiOS products family but are not supported as a core feature of ChibiOS/HAL.

Common constants

- `#define FALSE 0`
- `#define TRUE 1`
- `#define OSAL_SUCCESS false`
- `#define OSAL_FAILED true`

Messages

- `#define MSG_OK (msg_t)0`
- `#define MSG_TIMEOUT (msg_t)-1`
- `#define MSG_RESET (msg_t)-2`

Special time constants

- #define `TIME_IMMEDIATE` `((sysinterval_t)0)`
- #define `TIME_INFINITE` `((sysinterval_t)-1)`

Systick modes.

- #define `OSAL_ST_MODE_NONE` 0
- #define `OSAL_ST_MODE_PERIODIC` 1
- #define `OSAL_ST_MODE_FREERUNNING` 2

Systick parameters.

- #define `OSAL_ST_RESOLUTION` 32
Size in bits of the `systick_t` type.
- #define `OSAL_ST_FREQUENCY` 1000
Required systick frequency or resolution.
- #define `OSAL_ST_MODE` `OSAL_ST_MODE_PERIODIC`
Systick mode required by the underlying OS.

IRQ-related constants

- #define `OSAL_IRQ_PRIORITY_LEVELS` 16U
Total priority levels.
- #define `OSAL_IRQ_MAXIMUM_PRIORITY` 0U
Highest IRQ priority for HAL drivers.

Debug related macros

- #define `osalDbgAssert`(c, remark)
Condition assertion.
- #define `osalDbgCheck`(c)
Function parameters check.
- #define `osalDbgCheckClassI`()
I-Class state check.
- #define `osalDbgCheckClassS`()
S-Class state check.

IRQ service routines wrappers

- #define `OSAL_IRQ_IS_VALID_PRIORITY`(n)
Priority level verification macro.
- #define `OSAL_IRQ_PROLOGUE`()
IRQ prologue code.
- #define `OSAL_IRQ_EPILOGUE`()
IRQ epilogue code.
- #define `OSAL_IRQ_HANDLER`(id)
IRQ handler function declaration.

Time conversion utilities

- #define `OSAL_S2I`(secs)
Seconds to time interval.
- #define `OSAL_MS2I`(msecs)
Milliseconds to time interval.
- #define `OSAL_US2I`(usecs)
Microseconds to time interval.
- #define `OSAL_I2S`(interval)
Time interval to seconds.
- #define `OSAL_I2MS`(interval)
Time interval to milliseconds.
- #define `OSAL_I2US`(interval)
Time interval to microseconds.

Time conversion utilities for the realtime counter

- #define `OSAL_S2RTC`(freq, sec)
Seconds to realtime counter.
- #define `OSAL_MS2RTC`(freq, msec)
Milliseconds to realtime counter.
- #define `OSAL_US2RTC`(freq, usec)
Microseconds to realtime counter.

Sleep macros using absolute time

- #define `osalThreadSleepSeconds`(secs)
Delays the invoking thread for the specified number of seconds.
- #define `osalThreadSleepMilliseconds`(msecs)
Delays the invoking thread for the specified number of milliseconds.
- #define `osalThreadSleepMicroseconds`(usecs)
Delays the invoking thread for the specified number of microseconds.

Data Structures

- struct `event_source`
Events source object.
- struct `threads_queue_t`
Type of a thread queue.

Macros

- #define `OSAL_DBG_ENABLE_ASSERTS` FALSE
Enables OSAL assertions.
- #define `OSAL_DBG_ENABLE_CHECKS` FALSE
Enables OSAL functions parameters checks.

Typedefs

- typedef uint32_t [syssts_t](#)
Type of a system status word.
- typedef int32_t [msg_t](#)
Type of a message.
- typedef uint32_t [systime_t](#)
Type of system time counter.
- typedef uint32_t [sysinterval_t](#)
Type of system time interval.
- typedef uint64_t [time_conv_t](#)
Type of time conversion variable.
- typedef uint32_t [rtcnt_t](#)
Type of realtime counter.
- typedef void * [thread_reference_t](#)
Type of a thread reference.
- typedef uint32_t [eventflags_t](#)
Type of an event flags mask.
- typedef struct [event_source](#) [event_source_t](#)
Type of an event flags object.
- typedef void(* [eventcallback_t](#)) ([event_source_t](#) *esp)
Type of an event source callback.
- typedef uint32_t [mutex_t](#)
Type of a mutex.

Functions

- void [osalInit](#) (void)
OSAL module initialization.
- void [osalSysHalt](#) (const char *reason)
System halt with error message.
- void [osalSysPolledDelayX](#) ([rtcnt_t](#) cycles)
Polled delay.
- void [osalOsTimerHandlerI](#) (void)
System timer handler.
- void [osalOsRescheduleS](#) (void)
Checks if a reschedule is required and performs it.
- [systime_t](#) [osalOsGetSystemTimeX](#) (void)
Current system time.
- void [osalThreadSleepS](#) ([sysinterval_t](#) time)
Suspends the invoking thread for the specified time.
- void [osalThreadSleep](#) ([sysinterval_t](#) time)
Suspends the invoking thread for the specified time.
- [msg_t](#) [osalThreadSuspendS](#) ([thread_reference_t](#) *trp)
Sends the current thread sleeping and sets a reference variable.
- [msg_t](#) [osalThreadSuspendTimeoutS](#) ([thread_reference_t](#) *trp, [sysinterval_t](#) timeout)
Sends the current thread sleeping and sets a reference variable.
- void [osalThreadResumeI](#) ([thread_reference_t](#) *trp, [msg_t](#) msg)
Wakes up a thread waiting on a thread reference object.
- void [osalThreadResumeS](#) ([thread_reference_t](#) *trp, [msg_t](#) msg)

- Wakes up a thread waiting on a thread reference object.*
- `msg_t osalThreadEnqueueTimeoutS (threads_queue_t *tqp, sysinterval_t timeout)`
Enqueues the caller thread.
- `void osalThreadDequeueNextI (threads_queue_t *tqp, msg_t msg)`
Dequeues and wakes up one thread from the queue, if any.
- `void osalThreadDequeueAllI (threads_queue_t *tqp, msg_t msg)`
Dequeues and wakes up all threads from the queue.
- `void osalEventBroadcastFlagsI (event_source_t *esp, eventflags_t flags)`
Add flags to an event source object.
- `void osalEventBroadcastFlags (event_source_t *esp, eventflags_t flags)`
Add flags to an event source object.
- `void osalEventSetCallback (event_source_t *esp, eventcallback_t cb, void *param)`
Event callback setup.
- `void osalMutexLock (mutex_t *mp)`
Locks the specified mutex.
- `void osalMutexUnlock (mutex_t *mp)`
Unlocks the specified mutex.
- `static void osalSysDisable (void)`
Disables interrupts globally.
- `static void osalSysEnable (void)`
Enables interrupts globally.
- `static void osalSysLock (void)`
Enters a critical zone from thread context.
- `static void osalSysUnlock (void)`
Leaves a critical zone from thread context.
- `static void osalSysLockFromISR (void)`
Enters a critical zone from ISR context.
- `static void osalSysUnlockFromISR (void)`
Leaves a critical zone from ISR context.
- `static syssts_t osalSysGetStatusAndLockX (void)`
Returns the execution status and enters a critical zone.
- `static void osalSysRestoreStatusX (syssts_t sts)`
Restores the specified execution status and leaves a critical zone.
- `static systime_t osalTimeAddX (systime_t systime, sysinterval_t interval)`
Adds an interval to a system time returning a system time.
- `static sysinterval_t osalTimeDiffX (systime_t start, systime_t end)`
Subtracts two system times returning an interval.
- `static bool osalTimeInRangeX (systime_t time, systime_t start, systime_t end)`
Checks if the specified time is within the specified time window.
- `static void osalThreadQueueObjectInit (threads_queue_t *tqp)`
Initializes a threads queue object.
- `static void osalEventObjectInit (event_source_t *esp)`
Initializes an event source object.
- `static void osalMutexObjectInit (mutex_t *mp)`
Initializes a `mutex_t` object.

Variables

- `const char * osal_halt_msg`
Pointer to a halt error message.
- `const char * osal_halt_msg`
Pointer to a halt error message.

7.2.14.2 Macro Definition Documentation

7.2.14.2.1 FALSE

```
#define FALSE 0
```

7.2.14.2.2 TRUE

```
#define TRUE 1
```

7.2.14.2.3 OSAL_SUCCESS

```
#define OSAL_SUCCESS false
```

7.2.14.2.4 OSAL_FAILED

```
#define OSAL_FAILED true
```

7.2.14.2.5 MSG_OK

```
#define MSG_OK (msg_t)0
```

7.2.14.2.6 MSG_TIMEOUT

```
#define MSG_TIMEOUT (msg_t)-1
```

7.2.14.2.7 MSG_RESET

```
#define MSG_RESET (msg_t)-2
```

7.2.14.2.8 TIME_IMMEDIATE

```
#define TIME_IMMEDIATE ((sysinterval_t)0)
```

7.2.14.2.9 TIME_INFINITE

```
#define TIME_INFINITE ((sysinterval_t)-1)
```

7.2.14.2.10 OSAL_ST_MODE_NONE

```
#define OSAL_ST_MODE_NONE 0
```

7.2.14.2.11 OSAL_ST_MODE_PERIODIC

```
#define OSAL_ST_MODE_PERIODIC 1
```

7.2.14.2.12 OSAL_ST_MODE_FREERUNNING

```
#define OSAL_ST_MODE_FREERUNNING 2
```

7.2.14.2.13 OSAL_ST_RESOLUTION

```
#define OSAL_ST_RESOLUTION 32
```

Size in bits of the `systick_t` type.

7.2.14.2.14 OSAL_ST_FREQUENCY

```
#define OSAL_ST_FREQUENCY 1000
```

Required systick frequency or resolution.

7.2.14.2.15 OSAL_ST_MODE

```
#define OSAL_ST_MODE OSAL_ST_MODE_PERIODIC
```

Systick mode required by the underlying OS.

7.2.14.2.16 OSAL_IRQ_PRIORITY_LEVELS

```
#define OSAL_IRQ_PRIORITY_LEVELS 16U
```

Total priority levels.

Implementation not mandatory.

7.2.14.2.17 OSAL_IRQ_MAXIMUM_PRIORITY

```
#define OSAL_IRQ_MAXIMUM_PRIORITY 0U
```

Highest IRQ priority for HAL drivers.

Implementation not mandatory.

7.2.14.2.18 OSAL_DBG_ENABLE_ASSERTS

```
#define OSAL_DBG_ENABLE_ASSERTS FALSE
```

Enables OSAL assertions.

7.2.14.2.19 OSAL_DBG_ENABLE_CHECKS

```
#define OSAL_DBG_ENABLE_CHECKS FALSE
```

Enables OSAL functions parameters checks.

7.2.14.2.20 osalDbgAssert

```
#define osalDbgAssert(  
    c,  
    remark)
```

Value:

```
do {  
    /*lint -save -e506 -e774 [2.1, 14.3] Can be a constant by design.*/  
    if (OSAL_DBG_ENABLE_ASSERTS != FALSE) {  
        if (!(c)) {  
            /*lint -restore*/  
            osalSysHalt(__func__);  
        }  
    }  
} while (false)
```

Condition assertion.

If the condition check fails then the OSAL panics with a message and halts.

Note

The condition is tested only if the `OSAL_ENABLE_ASSERTIONS` switch is enabled.

The remark string is not currently used except for putting a comment in the code about the assertion.

Parameters

in	<i>c</i>	the condition to be verified to be true
in	<i>remark</i>	a remark string

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.21 osalDbgCheck

```
#define osalDbgCheck(  
    c)
```

Value:

```
do {  
    /*lint -save -e506 -e774 [2.1, 14.3] Can be a constant by design.*/  
    if (OSAL_DBG_ENABLE_CHECKS != FALSE) {  
        if (!(c)) {  
            /*lint -restore*/  
            osalSysHalt(__func__);  
        }  
    }  
} while (false)
```

Function parameters check.

If the condition check fails then the OSAL panics and halts.

Note

The condition is tested only if the `OSAL_ENABLE_CHECKS` switch is enabled.

Parameters

in	c	the condition to be verified to be true

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.22 osalDbgCheckClassI

```
#define osalDbgCheckClassI()
```

I-Class state check.

Note

Implementation is optional.

7.2.14.2.23 osalDbgCheckClassS

```
#define osalDbgCheckClassS()
```

S-Class state check.

Note

Implementation is optional.

7.2.14.2.24 OSAL_IRQ_IS_VALID_PRIORITY

```
#define OSAL_IRQ_IS_VALID_PRIORITY(  
    n)
```

Value:

```
((n) >= OSAL_IRQ_MAXIMUM_PRIORITY) && ((n) < OSAL_IRQ_PRIORITY_LEVELS)
```

Priority level verification macro.

7.2.14.2.25 OSAL_IRQ_PROLOGUE

```
#define OSAL_IRQ_PROLOGUE()
```

IRQ prologue code.

This macro must be inserted at the start of all IRQ handlers.

7.2.14.2.26 OSAL_IRQ_EPILOGUE

```
#define OSAL_IRQ_EPILOGUE()
```

IRQ epilogue code.

This macro must be inserted at the end of all IRQ handlers.

7.2.14.2.27 OSAL_IRQ_HANDLER

```
#define OSAL_IRQ_HANDLER(  
    id)
```

Value:

```
void id(void)
```

IRQ handler function declaration.

This macro hides the details of an ISR function declaration.

Parameters

in	<i>id</i>	a vector name as defined in <code>vectors.s</code>

7.2.14.2.28 OSAL_S2I

```
#define OSAL_S2I(  
    secs)
```

Value:

```
((sysinterval_t)((time_conv_t)(secs) * (time_conv_t)OSAL_ST_FREQUENCY))
```

Seconds to time interval.

Converts from seconds to system ticks number.

Note

The result is rounded upward to the next tick boundary.

Use of this macro for large values is not secure because integer overflows, make sure your value can be correctly converted.

Parameters

in	<i>secs</i>	number of seconds

Returns

The number of ticks.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.29 OSAL_MS2I

```
#define OSAL_MS2I(  
    msec)
```

Value:

```
((sysinterval_t) (((time_conv_t) (msec) *  
    (time_conv_t) OSAL_ST_FREQUENCY) +  
    (time_conv_t) 999) / (time_conv_t) 1000))
```

Milliseconds to time interval.

Converts from milliseconds to system ticks number.

Note

The result is rounded upward to the next tick boundary.

Use of this macro for large values is not secure because integer overflows, make sure your value can be correctly converted.

Parameters

in	<i>msec</i>	number of milliseconds

Returns

The number of ticks.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.30 OSAL_US2I

```
#define OSAL_US2I(  
    usecs)
```

Value:

```
((sysinterval_t) (((time_conv_t) (usecs) *  
    (time_conv_t) OSAL_ST_FREQUENCY) +  
    (time_conv_t) 999999) / (time_conv_t) 1000000))
```

Microseconds to time interval.

Converts from microseconds to system ticks number.

Note

The result is rounded upward to the next tick boundary.

Use of this macro for large values is not secure because integer overflows, make sure your value can be correctly converted.

Parameters

in	<i>usecs</i>	number of microseconds

Returns

The number of ticks.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.31 OSAL_I2S

```
#define OSAL_I2S(  
    interval)
```

Value:

```
(time_secs_t)((time_conv_t)(interval) +  
               (time_conv_t)OSAL_ST_FREQUENCY -  
               (time_conv_t)1) / (time_conv_t)OSAL_ST_FREQUENCY
```

Time interval to seconds.

Converts from system ticks number to seconds.

Note

The result is rounded up to the next second boundary.

Use of this macro for large values is not secure because integer overflows, make sure your value can be correctly converted.

Parameters

in	<i>interval</i>	interval in ticks

Returns

The number of seconds.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.32 OSAL_I2MS

```
#define OSAL_I2MS(  
    interval)
```

Value:

```
(time_msecs_t)((time_conv_t)(interval) * (time_conv_t)1000) +  
               (time_conv_t)OSAL_ST_FREQUENCY - (time_conv_t)1) /  
               (time_conv_t)OSAL_ST_FREQUENCY
```

Time interval to milliseconds.

Converts from system ticks number to milliseconds.

Note

The result is rounded up to the next millisecond boundary.

Use of this macro for large values is not secure because integer overflows, make sure your value can be correctly converted.

Parameters

in	<i>interval</i>	interval in ticks

Returns

The number of milliseconds.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.33 OSAL_I2US

```
#define OSAL_I2US(  
    interval)
```

Value:

```
((time_msecs_t) (((time_conv_t)(interval) * (time_conv_t)1000000) +  
    (time_conv_t)OSAL_ST_FREQUENCY - (time_conv_t)1) /  
    (time_conv_t)OSAL_ST_FREQUENCY) \
```

Time interval to microseconds.

Converts from system ticks number to microseconds.

Note

The result is rounded up to the next microsecond boundary.

Use of this macro for large values is not secure because integer overflows, make sure your value can be correctly converted.

Parameters

in	<i>interval</i>	interval in ticks

Returns

The number of microseconds.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.34 OSAL_S2RTC

```
#define OSAL_S2RTC(  
    freq,  
    sec)
```

Value:

```
((freq) * (sec))
```

Seconds to realtime counter.

Converts from seconds to realtime counter cycles.

Note

The macro assumes that *freq* >= 1.

Parameters

in	<i>freq</i>	clock frequency, in Hz, of the realtime counter
in	<i>sec</i>	number of seconds

Returns

The number of cycles.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.35 OSAL_MS2RTC

```
#define OSAL_MS2RTC(  
    freq,  
    msec)
```

Value:

```
(rtcnt_t) (((freq) + 999UL) / 1000UL) * (msec)
```

Milliseconds to realtime counter.

Converts from milliseconds to realtime counter cycles.

Note

The result is rounded upward to the next millisecond boundary.

The macro assumes that *freq* >= 1000.

Parameters

in	<i>freq</i>	clock frequency, in Hz, of the realtime counter
in	<i>msec</i>	number of milliseconds

Returns

The number of cycles.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.36 OSAL_US2RTC

```
#define OSAL_US2RTC(  
    freq,  
    usec)
```

Value:

```
(rtcnt_t) (((freq) + 999999UL) / 1000000UL) * (usec)
```

Microseconds to realtime counter.

Converts from microseconds to realtime counter cycles.

Note

The result is rounded upward to the next microsecond boundary.

The macro assumes that *freq* >= 1000000.

Parameters

in	<i>freq</i>	clock frequency, in Hz, of the realtime counter
in	<i>usec</i>	number of microseconds

Returns

The number of cycles.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.37 osalThreadSleepSeconds

```
#define osalThreadSleepSeconds(  
    secs)
```

Value:

```
osalThreadSleep(OSAL_S2I(secs))
```

Delays the invoking thread for the specified number of seconds.

Note

The specified time is rounded up to a value allowed by the real system tick clock.
The maximum specifiable value is implementation dependent.

Parameters

in	<i>secs</i>	time in seconds, must be different from zero

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.38 osalThreadSleepMilliseconds

```
#define osalThreadSleepMilliseconds(  
    msec)
```

Value:

```
osalThreadSleep(OSAL_MS2I(msec))
```

Delays the invoking thread for the specified number of milliseconds.

Note

The specified time is rounded up to a value allowed by the real system tick clock.
The maximum specifiable value is implementation dependent.

Parameters

in	<i>msecs</i>	time in milliseconds, must be different from zero

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.2.39 osalThreadSleepMicroseconds

```
#define osalThreadSleepMicroseconds(  
    usecs)
```

Value:

```
osalThreadSleep(OSAL_US2I(usecs))
```

Delays the invoking thread for the specified number of microseconds.

Note

The specified time is rounded up to a value allowed by the real system tick clock.

The maximum specifiable value is implementation dependent.

Parameters

in	<i>usecs</i>	time in microseconds, must be different from zero

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.3 Typedef Documentation**7.2.14.3.1 syssts_t**

```
typedef uint32_t syssts_t
```

Type of a system status word.

7.2.14.3.2 msg_t

```
typedef int32_t msg_t
```

Type of a message.

7.2.14.3.3 `systime_t`

```
typedef uint32_t systime_t
```

Type of system time counter.

7.2.14.3.4 `sysinterval_t`

```
typedef uint32_t sysinterval_t
```

Type of system time interval.

7.2.14.3.5 `time_conv_t`

```
typedef uint64_t time_conv_t
```

Type of time conversion variable.

Note

This type must have double width than other time types, it is only used internally for conversions.

7.2.14.3.6 `rtcnt_t`

```
typedef uint32_t rtcnt_t
```

Type of realtime counter.

7.2.14.3.7 `thread_reference_t`

```
typedef void* thread_reference_t
```

Type of a thread reference.

7.2.14.3.8 `eventflags_t`

```
typedef uint32_t eventflags_t
```

Type of an event flags mask.

7.2.14.3.9 `event_source_t`

```
typedef struct event_source event_source_t
```

Type of an event flags object.

Note

The content of this structure is not part of the API and should not be relied upon. Implementers may define this structure in an entirely different way.

Retrieval and clearing of the flags are not defined in this API and are implementation-dependent.

7.2.14.3.10 eventcallback_t

```
typedef void(* eventcallback_t) (event_source_t *esp)
```

Type of an event source callback.

Note

This type is not part of the OSAL API and is provided exclusively as an example and for convenience.

7.2.14.3.11 mutex_t

```
typedef uint32_t mutex_t
```

Type of a mutex.

Note

If the OS does not support mutexes or there is no OS then them mechanism can be simulated.

7.2.14.4 Function Documentation**7.2.14.4.1 osalInit()**

```
void osalInit (
    void )
```

OSAL module initialization.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.4.2 osalSysHalt()

```
void osalSysHalt (
    const char * reason)
```

System halt with error message.

Parameters

in	<i>reason</i>	the halt message pointer

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

Here is the call graph for this function:



7.2.14.4.3 osalSysPolledDelayX()

```
void osalSysPolledDelayX (
    rtcnt_t cycles)
```

Polled delay.

Note

The real delay is always few cycles in excess of the specified value.

Parameters

in	<i>cycles</i>	number of cycles

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.14.4.4 osalOsTimerHandlerI()

```
void osalOsTimerHandlerI (
    void )
```

System timer handler.

The handler is used for scheduling and Virtual Timers management.

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.14.4.5 osalOsRescheduleS()

```
void osalOsRescheduleS (
    void )
```

Checks if a reschedule is required and performs it.

Note

I-Class functions invoked from thread context must not reschedule by themselves, an explicit reschedule using this function is required in this scenario.

Not implemented in this simplified OSAL.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

7.2.14.4.6 osalOsGetSystemTimeX()

```
sysinterval_t osalOsGetSystemTimeX (
    void )
```

Current system time.

Returns the number of system ticks since the `osalInit()` invocation.

Note

The counter can reach its maximum and then restart from zero.

This function can be called from any context but its atomicity is not guaranteed on architectures whose word size is less than `sysinterval_t` size.

Returns

The system time in ticks.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.14.4.7 osalThreadSleepS()

```
void osalThreadSleepS (
    sysinterval_t time)
```

Suspends the invoking thread for the specified time.

Parameters

in	time	the delay in system ticks, the special values are handled as follow:
		• <code>TIME_INFINITE</code> is allowed but interpreted as a normal time specification. • <code>TIME_IMMEDIATE</code> this value is not allowed.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

7.2.14.4.8 osalThreadSleep()

```
void osalThreadSleep (
    sysinterval_t time)
```

Suspends the invoking thread for the specified time.

Parameters

in	<i>time</i>	the delay in system ticks, the special values are handled as follow: • <i>TIME_INFINITE</i> is allowed but interpreted as a normal time specification. • <i>TIME_IMMEDIATE</i> this value is not allowed.

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.4.9 **osalThreadSuspendS()**

```
msg_t osalThreadSuspendS (
    thread_reference_t * trp)
```

Sends the current thread sleeping and sets a reference variable.

Note

This function must reschedule, it can only be called from thread context.

Parameters

in	<i>trp</i>	a pointer to a thread reference object

Returns

The wake up message.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

7.2.14.4.10 **osalThreadSuspendTimeoutS()**

```
msg_t osalThreadSuspendTimeoutS (
    thread_reference_t * trp,
    sysinterval_t timeout)
```

Sends the current thread sleeping and sets a reference variable.

Note

This function must reschedule, it can only be called from thread context.

Parameters

in	<i>trp</i>	a pointer to a thread reference object
in	<i>timeout</i>	the timeout in system ticks, the special values are handled as follow: • <i>TIME_INFINITE</i> the thread enters an infinite sleep state. • <i>TIME_IMMEDIATE</i> the thread is not enqueued and the function returns <i>MSG_TIMEOUT</i> as if a timeout occurred.

Returns

The wake up message.

Return values

<i>MSG_TIMEOUT</i>	if the operation timed out.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

7.2.14.4.11 **osalThreadResumeI()**

```
void osalThreadResumeI (
    thread_reference_t * trp,
    msg_t msg)
```

Wakes up a thread waiting on a thread reference object.

Note

This function must not reschedule because it can be called from ISR context.

Parameters

in	<i>trp</i>	a pointer to a thread reference object
in	<i>msg</i>	the message code

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.14.4.12 **osalThreadResumeS()**

```
void osalThreadResumeS (
    thread_reference_t * trp,
    msg_t msg)
```

Wakes up a thread waiting on a thread reference object.

Note

This function must reschedule, it can only be called from thread context.

Parameters

in	<i>trp</i>	a pointer to a thread reference object
in	<i>msg</i>	the message code

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.14.4.13 osalThreadEnqueueTimeoutS()

```
msg_t osalThreadEnqueueTimeoutS (
    threads_queue_t * tqp,
    sysinterval_t timeout)
```

Enqueues the caller thread.

The caller thread is enqueued and put to sleep until it is dequeued or the specified timeouts expires.

Parameters

in	<i>tqp</i>	pointer to the threads queue object
in	<i>timeout</i>	the timeout in system ticks, the special values are handled as follow: • <i>TIME_INFINITE</i> the thread enters an infinite sleep state. • <i>TIME_IMMEDIATE</i> the thread is not enqueued and the function returns <i>MSG_TIMEOUT</i> as if a timeout occurred.

Returns

The message from *osalQueueWakeupOneI()* or *osalQueueWakeupAllI()* functions.

Return values

<i>MSG_TIMEOUT</i>	if the thread has not been dequeued within the specified timeout or if the function has been invoked with <i>TIME_IMMEDIATE</i> as timeout specification.

Function Class:

This is an **S-Class** API, this function can be invoked from within a system lock zone by threads only.

7.2.14.4.14 osalThreadDequeueNextI()

```
void osalThreadDequeueNextI (
    threads_queue_t * tqp,
    msg_t msg)
```

Dequeues and wakes up one thread from the queue, if any.

Parameters

in	<i>tqp</i>	pointer to the threads queue object
in	<i>msg</i>	the message code

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.14.4.15 osalThreadDequeueAllI()

```
void osalThreadDequeueAllI (  
    threads_queue_t * tqp,  
    msg_t msg)
```

Dequeues and wakes up all threads from the queue.

Parameters

in	<i>tqp</i>	pointer to the threads queue object
in	<i>msg</i>	the message code

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.14.4.16 osalEventBroadcastFlagsI()

```
void osalEventBroadcastFlagsI (  
    event_source_t * esp,  
    eventflags_t flags)
```

Add flags to an event source object.

Parameters

in	<i>esp</i>	pointer to the event flags object
in	<i>flags</i>	flags to be ORed to the flags mask

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

7.2.14.4.17 osalEventBroadcastFlags()

```
void osalEventBroadcastFlags (  
    event_source_t * esp,  
    eventflags_t flags)
```

Add flags to an event source object.

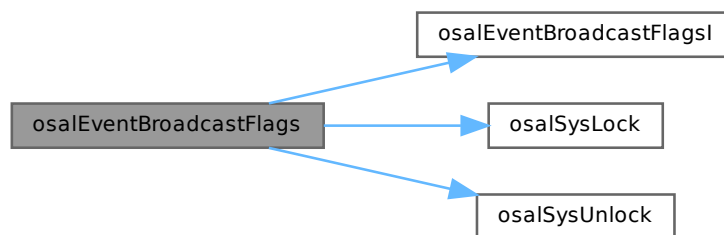
Parameters

in	<i>esp</i>	pointer to the event flags object
in	<i>flags</i>	flags to be ORed to the flags mask

Function Class:

This is an **I-Class** API, this function can be invoked from within a system lock zone by both threads and interrupt handlers.

Here is the call graph for this function:

**7.2.14.4.18 osalEventSetCallback()**

```

void osalEventSetCallback (
    event_source_t * esp,
    eventcallback_t cb,
    void * param)
  
```

Event callback setup.

Note

The callback is invoked from ISR context and can only invoke I-Class functions. The callback is meant to wakeup the task that will handle the event by calling `osalEventGetAndClearFlagsI()`.

This function is not part of the OSAL API and is provided exclusively as an example and for convenience.

Parameters

in	<i>esp</i>	pointer to the event flags object
in	<i>cb</i>	pointer to the callback function
in	<i>param</i>	parameter to be passed to the callback function

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.4.19 osalMutexLock()

```
void osalMutexLock (  
    mutex_t * mp)
```

Locks the specified mutex.

Postcondition

The mutex is locked and inserted in the per-thread stack of owned mutexes.

Parameters

in, out	<i>mp</i>	pointer to the <code>mutex_t</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.4.20 osalMutexUnlock()

```
void osalMutexUnlock (  
    mutex_t * mp)
```

Unlocks the specified mutex.

Note

The HAL guarantees to release mutex in reverse lock order. The mutex being unlocked is guaranteed to be the last locked mutex by the invoking thread. The implementation can rely on this behavior and eventually ignore the `mp` parameter which is supplied in order to support those OSes not supporting a stack of the owned mutexes.

Parameters

in, out	<i>mp</i>	pointer to the <code>mutex_t</code> object

Function Class:

Normal API, this function can be invoked by regular system threads but not from within a lock zone.

7.2.14.4.21 osalSysDisable()

```
void osalSysDisable (  
    void ) [inline], [static]
```

Disables interrupts globally.

Function Class:

Special function, this function has special requirements see the notes.

7.2.14.4.22 osalSysEnable()

```
void osalSysEnable (
    void ) [inline], [static]
```

Enables interrupts globally.

Function Class:

Special function, this function has special requirements see the notes.

7.2.14.4.23 osalSysLock()

```
void osalSysLock (
    void ) [inline], [static]
```

Enters a critical zone from thread context.

Note

This function cannot be used for reentrant critical zones.

Function Class:

Special function, this function has special requirements see the notes.

7.2.14.4.24 osalSysUnlock()

```
void osalSysUnlock (
    void ) [inline], [static]
```

Leaves a critical zone from thread context.

Note

This function cannot be used for reentrant critical zones.

Function Class:

Special function, this function has special requirements see the notes.

7.2.14.4.25 osalSysLockFromISR()

```
void osalSysLockFromISR (
    void ) [inline], [static]
```

Enters a critical zone from ISR context.

Note

This function cannot be used for reentrant critical zones.

Function Class:

Special function, this function has special requirements see the notes.

7.2.14.4.26 `osalSysUnlockFromISR()`

```
void osalSysUnlockFromISR (  
    void ) [inline], [static]
```

Leaves a critical zone from ISR context.

Note

This function cannot be used for reentrant critical zones.

Function Class:

Special function, this function has special requirements see the notes.

7.2.14.4.27 `osalSysGetStatusAndLockX()`

```
syssts_t osalSysGetStatusAndLockX (  
    void ) [inline], [static]
```

Returns the execution status and enters a critical zone.

This functions enters into a critical zone and can be called from any context. Because its flexibility it is less efficient than `chSysLock()` which is preferable when the calling context is known.

Postcondition

The system is in a critical zone.

Returns

The previous system status, the encoding of this status word is architecture-dependent and opaque.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.14.4.28 `osalSysRestoreStatusX()`

```
void osalSysRestoreStatusX (  
    syssts_t sts) [inline], [static]
```

Restores the specified execution status and leaves a critical zone.

Note

A call to `chSchRescheduleS()` is automatically performed if exiting the critical zone and if not in ISR context.

Parameters

in	<i>sts</i>	the system status to be restored.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.14.4.29 osalTimeAddX()

```
sys_time_t osalTimeAddX (  
    sys_time_t system,  
    sysinterval_t interval) [inline], [static]
```

Adds an interval to a system time returning a system time.

Parameters

in	<i>system</i>	base system time
in	<i>interval</i>	interval to be added

Returns

The new system time.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.14.4.30 osalTimeDiffX()

```
sysinterval_t osalTimeDiffX (  
    sys_time_t start,  
    sys_time_t end) [inline], [static]
```

Subtracts two system times returning an interval.

Parameters

in	<i>start</i>	first system time
in	<i>end</i>	second system time

Returns

The interval representing the time difference.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.14.4.31 osalTimeIsInRangeX()

```
bool osalTimeIsInRangeX (
    systime_t time,
    systime_t start,
    systime_t end) [inline], [static]
```

Checks if the specified time is within the specified time window.

Note

When start==end then the function returns always false because the time window has zero size.

This function can be called from any context.

Parameters

in	<i>time</i>	the time to be verified
in	<i>start</i>	the start of the time window (inclusive)
in	<i>end</i>	the end of the time window (non inclusive)

Return values

<i>true</i>	current time within the specified time window.
<i>false</i>	current time not within the specified time window.

Function Class:

This is an **X-Class** API, this function can be invoked from any context.

7.2.14.4.32 osalThreadQueueObjectInit()

```
void osalThreadQueueObjectInit (
    threads_queue_t * tqp) [inline], [static]
```

Initializes a threads queue object.

Parameters

out	<i>tqp</i>	pointer to the threads queue object

Function Class:

Object or module initializer function.

7.2.14.4.33 osalEventObjectInit()

```
void osalEventObjectInit (
    event_source_t * esp) [inline], [static]
```

Initializes an event source object.

Parameters

out	esp	pointer to the event source object

Function Class:

Object or module initializer function.

7.2.14.4.34 osalMutexObjectInit()

```
void osalMutexObjectInit (
    mutex_t * mp) [inline], [static]
```

Initializes a `mutex_t` object.

Parameters

out	mp	pointer to the <code>mutex_t</code> object

Function Class:

Object or module initializer function.

7.2.14.5 Variable Documentation**7.2.14.5.1 osal_halt_msg [1/2]**

```
const char* osal_halt_msg
```

Pointer to a halt error message.

Note

The message is meant to be retrieved by the debugger after the system halt caused by an unexpected error.

7.2.14.5.2 osal_halt_msg [2/2]

```
const char* osal_halt_msg [extern]
```

Pointer to a halt error message.

Note

The message is meant to be retrieved by the debugger after the system halt caused by an unexpected error.

Chapter 8

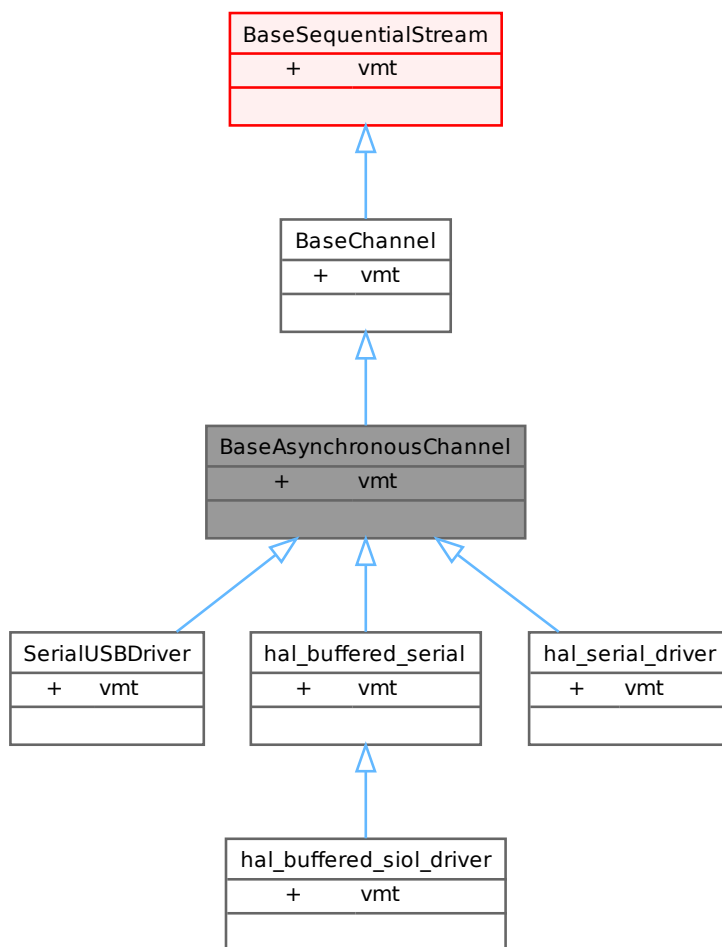
Data Structure Documentation

8.1 BaseAsynchronousChannel Struct Reference

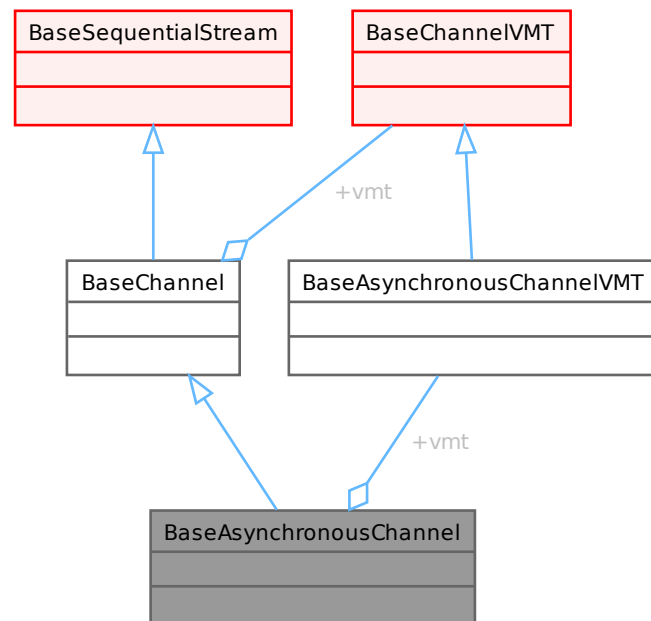
Base asynchronous channel class.

```
#include <hal_channels.h>
```

Inheritance diagram for BaseAsynchronousChannel:



Collaboration diagram for BaseAsynchronousChannel:



Data Fields

- const struct [BaseAsynchronousChannelVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseChannel](#)

- const struct [BaseChannelVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseSequentialStream](#)

- const struct [BaseSequentialStreamVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.1.1 Detailed Description

Base asynchronous channel class.

This class extends [BaseChannel](#) by adding event sources fields for asynchronous I/O for use in an event-driven environment.

8.1.2 Field Documentation

8.1.2.1 vmt

```
const struct BaseAsynchronousChannelVMT* BaseAsynchronousChannel::vmt
```

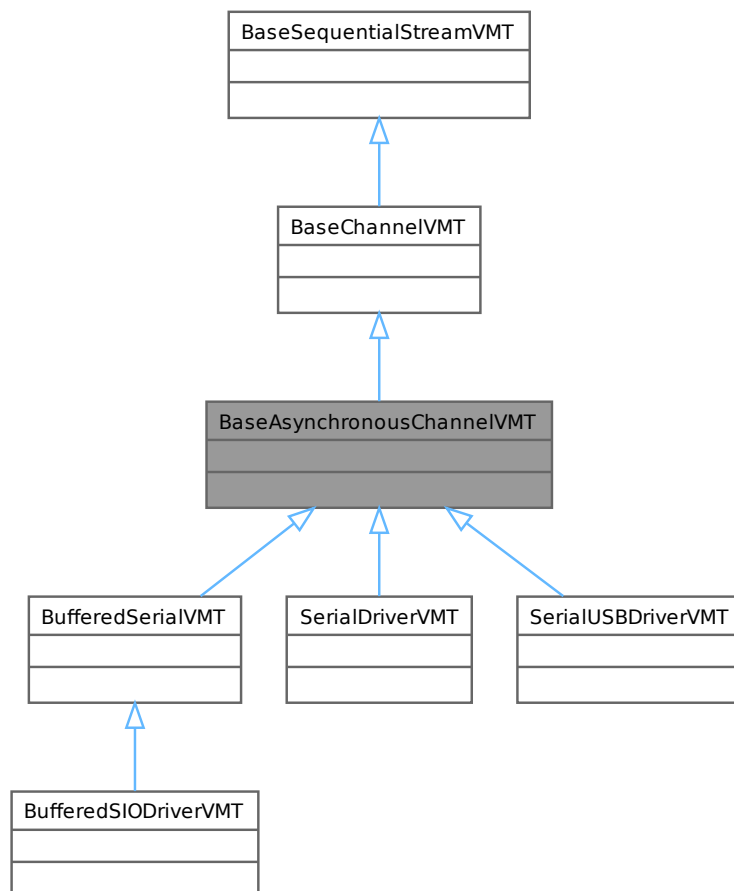
Virtual Methods Table.

8.2 BaseAsynchronousChannelVMT Struct Reference

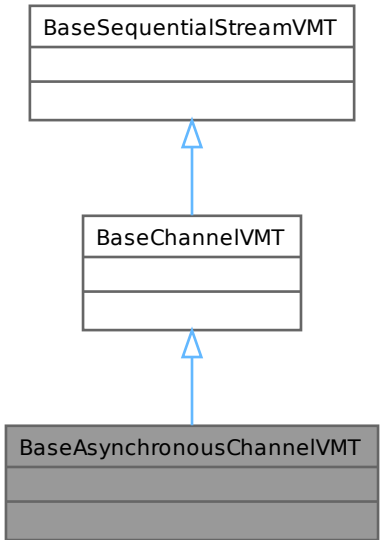
[BaseAsynchronousChannel](#) virtual methods table.

```
#include <hal_channels.h>
```

Inheritance diagram for BaseAsynchronousChannelVMT:



Collaboration diagram for BaseAsynchronousChannelVMT:



8.2.1 Detailed Description

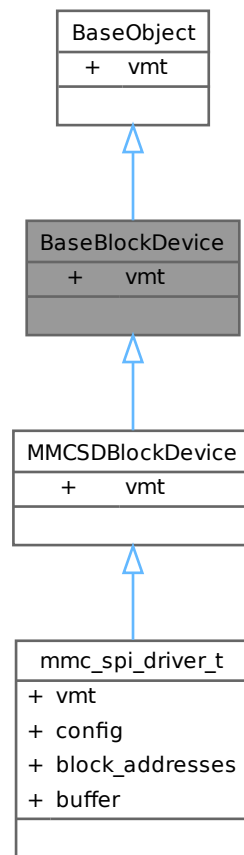
BaseAsynchronousChannel virtual methods table.

8.3 BaseBlockDevice Struct Reference

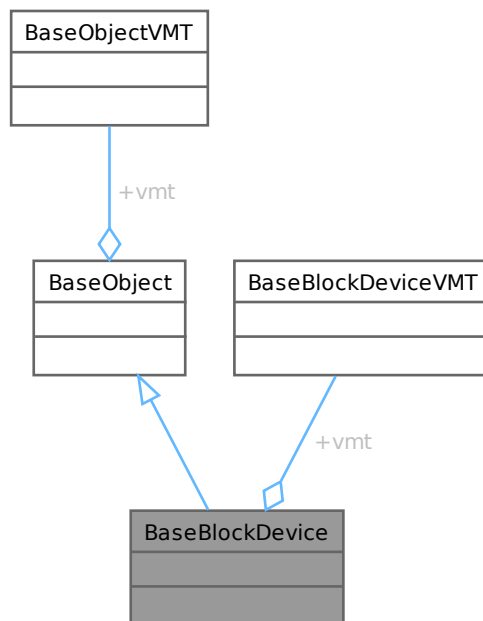
Base block device class.

```
#include <hal_ioblock.h>
```

Inheritance diagram for BaseBlockDevice:



Collaboration diagram for BaseBlockDevice:



Data Fields

- const struct [BaseBlockDeviceVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.3.1 Detailed Description

Base block device class.

This class represents a generic, block-accessible, device.

8.3.2 Field Documentation

8.3.2.1 vmt

```
const struct BaseBlockDeviceVMT* BaseBlockDevice::vmt
```

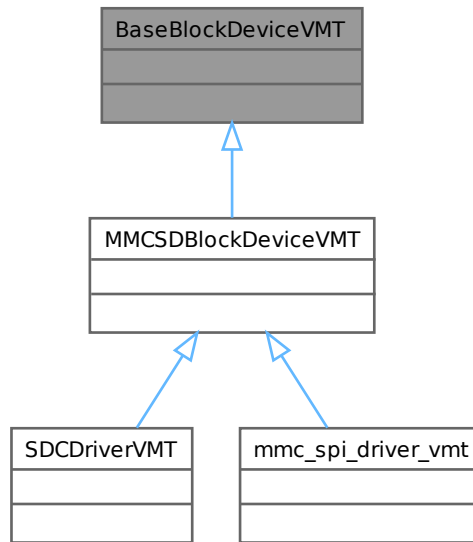
Virtual Methods Table.

8.4 BaseBlockDeviceVMT Struct Reference

[BaseBlockDevice](#) virtual methods table.

```
#include <hal_ioblock.h>
```

Inheritance diagram for BaseBlockDeviceVMT:



Collaboration diagram for BaseBlockDeviceVMT:



8.4.1 Detailed Description

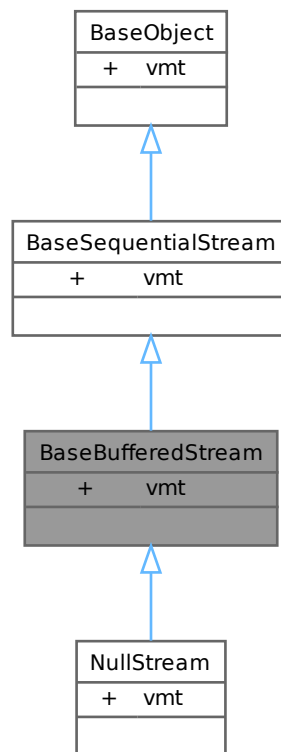
[BaseBlockDevice](#) virtual methods table.

8.5 BaseBufferedStream Struct Reference

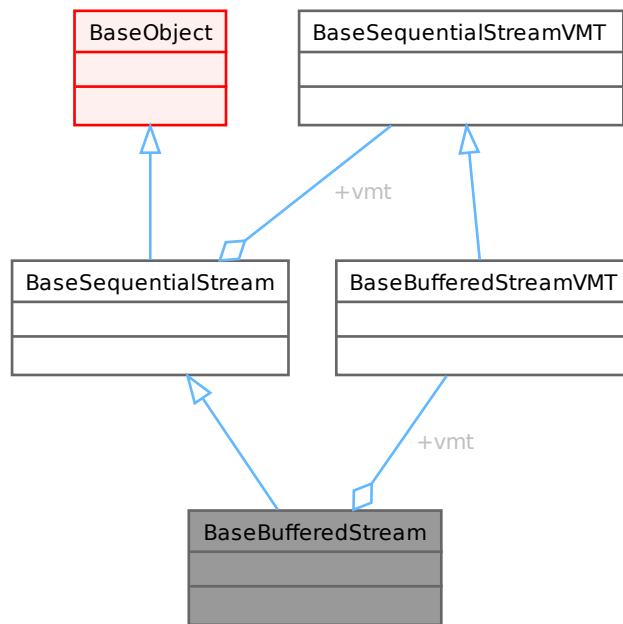
Buffered stream class.

```
#include <hal_streams.h>
```

Inheritance diagram for BaseBufferedStream:



Collaboration diagram for `BaseBufferedStream`:



Data Fields

- const struct `BaseBufferedStreamVMT` * `vmt`
Virtual Methods Table.

Data Fields inherited from `BaseSequentialStream`

- const struct `BaseSequentialStreamVMT` * `vmt`
Virtual Methods Table.

Data Fields inherited from `BaseObject`

- const struct `BaseObjectVMT` * `vmt`
Virtual Methods Table.

8.5.1 Detailed Description

Buffered stream class.

This class extends `BaseSequentialStream` to represent a generic blocking buffered sequential data stream.

8.5.2 Field Documentation

8.5.2.1 vmt

```
const struct BaseBufferedStreamVMT* BaseBufferedStream::vmt
```

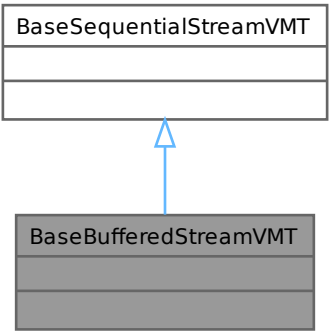
Virtual Methods Table.

8.6 BaseBufferedStreamVMT Struct Reference

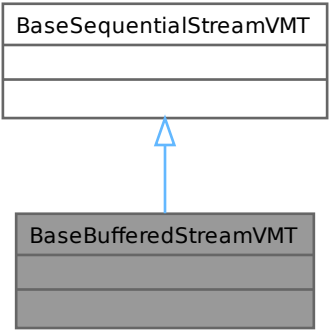
BaseBufferedStream virtual methods table.

```
#include <hal_streams.h>
```

Inheritance diagram for BaseBufferedStreamVMT:



Collaboration diagram for BaseBufferedStreamVMT:



8.6.1 Detailed Description

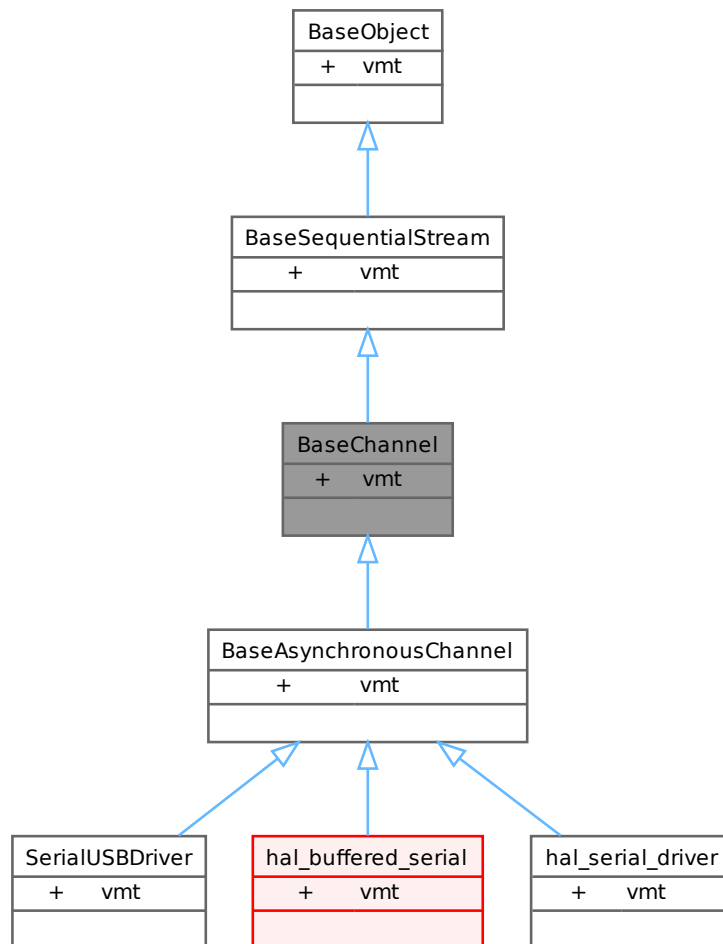
[BaseBufferedStream](#) virtual methods table.

8.7 BaseChannel Struct Reference

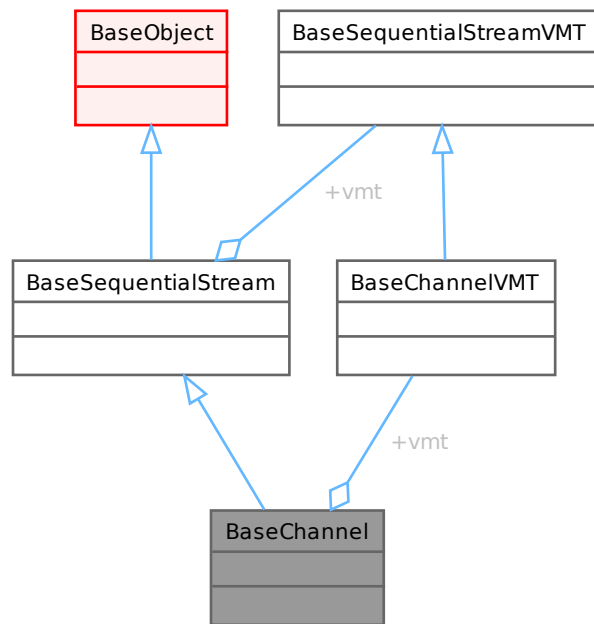
Base channel class.

```
#include <hal_channels.h>
```

Inheritance diagram for BaseChannel:



Collaboration diagram for BaseChannel:



Data Fields

- const struct [BaseChannelVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseSequentialStream](#)

- const struct [BaseSequentialStreamVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.7.1 Detailed Description

Base channel class.

This class represents a generic, byte-wide, I/O channel. This class introduces generic I/O primitives with timeout specification.

8.7.2 Field Documentation

8.7.2.1 vmt

```
const struct BaseChannelVMT* BaseChannel::vmt
```

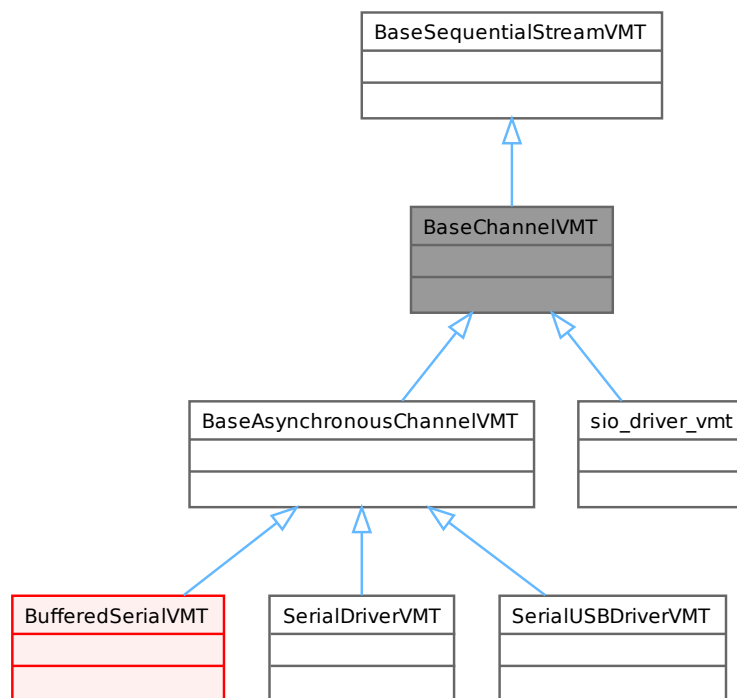
Virtual Methods Table.

8.8 BaseChannelVMT Struct Reference

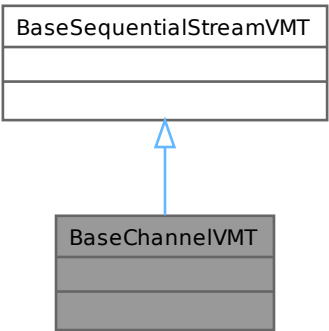
[BaseChannel](#) virtual methods table.

```
#include <hal_channels.h>
```

Inheritance diagram for BaseChannelVMT:



Collaboration diagram for BaseChannelVMT:



8.8.1 Detailed Description

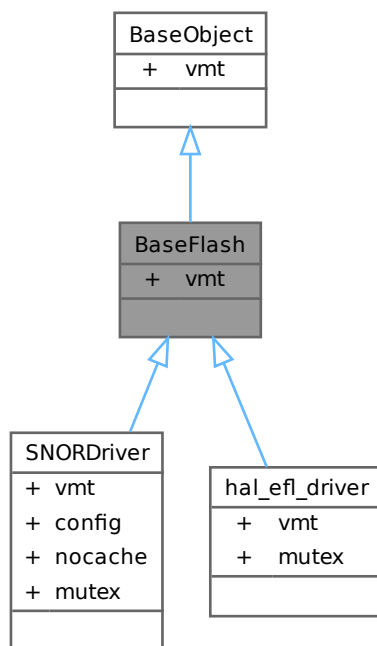
`BaseChannel` virtual methods table.

8.9 BaseFlash Struct Reference

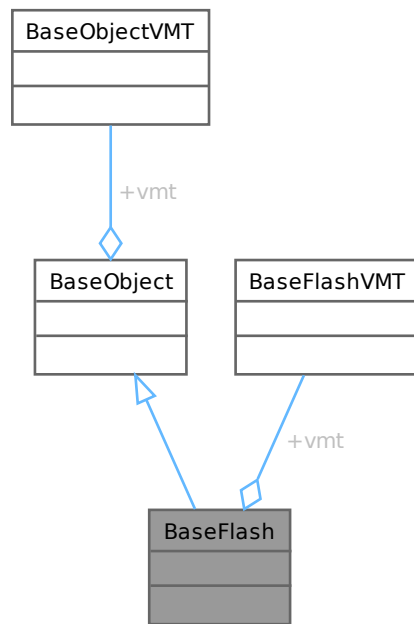
Base flash class.

```
#include <hal_flash.h>
```

Inheritance diagram for BaseFlash:



Collaboration diagram for BaseFlash:



Data Fields

- const struct [BaseFlashVMT](#) * [vmt](#)
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * [vmt](#)
Virtual Methods Table.

8.9.1 Detailed Description

Base flash class.

8.9.2 Field Documentation

8.9.2.1 vmt

```
const struct BaseFlashVMT* BaseFlash::vmt
```

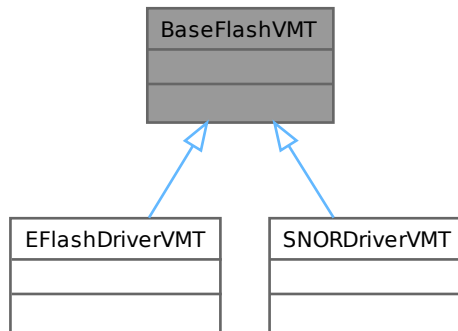
Virtual Methods Table.

8.10 BaseFlashVMT Struct Reference

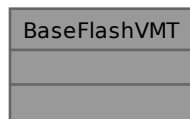
[BaseFlash](#) virtual methods table.

```
#include <hal_flash.h>
```

Inheritance diagram for BaseFlashVMT:



Collaboration diagram for BaseFlashVMT:



8.10.1 Detailed Description

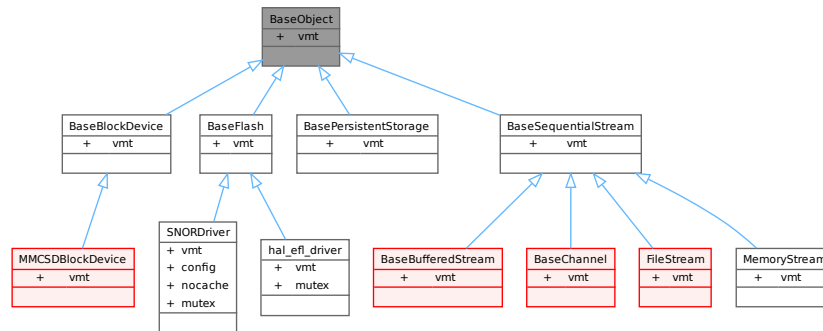
[BaseFlash](#) virtual methods table.

8.11 BaseObject Struct Reference

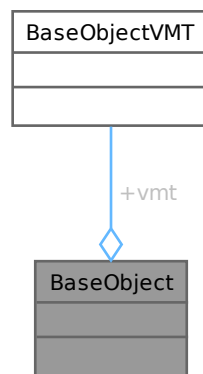
Base object class.

```
#include <hal_objects.h>
```

Inheritance diagram for BaseObject:



Collaboration diagram for BaseObject:



Data Fields

- const struct [BaseObjectVMT](#) * **vmt**
Virtual Methods Table.

8.11.1 Detailed Description

Base object class.

This class represents a generic object including a virtual methods table (VMT).

8.11.2 Field Documentation

8.11.2.1 vmt

```
const struct BaseObjectVMT* BaseObject::vmt
```

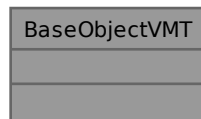
Virtual Methods Table.

8.12 BaseObjectVMT Struct Reference

[BaseObject](#) virtual methods table.

```
#include <hal_objects.h>
```

Collaboration diagram for BaseObjectVMT:



8.12.1 Detailed Description

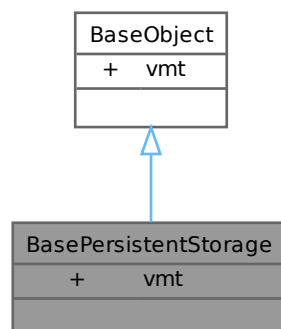
[BaseObject](#) virtual methods table.

8.13 BasePersistentStorage Struct Reference

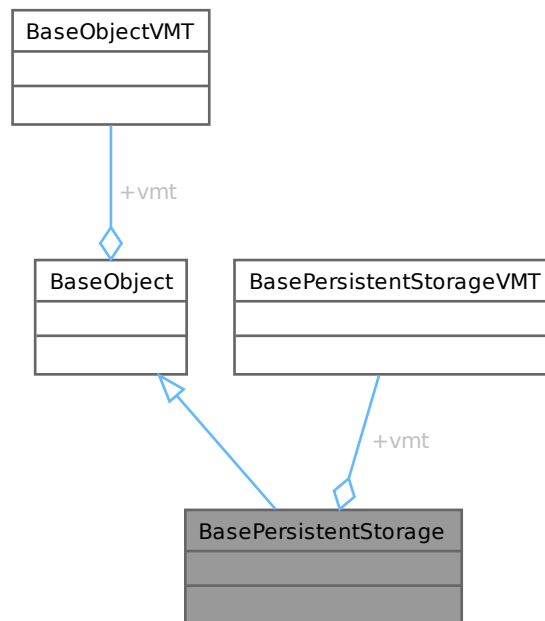
Base persistent storage class.

```
#include <hal_persistent.h>
```

Inheritance diagram for BasePersistentStorage:



Collaboration diagram for BasePersistentStorage:



Data Fields

- const struct [BasePersistentStorageVMT](#) * *vmt*
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * *vmt*
Virtual Methods Table.

8.13.1 Detailed Description

Base persistent storage class.

8.13.2 Field Documentation

8.13.2.1 *vmt*

```
const struct BasePersistentStorageVMT* BasePersistentStorage::vmt
```

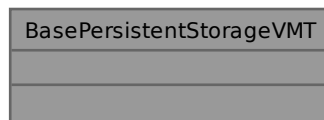
Virtual Methods Table.

8.14 BasePersistentStorageVMT Struct Reference

[BasePersistentStorage](#) virtual methods table.

```
#include <hal_persistent.h>
```

Collaboration diagram for BasePersistentStorageVMT:



8.14.1 Detailed Description

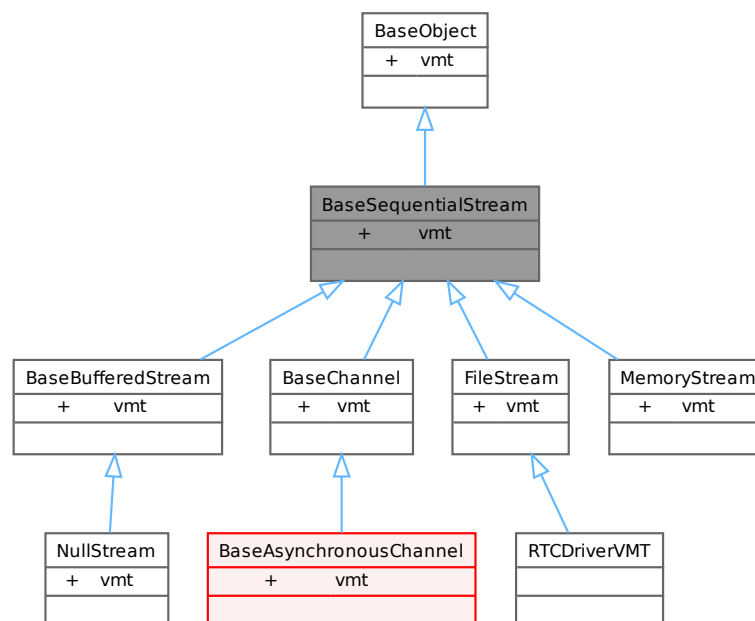
[BasePersistentStorage](#) virtual methods table.

8.15 BaseSequentialStream Struct Reference

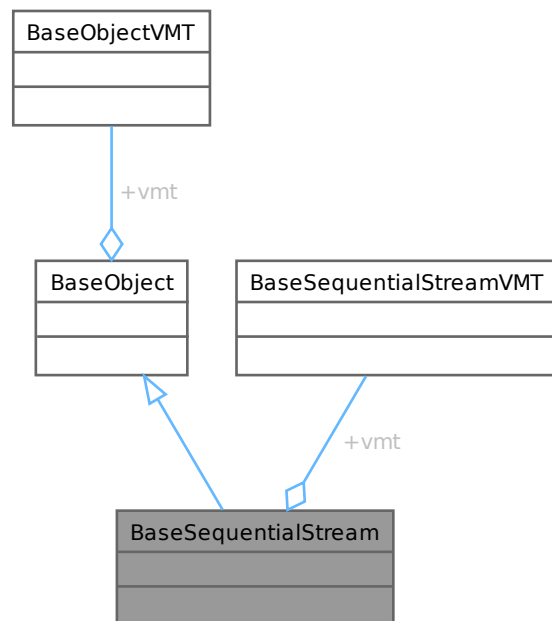
Base stream class.

```
#include <hal_streams.h>
```

Inheritance diagram for BaseSequentialStream:



Collaboration diagram for BaseSequentialStream:



Data Fields

- const struct [BaseSequentialStreamVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * `vmt`
Virtual Methods Table.

8.15.1 Detailed Description

Base stream class.

This class represents a generic blocking unbuffered sequential data stream.

8.15.2 Field Documentation

8.15.2.1 `vmt`

```
const struct BaseSequentialStreamVMT* BaseSequentialStream::vmt
```

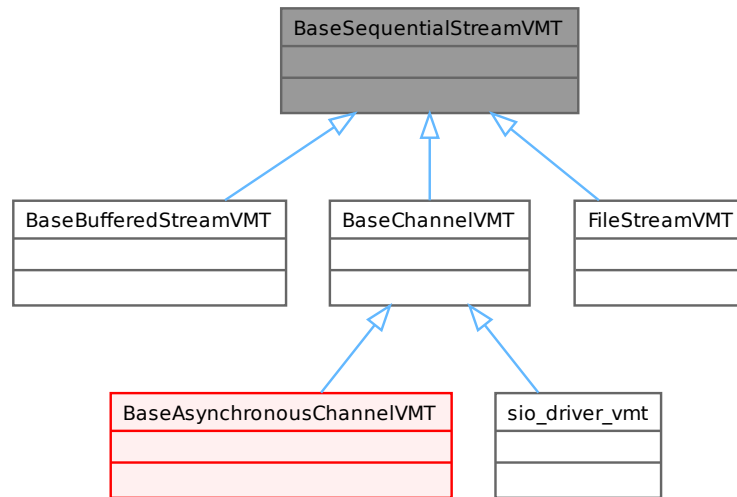
Virtual Methods Table.

8.16 BaseSequentialStreamVMT Struct Reference

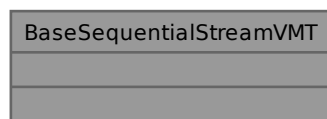
[BaseSequentialStream](#) virtual methods table.

```
#include <hal_streams.h>
```

Inheritance diagram for BaseSequentialStreamVMT:



Collaboration diagram for BaseSequentialStreamVMT:



8.16.1 Detailed Description

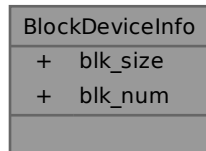
[BaseSequentialStream](#) virtual methods table.

8.17 BlockDeviceInfo Struct Reference

Block device info.

```
#include <hal_ioblock.h>
```

Collaboration diagram for BlockDeviceInfo:



Data Fields

- uint32_t [blk_size](#)
Block size in bytes.
- uint32_t [blk_num](#)
Total number of blocks.

8.17.1 Detailed Description

Block device info.

8.17.2 Field Documentation

8.17.2.1 blk_size

```
uint32_t BlockDeviceInfo::blk_size
```

Block size in bytes.

8.17.2.2 blk_num

```
uint32_t BlockDeviceInfo::blk_num
```

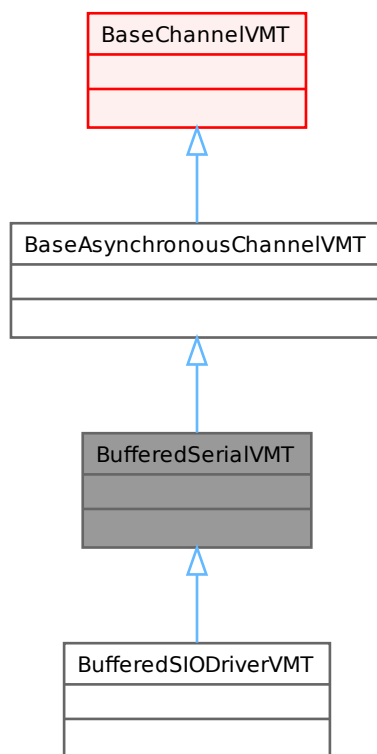
Total number of blocks.

8.18 BufferedSerialVMT Struct Reference

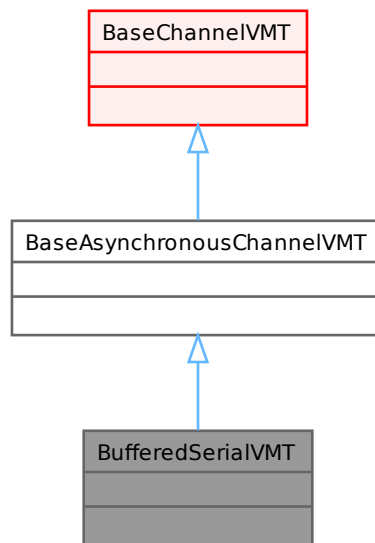
[BufferedSerial](#) virtual methods table.

```
#include <hal_buffered_serial.h>
```

Inheritance diagram for BufferedSerialVMT:



Collaboration diagram for BufferedSerialVMT:



8.18.1 Detailed Description

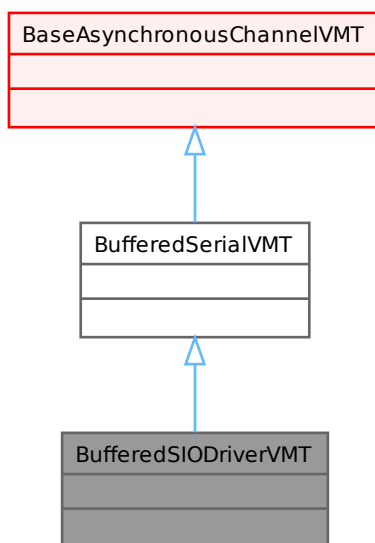
[BufferedSerial](#) virtual methods table.

8.19 BufferedSIODriverVMT Struct Reference

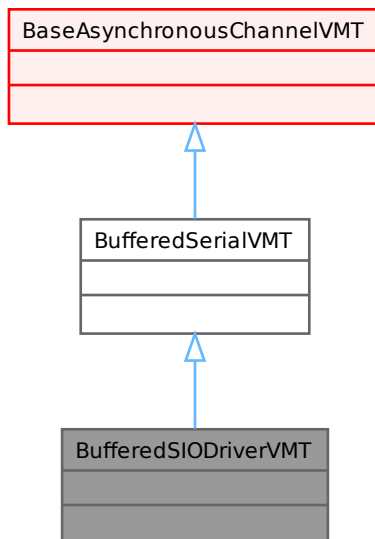
[BufferedSIODriver](#) virtual methods table.

```
#include <hal_buffered_sio.h>
```

Inheritance diagram for BufferedSIODriverVMT:



Collaboration diagram for BufferedSIODriverVMT:



8.19.1 Detailed Description

[BufferedSIODriver](#) virtual methods table.

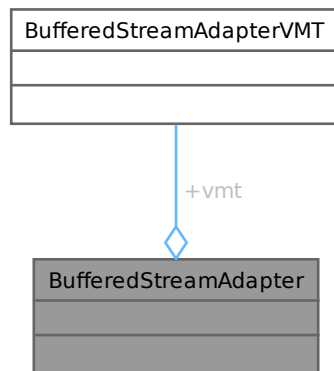
8.20 BufferedStreamAdapter Struct Reference

Buffered stream adapter object.

```
#include <bufstreams.h>
```

Inherits BufferedStream.

Collaboration diagram for BufferedStreamAdapter:



Data Fields

- const struct [BufferedStreamAdapterVMT](#) * vmt
Virtual Methods Table.

8.20.1 Detailed Description

Buffered stream adapter object.

8.20.2 Field Documentation

8.20.2.1 vmt

```
const struct BufferedStreamAdapterVMT* BufferedStreamAdapter::vmt
```

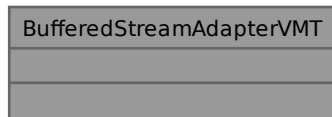
Virtual Methods Table.

8.21 BufferedStreamAdapterVMT Struct Reference

[BufferedStreamAdapter](#) virtual methods table.

```
#include <bufstreams.h>
```

Collaboration diagram for BufferedStreamAdapterVMT:



8.21.1 Detailed Description

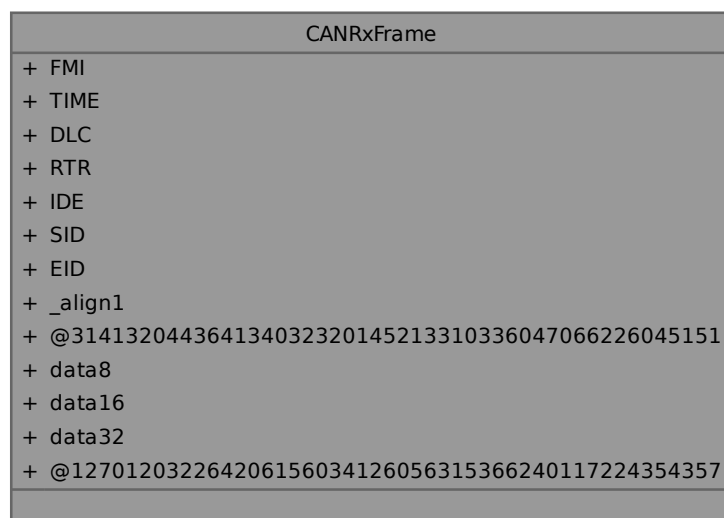
[BufferedStreamAdapter](#) virtual methods table.

8.22 CANRxFrame Struct Reference

CAN received frame.

```
#include <hal_can_lld.h>
```

Collaboration diagram for CANRxFrame:



Data Fields

- uint8_t **FMI**
Filter id.
- uint16_t **TIME**
Time stamp.
- uint8_t **DLC**:4
Data length.
- uint8_t **RTR**:1
Frame type.
- uint8_t **IDE**:1
Identifier type.
- union {
 - uint32_t **SID**:11
Standard identifier.
 - uint32_t **EID**:29
Extended identifier.
 - uint32_t **_align1**
- };
- union {
 - uint8_t **data8** [8]
Frame data.
 - uint16_t **data16** [4]
Frame data.
 - uint32_t **data32** [2]
Frame data.
- };

8.22.1 Detailed Description

CAN received frame.

Note

Accessing the frame data as word16 or word32 is not portable because machine data endianness, it can be still useful for a quick filling.

8.22.2 Field Documentation**8.22.2.1 FMI**

```
uint8_t CANRxFrame::FMI
```

Filter id.

8.22.2.2 TIME

```
uint16_t CANRxFrame::TIME
```

Time stamp.

8.22.2.3 DLC

```
uint8_t CANRxFrame::DLC
```

Data length.

8.22.2.4 RTR

```
uint8_t CANRxFrame::RTR
```

Frame type.

8.22.2.5 IDE

```
uint8_t CANRxFrame::IDE
```

Identifier type.

8.22.2.6 SID

```
uint32_t CANRxFrame::SID
```

Standard identifier.

8.22.2.7 EID

```
uint32_t CANRxFrame::EID
```

Extended identifier.

8.22.2.8 _align1

```
uint32_t CANRxFrame::_align1
```

8.22.2.9 [union]

```
union { ... } CANRxFrame
```

8.22.2.10 data8

```
uint8_t CANRxFrame::data8[8]
```

Frame data.

8.22.2.11 data16

```
uint16_t CANRxFrame::data16[4]
```

Frame data.

8.22.2.12 data32

```
uint32_t CANRxFrame::data32[2]
```

Frame data.

8.22.2.13 [union]

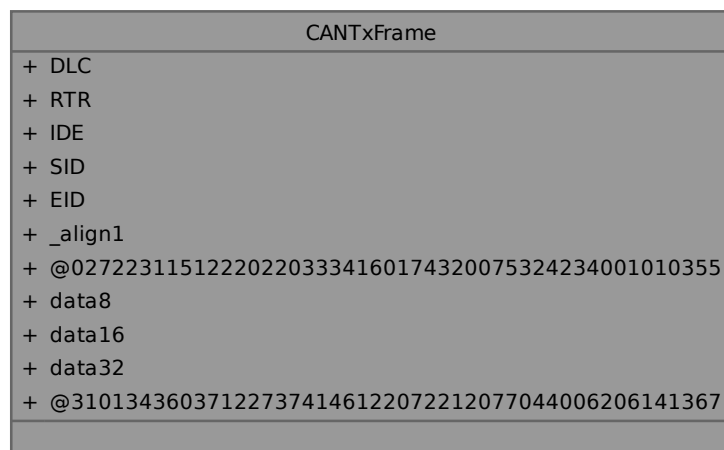
```
union { ... } CANRxFrame
```

8.23 CANTxFrame Struct Reference

CAN transmission frame.

```
#include <hal_can_ll.h>
```

Collaboration diagram for CANTxFrame:



Data Fields

- uint8_t **DLC**:4
Data length.
- uint8_t **RTR**:1
Frame type.
- uint8_t **IDE**:1
Identifier type.
- union {
 - uint32_t **SID**:11
Standard identifier.
 - uint32_t **EID**:29
Extended identifier.
 - uint32_t **_align1**
- };
- union {
 - uint8_t **data8** [8]
Frame data.
 - uint16_t **data16** [4]
Frame data.
 - uint32_t **data32** [2]
Frame data.
- };

8.23.1 Detailed Description

CAN transmission frame.

Note

Accessing the frame data as word16 or word32 is not portable because machine data endianness, it can be still useful for a quick filling.

8.23.2 Field Documentation

8.23.2.1 DLC

```
uint8_t CANTxFrame::DLC
```

Data length.

8.23.2.2 RTR

```
uint8_t CANTxFrame::RTR
```

Frame type.

8.23.2.3 IDE

```
uint8_t CANTxFrame::IDE
```

Identifier type.

8.23.2.4 SID

```
uint32_t CANTxFrame::SID
```

Standard identifier.

8.23.2.5 EID

```
uint32_t CANTxFrame::EID
```

Extended identifier.

8.23.2.6 _align1

```
uint32_t CANTxFrame::_align1
```

8.23.2.7 [union]

```
union { ... } CANTxFrame
```

8.23.2.8 data8

```
uint8_t CANTxFrame::data8[8]
```

Frame data.

8.23.2.9 data16

```
uint16_t CANTxFrame::data16[4]
```

Frame data.

8.23.2.10 data32

```
uint32_t CANTxFrame::data32[2]
```

Frame data.

8.23.2.11 [union]

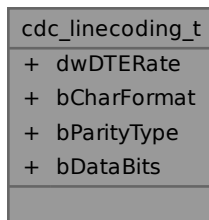
```
union { ... } CANTxFrame
```

8.24 cdc_linecoding_t Struct Reference

Type of Line Coding structure.

```
#include <hal_usb_cdc.h>
```

Collaboration diagram for cdc_linecoding_t:



Data Fields

- `uint8_t dwDTERate` [4]
- `uint8_t bCharFormat`
- `uint8_t bParityType`
- `uint8_t bDataBits`

8.24.1 Detailed Description

Type of Line Coding structure.

8.24.2 Field Documentation

8.24.2.1 dwDTERate

```
uint8_t cdc_linecoding_t::dwDTERate[4]
```

8.24.2.2 bCharFormat

```
uint8_t cdc_linecoding_t::bCharFormat
```

8.24.2.3 bParityType

```
uint8_t cdc_linecoding_t::bParityType
```

8.24.2.4 bDataBits

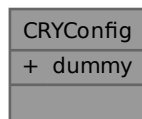
```
uint8_t cdc_linecoding_t::bDataBits
```

8.25 CRYConfig Struct Reference

Driver configuration structure.

```
#include <hal_crypto_lld.h>
```

Collaboration diagram for CRYConfig:



Data Fields

- `uint32_t dummy`

8.25.1 Detailed Description

Driver configuration structure.

Note

It could be empty on some architectures.

8.25.2 Field Documentation

8.25.2.1 dummy

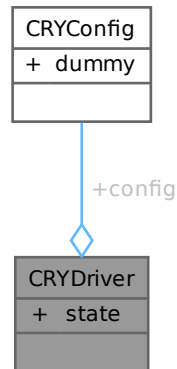
```
uint32_t CRYConfig::dummy
```

8.26 CRYDriver Struct Reference

Structure representing an CRY driver.

```
#include <hal_crypto_llid.h>
```

Collaboration diagram for CRYDriver:



Data Fields

- [crystate_t](#) `state`
Driver state.
- `const` [CRYConfig](#) * `config`
Current configuration data.

8.26.1 Detailed Description

Structure representing an CRY driver.

8.26.2 Field Documentation

8.26.2.1 state

```
crystate\_t CRYDriver::state
```

Driver state.

8.26.2.2 config

```
const CRYConfig* CRYDriver::config
```

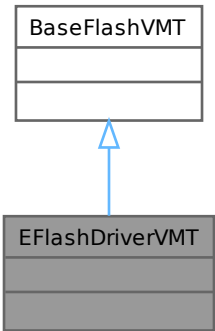
Current configuration data.

8.27 EFlashDriverVMT Struct Reference

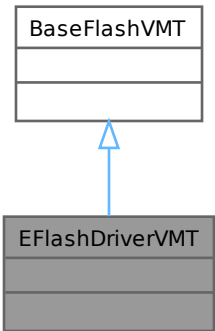
EFlash virtual methods table.

```
#include <hal_efl.h>
```

Inheritance diagram for EFlashDriverVMT:



Collaboration diagram for EFlashDriverVMT:



8.27.1 Detailed Description

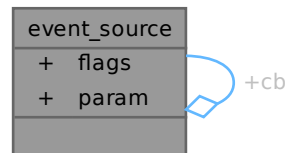
EFlash virtual methods table.

8.28 event_source Struct Reference

Events source object.

```
#include <osal.h>
```

Collaboration diagram for event_source:



Data Fields

- volatile [eventflags_t flags](#)
Stored event flags.
- [eventcallback_t cb](#)
Event source callback.
- void * [param](#)
User defined field.

8.28.1 Detailed Description

Events source object.

Note

The content of this structure is not part of the API and should not be relied upon. Implementers may define this structure in an entirely different way.

Retrieval and clearing of the flags are not defined in this API and are implementation-dependent.

8.28.2 Field Documentation

8.28.2.1 flags

```
volatile eventflags\_t event_source::flags
```

Stored event flags.

8.28.2.2 cb

```
eventcallback_t event_source::cb
```

Event source callback.

8.28.2.3 param

```
void* event_source::param
```

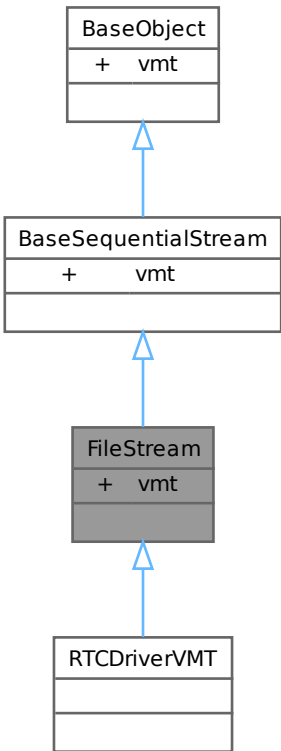
User defined field.

8.29 FileStream Struct Reference

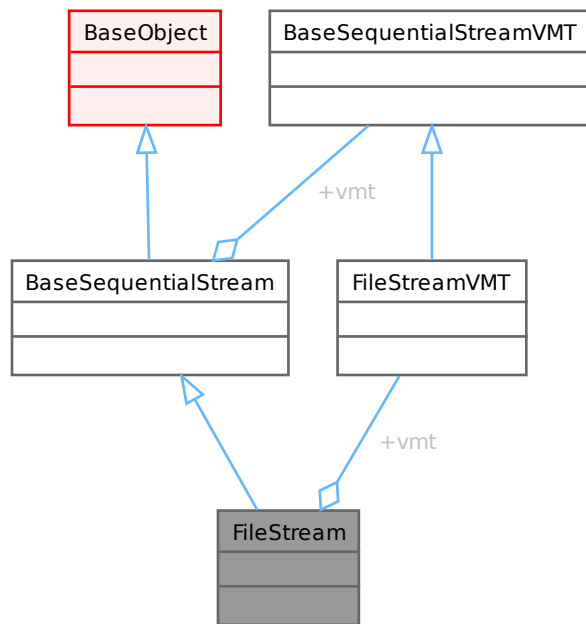
Base file stream class.

```
#include <hal_files.h>
```

Inheritance diagram for FileStream:



Collaboration diagram for FileStream:



Data Fields

- const struct [FileStreamVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseSequentialStream](#)

- const struct [BaseSequentialStreamVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.29.1 Detailed Description

Base file stream class.

This class represents a generic file data stream.

8.29.2 Field Documentation

8.29.2.1 vmt

```
const struct FileStreamVMT* FileStream::vmt
```

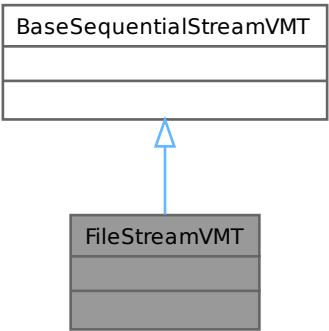
Virtual Methods Table.

8.30 FileStreamVMT Struct Reference

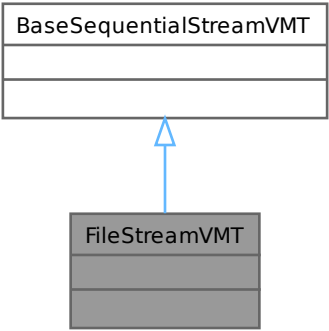
FileStream virtual methods table.

```
#include <hal_files.h>
```

Inheritance diagram for FileStreamVMT:



Collaboration diagram for FileStreamVMT:



8.30.1 Detailed Description

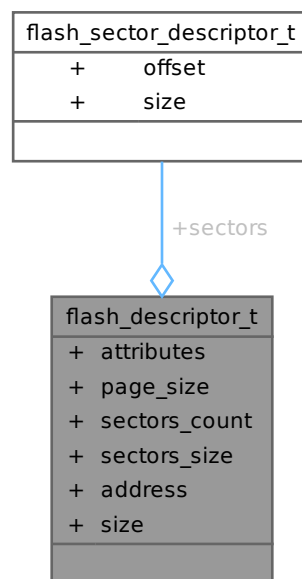
[FileStream](#) virtual methods table.

8.31 flash_descriptor_t Struct Reference

Type of a flash device descriptor.

```
#include <hal_flash.h>
```

Collaboration diagram for flash_descriptor_t:



Data Fields

- uint32_t [attributes](#)
Device_attributes.
- uint32_t [page_size](#)
Size of write page.
- [flash_sector_t sectors_count](#)
Number of sectors in the device.
- const [flash_sector_descriptor_t](#) * [sectors](#)
List of sectors for devices with non-uniform sector sizes.
- uint32_t [sectors_size](#)
Size of sectors for devices with uniform sector size.
- uint8_t * [address](#)
Flash address if memory mapped or zero.
- uint32_t [size](#)
Flash size.

8.31.1 Detailed Description

Type of a flash device descriptor.

8.31.2 Field Documentation

8.31.2.1 attributes

```
uint32_t flash_descriptor_t::attributes
```

Device_attributes.

8.31.2.2 page_size

```
uint32_t flash_descriptor_t::page_size
```

Size of write page.

8.31.2.3 sectors_count

```
flash_sector_t flash_descriptor_t::sectors_count
```

Number of sectors in the device.

8.31.2.4 sectors

```
const flash_sector_descriptor_t* flash_descriptor_t::sectors
```

List of sectors for devices with non-uniform sector sizes.

Note

If `NULL` then the device has uniform sectors size equal to `sector_size`.

8.31.2.5 sectors_size

```
uint32_t flash_descriptor_t::sectors_size
```

Size of sectors for devices with uniform sector size.

Note

If zero then the device has non uniform sectors described by the `sectors` array.

8.31.2.6 address

```
uint8_t* flash_descriptor_t::address
```

Flash address if memory mapped or zero.

Note

Conventionally, non memory mapped devices have address `NULL`.

8.31.2.7 size

```
uint32_t flash_descriptor_t::size
```

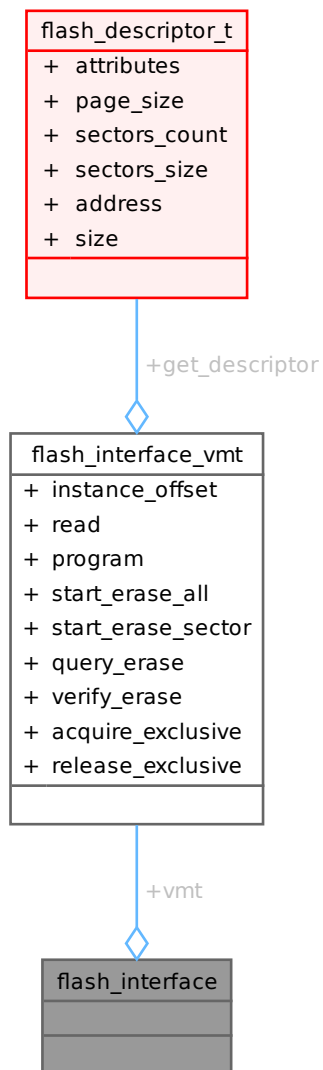
Flash size.

8.32 flash_interface Struct Reference

Structure representing a flash interface interface.

```
#include <hal_flash_interface.h>
```

Collaboration diagram for flash_interface:



Data Fields

- const struct [flash_interface_vmt](#) * `vmt`
Virtual Methods Table.

8.32.1 Detailed Description

Structure representing a flash interface interface.

8.32.2 Field Documentation

8.32.2.1 vmt

```
const struct flash_interface_vmt* flash_interface::vmt
```

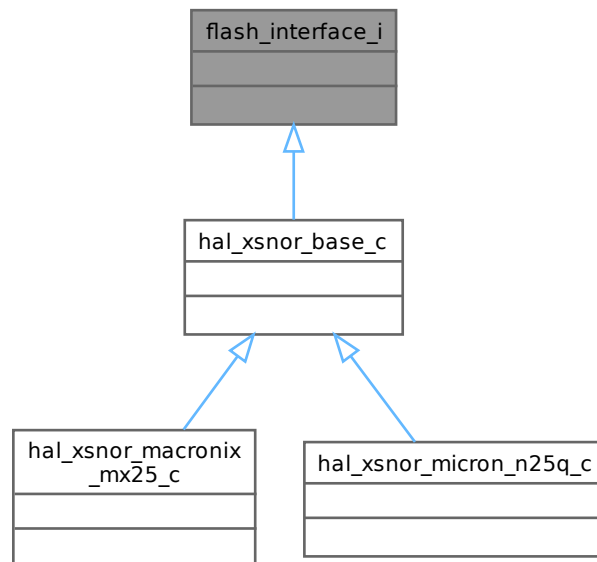
Virtual Methods Table.

8.33 flash_interface_i Interface Reference

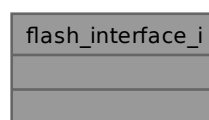
Generic flash interface.

```
#include <hal_flash_interface.h>
```

Inheritance diagram for flash_interface_i:



Collaboration diagram for flash_interface_i:



8.33.1 Detailed Description

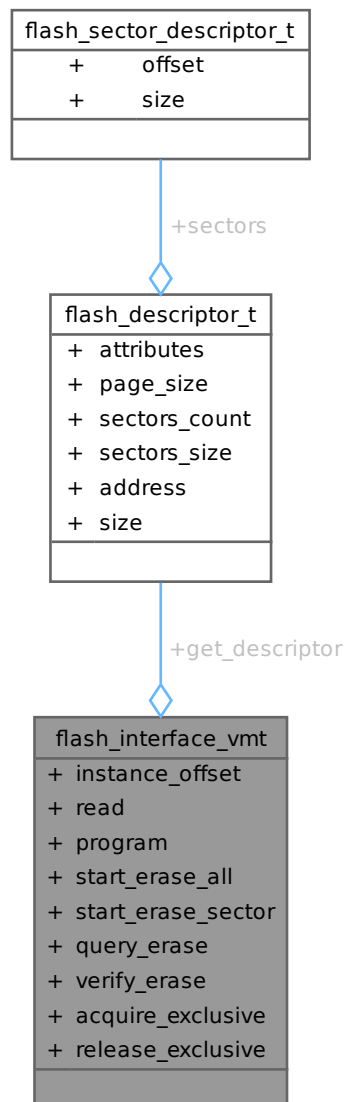
Generic flash interface.

8.34 flash_interface_vmt Struct Reference

Interface `flash_interface_i` virtual methods table.

```
#include <hal_flash_interface.h>
```

Collaboration diagram for `flash_interface_vmt`:



Data Fields

- `size_t instance_offset`
- `const flash_descriptor_t *(* get_descriptor )(void *ip)`
- `flash_error_t(* read )(void *ip, flash_offset_t offset, size_t n, uint8_t *rp)`
- `flash_error_t(* program )(void *ip, flash_offset_t offset, size_t n, const uint8_t *pp)`
- `flash_error_t(* start_erase_all )(void *ip)`
- `flash_error_t(* start_erase_sector )(void *ip, flash_sector_t sector)`
- `flash_error_t(* query_erase )(void *ip, unsigned *msec)`
- `flash_error_t(* verify_erase )(void *ip, flash_sector_t sector)`
- `flash_error_t(* acquire_exclusive )(void *ip)`
- `flash_error_t(* release_exclusive )(void *ip)`

8.34.1 Detailed Description

Interface `flash_interface_i` virtual methods table.

8.34.2 Field Documentation

8.34.2.1 instance_offset

```
size_t flash_interface_vmt::instance_offset
```

8.34.2.2 get_descriptor

```
const flash_descriptor_t *(* flash_interface_vmt::get_descriptor) (void *ip)
```

8.34.2.3 read

```
flash_error_t(* flash_interface_vmt::read) (void *ip, flash_offset_t offset, size_t n, uint8_t *rp)
```

8.34.2.4 program

```
flash_error_t(* flash_interface_vmt::program) (void *ip, flash_offset_t offset, size_t n, const uint8_t *pp)
```

8.34.2.5 start_erase_all

```
flash_error_t(* flash_interface_vmt::start_erase_all) (void *ip)
```

8.34.2.6 start_erase_sector

```
flash_error_t(* flash_interface_vmt::start_erase_sector) (void *ip, flash_sector_t sector)
```

8.34.2.7 query_erase

```
flash_error_t(* flash_interface_vmt::query_erase) (void *ip, unsigned *msec)
```

8.34.2.8 verify_erase

```
flash_error_t(* flash_interface_vmt::verify_erase) (void *ip, flash_sector_t sector)
```

8.34.2.9 acquire_exclusive

```
flash_error_t(* flash_interface_vmt::acquire_exclusive) (void *ip)
```

8.34.2.10 release_exclusive

```
flash_error_t(* flash_interface_vmt::release_exclusive) (void *ip)
```

8.35 flash_sector_descriptor_t Struct Reference

Flash sector descriptor.

```
#include <hal_flash.h>
```

Collaboration diagram for flash_sector_descriptor_t:

flash_sector_descriptor_t	
+	offset
+	size

Data Fields

- [flash_offset_t offset](#)
Sector offset.
- [uint32_t size](#)
Sector size.

8.35.1 Detailed Description

Flash sector descriptor.

8.35.2 Field Documentation

8.35.2.1 offset

`flash_offset_t` `flash_sector_descriptor_t::offset`

Sector offset.

8.35.2.2 size

`uint32_t` `flash_sector_descriptor_t::size`

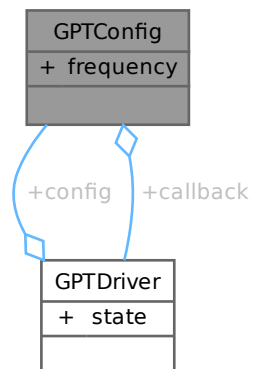
Sector size.

8.36 GPTConfig Struct Reference

Driver configuration structure.

```
#include <hal_gpt_lld.h>
```

Collaboration diagram for GPTConfig:



Data Fields

- `gptfreq_t` `frequency`
Timer clock in Hz.
- `gptcallback_t` `callback`
Timer callback pointer.

8.36.1 Detailed Description

Driver configuration structure.

Note

It could be empty on some architectures.

8.36.2 Field Documentation

8.36.2.1 frequency

`gptfreq_t` `GPTConfig::frequency`

Timer clock in Hz.

Note

The low level can use assertions in order to catch invalid frequency specifications.

8.36.2.2 callback

`gptcallback_t` `GPTConfig::callback`

Timer callback pointer.

Note

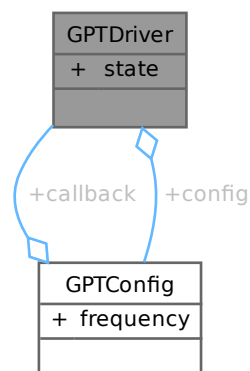
This callback is invoked on GPT counter events.

8.37 GPTDriver Struct Reference

Structure representing a GPT driver.

```
#include <hal_gpt_lld.h>
```

Collaboration diagram for GPTDriver:



Data Fields

- `gptstate_t state`
Driver state.
- `const GPTConfig * config`
Current configuration data.

8.37.1 Detailed Description

Structure representing a GPT driver.

8.37.2 Field Documentation

8.37.2.1 state

`gptstate_t` GPTDriver::state

Driver state.

8.37.2.2 config

`const GPTConfig*` GPTDriver::config

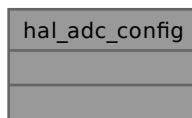
Current configuration data.

8.38 hal_adc_config Struct Reference

Driver configuration structure.

```
#include <hal_adc.h>
```

Collaboration diagram for hal_adc_config:



8.38.1 Detailed Description

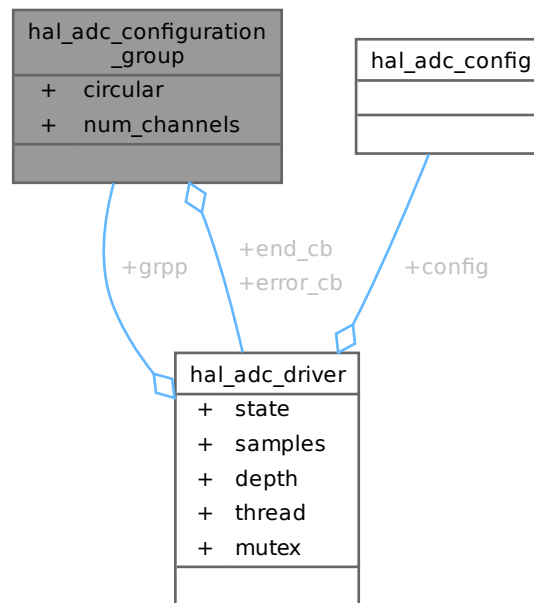
Driver configuration structure.

8.39 hal_adc_configuration_group Struct Reference

Conversion group configuration structure.

```
#include <hal_adc.h>
```

Collaboration diagram for hal_adc_configuration_group:



Data Fields

- bool [circular](#)
Enables the circular buffer mode for the group.
- [adc_channels_num_t](#) [num_channels](#)
Number of the analog channels belonging to the conversion group.
- [adccallback_t](#) [end_cb](#)
Callback function associated to the group or `NULL`.
- [adcerrorcallback_t](#) [error_cb](#)
Error callback or `NULL`.

8.39.1 Detailed Description

Conversion group configuration structure.

This implementation-dependent structure describes a conversion operation.

Note

The use of this configuration structure requires knowledge of STM32 ADC cell registers interface, please refer to the STM32 reference manual for details.

8.39.2 Field Documentation

8.39.2.1 circular

```
bool hal_adc_configuration_group::circular
```

Enables the circular buffer mode for the group.

8.39.2.2 num_channels

```
adc_channels_num_t hal_adc_configuration_group::num_channels
```

Number of the analog channels belonging to the conversion group.

8.39.2.3 end_cb

```
adccallback_t hal_adc_configuration_group::end_cb
```

Callback function associated to the group or NULL.

8.39.2.4 error_cb

```
adcerrorcallback_t hal_adc_configuration_group::error_cb
```

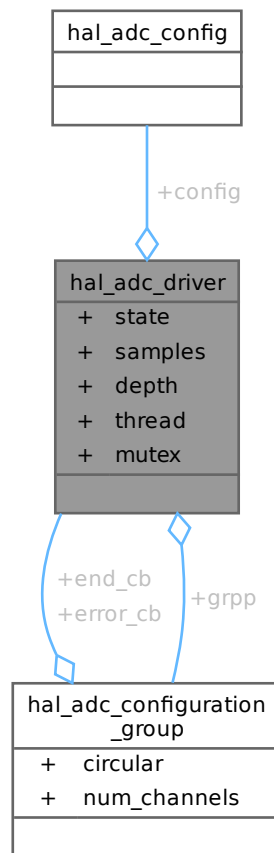
Error callback or NULL.

8.40 hal_adc_driver Struct Reference

Structure representing an ADC driver.

```
#include <hal_adc.h>
```


Collaboration diagram for hal_adc_driver:



Data Fields

- [adcstate_t state](#)
Driver state.
- `const ADCConfig * config`
Current configuration data.
- `adcsample_t * samples`
Current samples buffer pointer or `NULL`.
- `size_t depth`
Current samples buffer depth or 0.
- `const ADCConversionGroup * grpp`
Current conversion group pointer or `NULL`.
- `thread_reference_t thread`
Waiting thread.
- `mutex_t mutex`
Mutex protecting the peripheral.

8.40.1 Detailed Description

Structure representing an ADC driver.

8.40.2 Field Documentation

8.40.2.1 state

```
adcstate_t hal_adc_driver::state
```

Driver state.

8.40.2.2 config

```
const ADCConfig* hal_adc_driver::config
```

Current configuration data.

8.40.2.3 samples

```
adcsample_t* hal_adc_driver::samples
```

Current samples buffer pointer or NULL.

8.40.2.4 depth

```
size_t hal_adc_driver::depth
```

Current samples buffer depth or 0.

8.40.2.5 grpp

```
const ADCConversionGroup* hal_adc_driver::grpp
```

Current conversion group pointer or NULL.

8.40.2.6 thread

```
thread_reference_t hal_adc_driver::thread
```

Waiting thread.

8.40.2.7 mutex

```
mutex_t hal_adc_driver::mutex
```

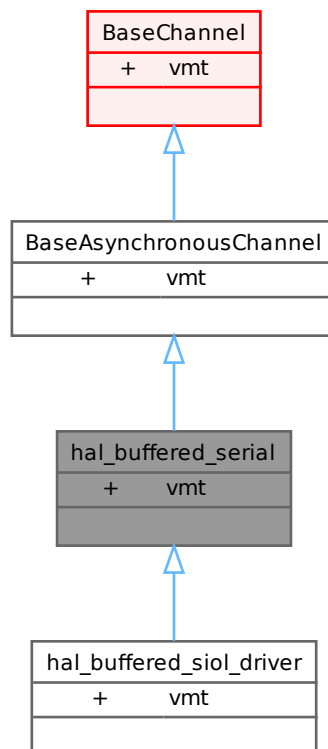
Mutex protecting the peripheral.

8.41 hal_buffered_serial Struct Reference

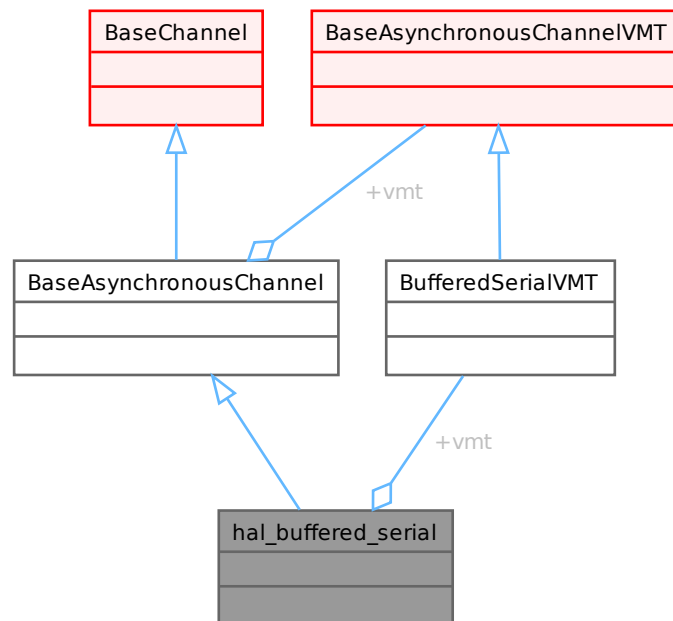
Buffered serial channel class.

```
#include <hal_buffered_serial.h>
```

Inheritance diagram for hal_buffered_serial:



Collaboration diagram for `hal_buffered_serial`:



Data Fields

- const struct [BufferedSerialVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from [BaseAsynchronousChannel](#)

- const struct [BaseAsynchronousChannelVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from [BaseChannel](#)

- const struct [BaseChannelVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from [BaseSequentialStream](#)

- const struct [BaseSequentialStreamVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.41.1 Detailed Description

Buffered serial channel class.

This class extends [BaseAsynchronousChannel](#) by adding physical I/O queues.

8.41.2 Field Documentation

8.41.2.1 vmt

```
const struct BufferedSerialVMT* hal_buffered_serial::vmt
```

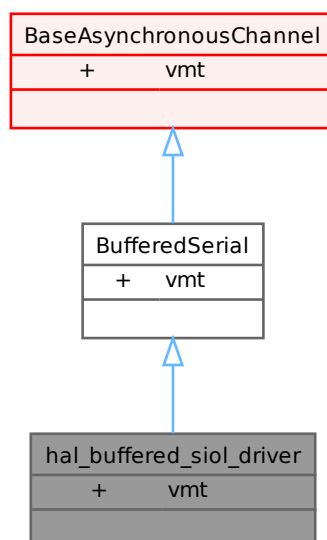
Virtual Methods Table.

8.42 hal_buffered_siol_driver Struct Reference

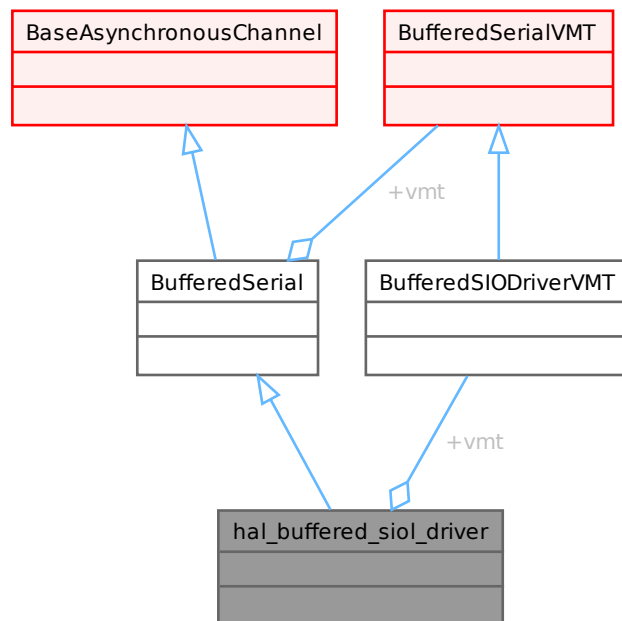
Buffered SIO driver class.

```
#include <hal_buffered_sio.h>
```

Inheritance diagram for `hal_buffered_siol_driver`:



Collaboration diagram for `hal_buffered_siol_driver`:



Data Fields

- const struct [BufferedSIODriverVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from [hal_buffered_serial](#)

- const struct [BufferedSerialVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from [BaseAsynchronousChannel](#)

- const struct [BaseAsynchronousChannelVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from [BaseChannel](#)

- const struct [BaseChannelVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from BaseSequentialStream

- const struct BaseSequentialStreamVMT * vmt
Virtual Methods Table.

Data Fields inherited from BaseObject

- const struct BaseObjectVMT * vmt
Virtual Methods Table.

8.42.1 Detailed Description

Buffered SIO driver class.

8.42.2 Field Documentation

8.42.2.1 vmt

```
const struct BufferedSIODriverVMT* hal_buffered_siol_driver::vmt
```

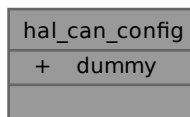
Virtual Methods Table.

8.43 hal_can_config Struct Reference

Type of a CAN configuration structure.

```
#include <hal_can_llid.h>
```

Collaboration diagram for hal_can_config:



Data Fields

- uint32_t dummy

8.43.1 Detailed Description

Type of a CAN configuration structure.

8.43.2 Field Documentation

8.43.2.1 dummy

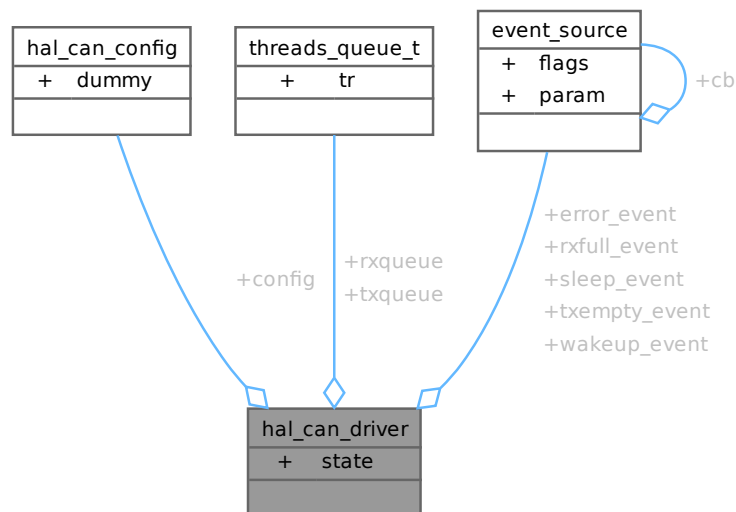
```
uint32_t hal_can_config::dummy
```

8.44 hal_can_driver Struct Reference

Structure representing an CAN driver.

```
#include <hal_can_lld.h>
```

Collaboration diagram for hal_can_driver:



Data Fields

- [canstate_t state](#)
Driver state.
- `const CANConfig * config`
Current configuration data.
- [threads_queue_t txqueue](#)
Transmission threads queue.

- [threads_queue_t rxqueue](#)
Receive threads queue.
- [event_source_t rxfull_event](#)
One or more frames become available.
- [event_source_t txempty_event](#)
One or more transmission mailbox become available.
- [event_source_t error_event](#)
A CAN bus error happened.
- [event_source_t sleep_event](#)
Entering sleep state event.
- [event_source_t wakeup_event](#)
Exiting sleep state event.

8.44.1 Detailed Description

Structure representing an CAN driver.

8.44.2 Field Documentation

8.44.2.1 state

```
canstate_t hal_can_driver::state
```

Driver state.

8.44.2.2 config

```
const CANConfig* hal_can_driver::config
```

Current configuration data.

8.44.2.3 txqueue

```
threads_queue_t hal_can_driver::txqueue
```

Transmission threads queue.

8.44.2.4 rxqueue

```
threads_queue_t hal_can_driver::rxqueue
```

Receive threads queue.

8.44.2.5 rxfull_event

```
event_source_t hal_can_driver::rxfull_event
```

One or more frames become available.

Note

After broadcasting this event it will not be broadcasted again until the received frames queue has been completely emptied. It is **not** broadcasted for each received frame. It is responsibility of the application to empty the queue by repeatedly invoking `chReceive()` when listening to this event. This behavior minimizes the interrupt served by the system because CAN traffic.

The flags associated to the listeners will indicate which receive mailboxes become non-empty.

8.44.2.6 txempty_event

```
event_source_t hal_can_driver::txempty_event
```

One or more transmission mailbox become available.

Note

The flags associated to the listeners will indicate which transmit mailboxes become empty.

8.44.2.7 error_event

```
event_source_t hal_can_driver::error_event
```

A CAN bus error happened.

Note

The flags associated to the listeners will indicate the error(s) that have occurred.

8.44.2.8 sleep_event

```
event_source_t hal_can_driver::sleep_event
```

Entering sleep state event.

8.44.2.9 wakeup_event

```
event_source_t hal_can_driver::wakeup_event
```

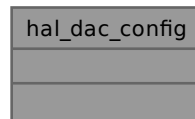
Exiting sleep state event.

8.45 hal_dac_config Struct Reference

Driver configuration structure.

```
#include <hal_dac.h>
```

Collaboration diagram for hal_dac_config:



8.45.1 Detailed Description

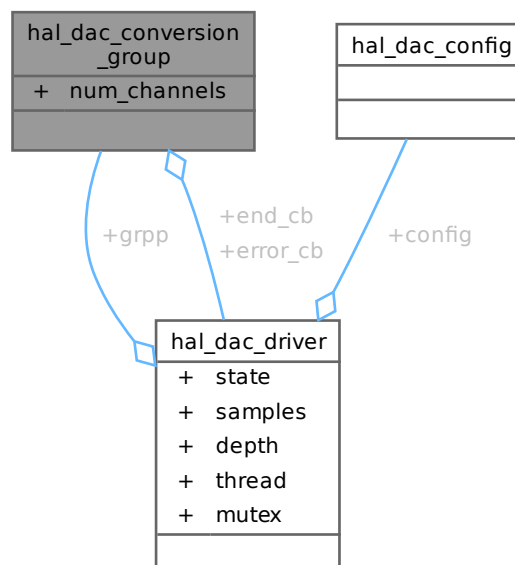
Driver configuration structure.

8.46 hal_dac_conversion_group Struct Reference

DAC Conversion group structure.

```
#include <hal_dac.h>
```

Collaboration diagram for hal_dac_conversion_group:



Data Fields

- `uint32_t num_channels`
Number of DAC channels.
- `daccallback_t end_cb`
Operation complete callback or NULL.
- `dacerrorcallback_t error_cb`
Error handling callback or NULL.

8.46.1 Detailed Description

DAC Conversion group structure.

8.46.2 Field Documentation

8.46.2.1 num_channels

```
uint32_t hal_dac_conversion_group::num_channels
```

Number of DAC channels.

8.46.2.2 end_cb

```
daccallback_t hal_dac_conversion_group::end_cb
```

Operation complete callback or NULL.

8.46.2.3 error_cb

```
dacerrorcallback_t hal_dac_conversion_group::error_cb
```

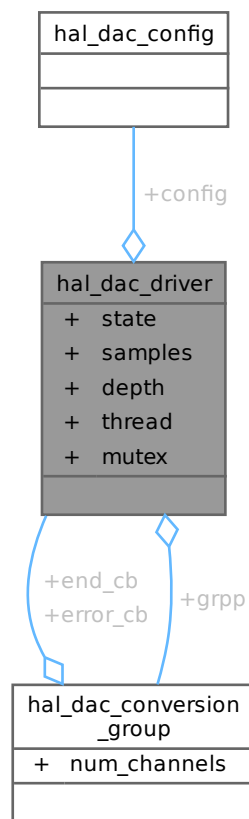
Error handling callback or NULL.

8.47 hal_dac_driver Struct Reference

Structure representing a DAC driver.

```
#include <hal_dac.h>
```

Collaboration diagram for hal_dac_driver:



Data Fields

- `dacstate_t state`
Driver state.
- `const DACConversionGroup * grpp`
Conversion group.
- `dacsample_t * samples`
Samples buffer pointer.
- `size_t depth`
Samples buffer size.
- `const DACConfig * config`
Current configuration data.
- `thread_reference_t thread`
Waiting thread.
- `mutex_t mutex`
Mutex protecting the bus.

8.47.1 Detailed Description

Structure representing a DAC driver.

8.47.2 Field Documentation

8.47.2.1 state

```
dacstate_t hal_dac_driver::state
```

Driver state.

8.47.2.2 grpp

```
const DACConversionGroup* hal_dac_driver::grpp
```

Conversion group.

8.47.2.3 samples

```
dacsample_t* hal_dac_driver::samples
```

Samples buffer pointer.

8.47.2.4 depth

```
size_t hal_dac_driver::depth
```

Samples buffer size.

8.47.2.5 config

```
const DACConfig* hal_dac_driver::config
```

Current configuration data.

8.47.2.6 thread

```
thread_reference_t hal_dac_driver::thread
```

Waiting thread.

8.47.2.7 mutex

```
mutex_t hal_dac_driver::mutex
```

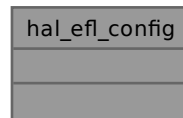
Mutex protecting the bus.

8.48 hal_efl_config Struct Reference

Type of a structure representing a flash driver configuration.

```
#include <hal_efl.h>
```

Collaboration diagram for hal_efl_config:



8.48.1 Detailed Description

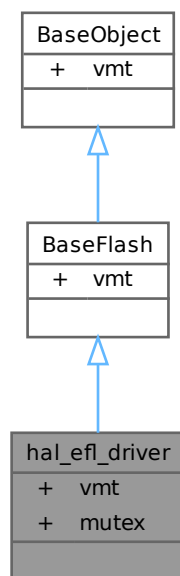
Type of a structure representing a flash driver configuration.

8.49 hal_efl_driver Struct Reference

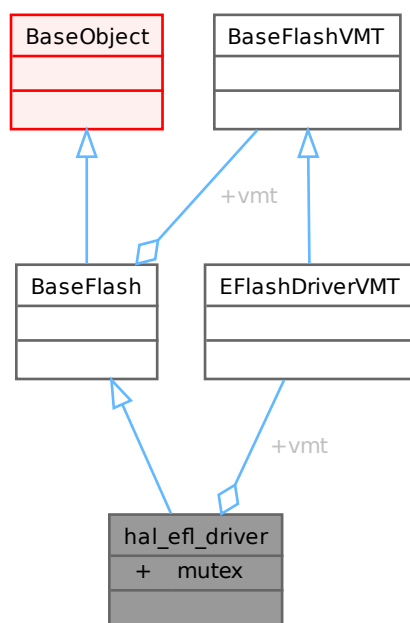
Structure representing an embedded flash driver.

```
#include <hal_efl.h>
```

Inheritance diagram for hal_efl_driver:



Collaboration diagram for hal_efl_driver:



Data Fields

- const struct [EFlashDriverVMT](#) * vmt
EFlashDriver Virtual Methods Table.
- [_efl_driver_data mutex_t](#) mutex
Mutex protecting EFL.

Data Fields inherited from [BaseFlash](#)

- const struct [BaseFlashVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.49.1 Detailed Description

Structure representing an embedded flash driver.

8.49.2 Field Documentation**8.49.2.1 vmt**

```
const struct EFlashDriverVMT* hal_efl_driver::vmt
```

[EFlashDriver](#) Virtual Methods Table.

8.49.2.2 mutex

```
\_efl\_driver\_data mutex\_t hal_efl_driver::mutex
```

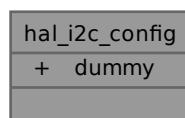
Mutex protecting EFL.

8.50 hal_i2c_config Struct Reference

I2C driver configuration structure.

```
#include <hal_i2c_ll.d.h>
```

Collaboration diagram for hal_i2c_config:



Data Fields

- uint32_t [dummy](#)

8.50.1 Detailed Description

I2C driver configuration structure.

Note

Implementations may extend this structure to contain more, architecture dependent, fields.

8.50.2 Field Documentation

8.50.2.1 dummy

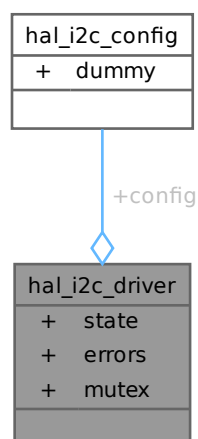
```
uint32_t hal_i2c_config::dummy
```

8.51 hal_i2c_driver Struct Reference

Structure representing an I2C driver.

```
#include <hal_i2c_lld.h>
```

Collaboration diagram for hal_i2c_driver:



Data Fields

- [i2cstate_t state](#)
Driver state.
- `const I2CConfig * config`
Current configuration data.
- [i2cflags_t errors](#)
Error flags.
- [mutex_t mutex](#)

8.51.1 Detailed Description

Structure representing an I2C driver.

8.51.2 Field Documentation

8.51.2.1 state

```
i2cstate\_t hal_i2c_driver::state
```

Driver state.

8.51.2.2 config

```
const I2CConfig* hal_i2c_driver::config
```

Current configuration data.

8.51.2.3 errors

```
i2cflags\_t hal_i2c_driver::errors
```

Error flags.

8.51.2.4 mutex

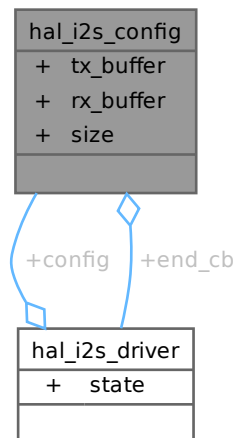
```
mutex\_t hal_i2c_driver::mutex
```

8.52 hal_i2s_config Struct Reference

Driver configuration structure.

```
#include <hal_i2s.h>
```

Collaboration diagram for hal_i2s_config:



Data Fields

- const void * [tx_buffer](#)
Transmission buffer pointer.
- void * [rx_buffer](#)
Receive buffer pointer.
- size_t [size](#)
TX and RX buffers size as number of samples.
- [i2scallback_t end_cb](#)
Callback function called during streaming.

8.52.1 Detailed Description

Driver configuration structure.

8.52.2 Field Documentation

8.52.2.1 tx_buffer

```
const void* hal_i2s_config::tx_buffer
```

Transmission buffer pointer.

Note

Can be `NULL` if TX is not required.

8.52.2.2 rx_buffer

```
void* hal_i2s_config::rx_buffer
```

Receive buffer pointer.

Note

Can be NULL if RX is not required.

8.52.2.3 size

```
size_t hal_i2s_config::size
```

TX and RX buffers size as number of samples.

8.52.2.4 end_cb

```
i2scallback_t hal_i2s_config::end_cb
```

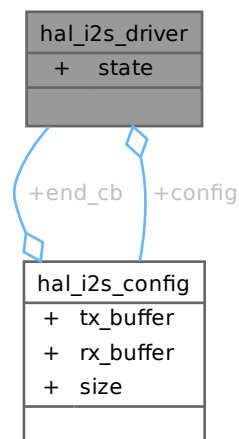
Callback function called during streaming.

8.53 hal_i2s_driver Struct Reference

Structure representing an I2S driver.

```
#include <hal_i2s.h>
```

Collaboration diagram for hal_i2s_driver:



Data Fields

- [i2sstate_t state](#)
Driver state.
- `const I2SConfig * config`
Current configuration data.

8.53.1 Detailed Description

Structure representing an I2S driver.

8.53.2 Field Documentation

8.53.2.1 state

```
i2sstate_t hal_i2s_driver::state
```

Driver state.

8.53.2.2 config

```
const I2SConfig* hal_i2s_driver::config
```

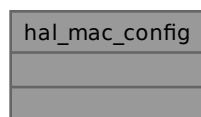
Current configuration data.

8.54 hal_mac_config Struct Reference

Driver configuration structure.

```
#include <hal_mac.h>
```

Collaboration diagram for hal_mac_config:



8.54.1 Detailed Description

Driver configuration structure.

Note

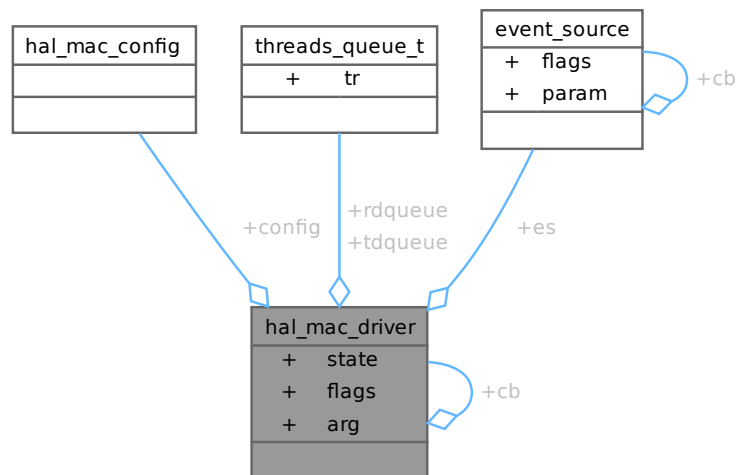
Implementations may extend this structure to contain more, architecture dependent, fields.

8.55 hal_mac_driver Struct Reference

Structure representing a MAC driver.

```
#include <hal_mac.h>
```

Collaboration diagram for hal_mac_driver:



Data Fields

- [macstate_t state](#)
Driver state.
- `const MACConfig * config`
Current configuration data.
- [threads_queue_t tdqueue](#)
Transmit threads queue.
- [threads_queue_t rdqueue](#)
Receive threads queue.
- [event_source_t es](#)
MAC events source.
- [eventflags_t flags](#)
Locally held event flags for callback use.
- [maccb_t cb](#)
Events callback.
- `void * arg`
User argument.

8.55.1 Detailed Description

Structure representing a MAC driver.

Note

Implementations may extend this structure to contain more, architecture dependent, fields.

8.55.2 Field Documentation

8.55.2.1 state

```
macstate_t hal_mac_driver::state
```

Driver state.

8.55.2.2 config

```
const MACConfig* hal_mac_driver::config
```

Current configuration data.

8.55.2.3 tdqueue

```
threads_queue_t hal_mac_driver::tdqueue
```

Transmit threads queue.

8.55.2.4 rdqueue

```
threads_queue_t hal_mac_driver::rdqueue
```

Receive threads queue.

8.55.2.5 es

```
event_source_t hal_mac_driver::es
```

MAC events source.

8.55.2.6 flags

```
eventflags_t hal_mac_driver::flags
```

Locally held event flags for callback use.

8.55.2.7 cb

```
maccb_t hal_mac_driver::cb
```

Events callback.

Note

Can be NULL.

8.55.2.8 arg

```
void* hal_mac_driver::arg
```

User argument.

Note

Can be retrieved through the `ethp` argument of the callback.

8.56 hal_mac_receive_descriptor Struct Reference

Structure representing a MAC receive descriptor.

```
#include <hal_mac.h>
```

Collaboration diagram for `hal_mac_receive_descriptor`:

hal_mac_receive_descriptor	
+	offset
+	size

Data Fields

- `size_t` [offset](#)
Current read offset.
- `size_t` [size](#)
Available data size.

8.56.1 Detailed Description

Structure representing a MAC receive descriptor.

8.56.2 Field Documentation

8.56.2.1 offset

```
size_t hal_mac_receive_descriptor::offset
```

Current read offset.

8.56.2.2 size

```
size_t hal_mac_receive_descriptor::size
```

Available data size.

8.57 hal_mac_transmit_descriptor Struct Reference

Structure representing a MAC transmit descriptor.

```
#include <hal_mac.h>
```

Collaboration diagram for hal_mac_transmit_descriptor:

hal_mac_transmit_descriptor	
+	offset
+	size

Data Fields

- `size_t` [offset](#)
Current write offset.
- `size_t` [size](#)
Available space size.

8.57.1 Detailed Description

Structure representing a MAC transmit descriptor.

8.57.2 Field Documentation

8.57.2.1 offset

```
size_t hal_mac_transmit_descriptor::offset
```

Current write offset.

8.57.2.2 size

```
size_t hal_mac_transmit_descriptor::size
```

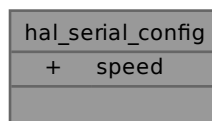
Available space size.

8.58 hal_serial_config Struct Reference

PLATFORM Serial Driver configuration structure.

```
#include <hal_serial_lld.h>
```

Collaboration diagram for hal_serial_config:



Data Fields

- uint32_t [speed](#)
Bit rate.

8.58.1 Detailed Description

PLATFORM Serial Driver configuration structure.

An instance of this structure must be passed to `sdStart()` in order to configure and start a serial driver operations.

Note

This structure content is architecture dependent, each driver implementation defines its own version and the custom static initializers.

8.58.2 Field Documentation

8.58.2.1 speed

```
uint32_t hal_serial_config::speed
```

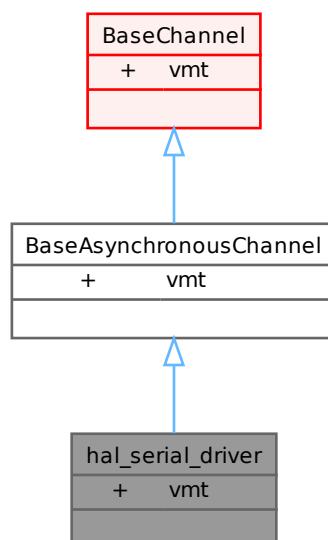
Bit rate.

8.59 hal_serial_driver Struct Reference

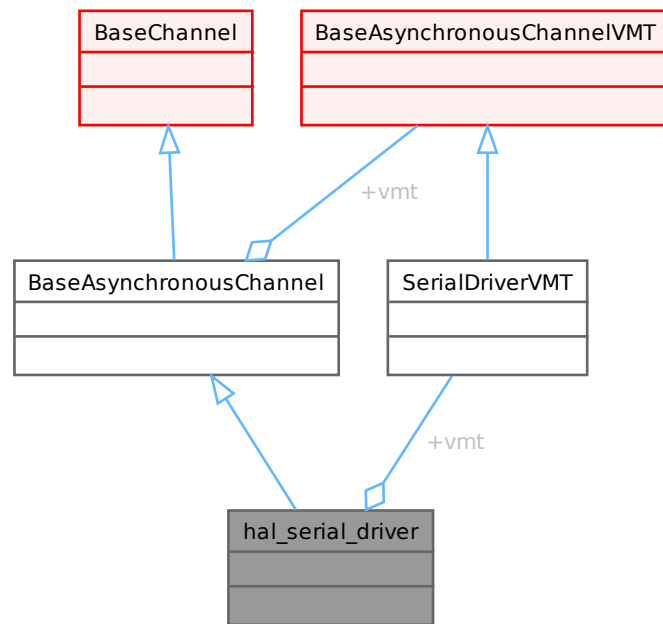
Full duplex serial driver class.

```
#include <hal_serial.h>
```

Inheritance diagram for `hal_serial_driver`:



Collaboration diagram for hal_serial_driver:



Data Fields

- const struct [SerialDriverVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseAsynchronousChannel](#)

- const struct [BaseAsynchronousChannelVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseChannel](#)

- const struct [BaseChannelVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseSequentialStream](#)

- const struct [BaseSequentialStreamVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.59.1 Detailed Description

Full duplex serial driver class.

This class extends [BaseAsynchronousChannel](#) by adding physical I/O queues.

8.59.2 Field Documentation

8.59.2.1 vmt

```
const struct SerialDriverVMT* hal_serial_driver::vmt
```

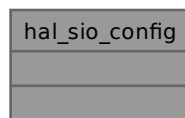
Virtual Methods Table.

8.60 hal_sio_config Struct Reference

Driver configuration structure.

```
#include <hal_sio.h>
```

Collaboration diagram for hal_sio_config:



8.60.1 Detailed Description

Driver configuration structure.

Note

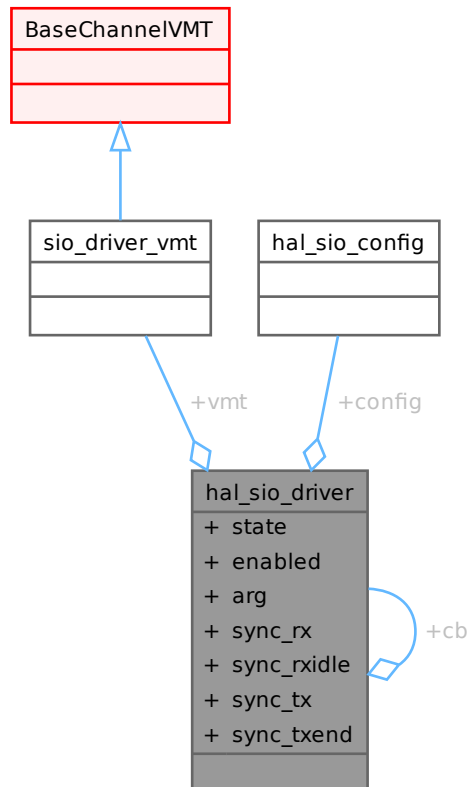
Implementations may extend this structure to contain more, architecture dependent, fields.

8.61 hal_sio_driver Struct Reference

Structure representing a SIO driver.

```
#include <hal_sio.h>
```

Collaboration diagram for hal_sio_driver:



Data Fields

- const struct [sio_driver_vmt](#) * [vmt](#)
Virtual Methods Table.
- [siostate_t](#) [state](#)
Driver state.
- const [SIOConfig](#) * [config](#)
Current configuration data.
- [sioevents_t](#) [enabled](#)
Enabled event flags.
- [siocb_t](#) [cb](#)
Events callback.
- void * [arg](#)

User argument.

- [thread_reference_t sync_rx](#)

Synchronization point for RX.

- [thread_reference_t sync_rxidle](#)

Synchronization point for RX idle.

- [thread_reference_t sync_tx](#)

Synchronization point for TX.

- [thread_reference_t sync_txend](#)

Synchronization point for TX-end.

8.61.1 Detailed Description

Structure representing a SIO driver.

Note

Implementations may extend this structure to contain more, architecture dependent, fields.

8.61.2 Field Documentation

8.61.2.1 vmt

```
const struct sio\_driver\_vmt* hal_sio_driver::vmt
```

Virtual Methods Table.

8.61.2.2 state

```
siostate\_t hal_sio_driver::state
```

Driver state.

8.61.2.3 config

```
const SIOConfig* hal_sio_driver::config
```

Current configuration data.

8.61.2.4 enabled

```
sioevents\_t hal_sio_driver::enabled
```

Enabled event flags.

8.61.2.5 cb

`siocb_t` hal_sio_driver::cb

Events callback.

Note

Can be NULL.

8.61.2.6 arg

`void*` hal_sio_driver::arg

User argument.

Note

Can be retrieved through the `sioop` argument of the callback.

8.61.2.7 sync_rx

`thread_reference_t` hal_sio_driver::sync_rx

Synchronization point for RX.

8.61.2.8 sync_rxidle

`thread_reference_t` hal_sio_driver::sync_rxidle

Synchronization point for RX idle.

8.61.2.9 sync_tx

`thread_reference_t` hal_sio_driver::sync_tx

Synchronization point for TX.

8.61.2.10 sync_txend

`thread_reference_t` hal_sio_driver::sync_txend

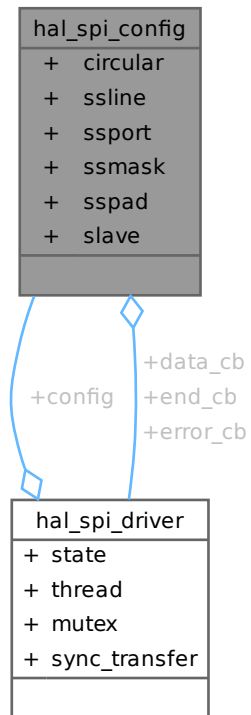
Synchronization point for TX-end.

8.62 hal_spi_config Struct Reference

Driver configuration structure.

```
#include <hal_spi_v2.h>
```

Collaboration diagram for hal_spi_config:



Data Fields

- bool [circular](#)
Enables the circular buffer mode.
- [spicallback_t](#) [end_cb](#)
Operation complete callback or `NULL`.
- [ioline_t](#) [ssline](#)
The chip select line.
- [ioportid_t](#) [ssport](#)
The chip select port.
- [ioportmask_t](#) [ssmask](#)
The chip select port mask.
- [uint_fast8_t](#) [sspad](#)
The chip select pad number.
- bool [slave](#)

Enables the slave mode.

- [spicb_t data_cb](#)

Operation data callback or NULL.

- [spicb_t error_cb](#)

Operation error callback or NULL.

8.62.1 Detailed Description

Driver configuration structure.

8.62.2 Field Documentation

8.62.2.1 circular

```
bool hal_spi_config::circular
```

Enables the circular buffer mode.

8.62.2.2 end_cb

```
spicallback_t hal_spi_config::end_cb
```

Operation complete callback or NULL.

8.62.2.3 ssline

```
ioline_t hal_spi_config::ssline
```

The chip select line.

Note

Only used in master mode.

8.62.2.4 ssport

```
ioportid_t hal_spi_config::ssport
```

The chip select port.

8.62.2.5 ssmask

```
ioportmask_t hal_spi_config::ssmask
```

The chip select port mask.

8.62.2.6 sspad

```
uint_fast8_t hal_spi_config::sspad
```

The chip select pad number.

8.62.2.7 slave

```
bool hal_spi_config::slave
```

Enables the slave mode.

8.62.2.8 data_cb

```
spicb_t hal_spi_config::data_cb
```

Operation data callback or NULL.

8.62.2.9 error_cb

```
spicb_t hal_spi_config::error_cb
```

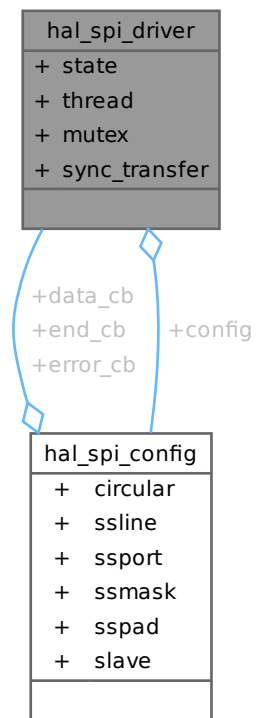
Operation error callback or NULL.

8.63 hal_spi_driver Struct Reference

Structure representing an SPI driver.

```
#include <hal_spi_v2.h>
```

Collaboration diagram for hal_spi_driver:



Data Fields

- [spistate_t state](#)
Driver state.
- `const SPIDriverConfig * config`
Current configuration data.
- [thread_reference_t thread](#)
Waiting thread.
- [mutex_t mutex](#)
Mutex protecting the peripheral.
- [thread_reference_t sync_transfer](#)
Synchronization point for transfer.

8.63.1 Detailed Description

Structure representing an SPI driver.

8.63.2 Field Documentation

8.63.2.1 state

```
spistate_t hal_spi_driver::state
```

Driver state.

8.63.2.2 config

```
const SPIConfig * hal_spi_driver::config
```

Current configuration data.

8.63.2.3 thread

```
thread_reference_t hal_spi_driver::thread
```

Waiting thread.

8.63.2.4 mutex

```
mutex_t hal_spi_driver::mutex
```

Mutex protecting the peripheral.

8.63.2.5 sync_transfer

```
thread_reference_t hal_spi_driver::sync_transfer
```

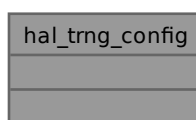
Synchronization point for transfer.

8.64 hal_trng_config Struct Reference

Driver configuration structure.

```
#include <hal_trng.h>
```

Collaboration diagram for hal_trng_config:



8.64.1 Detailed Description

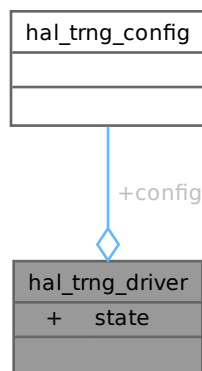
Driver configuration structure.

8.65 hal_trng_driver Struct Reference

Structure representing a TRNG driver.

```
#include <hal_trng.h>
```

Collaboration diagram for hal_trng_driver:



Data Fields

- [trngstate_t state](#)
Driver state.
- const [TRNGConfig * config](#)
Current configuration data.

8.65.1 Detailed Description

Structure representing a TRNG driver.

8.65.2 Field Documentation

8.65.2.1 state

```
trngstate\_t hal_trng_driver::state
```

Driver state.

8.65.2.2 config

```
const TRNGConfig* hal_trng_driver::config
```

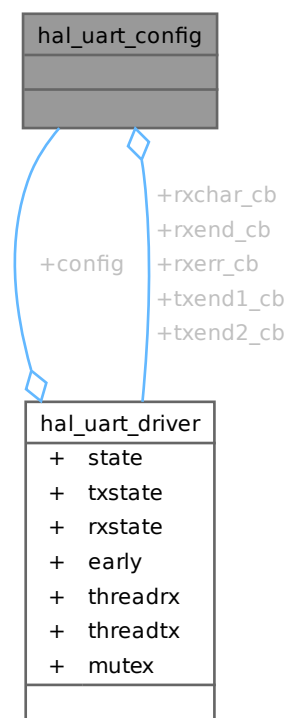
Current configuration data.

8.66 hal_uart_config Struct Reference

Driver configuration structure.

```
#include <hal_uart_lld.h>
```

Collaboration diagram for hal_uart_config:



Data Fields

- [uartcb_t txend1_cb](#)
End of transmission buffer callback.
- [uartcb_t txend2_cb](#)
Physical end of transmission callback.
- [uartcb_t rxend_cb](#)
Receive buffer filled callback.
- [uartccb_t rxchar_cb](#)
Character received while out if the `UART_RECEIVE` state.
- [uartecb_t rxerr_cb](#)
Receive error callback.

8.66.1 Detailed Description

Driver configuration structure.

Note

Implementations may extend this structure to contain more, architecture dependent, fields.

8.66.2 Field Documentation

8.66.2.1 txend1_cb

`uartcb_t hal_uart_config::txend1_cb`

End of transmission buffer callback.

8.66.2.2 txend2_cb

`uartcb_t hal_uart_config::txend2_cb`

Physical end of transmission callback.

8.66.2.3 rxend_cb

`uartcb_t hal_uart_config::rxend_cb`

Receive buffer filled callback.

8.66.2.4 rxchar_cb

`uartcb_t hal_uart_config::rxchar_cb`

Character received while out if the UART_RECEIVE state.

8.66.2.5 rxerr_cb

`uartecb_t hal_uart_config::rxerr_cb`

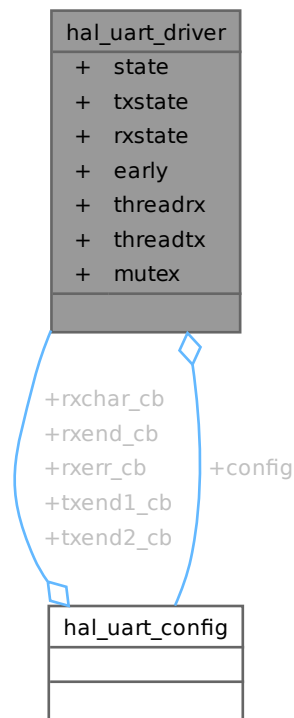
Receive error callback.

8.67 hal_uart_driver Struct Reference

Structure representing an UART driver.

```
#include <hal_uart_llld.h>
```

Collaboration diagram for hal_uart_driver:



Data Fields

- [uartstate_t state](#)
Driver state.
- [uarttxstate_t txstate](#)
Transmitter state.
- [uartrxstate_t rxstate](#)
Receiver state.
- `const UARTConfig * config`
Current configuration data.
- `bool early`
Synchronization flag for transmit operations.
- [thread_reference_t threadrx](#)
Waiting thread on RX.
- [thread_reference_t threadtx](#)
Waiting thread on TX.
- [mutex_t mutex](#)
Mutex protecting the peripheral.

8.67.1 Detailed Description

Structure representing an UART driver.

Note

Implementations may extend this structure to contain more, architecture dependent, fields.

8.67.2 Field Documentation

8.67.2.1 state

```
uartstate_t hal_uart_driver::state
```

Driver state.

8.67.2.2 txstate

```
uarttxstate_t hal_uart_driver::txstate
```

Transmitter state.

8.67.2.3 rxstate

```
uartrxstate_t hal_uart_driver::rxstate
```

Receiver state.

8.67.2.4 config

```
const UARTConfig* hal_uart_driver::config
```

Current configuration data.

8.67.2.5 early

```
bool hal_uart_driver::early
```

Synchronization flag for transmit operations.

8.67.2.6 threadrx

```
thread_reference_t hal_uart_driver::threadrx
```

Waiting thread on RX.

8.67.2.7 threadtx

```
thread_reference_t hal_uart_driver::threadtx
```

Waiting thread on TX.

8.67.2.8 mutex

```
mutex_t hal_uart_driver::mutex
```

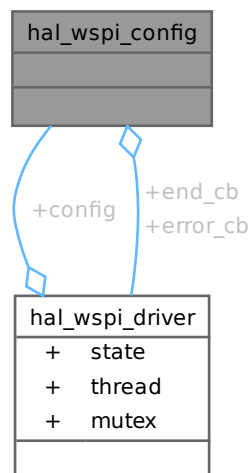
Mutex protecting the peripheral.

8.68 hal_wspi_config Struct Reference

Driver configuration structure.

```
#include <hal_wspi.h>
```

Collaboration diagram for hal_wspi_config:



Data Fields

- [`wspicallback\_t end\_cb`](#)
Operation complete callback or `NULL`.
- [`wspicallback\_t error\_cb`](#)
Operation error callback or `NULL`.

8.68.1 Detailed Description

Driver configuration structure.

8.68.2 Field Documentation

8.68.2.1 end_cb

```
wspicallback_t hal_wspi_config::end_cb
```

Operation complete callback or NULL.

8.68.2.2 error_cb

```
wspicallback_t hal_wspi_config::error_cb
```

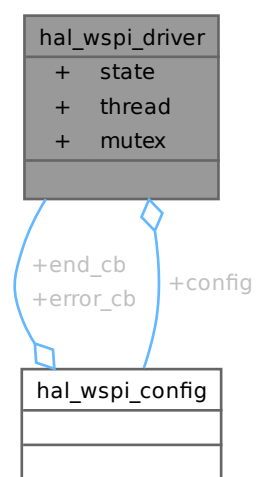
Operation error callback or NULL.

8.69 hal_wspi_driver Struct Reference

Structure representing an WSPI driver.

```
#include <hal_wspi.h>
```

Collaboration diagram for hal_wspi_driver:



Data Fields

- `wspistate_t state`
Driver state.
- `const WSPICConfig * config`
Current configuration data.
- `thread_reference_t thread`
Waiting thread.
- `mutex_t mutex`
Mutex protecting the peripheral.

8.69.1 Detailed Description

Structure representing an WSPi driver.

8.69.2 Field Documentation

8.69.2.1 state

```
wspistate_t hal_wspi_driver::state
```

Driver state.

8.69.2.2 config

```
const WSPICConfig* hal_wspi_driver::config
```

Current configuration data.

8.69.2.3 thread

```
thread_reference_t hal_wspi_driver::thread
```

Waiting thread.

8.69.2.4 mutex

```
mutex_t hal_wspi_driver::mutex
```

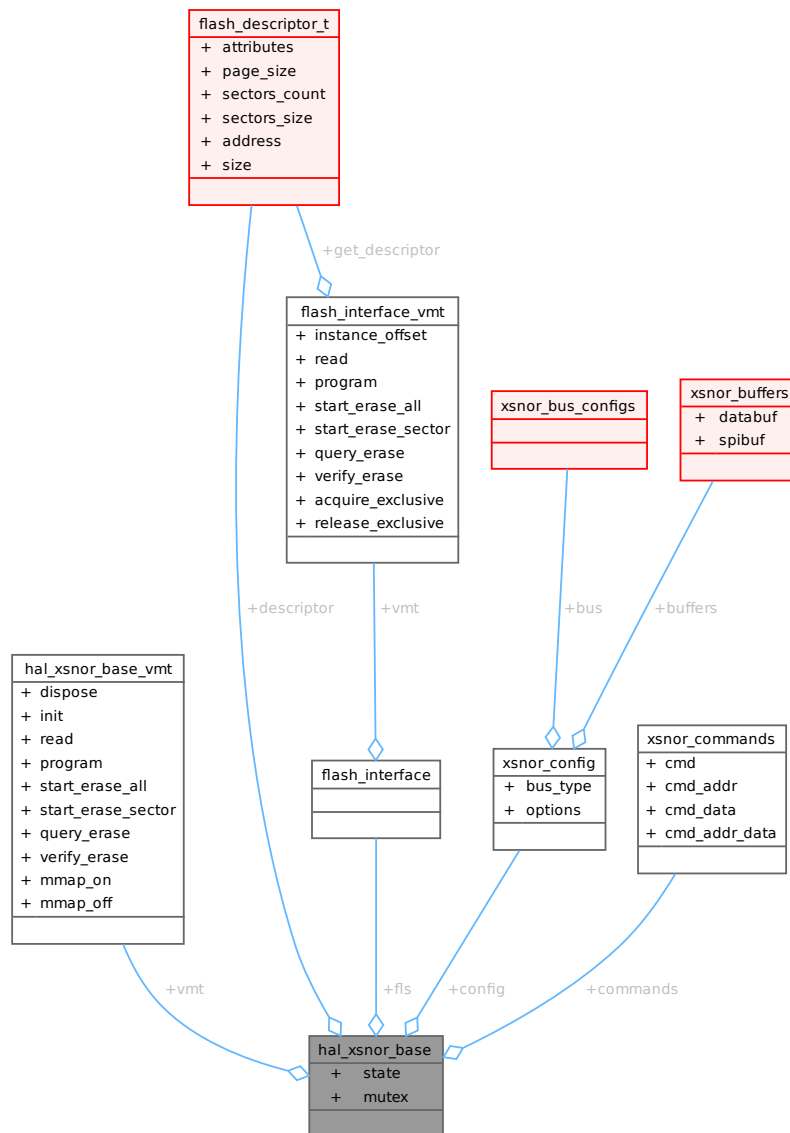
Mutex protecting the peripheral.

8.70 hal_xsnor_base Struct Reference

Structure representing a XSNOR base driver class.

```
#include <hal_xsnor_base.h>
```

Collaboration diagram for hal_xsnor_base:



Data Fields

- const struct [hal_xsnor_base_vmt](#) * `vmt`
Virtual Methods Table.
- [flash_interface_i](#) `fls`
Implemented interface [flash_interface_i](#).

- `flash_state_t` `state`
Driver state.
- `const xsnor_config_t *` `config`
Driver configuration.
- `const xsnor_commands_t *` `commands`
Current commands configuration.
- `mutex_t` `mutex`
Flash access mutex.
- `flash_descriptor_t` `descriptor`
Flash descriptor.

8.70.1 Detailed Description

Structure representing a XSNOR base driver class.

8.70.2 Field Documentation

8.70.2.1 vmt

```
const struct hal_xsnor_base_vmt* hal_xsnor_base::vmt
```

Virtual Methods Table.

8.70.2.2 fls

```
flash_interface_i hal_xsnor_base::fls
```

Implemented interface `flash_interface_i`.

8.70.2.3 state

```
flash_state_t hal_xsnor_base::state
```

Driver state.

8.70.2.4 config

```
const xsnor_config_t* hal_xsnor_base::config
```

Driver configuration.

8.70.2.5 commands

```
const xsnor_commands_t* hal_xsnor_base::commands
```

Current commands configuration.

Note

This field is meant to be initialized by subclasses on object creation.

8.70.2.6 mutex

```
mutex_t hal_xsnor_base::mutex
```

Flash access mutex.

8.70.2.7 descriptor

```
flash_descriptor_t hal_xsnor_base::descriptor
```

Flash descriptor.

Note

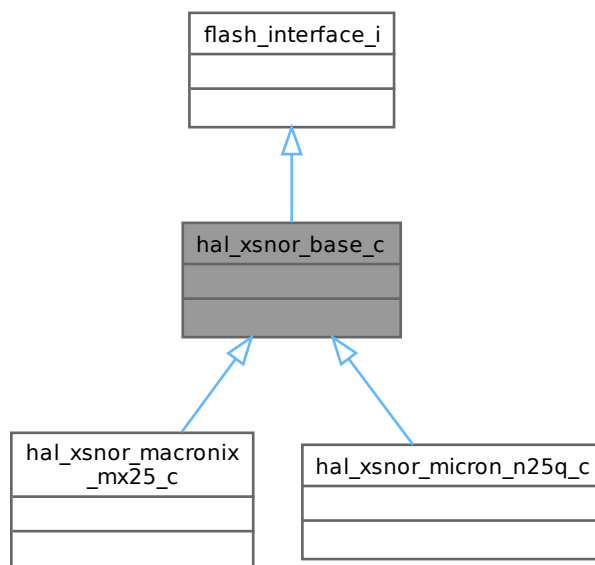
This field is meant to be initialized by subclasses on memory initialization.

8.71 hal_xsnor_base_c Class Reference

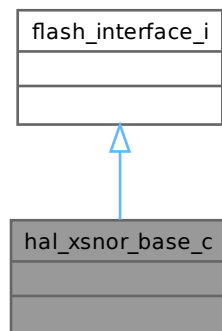
Base class of all SNOR drivers.

```
#include <hal_xsnor_base.h>
```

Inheritance diagram for hal_xsnor_base_c:



Collaboration diagram for `hal_xsnor_base_c`:



8.71.1 Detailed Description

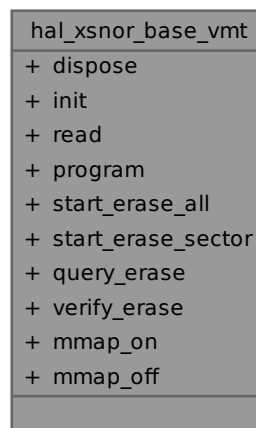
Base class of all SNOR drivers.

8.72 `hal_xsnor_base_vmt` Struct Reference

Class `hal_xsnor_base_c` virtual methods table.

```
#include <hal_xsnor_base.h>
```

Collaboration diagram for `hal_xsnor_base_vmt`:



Data Fields

- void(* [dispose](#))(void *ip)
- [flash_error_t](#)(* [init](#))(void *ip)
- [flash_error_t](#)(* [read](#))(void *ip, [flash_offset_t](#) offset, size_t n, uint8_t *rp)
- [flash_error_t](#)(* [program](#))(void *ip, [flash_offset_t](#) offset, size_t n, const uint8_t *pp)
- [flash_error_t](#)(* [start_erase_all](#))(void *ip)
- [flash_error_t](#)(* [start_erase_sector](#))(void *ip, [flash_sector_t](#) sector)
- [flash_error_t](#)(* [query_erase](#))(void *ip, unsigned *msec)
- [flash_error_t](#)(* [verify_erase](#))(void *ip, [flash_sector_t](#) sector)
- [flash_error_t](#)(* [mmap_on](#))(void *ip, uint8_t **addrp)
- void(* [mmap_off](#))(void *ip)

8.72.1 Detailed Description

Class [hal_xsnor_base_c](#) virtual methods table.

8.72.2 Field Documentation

8.72.2.1 dispose

```
void(* hal_xsnor_base_vmt::dispose) (void *ip)
```

8.72.2.2 init

```
flash\_error\_t(* hal_xsnor_base_vmt::init) (void *ip)
```

8.72.2.3 read

```
flash\_error\_t(* hal_xsnor_base_vmt::read) (void *ip, flash\_offset\_t offset, size_t n, uint8_t *rp)
```

8.72.2.4 program

```
flash\_error\_t(* hal_xsnor_base_vmt::program) (void *ip, flash\_offset\_t offset, size_t n, const uint8_t *pp)
```

8.72.2.5 start_erase_all

```
flash\_error\_t(* hal_xsnor_base_vmt::start_erase_all) (void *ip)
```

8.72.2.6 start_erase_sector

```
flash\_error\_t(* hal_xsnor_base_vmt::start_erase_sector) (void *ip, flash\_sector\_t sector)
```

8.72.2.7 query_erase

```
flash_error_t(* hal_xsnor_base_vmt::query_erase) (void *ip, unsigned *msec)
```

8.72.2.8 verify_erase

```
flash_error_t(* hal_xsnor_base_vmt::verify_erase) (void *ip, flash_sector_t sector)
```

8.72.2.9 mmap_on

```
flash_error_t(* hal_xsnor_base_vmt::mmap_on) (void *ip, uint8_t **addrp)
```

8.72.2.10 mmap_off

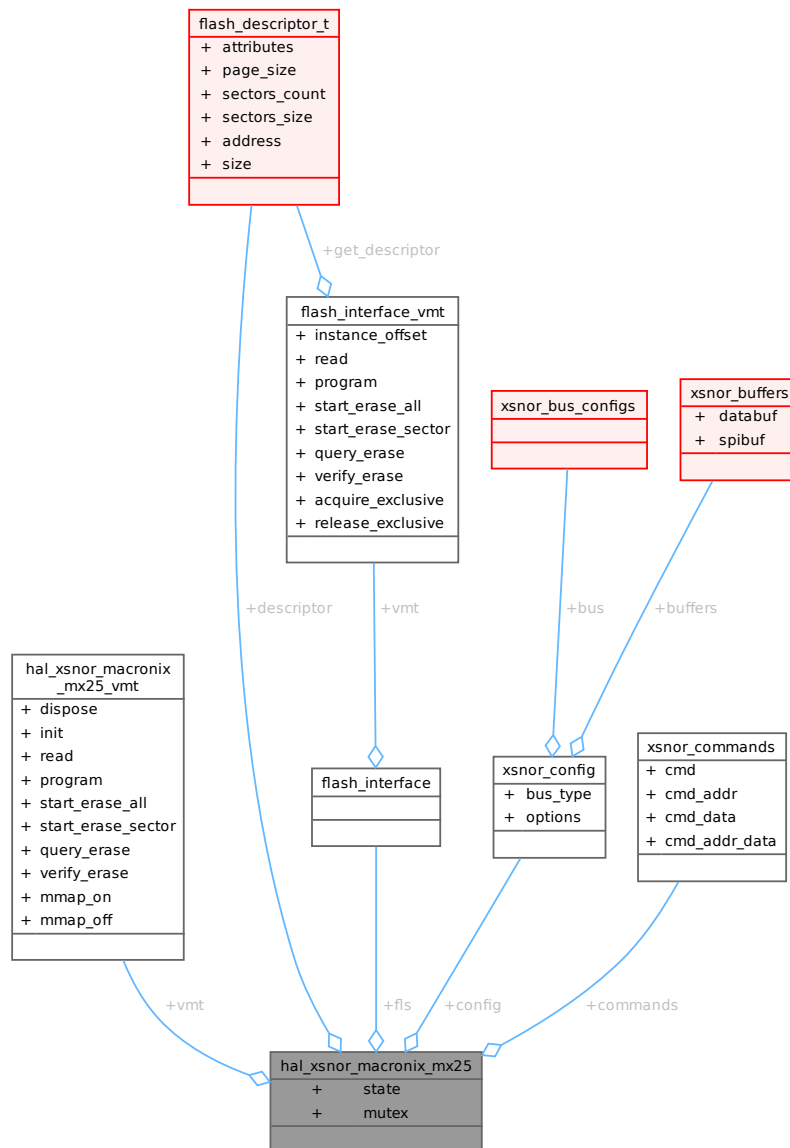
```
void(* hal_xsnor_base_vmt::mmap_off) (void *ip)
```

8.73 hal_xsnor_macronix_mx25 Struct Reference

Structure representing a SNOR Macronix MX25 driver class.

```
#include <hal_xsnor_macronix_mx25.h>
```

Collaboration diagram for hal_xsnor_macronix_mx25:



Data Fields

- const struct [hal_xsnor_macronix_mx25_vmt](#) * **vmt**
Virtual Methods Table.
- [flash_interface_i](#) **fls**
Implemented interface [flash_interface_i](#).
- [flash_state_t](#) **state**
Driver state.
- const [xsnor_config_t](#) * **config**
Driver configuration.
- const [xsnor_commands_t](#) * **commands**

Current commands configuration.

- `mutex_t mutex`

Flash access mutex.

- `flash_descriptor_t descriptor`

Flash descriptor.

8.73.1 Detailed Description

Structure representing a SNOR Macronix MX25 driver class.

8.73.2 Field Documentation

8.73.2.1 vmt

```
const struct hal_xsnor_macronix_mx25_vmt* hal_xsnor_macronix_mx25::vmt
```

Virtual Methods Table.

8.73.2.2 fls

```
flash_interface_i hal_xsnor_macronix_mx25::fls
```

Implemented interface `flash_interface_i`.

8.73.2.3 state

```
flash_state_t hal_xsnor_macronix_mx25::state
```

Driver state.

8.73.2.4 config

```
const xsnor_config_t* hal_xsnor_macronix_mx25::config
```

Driver configuration.

8.73.2.5 commands

```
const xsnor_commands_t* hal_xsnor_macronix_mx25::commands
```

Current commands configuration.

Note

This field is meant to be initialized by subclasses on object creation.

8.73.2.6 mutex

```
mutex_t hal_xsnor_macronix_mx25::mutex
```

Flash access mutex.

8.73.2.7 descriptor

```
flash_descriptor_t hal_xsnor_macronix_mx25::descriptor
```

Flash descriptor.

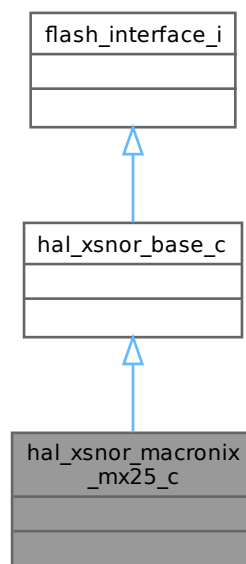
Note

This field is meant to be initialized by subclasses on memory initialization.

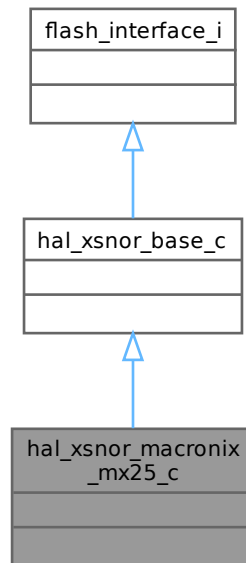
8.74 hal_xsnor_macronix_mx25_c Class Reference

```
#include <hal_xsnor_macronix_mx25.h>
```

Inheritance diagram for hal_xsnor_macronix_mx25_c:



Collaboration diagram for `hal_xsnor_macronix_mx25_c`:

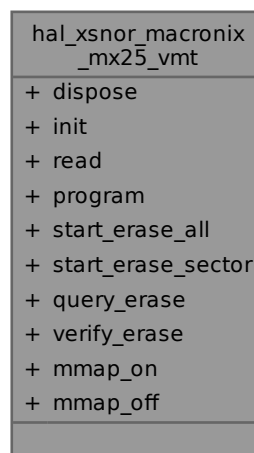


8.75 hal_xsnor_macronix_mx25_vmt Struct Reference

Class `hal_xsnor_macronix_mx25_c` virtual methods table.

```
#include <hal_xsnor_macronix_mx25.h>
```

Collaboration diagram for `hal_xsnor_macronix_mx25_vmt`:



Data Fields

- void(* [dispose](#))(void *ip)
- [flash_error_t](#)(* [init](#))(void *ip)
- [flash_error_t](#)(* [read](#))(void *ip, [flash_offset_t](#) offset, size_t n, uint8_t *rp)
- [flash_error_t](#)(* [program](#))(void *ip, [flash_offset_t](#) offset, size_t n, const uint8_t *pp)
- [flash_error_t](#)(* [start_erase_all](#))(void *ip)
- [flash_error_t](#)(* [start_erase_sector](#))(void *ip, [flash_sector_t](#) sector)
- [flash_error_t](#)(* [query_erase](#))(void *ip, unsigned *msec)
- [flash_error_t](#)(* [verify_erase](#))(void *ip, [flash_sector_t](#) sector)
- [flash_error_t](#)(* [mmap_on](#))(void *ip, uint8_t **addrp)
- void(* [mmap_off](#))(void *ip)

8.75.1 Detailed Description

Class [hal_xsnor_macronix_mx25_c](#) virtual methods table.

8.75.2 Field Documentation

8.75.2.1 dispose

```
void(* hal_xsnor_macronix_mx25_vmt::dispose) (void *ip)
```

8.75.2.2 init

```
flash\_error\_t(* hal_xsnor_macronix_mx25_vmt::init) (void *ip)
```

8.75.2.3 read

```
flash\_error\_t(* hal_xsnor_macronix_mx25_vmt::read) (void *ip, flash\_offset\_t offset, size_t n,  
uint8_t *rp)
```

8.75.2.4 program

```
flash\_error\_t(* hal_xsnor_macronix_mx25_vmt::program) (void *ip, flash\_offset\_t offset, size_t  
n, const uint8_t *pp)
```

8.75.2.5 start_erase_all

```
flash\_error\_t(* hal_xsnor_macronix_mx25_vmt::start_erase_all) (void *ip)
```

8.75.2.6 start_erase_sector

```
flash\_error\_t(* hal_xsnor_macronix_mx25_vmt::start_erase_sector) (void *ip, flash\_sector\_t  
sector)
```

8.75.2.7 query_erase

```
flash_error_t(* hal_xsnor_macronix_mx25_vmt::query_erase) (void *ip, unsigned *msec)
```

8.75.2.8 verify_erase

```
flash_error_t(* hal_xsnor_macronix_mx25_vmt::verify_erase) (void *ip, flash_sector_t sector)
```

8.75.2.9 mmap_on

```
flash_error_t(* hal_xsnor_macronix_mx25_vmt::mmap_on) (void *ip, uint8_t **addrp)
```

8.75.2.10 mmap_off

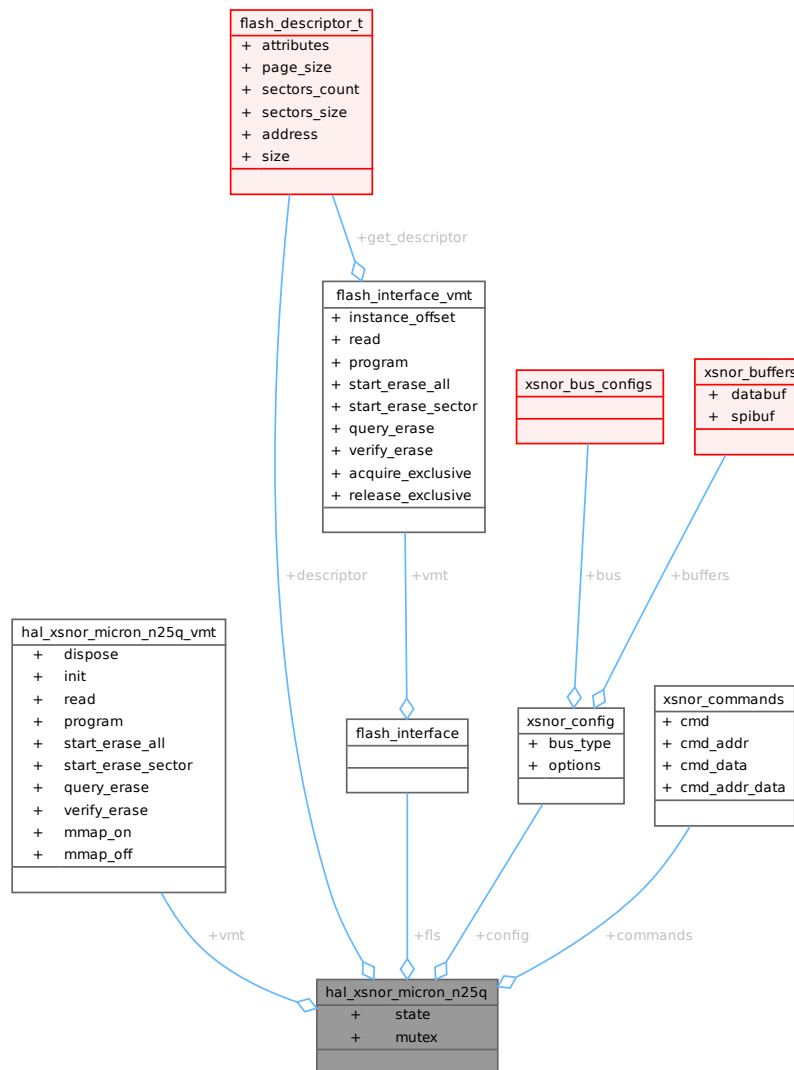
```
void(* hal_xsnor_macronix_mx25_vmt::mmap_off) (void *ip)
```

8.76 hal_xsnor_micron_n25q Struct Reference

Structure representing a SNOR Micron N25Q driver class.

```
#include <hal_xsnor_micron_n25q.h>
```

Collaboration diagram for hal_xsnor_micron_n25q:



Data Fields

- const struct [hal_xsnor_micron_n25q_vmt](#) * **vmt**
Virtual Methods Table.
- [flash_interface_i](#) **fls**
Implemented interface [flash_interface_i](#).
- [flash_state_t](#) **state**
Driver state.
- const [xsnor_config_t](#) * **config**
Driver configuration.
- const [xsnor_commands_t](#) * **commands**
Current commands configuration.
- [mutex_t](#) **mutex**
Flash access mutex.

- [flash_descriptor_t](#) descriptor

Flash descriptor.

8.76.1 Detailed Description

Structure representing a SNOR Micron N25Q driver class.

8.76.2 Field Documentation

8.76.2.1 vmt

```
const struct hal\_xsnor\_micron\_n25q\_vmt* hal_xsnor_micron_n25q::vmt
```

Virtual Methods Table.

8.76.2.2 fls

```
flash\_interface\_i hal_xsnor_micron_n25q::fls
```

Implemented interface [flash_interface_i](#).

8.76.2.3 state

```
flash\_state\_t hal_xsnor_micron_n25q::state
```

Driver state.

8.76.2.4 config

```
const xsnor\_config\_t* hal_xsnor_micron_n25q::config
```

Driver configuration.

8.76.2.5 commands

```
const xsnor\_commands\_t* hal_xsnor_micron_n25q::commands
```

Current commands configuration.

Note

This field is meant to be initialized by subclasses on object creation.

8.76.2.6 mutex

```
mutex_t hal_xsnor_micron_n25q::mutex
```

Flash access mutex.

8.76.2.7 descriptor

```
flash_descriptor_t hal_xsnor_micron_n25q::descriptor
```

Flash descriptor.

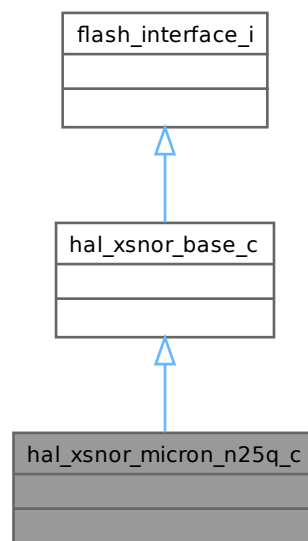
Note

This field is meant to be initialized by subclasses on memory initialization.

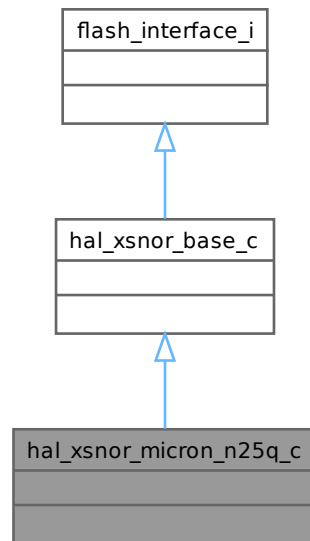
8.77 hal_xsnor_micron_n25q_c Class Reference

```
#include <hal_xsnor_micron_n25q.h>
```

Inheritance diagram for hal_xsnor_micron_n25q_c:



Collaboration diagram for `hal_xsnor_micron_n25q_c`:

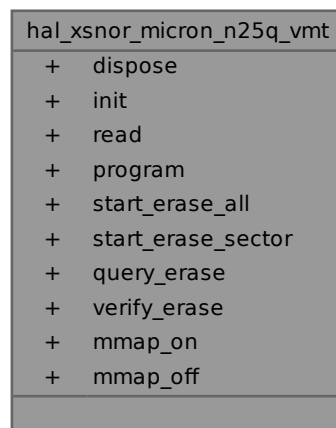


8.78 `hal_xsnor_micron_n25q_vmt` Struct Reference

Class `hal_xsnor_micron_n25q_c` virtual methods table.

```
#include <hal_xsnor_micron_n25q.h>
```

Collaboration diagram for `hal_xsnor_micron_n25q_vmt`:



Data Fields

- void(* [dispose](#))(void *ip)
- [flash_error_t](#)(* [init](#))(void *ip)
- [flash_error_t](#)(* [read](#))(void *ip, [flash_offset_t](#) offset, size_t n, uint8_t *rp)
- [flash_error_t](#)(* [program](#))(void *ip, [flash_offset_t](#) offset, size_t n, const uint8_t *pp)
- [flash_error_t](#)(* [start_erase_all](#))(void *ip)
- [flash_error_t](#)(* [start_erase_sector](#))(void *ip, [flash_sector_t](#) sector)
- [flash_error_t](#)(* [query_erase](#))(void *ip, unsigned *msec)
- [flash_error_t](#)(* [verify_erase](#))(void *ip, [flash_sector_t](#) sector)
- [flash_error_t](#)(* [mmap_on](#))(void *ip, uint8_t **addrp)
- void(* [mmap_off](#))(void *ip)

8.78.1 Detailed Description

Class [hal_xsnor_micron_n25q_c](#) virtual methods table.

8.78.2 Field Documentation

8.78.2.1 dispose

```
void(* hal_xsnor_micron_n25q_vmt::dispose) (void *ip)
```

8.78.2.2 init

```
flash\_error\_t(* hal_xsnor_micron_n25q_vmt::init) (void *ip)
```

8.78.2.3 read

```
flash\_error\_t(* hal_xsnor_micron_n25q_vmt::read) (void *ip, flash\_offset\_t offset, size_t n,  
uint8_t *rp)
```

8.78.2.4 program

```
flash\_error\_t(* hal_xsnor_micron_n25q_vmt::program) (void *ip, flash\_offset\_t offset, size_t n,  
const uint8_t *pp)
```

8.78.2.5 start_erase_all

```
flash\_error\_t(* hal_xsnor_micron_n25q_vmt::start_erase_all) (void *ip)
```

8.78.2.6 start_erase_sector

```
flash\_error\_t(* hal_xsnor_micron_n25q_vmt::start_erase_sector) (void *ip, flash\_sector\_t sector)
```

8.78.2.7 query_erase

```
flash_error_t (* hal_xsnor_micron_n25q_vmt::query_erase) (void *ip, unsigned *msec)
```

8.78.2.8 verify_erase

```
flash_error_t (* hal_xsnor_micron_n25q_vmt::verify_erase) (void *ip, flash_sector_t sector)
```

8.78.2.9 mmap_on

```
flash_error_t (* hal_xsnor_micron_n25q_vmt::mmap_on) (void *ip, uint8_t **addrp)
```

8.78.2.10 mmap_off

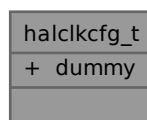
```
void (* hal_xsnor_micron_n25q_vmt::mmap_off) (void *ip)
```

8.79 halclkcfg_t Struct Reference

Type of a clock configuration structure.

```
#include <hal_lld.h>
```

Collaboration diagram for halclkcfg_t:



Data Fields

- uint32_t [dummy](#)

8.79.1 Detailed Description

Type of a clock configuration structure.

8.79.2 Field Documentation

8.79.2.1 dummy

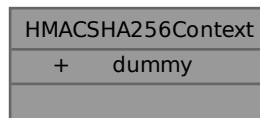
```
uint32_t halclkcfg_t::dummy
```

8.80 HMACSHA256Context Struct Reference

Type of a HMAC_SHA256 context.

```
#include <hal_crypto_lld.h>
```

Collaboration diagram for HMACSHA256Context:



Data Fields

- `uint32_t dummy`

8.80.1 Detailed Description

Type of a HMAC_SHA256 context.

8.80.2 Field Documentation

8.80.2.1 dummy

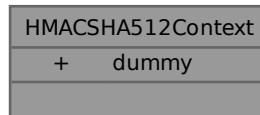
```
uint32_t HMACSHA256Context::dummy
```

8.81 HMACSHA512Context Struct Reference

Type of a HMAC_SHA512 context.

```
#include <hal_crypto_ll.h>
```

Collaboration diagram for HMACSHA512Context:



Data Fields

- `uint32_t dummy`

8.81.1 Detailed Description

Type of a HMAC_SHA512 context.

8.81.2 Field Documentation

8.81.2.1 dummy

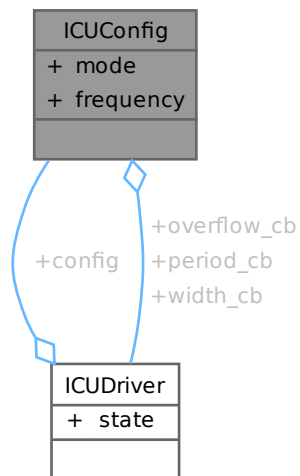
```
uint32_t HMACSHA512Context::dummy
```

8.82 ICUConfig Struct Reference

Driver configuration structure.

```
#include <hal_icu_ll.h>
```

Collaboration diagram for ICUConfig:



Data Fields

- [icumode_t mode](#)
Driver mode.
- [icufreq_t frequency](#)
Timer clock in Hz.
- [icucallback_t width_cb](#)
Callback for pulse width measurement.
- [icucallback_t period_cb](#)
Callback for cycle period measurement.
- [icucallback_t overflow_cb](#)
Callback for timer overflow.

8.82.1 Detailed Description

Driver configuration structure.

Note

It could be empty on some architectures.

8.82.2 Field Documentation

8.82.2.1 mode

[icumode_t](#) ICUConfig::mode

Driver mode.

8.82.2.2 frequency

`icufreq_t` `ICUConfig::frequency`

Timer clock in Hz.

Note

The low level can use assertions in order to catch invalid frequency specifications.

8.82.2.3 width_cb

`icucallback_t` `ICUConfig::width_cb`

Callback for pulse width measurement.

8.82.2.4 period_cb

`icucallback_t` `ICUConfig::period_cb`

Callback for cycle period measurement.

8.82.2.5 overflow_cb

`icucallback_t` `ICUConfig::overflow_cb`

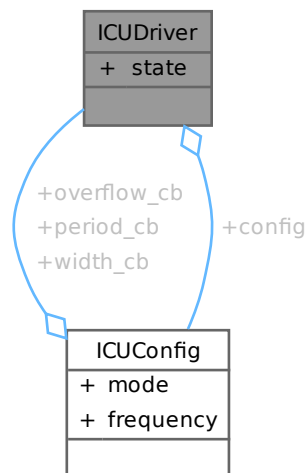
Callback for timer overflow.

8.83 ICUDriver Struct Reference

Structure representing an ICU driver.

```
#include <hal_icu_lld.h>
```

Collaboration diagram for ICUDriver:



Data Fields

- `icustate_t` `state`
Driver state.
- `const ICUConfig *` `config`
Current configuration data.

8.83.1 Detailed Description

Structure representing an ICU driver.

8.83.2 Field Documentation

8.83.2.1 state

```
icustate_t ICUDriver::state
```

Driver state.

8.83.2.2 config

```
const ICUConfig* ICUDriver::config
```

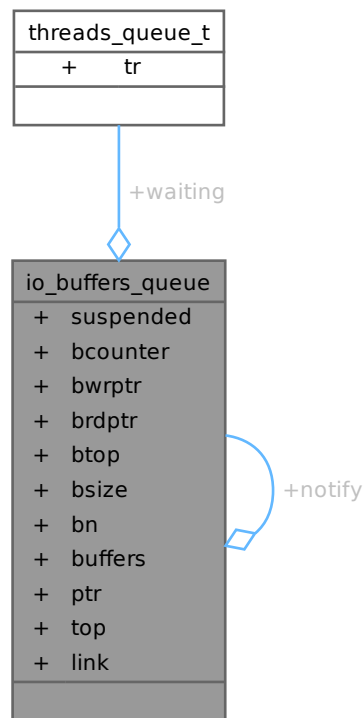
Current configuration data.

8.84 io_buffers_queue Struct Reference

Structure of a generic buffers queue.

```
#include <hal_buffers.h>
```

Collaboration diagram for `io_buffers_queue`:



Data Fields

- `threads_queue_t` `waiting`
Queue of waiting threads.
- `bool` `suspended`
Queue suspended state flag.
- `volatile size_t` `bcounter`
Active buffers counter.
- `uint8_t *` `bwrptr`
Buffer write pointer.
- `uint8_t *` `brdptr`
Buffer read pointer.
- `uint8_t *` `btop`
Pointer to the buffers boundary.
- `size_t` `bsize`
Size of buffers.
- `size_t` `bn`
Number of buffers.
- `uint8_t *` `buffers`
Queue of buffer objects.
- `uint8_t *` `ptr`

- Pointer for R/W sequential access.*
 - `uint8_t * top`
Boundary for R/W sequential access.
- `bqnotify_t notify`
Data notification callback.
- `void * link`
Application defined field.

8.84.1 Detailed Description

Structure of a generic buffers queue.

8.84.2 Field Documentation

8.84.2.1 waiting

```
threads_queue_t io_buffers_queue::waiting
```

Queue of waiting threads.

8.84.2.2 suspended

```
bool io_buffers_queue::suspended
```

Queue suspended state flag.

8.84.2.3 bcounter

```
volatile size_t io_buffers_queue::bcounter
```

Active buffers counter.

8.84.2.4 bwrptr

```
uint8_t* io_buffers_queue::bwrptr
```

Buffer write pointer.

8.84.2.5 brdptr

```
uint8_t* io_buffers_queue::brdptr
```

Buffer read pointer.

8.84.2.6 btop

```
uint8_t* io_buffers_queue::btop
```

Pointer to the buffers boundary.

8.84.2.7 bsize

```
size_t io_buffers_queue::bsize
```

Size of buffers.

Note

The buffer size must be not lower than `sizeof(size_t) + 2` because the first bytes are used to store the used size of the buffer.

8.84.2.8 bn

```
size_t io_buffers_queue::bn
```

Number of buffers.

8.84.2.9 buffers

```
uint8_t* io_buffers_queue::buffers
```

Queue of buffer objects.

8.84.2.10 ptr

```
uint8_t* io_buffers_queue::ptr
```

Pointer for R/W sequential access.

Note

It is `NULL` if a new buffer must be fetched from the queue.

8.84.2.11 top

```
uint8_t* io_buffers_queue::top
```

Boundary for R/W sequential access.

8.84.2.12 notify

```
bqnotify_t io_buffers_queue::notify
```

Data notification callback.

8.84.2.13 link

```
void* io_buffers_queue::link
```

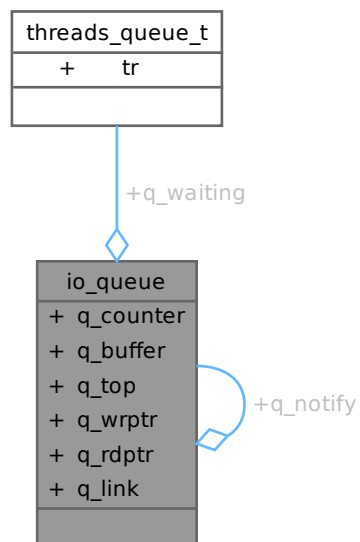
Application defined field.

8.85 io_queue Struct Reference

Generic I/O queue structure.

```
#include <hal_queues.h>
```

Collaboration diagram for io_queue:



Data Fields

- [threads_queue_t q_waiting](#)
Queue of waiting threads.
- volatile size_t [q_counter](#)
Resources counter.
- uint8_t * [q_buffer](#)
Pointer to the queue buffer.
- uint8_t * [q_top](#)
Pointer to the first location after the buffer.
- uint8_t * [q_wrpтр](#)
Write pointer.
- uint8_t * [q_rdpтр](#)
Read pointer.
- [qnotify_t q_notify](#)
Data notification callback.
- void * [q_link](#)
Application defined field.

8.85.1 Detailed Description

Generic I/O queue structure.

This structure represents a generic Input or Output asymmetrical queue. The queue is asymmetrical because one end is meant to be accessed from a thread context, and thus can be blocking, the other end is accessible from interrupt handlers or from within a kernel lock zone and is non-blocking.

8.85.2 Field Documentation

8.85.2.1 q_waiting

```
threads_queue_t io_queue::q_waiting
```

Queue of waiting threads.

8.85.2.2 q_counter

```
volatile size_t io_queue::q_counter
```

Resources counter.

8.85.2.3 q_buffer

```
uint8_t* io_queue::q_buffer
```

Pointer to the queue buffer.

8.85.2.4 q_top

```
uint8_t* io_queue::q_top
```

Pointer to the first location after the buffer.

8.85.2.5 q_wrptr

```
uint8_t* io_queue::q_wrptr
```

Write pointer.

8.85.2.6 q_rdptr

```
uint8_t* io_queue::q_rdptr
```

Read pointer.

8.85.2.7 q_notify

```
qnotify_t io_queue::q_notify
```

Data notification callback.

8.85.2.8 q_link

```
void* io_queue::q_link
```

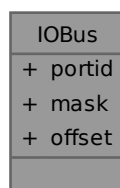
Application defined field.

8.86 IOBus Struct Reference

I/O bus descriptor.

```
#include <hal_pal.h>
```

Collaboration diagram for IOBus:



Data Fields

- [ioportid_t portid](#)
Port identifier.
- [ioportmask_t mask](#)
Bus mask aligned to port bit 0.
- [uint_fast8_t offset](#)
Offset, within the port, of the least significant bit of the bus.

8.86.1 Detailed Description

I/O bus descriptor.

This structure describes a group of contiguous digital I/O lines that have to be handled as bus.

Note

I/O operations on a bus do not affect I/O lines on the same port but not belonging to the bus.

8.86.2 Field Documentation

8.86.2.1 portid

[ioportid_t](#) IOBus::portid

Port identifier.

8.86.2.2 mask

[ioportmask_t](#) IOBus::mask

Bus mask aligned to port bit 0.

Note

The bus mask implicitly define the bus width. A logic AND is performed on the bus data.

8.86.2.3 offset

[uint_fast8_t](#) IOBus::offset

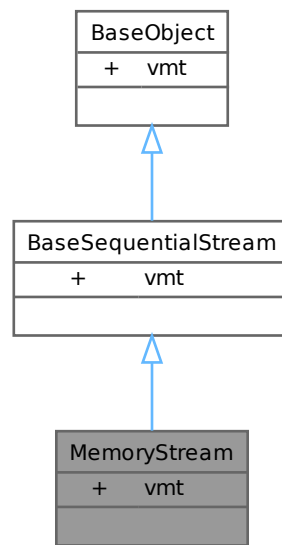
Offset, within the port, of the least significant bit of the bus.

8.87 MemoryStream Struct Reference

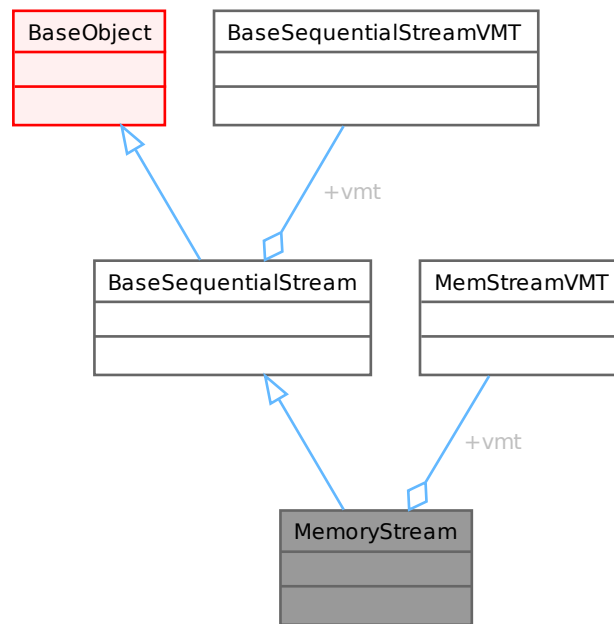
Memory stream object.

```
#include <memstreams.h>
```

Inheritance diagram for MemoryStream:



Collaboration diagram for MemoryStream:



Data Fields

- const struct [MemStreamVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseSequentialStream](#)

- const struct [BaseSequentialStreamVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.87.1 Detailed Description

Memory stream object.

8.87.2 Field Documentation

8.87.2.1 vmt

```
const struct MemStreamVMT* MemoryStream::vmt
```

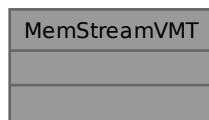
Virtual Methods Table.

8.88 MemStreamVMT Struct Reference

MemStream virtual methods table.

```
#include <memstreams.h>
```

Collaboration diagram for MemStreamVMT:



8.88.1 Detailed Description

MemStream virtual methods table.

8.89 mfs_bank_header_t Union Reference

Type of a bank header.

```
#include <hal_mfs.h>
```

Collaboration diagram for mfs_bank_header_t:

mfs_bank_header_t
+ magic1
+ magic2
+ counter
+ reserved1
+ crc
+ fields
+ hdr8
+ hdr32

Data Fields

- struct {
 - uint32_t [magic1](#)
Bank magic 1.
 - uint32_t [magic2](#)
Bank magic 2.
 - uint32_t [counter](#)
Usage counter of the bank.
 - uint16_t [reserved1](#)
Reserved field.
 - uint16_t [crc](#)
Header CRC.
- [fields](#)
- uint8_t [hdr8](#) [16]
- uint32_t [hdr32](#) [4]

8.89.1 Detailed Description

Type of a bank header.

Note

The header resides in the first 16 bytes of a bank.

8.89.2 Field Documentation

8.89.2.1 magic1

```
uint32_t mfs_bank_header_t::magic1
```

Bank magic 1.

8.89.2.2 magic2

```
uint32_t mfs_bank_header_t::magic2
```

Bank magic 2.

8.89.2.3 counter

```
uint32_t mfs_bank_header_t::counter
```

Usage counter of the bank.

This value is increased each time a bank swap is performed. It indicates how much wearing the flash has already endured.

8.89.2.4 reserved1

```
uint16_t mfs_bank_header_t::reserved1
```

Reserved field.

8.89.2.5 crc

```
uint16_t mfs_bank_header_t::crc
```

Header CRC.

8.89.2.6 [struct]

```
struct { ... } mfs_bank_header_t::fields
```

8.89.2.7 hdr8

```
uint8_t mfs_bank_header_t::hdr8[16]
```

8.89.2.8 hdr32

```
uint32_t mfs_bank_header_t::hdr32[4]
```

8.90 mfs_data_header_t Union Reference

Type of a data block header.

```
#include <hal_mfs.h>
```

Collaboration diagram for mfs_data_header_t:

mfs_data_header_t	
+	magic1
+	magic2
+	id
+	crc
+	size
+	fields
+	hdr8
+	hdr32

Data Fields

- struct {
 - uint32_t [magic1](#)
Data header magic 1.
 - uint32_t [magic2](#)
Data header magic 2.
 - uint16_t [id](#)
Record identifier.
 - uint16_t [crc](#)
Data CRC.
 - uint32_t [size](#)
Data size.
 - } [fields](#)
- uint8_t [hdr8](#) [16]
- uint32_t [hdr32](#) [4]

8.90.1 Detailed Description

Type of a data block header.

This structure is placed before each written data block.

8.90.2 Field Documentation

8.90.2.1 magic1

```
uint32_t mfs_data_header_t::magic1
```

Data header magic 1.

8.90.2.2 magic2

```
uint32_t mfs_data_header_t::magic2
```

Data header magic 2.

8.90.2.3 id

```
uint16_t mfs_data_header_t::id
```

Record identifier.

8.90.2.4 crc

```
uint16_t mfs_data_header_t::crc
```

Data CRC.

8.90.2.5 size

```
uint32_t mfs_data_header_t::size
```

Data size.

Note

The next record is located at [MFS_ALIGN_NEXT\(size\)](#).

8.90.2.6 [struct]

```
struct { ... } mfs_data_header_t::fields
```

8.90.2.7 hdr8

```
uint8_t mfs_data_header_t::hdr8[16]
```

8.90.2.8 hdr32

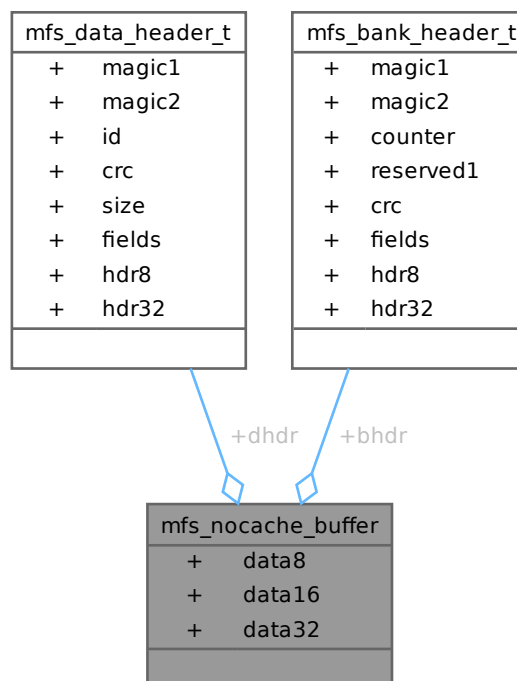
```
uint32_t mfs_data_header_t::hdr32[4]
```

8.91 mfs_nocache_buffer Union Reference

Type of an non-cacheable MFS buffer.

```
#include <hal_mfs.h>
```

Collaboration diagram for mfs_nocache_buffer:



Data Fields

- [mfs_data_header_t dhdr](#)
- [mfs_bank_header_t bhdr](#)
- [uint8_t data8 \[MFS_CFG_BUFFER_SIZE\]](#)
- [uint16_t data16 \[MFS_CFG_BUFFER_SIZE/sizeof\(uint16_t\)\]](#)
- [uint32_t data32 \[MFS_CFG_BUFFER_SIZE/sizeof\(uint32_t\)\]](#)

8.91.1 Detailed Description

Type of an non-cacheable MFS buffer.

8.91.2 Field Documentation

8.91.2.1 dhdr

`mfs_data_header_t` `mfs_nocache_buffer::dhdr`

8.91.2.2 bhdr

`mfs_bank_header_t` `mfs_nocache_buffer::bhdr`

8.91.2.3 data8

`uint8_t` `mfs_nocache_buffer::data8[MFS_CFG_BUFFER_SIZE]`

8.91.2.4 data16

`uint16_t` `mfs_nocache_buffer::data16[MFS_CFG_BUFFER_SIZE/sizeof(uint16_t)]`

8.91.2.5 data32

`uint32_t` `mfs_nocache_buffer::data32[MFS_CFG_BUFFER_SIZE/sizeof(uint32_t)]`

8.92 mfs_record_descriptor_t Struct Reference

```
#include <hal_mfs.h>
```

Collaboration diagram for `mfs_record_descriptor_t`:

mfs_record_descriptor_t	
+	offset
+	size

Data Fields

- `flash_offset_t` `offset`
Offset of the record header.
- `uint32_t` `size`
Record data size.

8.92.1 Field Documentation

8.92.1.1 offset

```
flash_offset_t mfs_record_descriptor_t::offset
```

Offset of the record header.

8.92.1.2 size

```
uint32_t mfs_record_descriptor_t::size
```

Record data size.

8.93 mfs_transaction_op_t Struct Reference

Type of a buffered write/erase operation within a transaction.

```
#include <hal_mfs.h>
```

Collaboration diagram for mfs_transaction_op_t:

mfs_transaction_op_t	
+	offset
+	size
+	id

Data Fields

- [flash_offset_t offset](#)
Written header offset.
- [size_t size](#)
Written data size.
- [mfs_id_t id](#)
Record identifier.

8.93.1 Detailed Description

Type of a buffered write/erase operation within a transaction.

8.93.2 Field Documentation

8.93.2.1 offset

`flash_offset_t mfs_transaction_op_t::offset`

Written header offset.

8.93.2.2 size

`size_t mfs_transaction_op_t::size`

Written data size.

8.93.2.3 id

`mfs_id_t mfs_transaction_op_t::id`

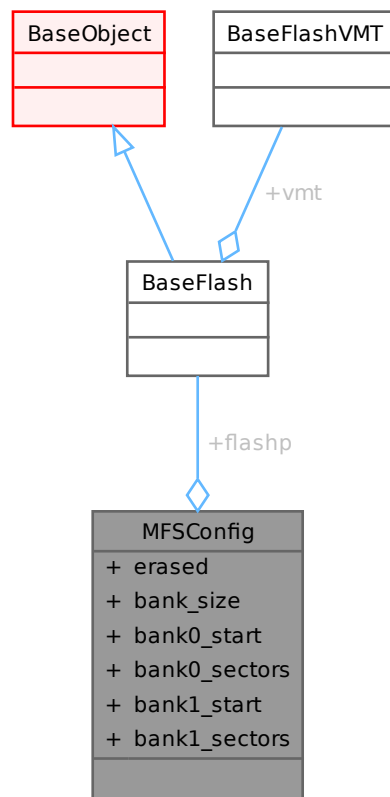
Record identifier.

8.94 MFSCConfig Struct Reference

Type of a MFS configuration structure.

```
#include <hal_mfs.h>
```

Collaboration diagram for MFSSConfig:



Data Fields

- [BaseFlash](#) * `flashp`
Flash driver associated to this MFS instance.
- `uint32_t` `erased`
Erased value.
- `flash_offset_t` `bank_size`
Banks size.
- `flash_sector_t` `bank0_start`
Base sector index for bank 0.
- `flash_sector_t` `bank0_sectors`
Number of sectors for bank 0.
- `flash_sector_t` `bank1_start`
Base sector index for bank 1.
- `flash_sector_t` `bank1_sectors`
Number of sectors for bank 1.

8.94.1 Detailed Description

Type of a MFS configuration structure.

8.94.2 Field Documentation

8.94.2.1 flashp

`BaseFlash*` MFSStruct::flashp

Flash driver associated to this MFS instance.

8.94.2.2 erased

`uint32_t` MFSStruct::erased

Erased value.

8.94.2.3 bank_size

`flash_offset_t` MFSStruct::bank_size

Banks size.

8.94.2.4 bank0_start

`flash_sector_t` MFSStruct::bank0_start

Base sector index for bank 0.

8.94.2.5 bank0_sectors

`flash_sector_t` MFSStruct::bank0_sectors

Number of sectors for bank 0.

Note

The total size of bank0 sectors must be greater or equal to `bank_size`.

8.94.2.6 bank1_start

`flash_sector_t` MFSStruct::bank1_start

Base sector index for bank 1.

8.94.2.7 bank1_sectors

`flash_sector_t` MFSConfig::bank1_sectors

Number of sectors for bank 1.

Note

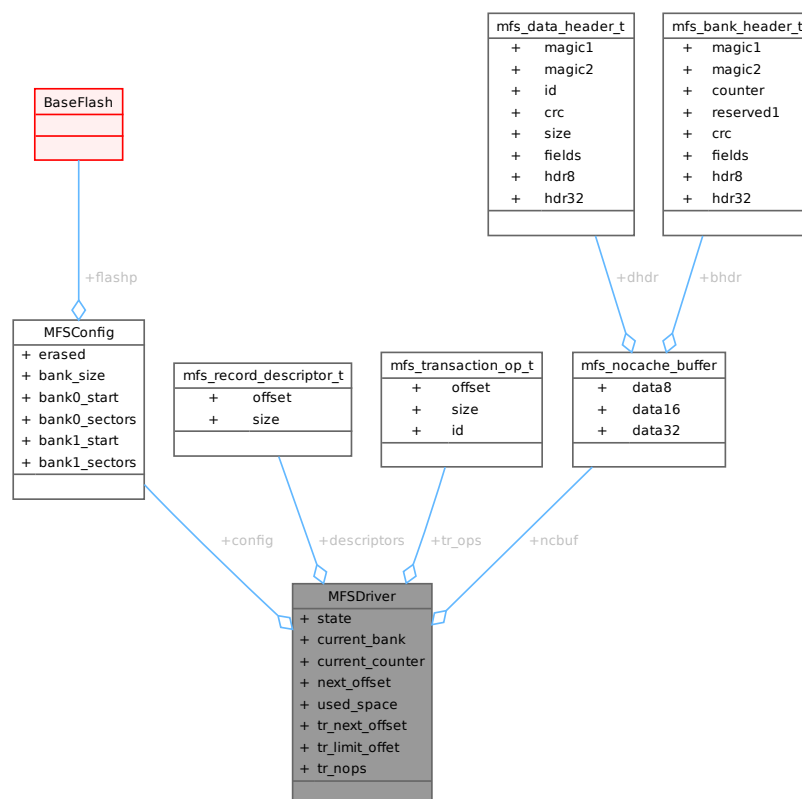
The total size of bank1 sectors must be greater or equal to `bank_size`.

8.95 MFSDriver Struct Reference

Type of an MFS instance.

```
#include <hal_mfs.h>
```

Collaboration diagram for MFSDriver:



Data Fields

- [mfs_state_t state](#)
Driver state.
- `const MFSCfg * config`
Current configuration data.
- [mfs_bank_t current_bank](#)
Bank currently in use.
- `uint32_t current_counter`
Usage counter of the current bank.
- [flash_offset_t next_offset](#)
Pointer to the next free position in the current bank.
- [flash_offset_t used_space](#)
Used space in the current bank without considering erased records.
- [mfs_record_descriptor_t descriptors](#) [MFS_CFG_MAX_RECORDS]
Offsets of the most recent instance of the records.
- [flash_offset_t tr_next_offset](#)
Next write offset for current transaction.
- [flash_offset_t tr_limit_offset](#)
Maximum offset for the transaction.
- `uint32_t tr_nops`
Number of buffered operations in current transaction.
- [mfs_transaction_op_t tr_ops](#) [MFS_CFG_TRANSACTION_MAX]
Buffered operations in current transaction.
- [mfs_nocache_buffer_t * ncbuf](#)
Associated non-cacheable buffer.

8.95.1 Detailed Description

Type of an MFS instance.

8.95.2 Field Documentation

8.95.2.1 state

```
mfs_state_t MFSDriver::state
```

Driver state.

8.95.2.2 config

```
const MFSCfg* MFSDriver::config
```

Current configuration data.

8.95.2.3 current_bank

`mfs_bank_t` MFSDriver::current_bank

Bank currently in use.

8.95.2.4 current_counter

`uint32_t` MFSDriver::current_counter

Usage counter of the current bank.

8.95.2.5 next_offset

`flash_offset_t` MFSDriver::next_offset

Pointer to the next free position in the current bank.

8.95.2.6 used_space

`flash_offset_t` MFSDriver::used_space

Used space in the current bank without considering erased records.

8.95.2.7 descriptors

`mfs_record_descriptor_t` MFSDriver::descriptors[MFS_CFG_MAX_RECORDS]

Offsets of the most recent instance of the records.

Note

Zero means that there is not a record with that id.

8.95.2.8 tr_next_offset

`flash_offset_t` MFSDriver::tr_next_offset

Next write offset for current transaction.

8.95.2.9 tr_limit_offset

`flash_offset_t` MFSDriver::tr_limit_offset

Maximum offset for the transaction.

8.95.2.10 tr_nops

```
uint32_t MFSDriver::tr_nops
```

Number of buffered operations in current transaction.

8.95.2.11 tr_ops

```
mfs_transaction_op_t MFSDriver::tr_ops[MFS_CFG_TRANSACTION_MAX]
```

Buffered operations in current transaction.

8.95.2.12 ncbuf

```
mfs_nocache_buffer_t* MFSDriver::ncbuf
```

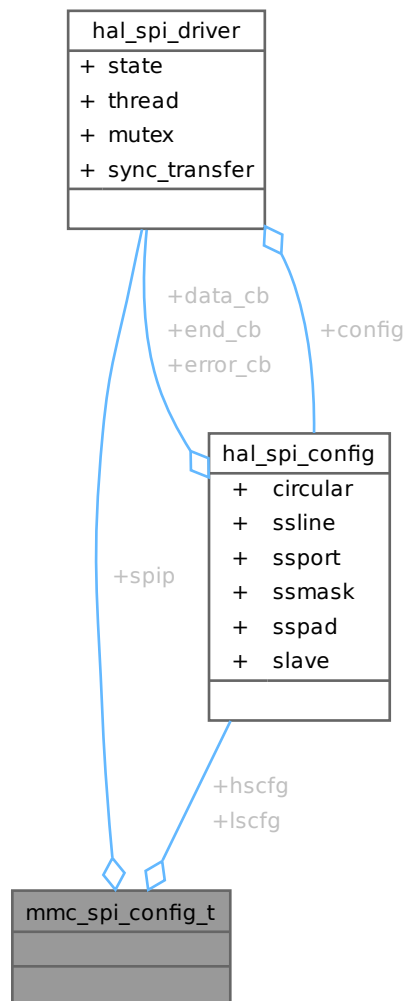
Associated non-cacheable buffer.

8.96 mmc_spi_config_t Struct Reference

Type of a MMC/SD over SPI driver configuration structure.

```
#include <hal_mmc_spi.h>
```

Collaboration diagram for `mmc_spi_config_t`:



Data Fields

- `SPIDriver * spip`
SPI driver associated to this MMC driver.
- `const SPIConfig * lscfg`
SPI low speed configuration used during initialization.
- `const SPIConfig * hscfg`
SPI high speed configuration used during transfers.

8.96.1 Detailed Description

Type of a MMC/SD over SPI driver configuration structure.

8.96.2 Field Documentation

8.96.2.1 spip

```
SPIDriver* mmc_spi_config_t::spip
```

SPI driver associated to this MMC driver.

8.96.2.2 lscfg

```
const SPIConfig* mmc_spi_config_t::lscfg
```

SPI low speed configuration used during initialization.

8.96.2.3 hscfg

```
const SPIConfig* mmc_spi_config_t::hscfg
```

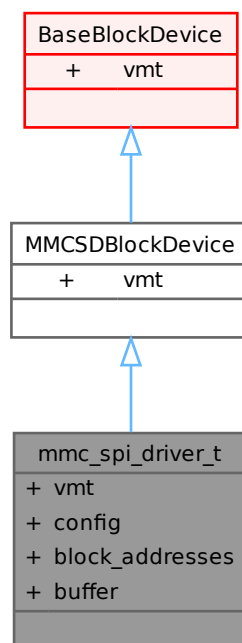
SPI high speed configuration used during transfers.

8.97 mmc_spi_driver_t Struct Reference

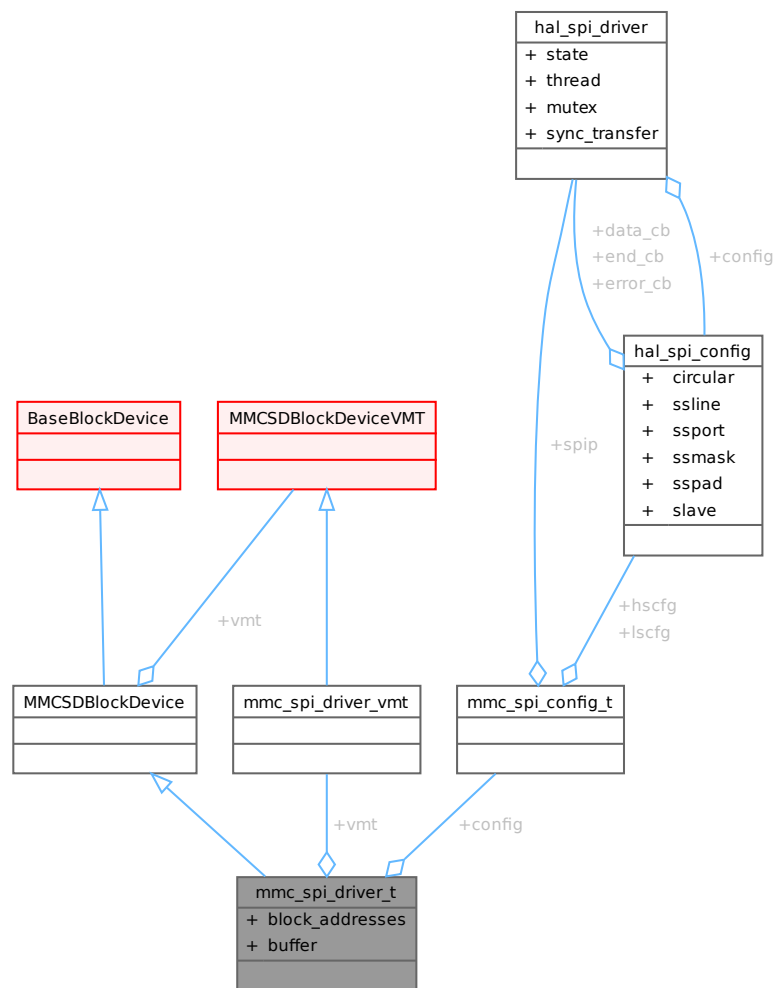
Structure representing a MMC/SD over SPI driver.

```
#include <hal_mmc_spi.h>
```

Inheritance diagram for mmc_spi_driver_t:



Collaboration diagram for `mmc_spi_driver_t`:



Data Fields

- const struct `mmc_spi_driver_vmt` * `vmt`
Virtual Methods Table.
- `_mmcsd_block_device_data` const `mmc_spi_config_t` * `config`
Current configuration data.
- bool `block_addresses`
Addresses use blocks instead of bytes.
- `uint8_t` * `buffer`
Pointer to an un-cacheable buffer of size `MMC_BUFFER_SIZE`.

Data Fields inherited from `MMCSDBlockDevice`

- const struct `MMCSDBlockDeviceVMT` * `vmt`
Virtual Methods Table.

Data Fields inherited from [BaseBlockDevice](#)

- const struct [BaseBlockDeviceVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.97.1 Detailed Description

Structure representing a MMC/SD over SPI driver.

8.97.2 Field Documentation

8.97.2.1 vmt

```
const struct mmc_spi_driver_vmt* mmc_spi_driver_t::vmt
```

Virtual Methods Table.

8.97.2.2 config

```
_mmc_ssd_block_device_data const mmc_spi_config_t* mmc_spi_driver_t::config
```

Current configuration data.

8.97.2.3 block_addresses

```
bool mmc_spi_driver_t::block_addresses
```

Addresses use blocks instead of bytes.

8.97.2.4 buffer

```
uint8_t* mmc_spi_driver_t::buffer
```

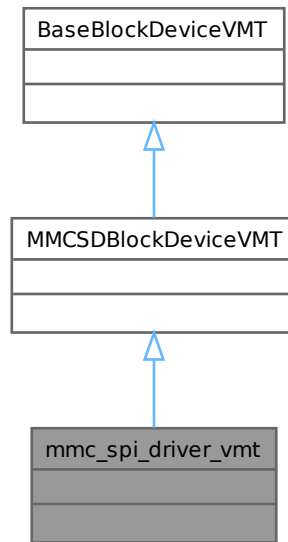
Pointer to an un-cacheable buffer of size MMC_BUFFER_SIZE.

8.98 mmc_spi_driver_vmt Struct Reference

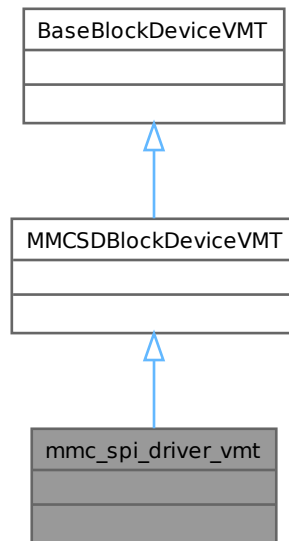
[MMCDriver](#) virtual methods table.

```
#include <hal_mmc_spi.h>
```

Inheritance diagram for mmc_spi_driver_vmt:



Collaboration diagram for mmc_spi_driver_vmt:



8.98.1 Detailed Description

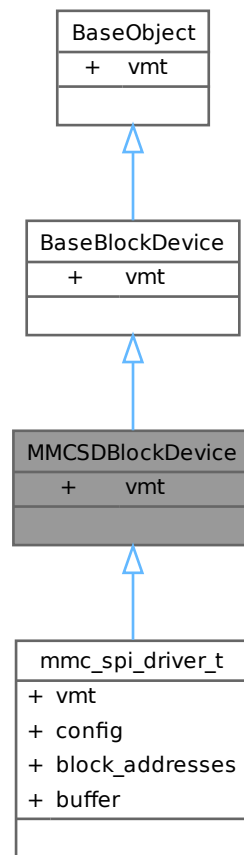
`MMCDriver` virtual methods table.

8.99 MMCSDBlockDevice Struct Reference

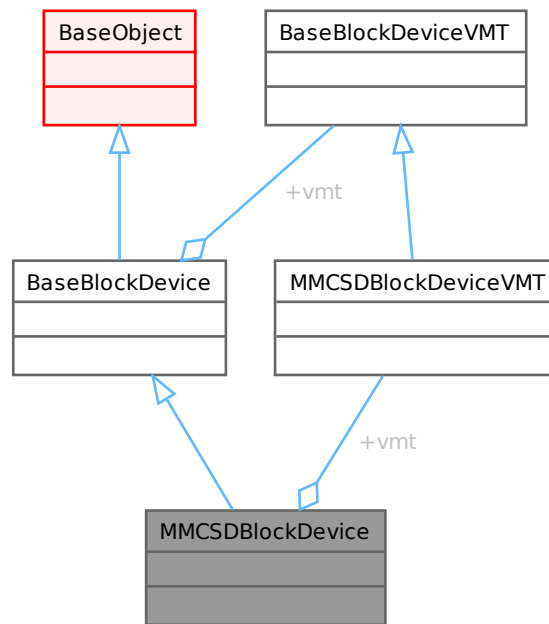
MCC/SD block device class.

```
#include <hal_mmc.h>
```

Inheritance diagram for MMCSDBlockDevice:



Collaboration diagram for MMCSDBlockDevice:



Data Fields

- const struct [MMCSDBlockDeviceVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseBlockDevice](#)

- const struct [BaseBlockDeviceVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.99.1 Detailed Description

MCC/SD block device class.

This class represents a, block-accessible, MMC/SD device.

8.99.2 Field Documentation

8.99.2.1 vmt

```
const struct MMCSDBlockDeviceVMT* MMCSDBlockDevice::vmt
```

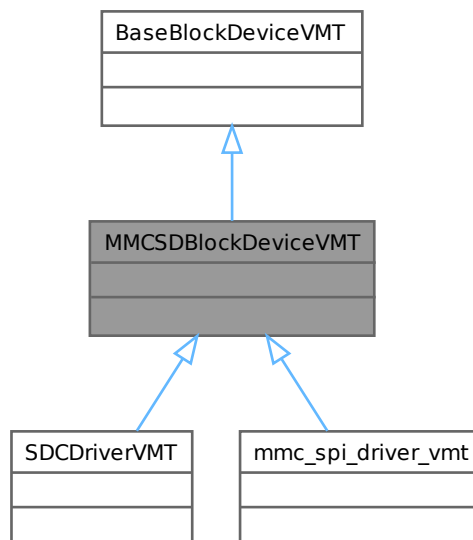
Virtual Methods Table.

8.100 MMCSDBlockDeviceVMT Struct Reference

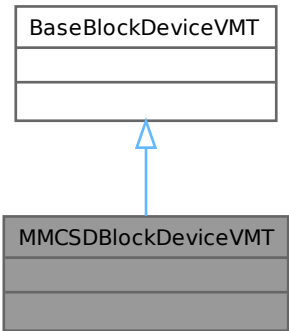
[MMCSDBlockDevice](#) virtual methods table.

```
#include <hal_mmcscd.h>
```

Inheritance diagram for MMCSDBlockDeviceVMT:



Collaboration diagram for MMCSDBlockDeviceVMT:



8.100.1 Detailed Description

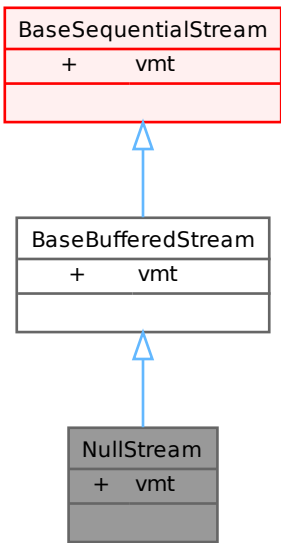
[MMCSDBlockDevice](#) virtual methods table.

8.101 NullStream Struct Reference

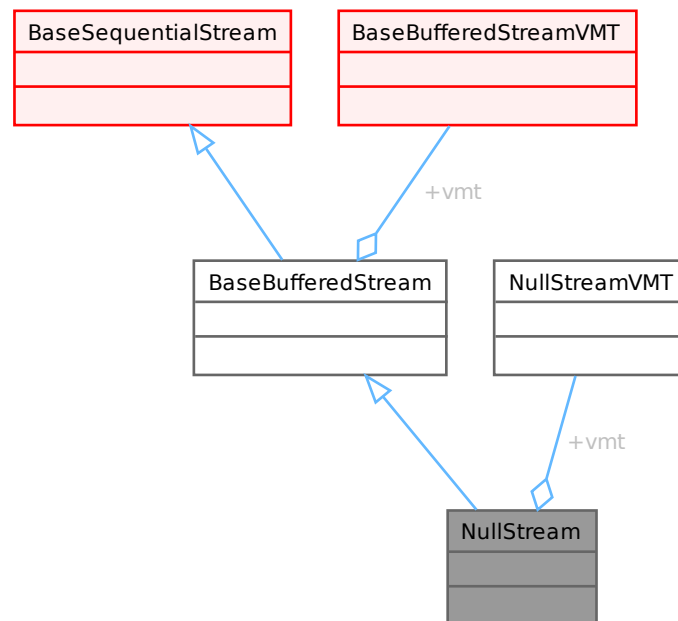
Null stream object.

```
#include <nullstreams.h>
```

Inheritance diagram for NullStream:



Collaboration diagram for `NullStream`:



Data Fields

- const struct `NullStreamVMT` * `vmt`
Virtual Methods Table.

Data Fields inherited from `BaseBufferedStream`

- const struct `BaseBufferedStreamVMT` * `vmt`
Virtual Methods Table.

Data Fields inherited from `BaseSequentialStream`

- const struct `BaseSequentialStreamVMT` * `vmt`
Virtual Methods Table.

Data Fields inherited from `BaseObject`

- const struct `BaseObjectVMT` * `vmt`
Virtual Methods Table.

8.101.1 Detailed Description

Null stream object.

8.101.2 Field Documentation

8.101.2.1 vmt

```
const struct NullStreamVMT* NullStream::vmt
```

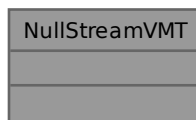
Virtual Methods Table.

8.102 NullStreamVMT Struct Reference

[NullStream](#) virtual methods table.

```
#include <nullstreams.h>
```

Collaboration diagram for NullStreamVMT:



8.102.1 Detailed Description

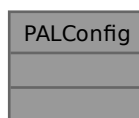
[NullStream](#) virtual methods table.

8.103 PALConfig Struct Reference

Generic I/O ports static initializer.

```
#include <hal_pal_lld.h>
```

Collaboration diagram for PALConfig:



8.103.1 Detailed Description

Generic I/O ports static initializer.

An instance of this structure must be passed to `palInit()` at system startup time in order to initialize the digital I/O subsystem. This represents only the initial setup, specific pads or whole ports can be reprogrammed at later time.

Note

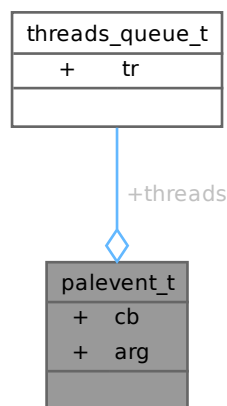
Implementations may extend this structure to contain more, architecture dependent, fields.

8.104 palevent_t Struct Reference

Type of a PAL event record.

```
#include <hal_pal.h>
```

Collaboration diagram for `palevent_t`:



Data Fields

- `threads_queue_t threads`
Threads queued for an event.
- `palcallback_t cb`
Event callback.
- `void * arg`
Event callback argument.

8.104.1 Detailed Description

Type of a PAL event record.

8.104.2 Field Documentation

8.104.2.1 threads

`threads_queue_t` `palevent_t::threads`

Threads queued for an event.

8.104.2.2 cb

`palcallback_t` `palevent_t::cb`

Event callback.

8.104.2.3 arg

`void*` `palevent_t::arg`

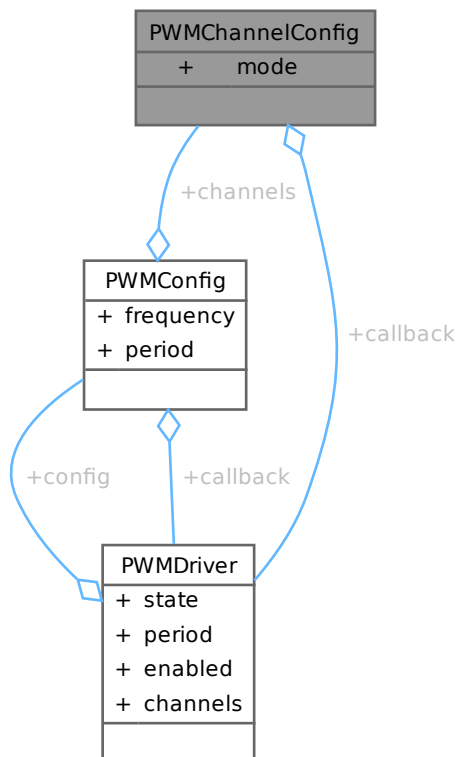
Event callback argument.

8.105 PWMChannelConfig Struct Reference

Type of a PWM driver channel configuration structure.

```
#include <hal_pwm_lld.h>
```

Collaboration diagram for PWMChannelConfig:



Data Fields

- [pwmmode_t mode](#)
Channel active logic level.
- [pwmcallback_t callback](#)
Channel callback pointer.

8.105.1 Detailed Description

Type of a PWM driver channel configuration structure.

8.105.2 Field Documentation

8.105.2.1 mode

[pwmmode_t](#) PWMChannelConfig::mode

Channel active logic level.

8.105.2.2 callback

[pwmcallback_t](#) PWMChannelConfig::callback

Channel callback pointer.

Note

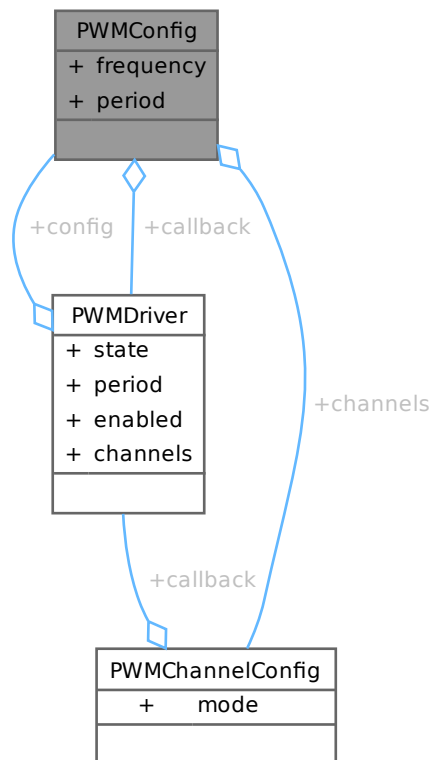
This callback is invoked on the channel compare event. If set to `NULL` then the callback is disabled.

8.106 PWMConfig Struct Reference

Type of a PWM driver configuration structure.

```
#include <hal_pwm_lld.h>
```

Collaboration diagram for PWMConfig:



Data Fields

- `uint32_t frequency`
Timer clock in Hz.
- `pwmcnt_t period`
PWM period in ticks.
- `pwmcallback_t callback`
Periodic callback pointer.
- `PWMChannelConfig channels` [`PWM_CHANNELS`]
Channels configurations.

8.106.1 Detailed Description

Type of a PWM driver configuration structure.

8.106.2 Field Documentation

8.106.2.1 frequency

`uint32_t PWMConfig::frequency`

Timer clock in Hz.

Note

The low level can use assertions in order to catch invalid frequency specifications.

8.106.2.2 period

`pwmcnt_t PWMConfig::period`

PWM period in ticks.

Note

The low level can use assertions in order to catch invalid period specifications.

8.106.2.3 callback

`pwmcallback_t PWMConfig::callback`

Periodic callback pointer.

Note

This callback is invoked on PWM counter reset. If set to `NULL` then the callback is disabled.

8.106.2.4 channels

`PWMChannelConfig PWMConfig::channels[PWM_CHANNELS]`

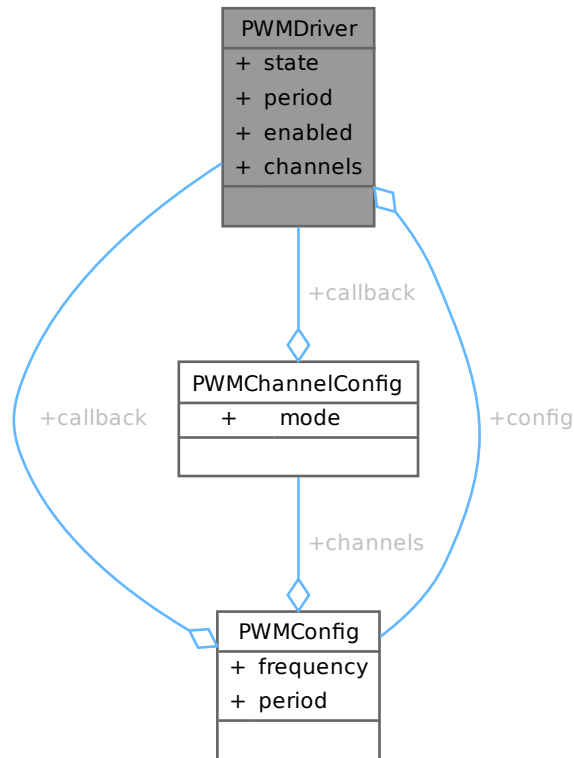
Channels configurations.

8.107 PWMDriver Struct Reference

Structure representing a PWM driver.

```
#include <hal_pwm_llid.h>
```

Collaboration diagram for PWMDriver:



Data Fields

- [pwmstate_t state](#)
Driver state.
- `const PWMConfig * config`
Current driver configuration data.
- [pwmcnt_t period](#)
Current PWM period in ticks.
- [pwmchnmsk_t enabled](#)
Mask of the enabled channels.
- [pwmchannel_t channels](#)
Number of channels in this instance.

8.107.1 Detailed Description

Structure representing a PWM driver.

8.107.2 Field Documentation

8.107.2.1 state

`pwmstate_t` PWMDriver::state

Driver state.

8.107.2.2 config

`const PWMConfig*` PWMDriver::config

Current driver configuration data.

8.107.2.3 period

`pwmcnt_t` PWMDriver::period

Current PWM period in ticks.

8.107.2.4 enabled

`pwmchnmsk_t` PWMDriver::enabled

Mask of the enabled channels.

8.107.2.5 channels

`pwmchannel_t` PWMDriver::channels

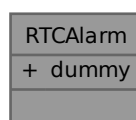
Number of channels in this instance.

8.108 RTCArm Struct Reference

Type of a structure representing an RTC alarm time stamp.

```
#include <hal_rtc_lld.h>
```

Collaboration diagram for RTCArm:



Data Fields

- uint32_t [dummy](#)

8.108.1 Detailed Description

Type of a structure representing an RTC alarm time stamp.

8.108.2 Field Documentation

8.108.2.1 dummy

```
uint32_t RTCArm::dummy
```

8.109 RTCDatetime Struct Reference

Type of a structure representing an RTC date/time stamp.

```
#include <hal_rtc.h>
```

Collaboration diagram for RTCDatetime:

RTCDatetime
+ year
+ month
+ dstflag
+ dayofweek
+ day
+ millisecond

Data Fields

- uint32_t [year](#): 8
Years since 1980.
- uint32_t [month](#): 4
Months 1..12.
- uint32_t [dstflag](#): 1
DST correction flag.
- uint32_t [dayofweek](#): 3
Day of week 1..7.
- uint32_t [day](#): 5
Day of the month 1..31.
- uint32_t [millisecond](#): 27
Milliseconds since midnight.

8.109.1 Detailed Description

Type of a structure representing an RTC date/time stamp.

8.109.2 Field Documentation

8.109.2.1 year

```
uint32_t RTCDatetime::year
```

Years since 1980.

8.109.2.2 month

```
uint32_t RTCDatetime::month
```

Months 1..12.

8.109.2.3 dstflag

```
uint32_t RTCDatetime::dstflag
```

DST correction flag.

8.109.2.4 dayofweek

```
uint32_t RTCDatetime::dayofweek
```

Day of week 1..7.

8.109.2.5 day

```
uint32_t RTCDatetime::day
```

Day of the month 1..31.

8.109.2.6 millisecond

```
uint32_t RTCDatetime::millisecond
```

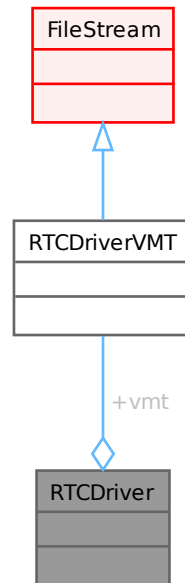
Milliseconds since midnight.

8.110 RTCDriver Struct Reference

Structure representing an RTC driver.

```
#include <hal_rtc.h>
```

Collaboration diagram for RTCDriver:



Data Fields

- const struct [RTCDriverVMT](#) * `vmt`
Virtual Methods Table.

8.110.1 Detailed Description

Structure representing an RTC driver.

8.110.2 Field Documentation

8.110.2.1 vmt

```
const struct RTCDriverVMT* RTCDriver::vmt
```

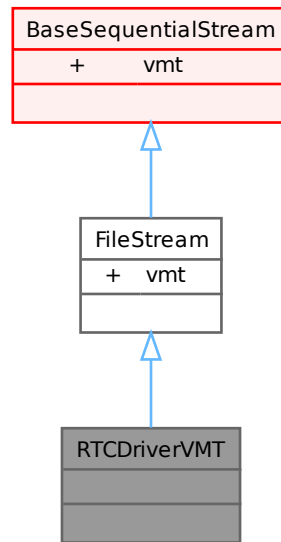
Virtual Methods Table.

8.111 RTCDriverVMT Struct Reference

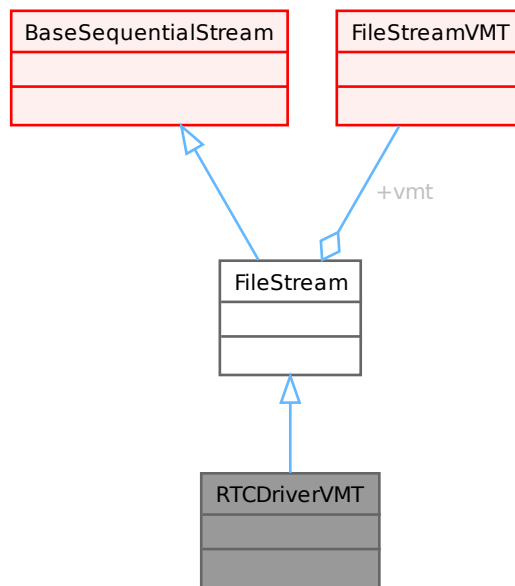
[RTCDriver](#) virtual methods table.

```
#include <hal_rtc.h>
```

Inheritance diagram for RTCDriverVMT:



Collaboration diagram for RTCDriverVMT:



Additional Inherited Members

Data Fields inherited from [FileStream](#)

- const struct [FileStreamVMT](#) * [vmt](#)
Virtual Methods Table.

Data Fields inherited from [BaseSequentialStream](#)

- const struct [BaseSequentialStreamVMT](#) * [vmt](#)
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * [vmt](#)
Virtual Methods Table.

8.111.1 Detailed Description

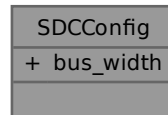
[RTCDriver](#) virtual methods table.

8.112 SDCConfig Struct Reference

Driver configuration structure.

```
#include <hal_sdc_lld.h>
```

Collaboration diagram for SDCConfig:



Data Fields

- [sdcbusmode_t bus_width](#)

Bus width.

8.112.1 Detailed Description

Driver configuration structure.

Note

It could be empty on some architectures.

8.112.2 Field Documentation

8.112.2.1 bus_width

[sdcbusmode_t](#) SDCConfig::bus_width

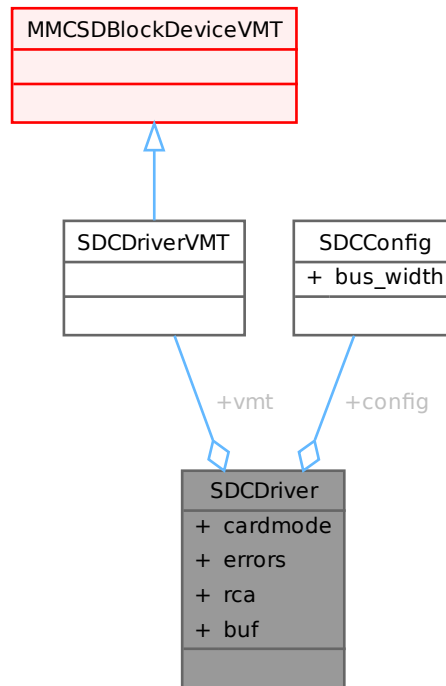
Bus width.

8.113 SDCDriver Struct Reference

Structure representing an SDC driver.

```
#include <hal_sdc_lld.h>
```

Collaboration diagram for SDCDriver:



Data Fields

- const struct [SDCDriverVMT](#) * [vmt](#)
Virtual Methods Table.
- [_mmc_sdb_block_device_data](#) const [SDCCConfig](#) * [config](#)
Current configuration data.
- [sdcmode_t](#) [cardmode](#)
Various flags regarding the mounted card.
- [sdclags_t](#) [errors](#)
Errors flags.
- [uint32_t](#) [rca](#)
Card RCA.
- [uint8_t](#) [buf](#) [[MMCSD_BLOCK_SIZE](#)]
Buffer for internal operations.

8.113.1 Detailed Description

Structure representing an SDC driver.

8.113.2 Field Documentation

8.113.2.1 vmt

```
const struct SDCDriverVMT* SDCDriver::vmt
```

Virtual Methods Table.

8.113.2.2 config

```
\_mmcsd\_block\_device\_data const SDCConfig* SDCDriver::config
```

Current configuration data.

8.113.2.3 cardmode

```
sdcmode\_t SDCDriver::cardmode
```

Various flags regarding the mounted card.

8.113.2.4 errors

```
sdcflags\_t SDCDriver::errors
```

Errors flags.

8.113.2.5 rca

```
uint32_t SDCDriver::rca
```

Card RCA.

8.113.2.6 buf

```
uint8_t SDCDriver::buf[MMCSD\_BLOCK\_SIZE]
```

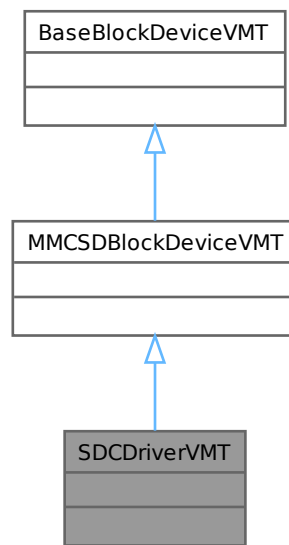
Buffer for internal operations.

8.114 SDCCDriverVMT Struct Reference

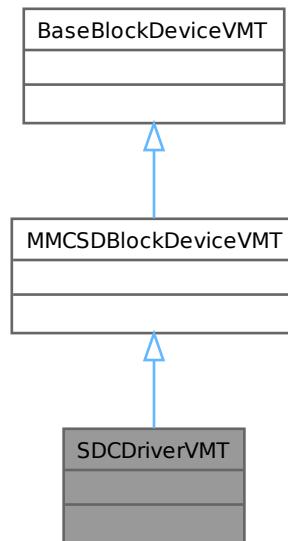
[SDCCDriver](#) virtual methods table.

```
#include <hal_sdc_lld.h>
```

Inheritance diagram for SDCCDriverVMT:



Collaboration diagram for SDCDriverVMT:



8.114.1 Detailed Description

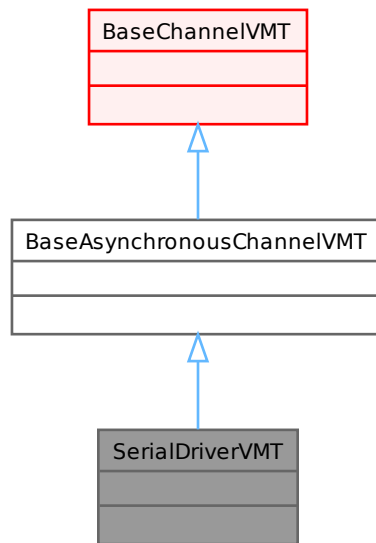
[SDCDriver](#) virtual methods table.

8.115 SerialDriverVMT Struct Reference

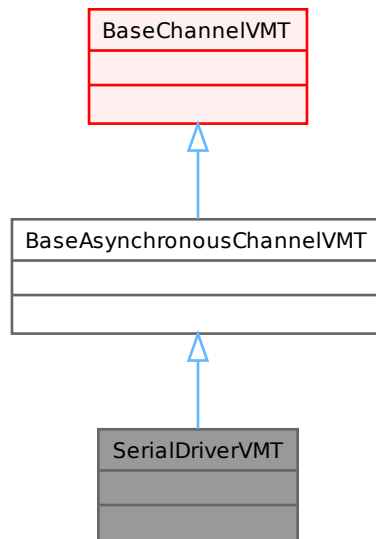
[SerialDriver](#) virtual methods table.

```
#include <hal_serial.h>
```

Inheritance diagram for SerialDriverVMT:



Collaboration diagram for SerialDriverVMT:



8.115.1 Detailed Description

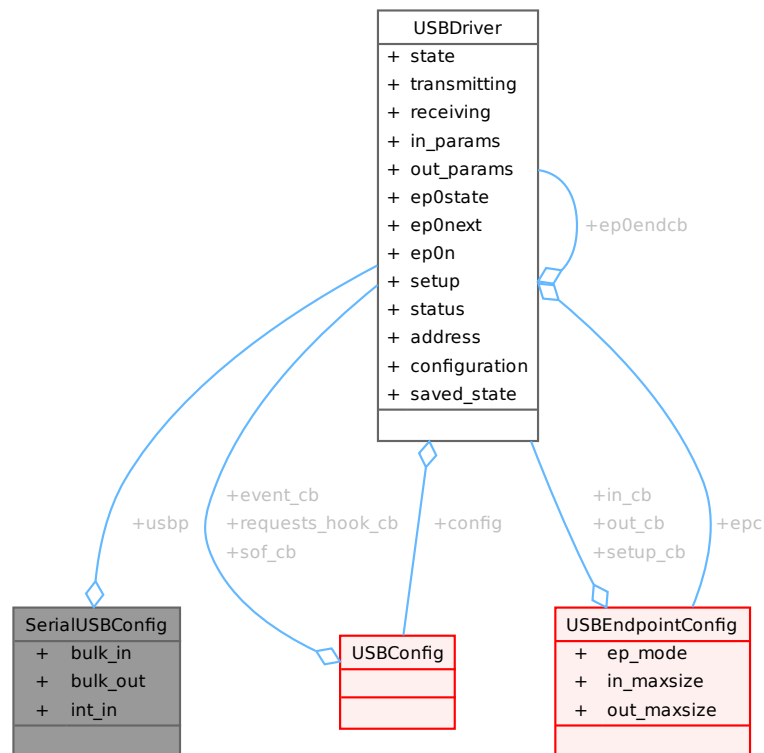
[SerialDriver](#) virtual methods table.

8.116 SerialUSBConfig Struct Reference

Serial over USB Driver configuration structure.

```
#include <hal_serial_usb.h>
```

Collaboration diagram for SerialUSBConfig:



Data Fields

- [USBDriver](#) * `usbp`
USB driver to use.
- [usbep_t](#) `bulk_in`
Bulk IN endpoint used for outgoing data transfer.
- [usbep_t](#) `bulk_out`
Bulk OUT endpoint used for incoming data transfer.
- [usbep_t](#) `int_in`
Interrupt IN endpoint used for notifications.

8.116.1 Detailed Description

Serial over USB Driver configuration structure.

An instance of this structure must be passed to [sduStart\(\)](#) in order to configure and start the driver operations.

8.116.2 Field Documentation

8.116.2.1 usbp

`USBDriver* SerialUSBConfig::usbp`

USB driver to use.

8.116.2.2 bulk_in

`usbep_t SerialUSBConfig::bulk_in`

Bulk IN endpoint used for outgoing data transfer.

8.116.2.3 bulk_out

`usbep_t SerialUSBConfig::bulk_out`

Bulk OUT endpoint used for incoming data transfer.

8.116.2.4 int_in

`usbep_t SerialUSBConfig::int_in`

Interrupt IN endpoint used for notifications.

Note

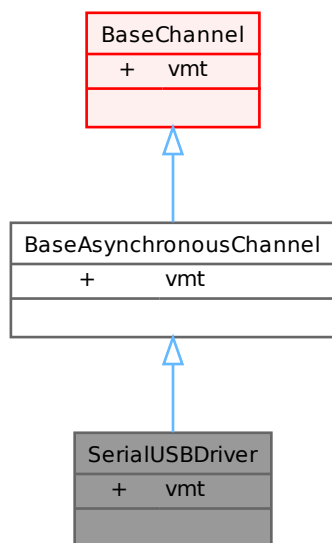
If set to zero then the INT endpoint is assumed to be not present, USB descriptors must be changed accordingly.

8.117 SerialUSBDriver Struct Reference

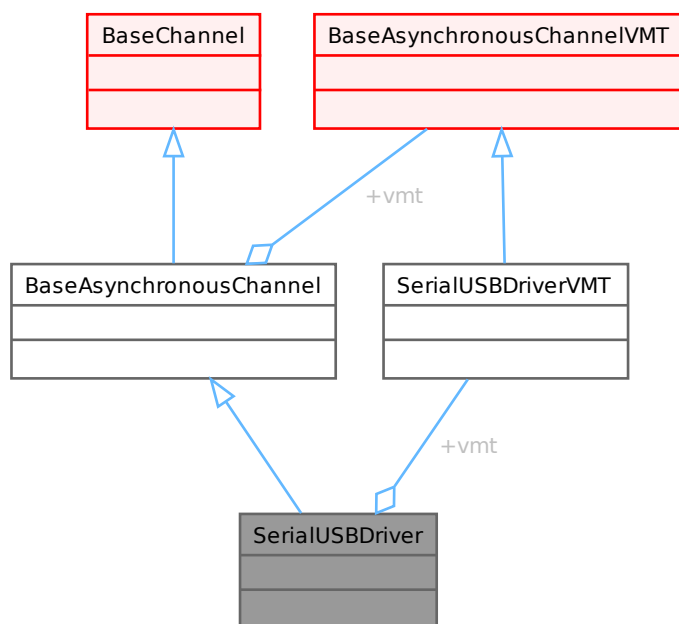
Full duplex serial driver class.

```
#include <hal_serial_usb.h>
```

Inheritance diagram for SerialUSBDriver:



Collaboration diagram for SerialUSBDriver:



Data Fields

- const struct [SerialUSBDriverVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseAsynchronousChannel](#)

- const struct [BaseAsynchronousChannelVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseChannel](#)

- const struct [BaseChannelVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseSequentialStream](#)

- const struct [BaseSequentialStreamVMT](#) * vmt
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * vmt
Virtual Methods Table.

8.117.1 Detailed Description

Full duplex serial driver class.

This class extends [BaseAsynchronousChannel](#) by adding physical I/O queues.

8.117.2 Field Documentation

8.117.2.1 vmt

```
const struct SerialUSBDriverVMT* SerialUSBDriver::vmt
```

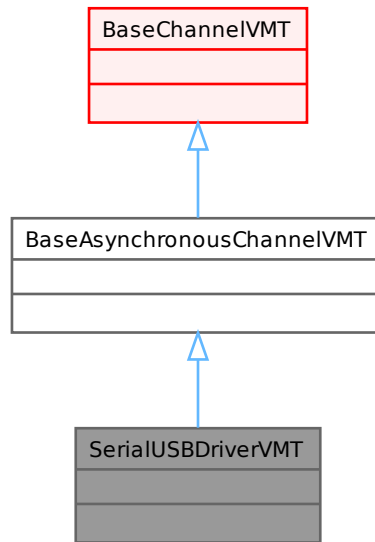
Virtual Methods Table.

8.118 SerialUSBDriverVMT Struct Reference

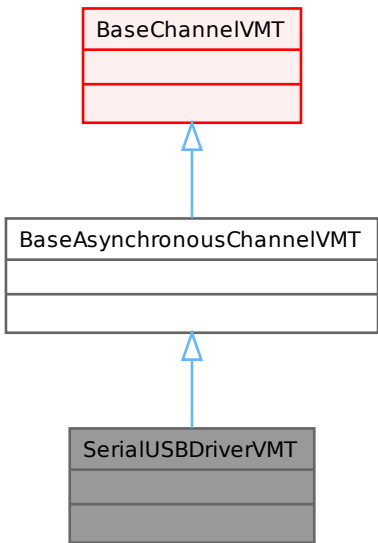
[SerialDriver](#) virtual methods table.

```
#include <hal_serial_usb.h>
```

Inheritance diagram for SerialUSBDriverVMT:



Collaboration diagram for SerialUSBDriverVMT:



8.118.1 Detailed Description

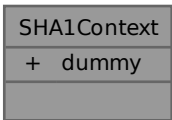
SerialDriver virtual methods table.

8.119 SHA1Context Struct Reference

Type of a SHA1 context.

```
#include <hal_crypto_llid.h>
```

Collaboration diagram for SHA1Context:



Data Fields

- uint32_t dummy

8.119.1 Detailed Description

Type of a SHA1 context.

8.119.2 Field Documentation

8.119.2.1 dummy

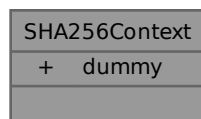
```
uint32_t SHA1Context::dummy
```

8.120 SHA256Context Struct Reference

Type of a SHA256 context.

```
#include <hal_crypto_lld.h>
```

Collaboration diagram for SHA256Context:



Data Fields

- `uint32_t` [dummy](#)

8.120.1 Detailed Description

Type of a SHA256 context.

8.120.2 Field Documentation

8.120.2.1 dummy

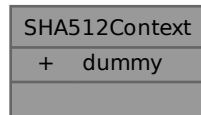
```
uint32_t SHA256Context::dummy
```

8.121 SHA512Context Struct Reference

Type of a SHA512 context.

```
#include <hal_crypto_ll.h>
```

Collaboration diagram for SHA512Context:



Data Fields

- `uint32_t dummy`

8.121.1 Detailed Description

Type of a SHA512 context.

8.121.2 Field Documentation

8.121.2.1 dummy

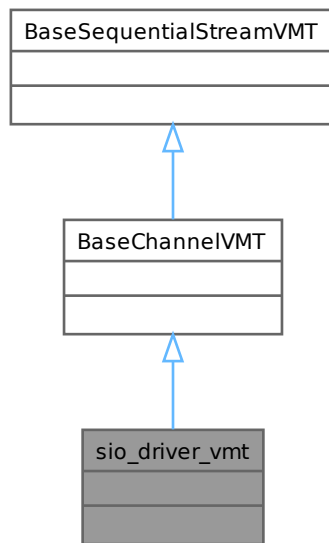
```
uint32_t SHA512Context::dummy
```

8.122 sio_driver_vmt Struct Reference

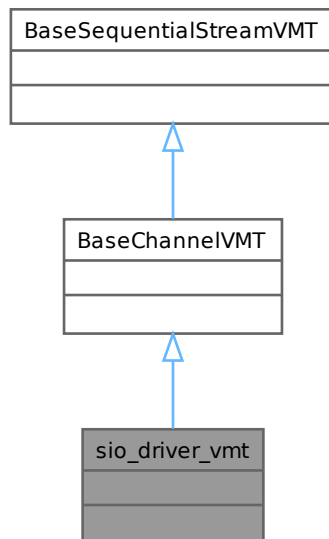
[SIODriver](#) virtual methods table.

```
#include <hal_sio.h>
```

Inheritance diagram for `sio_driver_vmt`:



Collaboration diagram for `sio_driver_vmt`:



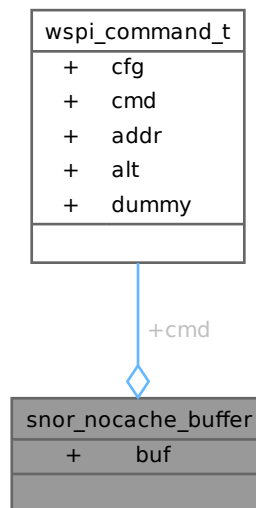
8.122.1 Detailed Description

`SIODriver` virtual methods table.

8.123 snor_nocache_buffer Struct Reference

```
#include <hal_serial_nor.h>
```

Collaboration diagram for snor_nocache_buffer:



Data Fields

- `uint8_t buf [SNOR_BUFFER_SIZE]`
Temporary generic buffer.
- `wspi_command_t cmd`
Temporary command buffer.

8.123.1 Field Documentation

8.123.1.1 buf

```
uint8_t snor_nocache_buffer::buf[SNOR_BUFFER_SIZE]
```

Temporary generic buffer.

8.123.1.2 cmd

```
wspi_command_t snor_nocache_buffer::cmd
```

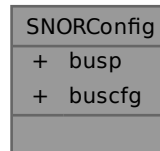
Temporary command buffer.

8.124 SNORConfig Struct Reference

Type of a SNOR configuration structure.

```
#include <hal_serial_nor.h>
```

Collaboration diagram for SNORConfig:



Data Fields

- [BUSDriver](#) * [busp](#)
- const [BUSConfig](#) * [buscfg](#)

8.124.1 Detailed Description

Type of a SNOR configuration structure.

8.124.2 Field Documentation

8.124.2.1 busp

```
BUSDriver* SNORConfig::busp
```

8.124.2.2 buscfg

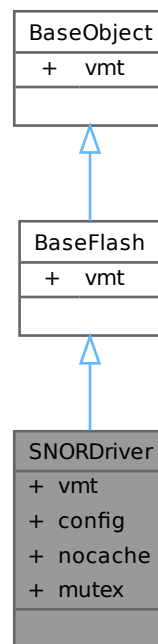
```
const BUSConfig* SNORConfig::buscfg
```

8.125 SNORDriver Struct Reference

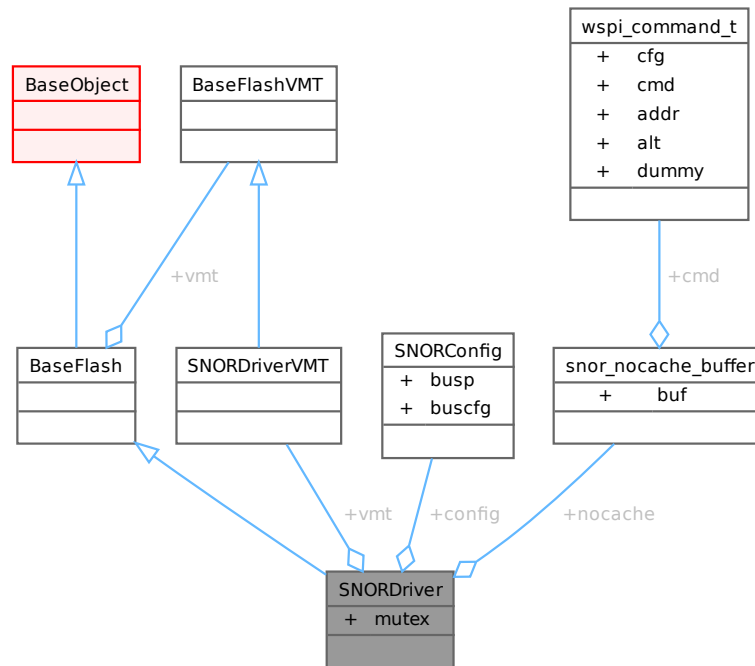
Type of SNOR flash class.

```
#include <hal_serial_nor.h>
```

Inheritance diagram for SNORDriver:



Collaboration diagram for SNORDriver:



Data Fields

- const struct [SNORDriverVMT](#) * `vmt`
SNORDriver Virtual Methods Table.
- `_base_flash_data` const [SNORConfig](#) * `config`
Current configuration data.
- [snor_nocache_buffer_t](#) * `nocache`
Non-cacheable buffer associated to this instance.
- [mutex_t](#) `mutex`
Mutex protecting SNOR.

Data Fields inherited from [BaseFlash](#)

- const struct [BaseFlashVMT](#) * `vmt`
Virtual Methods Table.

Data Fields inherited from [BaseObject](#)

- const struct [BaseObjectVMT](#) * `vmt`
Virtual Methods Table.

8.125.1 Detailed Description

Type of SNOR flash class.

8.125.2 Field Documentation

8.125.2.1 vmt

```
const struct SNORDriverVMT* SNORDriver::vmt
```

[SNORDriver](#) Virtual Methods Table.

8.125.2.2 config

```
_base_flash_data const SNORConfig* SNORDriver::config
```

Current configuration data.

8.125.2.3 nocache

```
snor_nocache_buffer_t* SNORDriver::nocache
```

Non-cacheable buffer associated to this instance.

8.125.2.4 mutex

```
mutex_t SNORDriver::mutex
```

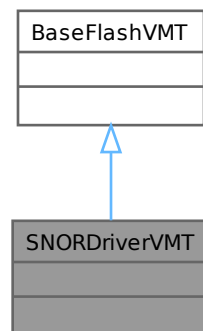
Mutex protecting SNOR.

8.126 SNORDriverVMT Struct Reference

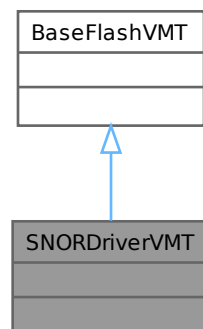
SNOR virtual methods table.

```
#include <hal_serial_nor.h>
```

Inheritance diagram for SNORDriverVMT:



Collaboration diagram for SNORDriverVMT:



8.126.1 Detailed Description

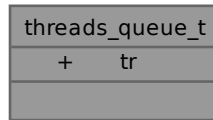
SNOR virtual methods table.

8.127 threads_queue_t Struct Reference

Type of a thread queue.

```
#include <osal.h>
```

Collaboration diagram for threads_queue_t:



Data Fields

- [thread_reference_t](#) tr

8.127.1 Detailed Description

Type of a thread queue.

A thread queue is a queue of sleeping threads, queued threads can be dequeued one at time or all together.

Note

If the OSAL is implemented on a bare metal machine without RTOS then the queue can be implemented as a single thread reference.

8.127.2 Field Documentation

8.127.2.1 tr

```
thread\_reference\_t threads_queue_t::tr
```

8.128 unpacked_mmc_cid_t Struct Reference

Unpacked CID register from MMC.

```
#include <hal_mmc_ssd.h>
```

Collaboration diagram for `unpacked_mmc_cid_t`:

unpacked_mmc_cid_t	
+	mid
+	oid
+	pnm
+	prv_n
+	prv_m
+	psn
+	mdt_m
+	mdt_y
+	crc

Data Fields

- `uint8_t` [mid](#)
- `uint16_t` [oid](#)
- `char` [pnm](#) [6]
- `uint8_t` [prv_n](#)
- `uint8_t` [prv_m](#)
- `uint32_t` [psn](#)
- `uint8_t` [mdt_m](#)
- `uint16_t` [mdt_y](#)
- `uint8_t` [crc](#)

8.128.1 Detailed Description

Unpacked CID register from MMC.

8.128.2 Field Documentation

8.128.2.1 mid

```
uint8_t unpacked_mmc_cid_t::mid
```

8.128.2.2 oid

```
uint16_t unpacked_mmc_cid_t::oid
```

8.128.2.3 pnm

```
char unpacked_mmc_cid_t::pnm[6]
```

8.128.2.4 prv_n

```
uint8_t unpacked_mmc_cid_t::prv_n
```

8.128.2.5 prv_m

```
uint8_t unpacked_mmc_cid_t::prv_m
```

8.128.2.6 psn

```
uint32_t unpacked_mmc_cid_t::psn
```

8.128.2.7 mdt_m

```
uint8_t unpacked_mmc_cid_t::mdt_m
```

8.128.2.8 mdt_y

```
uint16_t unpacked_mmc_cid_t::mdt_y
```

8.128.2.9 crc

```
uint8_t unpacked_mmc_cid_t::crc
```

8.129 unpacked_mmc_csd_t Struct Reference

Unpacked CSD register from MMC.

```
#include <hal_mmcsd.h>
```

Collaboration diagram for `unpacked_mmc_csd_t`:

unpacked_mmc_csd_t
+ csd_structure
+ spec_vers
+ taac
+ nsac
+ tran_speed
+ ccc
+ read_bl_len
+ read_bl_partial
+ write_blk_misalign
+ read_blk_misalign
and 23 more...

Data Fields

- `uint8_t` [csd_structure](#)
- `uint8_t` [spec_vers](#)
- `uint8_t` [taac](#)
- `uint8_t` [nsac](#)
- `uint8_t` [tran_speed](#)
- `uint16_t` [ccc](#)
- `uint8_t` [read_bl_len](#)
- `uint8_t` [read_bl_partial](#)
- `uint8_t` [write_blk_misalign](#)
- `uint8_t` [read_blk_misalign](#)
- `uint8_t` [dsr_imp](#)
- `uint16_t` [c_size](#)
- `uint8_t` [vdd_r_curr_min](#)
- `uint8_t` [vdd_r_curr_max](#)
- `uint8_t` [vdd_w_curr_min](#)
- `uint8_t` [vdd_w_curr_max](#)
- `uint8_t` [c_size_mult](#)
- `uint8_t` [erase_grp_size](#)
- `uint8_t` [erase_grp_mult](#)
- `uint8_t` [wp_grp_size](#)
- `uint8_t` [wp_grp_enable](#)
- `uint8_t` [default_ecc](#)
- `uint8_t` [r2w_factor](#)
- `uint8_t` [write_bl_len](#)
- `uint8_t` [write_bl_partial](#)
- `uint8_t` [content_prot_app](#)
- `uint8_t` [file_format_grp](#)
- `uint8_t` [copy](#)

- uint8_t [perm_write_protect](#)
- uint8_t [tmp_write_protect](#)
- uint8_t [file_format](#)
- uint8_t [ecc](#)
- uint8_t [crc](#)

8.129.1 Detailed Description

Unpacked CSD register from MMC.

8.129.2 Field Documentation

8.129.2.1 csd_structure

```
uint8_t unpacked_mmc_csd_t::csd_structure
```

8.129.2.2 spec_vers

```
uint8_t unpacked_mmc_csd_t::spec_vers
```

8.129.2.3 taac

```
uint8_t unpacked_mmc_csd_t::taac
```

8.129.2.4 nsac

```
uint8_t unpacked_mmc_csd_t::nsac
```

8.129.2.5 tran_speed

```
uint8_t unpacked_mmc_csd_t::tran_speed
```

8.129.2.6 ccc

```
uint16_t unpacked_mmc_csd_t::ccc
```

8.129.2.7 read_bl_len

```
uint8_t unpacked_mmc_csd_t::read_bl_len
```

8.129.2.8 read_bl_partial

```
uint8_t unpacked_mmc_csd_t::read_bl_partial
```

8.129.2.9 write_blk_misalign

```
uint8_t unpacked_mmc_csd_t::write_blk_misalign
```

8.129.2.10 read_blk_misalign

```
uint8_t unpacked_mmc_csd_t::read_blk_misalign
```

8.129.2.11 dsr_imp

```
uint8_t unpacked_mmc_csd_t::dsr_imp
```

8.129.2.12 c_size

```
uint16_t unpacked_mmc_csd_t::c_size
```

8.129.2.13 vdd_r_curr_min

```
uint8_t unpacked_mmc_csd_t::vdd_r_curr_min
```

8.129.2.14 vdd_r_curr_max

```
uint8_t unpacked_mmc_csd_t::vdd_r_curr_max
```

8.129.2.15 vdd_w_curr_min

```
uint8_t unpacked_mmc_csd_t::vdd_w_curr_min
```

8.129.2.16 vdd_w_curr_max

```
uint8_t unpacked_mmc_csd_t::vdd_w_curr_max
```

8.129.2.17 c_size_mult

```
uint8_t unpacked_mmc_csd_t::c_size_mult
```


8.129.2.18 erase_grp_size

```
uint8_t unpacked_mmc_csd_t::erase_grp_size
```

8.129.2.19 erase_grp_mult

```
uint8_t unpacked_mmc_csd_t::erase_grp_mult
```

8.129.2.20 wp_grp_size

```
uint8_t unpacked_mmc_csd_t::wp_grp_size
```

8.129.2.21 wp_grp_enable

```
uint8_t unpacked_mmc_csd_t::wp_grp_enable
```

8.129.2.22 default_ecc

```
uint8_t unpacked_mmc_csd_t::default_ecc
```

8.129.2.23 r2w_factor

```
uint8_t unpacked_mmc_csd_t::r2w_factor
```

8.129.2.24 write_bl_len

```
uint8_t unpacked_mmc_csd_t::write_bl_len
```

8.129.2.25 write_bl_partial

```
uint8_t unpacked_mmc_csd_t::write_bl_partial
```

8.129.2.26 content_prot_app

```
uint8_t unpacked_mmc_csd_t::content_prot_app
```

8.129.2.27 file_format_grp

```
uint8_t unpacked_mmc_csd_t::file_format_grp
```

8.129.2.28 copy

```
uint8_t unpacked_mmc_csd_t::copy
```

8.129.2.29 perm_write_protect

```
uint8_t unpacked_mmc_csd_t::perm_write_protect
```

8.129.2.30 tmp_write_protect

```
uint8_t unpacked_mmc_csd_t::tmp_write_protect
```

8.129.2.31 file_format

```
uint8_t unpacked_mmc_csd_t::file_format
```

8.129.2.32 ecc

```
uint8_t unpacked_mmc_csd_t::ecc
```

8.129.2.33 crc

```
uint8_t unpacked_mmc_csd_t::crc
```

8.130 unpacked_sdc_cid_t Struct Reference

Unpacked CID register from SDC.

```
#include <hal_mmc_sdc.h>
```

Collaboration diagram for unpacked_sdc_cid_t:

unpacked_sdc_cid_t	
+	mid
+	oid
+	pnm
+	prv_n
+	prv_m
+	psn
+	mdt_m
+	mdt_y
+	crc

Data Fields

- uint8_t [mid](#)
- uint16_t [oid](#)
- char [pnm](#) [5]
- uint8_t [prv_n](#)
- uint8_t [prv_m](#)
- uint32_t [psn](#)
- uint8_t [mdt_m](#)
- uint16_t [mdt_y](#)
- uint8_t [crc](#)

8.130.1 Detailed Description

Unpacked CID register from SDC.

8.130.2 Field Documentation

8.130.2.1 mid

```
uint8_t unpacked_sdc_cid_t::mid
```

8.130.2.2 oid

```
uint16_t unpacked_sdc_cid_t::oid
```

8.130.2.3 pnm

```
char unpacked_sdc_cid_t::pnm[5]
```

8.130.2.4 prv_n

```
uint8_t unpacked_sdc_cid_t::prv_n
```

8.130.2.5 prv_m

```
uint8_t unpacked_sdc_cid_t::prv_m
```

8.130.2.6 psn

```
uint32_t unpacked_sdc_cid_t::psn
```

8.130.2.7 mdt_m

```
uint8_t unpacked_sdc_cid_t::mdt_m
```

8.130.2.8 mdt_y

```
uint16_t unpacked_sdc_cid_t::mdt_y
```

8.130.2.9 crc

```
uint8_t unpacked_sdc_cid_t::crc
```

8.131 unpacked_sdc_csd_10_t Struct Reference

Unpacked CSD v1.0 register from SDC.

```
#include <hal_mmcscsd.h>
```

Collaboration diagram for `unpacked_sdc_csd_10_t`:

unpacked_sdc_csd_10_t
+ csd_structure
+ taac
+ nsac
+ tran_speed
+ ccc
+ read_bl_len
+ read_bl_partial
+ write_blk_misalign
+ read_blk_misalign
+ dsr_imp
and 19 more...

Data Fields

- uint8_t [csd_structure](#)
- uint8_t [taac](#)
- uint8_t [nsac](#)
- uint8_t [tran_speed](#)
- uint16_t [ccc](#)
- uint8_t [read_bl_len](#)
- uint8_t [read_bl_partial](#)
- uint8_t [write_blk_misalign](#)
- uint8_t [read_blk_misalign](#)
- uint8_t [dsr_imp](#)
- uint16_t [c_size](#)
- uint8_t [vdd_r_curr_min](#)
- uint8_t [vdd_r_curr_max](#)
- uint8_t [vdd_w_curr_min](#)
- uint8_t [vdd_w_curr_max](#)
- uint8_t [c_size_mult](#)
- uint8_t [erase_blk_en](#)
- uint8_t [erase_sector_size](#)
- uint8_t [wp_grp_size](#)
- uint8_t [wp_grp_enable](#)
- uint8_t [r2w_factor](#)
- uint8_t [write_bl_len](#)
- uint8_t [write_bl_partial](#)
- uint8_t [file_format_grp](#)
- uint8_t [copy](#)
- uint8_t [perm_write_protect](#)
- uint8_t [tmp_write_protect](#)
- uint8_t [file_format](#)
- uint8_t [crc](#)

8.131.1 Detailed Description

Unpacked CSD v1.0 register from SDC.

8.131.2 Field Documentation

8.131.2.1 csd_structure

```
uint8_t unpacked_sdc_csd_10_t::csd_structure
```

8.131.2.2 taac

```
uint8_t unpacked_sdc_csd_10_t::taac
```

8.131.2.3 nsac

```
uint8_t unpacked_sdc_csd_10_t::nsac
```

8.131.2.4 tran_speed

```
uint8_t unpacked_sdc_csd_10_t::tran_speed
```

8.131.2.5 ccc

```
uint16_t unpacked_sdc_csd_10_t::ccc
```

8.131.2.6 read_bl_len

```
uint8_t unpacked_sdc_csd_10_t::read_bl_len
```

8.131.2.7 read_bl_partial

```
uint8_t unpacked_sdc_csd_10_t::read_bl_partial
```

8.131.2.8 write_blk_misalign

```
uint8_t unpacked_sdc_csd_10_t::write_blk_misalign
```

8.131.2.9 read_blk_misalign

```
uint8_t unpacked_sdc_csd_10_t::read_blk_misalign
```

8.131.2.10 dsr_imp

```
uint8_t unpacked_sdc_csd_10_t::dsr_imp
```

8.131.2.11 c_size

```
uint16_t unpacked_sdc_csd_10_t::c_size
```

8.131.2.12 vdd_r_curr_min

```
uint8_t unpacked_sdc_csd_10_t::vdd_r_curr_min
```

8.131.2.13 vdd_r_curr_max

```
uint8_t unpacked_sdc_csd_10_t::vdd_r_curr_max
```

8.131.2.14 vdd_w_curr_min

```
uint8_t unpacked_sdc_csd_10_t::vdd_w_curr_min
```

8.131.2.15 vdd_w_curr_max

```
uint8_t unpacked_sdc_csd_10_t::vdd_w_curr_max
```

8.131.2.16 c_size_mult

```
uint8_t unpacked_sdc_csd_10_t::c_size_mult
```

8.131.2.17 erase_blk_en

```
uint8_t unpacked_sdc_csd_10_t::erase_blk_en
```

8.131.2.18 erase_sector_size

```
uint8_t unpacked_sdc_csd_10_t::erase_sector_size
```

8.131.2.19 wp_grp_size

```
uint8_t unpacked_sdc_csd_10_t::wp_grp_size
```

8.131.2.20 wp_grp_enable

```
uint8_t unpacked_sdc_csd_10_t::wp_grp_enable
```

8.131.2.21 r2w_factor

```
uint8_t unpacked_sdc_csd_10_t::r2w_factor
```

8.131.2.22 write_bl_len

```
uint8_t unpacked_sdc_csd_10_t::write_bl_len
```

8.131.2.23 write_bl_partial

```
uint8_t unpacked_sdc_csd_10_t::write_bl_partial
```

8.131.2.24 file_format_grp

```
uint8_t unpacked_sdc_csd_10_t::file_format_grp
```

8.131.2.25 copy

```
uint8_t unpacked_sdc_csd_10_t::copy
```

8.131.2.26 perm_write_protect

```
uint8_t unpacked_sdc_csd_10_t::perm_write_protect
```

8.131.2.27 tmp_write_protect

```
uint8_t unpacked_sdc_csd_10_t::tmp_write_protect
```

8.131.2.28 file_format

```
uint8_t unpacked_sdc_csd_10_t::file_format
```

8.131.2.29 crc

```
uint8_t unpacked_sdc_csd_10_t::crc
```

8.132 unpacked_sdc_csd_20_t Struct Reference

Unpacked CSD v2.0 register from SDC.

```
#include <hal_mmcstd.h>
```

Collaboration diagram for unpacked_sdc_csd_20_t:

unpacked_sdc_csd_20_t
+ csd_structure
+ taac
+ nsac
+ tran_speed
+ ccc
+ read_bl_len
+ read_bl_partial
+ write_blk_misalign
+ read_blk_misalign
+ dsr_imp
and 14 more...

Data Fields

- uint8_t [csd_structure](#)
- uint8_t [taac](#)
- uint8_t [nsac](#)
- uint8_t [tran_speed](#)
- uint16_t [ccc](#)
- uint8_t [read_bl_len](#)
- uint8_t [read_bl_partial](#)
- uint8_t [write_blk_misalign](#)
- uint8_t [read_blk_misalign](#)
- uint8_t [dsr_imp](#)
- uint32_t [c_size](#)
- uint8_t [erase_blk_en](#)
- uint8_t [erase_sector_size](#)
- uint8_t [wp_grp_size](#)
- uint8_t [wp_grp_enable](#)
- uint8_t [r2w_factor](#)
- uint8_t [write_bl_len](#)
- uint8_t [write_bl_partial](#)
- uint8_t [file_format_grp](#)
- uint8_t [copy](#)
- uint8_t [perm_write_protect](#)
- uint8_t [tmp_write_protect](#)
- uint8_t [file_format](#)
- uint8_t [crc](#)

8.132.1 Detailed Description

Unpacked CSD v2.0 register from SDC.

8.132.2 Field Documentation

8.132.2.1 csd_structure

```
uint8_t unpacked_sdc_csd_20_t::csd_structure
```

8.132.2.2 taac

```
uint8_t unpacked_sdc_csd_20_t::taac
```

8.132.2.3 nsac

```
uint8_t unpacked_sdc_csd_20_t::nsac
```

8.132.2.4 tran_speed

```
uint8_t unpacked_sdc_csd_20_t::tran_speed
```

8.132.2.5 ccc

```
uint16_t unpacked_sdc_csd_20_t::ccc
```

8.132.2.6 read_bl_len

```
uint8_t unpacked_sdc_csd_20_t::read_bl_len
```

8.132.2.7 read_bl_partial

```
uint8_t unpacked_sdc_csd_20_t::read_bl_partial
```

8.132.2.8 write_blk_misalign

```
uint8_t unpacked_sdc_csd_20_t::write_blk_misalign
```

8.132.2.9 read_blk_misalign

```
uint8_t unpacked_sdc_csd_20_t::read_blk_misalign
```

8.132.2.10 dsr_imp

```
uint8_t unpacked_sdc_csd_20_t::dsr_imp
```

8.132.2.11 c_size

```
uint32_t unpacked_sdc_csd_20_t::c_size
```

8.132.2.12 erase_blk_en

```
uint8_t unpacked_sdc_csd_20_t::erase_blk_en
```

8.132.2.13 erase_sector_size

```
uint8_t unpacked_sdc_csd_20_t::erase_sector_size
```

8.132.2.14 wp_grp_size

```
uint8_t unpacked_sdc_csd_20_t::wp_grp_size
```

8.132.2.15 wp_grp_enable

```
uint8_t unpacked_sdc_csd_20_t::wp_grp_enable
```

8.132.2.16 r2w_factor

```
uint8_t unpacked_sdc_csd_20_t::r2w_factor
```

8.132.2.17 write_bl_len

```
uint8_t unpacked_sdc_csd_20_t::write_bl_len
```

8.132.2.18 write_bl_partial

```
uint8_t unpacked_sdc_csd_20_t::write_bl_partial
```

8.132.2.19 file_format_grp

```
uint8_t unpacked_sdc_csd_20_t::file_format_grp
```

8.132.2.20 copy

```
uint8_t unpacked_sdc_csd_20_t::copy
```

8.132.2.21 perm_write_protect

```
uint8_t unpacked_sdc_csd_20_t::perm_write_protect
```

8.132.2.22 tmp_write_protect

```
uint8_t unpacked_sdc_csd_20_t::tmp_write_protect
```

8.132.2.23 file_format

```
uint8_t unpacked_sdc_csd_20_t::file_format
```

8.132.2.24 crc

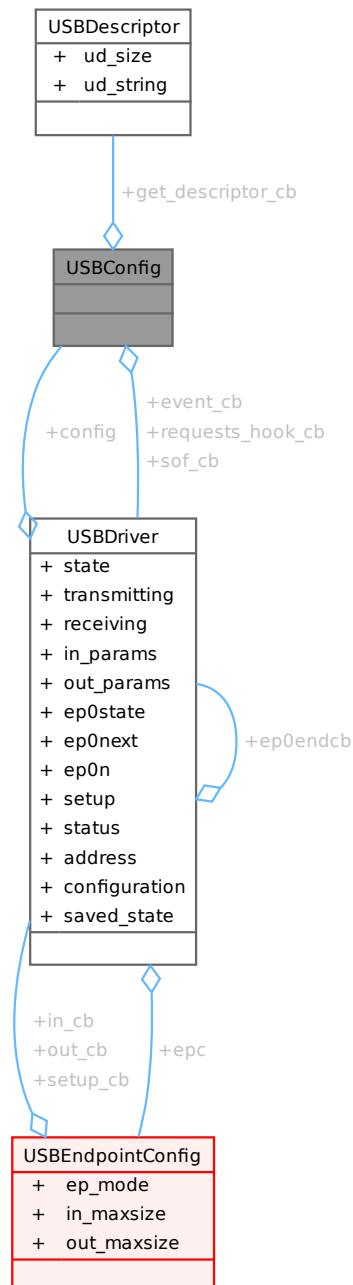
```
uint8_t unpacked_sdc_csd_20_t::crc
```

8.133 USBConfig Struct Reference

Type of an USB driver configuration structure.

```
#include <hal_usb_ll.h>
```

Collaboration diagram for USBConfig:



Data Fields

- [usbeventcb_t event_cb](#)
USB events callback.
- [usbgetdescriptor_t get_descriptor_cb](#)
Device GET_DESCRIPTOR request callback.
- [usbreqhandler_t requests_hook_cb](#)
Requests hook callback.
- [usbcallback_t sof_cb](#)
Start Of Frame callback.

8.133.1 Detailed Description

Type of an USB driver configuration structure.

8.133.2 Field Documentation

8.133.2.1 event_cb

[usbeventcb_t](#) USBConfig::event_cb

USB events callback.

This callback is invoked when an USB driver event is registered.

8.133.2.2 get_descriptor_cb

[usbgetdescriptor_t](#) USBConfig::get_descriptor_cb

Device GET_DESCRIPTOR request callback.

Note

This callback is mandatory and cannot be set to NULL.

8.133.2.3 requests_hook_cb

[usbreqhandler_t](#) USBConfig::requests_hook_cb

Requests hook callback.

This hook allows to be notified of standard requests or to handle non standard requests.

8.133.2.4 sof_cb

[usbcallback_t](#) USBConfig::sof_cb

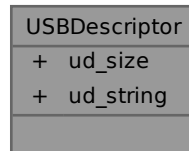
Start Of Frame callback.

8.134 USBDescriptor Struct Reference

Type of an USB descriptor.

```
#include <hal_usb.h>
```

Collaboration diagram for USBDescriptor:



Data Fields

- `size_t` [ud_size](#)
Descriptor size in unicode characters.
- `const uint8_t *` [ud_string](#)
Pointer to the descriptor.

8.134.1 Detailed Description

Type of an USB descriptor.

8.134.2 Field Documentation

8.134.2.1 `ud_size`

```
size_t USBDescriptor::ud_size
```

Descriptor size in unicode characters.

8.134.2.2 `ud_string`

```
const uint8_t* USBDescriptor::ud_string
```

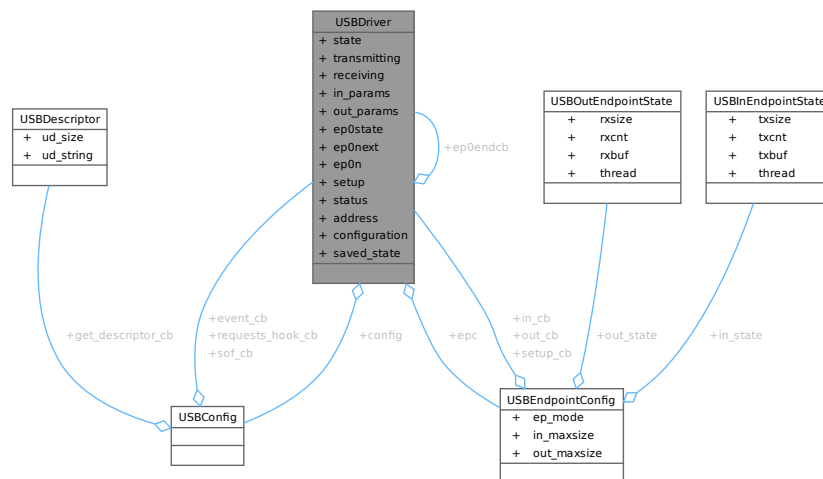
Pointer to the descriptor.

8.135 USBDriver Struct Reference

Structure representing an USB driver.

```
#include <hal_usb_ll.h>
```

Collaboration diagram for USBDriver:



Data Fields

- [usbstate_t state](#)
Driver state.
- `const USBConfig * config`
Current configuration data.
- `uint16_t transmitting`
Bit map of the transmitting IN endpoints.
- `uint16_t receiving`
Bit map of the receiving OUT endpoints.
- `const USBEndpointConfig * epc [USB_MAX_ENDPOINTS+1]`
Active endpoints configurations.
- `void * in_params [USB_MAX_ENDPOINTS]`
Fields available to user, it can be used to associate an application-defined handler to an IN endpoint.
- `void * out_params [USB_MAX_ENDPOINTS]`
Fields available to user, it can be used to associate an application-defined handler to an OUT endpoint.
- [usbep0state_t ep0state](#)
Endpoint 0 state.
- `uint8_t * ep0next`
Next position in the buffer to be transferred through endpoint 0.
- `size_t ep0n`
Number of bytes yet to be transferred through endpoint 0.
- [usbcallback_t ep0endcb](#)
Endpoint 0 end transaction callback.
- `uint8_t setup [8]`

- Setup packet buffer.*
- `uint16_t` `status`
Current USB device status.
- `uint8_t` `address`
Assigned USB address.
- `uint8_t` `configuration`
Current USB device configuration.
- `usbstate_t` `saved_state`
State of the driver when a suspend happened.

8.135.1 Detailed Description

Structure representing an USB driver.

8.135.2 Field Documentation

8.135.2.1 state

```
usbstate_t USBDriver::state
```

Driver state.

8.135.2.2 config

```
const USBConfig* USBDriver::config
```

Current configuration data.

8.135.2.3 transmitting

```
uint16_t USBDriver::transmitting
```

Bit map of the transmitting IN endpoints.

8.135.2.4 receiving

```
uint16_t USBDriver::receiving
```

Bit map of the receiving OUT endpoints.

8.135.2.5 epc

```
const USBEndpointConfig* USBDriver::epc[USB_MAX_ENDPOINTS+1]
```

Active endpoints configurations.

8.135.2.6 in_params

```
void* USBDriver::in_params[USB_MAX_ENDPOINTS]
```

Fields available to user, it can be used to associate an application-defined handler to an IN endpoint.

Note

The base index is one, the endpoint zero does not have a reserved element in this array.

8.135.2.7 out_params

```
void* USBDriver::out_params[USB_MAX_ENDPOINTS]
```

Fields available to user, it can be used to associate an application-defined handler to an OUT endpoint.

Note

The base index is one, the endpoint zero does not have a reserved element in this array.

8.135.2.8 ep0state

```
usbep0state_t USBDriver::ep0state
```

Endpoint 0 state.

8.135.2.9 ep0next

```
uint8_t* USBDriver::ep0next
```

Next position in the buffer to be transferred through endpoint 0.

8.135.2.10 ep0n

```
size_t USBDriver::ep0n
```

Number of bytes yet to be transferred through endpoint 0.

8.135.2.11 ep0endcb

```
usbcallback_t USBDriver::ep0endcb
```

Endpoint 0 end transaction callback.

8.135.2.12 setup

```
uint8_t USBDriver::setup[8]
```

Setup packet buffer.

8.135.2.13 status

```
uint16_t USBDriver::status
```

Current USB device status.

8.135.2.14 address

```
uint8_t USBDriver::address
```

Assigned USB address.

8.135.2.15 configuration

```
uint8_t USBDriver::configuration
```

Current USB device configuration.

8.135.2.16 saved_state

```
usbstate_t USBDriver::saved_state
```

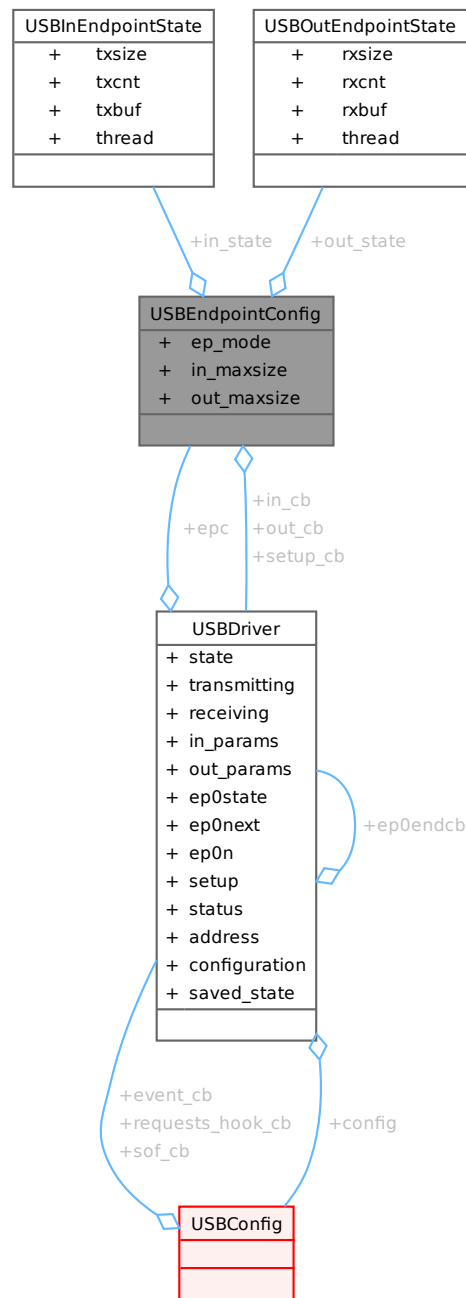
State of the driver when a suspend happened.

8.136 USBEndpointConfig Struct Reference

Type of an USB endpoint configuration structure.

```
#include <hal_usb_lld.h>
```

Collaboration diagram for USBEndpointConfig:



Data Fields

- `uint32_t ep_mode`
Type and mode of the endpoint.
- `usbepcallback_t setup_cb`
Setup packet notification callback.
- `usbepcallback_t in_cb`

- IN endpoint notification callback.*
- [usbepcallback_t out_cb](#)
- OUT endpoint notification callback.*
- [uint16_t in_maxsize](#)
- IN endpoint maximum packet size.*
- [uint16_t out_maxsize](#)
- OUT endpoint maximum packet size.*
- [USBInEndpointState * in_state](#)
- USBEndpointState associated to the IN endpoint.*
- [USBOutEndpointState * out_state](#)
- USBEndpointState associated to the OUT endpoint.*

8.136.1 Detailed Description

Type of an USB endpoint configuration structure.

Note

Platform specific restrictions may apply to endpoints.

8.136.2 Field Documentation

8.136.2.1 ep_mode

```
uint32_t USBEndpointConfig::ep_mode
```

Type and mode of the endpoint.

8.136.2.2 setup_cb

```
usbepcallback_t USBEndpointConfig::setup_cb
```

Setup packet notification callback.

This callback is invoked when a setup packet has been received.

Postcondition

The application must immediately call `usbReadPacket()` in order to access the received packet.

Note

This field is only valid for `USB_EP_MODE_TYPE_CTRL` endpoints, it should be set to `NULL` for other endpoint types.

8.136.2.3 in_cb

`usbepcallback_t USBEndpointConfig::in_cb`

IN endpoint notification callback.

This field must be set to `NULL` if the IN endpoint is not used.

8.136.2.4 out_cb

`usbepcallback_t USBEndpointConfig::out_cb`

OUT endpoint notification callback.

This field must be set to `NULL` if the OUT endpoint is not used.

8.136.2.5 in_maxsize

`uint16_t USBEndpointConfig::in_maxsize`

IN endpoint maximum packet size.

This field must be set to zero if the IN endpoint is not used.

8.136.2.6 out_maxsize

`uint16_t USBEndpointConfig::out_maxsize`

OUT endpoint maximum packet size.

This field must be set to zero if the OUT endpoint is not used.

8.136.2.7 in_state

`USBInEndpointState* USBEndpointConfig::in_state`

`USBEndpointState` associated to the IN endpoint.

This structure maintains the state of the IN endpoint.

8.136.2.8 out_state

`USBOutEndpointState* USBEndpointConfig::out_state`

`USBEndpointState` associated to the OUT endpoint.

This structure maintains the state of the OUT endpoint.

8.137 USBInEndpointState Struct Reference

Type of an IN endpoint state structure.

```
#include <hal_usb_llid.h>
```

Collaboration diagram for USBInEndpointState:

USBInEndpointState	
+	txsize
+	txcnt
+	txbuf
+	thread

Data Fields

- `size_t txsize`
Requested transmit transfer size.
- `size_t txcnt`
Transmitted bytes so far.
- `const uint8_t * txbuf`
Pointer to the transmission linear buffer.
- `thread_reference_t thread`
Waiting thread.

8.137.1 Detailed Description

Type of an IN endpoint state structure.

8.137.2 Field Documentation

8.137.2.1 txsize

```
size_t USBInEndpointState::txsize
```

Requested transmit transfer size.

8.137.2.2 txcnt

```
size_t USBInEndpointState::txcnt
```

Transmitted bytes so far.

8.137.2.3 txbuf

```
const uint8_t* USBInEndpointState::txbuf
```

Pointer to the transmission linear buffer.

8.137.2.4 thread

```
thread_reference_t USBInEndpointState::thread
```

Waiting thread.

8.138 USBOutEndpointState Struct Reference

Type of an OUT endpoint state structure.

```
#include <hal_usb_lld.h>
```

Collaboration diagram for USBOutEndpointState:

USBOutEndpointState	
+	rxsize
+	rxcnt
+	rxbuf
+	thread

Data Fields

- `size_t rxsize`
Requested receive transfer size.
- `size_t rxcnt`
Received bytes so far.
- `uint8_t * rxbuf`
Pointer to the receive linear buffer.
- `thread_reference_t thread`
Waiting thread.

8.138.1 Detailed Description

Type of an OUT endpoint state structure.

8.138.2 Field Documentation

8.138.2.1 rxsize

```
size_t USBOutEndpointState::rxsize
```

Requested receive transfer size.

8.138.2.2 rxcnt

```
size_t USBOutEndpointState::rxcnt
```

Received bytes so far.

8.138.2.3 rxbuf

```
uint8_t* USBOutEndpointState::rxbuf
```

Pointer to the receive linear buffer.

8.138.2.4 thread

```
thread_reference_t USBOutEndpointState::thread
```

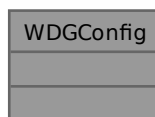
Waiting thread.

8.139 WDGConfig Struct Reference

Driver configuration structure.

```
#include <hal_wdg_lld.h>
```

Collaboration diagram for WDGConfig:



8.139.1 Detailed Description

Driver configuration structure.

Note

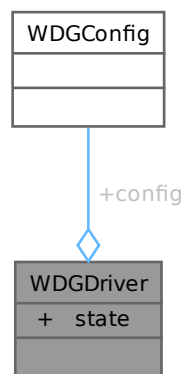
It could be empty on some architectures.

8.140 WDGDriver Struct Reference

Structure representing an WDG driver.

```
#include <hal_wdg_ll.h>
```

Collaboration diagram for WDGDriver:



Data Fields

- [wdgstate_t state](#)
Driver state.
- `const WDGConfig * config`
Current configuration data.

8.140.1 Detailed Description

Structure representing an WDG driver.

8.140.2 Field Documentation

8.140.2.1 state

`wdgstate_t` `WDGDriver::state`

Driver state.

8.140.2.2 config

`const WDGConfig*` `WDGDriver::config`

Current configuration data.

8.141 `wspi_command_t` Struct Reference

Type of a WSPI command descriptor.

```
#include <hal_wspi.h>
```

Collaboration diagram for `wspi_command_t`:

wspi_command_t	
+	cfg
+	cmd
+	addr
+	alt
+	dummy

Data Fields

- `uint32_t` `cfg`
Transfer configuration field.
- `uint32_t` `cmd`
Command phase data.
- `uint32_t` `addr`
Address phase data.
- `uint32_t` `alt`
Alternate phase data.
- `uint32_t` `dummy`
Number of dummy cycles to be inserted.

8.141.1 Detailed Description

Type of a WSPI command descriptor.

8.141.2 Field Documentation

8.141.2.1 cfg

```
uint32_t wspi_command_t::cfg
```

Transfer configuration field.

8.141.2.2 cmd

```
uint32_t wspi_command_t::cmd
```

Command phase data.

8.141.2.3 addr

```
uint32_t wspi_command_t::addr
```

Address phase data.

8.141.2.4 alt

```
uint32_t wspi_command_t::alt
```

Alternate phase data.

8.141.2.5 dummy

```
uint32_t wspi_command_t::dummy
```

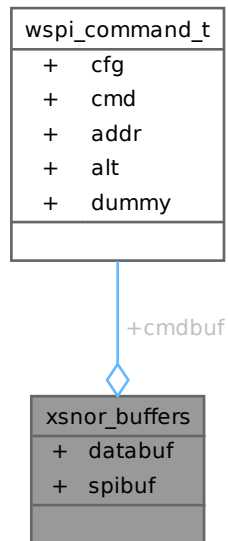
Number of dummy cycles to be inserted.

8.142 xsnor_buffers Struct Reference

SNOR driver configuration.

```
#include <hal_xsnor_base.h>
```

Collaboration diagram for xsnor_buffers:



Data Fields

- `uint8_t databuf [XSNOR_BUFFER_SIZE]`
Non-cacheable data buffer.
- `wspi_command_t cmdbuf`
Non-cacheable WSPI command buffer.
- `uint8_t spibuf [8]`
Non-cacheable SPI buffer.

8.142.1 Detailed Description

SNOR driver configuration.

8.142.2 Field Documentation

8.142.2.1 databuf

```
uint8_t xsnor_buffers::databuf[XSNOR_BUFFER_SIZE]
```

Non-cacheable data buffer.

8.142.2.2 cmdbuf

```
wspi_command_t xsnor_buffers::cmdbuf
```

Non-cacheable WSPI command buffer.

8.142.2.3 spibuf

```
uint8_t xsnor_buffers::spibuf[8]
```

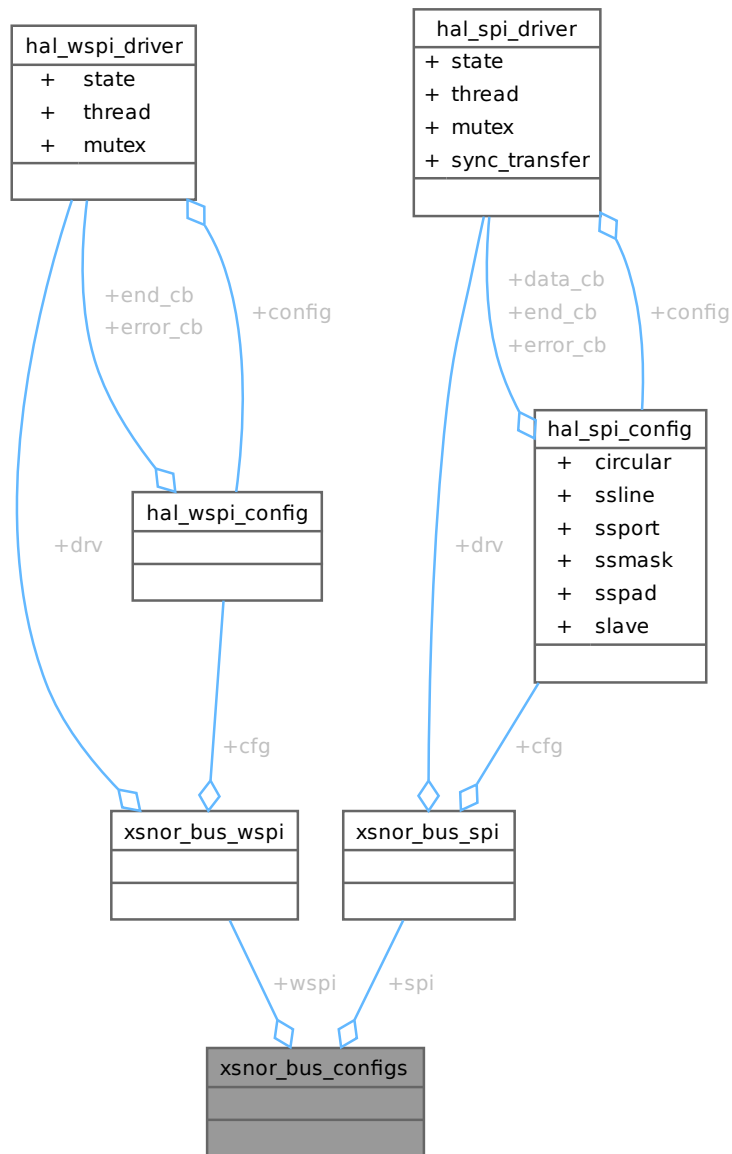
Non-cacheable SPI buffer.

8.143 xsnor_bus_configs Union Reference

Union of possible bus configurations.

```
#include <hal_xsnor_base.h>
```

Collaboration diagram for `xsnor_bus_configs`:



Data Fields

- struct `xsnor_bus_wspi` `wspi`
- struct `xsnor_bus_spi` `spi`

8.143.1 Detailed Description

Union of possible bus configurations.

8.143.2 Field Documentation

8.143.2.1 wspi

```
struct xsnor_bus_wspi xsnor_bus_configs::wspi
```

8.143.2.2 spi

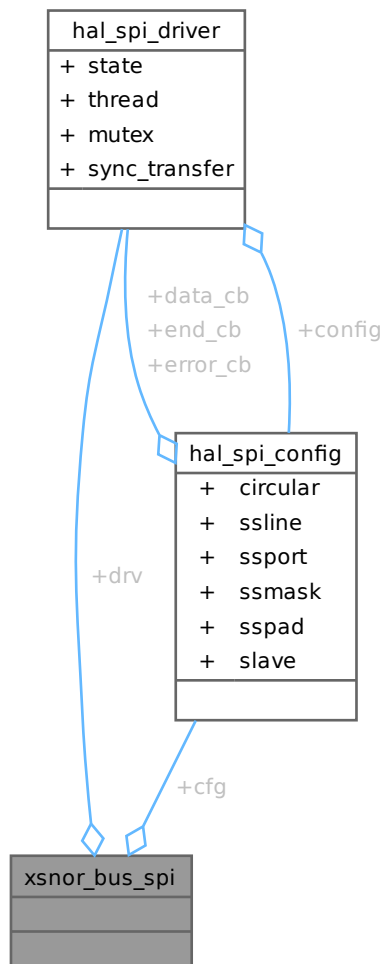
```
struct xsnor_bus_spi xsnor_bus_configs::spi
```

8.144 xsnor_bus_spi Struct Reference

SPI-specific configuration fields.

```
#include <hal_xsnor_base.h>
```

Collaboration diagram for xsnor_bus_spi:



Data Fields

- `SPIDriver * drv`
SPI driver to be used for physical communication.
- `const SPIConfig * cfg`
SPI driver configuration.

8.144.1 Detailed Description

SPI-specific configuration fields.

8.144.2 Field Documentation

8.144.2.1 `drv`

```
SPIDriver* xsnor_bus_spi::drv
```

SPI driver to be used for physical communication.

8.144.2.2 `cfg`

```
const SPIConfig* xsnor_bus_spi::cfg
```

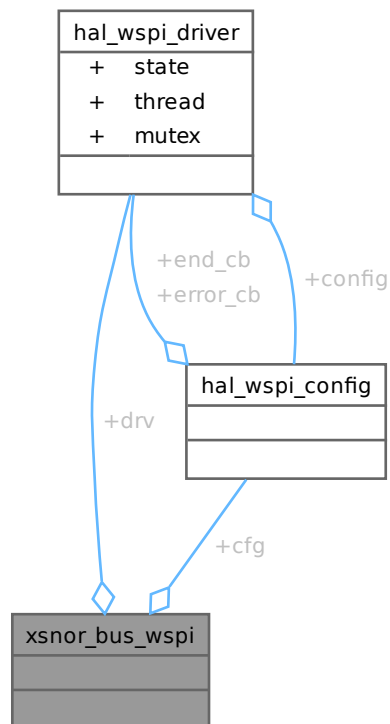
SPI driver configuration.

8.145 `xsnor_bus_wspi` Struct Reference

WSPI-specific configuration fields.

```
#include <hal_xsnor_base.h>
```


Collaboration diagram for xsnor_bus_wspi:



Data Fields

- [WSPIDriver](#) * `drv`
WSPi driver to be used for physical communication.
- const [WSPICfg](#) * `cfg`
WSPi driver configuration.

8.145.1 Detailed Description

WSPi-specific configuration fields.

8.145.2 Field Documentation

8.145.2.1 `drv`

[WSPIDriver](#)* `xsnor_bus_wspi::drv`

WSPi driver to be used for physical communication.

8.145.2.2 cfg

```
const WSPICConfig* xsnor_bus_wspi::cfg
```

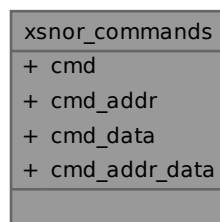
WSPI driver configuration.

8.146 xsnor_commands Struct Reference

SNOR command configuration structure.

```
#include <hal_xsnor_base.h>
```

Collaboration diagram for xsnor_commands:



Data Fields

- `uint32_t cmd`
Command only.
- `uint32_t cmd_addr`
Command and address.
- `uint32_t cmd_data`
Command and data.
- `uint32_t cmd_addr_data`
Command, address and data.

8.146.1 Detailed Description

SNOR command configuration structure.

8.146.2 Field Documentation

8.146.2.1 cmd

```
uint32_t xsnor_commands::cmd
```

Command only.

8.146.2.2 cmd_addr

```
uint32_t xsnor_commands::cmd_addr
```

Command and address.

8.146.2.3 cmd_data

```
uint32_t xsnor_commands::cmd_data
```

Command and data.

8.146.2.4 cmd_addr_data

```
uint32_t xsnor_commands::cmd_addr_data
```

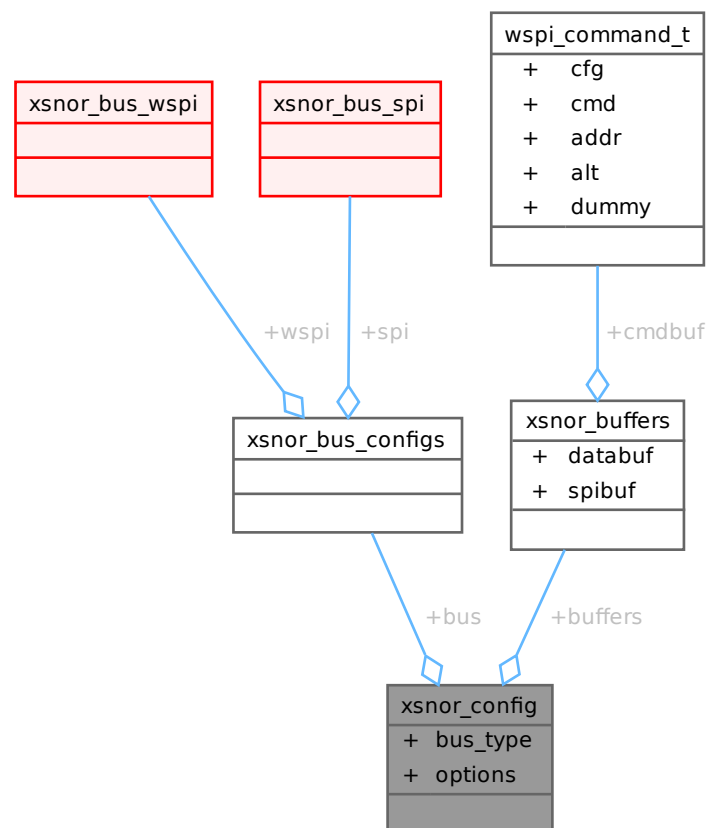
Command, address and data.

8.147 xsnor_config Struct Reference

SNOR driver configuration.

```
#include <hal_xsnor_base.h>
```

Collaboration diagram for xsnor_config:



Data Fields

- int [bus_type](#)
Bus type selection switch.
- union [xsnor_bus_configs](#) [bus](#)
WSPI driver configuration.
- [xsnor_buffers_t](#) * [buffers](#)
Pointer to the non-cacheable buffers.
- int [options](#)
Device-dependent options, used by subclasses only.

8.147.1 Detailed Description

SNOR driver configuration.

8.147.2 Field Documentation

8.147.2.1 bus_type

```
int xsnor_config::bus_type
```

Bus type selection switch.

8.147.2.2 bus

```
union xsnor\_bus\_configs xsnor_config::bus
```

WSPI driver configuration.

8.147.2.3 buffers

```
xsnor\_buffers\_t* xsnor_config::buffers
```

Pointer to the non-cacheable buffers.

8.147.2.4 options

```
int xsnor_config::options
```

Device-dependent options, used by subclasses only.

Chapter 9

File Documentation

9.1 main.dox File Reference

9.2 main.dox File Reference

9.3 ccportab.h File Reference

Compiler portability layer.

Macros

Compiler abstraction macros

- #define `CC_SECTION(s)`
Allocates a variable or function to a specific section.
- #define `CC_WEAK __attribute__((weak))`
Marks a function or variable as a weak symbol.
- #define `CC_USED __attribute__((used))`
Marks a function or variable as used.
- #define `CC_ALIGN_DATA(n)`
Enforces alignment of the variable declared afterward.
- #define `CC_ALIGN_CODE(n)`
Enforces alignment of a function declared afterward.
- #define `CC_PACK __attribute__((packed))`
Enforces packing of the structure declared afterward.
- #define `CC_NO_INLINE __attribute__((noinline))`
Marks a function as not inlineable.
- #define `CC_FORCE_INLINE __attribute__((always_inline))`
Enforces a function inline.
- #define `CC_NO_RETURN __attribute__((noreturn))`
Marks a function as non-returning.
- #define `CC_ROMCONST const`
Enforces a variable in a ROM area.
- #define `CC_LIKELY(x)`
Marks a boolean expression as likely true.
- #define `CC_UNLIKELY(x)`
Marks a boolean expression as likely false.

9.3.1 Detailed Description

Compiler portability layer.

9.4 hal.dox File Reference

9.5 hal_adc.dox File Reference

9.6 hal_buffererd_sio.dox File Reference

9.7 hal_buffers.dox File Reference

9.8 hal_can.dox File Reference

9.9 hal_crypto.dox File Reference

9.10 hal_dac.dox File Reference

9.11 hal_efl.dox File Reference

9.12 hal_flash.dox File Reference

9.13 hal_gpt.dox File Reference

9.14 hal_i2c.dox File Reference

9.15 hal_i2s.dox File Reference

9.16 hal_icu.dox File Reference

9.17 hal_interfaces.dox File Reference

9.18 hal_ioblock.dox File Reference

9.19 hal_mac.dox File Reference

9.20 hal_mfs.dox File Reference

9.21 hal_mii.dox File Reference

9.22 hal_mmc_spi.dox File Reference

9.23 hal_mmcstd.dox File Reference

9.24 hal_norflash.dox File Reference

9.25 hal_objects.dox File Reference

9.26 hal_pal.dox File Reference

9.27 hal_peripherals.dox File Reference


```
#include "ccportab.h"
#include "osal.h"
#include "board.h"
#include "halconf.h"
#include "hal_lld.h"
#include "hal_objects.h"
#include "hal_streams.h"
#include "hal_channels.h"
#include "hal_files.h"
#include "hal_ioblock.h"
#include "hal_mmcfsd.h"
#include "hal_persistent.h"
#include "hal_flash.h"
#include "hal_buffers.h"
#include "hal_queues.h"
#include "hal_buffered_serial.h"
#include "hal_pal.h"
#include "hal_adc.h"
#include "hal_can.h"
#include "hal_crypto.h"
#include "hal_dac.h"
#include "hal_efl.h"
#include "hal_gpt.h"
#include "hal_i2c.h"
#include "hal_i2s.h"
#include "hal_icu.h"
#include "hal_mac.h"
#include "hal_pwm.h"
#include "hal_rtc.h"
#include "hal_serial.h"
#include "hal_sdc.h"
#include "hal_sio.h"
#include "hal_spi.h"
#include "hal_trng.h"
#include "hal_uart.h"
#include "hal_usb.h"
#include "hal_wdg.h"
#include "hal_wspi.h"
#include "hal_mmc_spi.h"
#include "hal_serial_usb.h"
#include "hal_community.h"
```

Macros

- `#define __CHIBIOS_HAL__`
ChibiOS/HAL identification macro.
- `#define CH_HAL_STABLE 1`
Stable release flag.
- `#define HAL_USE_PAL FALSE`
- `#define HAL_USE_ADC FALSE`
- `#define HAL_USE_CAN FALSE`
- `#define HAL_USE_CRY FALSE`
- `#define HAL_USE_DAC FALSE`
- `#define HAL_USE_EFL FALSE`
- `#define HAL_USE_GPT FALSE`

- #define HAL_USE_I2C FALSE
- #define HAL_USE_I2S FALSE
- #define HAL_USE_ICU FALSE
- #define HAL_USE_MAC FALSE
- #define HAL_USE_PWM FALSE
- #define HAL_USE_RTC FALSE
- #define HAL_USE_SERIAL FALSE
- #define HAL_USE_SDC FALSE
- #define HAL_USE_SIO FALSE
- #define HAL_USE_SPI FALSE
- #define HAL_USE_TRNG FALSE
- #define HAL_USE_UART FALSE
- #define HAL_USE_USB FALSE
- #define HAL_USE_WDG FALSE
- #define HAL_USE_WSPI FALSE

ChibiOS/HAL version identification

- #define CH_HAL_VERSION "9.0.0"
HAL version string.
- #define CH_HAL_MAJOR 9
HAL version major number.
- #define CH_HAL_MINOR 0
HAL version minor number.
- #define CH_HAL_PATCH 0
HAL version patch number.

Return codes

- #define HAL_SUCCESS false
HAL operation success.
- #define HAL_FAILED true
HAL operation failed.

Return codes for HAL functions

- #define HAL_RET_SUCCESS MSG_OK
- #define HAL_RET_CONFIG_ERROR (msg_t)-16
Configuration error.
- #define HAL_RET_NO_RESOURCE (msg_t)-17
A required resource is not available.
- #define HAL_RET_HW_BUSY (msg_t)-18
The peripheral is busy.
- #define HAL_RET_HW_FAILURE (msg_t)-19
Peripheral failure.
- #define HAL_RET_UNKNOWN_CTL (msg_t)-20
Unknown control code.
- #define HAL_RET_IS_INVALID (msg_t)-21
Invalid instance pointer.

Typedefs

- typedef unsigned halclkpt_t
Type of a clock point identifier.
- typedef uint32_t halfreq_t
Type of a clock point frequency in Hz.

Functions

- void [halInit](#) (void)
HAL initialization.
- static [halfreq_t](#) [halClockGetPointX](#) ([halclkpt_t](#) clkpt)
Returns the frequency of a clock point in Hz.
- static bool [halClockSwitchMode](#) (const [halclkcfg_t](#) *ccp)
Switches to a different clock configuration.

9.45.1 Detailed Description

HAL subsystem header.

9.46 hal_adc.h File Reference

ADC Driver macros and structures.

```
#include "hal_adc_llld.h"
```

Data Structures

- struct [hal_adc_configuration_group](#)
Conversion group configuration structure.
- struct [hal_adc_config](#)
Driver configuration structure.
- struct [hal_adc_driver](#)
Structure representing an ADC driver.

Macros

ADC configuration options

- #define [ADC_USE_WAIT](#) TRUE
Enables synchronous APIs.
- #define [ADC_USE_MUTUAL_EXCLUSION](#) TRUE
Enables the [adcAcquireBus\(\)](#) and [adcReleaseBus\(\)](#) APIs.

Macro Functions

- #define [adclsbBufferComplete](#)(adcp)
Buffer state.

Low level driver helper macros

- #define [_adc_reset_i](#)(adcp)
Resumes a thread waiting for a conversion completion.
- #define [_adc_reset_s](#)(adcp)
Resumes a thread waiting for a conversion completion.
- #define [_adc_wakeup_isr](#)(adcp)
Wakes up the waiting thread.
- #define [_adc_timeout_isr](#)(adcp)
Wakes up the waiting thread with a timeout message.
- #define [_adc_isr_half_code](#)(adcp)
Common ISR code, half buffer event.
- #define [_adc_isr_full_code](#)(adcp)
Common ISR code, full buffer event.
- #define [_adc_isr_error_code](#)(adcp, err)
Common ISR code, error event.

Typedefs

- typedef struct [hal_adc_driver](#) ADCDriver
Type of a structure representing an ADC driver.
- typedef struct [hal_adc_config](#) ADCConfig
Type of a structure representing an ADC driver configuration.
- typedef struct [hal_adc_configuration_group](#) ADCConversionGroup
Conversion group configuration structure.
- typedef void(* [adccallback_t](#)) (ADCDriver *adcp)
Type of an ADC notification callback.
- typedef void(* [adcerrorcallback_t](#)) (ADCDriver *adcp, [adcerror_t](#) err)
Type of an ADC error callback.

Enumerations

- enum [adcstate_t](#) {
 [ADC_UNINIT](#) = 0 , [ADC_STOP](#) = 1 , [ADC_READY](#) = 2 , [ADC_ACTIVE](#) = 3 ,
 [ADC_COMPLETE](#) = 4 , [ADC_ERROR](#) = 5 }
Driver state machine possible states.

Functions

- void [adclnit](#) (void)
ADC Driver initialization.
- void [adcObjectInit](#) (ADCDriver *adcp)
Initializes the standard part of a [ADCDriver](#) structure.
- [msg_t](#) [adcStart](#) (ADCDriver *adcp, const ADCConfig *config)
Configures and activates the ADC peripheral.
- void [adcStop](#) (ADCDriver *adcp)
Deactivates the ADC peripheral.
- void [adcStartConversion](#) (ADCDriver *adcp, const ADCConversionGroup *grpp, [adcsample_t](#) *samples, size_t depth)
Starts an ADC conversion.
- void [adcStartConversionI](#) (ADCDriver *adcp, const ADCConversionGroup *grpp, [adcsample_t](#) *samples, size_t depth)
Starts an ADC conversion.
- void [adcStopConversion](#) (ADCDriver *adcp)
Stops an ongoing conversion.
- void [adcStopConversionI](#) (ADCDriver *adcp)
Stops an ongoing conversion.
- [msg_t](#) [adcConvert](#) (ADCDriver *adcp, const ADCConversionGroup *grpp, [adcsample_t](#) *samples, size_t depth)
Performs an ADC conversion.
- void [adcAcquireBus](#) (ADCDriver *adcp)
Gains exclusive access to the ADC peripheral.
- void [adcReleaseBus](#) (ADCDriver *adcp)
Releases exclusive access to the ADC peripheral.

9.46.1 Detailed Description

ADC Driver macros and structures.

9.47 hal_buffered_serial.h File Reference

Buffered Serial Driver code.

Data Structures

- struct [BufferedSerialVMT](#)
BufferedSerial virtual methods table.
- struct [hal_buffered_serial](#)
Buffered serial channel class.

Macros

- #define [__buffered_serial_methods_base_asynchronous_channel_methods](#)
BufferedSerial specific methods.
- #define [__buffered_serial_data](#)
BufferedSerial specific data.

Typedefs

- typedef struct [hal_buffered_serial](#) [BufferedSerial](#)
Structure representing a buffered serial class.

Enumerations

- enum [bsstate_t](#) { [BS_UNINIT](#) = 0 , [BS_STOP](#) = 1 , [BS_READY](#) = 2 }
BufferedSerial state machine states.

Functions

- void [bsIncomingData](#) ([BufferedSerial](#) *bsp, uint8_t b)
Handles incoming data.
- [msg_t bsRequestData](#) ([BufferedSerial](#) *bsp)
Handles outgoing data.

Methods implementations

- static [CC_FORCE_INLINE](#) void [__buffered_serial_objinit_impl](#) (void *ip, const void *vmt, uint8_t *ib, size_t ibsize, [qnotify_t](#) inotify, void *iarg, uint8_t *ob, size_t obsize, [qnotify_t](#) onotify, void *oarg)
Object initialization implementation.
- static [CC_FORCE_INLINE](#) size_t [__buffered_serial_write_impl](#) (void *ip, const uint8_t *bp, size_t n)
- static [CC_FORCE_INLINE](#) size_t [__buffered_serial_read_impl](#) (void *ip, uint8_t *bp, size_t n)
- static [CC_FORCE_INLINE](#) [msg_t](#) [__buffered_serial_put_impl](#) (void *ip, uint8_t b)
- static [CC_FORCE_INLINE](#) [msg_t](#) [__buffered_serial_get_impl](#) (void *ip)
- static [CC_FORCE_INLINE](#) [msg_t](#) [__buffered_serial_put_timeout_impl](#) (void *ip, uint8_t b, [sysinterval_t](#) timeout)
- static [CC_FORCE_INLINE](#) [msg_t](#) [__buffered_serial_get_timeout_impl](#) (void *ip, [sysinterval_t](#) timeout)
- static [CC_FORCE_INLINE](#) size_t [__buffered_serial_write_timeout_impl](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static [CC_FORCE_INLINE](#) size_t [__buffered_serial_read_timeout_impl](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static [CC_FORCE_INLINE](#) [msg_t](#) [__buffered_serial_ctl_impl](#) (void *ip, unsigned int operation, void *arg)

9.47.1 Detailed Description

Buffered Serial Driver code.

Buffered Serial Driver macros and structures.

9.48 hal_buffers.h File Reference

I/O Buffers macros and structures.

Data Structures

- struct [io_buffers_queue](#)
Structure of a generic buffers queue.

Macros

- #define [BUFFERS_CHUNKS_SIZE](#) 64
Maximum size of blocks copied in critical sections.
- #define [BQ_BUFFER_SIZE](#)(n, size)
Computes the size of a buffers queue buffer size.

Macro Functions

- #define [bqSizeX](#)(bqp)
Returns the queue's number of buffers.
- #define [bqSpacel](#)(bqp)
Return the ready buffers number.
- #define [bqGetLinkX](#)(bqp)
Returns the queue application-defined link.
- #define [bqSetLinkX](#)(bqp, lk)
Sets the queue application-defined link.
- #define [bqlsSuspendedX](#)(bqp)
Return the suspended state of the queue.
- #define [bqSuspendl](#)(bqp)
Puts the queue in suspended state.
- #define [bqResumeX](#)(bqp)
Resumes normal queue operations.
- #define [ibqlsEmptyl](#)(ibqp)
Evaluates to `true` if the specified input buffers queue is empty.
- #define [ibqlsFullI](#)(ibqp)
Evaluates to `true` if the specified input buffers queue is full.
- #define [obqlsEmptyl](#)(obqp)
Evaluates to `true` if the specified output buffers queue is empty.
- #define [obqlsFullI](#)(obqp)
Evaluates to `true` if the specified output buffers queue is full.

Typedefs

- typedef struct [io_buffers_queue](#) [io_buffers_queue_t](#)
Type of a generic queue of buffers.
- typedef void(* [bqnotify_t](#)) ([io_buffers_queue_t](#) *bqp)
Double buffer notification callback type.
- typedef [io_buffers_queue_t](#) [input_buffers_queue_t](#)
Type of an input buffers queue.
- typedef [io_buffers_queue_t](#) [output_buffers_queue_t](#)
Type of an output buffers queue.

Functions

- void [ibqObjectInit](#) ([input_buffers_queue_t](#) *ibqp, bool suspended, uint8_t *bp, size_t size, size_t n, [bqnotify_t](#) infy, void *link)
Initializes an input buffers queue object.
- void [ibqResetI](#) ([input_buffers_queue_t](#) *ibqp)
Resets an input buffers queue.
- uint8_t * [ibqGetEmptyBufferI](#) ([input_buffers_queue_t](#) *ibqp)
Gets the next empty buffer from the queue.
- void [ibqPostFullBufferI](#) ([input_buffers_queue_t](#) *ibqp, size_t size)
Posts a new filled buffer to the queue.
- [msg_t](#) [ibqGetFullBufferTimeout](#) ([input_buffers_queue_t](#) *ibqp, [sysinterval_t](#) timeout)
Gets the next filled buffer from the queue.
- [msg_t](#) [ibqGetFullBufferTimeoutS](#) ([input_buffers_queue_t](#) *ibqp, [sysinterval_t](#) timeout)
Gets the next filled buffer from the queue.
- void [ibqReleaseEmptyBuffer](#) ([input_buffers_queue_t](#) *ibqp)
Releases the buffer back in the queue.
- void [ibqReleaseEmptyBufferS](#) ([input_buffers_queue_t](#) *ibqp)
Releases the buffer back in the queue.
- [msg_t](#) [ibqGetTimeout](#) ([input_buffers_queue_t](#) *ibqp, [sysinterval_t](#) timeout)
Input queue read with timeout.
- size_t [ibqReadTimeout](#) ([input_buffers_queue_t](#) *ibqp, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
Input queue read with timeout.
- void [obqObjectInit](#) ([output_buffers_queue_t](#) *obqp, bool suspended, uint8_t *bp, size_t size, size_t n, [bqnotify_t](#) onfy, void *link)
Initializes an output buffers queue object.
- void [obqResetI](#) ([output_buffers_queue_t](#) *obqp)
Resets an output buffers queue.
- uint8_t * [obqGetFullBufferI](#) ([output_buffers_queue_t](#) *obqp, size_t *sizep)
Gets the next filled buffer from the queue.
- void [obqReleaseEmptyBufferI](#) ([output_buffers_queue_t](#) *obqp)
Releases the next filled buffer back in the queue.
- [msg_t](#) [obqGetEmptyBufferTimeout](#) ([output_buffers_queue_t](#) *obqp, [sysinterval_t](#) timeout)
Gets the next empty buffer from the queue.
- [msg_t](#) [obqGetEmptyBufferTimeoutS](#) ([output_buffers_queue_t](#) *obqp, [sysinterval_t](#) timeout)
Gets the next empty buffer from the queue.
- void [obqPostFullBuffer](#) ([output_buffers_queue_t](#) *obqp, size_t size)
Posts a new filled buffer to the queue.
- void [obqPostFullBufferS](#) ([output_buffers_queue_t](#) *obqp, size_t size)
Posts a new filled buffer to the queue.

- `msg_t obqPutTimeout` (`output_buffers_queue_t *obqp`, `uint8_t b`, `sysinterval_t timeout`)
Output queue write with timeout.
- `size_t obqWriteTimeout` (`output_buffers_queue_t *obqp`, `const uint8_t *bp`, `size_t n`, `sysinterval_t timeout`)
Output queue write with timeout.
- `bool obqTryFlush` (`output_buffers_queue_t *obqp`)
Flushes the current, partially filled, buffer to the queue.
- `void obqFlush` (`output_buffers_queue_t *obqp`)
Flushes the current, partially filled, buffer to the queue.

9.48.1 Detailed Description

I/O Buffers macros and structures.

9.49 hal_can.h File Reference

CAN Driver macros and structures.

```
#include "hal_can_11d.h"
```

Macros

- `#define CAN_ANY_MAILBOX 0U`
Special mailbox identifier.

CAN status flags

- `#define CAN_LIMIT_WARNING 1U`
Errors rate warning.
- `#define CAN_LIMIT_ERROR 2U`
Errors rate error.
- `#define CAN_BUS_OFF_ERROR 4U`
Bus off condition reached.
- `#define CAN_FRAMING_ERROR 8U`
Framing error of some kind on the CAN bus.
- `#define CAN_OVERFLOW_ERROR 16U`
Overflow in receive queue.

CAN configuration options

- `#define CAN_USE_SLEEP_MODE TRUE`
Sleep mode related APIs inclusion switch.
- `#define CAN_ENFORCE_USE_CALLBACKS FALSE`
Enforces the driver to use direct callbacks rather than OSAL events.

Macro Functions

- `#define CAN_MAILBOX_TO_MASK(mbx)`
Converts a mailbox index to a bit mask.
- `#define canTransmit`(`canp`, `mailbox`, `ctfp`, `timeout`)

- Legacy name for `canTransmitTimeout()`.
- #define `canReceive`(canp, mailbox, crfp, timeout)
Legacy name for `canReceiveTimeout()`.

Low level driver helper macros

- #define `_can_tx_empty_isr`(canp, flags)
TX mailbox empty event.
- #define `_can_rx_full_isr`(canp, flags)
RX mailbox empty full event.
- #define `_can_wakeup_isr`(canp)
Wakeup event.
- #define `_can_error_isr`(canp, flags)
Error event.

Enumerations

- enum `canstate_t` {
CAN_UNINIT = 0 , CAN_STOP = 1 , CAN_STARTING = 2 , CAN_STOPPING = 3 ,
CAN_READY = 4 , CAN_SLEEP = 5 }
Driver state machine possible states.

Functions

- void `canInit` (void)
CAN Driver initialization.
- void `canObjectInit` (CANDriver *canp)
Initializes the standard part of a `CANDriver` structure.
- `msg_t` `canStart` (CANDriver *canp, const CANConfig *config)
Configures and activates the CAN peripheral.
- void `canStop` (CANDriver *canp)
Deactivates the CAN peripheral.
- bool `canTryTransmitI` (CANDriver *canp, canmbx_t mailbox, const CANTxFrame *ctfp)
Can frame transmission attempt.
- bool `canTryReceiveI` (CANDriver *canp, canmbx_t mailbox, CANRxFrame *crfp)
Can frame receive attempt.
- void `canTryAbortX` (CANDriver *canp, canmbx_t mailbox)
Tries to abort an ongoing transmission.
- `msg_t` `canTransmitTimeout` (CANDriver *canp, canmbx_t mailbox, const CANTxFrame *ctfp, sysinterval_t timeout)
Can frame transmission.
- `msg_t` `canReceiveTimeout` (CANDriver *canp, canmbx_t mailbox, CANRxFrame *crfp, sysinterval_t timeout)
Can frame receive.

9.49.1 Detailed Description

CAN Driver macros and structures.

9.50 hal_channels.h File Reference

I/O channels access.

Data Structures

- struct [BaseChannelVMT](#)
BaseChannel virtual methods table.
- struct [BaseChannel](#)
Base channel class.
- struct [BaseAsynchronousChannelVMT](#)
BaseAsynchronousChannel virtual methods table.
- struct [BaseAsynchronousChannel](#)
Base asynchronous channel class.

Macros

- #define [_base_channel_methods](#)
BaseChannel specific methods.
- #define [_base_channel_data](#) [_base_sequential_stream_data](#)
BaseChannel specific data.
- #define [_base_asynchronous_channel_methods](#) [_base_channel_methods](#) \
BaseAsynchronousChannel specific methods.
- #define [_base_asynchronous_channel_data](#)
BaseAsynchronousChannel specific data.

Default control operation codes.

- #define [CHN_CTL_INVALID](#) 0
Invalid operation code.
- #define [CHN_CTL_NOP](#) 1
Does nothing.
- #define [CHN_CTL_TX_WAIT](#) 2
Wait for TX completion.

Macro Functions (BaseChannel)

- #define [chnPutTimeout](#)(ip, b, time)
Channel blocking byte write with timeout.
- #define [chnGetTimeout](#)(ip, time)
Channel blocking byte read with timeout.
- #define [chnWrite](#)(ip, bp, n)
Channel blocking write.
- #define [chnWriteTimeout](#)(ip, bp, n, time)
Channel blocking write with timeout.
- #define [chnRead](#)(ip, bp, n)
Channel blocking read.
- #define [chnReadTimeout](#)(ip, bp, n, time)
Channel blocking read with timeout.
- #define [chnControl](#)(ip, operation, arg)
Control operation on a channel.

I/O status flags added to the event listener

- #define `CHN_NO_ERROR` (`eventflags_t`)0
No pending conditions.
- #define `CHN_CONNECTED` (`eventflags_t`)1
Connection happened.
- #define `CHN_DISCONNECTED` (`eventflags_t`)2
Disconnection happened.
- #define `CHN_INPUT_AVAILABLE` (`eventflags_t`)4
Data available in the input queue.
- #define `CHN_OUTPUT_EMPTY` (`eventflags_t`)8
Output queue empty.
- #define `CHN_TRANSMISSION_END` (`eventflags_t`)16
Transmission end.
- #define `CHN_PARITY_ERROR` (`eventflags_t`)32
Parity error.
- #define `CHN_FRAMING_ERROR` (`eventflags_t`)64
Framing error.
- #define `CHN_NOISE_ERROR` (`eventflags_t`)128
Line noise error.
- #define `CHN_OVERRUN_ERROR` (`eventflags_t`)256
Overflow error.
- #define `CHN_IDLE_DETECTED` (`eventflags_t`)512
RX line idle.
- #define `CHN_BREAK_DETECTED` (`eventflags_t`)1024
LIN Break.
- #define `CHN_BUFFER_FULL_ERROR` (`eventflags_t`)2048

Macro Functions (BaseAsynchronousChannel)

- #define `chnGetEventSource`(ip)
Returns the I/O condition event source.
- #define `chnAddFlagsI`(ip, flags)
Adds status flags to the listeners's flags mask.

9.50.1 Detailed Description

I/O channels access.

This header defines an abstract interface useful to access generic I/O serial devices in a standardized way.

9.51 hal_crypto.h File Reference

Cryptographic Driver macros and structures.

```
#include "hal_crypto_llld.h"
#include "hal_crypto_fallback.h"
```

Data Structures

- struct [SHA1Context](#)
Type of a SHA1 context.
- struct [SHA256Context](#)
Type of a SHA256 context.
- struct [SHA512Context](#)
Type of a SHA512 context.
- struct [HMACSHA256Context](#)
Type of a HMAC_SHA256 context.
- struct [HMACSHA512Context](#)
Type of a HMAC_SHA512 context.

Macros

- #define [HAL_CRY_USE_FALLBACK FALSE](#)
Enables the SW fall-back of the cryptographic driver.
- #define [HAL_CRY_ENFORCE_FALLBACK FALSE](#)
Makes the driver forcibly use the fall-back implementations.
- #define [HAL_CRY_USE_FALLBACK TRUE](#)
Enables the SW fall-back of the cryptographic driver.

Typedefs

- typedef size_t [bitsize_t](#)
Size, in bits, of a crypto field or message.

Enumerations

- enum [crystate_t](#) { [CRY_UNINIT](#) = 0 , [CRY_STOP](#) = 1 , [CRY_READY](#) = 2 }
Driver state machine possible states.
- enum [cryerror_t](#) {
[CRY_NOERROR](#) = 0 , [CRY_ERR_INV_ALGO](#) = 1 , [CRY_ERR_INV_KEY_SIZE](#) = 2 , [CRY_ERR_INV_KEY_TYPE](#) = 3 ,
[CRY_ERR_INV_KEY_ID](#) = 4 , [CRY_ERR_AUTH_FAILED](#) = 5 , [CRY_ERR_OP_FAILURE](#) = 6 }
Driver error codes.
- enum [cryalgorithm_t](#) { [cry_algo_none](#) = 0 , [cry_algo_aes](#) , [cry_algo_des](#) , [cry_algo_hmac](#) }
Type of an algorithm identifier.

Functions

- void [cryInit](#) (void)
Cryptographic Driver initialization.
- void [cryObjectInit](#) (CRYDriver *cryp)
Initializes the standard part of a CRYDriver structure.
- [msg_t](#) [cryStart](#) (CRYDriver *cryp, const CRYConfig *config)
Configures and activates the cryptographic peripheral.
- void [cryStop](#) (CRYDriver *cryp)
Deactivates the cryptographic peripheral.
- [cryerror_t](#) [cryLoadAESTransientKey](#) (CRYDriver *cryp, size_t size, const uint8_t *keyp)
Initializes the AES transient key.
- [cryerror_t](#) [cryEncryptAES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Encryption of a single block using AES.
- [cryerror_t](#) [cryDecryptAES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Decryption of a single block using AES.
- [cryerror_t](#) [cryEncryptAES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Encryption operation using AES-ECB.
- [cryerror_t](#) [cryDecryptAES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Decryption operation using AES-ECB.
- [cryerror_t](#) [cryEncryptAES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CBC.
- [cryerror_t](#) [cryDecryptAES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CBC.
- [cryerror_t](#) [cryEncryptAES_CFB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CFB.
- [cryerror_t](#) [cryDecryptAES_CFB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CFB.
- [cryerror_t](#) [cryEncryptAES_CTR](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CTR.
- [cryerror_t](#) [cryDecryptAES_CTR](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CTR.
- [cryerror_t](#) [cryEncryptAES_GCM](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, uint8_t *tag_out)
Encryption operation using AES-GCM.
- [cryerror_t](#) [cryDecryptAES_GCM](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, const uint8_t *tag_in)
Decryption operation using AES-GCM.
- [cryerror_t](#) [cryLoadDESTransientKey](#) (CRYDriver *cryp, size_t size, const uint8_t *keyp)
Initializes the DES transient key.
- [cryerror_t](#) [cryEncryptDES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Encryption of a single block using (T)DES.
- [cryerror_t](#) [cryDecryptDES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)

Decryption of a single block using (T)DES.

- `cyerror_t cryEncryptDES_ECB (CRYDriver *cryp, crykey_t key_id, size_t size, const uint8_t *in, uint8_t *out)`

Encryption operation using (T)DES-ECB.

- `cyerror_t cryDecryptDES_ECB (CRYDriver *cryp, crykey_t key_id, size_t size, const uint8_t *in, uint8_t *out)`

Decryption operation using (T)DES-ECB.

- `cyerror_t cryEncryptDES_CBC (CRYDriver *cryp, crykey_t key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)`

Encryption operation using (T)DES-CBC.

- `cyerror_t cryDecryptDES_CBC (CRYDriver *cryp, crykey_t key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)`

Decryption operation using (T)DES-CBC.

- `cyerror_t crySHA1Init (CRYDriver *cryp, SHA1Context *sha1ctxp)`

Hash initialization using SHA1.

- `cyerror_t crySHA1Update (CRYDriver *cryp, SHA1Context *sha1ctxp, size_t size, const uint8_t *in)`

Hash update using SHA1.

- `cyerror_t crySHA1Final (CRYDriver *cryp, SHA1Context *sha1ctxp, uint8_t *out)`

Hash finalization using SHA1.

- `cyerror_t crySHA256Init (CRYDriver *cryp, SHA256Context *sha256ctxp)`

Hash initialization using SHA256.

- `cyerror_t crySHA256Update (CRYDriver *cryp, SHA256Context *sha256ctxp, size_t size, const uint8_t *in)`

Hash update using SHA256.

- `cyerror_t crySHA256Final (CRYDriver *cryp, SHA256Context *sha256ctxp, uint8_t *out)`

Hash finalization using SHA256.

- `cyerror_t crySHA512Init (CRYDriver *cryp, SHA512Context *sha512ctxp)`

Hash initialization using SHA512.

- `cyerror_t crySHA512Update (CRYDriver *cryp, SHA512Context *sha512ctxp, size_t size, const uint8_t *in)`

Hash update using SHA512.

- `cyerror_t crySHA512Final (CRYDriver *cryp, SHA512Context *sha512ctxp, uint8_t *out)`

Hash finalization using SHA512.

- `cyerror_t cryLoadHMACTransientKey (CRYDriver *cryp, size_t size, const uint8_t *keyp)`

Initializes the HMAC transient key.

- `cyerror_t cryHMACSHA256Init (CRYDriver *cryp, HMACSHA256Context *hmacsha256ctxp)`

Hash initialization using HMAC_SHA256.

- `cyerror_t cryHMACSHA256Update (CRYDriver *cryp, HMACSHA256Context *hmacsha256ctxp, size_t size, const uint8_t *in)`

Hash update using HMAC.

- `cyerror_t cryHMACSHA256Final (CRYDriver *cryp, HMACSHA256Context *hmacsha256ctxp, uint8_t *out)`

Hash finalization using HMAC.

- `cyerror_t cryHMACSHA512Init (CRYDriver *cryp, HMACSHA512Context *hmacsha512ctxp)`

Hash initialization using HMAC_SHA512.

- `cyerror_t cryHMACSHA512Update (CRYDriver *cryp, HMACSHA512Context *hmacsha512ctxp, size_t size, const uint8_t *in)`

Hash update using HMAC.

- `cyerror_t cryHMACSHA512Final (CRYDriver *cryp, HMACSHA512Context *hmacsha512ctxp, uint8_t *out)`

Hash finalization using HMAC.

9.51.1 Detailed Description

Cryptographic Driver macros and structures.

9.52 hal_dac.h File Reference

DAC Driver macros and structures.

```
#include "hal_dac_lld.h"
```

Data Structures

- struct [hal_dac_conversion_group](#)
DAC Conversion group structure.
- struct [hal_dac_config](#)
Driver configuration structure.
- struct [hal_dac_driver](#)
Structure representing a DAC driver.

Macros

DAC configuration options

- #define [DAC_USE_SYNCHRONIZATION FALSE](#)
Support for thread synchronization API.
- #define [DAC_USE_MUTUAL_EXCLUSION TRUE](#)
Enables the [dacAcquireBus\(\)](#) and [dacReleaseBus\(\)](#) APIs.

Low level driver helper macros

- #define [dacsBufferComplete\(dacp\)](#)
Buffer state.
- #define [_dac_wait_s\(dacp\)](#)
Waits for operation completion.
- #define [_dac_reset_i\(dacp\)](#)
Resumes a thread waiting for a conversion completion.
- #define [_dac_reset_s\(dacp\)](#)
Resumes a thread waiting for a conversion completion.
- #define [_dac_wakeup_isr\(dacp\)](#)
Wakes up the waiting thread.
- #define [_dac_timeout_isr\(dacp\)](#)
Wakes up the waiting thread with a timeout message.
- #define [_dac_isr_half_code\(dacp\)](#)
Common ISR code, half buffer event.
- #define [_dac_isr_full_code\(dacp\)](#)
Common ISR code, full buffer event.
- #define [_dac_isr_error_code\(dacp, err\)](#)
Common ISR code, error event.

Typedefs

- typedef struct [hal_dac_driver](#) DACDriver
Type of a structure representing an DAC driver.
- typedef struct [hal_dac_config](#) DACConfig
Type of a structure representing an DAC driver configuration.
- typedef struct [hal_dac_conversion_group](#) DACConversionGroup
Type of a DAC conversion group.
- typedef void(* [daccallback_t](#)) (DACDriver *dacp)
DAC notification callback type.
- typedef void(* [dacerrorcallback_t](#)) (DACDriver *dacp, [dacerror_t](#) err)
DAC error callback type.

Enumerations

- enum [dacstate_t](#) {
[DAC_UNINIT](#) = 0 , [DAC_STOP](#) = 1 , [DAC_READY](#) = 2 , [DAC_ACTIVE](#) = 3 ,
[DAC_COMPLETE](#) = 4 , [DAC_ERROR](#) = 5 }

Driver state machine possible states.

Functions

- void [dacInit](#) (void)
DAC Driver initialization.
- void [dacObjectInit](#) ([DACDriver](#) *daccp)
Initializes the standard part of a [DACDriver](#) structure.
- [msg_t](#) [dacStart](#) ([DACDriver](#) *daccp, const [DACConfig](#) *config)
Configures and activates the DAC peripheral.
- void [dacStop](#) ([DACDriver](#) *daccp)
Deactivates the DAC peripheral.
- [msg_t](#) [dacPutChannelX](#) ([DACDriver](#) *daccp, [dacchannel_t](#) channel, [dacsample_t](#) sample)
Outputs a value directly on a DAC channel.
- [msg_t](#) [dacStartConversion](#) ([DACDriver](#) *daccp, const [DACConversionGroup](#) *grpp, [dacsample_t](#) *samples, [size_t](#) depth)
Starts a DAC conversion.
- [msg_t](#) [dacStartConversionI](#) ([DACDriver](#) *daccp, const [DACConversionGroup](#) *grpp, [dacsample_t](#) *samples, [size_t](#) depth)
Starts a DAC conversion.
- void [dacStopConversion](#) ([DACDriver](#) *daccp)
Stops an ongoing conversion.
- void [dacStopConversionI](#) ([DACDriver](#) *daccp)
Stops an ongoing conversion.

9.52.1 Detailed Description

DAC Driver macros and structures.

9.53 hal_efl.h File Reference

Embedded Flash Driver macros and structures.

```
#include "hal_efl_lld.h"
```

Data Structures

- struct [EFlashDriverVMT](#)
EFlash virtual methods table.
- struct [hal_efl_config](#)
Type of a structure representing a flash driver configuration.
- struct [hal_efl_driver](#)
Structure representing an embedded flash driver.

Macros

- `#define EFL_USE_MUTUAL_EXCLUSION TRUE`
Enables the `eflAcquireUse()` and `eflReleaseUse()` APIs.
- `#define _efl_flash_methods_alone`
EFlashDriver specific methods.
- `#define _efl_flash_methods`
EFlashDriver specific methods with inherited ones.
- `#define _efl_driver_data`
EFlashDriver specific data.

Typedefs

- `typedef struct hal_efl_driver EFlashDriver`
Type of external flash driver class.
- `typedef struct hal_efl_config EFlashConfig`
Type of a structure representing a flash driver configuration.

Functions

- `void eflInit (void)`
Embedded Flash Driver initialization.
- `void eflObjectInit (EFlashDriver *eflp)`
Initializes a generic EFlashDriver object.
- `msg_t eflStart (EFlashDriver *eflp, const EFlashConfig *config)`
Configures and starts the driver.
- `void eflStop (EFlashDriver *eflp)`
Stops the driver.

9.53.1 Detailed Description

Embedded Flash Driver macros and structures.

9.54 hal_files.h File Reference

Data files.

Data Structures

- `struct FileStreamVMT`
FileStream virtual methods table.
- `struct FileStream`
Base file stream class.

Macros

- `#define _file_stream_methods`
FileStream specific methods.
- `#define _file_stream_data _base_sequential_stream_data`
FileStream specific data.

Files return codes

- `#define FILE_OK STM_OK`
No error return code.
- `#define FILE_ERROR STM_TIMEOUT`
Error code from the file stream methods.
- `#define FILE_EOF STM_RESET`
End-of-file condition for file get/put methods.

Macro Functions (FileStream)

- `#define fileStreamWrite(ip, bp, n)`
File stream write.
- `#define fileStreamRead(ip, bp, n)`
File stream read.
- `#define fileStreamPut(ip, b)`
File stream blocking byte write.
- `#define fileStreamGet(ip)`
File stream blocking byte read.
- `#define fileStreamClose(ip)`
File Stream close.
- `#define fileStreamGetError(ip)`
Returns an implementation dependent error code.
- `#define fileStreamGetSize(ip, offset)`
Returns the current file size.
- `#define fileStreamGetPosition(ip, offset)`
Returns the current file pointer position.
- `#define fileStreamSetPosition(ip, offset)`
Moves the file current pointer to an absolute position.

Typedefs

- `typedef uint32_t fileoffset_t`
File offset type.

9.54.1 Detailed Description

Data files.

This header defines abstract interfaces useful to access generic data files in a standardized way.

9.55 hal_flash.h File Reference

Generic flash driver class header.

Data Structures

- struct [flash_sector_descriptor_t](#)
Flash sector descriptor.
- struct [flash_descriptor_t](#)
Type of a flash device descriptor.
- struct [BaseFlashVMT](#)
BaseFlash virtual methods table.
- struct [BaseFlash](#)
Base flash class.

Macros

- #define [_base_flash_methods_alone](#)
BaseFlash specific methods.
- #define [_base_flash_methods](#)
BaseFlash specific methods with inherited ones.
- #define [_base_flash_data](#)
BaseFlash specific data.

Flash attributes

- #define [FLASH_ATTR_ERASED_IS_ONE](#) 0x00000001U
Defines one as the erased bit state.
- #define [FLASH_ATTR_MEMORY_MAPPED](#) 0x00000002U
The memory is accessible in a memory mapped mode.
- #define [FLASH_ATTR_REWRITABLE](#) 0x00000004U
Programmed pages can be programmed again.
- #define [FLASH_ATTR_ECC_CAPABLE](#) 0x00000008U
The memory is protected by an ECC mechanism.
- #define [FLASH_ATTR_ECC_ZERO_LINE_CAPABLE](#) 0x00000010U
The device is able to overwrite zero to a line.
- #define [FLASH_ATTR_SUSPEND_ERASE_CAPABLE](#) 0x00000020U
The device is able to suspend erase operations.

Macro Functions (BaseFlash)

- #define [getBaseFlash\(ip\)](#)
Instance getter.
- #define [flashGetDescriptor\(ip\)](#)
Gets the flash descriptor structure.
- #define [flashRead\(ip, offset, n, rp\)](#)
Read operation.
- #define [flashProgram\(ip, offset, n, pp\)](#)
Program operation.
- #define [flashStartEraseAll\(ip\)](#)
Starts a whole-device erase operation.
- #define [flashStartEraseSector\(ip, sector\)](#)
Starts an sector erase operation.
- #define [flashQueryErase\(ip, msec\)](#)
Queries the driver for erase operation progress.
- #define [flashVerifyErase\(ip, sector\)](#)
Returns the erase state of a sector.
- #define [flashAcquireExclusive\(ip\)](#)
Acquires exclusive access to flash.
- #define [flashReleaseExclusive\(ip\)](#)
Releases exclusive access to flash.

Typedefs

- typedef uint32_t `flash_offset_t`
Type of a flash offset.
- typedef uint32_t `flash_sector_t`
Type of a flash sector number.

Enumerations

- enum `flash_state_t` {
 `FLASH_UNINIT` = 0 , `FLASH_STOP` = 1 , `FLASH_READY` = 2 , `FLASH_READ` = 3 ,
 `FLASH_PGM` = 4 , `FLASH_ERASE` = 5 }
Driver state machine possible states.
- enum `flash_error_t` {
 `FLASH_NO_ERROR` = 0 , `FLASH_BUSY_ERASING` = 1 , `FLASH_ERROR_READ` = 2 , `FLASH_ERROR_PROGRAM`
 = 3 ,
 `FLASH_ERROR_ERASE` = 4 , `FLASH_ERROR_VERIFY` = 5 , `FLASH_ERROR_HW_FAILURE` = 6 ,
 `FLASH_ERROR_UNIMPLEMENTED` = 7 }
Type of a flash error code.

Functions

- `flash_error_t` `flashWaitErase` (`BaseFlash` *devp)
Waits until the current erase operation is finished.
- `flash_offset_t` `flashGetSectorOffset` (`BaseFlash` *devp, `flash_sector_t` sector)
Returns the offset of a sector.
- uint32_t `flashGetSectorSize` (`BaseFlash` *devp, `flash_sector_t` sector)
Returns the size of a sector.
- `flash_sector_t` `flashGetOffsetSector` (`BaseFlash` *devp, `flash_offset_t` offset)
Returns the sector of an offset.
- void * `flashGetOffsetAddress` (`BaseFlash` *devp, `flash_offset_t` offset)
Get absolute address from offset.
- `flash_offset_t` `flashGetAddressOffset` (`BaseFlash` *devp, void *addr)
Get offset from absolute address.

9.55.1 Detailed Description

Generic flash driver class header.

9.56 hal_gpt.h File Reference

GPT Driver macros and structures.

```
#include "hal_gpt_lld.h"
```

Macros

- #define [gptChangeIntervall](#)(gptp, interval)
Changes the interval of GPT peripheral.
- #define [gptGetIntervalX](#)(gptp)
Returns the interval of GPT peripheral.
- #define [gptGetCounterX](#)(gptp)
Returns the counter value of GPT peripheral.
- #define [_gpt_isr_invoke_cb](#)(gptp)
Common ISR code, GPT period event.

Typedefs

- typedef struct GPTDriver [GPTDriver](#)
Type of a structure representing a GPT driver.
- typedef void(* [gptcallback_t](#)) ([GPTDriver](#) *gptp)
GPT notification callback type.

Enumerations

- enum [gptstate_t](#) {
 [GPT_UNINIT](#) = 0 , [GPT_STOP](#) = 1 , [GPT_READY](#) = 2 , [GPT_CONTINUOUS](#) = 3 ,
 [GPT_ONESHOT](#) = 4 }
Driver state machine possible states.

Functions

- void [gptInit](#) (void)
GPT Driver initialization.
- void [gptObjectInit](#) ([GPTDriver](#) *gptp)
Initializes the standard part of a [GPTDriver](#) structure.
- [msg_t](#) [gptStart](#) ([GPTDriver](#) *gptp, const [GPTConfig](#) *config)
Configures and activates the GPT peripheral.
- void [gptStop](#) ([GPTDriver](#) *gptp)
Deactivates the GPT peripheral.
- void [gptStartContinuous](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in continuous mode.
- void [gptStartContinuousl](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in continuous mode.
- void [gptChangeInterval](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Changes the interval of GPT peripheral.
- void [gptStartOneShot](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in one shot mode.
- void [gptStartOneShotl](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in one shot mode.
- void [gptStopTimer](#) ([GPTDriver](#) *gptp)
Stops the timer.
- void [gptStopTimerl](#) ([GPTDriver](#) *gptp)
Stops the timer.
- void [gptPolledDelay](#) ([GPTDriver](#) *gptp, [gptcnt_t](#) interval)
Starts the timer in one shot mode and waits for completion.

9.56.1 Detailed Description

GPT Driver macros and structures.

9.57 hal_i2c.h File Reference

I2C Driver macros and structures.

```
#include "hal_i2c_lld.h"
```

Macros

- `#define I2C_USE_MUTUAL_EXCLUSION TRUE`
Enables the mutual exclusion APIs on the I2C bus.
- `#define I2C_ENABLE_SLAVE_MODE FALSE`
Slave mode API enable switch.
- `#define I2C_SUPPORTS_SLAVE_MODE FALSE`

I2C bus error conditions

- `#define I2C_NO_ERROR 0x00`
No error.
- `#define I2C_BUS_ERROR 0x01`
Bus Error.
- `#define I2C_ARBITRATION_LOST 0x02`
Arbitration Lost.
- `#define I2C_ACK_FAILURE 0x04`
Acknowledge Failure.
- `#define I2C_OVERRUN 0x08`
Overrun/Underrun.
- `#define I2C_PEC_ERROR 0x10`
PEC Error in reception.
- `#define I2C_TIMEOUT 0x20`
Hardware timeout.
- `#define I2C_SMB_ALERT 0x40`
SMBus Alert.

Macro Functions

- `#define _i2c_wakeup_isr(i2cp)`
Wakes up the waiting thread notifying no errors.
- `#define _i2c_wakeup_error_isr(i2cp)`
Wakes up the waiting thread notifying errors.
- `#define i2cMasterTransmit(i2cp, addr, txbuf, txbytes, rxbuf, rxbytes)`
Wrap i2cMasterTransmitTimeout function with TIME_INFINITE timeout.
- `#define i2cMasterReceive(i2cp, addr, rxbuf, rxbytes)`
Wrap i2cMasterReceiveTimeout function with TIME_INFINITE timeout.
- `#define i2cSlavelsAnswerRequired(i2cp)`
Answer required.

Enumerations

- enum `i2cstate_t` {
`I2C_UNINIT` = 0 , `I2C_STOP` = 1 , `I2C_READY` = 2 , `I2C_ACTIVE_TX` = 3 ,
`I2C_ACTIVE_RX` = 4 , `I2C_LOCKED` = 5 }

Driver state machine possible states.

Functions

- void `i2cInit` (void)
I2C Driver initialization.
- void `i2cObjectInit` (`I2CDriver` *i2cp)
Initializes the standard part of a `I2CDriver` structure.
- `msg_t` `i2cStart` (`I2CDriver` *i2cp, const `I2CConfig` *config)
Configures and activates the I2C peripheral.
- void `i2cStop` (`I2CDriver` *i2cp)
Deactivates the I2C peripheral.
- `i2cflags_t` `i2cGetErrors` (`I2CDriver` *i2cp)
Returns the errors mask associated to the previous operation.
- `msg_t` `i2cMasterTransmitTimeout` (`I2CDriver` *i2cp, `i2caddr_t` addr, const `uint8_t` *txbuf, `size_t` txbytes, `uint8_t` *rxbuf, `size_t` rxbytes, `sysinterval_t` timeout)
Sends data via the I2C bus.
- `msg_t` `i2cMasterReceiveTimeout` (`I2CDriver` *i2cp, `i2caddr_t` addr, `uint8_t` *rxbuf, `size_t` rxbytes, `sysinterval_t` timeout)
Receives data from the I2C bus.
- void `i2cAcquireBus` (`I2CDriver` *i2cp)
Gains exclusive access to the I2C bus.
- void `i2cReleaseBus` (`I2CDriver` *i2cp)
Releases exclusive access to the I2C bus.
- `msg_t` `i2cSlaveMatchAddress` (`I2CDriver` *i2cp, `i2caddr_t` addr)
Listen I2C bus for address match.
- `msg_t` `i2cSlaveReceiveTimeout` (`I2CDriver` *i2cp, `uint8_t` *rxbuf, `size_t` rxbytes, `sysinterval_t` timeout)
Receive data via the I2C bus as slave and call handler.
- `msg_t` `i2cSlaveTransmitTimeout` (`I2CDriver` *i2cp, const `uint8_t` *txbuf, `size_t` txbytes, `sysinterval_t` timeout)
Transmits data via the I2C bus as slave.

9.57.1 Detailed Description

I2C Driver macros and structures.

9.58 hal_i2s.h File Reference

I2S Driver macros and structures.

```
#include "hal_i2s_ll.h"
```

Data Structures

- struct [hal_i2s_driver](#)
Structure representing an I2S driver.
- struct [hal_i2s_config](#)
Driver configuration structure.

Macros

I2S modes

- #define [I2S_MODE_SLAVE](#) 0
- #define [I2S_MODE_MASTER](#) 1

Macro Functions

- #define [i2sIsBufferComplete](#)(i2sp)
Buffer state.
- #define [i2sStartExchange](#)(i2sp)
Starts a I2S data exchange.
- #define [i2sStopExchange](#)(i2sp)
Stops the ongoing data exchange.
- #define [_i2s_isr_half_code](#)(i2sp)
Common ISR code, half buffer event.
- #define [_i2s_isr_full_code](#)(i2sp)
Common ISR code.

Typedefs

- typedef struct [hal_i2s_driver](#) [I2SDriver](#)
Type of a structure representing an I2S driver.
- typedef struct [hal_i2s_config](#) [I2SConfig](#)
Type of a structure representing an I2S driver configuration.
- typedef void(* [i2scallback_t](#)) ([I2SDriver](#) *i2sp)
I2S notification callback type.

Enumerations

- enum [i2sstate_t](#) {
 [I2S_UNINIT](#) = 0 , [I2S_STOP](#) = 1 , [I2S_READY](#) = 2 , [I2S_ACTIVE](#) = 3 ,
 [I2S_COMPLETE](#) = 4 }
Driver state machine possible states.

Functions

- void [i2sInit](#) (void)
I2S Driver initialization.
- void [i2sObjectInit](#) ([I2SDriver](#) *i2sp)
Initializes the standard part of a [I2SDriver](#) structure.
- [msg_t](#) [i2sStart](#) ([I2SDriver](#) *i2sp, const [I2SConfig](#) *config)
Configures and activates the I2S peripheral.
- void [i2sStop](#) ([I2SDriver](#) *i2sp)
Deactivates the I2S peripheral.
- void [i2sStartExchange](#) ([I2SDriver](#) *i2sp)
Starts a I2S data exchange.
- void [i2sStopExchange](#) ([I2SDriver](#) *i2sp)
Stops the ongoing data exchange.

9.58.1 Detailed Description

I2S Driver macros and structures.

9.59 hal_icu.h File Reference

ICU Driver macros and structures.

```
#include "hal_icu_lld.h"
```

Macros

- `#define ICU_USE_OVERFLOW_SCALING FALSE`

Macro Functions

- `#define icuStartCaptureI(icup)`
Starts the input capture.
- `#define icuStopCaptureI(icup)`
Stops the input capture.
- `#define icuEnableNotificationsI(icup)`
Enables notifications.
- `#define icuDisableNotificationsI(icup)`
Disables notifications.
- `#define icuAreNotificationsEnabledX(icup)`
Check on notifications status.
- `#define icuGetWidthX(icup)`
Returns the width of the latest pulse.
- `#define icuGetPeriodX(icup)`
Returns the width of the latest cycle.

Low level driver helper macros

- `#define _icu_isr_invoke_width_cb(icup)`
Common ISR code, ICU width event.
- `#define _icu_isr_invoke_period_cb(icup)`
Common ISR code, ICU period event.
- `#define _icu_isr_invoke_overflow_cb(icup)`
Common ISR code, ICU timer overflow event.

Typedefs

- `typedef struct ICUDriver ICUDriver`
Type of a structure representing an ICU driver.
- `typedef void(* icucallback_t) (ICUDriver *icup)`
ICU notification callback type.

Enumerations

- enum `icustate_t` {
`ICU_UNINIT` = 0 , `ICU_STOP` = 1 , `ICU_READY` = 2 , `ICU_WAITING` = 3 ,
`ICU_ACTIVE` = 4 }

Driver state machine possible states.

Functions

- void `iculnit` (void)
ICU Driver initialization.
- void `icuObjectInit` (`ICUDriver` *icup)
Initializes the standard part of a `ICUDriver` structure.
- `msg_t` `icuStart` (`ICUDriver` *icup, const `ICUConfig` *config)
Configures and activates the ICU peripheral.
- void `icuStop` (`ICUDriver` *icup)
Deactivates the ICU peripheral.
- void `icuStartCapture` (`ICUDriver` *icup)
Starts the input capture.
- bool `icuWaitCapture` (`ICUDriver` *icup)
Waits for a completed capture.
- void `icuStopCapture` (`ICUDriver` *icup)
Stops the input capture.
- void `icuEnableNotifications` (`ICUDriver` *icup)
Enables notifications.
- void `icuDisableNotifications` (`ICUDriver` *icup)
Disables notifications.

9.59.1 Detailed Description

ICU Driver macros and structures.

9.60 hal_ioblock.h File Reference

I/O block devices access.

Data Structures

- struct `BlockDeviceInfo`
Block device info.
- struct `BaseBlockDeviceVMT`
`BaseBlockDevice` virtual methods table.
- struct `BaseBlockDevice`
Base block device class.

Macros

- #define `_base_block_device_methods`
BaseBlockDevice specific methods.
- #define `_base_block_device_data`
BaseBlockDevice specific data.

Macro Functions (BaseBlockDevice)

- #define `blkGetDriverState(ip)`
Returns the driver state.
- #define `blkIsTransferring(ip)`
Determines if the device is transferring data.
- #define `blkIsInserted(ip)`
Returns the media insertion status.
- #define `blkIsWriteProtected(ip)`
Returns the media write protection status.
- #define `blkConnect(ip)`
Performs the initialization procedure on the block device.
- #define `blkDisconnect(ip)`
Terminates operations on the block device.
- #define `blkRead(ip, startblk, buf, n)`
Reads one or more blocks.
- #define `blkWrite(ip, startblk, buf, n)`
Writes one or more blocks.
- #define `blkSync(ip)`
Ensures write synchronization.
- #define `blkGetInfo(ip, bdip)`
Returns a media information structure.

Enumerations

- enum `blkstate_t` {
 `BLK_UNINIT` = 0 , `BLK_STOP` = 1 , `BLK_ACTIVE` = 2 , `BLK_CONNECTING` = 3 ,
 `BLK_DISCONNECTING` = 4 , `BLK_READY` = 5 , `BLK_READING` = 6 , `BLK_WRITING` = 7 ,
 `BLK_SYNCING` = 8 }
Driver state machine possible states.

9.60.1 Detailed Description

I/O block devices access.

This header defines an abstract interface useful to access generic I/O block devices in a standardized way.

9.61 hal_mac.h File Reference

MAC Driver macros and structures.

```
#include "hal_mac_llid.h"
```

Data Structures

- struct [hal_mac_config](#)
Driver configuration structure.
- struct [hal_mac_driver](#)
Structure representing a MAC driver.
- struct [hal_mac_transmit_descriptor](#)
Structure representing a MAC transmit descriptor.
- struct [hal_mac_receive_descriptor](#)
Structure representing a MAC receive descriptor.

Macros

MAC event flags

- #define [MAC_FLAGS_TX](#) (1U << 0)
- #define [MAC_FLAGS_RX](#) (1U << 1)

MAC configuration options

- #define [MAC_USE_ZERO_COPY](#) FALSE
Enables an event sources for incoming packets.
- #define [MAC_USE_EVENTS](#) TRUE
Enables an event sources for incoming packets.

Low level driver helper macros

- #define [__mac_callback](#)(macp)
MAC callback.
- #define [__mac_tx_wakeup](#)(macp)
MAC TX wakeup and event.
- #define [__mac_rx_wakeup](#)(macp)
MAC RX wakeup and event.

Macro Functions

- #define [macSetCallbackX](#)(macp, f)
Associates a callback to the MAC instance.
- #define [macGetEventSource](#)(macp)
Returns the driver events source.
- #define [macGetTransmitDescriptorX](#)(macp, tdp)
Returns a transmission descriptor.
- #define [macReleaseTransmitDescriptorX](#)(tdp)
Releases a transmit descriptor and starts the transmission of the enqueued data as a single frame.
- #define [macGetReceiveDescriptorX](#)(macp, rdp)
Returns a receive descriptor.
- #define [macReleaseReceiveDescriptorX](#)(rdp)
Releases a receive descriptor.
- #define [macWriteTransmitDescriptor](#)(tdp, buf, size)
Writes to a transmit descriptor's stream.
- #define [macReadReceiveDescriptor](#)(rdp, buf, size)
Reads from a receive descriptor's stream.
- #define [macGetNextTransmitBuffer](#)(tdp, size, sizep)
Returns a pointer to the next transmit buffer in the descriptor chain.
- #define [macGetNextReceiveBuffer](#)(rdp, sizep)
Returns a pointer to the next receive buffer in the descriptor chain.

Typedefs

- typedef struct [hal_mac_driver](#) [MACDriver](#)
Type of a structure representing a MAC driver.
- typedef struct [hal_mac_config](#) [MACConfig](#)
Type of structure representing a MAC configuration.
- typedef struct [hal_mac_transmit_descriptor](#) [MACTransmitDescriptor](#)
Type of structure representing a MAC transmit descriptor.
- typedef struct [hal_mac_receive_descriptor](#) [MACReceiveDescriptor](#)
Type of structure representing a MAC receive descriptor.
- typedef void(* [maccb_t](#)) ([MACDriver](#) *macp)
Generic ETH notification callback type.

Enumerations

- enum [macstate_t](#) { [MAC_UNINIT](#) = 0 , [MAC_STOP](#) = 1 , [MAC_ACTIVE](#) = 2 }
Driver state machine possible states.

Functions

- void [macInit](#) (void)
MAC Driver initialization.
- void [macObjectInit](#) ([MACDriver](#) *macp)
Initialize the standard part of a [MACDriver](#) structure.
- [msg_t](#) [macStart](#) ([MACDriver](#) *macp, const [MACConfig](#) *config)
Configures and activates the MAC peripheral.
- void [macStop](#) ([MACDriver](#) *macp)
Deactivates the MAC peripheral.
- [eventflags_t](#) [macGetAndClearEventsI](#) ([MACDriver](#) *macp)
Get and clears MAC event flags.
- [msg_t](#) [macWaitTransmitDescriptor](#) ([MACDriver](#) *macp, [MACTransmitDescriptor](#) *tdp, [sysinterval_t](#) timeout)
Allocates a transmission descriptor.
- void [macReleaseTransmitDescriptor](#) ([MACTransmitDescriptor](#) *tdp)
- [msg_t](#) [macWaitReceiveDescriptor](#) ([MACDriver](#) *macp, [MACReceiveDescriptor](#) *rdp, [sysinterval_t](#) timeout)
Waits for a received frame.
- void [macReleaseReceiveDescriptor](#) ([MACReceiveDescriptor](#) *rdp)
- bool [macPollLinkStatus](#) ([MACDriver](#) *macp)
Updates and returns the link status.

9.61.1 Detailed Description

MAC Driver macros and structures.

9.62 hal_mii.h File Reference

MII macros and structures.

Macros

Generic MII registers

- #define [MII_BMCR](#) 0x00
- #define [MII_BMSR](#) 0x01
- #define [MII_PHYSID1](#) 0x02
- #define [MII_PHYSID2](#) 0x03
- #define [MII_ADVERTISE](#) 0x04
- #define [MII_LPA](#) 0x05
- #define [MII_EXPANSION](#) 0x06
- #define [MII_ANNPTR](#) 0x07
- #define [MII_CTRL1000](#) 0x09
- #define [MII_STAT1000](#) 0x0a
- #define [MII_ESTATUS](#) 0x0f
- #define [MII_PHYSTS](#) 0x10
- #define [MII_MICR](#) 0x11
- #define [MII_DCOUNTER](#) 0x12
- #define [MII_FCSCOUNTER](#) 0x13
- #define [MII_NWAYTEST](#) 0x14
- #define [MII_RERRCOUNTER](#) 0x15
- #define [MII_SREVISION](#) 0x16
- #define [MII_RESV1](#) 0x17
- #define [MII_LBRERROR](#) 0x18
- #define [MII_PHYADDR](#) 0x19
- #define [MII_RESV2](#) 0x1a
- #define [MII_TPISTATUS](#) 0x1b
- #define [MII_NCONFIG](#) 0x1c

Basic mode control register

- #define [BMCR_RESV](#) 0x007f
- #define [BMCR_CTST](#) 0x0080
- #define [BMCR_FULLDPLX](#) 0x0100
- #define [BMCR_ANRESTART](#) 0x0200
- #define [BMCR_ISOLATE](#) 0x0400
- #define [BMCR_PDOWN](#) 0x0800
- #define [BMCR_ANENABLE](#) 0x1000
- #define [BMCR_SPEED100](#) 0x2000
- #define [BMCR_LOOPBACK](#) 0x4000
- #define [BMCR_RESET](#) 0x8000

Basic mode status register

- #define [BMSR_ERCAP](#) 0x0001
- #define [BMSR_JCD](#) 0x0002
- #define [BMSR_LSTATUS](#) 0x0004
- #define [BMSR_ANEGCAPABLE](#) 0x0008
- #define [BMSR_RFAULT](#) 0x0010
- #define [BMSR_ANEGCOMPLETE](#) 0x0020
- #define [BMSR_MFPRESUPPCAP](#) 0x0040
- #define [BMSR_RESV](#) 0x0780
- #define [BMSR_10HALF](#) 0x0800
- #define [BMSR_10FULL](#) 0x1000
- #define [BMSR_100HALF](#) 0x2000
- #define [BMSR_100FULL](#) 0x4000
- #define [BMSR_100BASE4](#) 0x8000

Advertisement control register

- #define [ADVERTISE_SLCT](#) 0x001f

- #define [ADVERTISE_CSMA](#) 0x0001
- #define [ADVERTISE_10HALF](#) 0x0020
- #define [ADVERTISE_10FULL](#) 0x0040
- #define [ADVERTISE_100HALF](#) 0x0080
- #define [ADVERTISE_100FULL](#) 0x0100
- #define [ADVERTISE_100BASE4](#) 0x0200
- #define [ADVERTISE_PAUSE_CAP](#) 0x0400
- #define [ADVERTISE_PAUSE_ASYM](#) 0x0800
- #define [ADVERTISE_RESV](#) 0x1000
- #define [ADVERTISE_RFAULT](#) 0x2000
- #define [ADVERTISE_LPACK](#) 0x4000
- #define [ADVERTISE_NPAGE](#) 0x8000
- #define [ADVERTISE_FULL](#)
- #define [ADVERTISE_ALL](#)

Link partner ability register

- #define [LPA_SLCT](#) 0x001f
- #define [LPA_10HALF](#) 0x0020
- #define [LPA_10FULL](#) 0x0040
- #define [LPA_100HALF](#) 0x0080
- #define [LPA_100FULL](#) 0x0100
- #define [LPA_100BASE4](#) 0x0200
- #define [LPA_PAUSE_CAP](#) 0x0400
- #define [LPA_PAUSE_ASYM](#) 0x0800
- #define [LPA_RESV](#) 0x1000
- #define [LPA_RFAULT](#) 0x2000
- #define [LPA_LPACK](#) 0x4000
- #define [LPA_NPAGE](#) 0x8000
- #define [LPA_DUPLEX](#) ([LPA_10FULL](#) | [LPA_100FULL](#))
- #define [LPA_100](#) ([LPA_100FULL](#) | [LPA_100HALF](#) | [LPA_100BASE4](#))

Expansion register for auto-negotiation

- #define [EXPANSION_NWAY](#) 0x0001
- #define [EXPANSION_LCWP](#) 0x0002
- #define [EXPANSION_ENABLENPAGE](#) 0x0004
- #define [EXPANSION_NPCAPABLE](#) 0x0008
- #define [EXPANSION_MFAULTS](#) 0x0010
- #define [EXPANSION_RESV](#) 0xffe0

N-way test register

- #define [NWAYTEST_RESV1](#) 0x00ff
- #define [NWAYTEST_LOOPBACK](#) 0x0100
- #define [NWAYTEST_RESV2](#) 0xfe00

PHY identifiers

- #define [MII_DM9161_ID](#) 0x0181b8a0
- #define [MII_AM79C875_ID](#) 0x00225540
- #define [MII_KSZ8081_ID](#) 0x00221560
- #define [MII_KS8721_ID](#) 0x00221610
- #define [MII_STE101P_ID](#) 0x00061C50
- #define [MII_DP83848I_ID](#) 0x20005C90
- #define [MII_LAN8710A_ID](#) 0x0007C0F1
- #define [MII_LAN8720_ID](#) 0x0007C0F0
- #define [MII_LAN8742A_ID](#) 0x0007C130
- #define [MII_LAN8740A_ID](#) 0x0007C110

9.62.1 Detailed Description

MIl macros and structures.

9.63 hal_mmc_spi.h File Reference

MMC over SPI driver header.

Data Structures

- struct [mmc_spi_config_t](#)
Type of a MMC/SD over SPI driver configuration structure.
- struct [mmc_spi_driver_vmt](#)
MMCDriver virtual methods table.
- struct [mmc_spi_driver_t](#)
Structure representing a MMC/SD over SPI driver.

Macros

- #define [MMC_CMD0_RETRY](#) 10U
- #define [MMC_CMD1_RETRY](#) 100U
- #define [MMC_ACMD41_RETRY](#) 100U
- #define [MMC_WAIT_DATA](#) 10000U
- #define [MMC_BUFFER_SIZE](#) 16U
Size of the buffer to be supplied to the driver.
- #define [__mmc_driver_methods](#) [_mmcsd_block_device_methods](#)
MMCDriver specific methods.

MMC_SPI configuration options

- #define [MMC_IDLE_TIMEOUT_MS](#) 1000
Timeout before assuming a failure while waiting for card idle.
- #define [MMC_USE_MUTUAL_EXCLUSION](#) TRUE
Mutual exclusion on the SPI bus.

Macro Functions

- #define [mmclsCardInserted](#)(mmcp)
Returns the card insertion status.
- #define [mmclsWriteProtected](#)(mmcp)
Returns the write protect status.

Typedefs

- typedef [mmc_spi_config_t](#) MMCConfig
Legacy name for compatibility.
- typedef [mmc_spi_driver_t](#) MMCDriver
Legacy name for compatibility.

Functions

- void `mmcInit` (void)
MMC over SPI driver initialization.
- void `mmcObjectInit` (MMCDriver *mmcp, uint8_t *buffer)
Initializes an instance.
- msg_t `mmcStart` (MMCDriver *mmcp, const MMCCfg *config)
Configures and activates the MMC peripheral.
- void `mmcStop` (MMCDriver *mmcp)
Disables the MMC peripheral.
- bool `mmcConnect` (MMCDriver *mmcp)
Performs the initialization procedure on the inserted card.
- bool `mmcDisconnect` (MMCDriver *mmcp)
Brings the driver in a state safe for card removal.
- bool `mmcStartSequentialRead` (MMCDriver *mmcp, uint32_t startblk)
Starts a sequential read.
- bool `mmcSequentialRead` (MMCDriver *mmcp, uint8_t *buffer)
Reads a block within a sequential read operation.
- bool `mmcStopSequentialRead` (MMCDriver *mmcp)
Stops a sequential read gracefully.
- bool `mmcStartSequentialWrite` (MMCDriver *mmcp, uint32_t startblk)
Starts a sequential write.
- bool `mmcSequentialWrite` (MMCDriver *mmcp, const uint8_t *buffer)
Writes a block within a sequential write operation.
- bool `mmcStopSequentialWrite` (MMCDriver *mmcp)
Stops a sequential write gracefully.
- bool `mmcSync` (MMCDriver *mmcp)
Waits for card idle condition.
- bool `mmcGetInfo` (MMCDriver *mmcp, BlockDeviceInfo *bdip)
Returns the media info.
- bool `mmcErase` (MMCDriver *mmcp, uint32_t startblk, uint32_t endblk)
Erases blocks.
- bool `mmc_lld_is_card_inserted` (MMCDriver *mmcp)
- bool `mmc_lld_is_write_protected` (MMCDriver *mmcp)

9.63.1 Detailed Description

MMC over SPI driver header.

9.64 hal_mmcsd.h File Reference

MMC/SD cards common header.

Data Structures

- struct [MMCSDBlockDeviceVMT](#)
MMCSDBlockDevice virtual methods table.
- struct [MMCSDBlockDevice](#)
MCC/SD block device class.
- struct [unpacked_sdc_cid_t](#)
Unpacked CID register from SDC.
- struct [unpacked_mmc_cid_t](#)
Unpacked CID register from MMC.
- struct [unpacked_sdc_csd_10_t](#)
Unpacked CSD v1.0 register from SDC.
- struct [unpacked_sdc_csd_20_t](#)
Unpacked CSD v2.0 register from SDC.
- struct [unpacked_mmc_csd_t](#)
Unpacked CSD register from MMC.

Macros

- #define [MMCSD_BLOCK_SIZE](#) 512U
Fixed block size for MMC/SD block devices.
- #define [MMCSD_R1_ERROR_MASK](#) 0xFDFFE008U
Mask of error bits in R1 responses.
- #define [MMCSD_CMD8_PATTERN](#) 0x000001AAU
Fixed pattern for CMD8.
- #define [_mmcsd_block_device_methods](#) [_base_block_device_methods](#)
MMCSDBlockDevice specific methods.
- #define [_mmcsd_block_device_data](#)
MMCSDBlockDevice specific data.

SD/MMC status conditions

- #define [MMCSD_STS_IDLE](#) 0U
- #define [MMCSD_STS_READY](#) 1U
- #define [MMCSD_STS_IDENT](#) 2U
- #define [MMCSD_STS_STBY](#) 3U
- #define [MMCSD_STS_TRAN](#) 4U
- #define [MMCSD_STS_DATA](#) 5U
- #define [MMCSD_STS_RCV](#) 6U
- #define [MMCSD_STS_PRG](#) 7U
- #define [MMCSD_STS_DIS](#) 8U

SD/MMC commands

- #define [MMCSD_CMD_GO_IDLE_STATE](#) 0U
- #define [MMCSD_CMD_INIT](#) 1U
- #define [MMCSD_CMD_ALL_SEND_CID](#) 2U
- #define [MMCSD_CMD_SEND_RELATIVE_ADDR](#) 3U
- #define [MMCSD_CMD_SET_RELATIVE_ADDR](#) [MMCSD_CMD_SEND_RELATIVE_ADDR](#)
- #define [MMCSD_CMD_SET_BUS_WIDTH](#) 6U
- #define [MMCSD_CMD_SWITCH](#) [MMCSD_CMD_SET_BUS_WIDTH](#)
- #define [MMCSD_CMD_SEL_DESEL_CARD](#) 7U
- #define [MMCSD_CMD_SEND_IF_COND](#) 8U
- #define [MMCSD_CMD_SEND_EXT_CSD](#) [MMCSD_CMD_SEND_IF_COND](#)

- #define [MMCSD_CMD_SEND_CSD](#) 9U
- #define [MMCSD_CMD_SEND_CID](#) 10U
- #define [MMCSD_CMD_STOP_TRANSMISSION](#) 12U
- #define [MMCSD_CMD_SEND_STATUS](#) 13U
- #define [MMCSD_CMD_SET_BLOCKLEN](#) 16U
- #define [MMCSD_CMD_READ_SINGLE_BLOCK](#) 17U
- #define [MMCSD_CMD_READ_MULTIPLE_BLOCK](#) 18U
- #define [MMCSD_CMD_SET_BLOCK_COUNT](#) 23U
- #define [MMCSD_CMD_WRITE_BLOCK](#) 24U
- #define [MMCSD_CMD_WRITE_MULTIPLE_BLOCK](#) 25U
- #define [MMCSD_CMD_ERASE_RW_BLK_START](#) 32U
- #define [MMCSD_CMD_ERASE_RW_BLK_END](#) 33U
- #define [MMCSD_CMD_ERASE](#) 38U
- #define [MMCSD_CMD_APP_OP_COND](#) 41U
- #define [MMCSD_CMD_LOCK_UNLOCK](#) 42U
- #define [MMCSD_CMD_APP_CMD](#) 55U
- #define [MMCSD_CMD_READ_OCR](#) 58U

CSD record offsets

- #define [MMCSD_CSD_MMC_CSD_STRUCTURE_SLICE](#) 127U, 126U
- #define [MMCSD_CSD_MMC_SPEC_VERS_SLICE](#) 125U, 122U
- #define [MMCSD_CSD_MMC_TAAC_SLICE](#) 119U, 112U
- #define [MMCSD_CSD_MMC_NSAC_SLICE](#) 111U, 104U
- #define [MMCSD_CSD_MMC_TRAN_SPEED_SLICE](#) 103U, 96U
- #define [MMCSD_CSD_MMC_CCC_SLICE](#) 95U, 84U
- #define [MMCSD_CSD_MMC_READ_BL_LEN_SLICE](#) 83U, 80U
- #define [MMCSD_CSD_MMC_READ_BL_PARTIAL_SLICE](#) 79U, 79U
- #define [MMCSD_CSD_MMC_WRITE_BLK_MISALIGN_SLICE](#) 78U, 78U
- #define [MMCSD_CSD_MMC_READ_BLK_MISALIGN_SLICE](#) 77U, 77U
- #define [MMCSD_CSD_MMC_DSR_IMP_SLICE](#) 76U, 76U
- #define [MMCSD_CSD_MMC_C_SIZE_SLICE](#) 73U, 62U
- #define [MMCSD_CSD_MMC_VDD_R_CURR_MIN_SLICE](#) 61U, 59U
- #define [MMCSD_CSD_MMC_VDD_R_CURR_MAX_SLICE](#) 58U, 56U
- #define [MMCSD_CSD_MMC_VDD_W_CURR_MIN_SLICE](#) 55U, 53U
- #define [MMCSD_CSD_MMC_VDD_W_CURR_MAX_SLICE](#) 52U, 50U
- #define [MMCSD_CSD_MMC_C_SIZE_MULT_SLICE](#) 49U, 47U
- #define [MMCSD_CSD_MMC_ERASE_GRP_SIZE_SLICE](#) 46U, 42U
- #define [MMCSD_CSD_MMC_ERASE_GRP_MULT_SLICE](#) 41U, 37U
- #define [MMCSD_CSD_MMC_WP_GRP_SIZE_SLICE](#) 36U, 32U
- #define [MMCSD_CSD_MMC_WP_GRP_ENABLE_SLICE](#) 31U, 31U
- #define [MMCSD_CSD_MMC_DEFAULT_ECC_SLICE](#) 30U, 29U
- #define [MMCSD_CSD_MMC_R2W_FACTOR_SLICE](#) 28U, 26U
- #define [MMCSD_CSD_MMC_WRITE_BL_LEN_SLICE](#) 25U, 22U
- #define [MMCSD_CSD_MMC_WRITE_BL_PARTIAL_SLICE](#) 21U, 21U
- #define [MMCSD_CSD_MMC_CONTENT_PROT_APP_SLICE](#) 16U, 16U
- #define [MMCSD_CSD_MMC_FILE_FORMAT_GRP_SLICE](#) 15U, 15U
- #define [MMCSD_CSD_MMC_COPY_SLICE](#) 14U, 14U
- #define [MMCSD_CSD_MMC_PERM_WRITE_PROTECT_SLICE](#) 13U, 13U
- #define [MMCSD_CSD_MMC_TMP_WRITE_PROTECT_SLICE](#) 12U, 12U
- #define [MMCSD_CSD_MMC_FILE_FORMAT_SLICE](#) 11U, 10U
- #define [MMCSD_CSD_MMC_ECC_SLICE](#) 9U, 8U
- #define [MMCSD_CSD_MMC_CRC_SLICE](#) 7U, 1U
- #define [MMCSD_CSD_20_CRC_SLICE](#) 7U, 1U
- #define [MMCSD_CSD_20_FILE_FORMAT_SLICE](#) 11U, 10U
- #define [MMCSD_CSD_20_TMP_WRITE_PROTECT_SLICE](#) 12U, 12U
- #define [MMCSD_CSD_20_PERM_WRITE_PROTECT_SLICE](#) 13U, 13U
- #define [MMCSD_CSD_20_COPY_SLICE](#) 14U, 14U
- #define [MMCSD_CSD_20_FILE_FORMAT_GRP_SLICE](#) 15U, 15U
- #define [MMCSD_CSD_20_WRITE_BL_PARTIAL_SLICE](#) 21U, 21U
- #define [MMCSD_CSD_20_WRITE_BL_LEN_SLICE](#) 25U, 12U
- #define [MMCSD_CSD_20_R2W_FACTOR_SLICE](#) 28U, 26U
- #define [MMCSD_CSD_20_WP_GRP_ENABLE_SLICE](#) 31U, 31U

- #define MMCS_CSD_20_WP_GRP_SIZE_SLICE 38U, 32U
- #define MMCS_CSD_20_ERASE_SECTOR_SIZE_SLICE 45U, 39U
- #define MMCS_CSD_20_ERASE_BLK_EN_SLICE 46U, 46U
- #define MMCS_CSD_20_C_SIZE_SLICE 69U, 48U
- #define MMCS_CSD_20_DSR_IMP_SLICE 76U, 76U
- #define MMCS_CSD_20_READ_BLK_MISALIGN_SLICE 77U, 77U
- #define MMCS_CSD_20_WRITE_BLK_MISALIGN_SLICE 78U, 78U
- #define MMCS_CSD_20_READ_BL_PARTIAL_SLICE 79U, 79U
- #define MMCS_CSD_20_READ_BL_LEN_SLICE 83U, 80U
- #define MMCS_CSD_20_CCC_SLICE 95U, 84U
- #define MMCS_CSD_20_TRANS_SPEED_SLICE 103U, 96U
- #define MMCS_CSD_20_NSAC_SLICE 111U, 104U
- #define MMCS_CSD_20_TAAC_SLICE 119U, 112U
- #define MMCS_CSD_20_CSD_STRUCTURE_SLICE 127U, 126U
- #define MMCS_CSD_10_CRC_SLICE MMCS_CSD_20_CRC_SLICE
- #define MMCS_CSD_10_FILE_FORMAT_SLICE MMCS_CSD_20_FILE_FORMAT_SLICE
- #define MMCS_CSD_10_TMP_WRITE_PROTECT_SLICE MMCS_CSD_20_TMP_WRITE_PROTECT_SLICE
- #define MMCS_CSD_10_PERM_WRITE_PROTECT_SLICE MMCS_CSD_20_PERM_WRITE_PROTECT_SLICE
- #define MMCS_CSD_10_COPY_SLICE MMCS_CSD_20_COPY_SLICE
- #define MMCS_CSD_10_FILE_FORMAT_GRP_SLICE MMCS_CSD_20_FILE_FORMAT_GRP_SLICE
- #define MMCS_CSD_10_WRITE_BL_PARTIAL_SLICE MMCS_CSD_20_WRITE_BL_PARTIAL_SLICE
- #define MMCS_CSD_10_WRITE_BL_LEN_SLICE MMCS_CSD_20_WRITE_BL_LEN_SLICE
- #define MMCS_CSD_10_R2W_FACTOR_SLICE MMCS_CSD_20_R2W_FACTOR_SLICE
- #define MMCS_CSD_10_WP_GRP_ENABLE_SLICE MMCS_CSD_20_WP_GRP_ENABLE_SLICE
- #define MMCS_CSD_10_WP_GRP_SIZE_SLICE MMCS_CSD_20_WP_GRP_SIZE_SLICE
- #define MMCS_CSD_10_ERASE_SECTOR_SIZE_SLICE MMCS_CSD_20_ERASE_SECTOR_SIZE_SLICE
- #define MMCS_CSD_10_ERASE_BLK_EN_SLICE MMCS_CSD_20_ERASE_BLK_EN_SLICE
- #define MMCS_CSD_10_C_SIZE_MULT_SLICE 49U, 47U
- #define MMCS_CSD_10_VDD_W_CURR_MAX_SLICE 52U, 50U
- #define MMCS_CSD_10_VDD_W_CURR_MIN_SLICE 55U, 53U
- #define MMCS_CSD_10_VDD_R_CURR_MAX_SLICE 58U, 56U
- #define MMCS_CSD_10_VDD_R_CURR_MIX_SLICE 61U, 59U
- #define MMCS_CSD_10_C_SIZE_SLICE 73U, 62U
- #define MMCS_CSD_10_DSR_IMP_SLICE MMCS_CSD_20_DSR_IMP_SLICE
- #define MMCS_CSD_10_READ_BLK_MISALIGN_SLICE MMCS_CSD_20_READ_BLK_MISALIGN_SLICE
- #define MMCS_CSD_10_WRITE_BLK_MISALIGN_SLICE MMCS_CSD_20_WRITE_BLK_MISALIGN_SLICE
- #define MMCS_CSD_10_READ_BL_PARTIAL_SLICE MMCS_CSD_20_READ_BL_PARTIAL_SLICE
- #define MMCS_CSD_10_READ_BL_LEN_SLICE 83U, 80U
- #define MMCS_CSD_10_CCC_SLICE MMCS_CSD_20_CCC_SLICE
- #define MMCS_CSD_10_TRANS_SPEED_SLICE MMCS_CSD_20_TRANS_SPEED_SLICE
- #define MMCS_CSD_10_NSAC_SLICE MMCS_CSD_20_NSAC_SLICE
- #define MMCS_CSD_10_TAAC_SLICE MMCS_CSD_20_TAAC_SLICE
- #define MMCS_CSD_10_CSD_STRUCTURE_SLICE MMCS_CSD_20_CSD_STRUCTURE_SLICE

CID record offsets

- #define MMCS_CID_SDC_CRC_SLICE 7U, 1U
- #define MMCS_CID_SDC_MDT_M_SLICE 11U, 8U
- #define MMCS_CID_SDC_MDT_Y_SLICE 19U, 12U
- #define MMCS_CID_SDC_PSN_SLICE 55U, 24U
- #define MMCS_CID_SDC_PRV_M_SLICE 59U, 56U
- #define MMCS_CID_SDC_PRV_N_SLICE 63U, 60U
- #define MMCS_CID_SDC_PNM0_SLICE 71U, 64U
- #define MMCS_CID_SDC_PNM1_SLICE 79U, 72U
- #define MMCS_CID_SDC_PNM2_SLICE 87U, 80U
- #define MMCS_CID_SDC_PNM3_SLICE 95U, 88U
- #define MMCS_CID_SDC_PNM4_SLICE 103U, 96U
- #define MMCS_CID_SDC_OID_SLICE 119U, 104U
- #define MMCS_CID_SDC_MID_SLICE 127U, 120U
- #define MMCS_CID_MMC_CRC_SLICE 7U, 1U
- #define MMCS_CID_MMC_MDT_Y_SLICE 11U, 8U
- #define MMCS_CID_MMC_MDT_M_SLICE 15U, 12U
- #define MMCS_CID_MMC_PSN_SLICE 47U, 16U

- #define [MMCSDBLOCK_CID_MMC_PRV_M_SLICE](#) 51U, 48U
- #define [MMCSDBLOCK_CID_MMC_PRV_N_SLICE](#) 55U, 52U
- #define [MMCSDBLOCK_CID_MMC_PNM0_SLICE](#) 63U, 56U
- #define [MMCSDBLOCK_CID_MMC_PNM1_SLICE](#) 71U, 64U
- #define [MMCSDBLOCK_CID_MMC_PNM2_SLICE](#) 79U, 72U
- #define [MMCSDBLOCK_CID_MMC_PNM3_SLICE](#) 87U, 80U
- #define [MMCSDBLOCK_CID_MMC_PNM4_SLICE](#) 95U, 88U
- #define [MMCSDBLOCK_CID_MMC_PNM5_SLICE](#) 103U, 96U
- #define [MMCSDBLOCK_CID_MMC_OID_SLICE](#) 119U, 104U
- #define [MMCSDBLOCK_CID_MMC_MID_SLICE](#) 127U, 120U

R1 response utilities

- #define [MMCSDBLOCK_R1_ERROR](#)(r1)
Evaluates to true if the R1 response contains error flags.
- #define [MMCSDBLOCK_R1_STS](#)(r1)
Returns the status field of an R1 response.
- #define [MMCSDBLOCK_R1_IS_CARD_LOCKED](#)(r1)
Evaluates to true if the R1 response indicates a locked card.

Macro Functions

- #define [mmcGetCardCapacity](#)(ip)
Returns the card capacity in blocks.

Functions

- uint32_t [mmc_get_slice](#) (const uint32_t *data, uint32_t end, uint32_t start)
Gets a bit field from a words array.
- uint32_t [mmc_get_capacity](#) (const uint32_t *csd)
Extract card capacity from a CSD.
- uint32_t [mmc_get_capacity_ext](#) (const uint8_t *ext_csd)
Extract MMC card capacity from EXT_CSD.
- void [mmc_unpack_sdc_cid](#) (const MMCSDBlockDevice *sdcp, unpacked_sdc_cid_t *cidsdc)
Unpacks SDC CID array in structure.
- void [mmc_unpack_mmc_cid](#) (const MMCSDBlockDevice *sdcp, unpacked_mmc_cid_t *cidmmc)
Unpacks MMC CID array in structure.
- void [mmc_unpack_csd_mmc](#) (const MMCSDBlockDevice *sdcp, unpacked_mmc_csd_t *csdmmc)
Unpacks MMC CSD array in structure.
- void [mmc_unpack_csd_v10](#) (const MMCSDBlockDevice *sdcp, unpacked_sdc_csd_10_t *csd10)
Unpacks SDC CSD v1.0 array in structure.
- void [mmc_unpack_csd_v20](#) (const MMCSDBlockDevice *sdcp, unpacked_sdc_csd_20_t *csd20)
Unpacks SDC CSD v2.0 array in structure.

9.64.1 Detailed Description

MMC/SD cards common header.

This header defines an abstract interface useful to access MMC/SD I/O block devices in a standardized way.

9.65 hal_objects.h File Reference

Base object.

Data Structures

- struct [BaseObjectVMT](#)
BaseObject virtual methods table.
- struct [BaseObject](#)
Base object class.

Macros

- #define [_base_object_methods](#)
BaseObject specific methods.
- #define [_base_object_data](#)
BaseObject specific data.

Macro Functions (BaseObject)

- #define [objGetInstance](#)(type, ip)
Returns the instance pointer starting from an interface pointer.

9.65.1 Detailed Description

Base object.

This header defines a base object that is the root for the inheritance system.

9.66 hal_pal.h File Reference

I/O Ports Abstraction Layer macros, types and structures.

```
#include "hal_pal_lld.h"
```

Data Structures

- struct [palevent_t](#)
Type of a PAL event record.
- struct [IOBus](#)
I/O bus descriptor.

Macros

- #define `PAL_PORT_BIT`(n)
Port bit helper macro.
- #define `PAL_GROUP_MASK`(width)
Bits group mask helper.
- #define `_IOBUS_DATA`(name, port, width, offset)
Data part of a static I/O bus initializer.
- #define `IOBUS_DECL`(name, port, width, offset)
Static I/O bus initializer.

Pads mode constants

- #define `PAL_MODE_RESET` 0U
After reset state.
- #define `PAL_MODE_UNCONNECTED` 1U
*Safe state for **unconnected** pads.*
- #define `PAL_MODE_INPUT` 2U
Regular input high-Z pad.
- #define `PAL_MODE_INPUT_PULLUP` 3U
Input pad with weak pull up resistor.
- #define `PAL_MODE_INPUT_PULLDOWN` 4U
Input pad with weak pull down resistor.
- #define `PAL_MODE_INPUT_ANALOG` 5U
Analog input mode.
- #define `PAL_MODE_OUTPUT_PUSHPULL` 6U
Push-pull output pad.
- #define `PAL_MODE_OUTPUT_OPENDRAIN` 7U
Open-drain output pad.

Logic level constants

- #define `PAL_LOW` 0U
Logical low state.
- #define `PAL_HIGH` 1U
Logical high state.

PAL event modes

- #define `PAL_EVENT_MODE_EDGES_MASK` 3U
Mask of edges field.
- #define `PAL_EVENT_MODE_DISABLED` 0U
Channel disabled.
- #define `PAL_EVENT_MODE_RISING_EDGE` 1U
Rising edge callback.
- #define `PAL_EVENT_MODE_FALLING_EDGE` 2U
Falling edge callback.
- #define `PAL_EVENT_MODE_BOTH_EDGES` 3U
Both edges callback.

PAL configuration options

- #define `PAL_USE_CALLBACKS` TRUE
Enables synchronous APIs.
- #define `PAL_USE_WAIT` TRUE

Enables synchronous APIs.

Low level driver helper macros

- `#define _pal_init_event(e)`
Initializes a PAL event object.
- `#define _pal_init_event(e)`
Initializes a PAL event object.
- `#define _pal_init_event(e)`
Initializes a PAL event object.
- `#define _pal_clear_event(e)`
Clears a PAL event object.
- `#define _pal_clear_event(e)`
Clears a PAL event object.
- `#define _pal_clear_event(e)`
Clears a PAL event object.
- `#define _pal_isr_code(e)`
Common ISR code.
- `#define _pal_isr_code(e)`
Common ISR code.
- `#define _pal_isr_code(e)`
Common ISR code.

Macro Functions

- `#define palInit()`
PAL subsystem initialization.
- `#define palReadPort(port)`
Reads the physical I/O port states.
- `#define palReadLatch(port)`
Reads the output latch.
- `#define palWritePort(port, bits)`
Writes a bits mask on a I/O port.
- `#define palSetPort(port, bits)`
Sets a bits mask on a I/O port.
- `#define palClearPort(port, bits)`
Clears a bits mask on a I/O port.
- `#define palTogglePort(port, bits)`
Toggles a bits mask on a I/O port.
- `#define palReadGroup(port, mask, offset)`
Reads a group of bits.
- `#define palReadGroupLatch(port, mask, offset)`
Reads the group latch.
- `#define palWriteGroup(port, mask, offset, bits)`
Writes a group of bits.
- `#define palSetGroupMode(port, mask, offset, mode)`
Pads group mode setup.
- `#define palReadPad(port, pad)`
Reads an input pad logic state.
- `#define palWritePad(port, pad, bit)`
Writes a logic state on an output pad.
- `#define palSetPad(port, pad)`
Sets a pad logic state to `PAL_HIGH`.
- `#define palClearPad(port, pad)`
Clears a pad logic state to `PAL_LOW`.
- `#define palTogglePad(port, pad)`
Toggles a pad logic state.
- `#define palSetPadMode(port, pad, mode)`

- Pad mode setup.*
- #define [palReadLine](#)(line)
Reads an input line logic state.
- #define [palWriteLine](#)(line, bit)
Writes a logic state on an output line.
- #define [palSetLine](#)(line)
Sets a line logic state to `PAL_HIGH`.
- #define [palClearLine](#)(line)
Clears a line logic state to `PAL_LOW`.
- #define [palToggleLine](#)(line)
Toggles a line logic state.
- #define [palSetLineMode](#)(line, mode)
Line mode setup.
- #define [palEnablePadEventI](#)(port, pad, mode)
Pad event enable.
- #define [palDisablePadEventI](#)(port, pad)
Pad event disable.
- #define [palEnablePadEvent](#)(port, pad, mode)
Pad event enable.
- #define [palDisablePadEvent](#)(port, pad)
Pad event disable.
- #define [palEnableLineEventI](#)(line, mode)
Line event enable.
- #define [palDisableLineEventI](#)(line)
Line event disable.
- #define [palEnableLineEvent](#)(line, mode)
Line event enable.
- #define [palDisableLineEvent](#)(line)
Line event disable.
- #define [pallsPadEventEnabledX](#)(port, pad)
Pad event enable check.
- #define [pallsLineEventEnabledX](#)(line)
Line event enable check.
- #define [palSetPadCallback](#)(port, pad, cb, arg)
Associates a callback to a pad.
- #define [palSetLineCallback](#)(line, cb, arg)
Associates a callback to a line.

Typedefs

- typedef void(* [palcallback_t](#)) (void *arg)
Type of a PAL event callback.

Functions

- [ioportmask_t](#) [palReadBus](#) (const [IOBus](#) *bus)
Read from an I/O bus.
- void [palWriteBus](#) (const [IOBus](#) *bus, [ioportmask_t](#) bits)
Write to an I/O bus.
- void [palSetBusMode](#) (const [IOBus](#) *bus, [iomode_t](#) mode)
Programs a bus with the specified mode.
- void [palSetPadCallbackI](#) ([ioportid_t](#) port, [iopadid_t](#) pad, [palcallback_t](#) cb, void *arg)
Associates a callback to a port/pad.
- void [palSetLineCallbackI](#) ([ioline_t](#) line, [palcallback_t](#) cb, void *arg)
Associates a callback to a line.

- [msg_t palWaitPadTimeoutS](#) (ioportid_t port, iopadid_t pad, sysinterval_t timeout)
Waits for an edge on the specified port/pad.
- [msg_t palWaitPadTimeout](#) (ioportid_t port, iopadid_t pad, sysinterval_t timeout)
Waits for an edge on the specified port/pad.
- [msg_t palWaitLineTimeoutS](#) (ioline_t line, sysinterval_t timeout)
Waits for an edge on the specified line.
- [msg_t palWaitLineTimeout](#) (ioline_t line, sysinterval_t timeout)
Waits for an edge on the specified line.

9.66.1 Detailed Description

I/O Ports Abstraction Layer macros, types and structures.

9.67 hal_persistent.h File Reference

Generic persistent storage class header.

Data Structures

- struct [BasePersistentStorageVMT](#)
BasePersistentStorage virtual methods table.
- struct [BasePersistentStorage](#)
Base persistent storage class.

Macros

- #define [_base_pers_storage_methods_alone](#)
BasePersistentStorage specific methods.
- #define [_base_pers_storage_methods](#)
BasePersistentStorage specific methods with inherited ones.
- #define [_base_persistent_storage_data](#) [_base_object_data](#)
BasePersistentStorage specific data.

Macro Functions (BasePersistentStorage)

- #define [getBasePersistentStorage](#)(ip)
Instance getter.
- #define [psGetStorageSize](#)(ip)
Get storage size.
- #define [psRead](#)(ip, offset, n, rp)
Read operation.
- #define [psWrite](#)(ip, offset, n, wp)
Write operation.

Typedefs

- typedef uint32_t [ps_offset_t](#)
Type of a persistent storage offset.

Enumerations

- enum [ps_error_t](#) {
[PS_NO_ERROR](#) = 0 , [PS_ERROR_READ](#) = 2 , [PS_ERROR_WRITE](#) = 3 , [PS_ERROR_VERIFY](#) = 5 ,
[PS_ERROR_HW_FAILURE](#) = 6 }
Type of a persistent storage error code.

9.67.1 Detailed Description

Generic persistent storage class header.

9.68 hal_pwm.h File Reference

PWM Driver macros and structures.

```
#include "hal_pwm_lld.h"
```

Macros

PWM output mode macros

- #define [PWM_OUTPUT_MASK](#) 0x0FU
Standard output modes mask.
- #define [PWM_OUTPUT_DISABLED](#) 0x00U
Output not driven, callback only.
- #define [PWM_OUTPUT_ACTIVE_HIGH](#) 0x01U
Positive PWM logic, active is logic level one.
- #define [PWM_OUTPUT_ACTIVE_LOW](#) 0x02U
Inverse PWM logic, active is logic level zero.

PWM duty cycle conversion

- #define [PWM_FRACTION_TO_WIDTH](#)(pwmp, denominator, numerator)
Converts from fraction to pulse width.
- #define [PWM_DEGREES_TO_WIDTH](#)(pwmp, degrees)
Converts from degrees to pulse width.
- #define [PWM_PERCENTAGE_TO_WIDTH](#)(pwmp, percentage)
Converts from percentage to pulse width.

Macro Functions

- #define [pwmChangePeriodl](#)(pwmp, value)
Changes the period the PWM peripheral.
- #define [pwmEnableChannell](#)(pwmp, channel, width)
Enables a PWM channel.
- #define [pwmDisableChannell](#)(pwmp, channel)
Disables a PWM channel.
- #define [pwmlsChannelEnabledl](#)(pwmp, channel)
Returns a PWM channel status.
- #define [pwmEnablePeriodicNotificationl](#)(pwmp)
Enables the periodic activation edge notification.
- #define [pwmDisablePeriodicNotificationl](#)(pwmp)
Disables the periodic activation edge notification.
- #define [pwmEnableChannelNotificationl](#)(pwmp, channel)
Enables a channel de-activation edge notification.
- #define [pwmDisableChannelNotificationl](#)(pwmp, channel)
Disables a channel de-activation edge notification.

Typedefs

- typedef struct PWMDriver [PWMDriver](#)
Type of a structure representing a PWM driver.
- typedef void(* [pwmcallback_t](#)) ([PWMDriver](#) *pwmp)
Type of a PWM notification callback.

Enumerations

- enum [pwmstate_t](#) { [PWM_UNINIT](#) = 0 , [PWM_STOP](#) = 1 , [PWM_READY](#) = 2 }
Driver state machine possible states.

Functions

- void [pwmlnit](#) (void)
PWM Driver initialization.
- void [pwmObjectInit](#) ([PWMDriver](#) *pwmp)
Initializes the standard part of a [PWMDriver](#) structure.
- [msg_t](#) [pwmStart](#) ([PWMDriver](#) *pwmp, const [PWMConfig](#) *config)
Configures and activates the PWM peripheral.
- void [pwmStop](#) ([PWMDriver](#) *pwmp)
Deactivates the PWM peripheral.
- void [pwmChangePeriod](#) ([PWMDriver](#) *pwmp, [pwmcnt_t](#) period)
Changes the period the PWM peripheral.
- void [pwmEnableChannel](#) ([PWMDriver](#) *pwmp, [pwmchannel_t](#) channel, [pwmcnt_t](#) width)
Enables a PWM channel.
- void [pwmDisableChannel](#) ([PWMDriver](#) *pwmp, [pwmchannel_t](#) channel)
Disables a PWM channel and its notification.
- void [pwmEnablePeriodicNotification](#) ([PWMDriver](#) *pwmp)
Enables the periodic activation edge notification.
- void [pwmDisablePeriodicNotification](#) ([PWMDriver](#) *pwmp)
Disables the periodic activation edge notification.
- void [pwmEnableChannelNotification](#) ([PWMDriver](#) *pwmp, [pwmchannel_t](#) channel)
Enables a channel de-activation edge notification.
- void [pwmDisableChannelNotification](#) ([PWMDriver](#) *pwmp, [pwmchannel_t](#) channel)
Disables a channel de-activation edge notification.

9.68.1 Detailed Description

PWM Driver macros and structures.

9.69 hal_queues.h File Reference

I/O Queues macros and structures.

Data Structures

- struct [io_queue](#)
Generic I/O queue structure.

Macros

Queue functions returned status value

- #define [Q_OK](#) [MSG_OK](#)
Operation successful.
- #define [Q_TIMEOUT](#) [MSG_TIMEOUT](#)
Timeout condition.
- #define [Q_RESET](#) [MSG_RESET](#)
Queue has been reset.
- #define [Q_EMPTY](#) [MSG_TIMEOUT](#)
Queue empty.
- #define [Q_FULL](#) [MSG_TIMEOUT](#)
Queue full,.

Macro Functions

- #define [qSizeX](#)(qp)
Returns the queue's buffer size.
- #define [qSpacel](#)(qp)
Queue space.
- #define [qGetLink](#)(qp)
Returns the queue application-defined link.
- #define [qSetLink](#)(qp, lk)
Sets the queue application-defined link.
- #define [iqGetFulll](#)(iqp)
Returns the filled space into an input queue.
- #define [iqGetEmptyl](#)(iqp)
Returns the empty space into an input queue.
- #define [iqIsEmptyl](#)(iqp)
Evaluates to `true` if the specified input queue is empty.
- #define [iqIsFullll](#)(iqp)
Evaluates to `true` if the specified input queue is full.
- #define [iqGet](#)(iqp)
Input queue read.
- #define [oqGetFullll](#)(oqp)
Returns the filled space into an output queue.
- #define [oqGetEmptyl](#)(oqp)
Returns the empty space into an output queue.
- #define [oqlsEmptyl](#)(oqp)
Evaluates to `true` if the specified output queue is empty.
- #define [oqlsFullll](#)(oqp)
Evaluates to `true` if the specified output queue is full.
- #define [oqPut](#)(oqp, b)
Output queue write.

Typedefs

- typedef struct [io_queue](#) [io_queue_t](#)
Type of a generic I/O queue structure.
- typedef void(* [qnotify_t](#)) ([io_queue_t](#) *qp)
Queue notification callback type.
- typedef [io_queue_t](#) [input_queue_t](#)
Type of an input queue structure.
- typedef [io_queue_t](#) [output_queue_t](#)
Type of an output queue structure.

Functions

- void [iqObjectInit](#) ([input_queue_t](#) *iqp, [uint8_t](#) *bp, [size_t](#) size, [qnotify_t](#) infy, void *link)
Initializes an input queue.
- void [iqResetI](#) ([input_queue_t](#) *iqp)
Resets an input queue.
- [msg_t](#) [iqPutI](#) ([input_queue_t](#) *iqp, [uint8_t](#) b)
Input queue write.
- [msg_t](#) [iqGetI](#) ([input_queue_t](#) *iqp)
Input queue non-blocking read.
- [msg_t](#) [iqGetTimeout](#) ([input_queue_t](#) *iqp, [sysinterval_t](#) timeout)
Input queue read with timeout.
- [size_t](#) [iqReadI](#) ([input_queue_t](#) *iqp, [uint8_t](#) *bp, [size_t](#) n)
Input queue non-blocking read.
- [size_t](#) [iqReadTimeout](#) ([input_queue_t](#) *iqp, [uint8_t](#) *bp, [size_t](#) n, [sysinterval_t](#) timeout)
Input queue read with timeout.
- void [oqObjectInit](#) ([output_queue_t](#) *oqp, [uint8_t](#) *bp, [size_t](#) size, [qnotify_t](#) onfy, void *link)
Initializes an output queue.
- void [oqResetI](#) ([output_queue_t](#) *oqp)
Resets an output queue.
- [msg_t](#) [oqPutI](#) ([output_queue_t](#) *oqp, [uint8_t](#) b)
Output queue non-blocking write.
- [msg_t](#) [oqPutTimeout](#) ([output_queue_t](#) *oqp, [uint8_t](#) b, [sysinterval_t](#) timeout)
Output queue write with timeout.
- [msg_t](#) [oqGetI](#) ([output_queue_t](#) *oqp)
Output queue read.
- [size_t](#) [oqWriteI](#) ([output_queue_t](#) *oqp, const [uint8_t](#) *bp, [size_t](#) n)
Output queue non-blocking write.
- [size_t](#) [oqWriteTimeout](#) ([output_queue_t](#) *oqp, const [uint8_t](#) *bp, [size_t](#) n, [sysinterval_t](#) timeout)
Output queue write with timeout.

9.69.1 Detailed Description

I/O Queues macros and structures.

9.70 hal_rtc.h File Reference

RTC Driver macros and structures.

```
#include <time.h>
#include "hal_rtc_lld.h"
```

Data Structures

- struct [RTCDateTime](#)
Type of a structure representing an RTC date/time stamp.
- struct [RTCDriverVMT](#)
RTCDriver virtual methods table.
- struct [RTCDriver](#)
Structure representing an RTC driver.

Macros

- #define `RTC_BASE_YEAR` 1980U
Base year of the calendar.
- #define `_rtc_driver_methods _base_pers_storage_methods`
BasePersistentStorage specific methods.

Date/Time bit masks for FAT format

- #define `RTC_FAT_TIME_SECONDS_MASK` 0x0000001FU
- #define `RTC_FAT_TIME_MINUTES_MASK` 0x000007E0U
- #define `RTC_FAT_TIME_HOURS_MASK` 0x0000F800U
- #define `RTC_FAT_DATE_DAYS_MASK` 0x001F0000U
- #define `RTC_FAT_DATE_MONTHS_MASK` 0x01E00000U
- #define `RTC_FAT_DATE_YEARS_MASK` 0xFE000000U

Day of week encoding

- #define `RTC_DAY_CATURDAY` 0U
- #define `RTC_DAY_MONDAY` 1U
- #define `RTC_DAY_TUESDAY` 2U
- #define `RTC_DAY_WEDNESDAY` 3U
- #define `RTC_DAY_THURSDAY` 4U
- #define `RTC_DAY_FRIDAY` 5U
- #define `RTC_DAY_SATURDAY` 6U
- #define `RTC_DAY_SUNDAY` 7U

Typedefs

- typedef struct RTCDriver `RTCDriver`
Type of a structure representing an RTC driver.
- typedef unsigned int `rtcalarm_t`
Type of an RTC alarm number.

Functions

- void `rtcInit` (void)
RTC Driver initialization.
- void `rtcObjectInit` (RTCDriver *rtcp)
Initializes a generic RTC driver object.
- void `rtcSetTime` (RTCDriver *rtcp, const RTCDateTime *timespec)
Set current time.
- void `rtcGetTime` (RTCDriver *rtcp, RTCDateTime *timespec)
Get current time.
- void `rtcSetCallback` (RTCDriver *rtcp, rtccb_t callback)
Enables or disables RTC callbacks.
- void `rtcConvertDateTimeToStructTm` (const RTCDateTime *timespec, struct tm *timp, uint32_t *tv_msec)
Convert RTCDateTime to broken-down time structure.
- void `rtcConvertStructTmToDateTime` (const struct tm *timp, uint32_t tv_msec, RTCDateTime *timespec)
Convert broken-down time structure to RTCDateTime.
- uint32_t `rtcConvertDateTimeToFAT` (const RTCDateTime *timespec)
Get current time in format suitable for usage in FAT file system.

9.70.1 Detailed Description

RTC Driver macros and structures.

9.71 hal_sdc.h File Reference

SDC Driver macros and structures.

```
#include "hal_sdc_lld.h"
```

Macros

SD card types

- #define [SDC_MODE_CARDTYPE_MASK](#) 0xFU
- #define [SDC_MODE_CARDTYPE_SDV11](#) 0U
- #define [SDC_MODE_CARDTYPE_SDV20](#) 1U
- #define [SDC_MODE_CARDTYPE_MMC](#) 2U
- #define [SDC_MODE_HIGH_CAPACITY](#) 0x10U

SDC bus error conditions

- #define [SDC_NO_ERROR](#) 0U
- #define [SDC_CMD_CRC_ERROR](#) 1U
- #define [SDC_DATA_CRC_ERROR](#) 2U
- #define [SDC_DATA_TIMEOUT](#) 4U
- #define [SDC_COMMAND_TIMEOUT](#) 8U
- #define [SDC_TX_UNDERRUN](#) 16U
- #define [SDC_RX_OVERRUN](#) 32U
- #define [SDC_STARTBIT_ERROR](#) 64U
- #define [SDC_OVERFLOW_ERROR](#) 128U
- #define [SDC_UNHANDLED_ERROR](#) 0xFFFFFFFFU

SDC configuration options

- #define [SDC_INIT_RETRY](#) 100
Number of initialization attempts before rejecting the card.
- #define [SDC_MMC_SUPPORT](#) FALSE
Include support for MMC cards.
- #define [SDC_NICE_WAITING](#) TRUE
Delays insertions.
- #define [SDC_INIT_OCR_V20](#) 0x50FF8000U
OCR initialization constant for V20 cards.
- #define [SDC_INIT_OCR](#) 0x80100000U
OCR initialization constant for non-V20 cards.

Macro Functions

- #define [sdclsCardInserted](#)(sdcp)
Returns the card insertion status.
- #define [sdclsWriteProtected](#)(sdcp)
Returns the write protect status.

Enumerations

- enum `sdcbusmode_t` { `SDC_MODE_1BIT` = 0 , `SDC_MODE_4BIT` , `SDC_MODE_8BIT` }
Type of SDIO bus mode.
- enum `sdcbusclk_t` { `SDC_CLK_25MHz` = 0 , `SDC_CLK_50MHz` }
Max supported clock.

Functions

- void `sdclnit` (void)
SDC Driver initialization.
- void `sdcObjectInit` (SDCDriver *sdcp)
Initializes the standard part of a `SDCDriver` structure.
- `msg_t` `sdStart` (SDCDriver *sdcp, const `SDCConfig` *config)
Configures and activates the SDC peripheral.
- void `sdStop` (SDCDriver *sdcp)
Deactivates the SDC peripheral.
- bool `sdConnect` (SDCDriver *sdcp)
Performs the initialization procedure on the inserted card.
- bool `sdDisconnect` (SDCDriver *sdcp)
Brings the driver in a state safe for card removal.
- bool `sdRead` (SDCDriver *sdcp, uint32_t startblk, uint8_t *buf, uint32_t n)
Reads one or more blocks.
- bool `sdWrite` (SDCDriver *sdcp, uint32_t startblk, const uint8_t *buf, uint32_t n)
Writes one or more blocks.
- `sdflags_t` `sdGetAndClearErrors` (SDCDriver *sdcp)
Returns the errors mask associated to the previous operation.
- bool `sdSync` (SDCDriver *sdcp)
Waits for card idle condition.
- bool `sdGetInfo` (SDCDriver *sdcp, `BlockDeviceInfo` *bdip)
Returns the media info.
- bool `sdErase` (SDCDriver *sdcp, uint32_t startblk, uint32_t endblk)
Erases the supplied blocks.
- bool `_sd_wait_for_transfer_state` (SDCDriver *sdcp)
Wait for the card to complete pending operations with CRC check.
- bool `_sd_wait_for_transfer_state_nocrc` (SDCDriver *sdcp)
Wait for the card to complete pending operations without CRC check.

9.71.1 Detailed Description

SDC Driver macros and structures.

9.72 hal_serial.h File Reference

Serial Driver macros and structures.

```
#include "hal_serial_lld.h"
```

Data Structures

- struct [SerialDriverVMT](#)
SerialDriver virtual methods table.
- struct [hal_serial_driver](#)
Full duplex serial driver class.

Macros

- #define [_serial_driver_methods_base_asynchronous_channel_methods](#)
SerialDriver specific methods.

Serial status flags (legacy)

- #define [SD_PARITY_ERROR](#) [CHN_PARITY_ERROR](#)
- #define [SD_FRAMING_ERROR](#) [CHN_FRAMING_ERROR](#)
- #define [SD_OVERRUN_ERROR](#) [CHN_OVERRUN_ERROR](#)
- #define [SD_NOISE_ERROR](#) [CHN_NOISE_ERROR](#)
- #define [SD_BREAK_DETECTED](#) [CHN_BREAK_DETECTED](#)
- #define [SD_QUEUE_FULL_ERROR](#) [CHN_BUFFER_FULL_ERROR](#)

Serial configuration options

- #define [SERIAL_DEFAULT_BITRATE](#) 38400
Default bit rate.
- #define [SERIAL_BUFFERS_SIZE](#) 16
Serial buffers size.

Macro Functions

- #define [sdPutl](#)(sdp, b)
Direct write to a [SerialDriver](#).
- #define [sdPut](#)(sdp, b)
Direct write to a [SerialDriver](#).
- #define [sdPutTimeout](#)(sdp, b, t)
Direct write to a [SerialDriver](#) with timeout specification.
- #define [sdGetl](#)(sdp)
Direct read from a [SerialDriver](#).
- #define [sdGet](#)(sdp)
Direct read from a [SerialDriver](#).
- #define [sdGetTimeout](#)(sdp, t)
Direct read from a [SerialDriver](#) with timeout specification.
- #define [sdWritel](#)(sdp, b, n)
Direct non-blocking write to a [SerialDriver](#).
- #define [sdWrite](#)(sdp, b, n)
Direct blocking write to a [SerialDriver](#).
- #define [sdWriteTimeout](#)(sdp, b, n, t)
Direct blocking write to a [SerialDriver](#) with timeout specification.
- #define [sdAsynchronousWrite](#)(sdp, b, n)
Direct non-blocking write to a [SerialDriver](#).
- #define [sdReadl](#)(sdp, b, n)
Direct non-blocking read from a [SerialDriver](#).
- #define [sdRead](#)(sdp, b, n)
Direct blocking read from a [SerialDriver](#).
- #define [sdReadTimeout](#)(sdp, b, n, t)
Direct blocking read from a [SerialDriver](#) with timeout specification.
- #define [sdAsynchronousRead](#)(sdp, b, n)
Direct non-blocking read from a [SerialDriver](#).

Typedefs

- typedef struct [hal_serial_driver](#) [SerialDriver](#)
Structure representing a serial driver.

Enumerations

- enum [sdstate_t](#) { [SD_UNINIT](#) = 0 , [SD_STOP](#) = 1 , [SD_READY](#) = 2 }
Driver state machine possible states.

Functions

- void [sdInit](#) (void)
Serial Driver initialization.
- void [sdObjectInit](#) ([SerialDriver](#) *sdp, [qnotify_t](#) inotify, [qnotify_t](#) onotify)
Initializes a generic serial driver object.
- [msg_t](#) [sdStart](#) ([SerialDriver](#) *sdp, const [SerialConfig](#) *config)
Configures and starts the driver.
- void [sdStop](#) ([SerialDriver](#) *sdp)
Stops the driver.
- void [sdIncomingData](#) ([SerialDriver](#) *sdp, uint8_t b)
Handles incoming data.
- [msg_t](#) [sdRequestData](#) ([SerialDriver](#) *sdp)
Handles outgoing data.
- bool [sdPutWouldBlock](#) ([SerialDriver](#) *sdp)
Direct output check on a [SerialDriver](#).
- bool [sdGetWouldBlock](#) ([SerialDriver](#) *sdp)
Direct input check on a [SerialDriver](#).
- [msg_t](#) [sdControl](#) ([SerialDriver](#) *sdp, unsigned int operation, void *arg)
Control operation on a serial port.

9.72.1 Detailed Description

Serial Driver macros and structures.

9.73 hal_serial_usb.h File Reference

Serial over USB Driver macros and structures.

```
#include "hal_usb_cdc.h"
```

Data Structures

- struct [SerialUSBConfig](#)
Serial over USB Driver configuration structure.
- struct [SerialUSBDriverVMT](#)
[SerialDriver](#) virtual methods table.
- struct [SerialUSBDriver](#)
Full duplex serial driver class.

Macros

- `#define _serial_usb_driver_data`
SerialDriver specific data.
- `#define _serial_usb_driver_methods _base_asynchronous_channel_methods`
SerialUSBDriver specific methods.

SERIAL_USB configuration options

- `#define SERIAL_USB_BUFFERS_SIZE 256`
Serial over USB buffers size.
- `#define SERIAL_USB_BUFFERS_NUMBER 2`
Serial over USB number of buffers.

Typedefs

- `typedef struct SerialUSBDriver SerialUSBDriver`
Structure representing a serial over USB driver.

Enumerations

- `enum sdustate_t { SDU_UNINIT = 0 , SDU_STOP = 1 , SDU_READY = 2 }`
Driver state machine possible states.

Functions

- `void sduInit (void)`
Serial Driver initialization.
- `void sduObjectInit (SerialUSBDriver *sdup)`
Initializes a generic full duplex driver object.
- `msg_t sduStart (SerialUSBDriver *sdup, const SerialUSBConfig *config)`
Configures and starts the driver.
- `void sduStop (SerialUSBDriver *sdup)`
Stops the driver.
- `void sduSuspendHookI (SerialUSBDriver *sdup)`
USB device suspend handler.
- `void sduWakeupHookI (SerialUSBDriver *sdup)`
USB device wakeup handler.
- `void sduConfigureHookI (SerialUSBDriver *sdup)`
USB device configured handler.
- `bool sduRequestsHook (USBDriver *usbp)`
Default requests hook.
- `void sduSOFHookI (SerialUSBDriver *sdup)`
SOF handler.
- `void sduDataTransmitted (USBDriver *usbp, usbep_t ep)`
Default data transmitted callback.
- `void sduDataReceived (USBDriver *usbp, usbep_t ep)`
Default data received callback.
- `void sduInterruptTransmitted (USBDriver *usbp, usbep_t ep)`
Default data received callback.
- `msg_t sduControl (USBDriver *usbp, unsigned int operation, void *arg)`
Control operation on a serial USB port.

9.73.1 Detailed Description

Serial over USB Driver macros and structures.

9.74 hal_sio.h File Reference

SIO Driver macros and structures.

```
#include "hal_sio_llid.h"
```

Data Structures

- struct [hal_sio_config](#)
Driver configuration structure.
- struct [sio_driver_vmt](#)
SIODriver virtual methods table.
- struct [hal_sio_driver](#)
Structure representing a SIO driver.

Macros

- #define [_sio_driver_methods_base_channel_methods](#)
SIODriver specific methods.
- #define [sioSetCallbackX](#)(siop, f)
Associates a callback to the SIO instance.
- #define [siolsRXEmptyX](#)(siop)
Determines the state of the RX FIFO.
- #define [siolsRXIdleX](#)(siop)
Determines the activity state of the receiver.
- #define [sioHasRXErrorsX](#)(siop)
Determines if RX has pending errors to be read and cleared.
- #define [siolsTXFullX](#)(siop)
Determines the state of the TX FIFO.
- #define [siolsTXOngoingX](#)(siop)
Determines the transmission state.
- #define [sioWriteEnableFlagsX](#)(siop, mask)
Writes the enabled events mask.
- #define [sioSetEnableFlagsX](#)(siop, mask)
Sets flags into the enabled events flags mask.
- #define [sioClearEnableFlagsX](#)(siop, mask)
Clears flags from the enabled events flags mask.
- #define [sioGetEnableFlagsX](#)(siop)
Return the enabled condition flags mask.
- #define [sioGetAndClearErrorsX](#)(siop)
Get and clears SIO error event flags.
- #define [sioGetAndClearEventsX](#)(siop)
Get and clears SIO event flags.

- `#define sioGetEventsX(siop)`
Returns the pending SIO event flags.
- `#define sioGetX(siop)`
Returns one frame from the RX FIFO.
- `#define sioPutX(siop, data)`
Pushes one frame into the TX FIFO.
- `#define sioAsyncReadX(siop, size, buffer)`
Reads data from the RX FIFO.
- `#define sioAsyncWriteX(siop, size, buffer)`
Writes data into the TX FIFO.
- `#define sioControlX(siop, operation, arg)`
Control operation on a serial port.
- `#define __sio_reloc_field(v, m, s, d)`
Relocates a bit field.

SIO masks offsets

- `#define SIO_EV_RXNOTEMPY_POS 2 /* CHN_INPUT_AVAILABLE */`
- `#define SIO_EV_TXNOTFULL_POS 3 /* CHN_OUTPUT_EMPTY */`
- `#define SIO_EV_TXDONE_POS 4 /* CHN_TRANSMISSION_END */`
- `#define SIO_EV_ALL_ERRORS_POS SIO_EV_PARITY_ERR_POS`
- `#define SIO_EV_PARITY_ERR_POS 5 /* CHN_PARITY_ERROR */`
- `#define SIO_EV_FRAMING_ERR_POS 6 /* CHN_FRAMING_ERROR */`
- `#define SIO_EV_NOISE_ERR_POS 7 /* CHN_NOISE_ERROR */`
- `#define SIO_EV_OVERRUN_ERR_POS 8 /* CHN_OVERRUN_ERROR */`
- `#define SIO_EV_RXIDLE_POS 9 /* CHN_IDLE_DETECTED */`
- `#define SIO_EV_RXBREAK_POS 10 /* CHN_BREAK_DETECTED */`

Event flags (compatible with channel and serial events)

- `#define SIO_EV_NONE 0U`
- `#define SIO_EV_RXNOTEMPY (1U << SIO_EV_RXNOTEMPY_POS)`
- `#define SIO_EV_TXNOTFULL (1U << SIO_EV_TXNOTFULL_POS)`
- `#define SIO_EV_ALL_DATA (SIO_EV_RXNOTEMPY | SIO_EV_TXNOTFULL)`
- `#define SIO_EV_TXDONE (1U << SIO_EV_TXDONE_POS)`
- `#define SIO_EV_PARITY_ERR (1U << SIO_EV_PARITY_ERR_POS)`
- `#define SIO_EV_FRAMING_ERR (1U << SIO_EV_FRAMING_ERR_POS)`
- `#define SIO_EV_NOISE_ERR (1U << SIO_EV_NOISE_ERR_POS)`
- `#define SIO_EV_OVERRUN_ERR (1U << SIO_EV_OVERRUN_ERR_POS)`
- `#define SIO_EV_RXIDLE (1U << SIO_EV_RXIDLE_POS)`
- `#define SIO_EV_RXBREAK (1U << SIO_EV_RXBREAK_POS)`
- `#define SIO_EV_ALL_ERRORS`
- `#define SIO_EV_ALL_EVENTS`

Additional messages

- `#define SIO_MSG_ERRORS (msg_t)1`

SIO configuration options

- `#define SIO_DEFAULT_BITRATE 38400`
Default bit rate.
- `#define SIO_USE_SYNCHRONIZATION TRUE`
Support for thread synchronization API.
- `#define SIO_USE_STREAMS_INTERFACE SIO_USE_SYNCHRONIZATION`
Support for streams interfacwe.

Low level driver helper macros

- `#define __sio_callback(siop)`
SIO callback.
- `#define __sio_wakeup_errors(siop)`
Wakes up because RX errors.
- `#define __sio_wakeup_rx(siop)`
Wakes up the RX-waiting thread.
- `#define __sio_wakeup_rxidle(siop)`
Wakes up the RX-idle-waiting thread.
- `#define __sio_wakeup_tx(siop)`
Wakes up the TX-waiting thread.
- `#define __sio_wakeup_txend(siop)`
Wakes up the TXend-waiting thread.

Typedefs

- `typedef eventflags_t sioevents_t`
Type of event flags.
- `typedef struct hal_sio_driver SIODriver`
Type of structure representing a SIO driver.
- `typedef struct hal_sio_config SIOConfig`
Type of structure representing a SIO configuration.
- `typedef void(* siocb_t) (SIODriver *siop)`
Generic SIO notification callback type.

Enumerations

- `enum siostate_t { SIO_UNINIT = 0 , SIO_STOP = 1 , SIO_READY = 2 }`
Driver state machine possible states.

Functions

- `void siolnit (void)`
SIO Driver initialization.
- `void sioObjectInit (SIODriver *siop)`
*Initializes the standard part of a *SIODriver* structure.*
- `msg_t sioStart (SIODriver *siop, const SIOConfig *config)`
Configures and activates the SIO peripheral.
- `void sioStop (SIODriver *siop)`
Deactivates the SIO peripheral.
- `void sioWriteEnableFlags (SIODriver *siop, sioevents_t mask)`
Writes the enabled events flags mask.
- `void sioSetEnableFlags (SIODriver *siop, sioevents_t mask)`
Sets flags into the enabled events flags mask.
- `void sioClearEnableFlags (SIODriver *siop, sioevents_t mask)`
Clears flags from the enabled events flags mask.
- `sioevents_t sioGetAndClearErrors (SIODriver *siop)`
Get and clears SIO error event flags.
- `sioevents_t sioGetAndClearEvents (SIODriver *siop)`
Get and clears SIO event flags.

- [sioevents_t sioGetEvents](#) ([SIODriver](#) *siop)
Returns the pending SIO event flags.
- [size_t sioAsyncRead](#) ([SIODriver](#) *siop, [uint8_t](#) *buffer, [size_t](#) n)
Reads data from the RX FIFO.
- [size_t sioAsyncWrite](#) ([SIODriver](#) *siop, [const uint8_t](#) *buffer, [size_t](#) n)
Writes data into the TX FIFO.
- [msg_t sioSynchronizeRX](#) ([SIODriver](#) *siop, [sysinterval_t](#) timeout)
Synchronizes with RX FIFO data availability.
- [msg_t sioSynchronizeRXIdle](#) ([SIODriver](#) *siop, [sysinterval_t](#) timeout)
Synchronizes with RX going idle.
- [msg_t sioSynchronizeTX](#) ([SIODriver](#) *siop, [sysinterval_t](#) timeout)
Synchronizes with TX FIFO space availability.
- [msg_t sioSynchronizeTXEnd](#) ([SIODriver](#) *siop, [sysinterval_t](#) timeout)
Synchronizes with TX completion.

9.74.1 Detailed Description

SIO Driver macros and structures.

9.75 hal_spi.h File Reference

SPI Driver selector macros and structures.

```
#include "hal_spi_v2.h"
#include "hal_spi_v1.h"
```

9.75.1 Detailed Description

SPI Driver selector macros and structures.

9.76 hal_spi_v1.h File Reference

SPI (v1) Driver macros and structures.

```
#include "hal_spi_ll.h"
```

Data Structures

- struct [hal_spi_config](#)
Driver configuration structure.
- struct [hal_spi_driver](#)
Structure representing an SPI driver.

Macros

Chip Select modes

- #define `SPI_SELECT_MODE_NONE`
- #define `SPI_SELECT_MODE_PAD` 1 /** @brief Legacy mode. */
- #define `SPI_SELECT_MODE_PORT` 2 /** @brief Fastest mode. */
- #define `SPI_SELECT_MODE_LINE` 3 /** @brief Packed mode. */
- #define `SPI_SELECT_MODE_LLD` 4 /** @brief LLD-defined mode.*/

SPI configuration options

- #define `SPI_USE_WAIT` TRUE
Enables synchronous APIs.
- #define `SPI_USE_MUTUAL_EXCLUSION` TRUE
Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.
- #define `SPI_SELECT_MODE_SPI_SELECT_MODE_PAD`
Handling method for SPI CS line.

Macro Functions

- #define `spisBufferComplete`(spip)
Buffer state.
- #define `spiSelectl`(spip)
Asserts the slave select signal and prepares for transfers.
- #define `spiUnselectl`(spip)
Deasserts the slave select signal.
- #define `spiStartIgnorel`(spip, n)
Ignores data on the SPI bus.
- #define `spiStartExchangel`(spip, n, txbuf, rxbuf)
Exchanges data on the SPI bus.
- #define `spiStartSendl`(spip, n, txbuf)
Sends data over the SPI bus.
- #define `spiStartReceivel`(spip, n, rxbuf)
Receives data from the SPI bus.
- #define `spiPolledExchange`(spip, frame)
Exchanges one frame using a polled wait.

Low level driver helper macros

- #define `_spi_wakeup_isr`(spip)
Wakes up the waiting thread.
- #define `_spi_isr_code`(spip)
Common ISR code when circular mode is not supported.
- #define `_spi_isr_half_code`(spip)
Half buffer filled ISR code in circular mode.
- #define `_spi_isr_full_code`(spip)
Full buffer filled ISR code in circular mode.

Typedefs

- typedef struct `hal_spi_driver` SPIDriver
Type of a structure representing an SPI driver.
- typedef struct `hal_spi_config` SPIConfig
Type of a SPI driver configuration structure.
- typedef void(* `spicallback_t`) (SPIDriver *spip)
SPI notification callback type.

Enumerations

- enum `spistate_t` {
`SPI_UNINIT` = 0 , `SPI_STOP` = 1 , `SPI_READY` = 2 , `SPI_ACTIVE` = 3 ,
`SPI_COMPLETE` = 4 }

Driver state machine possible states.

Functions

- void `spiInit` (void)
SPI Driver initialization.
- void `spiObjectInit` (`SPIDriver` *`spip`)
Initializes the standard part of a `SPIDriver` structure.
- `msg_t` `spiStart` (`SPIDriver` *`spip`, const `SPIConfig` *`config`)
Configures and activates the SPI peripheral.
- void `spiStop` (`SPIDriver` *`spip`)
Deactivates the SPI peripheral.
- void `spiSelect` (`SPIDriver` *`spip`)
Asserts the slave select signal and prepares for transfers.
- void `spiUnselect` (`SPIDriver` *`spip`)
Deasserts the slave select signal.
- void `spiStartIgnore` (`SPIDriver` *`spip`, `size_t` `n`)
Ignores data on the SPI bus.
- void `spiStartExchange` (`SPIDriver` *`spip`, `size_t` `n`, const void *`txbuf`, void *`rxbuf`)
Exchanges data on the SPI bus.
- void `spiStartSend` (`SPIDriver` *`spip`, `size_t` `n`, const void *`txbuf`)
Sends data over the SPI bus.
- void `spiStartReceive` (`SPIDriver` *`spip`, `size_t` `n`, void *`rxbuf`)
Receives data from the SPI bus.
- void `spiAbortl` (`SPIDriver` *`spip`)
Aborts the ongoing SPI operation.
- void `spiAbort` (`SPIDriver` *`spip`)
Aborts the ongoing SPI operation, if any.
- void `spiIgnore` (`SPIDriver` *`spip`, `size_t` `n`)
Ignores data on the SPI bus.
- void `spiExchange` (`SPIDriver` *`spip`, `size_t` `n`, const void *`txbuf`, void *`rxbuf`)
Exchanges data on the SPI bus.
- void `spiSend` (`SPIDriver` *`spip`, `size_t` `n`, const void *`txbuf`)
Sends data over the SPI bus.
- void `spiReceive` (`SPIDriver` *`spip`, `size_t` `n`, void *`rxbuf`)
Receives data from the SPI bus.
- void `spiAcquireBus` (`SPIDriver` *`spip`)
Gains exclusive access to the SPI bus.
- void `spiReleaseBus` (`SPIDriver` *`spip`)
Releases exclusive access to the SPI bus.

9.76.1 Detailed Description

SPI (v1) Driver macros and structures.

9.77 hal_spi_v2.h File Reference

SPI (v2) Driver macros and structures.

```
#include "hal_spi_v2_lld.h"
```

Data Structures

- struct [hal_spi_config](#)
Driver configuration structure.
- struct [hal_spi_driver](#)
Structure representing an SPI driver.

Macros

Chip Select modes

- #define [SPI_SELECT_MODE_NONE](#)
- #define [SPI_SELECT_MODE_PAD](#) 1 /** @brief Legacy mode. */
- #define [SPI_SELECT_MODE_PORT](#) 2 /** @brief Fastest mode. */
- #define [SPI_SELECT_MODE_LINE](#) 3 /** @brief Packed mode. */
- #define [SPI_SELECT_MODE_LLD](#) 4 /** @brief LLD-defined mode.*/

SPI configuration options

- #define [SPI_USE_SYNCHRONIZATION](#) FALSE
Support for thread synchronization API.
- #define [SPI_USE_ASSERT_ON_ERROR](#) TRUE
Inserts an assertion on function errors before returning.
- #define [SPI_USE_MUTUAL_EXCLUSION](#) TRUE
Enables the [spiAcquireBus\(\)](#) and [spiReleaseBus\(\)](#) APIs.
- #define [SPI_SELECT_MODE](#) [SPI_SELECT_MODE_PAD](#)
Handling method for SPI CS line.

Macro Functions

- #define [spilsBufferComplete](#)(spip)
Buffer state.
- #define [spiSelectl](#)(spip)
Asserts the slave select signal and prepares for transfers.
- #define [spiUnselectl](#)(spip)
Deasserts the slave select signal.
- #define [spiPolledExchange](#)(spip, frame)
Exchanges one frame using a polled wait.
- #define [spiAbortl](#)(spip)
Compatibility API with SPI driver v1.
- #define [spiAbort](#)(spip)
Compatibility API with SPI driver v1.

Low level driver helper macros

- #define [__spi_wakeup_isr](#)(spip, msg)
Wakes up the waiting thread.
- #define [__spi_isr_complete_code](#)(spip)
Common ISR code in linear mode.
- #define [__spi_isr_half_code](#)(spip)
Half buffer filled ISR code in circular mode.
- #define [__spi_isr_full_code](#)(spip)
Full buffer filled ISR code in circular mode.
- #define [__spi_isr_error_code](#)(spip, msg)
ISR error reporting code..

Typedefs

- typedef struct [hal_spi_driver](#) SPIDriver
Type of a structure representing an SPI driver.
- typedef struct [hal_spi_config](#) SPIConfig
Type of a SPI driver configuration structure.
- typedef void(* [spicb_t](#)) (SPIDriver *spip)
SPI notification callback type.

Enumerations

- enum [spistate_t](#) {
 [SPI_UNINIT](#) = 0 , [SPI_STOP](#) = 1 , [SPI_READY](#) = 2 , [SPI_ACTIVE](#) = 3 ,
 [SPI_COMPLETE](#) = 4 }
Driver state machine possible states.

Functions

- void [spilnit](#) (void)
SPI Driver initialization.
- void [spiObjectInit](#) (SPIDriver *spip)
Initializes the standard part of a [SPIDriver](#) structure.
- [msg_t](#) [spiStart](#) (SPIDriver *spip, const SPIConfig *config)
Configures and activates the SPI peripheral.
- void [spiStop](#) (SPIDriver *spip)
Deactivates the SPI peripheral.
- void [spiSelect](#) (SPIDriver *spip)
Asserts the slave select signal and prepares for transfers.
- void [spiUnselect](#) (SPIDriver *spip)
Deasserts the slave select signal.
- [msg_t](#) [spiStartIgnoreI](#) (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- [msg_t](#) [spiStartIgnoreO](#) (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- [msg_t](#) [spiStartExchangeI](#) (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- [msg_t](#) [spiStartExchangeO](#) (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- [msg_t](#) [spiStartSendI](#) (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- [msg_t](#) [spiStartSendO](#) (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- [msg_t](#) [spiStartReceiveI](#) (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- [msg_t](#) [spiStartReceiveO](#) (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- [msg_t](#) [spiStopTransferI](#) (SPIDriver *spip, size_t *sizep)
Stops the ongoing SPI operation.
- [msg_t](#) [spiStopTransferO](#) (SPIDriver *spip, size_t *sizep)
Stops the ongoing SPI operation, if any.

- [msg_t spiSynchronizeS](#) ([SPIDriver](#) *spip, [sysinterval_t](#) timeout)
Synchronizes with current transfer completion.
- [msg_t spiSynchronize](#) ([SPIDriver](#) *spip, [sysinterval_t](#) timeout)
Synchronizes with current transfer completion.
- [msg_t spiIgnore](#) ([SPIDriver](#) *spip, [size_t](#) n)
Ignores data on the SPI bus.
- [msg_t spiExchange](#) ([SPIDriver](#) *spip, [size_t](#) n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- [msg_t spiSend](#) ([SPIDriver](#) *spip, [size_t](#) n, const void *txbuf)
Sends data over the SPI bus.
- [msg_t spiReceive](#) ([SPIDriver](#) *spip, [size_t](#) n, void *rxbuf)
Receives data from the SPI bus.
- void [spiAcquireBus](#) ([SPIDriver](#) *spip)
Gains exclusive access to the SPI bus.
- void [spiReleaseBus](#) ([SPIDriver](#) *spip)
Releases exclusive access to the SPI bus.

9.77.1 Detailed Description

SPI (v2) Driver macros and structures.

9.78 hal_st.h File Reference

ST Driver macros and structures.

```
#include "hal_st_llld.h"
```

Macros

- `#define ST\_LLD\_NUM\_ALARMS 1`

Typedefs

- typedef void(* [st_callback_t](#)) (unsigned alarm)

Functions

- void [stInit](#) (void)
ST Driver initialization.
- [sys_time_t](#) [stGetCounter](#) (void)
Returns the time counter value.
- void [stStartAlarm](#) ([sys_time_t](#) abstime)
Starts the alarm zero.
- void [stStopAlarm](#) (void)
Stops the alarm zero interrupt.
- void [stSetAlarm](#) ([sys_time_t](#) abstime)
Sets the alarm zero time.
- [sys_time_t](#) [stGetAlarm](#) (void)
Returns the alarm zero current time.
- bool [stIsAlarmActive](#) (void)
Determines if the alarm zero is active.

9.78.1 Detailed Description

ST Driver macros and structures.

This header is designed to be include-able without having to include other files from the HAL.

9.79 hal_streams.h File Reference

Data streams.

Data Structures

- struct [BaseSequentialStreamVMT](#)
BaseSequentialStream virtual methods table.
- struct [BaseSequentialStream](#)
Base stream class.
- struct [BaseBufferedStreamVMT](#)
BaseBufferedStream virtual methods table.
- struct [BaseBufferedStream](#)
Buffered stream class.

Macros

- #define [_base_sequential_stream_methods](#)
BaseSequentialStream specific methods.
- #define [_base_sequential_stream_data _base_object_data](#)
BaseSequentialStream specific data.
- #define [_base_buffered_stream_methods](#)
BaseBufferedStream specific methods.
- #define [_base_buffered_stream_data _base_sequential_stream_data](#)
BaseBufferedStream specific data.

Streams return codes

- #define [STM_OK MSG_OK](#)
- #define [STM_TIMEOUT MSG_TIMEOUT](#)
- #define [STM_RESET MSG_RESET](#)

Macro Functions (BaseSequentialStream)

- #define [streamWrite](#)(ip, bp, n)
Sequential Stream write.
- #define [streamRead](#)(ip, bp, n)
Sequential Stream read.
- #define [streamPut](#)(ip, b)
Sequential Stream blocking byte write.
- #define [streamGet](#)(ip)
Sequential Stream blocking byte read.

Macro Functions (BaseBufferedStream)

- #define [streamUnget](#)(ip, b)
Buffered Stream unget.

9.79.1 Detailed Description

Data streams.

This header defines abstract interfaces useful to access generic data streams in a standardized way.

9.80 hal_trng.h File Reference

TRNG Driver macros and structures.

```
#include "hal_trng_lld.h"
```

Data Structures

- struct [hal_trng_config](#)
Driver configuration structure.
- struct [hal_trng_driver](#)
Structure representing a TRNG driver.

Typedefs

- typedef struct [hal_trng_driver](#) TRNGDriver
Type of a structure representing a TRNG driver.
- typedef struct [hal_trng_config](#) TRNGConfig
Driver configuration structure.

Enumerations

- enum [trngstate_t](#) { [TRNG_UNINIT](#) = 0 , [TRNG_STOP](#) = 1 , [TRNG_READY](#) = 2 , [TRNG_RUNNING](#) = 3 }
Driver state machine possible states.

Functions

- void [trngInit](#) (void)
TRNG Driver initialization.
- void [trngObjectInit](#) (TRNGDriver *trngp)
Initializes the standard part of a [TRNGDriver](#) structure.
- [msg_t](#) [trngStart](#) (TRNGDriver *trngp, const TRNGConfig *config)
Configures and activates the TRNG peripheral.
- void [trngStop](#) (TRNGDriver *trngp)
Deactivates the TRNG peripheral.
- bool [trngGenerate](#) (TRNGDriver *trngp, size_t size, uint8_t *out)
True random numbers generator.

9.80.1 Detailed Description

TRNG Driver macros and structures.

9.81 hal_uart.h File Reference

UART Driver macros and structures.

```
#include "hal_uart_ll.h"
```

Macros

UART status flags

- `#define UART_NO_ERROR 0`
No pending conditions.
- `#define UART_PARITY_ERROR 4`
Parity error happened.
- `#define UART_FRAMING_ERROR 8`
Framing error happened.
- `#define UART_OVERRUN_ERROR 16`
Overflow happened.
- `#define UART_NOISE_ERROR 32`
Noise on the line.
- `#define UART_BREAK_DETECTED 64`
Break detected.

UART error conditions

- `#define UART_ERR_NOT_ACTIVE (size_t)-1`

UART configuration options

- `#define UART_USE_WAIT FALSE`
Enables synchronous APIs.
- `#define UART_USE_MUTUAL_EXCLUSION FALSE`
Enables the `uartAcquireBus()` and `uartReleaseBus()` APIs.

Low level driver helper macros

- `#define _uart_wakeup_tx1_isr(uartp)`
Wakes up the waiting thread in case of early TX complete.
- `#define _uart_wakeup_tx2_isr(uartp)`
Wakes up the waiting thread in case of late TX complete.
- `#define _uart_wakeup_rx_complete_isr(uartp)`
Wakes up the waiting thread in case of RX complete.
- `#define _uart_wakeup_rx_error_isr(uartp)`
Wakes up the waiting thread in case of RX error.
- `#define _uart_wakeup_rx_cm_isr(uartp)`
Wakes up the waiting thread in case of RX character match.
- `#define _uart_wakeup_rx_timeout_isr(uartp)`
Wakes up the waiting thread in case of RX timeout.

- `#define _uart_tx1_isr_code(uartp)`
Common ISR code for early TX.
- `#define _uart_tx2_isr_code(uartp)`
Common ISR code for late TX.
- `#define _uart_rx_complete_isr_code(uartp)`
Common ISR code for RX complete.
- `#define _uart_rx_error_isr_code(uartp, errors)`
Common ISR code for RX error.
- `#define _uart_rx_idle_code(uartp)`
Common ISR code for RX on idle.
- `#define _uart_timeout_isr_code(uartp)`
Timeout ISR code for receiver.
- `#define _uart_rx_char_match_isr_code(uartp)`
Character match ISR code for receiver.

Enumerations

- enum `uartstate_t` { `UART_UNINIT` = 0 , `UART_STOP` = 1 , `UART_READY` = 2 }
Driver state machine possible states.
- enum `uarttxstate_t` { `UART_TX_IDLE` = 0 , `UART_TX_ACTIVE` = 1 , `UART_TX_COMPLETE` = 2 }
Transmitter state machine states.
- enum `uartrxstate_t` { `UART_RX_IDLE` = 0 , `UART_RX_ACTIVE` = 1 , `UART_RX_COMPLETE` = 2 }
Receiver state machine states.

Functions

- void `uartInit` (void)
UART Driver initialization.
- void `uartObjectInit` (`UARTDriver` *uartp)
Initializes the standard part of a `UARTDriver` structure.
- `msg_t` `uartStart` (`UARTDriver` *uartp, const `UARTConfig` *config)
Configures and activates the UART peripheral.
- void `uartStop` (`UARTDriver` *uartp)
Deactivates the UART peripheral.
- void `uartStartSend` (`UARTDriver` *uartp, `size_t` n, const void *txbuf)
Starts a transmission on the UART peripheral.
- void `uartStartSendI` (`UARTDriver` *uartp, `size_t` n, const void *txbuf)
Starts a transmission on the UART peripheral.
- `size_t` `uartStopSend` (`UARTDriver` *uartp)
Stops any ongoing transmission.
- `size_t` `uartStopSendI` (`UARTDriver` *uartp)
Stops any ongoing transmission.
- void `uartStartReceive` (`UARTDriver` *uartp, `size_t` n, void *rxbuf)
Starts a receive operation on the UART peripheral.
- void `uartStartReceiveI` (`UARTDriver` *uartp, `size_t` n, void *rxbuf)
Starts a receive operation on the UART peripheral.
- `size_t` `uartStopReceive` (`UARTDriver` *uartp)
Stops any ongoing receive operation.
- `size_t` `uartStopReceiveI` (`UARTDriver` *uartp)
Stops any ongoing receive operation.
- `msg_t` `uartSendTimeout` (`UARTDriver` *uartp, `size_t` *np, const void *txbuf, `sysinterval_t` timeout)

- Performs a transmission on the UART peripheral.*
- `msg_t uartSendFullTimeout` (`UARTDriver *uartp`, `size_t *np`, `const void *txbuf`, `sysinterval_t timeout`)
Performs a transmission on the UART peripheral.
- `msg_t uartReceiveTimeout` (`UARTDriver *uartp`, `size_t *np`, `void *rxbuf`, `sysinterval_t timeout`)
Performs a receive operation on the UART peripheral.
- `void uartAcquireBus` (`UARTDriver *uartp`)
Gains exclusive access to the UART bus.
- `void uartReleaseBus` (`UARTDriver *uartp`)
Releases exclusive access to the UART bus.

9.81.1 Detailed Description

UART Driver macros and structures.

9.82 hal_usb.h File Reference

USB Driver macros and structures.

```
#include "hal_usb_lld.h"
```

Data Structures

- struct `USBDescriptor`
Type of an USB descriptor.

Macros

- `#define USB_ENDPOINT_OUT(ep)`
- `#define USB_ENDPOINT_IN(ep)`
- `#define USB_RTYPE_DIR_MASK 0x80U`
- `#define USB_RTYPE_DIR_HOST2DEV 0x00U`
- `#define USB_RTYPE_DIR_DEV2HOST 0x80U`
- `#define USB_RTYPE_TYPE_MASK 0x60U`
- `#define USB_RTYPE_TYPE_STD 0x00U`
- `#define USB_RTYPE_TYPE_CLASS 0x20U`
- `#define USB_RTYPE_TYPE_VENDOR 0x40U`
- `#define USB_RTYPE_TYPE_RESERVED 0x60U`
- `#define USB_RTYPE_RECIPIENT_MASK 0x1FU`
- `#define USB_RTYPE_RECIPIENT_DEVICE 0x00U`
- `#define USB_RTYPE_RECIPIENT_INTERFACE 0x01U`
- `#define USB_RTYPE_RECIPIENT_ENDPOINT 0x02U`
- `#define USB_RTYPE_RECIPIENT_OTHER 0x03U`
- `#define USB_REQ_GET_STATUS 0U`
- `#define USB_REQ_CLEAR_FEATURE 1U`
- `#define USB_REQ_SET_FEATURE 3U`
- `#define USB_REQ_SET_ADDRESS 5U`
- `#define USB_REQ_GET_DESCRIPTOR 6U`
- `#define USB_REQ_SET_DESCRIPTOR 7U`

- #define [USB_REQ_GET_CONFIGURATION](#) 8U
- #define [USB_REQ_SET_CONFIGURATION](#) 9U
- #define [USB_REQ_GET_INTERFACE](#) 10U
- #define [USB_REQ_SET_INTERFACE](#) 11U
- #define [USB_REQ_SYNCH_FRAME](#) 12U
- #define [USB_DESCRIPTOR_DEVICE](#) 1U
- #define [USB_DESCRIPTOR_CONFIGURATION](#) 2U
- #define [USB_DESCRIPTOR_STRING](#) 3U
- #define [USB_DESCRIPTOR_INTERFACE](#) 4U
- #define [USB_DESCRIPTOR_ENDPOINT](#) 5U
- #define [USB_DESCRIPTOR_DEVICE_QUALIFIER](#) 6U
- #define [USB_DESCRIPTOR_OTHER_SPEED_CFG](#) 7U
- #define [USB_DESCRIPTOR_INTERFACE_POWER](#) 8U
- #define [USB_DESCRIPTOR_INTERFACE_ASSOCIATION](#) 11U
- #define [USB_FEATURE_ENDPOINT_HALT](#) 0U
- #define [USB_FEATURE_DEVICE_REMOTE_WAKEUP](#) 1U
- #define [USB_FEATURE_TEST_MODE](#) 2U
- #define [USB_EARLY_SET_ADDRESS](#) 0
- #define [USB_LATE_SET_ADDRESS](#) 1
- #define [USB_EP0_STATUS_STAGE_SW](#) 0
- #define [USB_EP0_STATUS_STAGE_HW](#) 1
- #define [USB_SET_ADDRESS_ACK_SW](#) 0
- #define [USB_SET_ADDRESS_ACK_HW](#) 1
- #define [USB_IN_STATE](#) 0x08U
- #define [USB_OUT_STATE](#) 0x10U
- #define [USB_USE_WAIT](#) FALSE

Enables synchronous APIs.

Helper macros for USB descriptors

- #define [USB_DESC_INDEX](#)(i)
Helper macro for index values into descriptor strings.
- #define [USB_DESC_BYTE](#)(b)
Helper macro for byte values into descriptor strings.
- #define [USB_DESC_WORD](#)(w)
Helper macro for word values into descriptor strings.
- #define [USB_DESC_BCD](#)(bcd)
Helper macro for BCD values into descriptor strings.
- #define [USB_DESC_DEVICE_SIZE](#) 18U
- #define [USB_DESC_DEVICE](#)(bcdUSB, bDeviceClass, bDeviceSubClass, bDeviceProtocol, bMaxPacketSize, idVendor, idProduct, bcdDevice, iManufacturer, iProduct, iSerialNumber, bNumConfigurations)
Device Descriptor helper macro.
- #define [USB_DESC_CONFIGURATION_SIZE](#) 9U
Configuration Descriptor size.
- #define [USB_DESC_CONFIGURATION](#)(wTotalLength, bNumInterfaces, bConfigurationValue, iConfiguration, bmAttributes, bMaxPower)
Configuration Descriptor helper macro.
- #define [USB_DESC_INTERFACE_SIZE](#) 9U
Interface Descriptor size.
- #define [USB_DESC_INTERFACE](#)(bInterfaceNumber, bAlternateSetting, bNumEndpoints, bInterfaceClass, bInterfaceSubClass, bInterfaceProtocol, iInterface)
Interface Descriptor helper macro.
- #define [USB_DESC_INTERFACE_ASSOCIATION_SIZE](#) 8U
Interface Association Descriptor size.

- #define [USB_DESC_INTERFACE_ASSOCIATION](#)(bFirstInterface, bInterfaceCount, bFunctionClass, b↵
FunctionSubClass, bFunctionProcotol, iInterface)
Interface Association Descriptor helper macro.
- #define [USB_DESC_ENDPOINT_SIZE](#) 7U
Endpoint Descriptor size.
- #define [USB_DESC_ENDPOINT](#)(bEndpointAddress, bmAttributes, wMaxPacketSize, bInterval)
Endpoint Descriptor helper macro.

Endpoint types and settings

- #define [USB_EP_MODE_TYPE](#) 0x0003U
- #define [USB_EP_MODE_TYPE_CTRL](#) 0x0000U
- #define [USB_EP_MODE_TYPE_ISOC](#) 0x0001U
- #define [USB_EP_MODE_TYPE_BULK](#) 0x0002U
- #define [USB_EP_MODE_TYPE_INTR](#) 0x0003U

Macro Functions

- #define [usbGetDriverStatel](#)(usbp)
Returns the driver state.
- #define [usbConnectBus](#)(usbp)
Connects the USB device.
- #define [usbDisconnectBus](#)(usbp)
Disconnect the USB device.
- #define [usbGetFrameNumberX](#)(usbp)
Returns the current frame number.
- #define [usbGetTransmitStatusl](#)(usbp, ep)
Returns the status of an IN endpoint.
- #define [usbGetReceiveStatusl](#)(usbp, ep)
Returns the status of an OUT endpoint.
- #define [usbGetReceiveTransactionSizeX](#)(usbp, ep)
Returns the exact size of a receive transaction.
- #define [usbSetupTransfer](#)(usbp, buf, n, endcb)
Request transfer setup.
- #define [usbReadSetup](#)(usbp, ep, buf)
Reads a setup packet from the dedicated packet buffer.

Low level driver helper macros

- #define [_usb_isr_invoke_event_cb](#)(usbp, evt)
Common ISR code, usb event callback.
- #define [_usb_isr_invoke_sof_cb](#)(usbp)
Common ISR code, SOF callback.
- #define [_usb_isr_invoke_setup_cb](#)(usbp, ep)
Common ISR code, setup packet callback.
- #define [_usb_isr_invoke_in_cb](#)(usbp, ep)
Common ISR code, IN endpoint callback.
- #define [_usb_isr_invoke_out_cb](#)(usbp, ep)
Common ISR code, OUT endpoint event.

Typedefs

- typedef struct USBDriver [USBDriver](#)
Type of a structure representing an USB driver.
- typedef uint8_t [usbep_t](#)
Type of an endpoint identifier.
- typedef void(* [usbcallback_t](#)) ([USBDriver](#) *usbp)
Type of an USB generic notification callback.
- typedef void(* [usbepcallback_t](#)) ([USBDriver](#) *usbp, [usbep_t](#) ep)
Type of an USB endpoint callback.
- typedef void(* [usbeventcb_t](#)) ([USBDriver](#) *usbp, [usbevent_t](#) event)
Type of an USB event notification callback.
- typedef bool(* [usbreqhandler_t](#)) ([USBDriver](#) *usbp)
Type of a requests handler callback.
- typedef const [USBDescriptor](#) *(* [usbgetdescriptor_t](#)) ([USBDriver](#) *usbp, uint8_t dtype, uint8_t dindex, uint16_t lang)
Type of an USB descriptor-retrieving callback.

Enumerations

- enum [usbstate_t](#) {
 [USB_UNINIT](#) = 0 , [USB_STOP](#) = 1 , [USB_READY](#) = 2 , [USB_SELECTED](#) = 3 ,
 [USB_ACTIVE](#) = 4 , [USB_SUSPENDED](#) = 5 }
Type of a driver state machine possible states.
- enum [usbepstatus_t](#) { [EP_STATUS_DISABLED](#) = 0 , [EP_STATUS_STALLED](#) = 1 , [EP_STATUS_ACTIVE](#) = 2 }
Type of an endpoint status.
- enum [usbep0state_t](#) {
 [USB_EP0_STP_WAITING](#) = 0U , [USB_EP0_IN_TX](#) = [USB_IN_STATE](#) | 1U , [USB_EP0_IN_WAITING_TX0](#)
 = [USB_IN_STATE](#) | 2U , [USB_EP0_IN_SENDING_STS](#) = [USB_IN_STATE](#) | 3U ,
 [USB_EP0_OUT_WAITING_STS](#) = [USB_OUT_STATE](#) | 4U , [USB_EP0_OUT_RX](#) = [USB_OUT_STATE](#) | 5U
 , [USB_EP0_ERROR](#) = 6U }
Type of an endpoint zero state machine states.
- enum [usbevent_t](#) {
 [USB_EVENT_RESET](#) = 0 , [USB_EVENT_ADDRESS](#) = 1 , [USB_EVENT_CONFIGURED](#) = 2 ,
 [USB_EVENT_UNCONFIGURED](#) = 3 ,
 [USB_EVENT_SUSPEND](#) = 4 , [USB_EVENT_WAKEUP](#) = 5 , [USB_EVENT_STALLED](#) = 6 }
Type of an enumeration of the possible USB events.

Functions

- void [usbInit](#) (void)
USB Driver initialization.
- void [usbObjectInit](#) ([USBDriver](#) *usbp)
Initializes the standard part of a [USBDriver](#) structure.
- [msg_t](#) [usbStart](#) ([USBDriver](#) *usbp, const [USBConfig](#) *config)
Configures and activates the USB peripheral.
- void [usbStop](#) ([USBDriver](#) *usbp)
Deactivates the USB peripheral.
- void [usbInitEndpointI](#) ([USBDriver](#) *usbp, [usbep_t](#) ep, const [USBEndpointConfig](#) *epcp)
Enables an endpoint.

- void `usbDisableEndpointsI` (`USBDriver *usbp`)
Disables all the active endpoints.
- void `usbReadSetupI` (`USBDriver *usbp`, `usbep_t ep`, `uint8_t *buf`)
- void `usbStartReceiveI` (`USBDriver *usbp`, `usbep_t ep`, `uint8_t *buf`, `size_t n`)
Starts a receive transaction on an OUT endpoint.
- void `usbStartTransmitI` (`USBDriver *usbp`, `usbep_t ep`, `const uint8_t *buf`, `size_t n`)
Starts a transmit transaction on an IN endpoint.
- `msg_t usbReceive` (`USBDriver *usbp`, `usbep_t ep`, `uint8_t *buf`, `size_t n`)
Performs a receive transaction on an OUT endpoint.
- `msg_t usbTransmit` (`USBDriver *usbp`, `usbep_t ep`, `const uint8_t *buf`, `size_t n`)
Performs a transmit transaction on an IN endpoint.
- bool `usbStallReceiveI` (`USBDriver *usbp`, `usbep_t ep`)
Stalls an OUT endpoint.
- bool `usbStallTransmitI` (`USBDriver *usbp`, `usbep_t ep`)
Stalls an IN endpoint.
- void `usbWakeupHost` (`USBDriver *usbp`)
Host wake-up procedure.
- void `_usb_reset` (`USBDriver *usbp`)
USB reset routine.
- void `_usb_suspend` (`USBDriver *usbp`)
USB suspend routine.
- void `_usb_wakeup` (`USBDriver *usbp`)
USB wake-up routine.
- void `_usb_ep0setup` (`USBDriver *usbp`, `usbep_t ep`)
Default EP0 SETUP callback.
- void `_usb_ep0in` (`USBDriver *usbp`, `usbep_t ep`)
Default EP0 IN callback.
- void `_usb_ep0out` (`USBDriver *usbp`, `usbep_t ep`)
Default EP0 OUT callback.

9.82.1 Detailed Description

USB Driver macros and structures.

9.83 hal_usb_cdc.h File Reference

USB CDC macros and structures.

Data Structures

- struct `cdc_linecoding_t`
Type of Line Coding structure.

Macros

CDC specific messages.

- #define [CDC_SEND_ENCAPSULATED_COMMAND](#) 0x00U
- #define [CDC_GET_ENCAPSULATED_RESPONSE](#) 0x01U
- #define [CDC_SET_COMM_FEATURE](#) 0x02U
- #define [CDC_GET_COMM_FEATURE](#) 0x03U
- #define [CDC_CLEAR_COMM_FEATURE](#) 0x04U
- #define [CDC_SET_AUX_LINE_STATE](#) 0x10U
- #define [CDC_SET_HOOK_STATE](#) 0x11U
- #define [CDC_PULSE_SETUP](#) 0x12U
- #define [CDC_SEND_PULSE](#) 0x13U
- #define [CDC_SET_PULSE_TIME](#) 0x14U
- #define [CDC_RING_AUX_JACK](#) 0x15U
- #define [CDC_SET_LINE_CODING](#) 0x20U
- #define [CDC_GET_LINE_CODING](#) 0x21U
- #define [CDC_SET_CONTROL_LINE_STATE](#) 0x22U
- #define [CDC_SEND_BREAK](#) 0x23U
- #define [CDC_SET_RINGER_PARMS](#) 0x30U
- #define [CDC_GET_RINGER_PARMS](#) 0x31U
- #define [CDC_SET_OPERATION_PARMS](#) 0x32U
- #define [CDC_GET_OPERATION_PARMS](#) 0x33U

CDC classes

- #define [CDC_COMMUNICATION_INTERFACE_CLASS](#) 0x02U
- #define [CDC_DATA_INTERFACE_CLASS](#) 0x0AU

CDC subclasses

- #define [CDC_ABSTRACT_CONTROL_MODEL](#) 0x02U

CDC descriptors

- #define [CDC_CS_INTERFACE](#) 0x24U

CDC subdescriptors

- #define [CDC_HEADER](#) 0x00U
- #define [CDC_CALL_MANAGEMENT](#) 0x01U
- #define [CDC_ABSTRACT_CONTROL_MANAGEMENT](#) 0x02U
- #define [CDC_UNION](#) 0x06U

Line Control bit definitions.

- #define [LC_STOP_1](#) 0U
- #define [LC_STOP_1P5](#) 1U
- #define [LC_STOP_2](#) 2U
- #define [LC_PARITY_NONE](#) 0U
- #define [LC_PARITY_ODD](#) 1U
- #define [LC_PARITY_EVEN](#) 2U
- #define [LC_PARITY_MARK](#) 3U
- #define [LC_PARITY_SPACE](#) 4U

9.83.1 Detailed Description

USB CDC macros and structures.

9.84 hal_wdg.h File Reference

WDG Driver macros and structures.

```
#include "hal_wdg_lld.h"
```

Macros

- #define [wdgResetl](#)(wdgp)
Resets WDG's counter.

Enumerations

- enum [wdgstate_t](#) { [WDG_UNINIT](#) = 0 , [WDG_STOP](#) = 1 , [WDG_READY](#) = 2 }
Driver state machine possible states.

Functions

- void [wdgInit](#) (void)
WDG Driver initialization.
- [msg_t](#) [wdgStart](#) ([WDGDriver](#) *wdgp, const [WDGConfig](#) *config)
Configures and activates the WDG peripheral.
- void [wdgStop](#) ([WDGDriver](#) *wdgp)
Deactivates the WDG peripheral.
- void [wdgReset](#) ([WDGDriver](#) *wdgp)
Resets WDG's counter.

9.84.1 Detailed Description

WDG Driver macros and structures.

9.85 hal_wspi.h File Reference

WSPI Driver macros and structures.

```
#include "hal_wspi_lld.h"
```

Data Structures

- struct [wspi_command_t](#)
Type of a WSPI command descriptor.
- struct [hal_wspi_config](#)
Driver configuration structure.
- struct [hal_wspi_driver](#)
Structure representing an WSPI driver.

Macros

WSPI configuration options

- #define `WSPI_USE_WAIT TRUE`
Enables synchronous APIs.
- #define `WSPI_USE_MUTUAL_EXCLUSION TRUE`
Enables the `wspiAcquireBus()` and `wspiReleaseBus()` APIs.

Transfer options

Note

The low level driver has the option to override the following definitions and use its own ones. In must take care to use the same name for the same function or compatibility is not ensured.

- #define `WSPI_CFG_CMD_MODE_MASK` (7LU << 0LU)
- #define `WSPI_CFG_CMD_MODE_NONE` (0LU << 0LU)
- #define `WSPI_CFG_CMD_MODE_ONE_LINE` (1LU << 0LU)
- #define `WSPI_CFG_CMD_MODE_TWO_LINES` (2LU << 0LU)
- #define `WSPI_CFG_CMD_MODE_FOUR_LINES` (3LU << 0LU)
- #define `WSPI_CFG_CMD_MODE_EIGHT_LINES` (4LU << 0LU)
- #define `WSPI_CFG_CMD_DTR` (1LU << 3LU)
- #define `WSPI_CFG_CMD_SIZE_MASK` (3LU << 4LU)
- #define `WSPI_CFG_CMD_SIZE_8` (0LU << 4LU)
- #define `WSPI_CFG_CMD_SIZE_16` (1LU << 4LU)
- #define `WSPI_CFG_CMD_SIZE_24` (2LU << 4LU)
- #define `WSPI_CFG_CMD_SIZE_32` (3LU << 4LU)
- #define `WSPI_CFG_ADDR_MODE_MASK` (7LU << 8LU)
- #define `WSPI_CFG_ADDR_MODE_NONE` (0LU << 8LU)
- #define `WSPI_CFG_ADDR_MODE_ONE_LINE` (1LU << 8LU)
- #define `WSPI_CFG_ADDR_MODE_TWO_LINES` (2LU << 8LU)
- #define `WSPI_CFG_ADDR_MODE_FOUR_LINES` (3LU << 8LU)
- #define `WSPI_CFG_ADDR_MODE_EIGHT_LINES` (4LU << 8LU)
- #define `WSPI_CFG_ADDR_DTR` (1LU << 11LU)
- #define `WSPI_CFG_ADDR_SIZE_MASK` (3LU << 12LU)
- #define `WSPI_CFG_ADDR_SIZE_8` (0LU << 12LU)
- #define `WSPI_CFG_ADDR_SIZE_16` (1LU << 12LU)
- #define `WSPI_CFG_ADDR_SIZE_24` (2LU << 12LU)
- #define `WSPI_CFG_ADDR_SIZE_32` (3LU << 12LU)
- #define `WSPI_CFG_ALT_MODE_MASK` (7LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_NONE` (0LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_ONE_LINE` (1LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_TWO_LINES` (2LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_FOUR_LINES` (3LU << 16LU)
- #define `WSPI_CFG_ALT_MODE_EIGHT_LINES` (4LU << 16LU)
- #define `WSPI_CFG_ALT_DTR` (1LU << 19LU)
- #define `WSPI_CFG_ALT_SIZE_MASK` (3LU << 20LU)
- #define `WSPI_CFG_ALT_SIZE_8` (0LU << 20LU)
- #define `WSPI_CFG_ALT_SIZE_16` (1LU << 20LU)
- #define `WSPI_CFG_ALT_SIZE_24` (2LU << 20LU)
- #define `WSPI_CFG_ALT_SIZE_32` (3LU << 20LU)
- #define `WSPI_CFG_DATA_MODE_MASK` (7LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_NONE` (0LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_ONE_LINE` (1LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_TWO_LINES` (2LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_FOUR_LINES` (3LU << 24LU)
- #define `WSPI_CFG_DATA_MODE_EIGHT_LINES` (4LU << 24LU)
- #define `WSPI_CFG_DATA_DTR` (1LU << 27LU)
- #define `WSPI_CFG_DQS_ENABLE` (1LU << 29LU)
- #define `WSPI_CFG_SIOO` (1LU << 31LU)
- #define `WSPI_CFG_ALL_DTR`

Macro Functions

- #define `wspiStartCommandI(wspip, cmdp)`
Sends a command without data phase.
- #define `wspiStartSendI(wspip, cmdp, n, txbuf)`
Sends data over the WSPI bus.
- #define `wspiStartReceiveI(wspip, cmdp, n, rxbuf)`
Receives data from the WSPI bus.
- #define `wspiMapFlashI(wspip, cmdp, addrp)`
Maps in memory space a WSPI flash device.
- #define `wspiUnmapFlashI(wspip)`
Maps in memory space a WSPI flash device.

Low level driver helper macros

- #define `_wspi_wakeup_isr(wspip, msg)`
Wakes up the waiting thread.
- #define `_wspi_isr_code(wspip)`
Common ISR code.
- #define `_wspi_error_code(wspip)`
Common error ISR code.

Typedefs

- typedef struct `hal_wspi_driver WSPIDriver`
Type of a structure representing an WSPI driver.
- typedef struct `hal_wspi_config WSPIDriverConfig`
Type of a structure representing an WSPI driver configuration.
- typedef void(* `wspicallback_t`) (`WSPIDriver *wspip`)
Type of a WSPI notification callback.

Enumerations

- enum `wspistate_t` {
`WSPIDriverConfig` = 0 , `WSPIDriverConfig` = 1 , `WSPIDriverConfig` = 2 , `WSPIDriverConfig` = 3 ,
`WSPIDriverConfig` = 4 , `WSPIDriverConfig` = 5 , `WSPIDriverConfig` = 6 }
Driver state machine possible states.

Functions

- void `wspiInit` (void)
WSPI Driver initialization.
- void `wspiObjectInit` (`WSPIDriver *wspip`)
Initializes the standard part of a `WSPIDriver` structure.
- `msg_t` `wspiStart` (`WSPIDriver *wspip`, const `WSPIDriverConfig *config`)
Configures and activates the WSPI peripheral.
- void `wspiStop` (`WSPIDriver *wspip`)
Deactivates the WSPI peripheral.
- void `wspiStartCommand` (`WSPIDriver *wspip`, const `wspi_command_t *cmdp`)
Sends a command without data phase.
- void `wspiStartSend` (`WSPIDriver *wspip`, const `wspi_command_t *cmdp`, size_t n, const uint8_t *txbuf)
Sends a command with data over the WSPI bus.

- void [wspiStartReceive](#) ([WSPIDriver](#) *wspip, const [wspi_command_t](#) *cmdp, size_t n, uint8_t *rxbuf)
Sends a command then receives data over the WSPI bus.
- bool [wspiCommand](#) ([WSPIDriver](#) *wspip, const [wspi_command_t](#) *cmdp)
Sends a command without data phase.
- bool [wspiSend](#) ([WSPIDriver](#) *wspip, const [wspi_command_t](#) *cmdp, size_t n, const uint8_t *txbuf)
Sends a command with data over the WSPI bus.
- bool [wspiReceive](#) ([WSPIDriver](#) *wspip, const [wspi_command_t](#) *cmdp, size_t n, uint8_t *rxbuf)
Sends a command then receives data over the WSPI bus.
- void [wspiMapFlash](#) ([WSPIDriver](#) *wspip, const [wspi_command_t](#) *cmdp, uint8_t **addrp)
Maps in memory space a WSPI flash device.
- void [wspiUnmapFlash](#) ([WSPIDriver](#) *wspip)
Unmaps from memory space a WSPI flash device.
- void [wspiAcquireBus](#) ([WSPIDriver](#) *wspip)
Gains exclusive access to the WSPI bus.
- void [wspiReleaseBus](#) ([WSPIDriver](#) *wspip)
Releases exclusive access to the WSPI bus.

9.85.1 Detailed Description

WSPI Driver macros and structures.

9.86 hal_buffered_sio.c File Reference

Buffered SIO Driver code.

```
#include "hal.h"
#include "hal_buffered_sio.h"
```

Functions

- static void [__bsio_push_data](#) ([BufferedSIODriver](#) *bsiop)
- static void [__bsio_pop_data](#) ([BufferedSIODriver](#) *bsiop)
- static void [__bsio_default_cb](#) ([SIODriver](#) *siop)
- static void [__bsio_onotify](#) ([io_queue_t](#) *qp)
- static size_t [__bsio_write](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [__bsio_read](#) (void *ip, uint8_t *bp, size_t n)
- static [msg_t](#) [__bsio_put](#) (void *ip, uint8_t b)
- static [msg_t](#) [__bsio_get](#) (void *ip)
- static [msg_t](#) [__bsio_putt](#) (void *ip, uint8_t b, [sysinterval_t](#) timeout)
- static [msg_t](#) [__bsio_gett](#) (void *ip, [sysinterval_t](#) timeout)
- static size_t [__bsio_writet](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static size_t [__bsio_readt](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static [msg_t](#) [__bsio_ctl](#) (void *ip, unsigned int operation, void *arg)
- void [bsioObjectInit](#) ([BufferedSIODriver](#) *bsiop, [SIODriver](#) *siop, uint8_t *ib, size_t ibsize, uint8_t *ob, size_t obsize)
Initializes a generic serial driver object.
- [msg_t](#) [bsioStart](#) ([BufferedSIODriver](#) *bsiop, const [BufferedSIOConfig](#) *config)
Configures and starts the driver.
- void [bsioStop](#) ([BufferedSIODriver](#) *bsiop)
Stops the driver.

Variables

- static const struct [BufferedSIODriverVMT](#) vmt

9.86.1 Detailed Description

Buffered SIO Driver code.

9.87 hal_buffered_sio.h File Reference

Buffered SIO Driver macros and structures.

```
#include "hal.h"
```

Data Structures

- struct [BufferedSIODriverVMT](#)
BufferedSIODriver virtual methods table.
- struct [hal_buffered_sio_driver](#)
Buffered SIO driver class.

Macros

- #define [__buffered_sio_driver_methods](#) [__buffered_serial_methods](#)
BufferedSIODriver specific methods.
- #define [__buffered_sio_driver_data](#)
SerialDriver specific data.

Macro Functions

- #define [bsioPutl](#)(bsiop, b)
Direct write to a [BufferedSIODriver](#).
- #define [bsioPut](#)(bsiop, b)
Direct write to a [BufferedSIODriver](#).
- #define [bsioPutTimeout](#)(bsiop, b, t)
Direct write to a [BufferedSIODriver](#) with timeout specification.
- #define [bsioGetl](#)(bsiop)
Direct read from a [BufferedSIODriver](#).
- #define [bsioGet](#)(bsiop)
Direct read from a [BufferedSIODriver](#).
- #define [bsioGetTimeout](#)(bsiop, t)
Direct read from a [BufferedSIODriver](#) with timeout specification.
- #define [bsioWritel](#)(bsiop, b, n)
Direct blocking write to a [BufferedSIODriver](#).
- #define [bsioWrite](#)(bsiop, b, n)
Direct blocking write to a [BufferedSIODriver](#).
- #define [bsioWriteTimeout](#)(bsiop, b, n, t)
Direct blocking write to a [BufferedSIODriver](#) with timeout specification.
- #define [bsioAsynchronousWrite](#)(bsiop, b, n)
Direct non-blocking write to a [BufferedSIODriver](#).
- #define [bsioReadl](#)(bsiop, b, n)
Direct blocking read from a [BufferedSIODriver](#).
- #define [bsioRead](#)(bsiop, b, n)
Direct blocking read from a [BufferedSIODriver](#).
- #define [bsioReadTimeout](#)(bsiop, b, n, t)
Direct blocking read from a [BufferedSIODriver](#) with timeout specification.
- #define [bsioAsynchronousRead](#)(bsiop, b, n)
Direct non-blocking read from a [BufferedSIODriver](#).

Typedefs

- typedef struct [hal_buffered_siop_driver](#) [BufferedSIODriver](#)
Buffered SIO driver class.
- typedef [SIOConfig](#) [BufferedSIOConfig](#)
Type of a buffered SIO configuration.

Functions

- void [bsioObjectInit](#) ([BufferedSIODriver](#) *bsiop, [SIODriver](#) *siop, uint8_t *ib, size_t ibsize, uint8_t *ob, size_t obsize)
Initializes a generic serial driver object.
- [msg_t bsioStart](#) ([BufferedSIODriver](#) *bsiop, const [BufferedSIOConfig](#) *config)
Configures and starts the driver.
- void [bsioStop](#) ([BufferedSIODriver](#) *bsiop)
Stops the driver.

9.87.1 Detailed Description

Buffered SIO Driver macros and structures.

9.88 hal_mfs.c File Reference

Managed Flash Storage module code.

```
#include <string.h>
#include "hal.h"
#include "hal_mfs.h"
```

Macros

- #define [ALIGNED_REC_SIZE](#)(n)
Data record size aligned.
- #define [ALIGNED_DHDR_SIZE](#) [ALIGNED_REC_SIZE](#)(0)
Data record header size aligned.
- #define [ALIGNED_SIZEOF](#)(t)
Aligned size of a type.
- #define [PAIR](#)(a, b)
Combines two values (0..3) in one (0..15).
- #define [RET_ON_ERROR](#)(err)
Error check helper.

Functions

- uint16_t [crc16](#) (uint16_t crc, const uint8_t *data, size_t n)
- static void [mfs_state_reset](#) (MFSDriver *mfsp)
- static flash_offset_t [mfs_flash_get_bank_offset](#) (MFSDriver *mfsp, mfs_bank_t bank)
- static mfs_error_t [mfs_flash_read](#) (MFSDriver *mfsp, flash_offset_t offset, size_t n, uint8_t *rp)
Flash read.
- static mfs_error_t [mfs_flash_write](#) (MFSDriver *mfsp, flash_offset_t offset, size_t n, const uint8_t *wp)
Flash write.
- static mfs_error_t [mfs_flash_copy](#) (MFSDriver *mfsp, flash_offset_t doffset, flash_offset_t soffset, uint32_t n)
Flash copy.
- static mfs_error_t [mfs_bank_erase](#) (MFSDriver *mfsp, mfs_bank_t bank)
Erases and verifies all sectors belonging to a bank.
- static mfs_error_t [mfs_bank_verify_erase](#) (MFSDriver *mfsp, mfs_bank_t bank)
Erases and verifies all sectors belonging to a bank.
- static mfs_error_t [mfs_bank_write_header](#) (MFSDriver *mfsp, mfs_bank_t bank, uint32_t cnt)
Writes the validation header in a bank.
- static mfs_bank_state_t [mfs_bank_check_header](#) (MFSDriver *mfsp)
Checks integrity of the header in the shared buffer.
- static mfs_error_t [mfs_bank_scan_records](#) (MFSDriver *mfsp, mfs_bank_t bank, bool *wflagp)
Scans blocks searching for records.
- static mfs_error_t [mfs_bank_get_state](#) (MFSDriver *mfsp, mfs_bank_t bank, mfs_bank_state_t *statep, uint32_t *cntp)
Determines the state of a bank.
- static mfs_error_t [mfs_garbage_collect](#) (MFSDriver *mfsp)
Enforces a garbage collection.
- static mfs_error_t [mfs_try_mount](#) (MFSDriver *mfsp)
Performs a flash partition mount attempt.
- [mfs_error_t mfs_mount](#) (MFSDriver *mfsp)
Configures and activates a MFS driver.
- void [mfsObjectInit](#) (MFSDriver *mfsp, mfs_nocache_buffer_t *ncbuf)
Initializes an instance.
- [mfs_error_t mfsStart](#) (MFSDriver *mfsp, const MFSCfg *config)
Configures and activates a MFS driver.
- void [mfsStop](#) (MFSDriver *mfsp)
Deactivates a MFS driver.
- [mfs_error_t mfsErase](#) (MFSDriver *mfsp)
Destroys the state of the managed storage by erasing the flash.
- [mfs_error_t mfsReadRecord](#) (MFSDriver *mfsp, mfs_id_t id, size_t *np, uint8_t *buffer)
Retrieves and reads a data record.
- [mfs_error_t mfsWriteRecord](#) (MFSDriver *mfsp, mfs_id_t id, size_t n, const uint8_t *buffer)
Creates or updates a data record.
- [mfs_error_t mfsEraseRecord](#) (MFSDriver *mfsp, mfs_id_t id)
Erases a data record.
- [mfs_error_t mfsPerformGarbageCollection](#) (MFSDriver *mfsp)
Enforces a garbage collection operation.
- [mfs_error_t mfsStartTransaction](#) (MFSDriver *mfsp, size_t size)
Puts the driver in transaction mode.
- [mfs_error_t mfsCommitTransaction](#) (MFSDriver *mfsp)
A transaction is committed and finalized atomically.
- [mfs_error_t mfsRollbackTransaction](#) (MFSDriver *mfsp)
A transaction is rolled back atomically.

Variables

- static const uint16_t [crc16_table](#) [256]

9.88.1 Detailed Description

Managed Flash Storage module code.

This module manages a flash partition as a generic storage where arbitrary data records can be created, updated, deleted and retrieved.

A managed partition is composed of two banks of equal size, a bank is composed of one or more erasable sectors, a sector is divided in writable pages.

The module handles flash wear leveling and recovery of damaged banks (where possible) caused by power loss during operations. Both operations are transparent to the user.

9.89 hal_mfs.h File Reference

Managed Flash Storage module header.

```
#include "hal_flash.h"
```

Data Structures

- union [mfs_bank_header_t](#)
Type of a bank header.
- union [mfs_data_header_t](#)
Type of a data block header.
- struct [mfs_record_descriptor_t](#)
- struct [MFSConfig](#)
Type of a MFS configuration structure.
- struct [mfs_transaction_op_t](#)
Type of a buffered write/erase operation within a transaction.
- union [mfs_nocache_buffer](#)
Type of an non-cacheable MFS buffer.
- struct [MFSDriver](#)
Type of an MFS instance.

Macros

- #define `MFS_BANK_MAGIC_1` 0xEC705ADEU
- #define `MFS_BANK_MAGIC_2` 0xF0339CC5U
- #define `MFS_HEADER_MAGIC_1` 0x5FAE45F0U
- #define `MFS_HEADER_MAGIC_2` 0xF045AE5FU

Configuration options

- #define `MFS_CFG_MAX_RECORDS` 32
Maximum number of indexed records in the managed storage.
- #define `MFS_CFG_MAX_REPAIR_ATTEMPTS` 3
Maximum number of repair attempts on partition mount.
- #define `MFS_CFG_WRITE_VERIFY` TRUE
Verify written data.
- #define `MFS_CFG_STRONG_CHECKING` TRUE
Enables a stronger and slower check procedure on mount.
- #define `MFS_CFG_BUFFER_SIZE` 32
Size of the buffer used for data copying.
- #define `MFS_CFG_MEMORY_ALIGNMENT` 2
Enforced memory alignment.
- #define `MFS_CFG_TRANSACTION_MAX` 16
Maximum number of objects writable in a single transaction.

Error codes handling macros

- #define `MFS_IS_ERROR`(err)
- #define `MFS_IS_WARNING`(err)

Alignment macros

- #define `MFS_ALIGN_MASK` ((uint32_t)MFS_CFG_MEMORY_ALIGNMENT - 1U)
- #define `MFS_IS_ALIGNED`(v)
- #define `MFS_ALIGN_PREV`(v)
- #define `MFS_ALIGN_NEXT`(v)

Typedefs

- typedef uint32_t `mfs_id_t`
Type of a record identifier.
- typedef union `mfs_nocache_buffer` `mfs_nocache_buffer_t`
Type of an non-cacheable MFS buffer.

Enumerations

- enum `mfs_bank_t` { `MFS_BANK_0` = 0 , `MFS_BANK_1` = 1 }
Type of a flash bank.
- enum `mfs_state_t` {
`MFS_UNINIT` = 0 , `MFS_STOP` = 1 , `MFS_READY` = 2 , `MFS_TRANSACTION` = 3 ,
`MFS_ERROR` = 4 }
Type of driver state machine states.
- enum `mfs_error_t` {
`MFS_NO_ERROR` = 0 , `MFS_WARN_REPAIR` = 1 , `MFS_WARN_GC` = 2 , `MFS_ERR_INV_STATE` = -1 ,
`MFS_ERR_INV_SIZE` = -2 , `MFS_ERR_NOT_FOUND` = -3 , `MFS_ERR_OUT_OF_MEM` = -4 ,
`MFS_ERR_TRANSACTION_NUM` = -5 ,
`MFS_ERR_TRANSACTION_SIZE` = -6 , `MFS_ERR_NOT_ERASED` = -7 , `MFS_ERR_FLASH_FAILURE` =
-8 , `MFS_ERR_INTERNAL` = -9 }
Type of an MFS error code.
- enum `mfs_bank_state_t` { `MFS_BANK_ERASED` = 0 , `MFS_BANK_OK` = 1 , `MFS_BANK_GARBAGE` = 2 }
Type of a bank state assessment.

Functions

- void [mfsObjectInit](#) (MFSDriver *mfsp, [mfs_nocache_buffer_t](#) *ncbuf)
Initializes an instance.
- [mfs_error_t](#) [mfsStart](#) (MFSDriver *mfsp, const [MFSCConfig](#) *config)
Configures and activates a MFS driver.
- void [mfsStop](#) (MFSDriver *mfsp)
Deactivates a MFS driver.
- [mfs_error_t](#) [mfsErase](#) (MFSDriver *mfsp)
Destroys the state of the managed storage by erasing the flash.
- [mfs_error_t](#) [mfsReadRecord](#) (MFSDriver *mfsp, [mfs_id_t](#) id, size_t *np, uint8_t *buffer)
Retrieves and reads a data record.
- [mfs_error_t](#) [mfsWriteRecord](#) (MFSDriver *mfsp, [mfs_id_t](#) id, size_t n, const uint8_t *buffer)
Creates or updates a data record.
- [mfs_error_t](#) [mfsEraseRecord](#) (MFSDriver *mfsp, [mfs_id_t](#) id)
Erases a data record.
- [mfs_error_t](#) [mfsPerformGarbageCollection](#) (MFSDriver *mfsp)
Enforces a garbage collection operation.

9.89.1 Detailed Description

Managed Flash Storage module header.

9.90 hal_serial_nor.c File Reference

Serial NOR serial flash driver code.

```
#include "hal.h"
#include "hal_serial_nor.h"
```

Functions

- static const [flash_descriptor_t](#) * [snor_get_descriptor](#) (void *instance)
Returns a pointer to the device descriptor.
- static [flash_error_t](#) [snor_read](#) (void *instance, [flash_offset_t](#) offset, size_t n, uint8_t *rp)
- static [flash_error_t](#) [snor_program](#) (void *instance, [flash_offset_t](#) offset, size_t n, const uint8_t *pp)
- static [flash_error_t](#) [snor_start_erase_all](#) (void *instance)
- static [flash_error_t](#) [snor_start_erase_sector](#) (void *instance, [flash_sector_t](#) sector)
- static [flash_error_t](#) [snor_verify_erase](#) (void *instance, [flash_sector_t](#) sector)
- static [flash_error_t](#) [snor_query_erase](#) (void *instance, uint32_t *msec)
- static [flash_error_t](#) [snor_acquire_exclusive](#) (void *instance)
- static [flash_error_t](#) [snor_release_exclusive](#) (void *instance)
- static [flash_error_t](#) [snor_read_sfdp](#) (void *instance, [flash_offset_t](#) offset, size_t n, uint8_t *rp)
- void [snor_spi_cmd_addr](#) (BUSDriver *busp, uint32_t cmd, [flash_offset_t](#) offset)
- void [bus_acquire](#) (BUSDriver *busp, const [BUSConfig](#) *config)
Bus acquisition and lock.
- void [bus_release](#) (BUSDriver *busp)
Bus release.

- void `bus_stop` (`BUSDriver` *busp)
Stops the underlying bus driver.
- void `bus_cmd` (`BUSDriver` *busp, uint32_t cmd)
Sends a naked command.
- void `bus_cmd_send` (`BUSDriver` *busp, uint32_t cmd, size_t n, const uint8_t *p)
Sends a command followed by a data transmit phase.
- void `bus_cmd_receive` (`BUSDriver` *busp, uint32_t cmd, size_t n, uint8_t *p)
Sends a command followed by a data receive phase.
- void `bus_cmd_addr` (`BUSDriver` *busp, uint32_t cmd, flash_offset_t offset)
Sends a command followed by a flash address.
- void `bus_cmd_addr_send` (`BUSDriver` *busp, uint32_t cmd, flash_offset_t offset, size_t n, const uint8_t *p)
Sends a command followed by a flash address and a data transmit phase.
- void `bus_cmd_addr_receive` (`BUSDriver` *busp, uint32_t cmd, flash_offset_t offset, size_t n, uint8_t *p)
Sends a command followed by a flash address and a data receive phase.
- void `bus_cmd_dummy_receive` (`BUSDriver` *busp, uint32_t cmd, uint32_t dummy, size_t n, uint8_t *p)
Sends a command followed by dummy cycles and a data receive phase.
- void `bus_cmd_addr_dummy_receive` (`BUSDriver` *busp, uint32_t cmd, flash_offset_t offset, uint32_t dummy, size_t n, uint8_t *p)
Sends a command followed by a flash address, dummy cycles and a data receive phase.
- void `snorObjectInit` (`SNORDriver` *devp, snor_nocache_buffer_t *nocache)
Initializes an instance.
- void `snorStart` (`SNORDriver` *devp, const `SNORConfig` *config)
Configures and activates SNOR driver.
- void `snorStop` (`SNORDriver` *devp)
Deactivates the SNOR driver.
- void `snorMemoryMap` (`SNORDriver` *devp, uint8_t **addrp)
Enters the memory Mapping mode.
- void `snorMemoryUnmap` (`SNORDriver` *devp)
Leaves the memory Mapping mode.

Variables

- static const struct `SNORDriverVMT` `snor_vmt`
Virtual methods table.

9.90.1 Detailed Description

Serial NOR serial flash driver code.

9.91 hal_serial_nor.h File Reference

Serial NOR driver header.

```
#include "hal_flash.h"
#include "hal_flash_device.h"
```

Data Structures

- struct [SNORConfig](#)
Type of a SNOR configuration structure.
- struct [SNORDriverVMT](#)
SNOR virtual methods table.
- struct [snor_nocache_buffer](#)
- struct [SNORDriver](#)
Type of SNOR flash class.

Macros

- #define [SNOR_BUFFER_SIZE](#) 32
Size of the buffer for internal operations.
- #define [BUSConfig](#) [SPICongig](#)
- #define [BUSDriver](#) [SPIDriver](#)
- #define [_snor_flash_methods_alone](#)
SNORDriver specific methods.
- #define [_snor_flash_methods](#)
SNORDriver specific methods with inherited ones.
- #define [bus_acquire](#)(busp, config)
- #define [bus_release](#)(busp)

Bus interface modes.

- #define [SNOR_BUS_DRIVER_SPI](#) 0U
- #define [SNOR_BUS_DRIVER_WSPI](#) 1U

Configuration options

- #define [SNOR_BUS_DRIVER](#) [SNOR_BUS_DRIVER_WSPI](#)
Physical transport interface.
- #define [SNOR_SHARED_BUS](#) TRUE
Shared bus switch.
- #define [SNOR_USE_MUTUAL_EXCLUSION](#) TRUE
Exclusive access control.
- #define [SNOR_SPI_4BYTES_ADDRESS](#) FALSE
SPI 4-bytes address switch.

Typedefs

- typedef struct [snor_nocache_buffer](#) [snor_nocache_buffer_t](#)

Functions

- void [bus_acquire](#) ([BUSDriver](#) *busp, const [BUSConfig](#) *config)
Bus acquisition and lock.
- void [bus_release](#) ([BUSDriver](#) *busp)
Bus release.
- void [bus_cmd](#) ([BUSDriver](#) *busp, uint32_t cmd)
Sends a naked command.
- void [bus_cmd_send](#) ([BUSDriver](#) *busp, uint32_t cmd, size_t n, const uint8_t *p)
Sends a command followed by a data transmit phase.
- void [bus_cmd_receive](#) ([BUSDriver](#) *busp, uint32_t cmd, size_t n, uint8_t *p)
Sends a command followed by a data receive phase.
- void [bus_cmd_addr](#) ([BUSDriver](#) *busp, uint32_t cmd, [flash_offset_t](#) offset)
Sends a command followed by a flash address.
- void [bus_cmd_addr_send](#) ([BUSDriver](#) *busp, uint32_t cmd, [flash_offset_t](#) offset, size_t n, const uint8_t *p)
Sends a command followed by a flash address and a data transmit phase.
- void [bus_cmd_addr_receive](#) ([BUSDriver](#) *busp, uint32_t cmd, [flash_offset_t](#) offset, size_t n, uint8_t *p)
Sends a command followed by a flash address and a data receive phase.
- void [bus_cmd_dummy_receive](#) ([BUSDriver](#) *busp, uint32_t cmd, uint32_t dummy, size_t n, uint8_t *p)
Sends a command followed by dummy cycles and a data receive phase.
- void [bus_cmd_addr_dummy_receive](#) ([BUSDriver](#) *busp, uint32_t cmd, [flash_offset_t](#) offset, uint32_t dummy, size_t n, uint8_t *p)
Sends a command followed by a flash address, dummy cycles and a data receive phase.
- void [snorObjectInit](#) ([SNORDriver](#) *devp, [snor_nocache_buffer_t](#) *nocache)
Initializes an instance.
- void [snorStart](#) ([SNORDriver](#) *devp, const [SNORConfig](#) *config)
Configures and activates SNOR driver.
- void [snorStop](#) ([SNORDriver](#) *devp)
Deactivates the SNOR driver.
- void [snorMemoryMap](#) ([SNORDriver](#) *devp, uint8_t **addrp)
Enters the memory Mapping mode.
- void [snorMemoryUnmap](#) ([SNORDriver](#) *devp)
Leaves the memory Mapping mode.

9.91.1 Detailed Description

Serial NOR driver header.

9.92 hal_xsnor_macronix_mx25.c File Reference

Generated SNOR Macronix MX25 source.

```
#include "hal.h"
#include "hal_xsnor_macronix_mx25.h"
#include "hal_xsnor_macronix_mx25_impl.inc"
```

9.92.1 Detailed Description

Generated SNOR Macronix MX25 source.

Note

This is a generated file, do not edit directly.

9.93 hal_xsnor_macronix_mx25.h File Reference

Generated SNOR Macronix MX25 header.

```
#include "oop_base_object.h"
#include "hal_xsnor_base.h"
```

Data Structures

- struct [hal_xsnor_macronix_mx25_vmt](#)
Class [hal_xsnor_macronix_mx25_c](#) virtual methods table.
- struct [hal_xsnor_macronix_mx25](#)
Structure representing a SNOR Macronix MX25 driver class.

Macros

Device options

- #define [MX25_OPT_DUMMY_CYCLES_MASK](#) (15U << 0)
Mask of the dummy cycles field.
- #define [MX25_OPT_DUMMY_CYCLES](#)(n)
Number of dummy cycles.
- #define [MX25_OPT_NO_WIDTH_SWITCH](#) (1U << 4)
Switch bus width on initialization.
- #define [MX25_OPT_USE_SUBSECTORS](#) (1U << 5)
Use 4kB sub-sectors rather than 64kB sectors.
- #define [MX25_OPT_NICE_WAITING](#) (1U << 6)
Delays insertion.
- #define [MX25_OPT_USE_DTR](#) (1U << 7)
Enables DTR in 8 lines mode.

Typedefs

Class @p hal_xsnor_macronix_mx25_c structures

- typedef struct [hal_xsnor_macronix_mx25](#) [hal_xsnor_macronix_mx25_c](#)
Type of a SNOR Macronix MX25 driver class.

Functions

Methods implementations of `hal_xsnor_macronix_mx25_c`

- `void * __mx25_objinit_impl (void *ip, const void *vmt)`
Implementation of object creation.
- `void __mx25_dispose_impl (void *ip)`
Implementation of object finalization.
- `flash_error_t __mx25_init_impl (void *ip)`
Override of method `xsnor_device_init()`.
- `flash_error_t __mx25_read_impl (void *ip, flash_offset_t offset, size_t n, uint8_t *rp)`
Override of method `xsnor_device_read()`.
- `flash_error_t __mx25_program_impl (void *ip, flash_offset_t offset, size_t n, const uint8_t *pp)`
Override of method `xsnor_device_program()`.
- `flash_error_t __mx25_start_erase_all_impl (void *ip)`
Override of method `xsnor_device_start_erase_all()`.
- `flash_error_t __mx25_start_erase_sector_impl (void *ip, flash_sector_t sector)`
Override of method `xsnor_device_start_erase_sector()`.
- `flash_error_t __mx25_query_erase_impl (void *ip, unsigned *msec)`
Override of method `xsnor_device_query_erase()`.
- `flash_error_t __mx25_verify_erase_impl (void *ip, flash_sector_t sector)`
Override of method `xsnor_device_verify_erase()`.
- `flash_error_t __mx25_mmap_on_impl (void *ip, uint8_t **addrp)`
Override of method `xsnor_device_mmap_on()`.
- `void __mx25_mmap_off_impl (void *ip)`
Override of method `xsnor_device_mmap_off()`.

Default constructor of `hal_xsnor_macronix_mx25_c`

- `static CC_FORCE_INLINE hal_xsnor_macronix_mx25_c * mx25ObjectInit (hal_xsnor_macronix_mx25_c *self)`
Default initialization function of `hal_xsnor_macronix_mx25_c`.

9.93.1 Detailed Description

Generated SNOR Macronix MX25 header.

Note

This is a generated file, do not edit directly.

9.94 `hal_xsnor_macronix_mx25_impl.inc` File Reference

Template of SNOR Macronix MX25 source.

```
#include <string.h>
```

Macros

- #define `PAGE_SIZE` 256U
- #define `PAGE_MASK` (`PAGE_SIZE` - 1U)

Command codes, SPI mode

- #define `CMD_SPI_READ3B` 0x03U
- #define `CMD_SPI_FAST_READ3B` 0x0BU
- #define `CMD_SPI_PP3B` 0x02U
- #define `CMD_SPI_SE3B` 0x20U
- #define `CMD_SPI_BE3B` 0xD8U
- #define `CMD_SPI_CE` 0xC7U
- #define `CMD_SPI_READ4B` 0x13U
- #define `CMD_SPI_FAST_READ4B` 0x0CU
- #define `CMD_SPI_PP4B` 0x12U
- #define `CMD_SPI_SE4B` 0x21U
- #define `CMD_SPI_BE4B` 0xDCU
- #define `CMD_SPI_WREN` 0x06U
- #define `CMD_SPI_WRDI` 0x04U
- #define `CMD_SPI_PE_SUSPEND` 0xB0U
- #define `CMD_SPI_PE_RESUME` 0x30U
- #define `CMD_SPI_DP` 0xB9U
- #define `CMD_SPI_SBL` 0xC0U
- #define `CMD_SPI_ENSO` 0xB1U
- #define `CMD_SPI_EXSO` 0xC1U
- #define `CMD_SPI_NOP` 0x00U
- #define `CMD_SPI_RSTEN` 0x66U
- #define `CMD_SPI_RST` 0x99U
- #define `CMD_SPI_RDID` 0x9FU
- #define `CMD_SPI_RDSFDP` 0x5AU
- #define `CMD_SPI_RDSR` 0x05U
- #define `CMD_SPI_RDCCR` 0x15U
- #define `CMD_SPI_WRSR` 0x01U
- #define `CMD_SPI_RDCCR2` 0x71U
- #define `CMD_SPI_WRCR2` 0x72U
- #define `CMD_SPI_RDFBR` 0x16U
- #define `CMD_SPI_WRFBR` 0x17U
- #define `CMD_SPI_ESFBR` 0x18U
- #define `CMD_SPI_RDSCUR` 0x2BU
- #define `CMD_SPI_WRSCUR` 0x2FU
- #define `CMD_SPI_WRLR` 0x2CU
- #define `CMD_SPI_RDLR` 0x2DU
- #define `CMD_SPI_WRSPB` 0xE3U
- #define `CMD_SPI_ESSPB` 0xE4U
- #define `CMD_SPI_RDSPB` 0xE2U
- #define `CMD_SPI_WRDPB` 0xE1U
- #define `CMD_SPI_RDDPB` 0xE0U
- #define `CMD_SPI_WPSEL` 0x68U
- #define `CMD_SPI_GBLK` 0x7EU
- #define `CMD_SPI_GBULK` 0x98U
- #define `CMD_SPI_RDPASS` 0x27U
- #define `CMD_SPI_WRPASS` 0x28U
- #define `CMD_SPI_PASSULK` 0x29U

Command codes, OPI mode

- #define `CMD_OPI_8READ` 0xEC13U
- #define `CMD_OPI_8DTRD` 0xEE11U
- #define `CMD_OPI_RDID` 0x9F60U
- #define `CMD_OPI_RDSFDP` 0x5AA5U
- #define `CMD_OPI_PP` 0x12EDU

- #define `CMD_OPI_SE` 0x21DEU
- #define `CMD_OPI_BE` 0xDC23U
- #define `CMD_OPI_CE` 0xC738U
- #define `CMD_OPI_WREN` 0x06F9U
- #define `CMD_OPI_WRDI` 0x04FBU
- #define `CMD_OPI_PE_SUSPEND` 0xB04FU
- #define `CMD_OPI_PE_RESUME` 0x30CFU
- #define `CMD_OPI_DP` 0xB946U
- #define `CMD_OPI_SBL` 0xC03FU
- #define `CMD_OPI_ENSO` 0xB14EU
- #define `CMD_OPI_EXSO` 0xC13EU
- #define `CMD_OPI_NOP` 0x00FFU
- #define `CMD_OPI_RSTEN` 0x6699U
- #define `CMD_OPI_RST` 0x9966U
- #define `CMD_OPI_RDSR` 0x05FAU
- #define `CMD_OPI_RDCR` 0x15EAU
- #define `CMD_OPI_WRSR` 0x01FEU
- #define `CMD_OPI_WRCR` 0x01FEU
- #define `CMD_OPI_RDCR2` 0x718EU
- #define `CMD_OPI_WRCR2` 0x728DU
- #define `CMD_OPI_RDFBR` 0x16E9U
- #define `CMD_OPI_WRFBR` 0x17E8U
- #define `CMD_OPI_ESFBR` 0x18E7U
- #define `CMD_OPI_RDSCUR` 0x2BD4U
- #define `CMD_OPI_WRSCUR` 0x2FD0U
- #define `CMD_OPI_WRLR` 0x2CD3U
- #define `CMD_OPI_RDLR` 0x2DD2U
- #define `CMD_OPI_WRSPB` 0xE31CU
- #define `CMD_OPI_ESSPB` 0xE41BU
- #define `CMD_OPI_RDSPB` 0xE21DU
- #define `CMD_OPI_WRDPB` 0xE11EU
- #define `CMD_OPI_RDDPB` 0xE01FU
- #define `CMD_OPI_WPSEL` 0x6897U
- #define `CMD_OPI_GBLK` 0x7E81U
- #define `CMD_OPI_GBULK` 0x9867U
- #define `CMD_OPI_RDPASS` 0x27D8U
- #define `CMD_OPI_WRPASS` 0x28D7U
- #define `CMD_OPI_PASSULK` 0x29D6U

Flags status register bits

- #define `FLAGS_WPSEL` 0x80U
- #define `FLAGS_E_FAIL` 0x40U
- #define `FLAGS_P_FAIL` 0x20U
- #define `FLAGS_ESB` 0x08U
- #define `FLAGS_PSB` 0x04U
- #define `FLAGS_LDSO` 0x02U
- #define `FLAGS_SECURED_OTP` 0x01U
- #define `FLAGS_ALL_ERRORS` (`FLAGS_E_FAIL` | `FLAGS_P_FAIL`)

MX25-required transfer modes

- #define `CFG_C8_SPI`
- #define `CFG_C16_8STR`
- #define `CFG_C16_8DTR`
- #define `CFG_C8_A32_SPI`
- #define `CFG_C16_A32_8STR`
- #define `CFG_C16_A32_8DTR`
- #define `CFG_C8_DATA_SPI`
- #define `CFG_C16_DATA_8STR`
- #define `CFG_C16_DATA_8DTR`
- #define `CFG_C8_A32_DATA_SPI`
- #define `CFG_C16_A32_DATA_8STR`
- #define `CFG_C16_A32_DATA_8DTR`

Functions

- static bool [mx25_find_id](#) (const uint8_t *set, size_t size, uint8_t element)
- static flash_error_t [mx25_poll_status](#) (hal_xsnor_macronix_mx25_c *self)
- static void [mx25_reset_memory](#) (hal_xsnor_macronix_mx25_c *self)
- static void [mx25_read_id](#) (hal_xsnor_macronix_mx25_c *self, uint8_t *buf)
- static void [mx25_write_cr2](#) (hal_xsnor_macronix_mx25_c *self, uint32_t addr, const uint8_t *value)

Methods implementations of hal_xsnor_macronix_mx25_c

- void * [__mx25_objinit_impl](#) (void *ip, const void *vmt)
Implementation of object creation.
- void [__mx25_dispose_impl](#) (void *ip)
Implementation of object finalization.
- flash_error_t [__mx25_init_impl](#) (void *ip)
Override of method [xsnor_device_init\(\)](#).
- flash_error_t [__mx25_read_impl](#) (void *ip, flash_offset_t offset, size_t n, uint8_t *rp)
Override of method [xsnor_device_read\(\)](#).
- flash_error_t [__mx25_program_impl](#) (void *ip, flash_offset_t offset, size_t n, const uint8_t *pp)
Override of method [xsnor_device_program\(\)](#).
- flash_error_t [__mx25_start_erase_all_impl](#) (void *ip)
Override of method [xsnor_device_start_erase_all\(\)](#).
- flash_error_t [__mx25_start_erase_sector_impl](#) (void *ip, flash_sector_t sector)
Override of method [xsnor_device_start_erase_sector\(\)](#).
- flash_error_t [__mx25_query_erase_impl](#) (void *ip, unsigned *msec)
Override of method [xsnor_device_query_erase\(\)](#).
- flash_error_t [__mx25_verify_erase_impl](#) (void *ip, flash_sector_t sector)
Override of method [xsnor_device_verify_erase\(\)](#).
- flash_error_t [__mx25_mmap_on_impl](#) (void *ip, uint8_t **addrp)
Override of method [xsnor_device_mmap_on\(\)](#).
- void [__mx25_mmap_off_impl](#) (void *ip)
Override of method [xsnor_device_mmap_off\(\)](#).

Variables

- static const [xsnor_commands_t](#) cmd_spi
- static const [xsnor_commands_t](#) cmd_8str
- static const [xsnor_commands_t](#) cmd_8dtr
- static const [wspi_command_t](#) cmd_reset_enable_spi
- static const [wspi_command_t](#) cmd_reset_memory_spi
- static const [wspi_command_t](#) cmd_reset_enable_8str
- static const [wspi_command_t](#) cmd_reset_memory_8str
- static const [wspi_command_t](#) cmd_reset_enable_8dtr
- static const [wspi_command_t](#) cmd_reset_memory_8dtr
- static const uint8_t [mx25_manufacturer_ids](#) [] = {0xC2}
- static const uint8_t [mx25_memory_type_ids](#) [] = {0x85}
- const struct [hal_xsnor_macronix_mx25_vmt](#) [__hal_xsnor_macronix_mx25_vmt](#)
VMT structure of SNOR Macronix MX25 driver class.

9.94.1 Detailed Description

Template of SNOR Macronix MX25 source.

Note

This is a template file, can be edited directly.

9.95 hal_xsnor_micron_n25q.c File Reference

Generated SNOR Micron N25Q source.

```
#include "hal.h"
#include "hal_xsnor_micron_n25q.h"
#include "hal_xsnor_micron_n25q_impl.inc"
```

9.95.1 Detailed Description

Generated SNOR Micron N25Q source.

Note

This is a generated file, do not edit directly.

9.96 hal_xsnor_micron_n25q.h File Reference

Generated SNOR Micron N25Q header.

```
#include "oop_base_object.h"
#include "hal_xsnor_base.h"
```

Data Structures

- struct [hal_xsnor_micron_n25q_vmt](#)
Class [hal_xsnor_micron_n25q_c](#) virtual methods table.
- struct [hal_xsnor_micron_n25q](#)
Structure representing a SNOR Micron N25Q driver class.

Macros

Device options

- #define [N25Q_OPT_DUMMY_CYCLES_MASK](#) (15U << 0)
Mask of the dummy cycles field.
- #define [N25Q_OPT_DUMMY_CYCLES](#)(n)
Number of dummy cycles.
- #define [N25Q_OPT_NO_WIDTH_SWITCH](#) (1U << 4)
Switch bus width on initialization.
- #define [N25Q_OPT_USE_SUBSECTORS](#) (1U << 5)
Use 4kB sub-sectors rather than 64kB sectors.
- #define [N25Q_OPT_NICE_WAITING](#) (1U << 6)
Delays insertion.

Typedefs

Class @p hal_xsnor_micron_n25q_c structures

- typedef struct [hal_xsnor_micron_n25q](#) [hal_xsnor_micron_n25q_c](#)
Type of a SNOR Micron N25Q driver class.

Functions

Methods implementations of hal_xsnor_micron_n25q_c

- void * [__n25q_objinit_impl](#) (void *ip, const void *vmt)
Implementation of object creation.
- void [__n25q_dispose_impl](#) (void *ip)
Implementation of object finalization.
- [flash_error_t __n25q_init_impl](#) (void *ip)
Override of method [xsnor_device_init\(\)](#).
- [flash_error_t __n25q_read_impl](#) (void *ip, [flash_offset_t](#) offset, [size_t](#) n, [uint8_t](#) *rp)
Override of method [xsnor_device_read\(\)](#).
- [flash_error_t __n25q_program_impl](#) (void *ip, [flash_offset_t](#) offset, [size_t](#) n, const [uint8_t](#) *pp)
Override of method [xsnor_device_program\(\)](#).
- [flash_error_t __n25q_start_erase_all_impl](#) (void *ip)
Override of method [xsnor_device_start_erase_all\(\)](#).
- [flash_error_t __n25q_start_erase_sector_impl](#) (void *ip, [flash_sector_t](#) sector)
Override of method [xsnor_device_start_erase_sector\(\)](#).
- [flash_error_t __n25q_query_erase_impl](#) (void *ip, unsigned *msec)
Override of method [xsnor_device_query_erase\(\)](#).
- [flash_error_t __n25q_verify_erase_impl](#) (void *ip, [flash_sector_t](#) sector)
Override of method [xsnor_device_verify_erase\(\)](#).
- [flash_error_t __n25q_mmap_on_impl](#) (void *ip, [uint8_t](#) **addrp)
Override of method [xsnor_device_mmap_on\(\)](#).
- void [__n25q_mmap_off_impl](#) (void *ip)
Override of method [xsnor_device_mmap_off\(\)](#).

Default constructor of hal_xsnor_micron_n25q_c

- static [CC_FORCE_INLINE](#) [hal_xsnor_micron_n25q_c](#) * [n25qObjectInit](#) ([hal_xsnor_micron_n25q_c](#) *self)
Default initialization function of [hal_xsnor_micron_n25q_c](#).

9.96.1 Detailed Description

Generated SNOR Micron N25Q header.

Note

This is a generated file, do not edit directly.

9.97 hal_xsnor_micron_n25q_impl.inc File Reference

Template of SNOR Micron N25Q source.

Macros

- #define `PAGE_SIZE` 256U
- #define `PAGE_MASK` (`PAGE_SIZE` - 1U)

Command codes

- #define `CMD_RESET_ENABLE` 0x66
- #define `CMD_RESET_MEMORY` 0x99
- #define `CMD_READ_ID` 0x9F
- #define `CMD_MULTIPLE_IO_READ_ID` 0xAF
- #define `CMD_READ_DISCOVERY_PARAMETER` 0x5A
- #define `CMD_READ` 0x03
- #define `CMD_FAST_READ` 0x0B
- #define `CMD_WRITE_ENABLE` 0x06
- #define `CMD_WRITE_DISABLE` 0x04
- #define `CMD_READ_STATUS_REGISTER` 0x05
- #define `CMD_WRITE_STATUS_REGISTER` 0x01
- #define `CMD_READ_LOCK_REGISTER` 0xE8
- #define `CMD_WRITE_LOCK_REGISTER` 0xE5
- #define `CMD_READ_FLAG_STATUS_REGISTER` 0x70
- #define `CMD_CLEAR_FLAG_STATUS_REGISTER` 0x50
- #define `CMD_READ_NV_CONFIGURATION_REGISTER` 0xB5
- #define `CMD_WRITE_NV_CONFIGURATION_REGISTER` 0xB1
- #define `CMD_READ_V_CONF_REGISTER` 0x85
- #define `CMD_WRITE_V_CONF_REGISTER` 0x81
- #define `CMD_READ_ENHANCED_V_CONF_REGISTER` 0x65
- #define `CMD_WRITE_ENHANCED_V_CONF_REGISTER` 0x61
- #define `CMD_PAGE_PROGRAM` 0x02
- #define `CMD_SUBSECTOR_ERASE` 0x20
- #define `CMD_SECTOR_ERASE` 0xD8
- #define `CMD_BULK_ERASE` 0xC7
- #define `CMD_PROGRAM_ERASE_RESUME` 0x7A
- #define `CMD_PROGRAM_ERASE_SUSPEND` 0x75
- #define `CMD_READ_OTP_ARRAY` 0x4B
- #define `CMD_PROGRAM_OTP_ARRAY` 0x42

Flags status register bits

- #define `FLAGS_PROGRAM_ERASE` 0x80U
- #define `FLAGS_ERASE_SUSPEND` 0x40U
- #define `FLAGS_ERASE_ERROR` 0x20U
- #define `FLAGS_PROGRAM_ERROR` 0x10U
- #define `FLAGS_VPP_ERROR` 0x08U
- #define `FLAGS_PROGRAM_SUSPEND` 0x04U
- #define `FLAGS_PROTECTION_ERROR` 0x02U
- #define `FLAGS_RESERVED` 0x01U
- #define `FLAGS_ALL_ERRORS`

Functions

- static bool `n25q_find_id` (const uint8_t *set, size_t size, uint8_t element)
- static flash_error_t `n25q_poll_status` (hal_xsnor_micron_n25q_c *self)
- static void `n25q_activate_xip` (hal_xsnor_micron_n25q_c *self)
- static void `n25q_reset_xip` (hal_xsnor_micron_n25q_c *self)
- static void `n25q_reset_memory` (hal_xsnor_micron_n25q_c *self)
- static void `n25q_read_id` (hal_xsnor_micron_n25q_c *self, uint8_t *buf)

Methods implementations of hal_xsnor_micron_n25q_c

- void * [__n25q_objinit_impl](#) (void *ip, const void *vmt)
Implementation of object creation.
- void [__n25q_dispose_impl](#) (void *ip)
Implementation of object finalization.
- [flash_error_t __n25q_init_impl](#) (void *ip)
Override of method [xsnor_device_init\(\)](#).
- [flash_error_t __n25q_read_impl](#) (void *ip, [flash_offset_t](#) offset, [size_t](#) n, [uint8_t](#) *rp)
Override of method [xsnor_device_read\(\)](#).
- [flash_error_t __n25q_program_impl](#) (void *ip, [flash_offset_t](#) offset, [size_t](#) n, const [uint8_t](#) *pp)
Override of method [xsnor_device_program\(\)](#).
- [flash_error_t __n25q_start_erase_all_impl](#) (void *ip)
Override of method [xsnor_device_start_erase_all\(\)](#).
- [flash_error_t __n25q_start_erase_sector_impl](#) (void *ip, [flash_sector_t](#) sector)
Override of method [xsnor_device_start_erase_sector\(\)](#).
- [flash_error_t __n25q_query_erase_impl](#) (void *ip, unsigned *msec)
Override of method [xsnor_device_query_erase\(\)](#).
- [flash_error_t __n25q_verify_erase_impl](#) (void *ip, [flash_sector_t](#) sector)
Override of method [xsnor_device_verify_erase\(\)](#).
- [flash_error_t __n25q_mmap_on_impl](#) (void *ip, [uint8_t](#) **addrp)
Override of method [xsnor_device_mmap_on\(\)](#).
- void [__n25q_mmap_off_impl](#) (void *ip)
Override of method [xsnor_device_mmap_off\(\)](#).

Variables

- static const [xsnor_commands_t](#) cmd1l
- static const [xsnor_commands_t](#) cmd2l
- static const [xsnor_commands_t](#) cmd4l
- static const [wspi_command_t](#) cmd_reset_enable_1
- static const [wspi_command_t](#) cmd_reset_memory_1
- static const [wspi_command_t](#) cmd_reset_enable_2
- static const [wspi_command_t](#) cmd_reset_memory_2
- static const [wspi_command_t](#) cmd_reset_enable_4
- static const [wspi_command_t](#) cmd_reset_memory_4
- static const [uint8_t](#) [n25q_manufacturer_ids](#) [] = {0x20}
- static const [uint8_t](#) [n25q_memory_type_ids](#) [] = {0xBA, 0xBB}
- const struct [hal_xsnor_micron_n25q_vmt](#) [__hal_xsnor_micron_n25q_vmt](#)
VMT structure of SNOR Micron N25Q driver class.

9.97.1 Detailed Description

Template of SNOR Micron N25Q source.

Note

This is a template file, can be edited directly.

9.98 xsnor.dox File Reference

9.99 xsnor_arch.dox File Reference

9.100 xsnor_macronix_mx25.dox File Reference

9.101 xsnor_micron_n25q.dox File Reference

9.102 hal_flash_interface.h File Reference

Generated Flash Interface header.

Data Structures

- struct [flash_interface_vmt](#)
Interface [flash_interface_i](#) virtual methods table.
- struct [flash_interface](#)
Structure representing a flash interface interface.

Typedefs

Interface @p flash_interface_i structures

- typedef struct [flash_interface](#) [flash_interface_i](#)
Type of a flash interface interface.

Functions

Virtual methods of flash_interface_i

- static [CC_FORCE_INLINE](#) const [flash_descriptor_t](#) * [flsGetDescriptor](#) (void *ip)
Gets the flash descriptor structure.
- static [CC_FORCE_INLINE](#) [flash_error_t](#) [flsRead](#) (void *ip, [flash_offset_t](#) offset, [size_t](#) n, [uint8_t](#) *rp)
Read operation.
- static [CC_FORCE_INLINE](#) [flash_error_t](#) [flsProgram](#) (void *ip, [flash_offset_t](#) offset, [size_t](#) n, const [uint8_t](#) *pp)
Program operation.
- static [CC_FORCE_INLINE](#) [flash_error_t](#) [flsStartEraseAll](#) (void *ip)
Starts a whole-device erase operation.
- static [CC_FORCE_INLINE](#) [flash_error_t](#) [flsStartEraseSector](#) (void *ip, [flash_sector_t](#) sector)
Starts an sector erase operation.
- static [CC_FORCE_INLINE](#) [flash_error_t](#) [flsQueryErase](#) (void *ip, unsigned *msec)
Queries the driver for erase operation progress.
- static [CC_FORCE_INLINE](#) [flash_error_t](#) [flsVerifyErase](#) (void *ip, [flash_sector_t](#) sector)
Returns the erase state of a sector.
- static [CC_FORCE_INLINE](#) [flash_error_t](#) [flsAcquireExclusive](#) (void *ip)
Acquires exclusive access to flash.
- static [CC_FORCE_INLINE](#) [flash_error_t](#) [flsReleaseExclusive](#) (void *ip)
Releases exclusive access to flash.

9.102.1 Detailed Description

Generated Flash Interface header.

Note

This is a generated file, do not edit directly.

9.103 hal_xsnor_base.h File Reference

Generated SNOR Base Driver header.

```
#include "oop_base_object.h"
#include "oop_base_interface.h"
#include "hal_flash_interface.h"
```

Data Structures

- struct [xsnor_buffers](#)
SNOR driver configuration.
- struct [xsnor_commands](#)
SNOR command configuration structure.
- struct [xsnor_bus_wspi](#)
WSPI-specific configuration fields.
- struct [xsnor_bus_spi](#)
SPI-specific configuration fields.
- union [xsnor_bus_configs](#)
Union of possible bus configurations.
- struct [xsnor_config](#)
SNOR driver configuration.
- struct [hal_xsnor_base_vmt](#)
Class [hal_xsnor_base_c](#) virtual methods table.
- struct [hal_xsnor_base](#)
Structure representing a XSNOR base driver class.

Macros

- #define [FLASH_ATTR_SPI_4BYTES_ADDR_HINT](#) 0x00008000U
Hint to use 4 bytes addresses in SPI protocol.
- #define [XSNOR_USE_BOTH](#) (([XSNOR_USE_SPI](#) == [TRUE](#)) && ([XSNOR_USE_WSPI](#) == [TRUE](#)))
This switch is [TRUE](#) if both SPI and WSPI are in use.

Bus type and width options

- #define [XSNOR_BUS_MODE_SPI](#) 0U
- #define [XSNOR_BUS_MODE_WSPI_1LINE](#) 1U
- #define [XSNOR_BUS_MODE_WSPI_2LINES](#) 2U
- #define [XSNOR_BUS_MODE_WSPI_4LINES](#) 3U
- #define [XSNOR_BUS_MODE_WSPI_8LINES](#) 4U

Configuration options

- #define `XSNOR_BUFFER_SIZE` 32
Non-cacheable operations buffer.
- #define `XSNOR_USE_SPI` TRUE
SPI support enable switch.
- #define `XSNOR_USE_WSPI` TRUE
WSPI support enable switch.
- #define `XSNOR_SHARED_BUS` TRUE
Bus share support enable switch.

Bus mutex macros when sharing is disabled

- #define `__xsnor_bus_acquire`(self)
- #define `__xsnor_bus_release`(self)

Typedefs

- typedef struct `xsnor_buffers` `xsnor_buffers_t`
Type of a non-cacheable buffer.
- typedef struct `xsnor_config` `xsnor_config_t`
Type of a SNOR configuration structure.
- typedef struct `xsnor_commands` `xsnor_commands_t`
Type of a commands configuration structure.

Class @p hal_xsnor_base_c structures

- typedef struct `hal_xsnor_base` `hal_xsnor_base_c`
Type of a XSNOR base driver class.

Functions

Methods implementations of hal_xsnor_base_c

- void * `__xsnor_objinit_impl` (void *ip, const void *vmt)
Implementation of object creation.
- void `__xsnor_dispose_impl` (void *ip)
Implementation of object finalization.

Regular methods of hal_xsnor_base_c

- void `__xsnor_spi_cmd_addr` (void *ip, uint32_t cmd, `flash_offset_t` offset)
Sends command and address over SPI.
- void `__xsnor_bus_acquire` (void *ip)
Bus acquisition and lock.
- void `__xsnor_bus_release` (void *ip)
Bus release and unlock.
- void `__xsnor_bus_cmd` (void *ip, uint32_t cmd)
Sends a naked command.
- void `__xsnor_bus_cmd_send` (void *ip, uint32_t cmd, size_t n, const uint8_t *p)
Sends a command followed by a data transmit phase.
- void `__xsnor_bus_cmd_receive` (void *ip, uint32_t cmd, size_t n, uint8_t *p)
Sends a command followed by a data receive phase.
- void `__xsnor_bus_cmd_addr` (void *ip, uint32_t cmd, `flash_offset_t` offset)

- Sends a command followed by a flash address.*
- void `__xsnor_bus_cmd_addr_send` (void *ip, uint32_t cmd, `flash_offset_t` offset, size_t n, const uint8_t *p)
- Sends a command followed by a flash address and a data transmit phase.*
- void `__xsnor_bus_cmd_addr_receive` (void *ip, uint32_t cmd, `flash_offset_t` offset, size_t n, uint8_t *p)
- Sends a command followed by a flash address and a data receive phase.*
- void `__xsnor_bus_cmd_dummy_receive` (void *ip, uint32_t cmd, uint32_t dummy, size_t n, uint8_t *p)
- Sends a command followed by dummy cycles and a data receive phase.*
- void `__xsnor_bus_cmd_addr_dummy_receive` (void *ip, uint32_t cmd, `flash_offset_t` offset, uint32_t dummy, size_t n, uint8_t *p)
- Sends a complete header followed by a data receive phase.*
- `flash_error_t` `xsnorStart` (void *ip, const `xsnor_config_t` *config)
- Configures and activates a SNOR driver.*
- void `xsnorStop` (void *ip)
- Deactivates a SNOR driver.*
- `flash_error_t` `xsnorMemoryMap` (void *ip, uint8_t **addrp)
- Enters the memory mapped mode.*
- void `xsnorMemoryUnmap` (void *ip)
- Leaves the memory mapped mode.*

Virtual methods of hal_xsnor_base_c

- static `CC_FORCE_INLINE` `flash_error_t` `xsnor_device_init` (void *ip)
- static `CC_FORCE_INLINE` `flash_error_t` `xsnor_device_read` (void *ip, `flash_offset_t` offset, size_t n, uint8_t *rp)
- Read operation.*
- static `CC_FORCE_INLINE` `flash_error_t` `xsnor_device_program` (void *ip, `flash_offset_t` offset, size_t n, const uint8_t *pp)
- Program operation.*
- static `CC_FORCE_INLINE` `flash_error_t` `xsnor_device_start_erase_all` (void *ip)
- Starts a whole-device erase operation.*
- static `CC_FORCE_INLINE` `flash_error_t` `xsnor_device_start_erase_sector` (void *ip, `flash_sector_t` sector)
- Starts an sector erase operation.*
- static `CC_FORCE_INLINE` `flash_error_t` `xsnor_device_query_erase` (void *ip, unsigned *msec)
- Queries the driver for erase operation progress.*
- static `CC_FORCE_INLINE` `flash_error_t` `xsnor_device_verify_erase` (void *ip, `flash_sector_t` sector)
- Returns the erase state of a sector.*
- static `CC_FORCE_INLINE` `flash_error_t` `xsnor_device_mmap_on` (void *ip, uint8_t **addrp)
- static `CC_FORCE_INLINE` void `xsnor_device_mmap_off` (void *ip)

9.103.1 Detailed Description

Generated SNOR Base Driver header.

Note

This is a generated file, do not edit directly.

9.104 hal_xsnor_base.c File Reference

Generated SNOR Base Driver source.

```
#include "hal.h"
#include "hal_xsnor_base.h"
```

Functions

Interfaces implementation of hal_xsnor_base_c

- static const [flash_descriptor_t](#) * [__xsnor_fls_get_descriptor_impl](#) (void *ip)
Implementation of interface method [flsGetDescriptor\(\)](#).
- static [flash_error_t](#) [__xsnor_fls_read_impl](#) (void *ip, [flash_offset_t](#) offset, [size_t](#) n, [uint8_t](#) *rp)
Implementation of interface method [flsRead\(\)](#).
- static [flash_error_t](#) [__xsnor_fls_program_impl](#) (void *ip, [flash_offset_t](#) offset, [size_t](#) n, const [uint8_t](#) *pp)
Implementation of interface method [flsProgram\(\)](#).
- static [flash_error_t](#) [__xsnor_fls_start_erase_all_impl](#) (void *ip)
Implementation of interface method [flsStartEraseAll\(\)](#).
- static [flash_error_t](#) [__xsnor_fls_start_erase_sector_impl](#) (void *ip, [flash_sector_t](#) sector)
Implementation of interface method [flsStartEraseSector\(\)](#).
- static [flash_error_t](#) [__xsnor_fls_query_erase_impl](#) (void *ip, unsigned *msec)
Implementation of interface method [flsQueryErase\(\)](#).
- static [flash_error_t](#) [__xsnor_fls_verify_erase_impl](#) (void *ip, [flash_sector_t](#) sector)
Implementation of interface method [flsVerifyErase\(\)](#).
- static [flash_error_t](#) [__xsnor_fls_acquire_exclusive_impl](#) (void *ip)
Implementation of interface method [flsAcquireExclusive\(\)](#).
- static [flash_error_t](#) [__xsnor_fls_release_exclusive_impl](#) (void *ip)
Implementation of interface method [flsReleaseExclusive\(\)](#).

Methods implementations of hal_xsnor_base_c

- void * [__xsnor_objinit_impl](#) (void *ip, const void *vmt)
Implementation of object creation.
- void [__xsnor_dispose_impl](#) (void *ip)
Implementation of object finalization.

Regular methods of hal_xsnor_base_c

- void [__xsnor_spi_cmd_addr](#) (void *ip, [uint32_t](#) cmd, [flash_offset_t](#) offset)
Sends command and address over SPI.
- void [__xsnor_bus_acquire](#) (void *ip)
Bus acquisition and lock.
- void [__xsnor_bus_release](#) (void *ip)
Bus release and unlock.
- void [__xsnor_bus_cmd](#) (void *ip, [uint32_t](#) cmd)
Sends a naked command.
- void [__xsnor_bus_cmd_send](#) (void *ip, [uint32_t](#) cmd, [size_t](#) n, const [uint8_t](#) *p)
Sends a command followed by a data transmit phase.
- void [__xsnor_bus_cmd_receive](#) (void *ip, [uint32_t](#) cmd, [size_t](#) n, [uint8_t](#) *p)
Sends a command followed by a data receive phase.
- void [__xsnor_bus_cmd_addr](#) (void *ip, [uint32_t](#) cmd, [flash_offset_t](#) offset)
Sends a command followed by a flash address.
- void [__xsnor_bus_cmd_addr_send](#) (void *ip, [uint32_t](#) cmd, [flash_offset_t](#) offset, [size_t](#) n, const [uint8_t](#) *p)
Sends a command followed by a flash address and a data transmit phase.
- void [__xsnor_bus_cmd_addr_receive](#) (void *ip, [uint32_t](#) cmd, [flash_offset_t](#) offset, [size_t](#) n, [uint8_t](#) *p)
Sends a command followed by a flash address and a data receive phase.
- void [__xsnor_bus_cmd_dummy_receive](#) (void *ip, [uint32_t](#) cmd, [uint32_t](#) dummy, [size_t](#) n, [uint8_t](#) *p)
Sends a command followed by dummy cycles and a data receive phase.
- void [__xsnor_bus_cmd_addr_dummy_receive](#) (void *ip, [uint32_t](#) cmd, [flash_offset_t](#) offset, [uint32_t](#) dummy, [size_t](#) n, [uint8_t](#) *p)
Sends a complete header followed by a data receive phase.
- [flash_error_t](#) [xsnorStart](#) (void *ip, const [xsnor_config_t](#) *config)
Configures and activates a SNOR driver.
- void [xsnorStop](#) (void *ip)
Deactivates a SNOR driver.
- [flash_error_t](#) [xsnorMemoryMap](#) (void *ip, [uint8_t](#) **addrp)
Enters the memory mapped mode.
- void [xsnorMemoryUnmap](#) (void *ip)
Leaves the memory mapped mode.

9.104.1 Detailed Description

Generated SNOR Base Driver source.

Note

This is a generated file, do not edit directly.

9.105 bufstreams.c File Reference

Buffered streams code.

```
#include <string.h>
#include "hal.h"
#include "bufstreams.h"
```

Functions

- static size_t [_writes](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [_reads](#) (void *ip, uint8_t *bp, size_t n)
- static msg_t [_put](#) (void *ip, uint8_t b)
- static msg_t [_get](#) (void *ip)
- static msg_t [_unget](#) (void *ip, uint8_t b)
- void [bsaObjectInit](#) ([BufferedStreamAdapter](#) *bsap, [BaseSequentialStream](#) *bssp)
Buffered stream adapter object initialization.

Variables

- static const struct [BufferedStreamAdapterVMT](#) vmt

9.105.1 Detailed Description

Buffered streams code.

9.106 bufstreams.h File Reference

Buffered streams structures and macros.

Data Structures

- struct [BufferedStreamAdapterVMT](#)
[BufferedStreamAdapter](#) virtual methods table.
- struct [BufferedStreamAdapter](#)
Buffered stream adapter object.

Macros

- #define `BUFSTREAM_BUFFER_SIZE` 1
Buffer size for unget.
- #define `_buffered_stream_adapter_data`
BufferedStreamAdapter specific data.

Functions

- void `bsaObjectInit` (`BufferedStreamAdapter` *bsap, `BaseSequentialStream` *bssp)
Buffered stream adapter object initialization.

9.106.1 Detailed Description

Buffered streams structures and macros.

9.107 chprintf.c File Reference

Mini printf-like functionality.

```
#include "hal.h"
#include "chprintf.h"
#include "memstreams.h"
```

Macros

- #define `MAX_FILLER` 11
- #define `FLOAT_PRECISION` 9

Functions

- static char * `long_to_string_with_divisor` (char *p, long num, unsigned radix, long divisor)
- static char * `ch_ltoa` (char *p, long num, unsigned radix)
- int `chvprintf` (`BaseSequentialStream` *chp, const char *fmt, va_list ap)
System formatted output function.
- int `chprintf` (`BaseSequentialStream` *chp, const char *fmt,...)
System formatted output function.
- int `chsnprintf` (char *str, size_t size, const char *fmt,...)
System formatted output function.
- int `chvsnprintf` (char *str, size_t size, const char *fmt, va_list ap)
System formatted output function.

9.107.1 Detailed Description

Mini printf-like functionality.

9.108 chprintf.h File Reference

Mini printf-like functionality.

```
#include <stdarg.h>
#include "hal.h"
```

Macros

- `#define CHPRINTF_USE_FLOAT FALSE`
Float type support.

Functions

- `int chvprintf (BaseSequentialStream *chp, const char *fmt, va_list ap)`
System formatted output function.
- `int chprintf (BaseSequentialStream *chp, const char *fmt,...)`
System formatted output function.
- `int chsnprintf (char *str, size_t size, const char *fmt,...)`
System formatted output function.
- `int chvsnprintf (char *str, size_t size, const char *fmt, va_list ap)`
System formatted output function.

9.108.1 Detailed Description

Mini printf-like functionality.

9.109 chscanf.c File Reference

Mini scanf-like functionality.

```
#include <ctype.h>
#include "hal.h"
#include "chscanf.h"
#include "memstreams.h"
```

Functions

- `static long sym_to_val (char sym, int base)`
- `int chvscanf (BaseBufferedStream *chp, const char *fmt, va_list ap)`
System formatted input function.
- `int chscanf (BaseBufferedStream *chp, const char *fmt,...)`
System formatted input function.
- `int chnsnscanf (char *str, size_t size, const char *fmt,...)`
System formatted input function.
- `int chvnsnscanf (char *str, size_t size, const char *fmt, va_list ap)`
System formatted input function.

9.109.1 Detailed Description

Mini scanf-like functionality.

9.110 chscanf.h File Reference

Mini scanf-like functionality.

```
#include <stdarg.h>
#include "bufstreams.h"
```

Macros

- `#define CHSCANF_USE_FLOAT FALSE`
Float type support.

Functions

- `int chvscanf (BaseBufferedStream *chp, const char *fmt, va_list ap)`
System formatted input function.
- `int chscanf (BaseBufferedStream *chp, const char *fmt,...)`
System formatted input function.
- `int chnsnscanf (char *str, size_t size, const char *fmt,...)`
System formatted input function.
- `int chvnsnscanf (char *str, size_t size, const char *fmt, va_list ap)`
System formatted input function.

9.110.1 Detailed Description

Mini scanf-like functionality.

9.111 memstreams.c File Reference

Memory streams code.

```
#include <string.h>
#include "hal.h"
#include "memstreams.h"
```

Functions

- static `size_t _writes` (`void *ip, const uint8_t *bp, size_t n`)
- static `size_t _reads` (`void *ip, uint8_t *bp, size_t n`)
- static `msg_t _put` (`void *ip, uint8_t b`)
- static `msg_t _get` (`void *ip`)
- static `msg_t _unget` (`void *ip, uint8_t b`)
- void `msObjectInit` (`MemoryStream *msp, uint8_t *buffer, size_t size, size_t eos`)
Memory stream object initialization.

Variables

- static const struct `MemStreamVMT vmt` = {(size_t)0, `_writes`, `_reads`, `_put`, `_get`, `_unget`}

9.111.1 Detailed Description

Memory streams code.

9.112 memstreams.h File Reference

Memory streams structures and macros.

Data Structures

- struct `MemStreamVMT`
MemStream virtual methods table.
- struct `MemoryStream`
Memory stream object.

Macros

- `#define _memory_stream_data`
MemStream specific data.

Functions

- void `msObjectInit` (`MemoryStream *msp, uint8_t *buffer, size_t size, size_t eos`)
Memory stream object initialization.

9.112.1 Detailed Description

Memory streams structures and macros.

9.113 nullstreams.c File Reference

Null streams code.

```
#include "hal.h"
#include "nullstreams.h"
```

Functions

- static size_t [writes](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [reads](#) (void *ip, uint8_t *bp, size_t n)
- static msg_t [put](#) (void *ip, uint8_t b)
- static msg_t [get](#) (void *ip)
- static msg_t [unget](#) (void *ip, uint8_t b)
- void [nullObjectInit](#) (NullStream *nsp)

Null stream object initialization.

Variables

- static const struct [NullStreamVMT](#) vmt = {(size_t)0, [writes](#), [reads](#), [put](#), [get](#), [unget](#)}

9.113.1 Detailed Description

Null streams code.

9.114 nullstreams.h File Reference

Null streams structures and macros.

Data Structures

- struct [NullStreamVMT](#)
NullStream virtual methods table.
- struct [NullStream](#)
Null stream object.

Macros

- #define [_null_stream_data_base_buffered_stream_data](#)
NullStream specific data.

Functions

- void [nullObjectInit](#) (NullStream *nsp)
Null stream object initialization.

9.114.1 Detailed Description

Null streams structures and macros.

9.115 hal.c File Reference

HAL subsystem code.

```
#include "hal.h"
```

Functions

- void [hallnit](#) (void)
HAL initialization.

9.115.1 Detailed Description

HAL subsystem code.

9.116 hal_adc.c File Reference

ADC Driver code.

```
#include "hal.h"
```

Functions

- void [adclnit](#) (void)
ADC Driver initialization.
- void [adcObjectInit](#) (ADCDriver *adcp)
Initializes the standard part of a [ADCDriver](#) structure.
- [msg_t](#) [adcStart](#) (ADCDriver *adcp, const [ADCConfig](#) *config)
Configures and activates the ADC peripheral.
- void [adcStop](#) (ADCDriver *adcp)
Deactivates the ADC peripheral.
- void [adcStartConversion](#) (ADCDriver *adcp, const [ADCConversionGroup](#) *grpp, [adcsample_t](#) *samples, [size_t](#) depth)
Starts an ADC conversion.
- void [adcStartConversionI](#) (ADCDriver *adcp, const [ADCConversionGroup](#) *grpp, [adcsample_t](#) *samples, [size_t](#) depth)
Starts an ADC conversion.
- void [adcStopConversion](#) (ADCDriver *adcp)
Stops an ongoing conversion.
- void [adcStopConversionI](#) (ADCDriver *adcp)
Stops an ongoing conversion.
- [msg_t](#) [adcConvert](#) (ADCDriver *adcp, const [ADCConversionGroup](#) *grpp, [adcsample_t](#) *samples, [size_t](#) depth)
Performs an ADC conversion.
- void [adcAcquireBus](#) (ADCDriver *adcp)
Gains exclusive access to the ADC peripheral.
- void [adcReleaseBus](#) (ADCDriver *adcp)
Releases exclusive access to the ADC peripheral.

9.116.1 Detailed Description

ADC Driver code.

9.117 hal_buffered_serial.c File Reference

```
#include "hal.h"
```

Functions

- void [bsIncomingData](#) ([BufferedSerial](#) *bsp, uint8_t b)
Handles incoming data.
- [msg_t bsRequestData](#) ([BufferedSerial](#) *bsp)
Handles outgoing data.

9.118 hal_buffers.c File Reference

I/O Buffers code.

```
#include <string.h>
#include "hal.h"
```

Functions

- void [ibqObjectInit](#) ([input_buffers_queue_t](#) *ibqp, bool suspended, uint8_t *bp, size_t size, size_t n, [bqnotify_t](#) infy, void *link)
Initializes an input buffers queue object.
- void [ibqResetI](#) ([input_buffers_queue_t](#) *ibqp)
Resets an input buffers queue.
- uint8_t * [ibqGetEmptyBufferI](#) ([input_buffers_queue_t](#) *ibqp)
Gets the next empty buffer from the queue.
- void [ibqPostFullBufferI](#) ([input_buffers_queue_t](#) *ibqp, size_t size)
Posts a new filled buffer to the queue.
- [msg_t ibqGetFullBufferTimeout](#) ([input_buffers_queue_t](#) *ibqp, [sysinterval_t](#) timeout)
Gets the next filled buffer from the queue.
- [msg_t ibqGetFullBufferTimeoutS](#) ([input_buffers_queue_t](#) *ibqp, [sysinterval_t](#) timeout)
Gets the next filled buffer from the queue.
- void [ibqReleaseEmptyBuffer](#) ([input_buffers_queue_t](#) *ibqp)
Releases the buffer back in the queue.
- void [ibqReleaseEmptyBufferS](#) ([input_buffers_queue_t](#) *ibqp)
Releases the buffer back in the queue.
- [msg_t ibqGetTimeout](#) ([input_buffers_queue_t](#) *ibqp, [sysinterval_t](#) timeout)
Input queue read with timeout.
- size_t [ibqReadTimeout](#) ([input_buffers_queue_t](#) *ibqp, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
Input queue read with timeout.

- void `obqObjectInit` (`output_buffers_queue_t` *obqp, bool suspended, `uint8_t` *bp, `size_t` size, `size_t` n, `bqnotify_t` onfy, void *link)
Initializes an output buffers queue object.
- void `obqResetI` (`output_buffers_queue_t` *obqp)
Resets an output buffers queue.
- `uint8_t` * `obqGetFullBufferI` (`output_buffers_queue_t` *obqp, `size_t` *sizep)
Gets the next filled buffer from the queue.
- void `obqReleaseEmptyBufferI` (`output_buffers_queue_t` *obqp)
Releases the next filled buffer back in the queue.
- `msg_t` `obqGetEmptyBufferTimeout` (`output_buffers_queue_t` *obqp, `sysinterval_t` timeout)
Gets the next empty buffer from the queue.
- `msg_t` `obqGetEmptyBufferTimeoutS` (`output_buffers_queue_t` *obqp, `sysinterval_t` timeout)
Gets the next empty buffer from the queue.
- void `obqPostFullBuffer` (`output_buffers_queue_t` *obqp, `size_t` size)
Posts a new filled buffer to the queue.
- void `obqPostFullBufferS` (`output_buffers_queue_t` *obqp, `size_t` size)
Posts a new filled buffer to the queue.
- `msg_t` `obqPutTimeout` (`output_buffers_queue_t` *obqp, `uint8_t` b, `sysinterval_t` timeout)
Output queue write with timeout.
- `size_t` `obqWriteTimeout` (`output_buffers_queue_t` *obqp, const `uint8_t` *bp, `size_t` n, `sysinterval_t` timeout)
Output queue write with timeout.
- bool `obqTryFlushI` (`output_buffers_queue_t` *obqp)
Flushes the current, partially filled, buffer to the queue.
- void `obqFlush` (`output_buffers_queue_t` *obqp)
Flushes the current, partially filled, buffer to the queue.

9.118.1 Detailed Description

I/O Buffers code.

9.119 hal_can.c File Reference

CAN Driver code.

```
#include "hal.h"
```

Functions

- void `canInit` (void)
CAN Driver initialization.
- void `canObjectInit` (`CANDriver` *canp)
Initializes the standard part of a `CANDriver` structure.
- `msg_t` `canStart` (`CANDriver` *canp, const `CANConfig` *config)
Configures and activates the CAN peripheral.
- void `canStop` (`CANDriver` *canp)
Deactivates the CAN peripheral.
- bool `canTryTransmitI` (`CANDriver` *canp, `canmbx_t` mailbox, const `CANTxFrame` *ctfp)

- Can frame transmission attempt.*
- bool `canTryReceiveI` (`CANDriver` *canp, `canmbx_t` mailbox, `CANRxFrame` *crfp)
- Can frame receive attempt.*
- void `canTryAbortX` (`CANDriver` *canp, `canmbx_t` mailbox)
- Tries to abort an ongoing transmission.*
- `msg_t` `canTransmitTimeout` (`CANDriver` *canp, `canmbx_t` mailbox, const `CANTxFrame` *ctfp, `sysinterval_t` timeout)
- Can frame transmission.*
- `msg_t` `canReceiveTimeout` (`CANDriver` *canp, `canmbx_t` mailbox, `CANRxFrame` *crfp, `sysinterval_t` timeout)
- Can frame receive.*
- void `canSleep` (`CANDriver` *canp)
- Enters the sleep mode.*
- void `canWakeup` (`CANDriver` *canp)
- Enforces leaving the sleep mode.*

9.119.1 Detailed Description

CAN Driver code.

9.120 hal_crypto.c File Reference

Cryptographic Driver code.

```
#include "hal.h"
```

Functions

- void `cryInit` (void)
- Cryptographic Driver initialization.*
- void `cryObjectInit` (`CRYDriver` *cryp)
- Initializes the standard part of a `CRYDriver` structure.*
- `msg_t` `cryStart` (`CRYDriver` *cryp, const `CRYConfig` *config)
- Configures and activates the cryptographic peripheral.*
- void `cryStop` (`CRYDriver` *cryp)
- Deactivates the cryptographic peripheral.*
- `cryerror_t` `cryLoadAESTransientKey` (`CRYDriver` *cryp, `size_t` size, const `uint8_t` *keyp)
- Initializes the AES transient key.*
- `cryerror_t` `cryEncryptAES` (`CRYDriver` *cryp, `crykey_t` key_id, const `uint8_t` *in, `uint8_t` *out)
- Encryption of a single block using AES.*
- `cryerror_t` `cryDecryptAES` (`CRYDriver` *cryp, `crykey_t` key_id, const `uint8_t` *in, `uint8_t` *out)
- Decryption of a single block using AES.*
- `cryerror_t` `cryEncryptAES_ECB` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, const `uint8_t` *in, `uint8_t` *out)
- Encryption operation using AES-ECB.*
- `cryerror_t` `cryDecryptAES_ECB` (`CRYDriver` *cryp, `crykey_t` key_id, `size_t` size, const `uint8_t` *in, `uint8_t` *out)
- Decryption operation using AES-ECB.*

- [cryerror_t cryEncryptAES_CBC](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CBC.
- [cryerror_t cryDecryptAES_CBC](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CBC.
- [cryerror_t cryEncryptAES_CFB](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CFB.
- [cryerror_t cryDecryptAES_CFB](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CFB.
- [cryerror_t cryEncryptAES_CTR](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CTR.
- [cryerror_t cryDecryptAES_CTR](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CTR.
- [cryerror_t cryEncryptAES_GCM](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, uint8_t *tag_out)
Encryption operation using AES-GCM.
- [cryerror_t cryDecryptAES_GCM](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, const uint8_t *tag_in)
Decryption operation using AES-GCM.
- [cryerror_t cryLoadDESTransientKey](#) ([CRYDriver](#) *cryp, size_t size, const uint8_t *keyp)
Initializes the DES transient key.
- [cryerror_t cryEncryptDES](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Encryption of a single block using (T)DES.
- [cryerror_t cryDecryptDES](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Decryption of a single block using (T)DES.
- [cryerror_t cryEncryptDES_ECB](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Encryption operation using (T)DES-ECB.
- [cryerror_t cryDecryptDES_ECB](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Decryption operation using (T)DES-ECB.
- [cryerror_t cryEncryptDES_CBC](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using (T)DES-CBC.
- [cryerror_t cryDecryptDES_CBC](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using (T)DES-CBC.
- [cryerror_t crySHA1Init](#) ([CRYDriver](#) *cryp, [SHA1Context](#) *sha1ctxp)
Hash initialization using SHA1.
- [cryerror_t crySHA1Update](#) ([CRYDriver](#) *cryp, [SHA1Context](#) *sha1ctxp, size_t size, const uint8_t *in)
Hash update using SHA1.
- [cryerror_t crySHA1Final](#) ([CRYDriver](#) *cryp, [SHA1Context](#) *sha1ctxp, uint8_t *out)
Hash finalization using SHA1.
- [cryerror_t crySHA256Init](#) ([CRYDriver](#) *cryp, [SHA256Context](#) *sha256ctxp)
Hash initialization using SHA256.
- [cryerror_t crySHA256Update](#) ([CRYDriver](#) *cryp, [SHA256Context](#) *sha256ctxp, size_t size, const uint8_t *in)

- Hash update using SHA256.*
- `cryerror_t crySHA256Final` (`CRYDriver *cryp`, `SHA256Context *sha256ctxp`, `uint8_t *out`)
- Hash finalization using SHA256.*
- `cryerror_t crySHA512Init` (`CRYDriver *cryp`, `SHA512Context *sha512ctxp`)
- Hash initialization using SHA512.*
- `cryerror_t crySHA512Update` (`CRYDriver *cryp`, `SHA512Context *sha512ctxp`, `size_t size`, `const uint8_t *in`)
- Hash update using SHA512.*
- `cryerror_t crySHA512Final` (`CRYDriver *cryp`, `SHA512Context *sha512ctxp`, `uint8_t *out`)
- Hash finalization using SHA512.*
- `cryerror_t cryLoadHMACTransientKey` (`CRYDriver *cryp`, `size_t size`, `const uint8_t *keyp`)
- Initializes the HMAC transient key.*
- `cryerror_t cryHMACSHA256Init` (`CRYDriver *cryp`, `HMACSHA256Context *hmacsha256ctxp`)
- Hash initialization using HMAC_SHA256.*
- `cryerror_t cryHMACSHA256Update` (`CRYDriver *cryp`, `HMACSHA256Context *hmacsha256ctxp`, `size_t size`, `const uint8_t *in`)
- Hash update using HMAC.*
- `cryerror_t cryHMACSHA256Final` (`CRYDriver *cryp`, `HMACSHA256Context *hmacsha256ctxp`, `uint8_t *out`)
- Hash finalization using HMAC.*
- `cryerror_t cryHMACSHA512Init` (`CRYDriver *cryp`, `HMACSHA512Context *hmacsha512ctxp`)
- Hash initialization using HMAC_SHA512.*
- `cryerror_t cryHMACSHA512Update` (`CRYDriver *cryp`, `HMACSHA512Context *hmacsha512ctxp`, `size_t size`, `const uint8_t *in`)
- Hash update using HMAC.*
- `cryerror_t cryHMACSHA512Final` (`CRYDriver *cryp`, `HMACSHA512Context *hmacsha512ctxp`, `uint8_t *out`)
- Hash finalization using HMAC.*

9.120.1 Detailed Description

Cryptographic Driver code.

9.121 hal_dac.c File Reference

DAC Driver code.

```
#include "hal.h"
```

Functions

- void `daclnit` (void)
- DAC Driver initialization.*
- void `dacObjectInit` (`DACDriver *daccp`)
- Initializes the standard part of a `DACDriver` structure.*
- `msg_t dacStart` (`DACDriver *daccp`, `const DACConfig *config`)
- Configures and activates the DAC peripheral.*
- void `dacStop` (`DACDriver *daccp`)
- Deactivates the DAC peripheral.*
- `msg_t dacPutChannelX` (`DACDriver *daccp`, `dacchannel_t channel`, `dacsample_t sample`)

- Outputs a value directly on a DAC channel.*
- `msg_t dacStartConversion` (`DACDriver *daccp`, const `DACConversionGroup *grpp`, `dacsample_t *samples`, `size_t depth`)
Starts a DAC conversion.
- `msg_t dacStartConversionI` (`DACDriver *daccp`, const `DACConversionGroup *grpp`, `dacsample_t *samples`, `size_t depth`)
Starts a DAC conversion.
- `void dacStopConversion` (`DACDriver *daccp`)
Stops an ongoing conversion.
- `void dacStopConversionI` (`DACDriver *daccp`)
Stops an ongoing conversion.
- `msg_t dacConvert` (`DACDriver *daccp`, const `DACConversionGroup *grpp`, `dacsample_t *samples`, `size_t depth`)
Performs a DAC conversion.
- `msg_t dacSynchronizeS` (`DACDriver *daccp`, `sysinterval_t timeout`)
Synchronize to a conversion completion.
- `msg_t dacSynchronize` (`DACDriver *daccp`, `sysinterval_t timeout`)
Synchronize to a conversion completion.
- `void dacAcquireBus` (`DACDriver *daccp`)
Gains exclusive access to the DAC bus.
- `void dacReleaseBus` (`DACDriver *daccp`)
Releases exclusive access to the DAC bus.

9.121.1 Detailed Description

DAC Driver code.

9.122 hal_efl.c File Reference

Embedded Flash Driver code.

```
#include "hal.h"
```

Functions

- static `flash_error_t efl_acquire_exclusive` (`void *instance`)
- static `flash_error_t efl_release_exclusive` (`void *instance`)
- `void eflInit` (`void`)
Embedded Flash Driver initialization.
- `void eflObjectInit` (`EFlashDriver *eflp`)
Initializes a generic `EFlashDriver` object.
- `msg_t eflStart` (`EFlashDriver *eflp`, const `EFlashConfig *config`)
Configures and starts the driver.
- `void eflStop` (`EFlashDriver *eflp`)
Stops the driver.

Variables

- static const struct [EFlashDriverVMT vmt](#)

9.122.1 Detailed Description

Embedded Flash Driver code.

9.123 hal_flash.c File Reference

Generic flash driver class code.

```
#include "hal.h"
#include "hal_flash.h"
```

Functions

- [flash_error_t flashWaitErase](#) ([BaseFlash](#) *devp)
Waits until the current erase operation is finished.
- [flash_offset_t flashGetSectorOffset](#) ([BaseFlash](#) *devp, [flash_sector_t](#) sector)
Returns the offset of a sector.
- [uint32_t flashGetSectorSize](#) ([BaseFlash](#) *devp, [flash_sector_t](#) sector)
Returns the size of a sector.
- [flash_sector_t flashGetOffsetSector](#) ([BaseFlash](#) *devp, [flash_offset_t](#) offset)
Returns the sector of an offset.
- [void * flashGetOffsetAddress](#) ([BaseFlash](#) *devp, [flash_offset_t](#) offset)
Get absolute address from offset.
- [flash_offset_t flashGetAddressOffset](#) ([BaseFlash](#) *devp, [void *addr](#))
Get offset from absolute address.

9.123.1 Detailed Description

Generic flash driver class code.

9.124 hal_gpt.c File Reference

GPT Driver code.

```
#include "hal.h"
```


Functions

- void [gptInit](#) (void)
GPT Driver initialization.
- void [gptObjectInit](#) (GPTDriver *gtp)
Initializes the standard part of a [GPTDriver](#) structure.
- [msg_t](#) [gptStart](#) (GPTDriver *gtp, const GPTConfig *config)
Configures and activates the GPT peripheral.
- void [gptStop](#) (GPTDriver *gtp)
Deactivates the GPT peripheral.
- void [gptChangeInterval](#) (GPTDriver *gtp, [gptcnt_t](#) interval)
Changes the interval of GPT peripheral.
- void [gptStartContinuous](#) (GPTDriver *gtp, [gptcnt_t](#) interval)
Starts the timer in continuous mode.
- void [gptStartContinuousl](#) (GPTDriver *gtp, [gptcnt_t](#) interval)
Starts the timer in continuous mode.
- void [gptStartOneShot](#) (GPTDriver *gtp, [gptcnt_t](#) interval)
Starts the timer in one shot mode.
- void [gptStartOneShotl](#) (GPTDriver *gtp, [gptcnt_t](#) interval)
Starts the timer in one shot mode.
- void [gptStopTimer](#) (GPTDriver *gtp)
Stops the timer.
- void [gptStopTimerl](#) (GPTDriver *gtp)
Stops the timer.
- void [gptPolledDelay](#) (GPTDriver *gtp, [gptcnt_t](#) interval)
Starts the timer in one shot mode and waits for completion.

9.124.1 Detailed Description

GPT Driver code.

9.125 hal_i2c.c File Reference

I2C Driver code.

```
#include "hal.h"
```

Functions

- void [i2cInit](#) (void)
I2C Driver initialization.
- void [i2cObjectInit](#) (I2CDriver *i2cp)
Initializes the standard part of a [I2CDriver](#) structure.
- [msg_t](#) [i2cStart](#) (I2CDriver *i2cp, const I2CConfig *config)
Configures and activates the I2C peripheral.
- void [i2cStop](#) (I2CDriver *i2cp)
Deactivates the I2C peripheral.

- [i2cflags_t i2cGetErrors](#) ([I2CDriver](#) *i2cp)
Returns the errors mask associated to the previous operation.
- [msg_t i2cMasterTransmitTimeout](#) ([I2CDriver](#) *i2cp, [i2caddr_t](#) addr, const [uint8_t](#) *txbuf, [size_t](#) txbytes, [uint8_t](#) *rxbuf, [size_t](#) rxbytes, [sysinterval_t](#) timeout)
Sends data via the I2C bus.
- [msg_t i2cMasterReceiveTimeout](#) ([I2CDriver](#) *i2cp, [i2caddr_t](#) addr, [uint8_t](#) *rxbuf, [size_t](#) rxbytes, [sysinterval_t](#) timeout)
Receives data from the I2C bus.
- void [i2cAcquireBus](#) ([I2CDriver](#) *i2cp)
Gains exclusive access to the I2C bus.
- void [i2cReleaseBus](#) ([I2CDriver](#) *i2cp)
Releases exclusive access to the I2C bus.
- [msg_t i2cSlaveMatchAddress](#) ([I2CDriver](#) *i2cp, [i2caddr_t](#) addr)
Listen I2C bus for address match.
- [msg_t i2cSlaveReceiveTimeout](#) ([I2CDriver](#) *i2cp, [uint8_t](#) *rxbuf, [size_t](#) rxbytes, [sysinterval_t](#) timeout)
Receive data via the I2C bus as slave and call handler.
- [msg_t i2cSlaveTransmitTimeout](#) ([I2CDriver](#) *i2cp, const [uint8_t](#) *txbuf, [size_t](#) txbytes, [sysinterval_t](#) timeout)
Transmits data via the I2C bus as slave.

9.125.1 Detailed Description

I2C Driver code.

9.126 hal_i2s.c File Reference

I2S Driver code.

```
#include "hal.h"
```

Functions

- void [i2sInit](#) (void)
I2S Driver initialization.
- void [i2sObjectInit](#) ([I2SDriver](#) *i2sp)
Initializes the standard part of a [I2SDriver](#) structure.
- [msg_t i2sStart](#) ([I2SDriver](#) *i2sp, const [I2SConfig](#) *config)
Configures and activates the I2S peripheral.
- void [i2sStop](#) ([I2SDriver](#) *i2sp)
Deactivates the I2S peripheral.
- void [i2sStartExchange](#) ([I2SDriver](#) *i2sp)
Starts a I2S data exchange.
- void [i2sStopExchange](#) ([I2SDriver](#) *i2sp)
Stops the ongoing data exchange.

9.126.1 Detailed Description

I2S Driver code.

9.127 hal_icu.c File Reference

ICU Driver code.

```
#include "hal.h"
```

Functions

- void [icuInit](#) (void)
ICU Driver initialization.
- void [icuObjectInit](#) (ICUDriver *icup)
Initializes the standard part of a [ICUDriver](#) structure.
- [msg_t icuStart](#) (ICUDriver *icup, const [ICUConfig](#) *config)
Configures and activates the ICU peripheral.
- void [icuStop](#) (ICUDriver *icup)
Deactivates the ICU peripheral.
- void [icuStartCapture](#) (ICUDriver *icup)
Starts the input capture.
- bool [icuWaitCapture](#) (ICUDriver *icup)
Waits for a completed capture.
- void [icuStopCapture](#) (ICUDriver *icup)
Stops the input capture.
- void [icuEnableNotifications](#) (ICUDriver *icup)
Enables notifications.
- void [icuDisableNotifications](#) (ICUDriver *icup)
Disables notifications.

9.127.1 Detailed Description

ICU Driver code.

9.128 hal_mac.c File Reference

MAC Driver code.

```
#include "hal.h"
```

Functions

- void `macInit` (void)
MAC Driver initialization.
- void `macObjectInit` (MACDriver *macp)
Initialize the standard part of a `MACDriver` structure.
- `msg_t` `macStart` (MACDriver *macp, const MACConfig *config)
Configures and activates the MAC peripheral.
- void `macStop` (MACDriver *macp)
Deactivates the MAC peripheral.
- `eventflags_t` `macGetAndClearEvents` (MACDriver *macp)
Get and clears MAC event flags.
- `msg_t` `macWaitTransmitDescriptor` (MACDriver *macp, MACTransmitDescriptor *tdp, `sysinterval_t` timeout)
Allocates a transmission descriptor.
- `msg_t` `macWaitReceiveDescriptor` (MACDriver *macp, MACReceiveDescriptor *rdp, `sysinterval_t` timeout)
Waits for a received frame.
- bool `macPollLinkStatus` (MACDriver *macp)
Updates and returns the link status.

9.128.1 Detailed Description

MAC Driver code.

9.129 hal_mmc_spi.c File Reference

MMC over SPI driver code.

```
#include <string.h>
#include "hal.h"
```

Functions

- static bool `mmc_is_card_inserted` (void *instance)
- static bool `mmc_is_write_protected` (void *instance)
- static bool `mmc_connect` (void *instance)
- static bool `mmc_disconnect` (void *instance)
- static bool `mmc_read` (void *instance, `uint32_t` startblk, `uint8_t` *buffer, `uint32_t` n)
- static bool `mmc_write` (void *instance, `uint32_t` startblk, const `uint8_t` *buffer, `uint32_t` n)
- static bool `mmc_sync` (void *instance)
- static bool `mmc_get_info` (void *instance, BlockDeviceInfo *bdip)
- static `uint8_t` `mmc_crc7` (`uint8_t` crc, const `uint8_t` *buffer, `size_t` len)
Calculate the MMC standard CRC-7 based on a lookup table.
- static bool `mmc_wait_idle` (MMCDriver *mmcp)
Waits an idle condition.
- static bool `mmc_send_hdr` (MMCDriver *mmcp, `uint8_t` cmd, `uint32_t` arg)
Sends a command header.
- static bool `mmc_recvr1` (MMCDriver *mmcp, `uint8_t` *r1p)

- Receives a single byte response.*
- static bool [mmc_recvr3](#) ([MMCDriver](#) *mmcp, uint8_t *r1p)
- Receives a three byte response.*
- static bool [mmc_send_command_R1](#) ([MMCDriver](#) *mmcp, uint8_t cmd, uint32_t arg, uint8_t *r1p)
- Sends a command and returns a single byte response.*
- static bool [mmc_send_command_R3](#) ([MMCDriver](#) *mmcp, uint8_t cmd, uint32_t arg, uint8_t *r1p)
- Sends a command which returns a five bytes response (R3).*
- static bool [mmc_read_CxD](#) ([MMCDriver](#) *mmcp, uint8_t cmd, uint32_t cxd[4])
- Reads the CSD.*
- void [mmcInit](#) (void)
- MMC over SPI driver initialization.*
- void [mmcObjectInit](#) ([MMCDriver](#) *mmcp, uint8_t *buffer)
- Initializes an instance.*
- [msg_t](#) [mmcStart](#) ([MMCDriver](#) *mmcp, const [MMCConfig](#) *config)
- Configures and activates the MMC peripheral.*
- void [mmcStop](#) ([MMCDriver](#) *mmcp)
- Disables the MMC peripheral.*
- bool [mmcConnect](#) ([MMCDriver](#) *mmcp)
- Performs the initialization procedure on the inserted card.*
- bool [mmcDisconnect](#) ([MMCDriver](#) *mmcp)
- Brings the driver in a state safe for card removal.*
- bool [mmcStartSequentialRead](#) ([MMCDriver](#) *mmcp, uint32_t startblk)
- Starts a sequential read.*
- bool [mmcSequentialRead](#) ([MMCDriver](#) *mmcp, uint8_t *buffer)
- Reads a block within a sequential read operation.*
- bool [mmcStopSequentialRead](#) ([MMCDriver](#) *mmcp)
- Stops a sequential read gracefully.*
- bool [mmcStartSequentialWrite](#) ([MMCDriver](#) *mmcp, uint32_t startblk)
- Starts a sequential write.*
- bool [mmcSequentialWrite](#) ([MMCDriver](#) *mmcp, const uint8_t *buffer)
- Writes a block within a sequential write operation.*
- bool [mmcStopSequentialWrite](#) ([MMCDriver](#) *mmcp)
- Stops a sequential write gracefully.*
- bool [mmcSync](#) ([MMCDriver](#) *mmcp)
- Waits for card idle condition.*
- bool [mmcGetInfo](#) ([MMCDriver](#) *mmcp, [BlockDeviceInfo](#) *bdip)
- Returns the media info.*
- bool [mmcErase](#) ([MMCDriver](#) *mmcp, uint32_t startblk, uint32_t endblk)
- Erases blocks.*

Variables

- static const struct [mmc_spi_driver_vmt](#) [mmc_vmt](#)
- Virtual methods table.*
- static const uint8_t [mmc_crc7_lookup_table](#) [256]
- Lookup table for CRC-7 (based on polynomial $x^7 + x^3 + 1$).*

9.129.1 Detailed Description

MMC over SPI driver code.

9.130 hal_mmcsd.c File Reference

MMC/SD cards common code.

```
#include "hal.h"
```

Functions

- uint32_t [_mmcsd_get_slice](#) (const uint32_t *data, uint32_t end, uint32_t start)
Gets a bit field from a words array.
- uint32_t [_mmcsd_get_capacity](#) (const uint32_t *csd)
Extract card capacity from a CSD.
- uint32_t [_mmcsd_get_capacity_ext](#) (const uint8_t *ext_csd)
Extract MMC card capacity from EXT_CSD.
- void [_mmcsd_unpack_sdc_cid](#) (const MMCSDBlockDevice *sdcp, [unpacked_sdc_cid_t](#) *cid_sdc)
Unpacks SDC CID array in structure.
- void [_mmcsd_unpack_mmc_cid](#) (const MMCSDBlockDevice *sdcp, [unpacked_mmc_cid_t](#) *cid_mmc)
Unpacks MMC CID array in structure.
- void [_mmcsd_unpack_csd_mmc](#) (const MMCSDBlockDevice *sdcp, [unpacked_mmc_csd_t](#) *csd_mmc)
Unpacks MMC CSD array in structure.
- void [_mmcsd_unpack_csd_v10](#) (const MMCSDBlockDevice *sdcp, [unpacked_sdc_csd_10_t](#) *csd10)
Unpacks SDC CSD v1.0 array in structure.
- void [_mmcsd_unpack_csd_v20](#) (const MMCSDBlockDevice *sdcp, [unpacked_sdc_csd_20_t](#) *csd20)
Unpacks SDC CSD v2.0 array in structure.

9.130.1 Detailed Description

MMC/SD cards common code.

9.131 hal_pal.c File Reference

I/O Ports Abstraction Layer code.

```
#include "hal.h"
```

Functions

- `ioportmask_t palReadBus` (const `IOBus` *bus)
Read from an I/O bus.
- void `palWriteBus` (const `IOBus` *bus, `ioportmask_t` bits)
Write to an I/O bus.
- void `palSetBusMode` (const `IOBus` *bus, `iomode_t` mode)
Programs a bus with the specified mode.
- void `palSetPadCallbackI` (`ioportid_t` port, `iopadid_t` pad, `palcallback_t` cb, void *arg)
Associates a callback to a port/pad.
- void `palSetLineCallbackI` (`ioline_t` line, `palcallback_t` cb, void *arg)
Associates a callback to a line.
- `msg_t palWaitPadTimeoutS` (`ioportid_t` port, `iopadid_t` pad, `sysinterval_t` timeout)
Waits for an edge on the specified port/pad.
- `msg_t palWaitPadTimeout` (`ioportid_t` port, `iopadid_t` pad, `sysinterval_t` timeout)
Waits for an edge on the specified port/pad.
- `msg_t palWaitLineTimeoutS` (`ioline_t` line, `sysinterval_t` timeout)
Waits for an edge on the specified line.
- `msg_t palWaitLineTimeout` (`ioline_t` line, `sysinterval_t` timeout)
Waits for an edge on the specified line.

9.131.1 Detailed Description

I/O Ports Abstraction Layer code.

9.132 hal_pwm.c File Reference

PWM Driver code.

```
#include "hal.h"
```

Functions

- void `pwmInit` (void)
PWM Driver initialization.
- void `pwmObjectInit` (`PWMDriver` *pwmp)
Initializes the standard part of a `PWMDriver` structure.
- `msg_t pwmStart` (`PWMDriver` *pwmp, const `PWMConfig` *config)
Configures and activates the PWM peripheral.
- void `pwmStop` (`PWMDriver` *pwmp)
Deactivates the PWM peripheral.
- void `pwmChangePeriod` (`PWMDriver` *pwmp, `pwmcnt_t` period)
Changes the period the PWM peripheral.
- void `pwmEnableChannel` (`PWMDriver` *pwmp, `pwmchannel_t` channel, `pwmcnt_t` width)
Enables a PWM channel.
- void `pwmDisableChannel` (`PWMDriver` *pwmp, `pwmchannel_t` channel)
Disables a PWM channel and its notification.

- void [pwmEnablePeriodicNotification](#) ([PWMDriver](#) *pwmp)
Enables the periodic activation edge notification.
- void [pwmDisablePeriodicNotification](#) ([PWMDriver](#) *pwmp)
Disables the periodic activation edge notification.
- void [pwmEnableChannelNotification](#) ([PWMDriver](#) *pwmp, [pwmchannel_t](#) channel)
Enables a channel de-activation edge notification.
- void [pwmDisableChannelNotification](#) ([PWMDriver](#) *pwmp, [pwmchannel_t](#) channel)
Disables a channel de-activation edge notification.

9.132.1 Detailed Description

PWM Driver code.

9.133 hal_queues.c File Reference

I/O Queues code.

```
#include <string.h>
#include "hal.h"
```

Functions

- static size_t [iq_read](#) ([input_queue_t](#) *iqp, uint8_t *bp, size_t n)
Non-blocking input queue read.
- static size_t [oq_write](#) ([output_queue_t](#) *oqp, const uint8_t *bp, size_t n)
Non-blocking output queue write.
- void [iqObjectInit](#) ([input_queue_t](#) *iqp, uint8_t *bp, size_t size, [qnotify_t](#) infy, void *link)
Initializes an input queue.
- void [iqResetI](#) ([input_queue_t](#) *iqp)
Resets an input queue.
- [msg_t](#) [iqPutI](#) ([input_queue_t](#) *iqp, uint8_t b)
Input queue write.
- [msg_t](#) [iqGetI](#) ([input_queue_t](#) *iqp)
Input queue non-blocking read.
- [msg_t](#) [iqGetTimeout](#) ([input_queue_t](#) *iqp, [sysinterval_t](#) timeout)
Input queue read with timeout.
- size_t [iqReadI](#) ([input_queue_t](#) *iqp, uint8_t *bp, size_t n)
Input queue non-blocking read.
- size_t [iqReadTimeout](#) ([input_queue_t](#) *iqp, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
Input queue read with timeout.
- void [oqObjectInit](#) ([output_queue_t](#) *oqp, uint8_t *bp, size_t size, [qnotify_t](#) onfy, void *link)
Initializes an output queue.
- void [oqResetI](#) ([output_queue_t](#) *oqp)
Resets an output queue.
- [msg_t](#) [oqPutI](#) ([output_queue_t](#) *oqp, uint8_t b)
Output queue non-blocking write.
- [msg_t](#) [oqPutTimeout](#) ([output_queue_t](#) *oqp, uint8_t b, [sysinterval_t](#) timeout)

Output queue write with timeout.

- `msg_t oqGetl (output_queue_t *oqp)`

Output queue read.

- `size_t oqWritel (output_queue_t *oqp, const uint8_t *bp, size_t n)`

Output queue non-blocking write.

- `size_t oqWriteTimeout (output_queue_t *oqp, const uint8_t *bp, size_t n, sysinterval_t timeout)`

Output queue write with timeout.

9.133.1 Detailed Description

I/O Queues code.

9.134 hal_rtc.c File Reference

RTC Driver code.

```
#include "hal.h"
```

Functions

- void `rtcInit` (void)
RTC Driver initialization.
- void `rtcObjectInit` (RTCDriver *rtcp)
Initializes a generic RTC driver object.
- void `rtcSetTime` (RTCDriver *rtcp, const RTCDatetime *timespec)
Set current time.
- void `rtcGetTime` (RTCDriver *rtcp, RTCDatetime *timespec)
Get current time.
- void `rtcSetAlarm` (RTCDriver *rtcp, rtcalarm_t alarm, const RTCAalarm *alarmspec)
Set alarm time.
- void `rtcGetAlarm` (RTCDriver *rtcp, rtcalarm_t alarm, RTCAalarm *alarmspec)
Get current alarm.
- void `rtcSetCallback` (RTCDriver *rtcp, rtccb_t callback)
Enables or disables RTC callbacks.
- void `rtcConvertDateTimeToStructTm` (const RTCDatetime *timespec, struct tm *timp, uint32_t *tv_msec)
Convert RTCDatetime to broken-down time structure.
- void `rtcConvertStructTmToDateTime` (const struct tm *timp, uint32_t tv_msec, RTCDatetime *timespec)
Convert broken-down time structure to RTCDatetime.
- uint32_t `rtcConvertDateTimeToFAT` (const RTCDatetime *timespec)
Get current time in format suitable for usage in FAT file system.

Variables

- static const uint8_t `month_len` [12]
- static const uint16_t `accu_month_len` [12]

9.134.1 Detailed Description

RTC Driver code.

9.135 hal_sdc.c File Reference

SDC Driver code.

```
#include <string.h>
#include "hal.h"
```

Enumerations

- enum `mmc_switch_t` { `MMC_SWITCH_COMMAND_SET` = 0 , `MMC_SWITCH_SET_BITS` = 1 , `MMC_SWITCH_CLEAR_BITS` = 2 , `MMC_SWITCH_WRITE_BYTE` = 3 }
MMC switch mode.
- enum `sd_switch_t` { `SD_SWITCH_CHECK` = 0 , `SD_SWITCH_SET` = 1 }
SDC switch mode.
- enum `sd_switch_function_t` { `SD_SWITCH_FUNCTION_SPEED` = 0 , `SD_SWITCH_FUNCTION_CMD_SYSTEM` = 1 , `SD_SWITCH_FUNCTION_DRIVER_STRENGTH` = 2 , `SD_SWITCH_FUNCTION_CURRENT_LIMIT` = 3 }
SDC switch function.

Functions

- static bool `mode_detect` (`SDCDriver` *sdcp)
Detects card mode.
- static bool `mmc_init` (`SDCDriver` *sdcp)
Init procedure for MMC.
- static bool `sd_init` (`SDCDriver` *sdcp)
Init procedure for SDC.
- static uint32_t `mmc_cmd6_construct` (`mmc_switch_t` access, uint32_t idx, uint32_t value, uint32_t cmd_set)
Constructs CMD6 argument for MMC.
- static uint32_t `sd_cmd6_construct` (`sd_switch_t` mode, `sd_switch_function_t` function, uint32_t value)
Constructs CMD6 argument for SDC.
- static uint16_t `sd_cmd6_extract_info` (`sd_switch_function_t` function, const uint8_t *buf)
Extracts information from CMD6 answer.
- static bool `sd_cmd6_check_status` (`sd_switch_function_t` function, const uint8_t *buf)
Checks status after switching using CMD6.
- static bool `sd_detect_bus_clk` (`SDCDriver` *sdcp, `sdcbusclk_t` *clk)
Reads supported bus clock and switch SDC to appropriate mode.
- static bool `mmc_detect_bus_clk` (`SDCDriver` *sdcp, `sdcbusclk_t` *clk)
Reads supported bus clock and switch MMC to appropriate mode.
- static bool `detect_bus_clk` (`SDCDriver` *sdcp, `sdcbusclk_t` *clk)
Reads supported bus clock and switch card to appropriate mode.
- static bool `sd_set_bus_width` (`SDCDriver` *sdcp)
Sets bus width for SDC.
- static bool `mmc_set_bus_width` (`SDCDriver` *sdcp)

- Sets bus width for MMC.*
- static bool `_sd_wait_for_transfer_state_internal` (`SDCDriver *sdcp`, bool `crc_check`)
Wait for the card to complete pending operations.
- bool `_sd_wait_for_transfer_state` (`SDCDriver *sdcp`)
Wait for the card to complete pending operations with CRC check.
- bool `_sd_wait_for_transfer_state_nocrc` (`SDCDriver *sdcp`)
Wait for the card to complete pending operations without CRC check.
- void `sdclnit` (void)
SDC Driver initialization.
- void `sdObjectInit` (`SDCDriver *sdcp`)
Initializes the standard part of a `SDCDriver` structure.
- `msg_t` `sdStart` (`SDCDriver *sdcp`, const `SDCCConfig *config`)
Configures and activates the SDC peripheral.
- void `sdStop` (`SDCDriver *sdcp`)
Deactivates the SDC peripheral.
- bool `sdConnect` (`SDCDriver *sdcp`)
Performs the initialization procedure on the inserted card.
- bool `sdDisconnect` (`SDCDriver *sdcp`)
Brings the driver in a state safe for card removal.
- bool `sdRead` (`SDCDriver *sdcp`, `uint32_t startblk`, `uint8_t *buf`, `uint32_t n`)
Reads one or more blocks.
- bool `sdWrite` (`SDCDriver *sdcp`, `uint32_t startblk`, const `uint8_t *buf`, `uint32_t n`)
Writes one or more blocks.
- `sdflags_t` `sdGetAndClearErrors` (`SDCDriver *sdcp`)
Returns the errors mask associated to the previous operation.
- bool `sdSync` (`SDCDriver *sdcp`)
Waits for card idle condition.
- bool `sdGetInfo` (`SDCDriver *sdcp`, `BlockDeviceInfo *bdip`)
Returns the media info.
- bool `sdErase` (`SDCDriver *sdcp`, `uint32_t startblk`, `uint32_t endblk`)
Erases the supplied blocks.

Variables

- static const struct `SDCDriverVMT` `sd_vmt`
Virtual methods table.

9.135.1 Detailed Description

SDC Driver code.

9.136 hal_serial.c File Reference

Serial Driver code.

```
#include "hal.h"
```

Functions

- static size_t [_write](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [_read](#) (void *ip, uint8_t *bp, size_t n)
- static [msg_t _put](#) (void *ip, uint8_t b)
- static [msg_t _get](#) (void *ip)
- static [msg_t _putt](#) (void *ip, uint8_t b, [sysinterval_t](#) timeout)
- static [msg_t _gett](#) (void *ip, [sysinterval_t](#) timeout)
- static size_t [_writet](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static size_t [_readt](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static [msg_t _ctl](#) (void *ip, unsigned int operation, void *arg)
- void [sdInit](#) (void)
Serial Driver initialization.
- void [sdObjectInit](#) ([SerialDriver](#) *sdp, [qnotify_t](#) inotify, [qnotify_t](#) onotify)
Initializes a generic serial driver object.
- [msg_t sdStart](#) ([SerialDriver](#) *sdp, const [SerialConfig](#) *config)
Configures and starts the driver.
- void [sdStop](#) ([SerialDriver](#) *sdp)
Stops the driver.
- void [sdIncomingDataI](#) ([SerialDriver](#) *sdp, uint8_t b)
Handles incoming data.
- [msg_t sdRequestDataI](#) ([SerialDriver](#) *sdp)
Handles outgoing data.
- bool [sdPutWouldBlock](#) ([SerialDriver](#) *sdp)
Direct output check on a [SerialDriver](#).
- bool [sdGetWouldBlock](#) ([SerialDriver](#) *sdp)
Direct input check on a [SerialDriver](#).
- [msg_t sdControl](#) ([SerialDriver](#) *sdp, unsigned int operation, void *arg)
Control operation on a serial port.

Variables

- static const struct [SerialDriverVMT](#) vmt

9.136.1 Detailed Description

Serial Driver code.

9.137 hal_serial_usb.c File Reference

Serial over USB Driver code.

```
#include "hal.h"
```

Functions

- static bool [sdu_start_receive](#) ([SerialUSBDriver](#) *sdup)
- static size_t [_write](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [_read](#) (void *ip, uint8_t *bp, size_t n)
- static [msg_t](#) [_put](#) (void *ip, uint8_t b)
- static [msg_t](#) [_get](#) (void *ip)
- static [msg_t](#) [_putt](#) (void *ip, uint8_t b, [sysinterval_t](#) timeout)
- static [msg_t](#) [_gett](#) (void *ip, [sysinterval_t](#) timeout)
- static size_t [_writet](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static size_t [_readt](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static [msg_t](#) [_ctl](#) (void *ip, unsigned int operation, void *arg)
- static void [ibnotify](#) ([io_buffers_queue_t](#) *bqp)
 - Notification of empty buffer released into the input buffers queue.*
- static void [obnotify](#) ([io_buffers_queue_t](#) *bqp)
 - Notification of filled buffer inserted into the output buffers queue.*
- void [sduInit](#) (void)
 - Serial Driver initialization.*
- void [sduObjectInit](#) ([SerialUSBDriver](#) *sdup)
 - Initializes a generic full duplex driver object.*
- [msg_t](#) [sduStart](#) ([SerialUSBDriver](#) *sdup, const [SerialUSBConfig](#) *config)
 - Configures and starts the driver.*
- void [sduStop](#) ([SerialUSBDriver](#) *sdup)
 - Stops the driver.*
- void [sduSuspendHookI](#) ([SerialUSBDriver](#) *sdup)
 - USB device suspend handler.*
- void [sduWakeupHookI](#) ([SerialUSBDriver](#) *sdup)
 - USB device wakeup handler.*
- void [sduConfigureHookI](#) ([SerialUSBDriver](#) *sdup)
 - USB device configured handler.*
- bool [sduRequestsHook](#) ([USBDriver](#) *usbp)
 - Default requests hook.*
- void [sduSOFHookI](#) ([SerialUSBDriver](#) *sdup)
 - SOF handler.*
- void [sduDataTransmitted](#) ([USBDriver](#) *usbp, [usbep_t](#) ep)
 - Default data transmitted callback.*
- void [sduDataReceived](#) ([USBDriver](#) *usbp, [usbep_t](#) ep)
 - Default data received callback.*
- void [sduInterruptTransmitted](#) ([USBDriver](#) *usbp, [usbep_t](#) ep)
 - Default data received callback.*
- [msg_t](#) [sduControl](#) ([USBDriver](#) *usbp, unsigned int operation, void *arg)
 - Control operation on a serial USB port.*

Variables

- static [cdc_linecoding_t](#) linecoding
- static const struct [SerialUSBDriverVMT](#) vmt

9.137.1 Detailed Description

Serial over USB Driver code.

9.138 hal_sio.c File Reference

SIO Driver code.

```
#include "hal.h"
```

Functions

- static size_t [sync_write](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static size_t [sync_read](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static size_t [__write](#) (void *ip, const uint8_t *bp, size_t n)
- static size_t [__read](#) (void *ip, uint8_t *bp, size_t n)
- static msg_t [__put](#) (void *ip, uint8_t b)
- static msg_t [__get](#) (void *ip)
- static msg_t [__putt](#) (void *ip, uint8_t b, [sysinterval_t](#) timeout)
- static msg_t [__gett](#) (void *ip, [sysinterval_t](#) timeout)
- static size_t [__writet](#) (void *ip, const uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static size_t [__readt](#) (void *ip, uint8_t *bp, size_t n, [sysinterval_t](#) timeout)
- static msg_t [__ctl](#) (void *ip, unsigned int operation, void *arg)
- void [siolnit](#) (void)

SIO Driver initialization.
- void [sioObjectInit](#) (SIODriver *siop)

Initializes the standard part of a SIODriver structure.
- [msg_t](#) [sioStart](#) (SIODriver *siop, const SIOConfig *config)

Configures and activates the SIO peripheral.
- void [sioStop](#) (SIODriver *siop)

Deactivates the SIO peripheral.
- void [sioWriteEnableFlags](#) (SIODriver *siop, [sioevents_t](#) mask)

Writes the enabled events flags mask.
- void [sioSetEnableFlags](#) (SIODriver *siop, [sioevents_t](#) mask)

Sets flags into the enabled events flags mask.
- void [sioClearEnableFlags](#) (SIODriver *siop, [sioevents_t](#) mask)

Clears flags from the enabled events flags mask.
- [sioevents_t](#) [sioGetAndClearErrors](#) (SIODriver *siop)

Get and clears SIO error event flags.
- [sioevents_t](#) [sioGetAndClearEvents](#) (SIODriver *siop)

Get and clears SIO event flags.
- [sioevents_t](#) [sioGetEvents](#) (SIODriver *siop)

Returns the pending SIO event flags.
- size_t [sioAsyncRead](#) (SIODriver *siop, uint8_t *buffer, size_t n)

Reads data from the RX FIFO.
- size_t [sioAsyncWrite](#) (SIODriver *siop, const uint8_t *buffer, size_t n)

Writes data into the TX FIFO.
- [msg_t](#) [sioSynchronizeRX](#) (SIODriver *siop, [sysinterval_t](#) timeout)

Synchronizes with RX FIFO data availability.
- [msg_t](#) [sioSynchronizeRXIdle](#) (SIODriver *siop, [sysinterval_t](#) timeout)

Synchronizes with RX going idle.
- [msg_t](#) [sioSynchronizeTX](#) (SIODriver *siop, [sysinterval_t](#) timeout)

Synchronizes with TX FIFO space availability.
- [msg_t](#) [sioSynchronizeTXEnd](#) (SIODriver *siop, [sysinterval_t](#) timeout)

Synchronizes with TX completion.

Variables

- static const struct [sio_driver_vmt](#) vmt

9.138.1 Detailed Description

SIO Driver code.

9.139 hal_spi.c File Reference

SPI Driver selector code.

```
#include "hal.h"
#include "hal_spi_v2.inc"
#include "hal_spi_v1.inc"
```

9.139.1 Detailed Description

SPI Driver selector code.

9.140 hal_spi_v1.inc File Reference

SPI (v1) Driver code.

Functions

- void [spilnit](#) (void)
SPI Driver initialization.
- void [spiObjectInit](#) ([SPIDriver](#) *spip)
Initializes the standard part of a [SPIDriver](#) structure.
- [msg_t](#) [spiStart](#) ([SPIDriver](#) *spip, const [SPIConfig](#) *config)
Configures and activates the SPI peripheral.
- void [spiStop](#) ([SPIDriver](#) *spip)
Deactivates the SPI peripheral.
- void [spiSelect](#) ([SPIDriver](#) *spip)
Asserts the slave select signal and prepares for transfers.
- void [spiUnselect](#) ([SPIDriver](#) *spip)
Deasserts the slave select signal.
- void [spiStartIgnore](#) ([SPIDriver](#) *spip, [size_t](#) n)
Ignores data on the SPI bus.
- void [spiStartExchange](#) ([SPIDriver](#) *spip, [size_t](#) n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- void [spiStartSend](#) ([SPIDriver](#) *spip, [size_t](#) n, const void *txbuf)
Sends data over the SPI bus.

- void [spiStartReceive](#) ([SPIDriver](#) *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- void [spiAbortl](#) ([SPIDriver](#) *spip)
Aborts the ongoing SPI operation.
- void [spiAbort](#) ([SPIDriver](#) *spip)
Aborts the ongoing SPI operation, if any.
- void [spiIgnore](#) ([SPIDriver](#) *spip, size_t n)
Ignores data on the SPI bus.
- void [spiExchange](#) ([SPIDriver](#) *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- void [spiSend](#) ([SPIDriver](#) *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- void [spiReceive](#) ([SPIDriver](#) *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- void [spiAcquireBus](#) ([SPIDriver](#) *spip)
Gains exclusive access to the SPI bus.
- void [spiReleaseBus](#) ([SPIDriver](#) *spip)
Releases exclusive access to the SPI bus.

9.140.1 Detailed Description

SPI (v1) Driver code.

9.141 hal_spi_v2.inc File Reference

SPI (v2) Driver code.

Functions

- void [spilnit](#) (void)
SPI Driver initialization.
- void [spiObjectInit](#) ([SPIDriver](#) *spip)
Initializes the standard part of a [SPIDriver](#) structure.
- [msg_t](#) [spiStart](#) ([SPIDriver](#) *spip, const [SPIConfig](#) *config)
Configures and activates the SPI peripheral.
- void [spiStop](#) ([SPIDriver](#) *spip)
Deactivates the SPI peripheral.
- void [spiSelect](#) ([SPIDriver](#) *spip)
Asserts the slave select signal and prepares for transfers.
- void [spiUnselect](#) ([SPIDriver](#) *spip)
Deasserts the slave select signal.
- [msg_t](#) [spiStartIgnorel](#) ([SPIDriver](#) *spip, size_t n)
Ignores data on the SPI bus.
- [msg_t](#) [spiStartIgnore](#) ([SPIDriver](#) *spip, size_t n)
Ignores data on the SPI bus.
- [msg_t](#) [spiStartExchange](#) ([SPIDriver](#) *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.

- [msg_t spiStartExchange](#) ([SPIDriver](#) *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- [msg_t spiStartSendI](#) ([SPIDriver](#) *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- [msg_t spiStartSend](#) ([SPIDriver](#) *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- [msg_t spiStartReceiveI](#) ([SPIDriver](#) *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- [msg_t spiStartReceive](#) ([SPIDriver](#) *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- [msg_t spiStopTransferI](#) ([SPIDriver](#) *spip, size_t *sizep)
Stops the ongoing SPI operation.
- [msg_t spiStopTransfer](#) ([SPIDriver](#) *spip, size_t *sizep)
Stops the ongoing SPI operation, if any.
- [msg_t spiSynchronizeS](#) ([SPIDriver](#) *spip, [sysinterval_t](#) timeout)
Synchronizes with current transfer completion.
- [msg_t spiSynchronize](#) ([SPIDriver](#) *spip, [sysinterval_t](#) timeout)
Synchronizes with current transfer completion.
- [msg_t spiIgnore](#) ([SPIDriver](#) *spip, size_t n)
Ignores data on the SPI bus.
- [msg_t spiExchange](#) ([SPIDriver](#) *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- [msg_t spiSend](#) ([SPIDriver](#) *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- [msg_t spiReceive](#) ([SPIDriver](#) *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- void [spiAcquireBus](#) ([SPIDriver](#) *spip)
Gains exclusive access to the SPI bus.
- void [spiReleaseBus](#) ([SPIDriver](#) *spip)
Releases exclusive access to the SPI bus.

9.141.1 Detailed Description

SPI (v2) Driver code.

9.142 hal_st.c File Reference

ST Driver code.

```
#include "hal.h"
```

Functions

- void [stInit](#) (void)
ST Driver initialization.
- void [stBind](#) (void)
Enables an alarm interrupt on the invoking core.
- [systime_t stGetCounter](#) (void)
Returns the time counter value.
- void [stStartAlarm](#) ([systime_t](#) abstime)
Starts the alarm zero.
- void [stStopAlarm](#) (void)
Stops the alarm zero interrupt.
- void [stSetAlarm](#) ([systime_t](#) abstime)
Sets the alarm zero time.
- [systime_t stGetAlarm](#) (void)
Returns the alarm zero current time.
- bool [stIsAlarmActive](#) (void)
Determines if the alarm zero is active.
- void [stSetCallback](#) (unsigned alarm, [st_callback_t](#) cb)
Associates a callback to an alarm.
- void [stBindAlarmN](#) (unsigned alarm)
Enables an alarm interrupt on the invoking core.
- void [stStartAlarmN](#) (unsigned alarm, [systime_t](#) abstime)
Starts an additional alarm.
- void [stStopAlarmN](#) (unsigned alarm)
Stops an additional alarm.
- void [stSetAlarmN](#) (unsigned alarm, [systime_t](#) abstime)
Sets an additional alarm time.
- [systime_t stGetAlarmN](#) (unsigned alarm)
Returns an additional alarm current time.
- bool [stIsAlarmActiveN](#) (unsigned alarm)
Determines if the specified alarm is active.

Variables

- [st_callback_t st_callbacks](#) [[ST_LLD_NUM_ALARMS](#)]
Callback pointers for each alarm.

9.142.1 Detailed Description

ST Driver code.

9.143 hal_trng.c File Reference

TRNG Driver code.

```
#include "hal.h"
```

Functions

- void [trngInit](#) (void)
TRNG Driver initialization.
- void [trngObjectInit](#) (TRNGDriver *trngp)
Initializes the standard part of a TRNGDriver structure.
- [msg_t trngStart](#) (TRNGDriver *trngp, const TRNGConfig *config)
Configures and activates the TRNG peripheral.
- void [trngStop](#) (TRNGDriver *trngp)
Deactivates the TRNG peripheral.
- bool [trngGenerate](#) (TRNGDriver *trngp, size_t size, uint8_t *out)
True random numbers generator.

9.143.1 Detailed Description

TRNG Driver code.

9.144 hal_uart.c File Reference

UART Driver code.

```
#include "hal.h"
```

Functions

- void [uartInit](#) (void)
UART Driver initialization.
- void [uartObjectInit](#) (UARTDriver *uartp)
Initializes the standard part of a UARTDriver structure.
- [msg_t uartStart](#) (UARTDriver *uartp, const UARTConfig *config)
Configures and activates the UART peripheral.
- void [uartStop](#) (UARTDriver *uartp)
Deactivates the UART peripheral.
- void [uartStartSend](#) (UARTDriver *uartp, size_t n, const void *txbuf)
Starts a transmission on the UART peripheral.
- void [uartStartSendI](#) (UARTDriver *uartp, size_t n, const void *txbuf)
Starts a transmission on the UART peripheral.
- size_t [uartStopSend](#) (UARTDriver *uartp)
Stops any ongoing transmission.
- size_t [uartStopSendI](#) (UARTDriver *uartp)
Stops any ongoing transmission.
- void [uartStartReceive](#) (UARTDriver *uartp, size_t n, void *rxbuf)
Starts a receive operation on the UART peripheral.
- void [uartStartReceiveI](#) (UARTDriver *uartp, size_t n, void *rxbuf)
Starts a receive operation on the UART peripheral.
- size_t [uartStopReceive](#) (UARTDriver *uartp)
Stops any ongoing receive operation.

- `size_t uartStopReceive (UARTDriver *uartp)`
Stops any ongoing receive operation.
- `msg_t uartSendTimeout (UARTDriver *uartp, size_t *np, const void *txbuf, sysinterval_t timeout)`
Performs a transmission on the UART peripheral.
- `msg_t uartSendFullTimeout (UARTDriver *uartp, size_t *np, const void *txbuf, sysinterval_t timeout)`
Performs a transmission on the UART peripheral.
- `msg_t uartReceiveTimeout (UARTDriver *uartp, size_t *np, void *rxbuf, sysinterval_t timeout)`
Performs a receive operation on the UART peripheral.
- `void uartAcquireBus (UARTDriver *uartp)`
Gains exclusive access to the UART bus.
- `void uartReleaseBus (UARTDriver *uartp)`
Releases exclusive access to the UART bus.

9.144.1 Detailed Description

UART Driver code.

9.145 hal_usb.c File Reference

USB Driver code.

```
#include <string.h>
#include "hal.h"
```

Functions

- `static uint16_t get_hword (uint8_t *p)`
- `static void set_address (USBDriver *usbp)`
SET ADDRESS transaction callback.
- `static bool default_handler (USBDriver *usbp)`
Standard requests handler.
- `void usblnit (void)`
USB Driver initialization.
- `void usbObjectInit (USBDriver *usbp)`
Initializes the standard part of a USBDriver structure.
- `msg_t usbStart (USBDriver *usbp, const USBConfig *config)`
Configures and activates the USB peripheral.
- `void usbStop (USBDriver *usbp)`
Deactivates the USB peripheral.
- `void usblnitEndpointI (USBDriver *usbp, usbep_t ep, const USBEndpointConfig *epcp)`
Enables an endpoint.
- `void usbDisableEndpointsI (USBDriver *usbp)`
Disables all the active endpoints.
- `void usbStartReceiveI (USBDriver *usbp, usbep_t ep, uint8_t *buf, size_t n)`
Starts a receive transaction on an OUT endpoint.
- `void usbStartTransmitI (USBDriver *usbp, usbep_t ep, const uint8_t *buf, size_t n)`
Starts a transmit transaction on an IN endpoint.

- `msg_t usbReceive (USBDriver *usbp, usbep_t ep, uint8_t *buf, size_t n)`
Performs a receive transaction on an OUT endpoint.
- `msg_t usbTransmit (USBDriver *usbp, usbep_t ep, const uint8_t *buf, size_t n)`
Performs a transmit transaction on an IN endpoint.
- `bool usbStallReceiveI (USBDriver *usbp, usbep_t ep)`
Stalls an OUT endpoint.
- `bool usbStallTransmitI (USBDriver *usbp, usbep_t ep)`
Stalls an IN endpoint.
- `void usbWakeupHost (USBDriver *usbp)`
Host wake-up procedure.
- `void _usb_reset (USBDriver *usbp)`
USB reset routine.
- `void _usb_suspend (USBDriver *usbp)`
USB suspend routine.
- `void _usb_wakeup (USBDriver *usbp)`
USB wake-up routine.
- `void _usb_ep0setup (USBDriver *usbp, usbep_t ep)`
Default EP0 SETUP callback.
- `void _usb_ep0in (USBDriver *usbp, usbep_t ep)`
Default EP0 IN callback.
- `void _usb_ep0out (USBDriver *usbp, usbep_t ep)`
Default EP0 OUT callback.

Variables

- `static const uint8_t zero_status [] = {0x00, 0x00}`
- `static const uint8_t active_status [] = {0x00, 0x00}`
- `static const uint8_t halted_status [] = {0x01, 0x00}`

9.145.1 Detailed Description

USB Driver code.

9.146 hal_wdg.c File Reference

WDG Driver code.

```
#include "hal.h"
```

Functions

- `void wdgInit (void)`
WDG Driver initialization.
- `msg_t wdgStart (WDGDriver *wdgp, const WDGConfig *config)`
Configures and activates the WDG peripheral.
- `void wdgStop (WDGDriver *wdgp)`
Deactivates the WDG peripheral.
- `void wdgReset (WDGDriver *wdgp)`
Resets WDG's counter.

9.146.1 Detailed Description

WDG Driver code.

9.147 hal_wspi.c File Reference

WSPI Driver code.

```
#include "hal.h"
```

Functions

- void [wsplInit](#) (void)
WSPI Driver initialization.
- void [wspiObjectInit](#) (WSPIDriver *wspip)
Initializes the standard part of a [WSPIDriver](#) structure.
- [msg_t](#) [wspiStart](#) (WSPIDriver *wspip, const [WSPIConfig](#) *config)
Configures and activates the WSPI peripheral.
- void [wspiStop](#) (WSPIDriver *wspip)
Deactivates the WSPI peripheral.
- void [wspiStartCommand](#) (WSPIDriver *wspip, const [wspi_command_t](#) *cmdp)
Sends a command without data phase.
- void [wspiStartSend](#) (WSPIDriver *wspip, const [wspi_command_t](#) *cmdp, size_t n, const uint8_t *txbuf)
Sends a command with data over the WSPI bus.
- void [wspiStartReceive](#) (WSPIDriver *wspip, const [wspi_command_t](#) *cmdp, size_t n, uint8_t *rxbuf)
Sends a command then receives data over the WSPI bus.
- bool [wspiCommand](#) (WSPIDriver *wspip, const [wspi_command_t](#) *cmdp)
Sends a command without data phase.
- bool [wspiSend](#) (WSPIDriver *wspip, const [wspi_command_t](#) *cmdp, size_t n, const uint8_t *txbuf)
Sends a command with data over the WSPI bus.
- bool [wspiReceive](#) (WSPIDriver *wspip, const [wspi_command_t](#) *cmdp, size_t n, uint8_t *rxbuf)
Sends a command then receives data over the WSPI bus.
- void [wspiMapFlash](#) (WSPIDriver *wspip, const [wspi_command_t](#) *cmdp, uint8_t **addrp)
Maps in memory space a WSPI flash device.
- void [wspiUnmapFlash](#) (WSPIDriver *wspip)
Unmaps from memory space a WSPI flash device.
- void [wspiAcquireBus](#) (WSPIDriver *wspip)
Gains exclusive access to the WSPI bus.
- void [wspiReleaseBus](#) (WSPIDriver *wspip)
Releases exclusive access to the WSPI bus.

9.147.1 Detailed Description

WSPI Driver code.

9.148 hal_adc_ild.c File Reference

PLATFORM ADC subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [adc_ild_init](#) (void)
Low level ADC driver initialization.
- void [adc_ild_start](#) (ADCDriver *adcp)
Configures and activates the ADC peripheral.
- void [adc_ild_stop](#) (ADCDriver *adcp)
Deactivates the ADC peripheral.
- void [adc_ild_start_conversion](#) (ADCDriver *adcp)
Starts an ADC conversion.
- void [adc_ild_stop_conversion](#) (ADCDriver *adcp)
Stops an ongoing conversion.

Variables

- [ADCDriver ADCD1](#)
ADC1 driver identifier.

9.148.1 Detailed Description

PLATFORM ADC subsystem low level driver source.

9.149 hal_adc_ild.h File Reference

PLATFORM ADC subsystem low level driver header.

Macros

- #define [adc_ild_driver_fields](#)
Low level fields of the ADC driver structure.
- #define [adc_ild_config_fields](#)
Low level fields of the ADC configuration structure.
- #define [adc_ild_configuration_group_fields](#)
Low level fields of the ADC configuration structure.

Possible ADC errors mask bits.

- #define [ADC_ERR_DMAFAILURE](#) 1U
- #define [ADC_ERR_OVERFLOW](#) 2U
- #define [ADC_ERR_AWD](#) 4U

PLATFORM configuration options

- #define [PLATFORM_ADC_USE_ADC1](#) FALSE
ADC1 driver enable switch.

Typedefs

- typedef uint16_t [adcsample_t](#)
ADC sample data type.
- typedef uint16_t [adc_channels_num_t](#)
Channels number in a conversion group.
- typedef uint32_t [adcerror_t](#)
Type of an ADC error mask.

Functions

- void [adc_lld_init](#) (void)
Low level ADC driver initialization.
- void [adc_lld_start](#) (ADCDriver *adcp)
Configures and activates the ADC peripheral.
- void [adc_lld_stop](#) (ADCDriver *adcp)
Deactivates the ADC peripheral.
- void [adc_lld_start_conversion](#) (ADCDriver *adcp)
Starts an ADC conversion.
- void [adc_lld_stop_conversion](#) (ADCDriver *adcp)
Stops an ongoing conversion.

9.149.1 Detailed Description

PLATFORM ADC subsystem low level driver header.

9.150 hal_can_lld.c File Reference

PLATFORM CAN subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [can_lld_init](#) (void)
Low level CAN driver initialization.
- void [can_lld_start](#) (CANDriver *canp)
Configures and activates the CAN peripheral.
- void [can_lld_stop](#) (CANDriver *canp)
Deactivates the CAN peripheral.
- bool [can_lld_is_tx_empty](#) (CANDriver *canp, canmbx_t mailbox)
Determines whether a frame can be transmitted.
- void [can_lld_transmit](#) (CANDriver *canp, canmbx_t mailbox, const CANTxFrame *ctfp)
Inserts a frame into the transmit queue.
- bool [can_lld_is_rx_nonempty](#) (CANDriver *canp, canmbx_t mailbox)
Determines whether a frame has been received.

- void [can_ild_receive](#) ([CANDriver](#) *canp, [canmbx_t](#) mailbox, [CANRxFrame](#) *crfp)
Receives a frame from the input queue.
- void [can_ild_abort](#) ([CANDriver](#) *canp, [canmbx_t](#) mailbox)
Tries to abort an ongoing transmission.
- void [can_ild_sleep](#) ([CANDriver](#) *canp)
Enters the sleep mode.
- void [can_ild_wakeup](#) ([CANDriver](#) *canp)
Enforces leaving the sleep mode.

Variables

- [CANDriver](#) [CAND1](#)
CAN1 driver identifier.

9.150.1 Detailed Description

PLATFORM CAN subsystem low level driver source.

9.151 hal_can_ild.h File Reference

PLATFORM CAN subsystem low level driver header.

Data Structures

- struct [CANTxFrame](#)
CAN transmission frame.
- struct [CANRxFrame](#)
CAN received frame.
- struct [hal_can_config](#)
Type of a CAN configuration structure.
- struct [hal_can_driver](#)
Structure representing an CAN driver.

Macros

- #define [CAN_TX_MAILBOXES](#) 1
Number of transmit mailboxes.
- #define [CAN_RX_MAILBOXES](#) 1
Number of receive mailboxes.

PLATFORM configuration options

- #define [PLATFORM_CAN_USE_CAN1](#) FALSE
CAN1 driver enable switch.

Typedefs

- typedef struct [hal_can_driver](#) CANDriver
Type of a structure representing an CAN driver.
- typedef uint32_t [canmbx_t](#)
Type of a transmission mailbox index.
- typedef void(* [can_callback_t](#)) (CANDriver *canp, uint32_t flags)
Type of a CAN notification callback.
- typedef struct [hal_can_config](#) CANConfig
Type of a CAN configuration structure.

Functions

- void [can_ild_init](#) (void)
Low level CAN driver initialization.
- void [can_ild_start](#) (CANDriver *canp)
Configures and activates the CAN peripheral.
- void [can_ild_stop](#) (CANDriver *canp)
Deactivates the CAN peripheral.
- bool [can_ild_is_tx_empty](#) (CANDriver *canp, [canmbx_t](#) mailbox)
Determines whether a frame can be transmitted.
- void [can_ild_transmit](#) (CANDriver *canp, [canmbx_t](#) mailbox, const [CANTxFrame](#) *ctfp)
Inserts a frame into the transmit queue.
- bool [can_ild_is_rx_nonempty](#) (CANDriver *canp, [canmbx_t](#) mailbox)
Determines whether a frame has been received.
- void [can_ild_receive](#) (CANDriver *canp, [canmbx_t](#) mailbox, [CANRxFrame](#) *crfp)
Receives a frame from the input queue.
- void [can_ild_abort](#) (CANDriver *canp, [canmbx_t](#) mailbox)
Tries to abort an ongoing transmission.
- void [can_ild_sleep](#) (CANDriver *canp)
Enters the sleep mode.
- void [can_ild_wakeup](#) (CANDriver *canp)
Enforces leaving the sleep mode.

9.151.1 Detailed Description

PLATFORM CAN subsystem low level driver header.

9.152 hal_crypto_ild.c File Reference

PLATFORM cryptographic subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [cry_ild_init](#) (void)
Low level crypto driver initialization.
- void [cry_ild_start](#) (CRYDriver *cryp)
Configures and activates the crypto peripheral.
- void [cry_ild_stop](#) (CRYDriver *cryp)
Deactivates the crypto peripheral.
- [cryerror_t cry_ild_aes_loadkey](#) (CRYDriver *cryp, size_t size, const uint8_t *keyp)
Initializes the AES transient key.
- [cryerror_t cry_ild_encrypt_AES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Encryption of a single block using AES.
- [cryerror_t cry_ild_decrypt_AES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Decryption of a single block using AES.
- [cryerror_t cry_ild_encrypt_AES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Encryption operation using AES-ECB.
- [cryerror_t cry_ild_decrypt_AES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Decryption operation using AES-ECB.
- [cryerror_t cry_ild_encrypt_AES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CBC.
- [cryerror_t cry_ild_decrypt_AES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CBC.
- [cryerror_t cry_ild_encrypt_AES_CFB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CFB.
- [cryerror_t cry_ild_decrypt_AES_CFB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CFB.
- [cryerror_t cry_ild_encrypt_AES_CTR](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CTR.
- [cryerror_t cry_ild_decrypt_AES_CTR](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CTR.
- [cryerror_t cry_ild_encrypt_AES_GCM](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, uint8_t *tag_out)
Encryption operation using AES-GCM.
- [cryerror_t cry_ild_decrypt_AES_GCM](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, const uint8_t *tag_in)
Decryption operation using AES-GCM.
- [cryerror_t cry_ild_des_loadkey](#) (CRYDriver *cryp, size_t size, const uint8_t *keyp)
Initializes the DES transient key.
- [cryerror_t cry_ild_encrypt_DES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Encryption of a single block using (T)DES.
- [cryerror_t cry_ild_decrypt_DES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Decryption of a single block using (T)DES.

- [cryerror_t cry_lld_encrypt_DES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Encryption operation using (T)DES-ECB.
- [cryerror_t cry_lld_decrypt_DES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Decryption operation using (T)DES-ECB.
- [cryerror_t cry_lld_encrypt_DES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using (T)DES-CBC.
- [cryerror_t cry_lld_decrypt_DES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using (T)DES-CBC.
- [cryerror_t cry_lld_SHA1_init](#) (CRYDriver *cryp, [SHA1Context](#) *sha1ctxp)
Hash initialization using SHA1.
- [cryerror_t cry_lld_SHA1_update](#) (CRYDriver *cryp, [SHA1Context](#) *sha1ctxp, size_t size, const uint8_t *in)
Hash update using SHA1.
- [cryerror_t cry_lld_SHA1_final](#) (CRYDriver *cryp, [SHA1Context](#) *sha1ctxp, uint8_t *out)
Hash finalization using SHA1.
- [cryerror_t cry_lld_SHA256_init](#) (CRYDriver *cryp, [SHA256Context](#) *sha256ctxp)
Hash initialization using SHA256.
- [cryerror_t cry_lld_SHA256_update](#) (CRYDriver *cryp, [SHA256Context](#) *sha256ctxp, size_t size, const uint8_t *in)
Hash update using SHA256.
- [cryerror_t cry_lld_SHA256_final](#) (CRYDriver *cryp, [SHA256Context](#) *sha256ctxp, uint8_t *out)
Hash finalization using SHA256.
- [cryerror_t cry_lld_SHA512_init](#) (CRYDriver *cryp, [SHA512Context](#) *sha512ctxp)
Hash initialization using SHA512.
- [cryerror_t cry_lld_SHA512_update](#) (CRYDriver *cryp, [SHA512Context](#) *sha512ctxp, size_t size, const uint8_t *in)
Hash update using SHA512.
- [cryerror_t cry_lld_SHA512_final](#) (CRYDriver *cryp, [SHA512Context](#) *sha512ctxp, uint8_t *out)
Hash finalization using SHA512.
- [cryerror_t cry_lld_hmac_loadkey](#) (CRYDriver *cryp, size_t size, const uint8_t *keyp)
Initializes the HMAC transient key.
- [cryerror_t cry_lld_HMACSHA256_init](#) (CRYDriver *cryp, [HMACSHA256Context](#) *hmacsha256ctxp)
Hash initialization using HMAC_SHA256.
- [cryerror_t cry_lld_HMACSHA256_update](#) (CRYDriver *cryp, [HMACSHA256Context](#) *hmacsha256ctxp, size_t size, const uint8_t *in)
Hash update using HMAC.
- [cryerror_t cry_lld_HMACSHA256_final](#) (CRYDriver *cryp, [HMACSHA256Context](#) *hmacsha256ctxp, uint8_t *out)
Hash finalization using HMAC.
- [cryerror_t cry_lld_HMACSHA512_init](#) (CRYDriver *cryp, [HMACSHA512Context](#) *hmacsha512ctxp)
Hash initialization using HMAC_SHA512.
- [cryerror_t cry_lld_HMACSHA512_update](#) (CRYDriver *cryp, [HMACSHA512Context](#) *hmacsha512ctxp, size_t size, const uint8_t *in)
Hash update using HMAC.
- [cryerror_t cry_lld_HMACSHA512_final](#) (CRYDriver *cryp, [HMACSHA512Context](#) *hmacsha512ctxp, uint8_t *out)
Hash finalization using HMAC.

Variables

- [CRYDriver](#) CRYD1
CRY1 driver identifier.

9.152.1 Detailed Description

PLATFORM cryptographic subsystem low level driver source.

9.153 hal_crypto_lld.h File Reference

PLATFORM cryptographic subsystem low level driver header.

Data Structures

- struct [CRYConfig](#)
Driver configuration structure.
- struct [CRYDriver](#)
Structure representing an CRY driver.
- struct [SHA1Context](#)
Type of a SHA1 context.
- struct [SHA256Context](#)
Type of a SHA256 context.
- struct [SHA512Context](#)
Type of a SHA512 context.
- struct [HMACSHA256Context](#)
Type of a HMAC_SHA256 context.
- struct [HMACSHA512Context](#)
Type of a HMAC_SHA512 context.

Macros**Driver capability switches**

- `#define CRY_LLD_SUPPORTS_AES TRUE`
- `#define CRY_LLD_SUPPORTS_AES_ECB TRUE`
- `#define CRY_LLD_SUPPORTS_AES_CBC TRUE`
- `#define CRY_LLD_SUPPORTS_AES_CFB TRUE`
- `#define CRY_LLD_SUPPORTS_AES_CTR TRUE`
- `#define CRY_LLD_SUPPORTS_AES_GCM TRUE`
- `#define CRY_LLD_SUPPORTS_DES TRUE`
- `#define CRY_LLD_SUPPORTS_DES_ECB TRUE`
- `#define CRY_LLD_SUPPORTS_DES_CBC TRUE`
- `#define CRY_LLD_SUPPORTS_SHA1 TRUE`
- `#define CRY_LLD_SUPPORTS_SHA256 TRUE`
- `#define CRY_LLD_SUPPORTS_SHA512 TRUE`
- `#define CRY_LLD_SUPPORTS_HMAC_SHA256 TRUE`
- `#define CRY_LLD_SUPPORTS_HMAC_SHA512 TRUE`

PLATFORM configuration options

- `#define PLATFORM_CRY_USE_CRY1 FALSE`
CRY1 driver enable switch.

Typedefs

- typedef uint32_t [crykey_t](#)
CRY key identifier type.
- typedef struct CRYDriver [CRYDriver](#)
Type of a structure representing an CRY driver.

Functions

- void [cry_lld_init](#) (void)
Low level crypto driver initialization.
- void [cry_lld_start](#) ([CRYDriver](#) *cryp)
Configures and activates the crypto peripheral.
- void [cry_lld_stop](#) ([CRYDriver](#) *cryp)
Deactivates the crypto peripheral.
- [cryerror_t](#) [cry_lld_aes_loadkey](#) ([CRYDriver](#) *cryp, size_t size, const uint8_t *keyp)
Initializes the AES transient key.
- [cryerror_t](#) [cry_lld_encrypt_AES](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Encryption of a single block using AES.
- [cryerror_t](#) [cry_lld_decrypt_AES](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)
Decryption of a single block using AES.
- [cryerror_t](#) [cry_lld_encrypt_AES_ECB](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Encryption operation using AES-ECB.
- [cryerror_t](#) [cry_lld_decrypt_AES_ECB](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)
Decryption operation using AES-ECB.
- [cryerror_t](#) [cry_lld_encrypt_AES_CBC](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CBC.
- [cryerror_t](#) [cry_lld_decrypt_AES_CBC](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CBC.
- [cryerror_t](#) [cry_lld_encrypt_AES_CFB](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CFB.
- [cryerror_t](#) [cry_lld_decrypt_AES_CFB](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CFB.
- [cryerror_t](#) [cry_lld_encrypt_AES_CTR](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Encryption operation using AES-CTR.
- [cryerror_t](#) [cry_lld_decrypt_AES_CTR](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)
Decryption operation using AES-CTR.
- [cryerror_t](#) [cry_lld_encrypt_AES_GCM](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, uint8_t *tag_out)
Encryption operation using AES-GCM.
- [cryerror_t](#) [cry_lld_decrypt_AES_GCM](#) ([CRYDriver](#) *cryp, [crykey_t](#) key_id, size_t auth_size, const uint8_t *auth_in, size_t text_size, const uint8_t *text_in, uint8_t *text_out, const uint8_t *iv, size_t tag_size, const uint8_t *tag_in)

Decryption operation using AES-GCM.

- [cryerror_t cry_lld_des_loadkey](#) (CRYDriver *cryp, size_t size, const uint8_t *keyp)

Initializes the DES transient key.

- [cryerror_t cry_lld_encrypt_DES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)

Encryption of a single block using (T)DES.

- [cryerror_t cry_lld_decrypt_DES](#) (CRYDriver *cryp, [crykey_t](#) key_id, const uint8_t *in, uint8_t *out)

Decryption of a single block using (T)DES.

- [cryerror_t cry_lld_encrypt_DES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)

Encryption operation using (T)DES-ECB.

- [cryerror_t cry_lld_decrypt_DES_ECB](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out)

Decryption operation using (T)DES-ECB.

- [cryerror_t cry_lld_encrypt_DES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)

Encryption operation using (T)DES-CBC.

- [cryerror_t cry_lld_decrypt_DES_CBC](#) (CRYDriver *cryp, [crykey_t](#) key_id, size_t size, const uint8_t *in, uint8_t *out, const uint8_t *iv)

Decryption operation using (T)DES-CBC.

- [cryerror_t cry_lld_SHA1_init](#) (CRYDriver *cryp, [SHA1Context](#) *sha1ctxp)

Hash initialization using SHA1.

- [cryerror_t cry_lld_SHA1_update](#) (CRYDriver *cryp, [SHA1Context](#) *sha1ctxp, size_t size, const uint8_t *in)

Hash update using SHA1.

- [cryerror_t cry_lld_SHA1_final](#) (CRYDriver *cryp, [SHA1Context](#) *sha1ctxp, uint8_t *out)

Hash finalization using SHA1.

- [cryerror_t cry_lld_SHA256_init](#) (CRYDriver *cryp, [SHA256Context](#) *sha256ctxp)

Hash initialization using SHA256.

- [cryerror_t cry_lld_SHA256_update](#) (CRYDriver *cryp, [SHA256Context](#) *sha256ctxp, size_t size, const uint8_t *in)

Hash update using SHA256.

- [cryerror_t cry_lld_SHA256_final](#) (CRYDriver *cryp, [SHA256Context](#) *sha256ctxp, uint8_t *out)

Hash finalization using SHA256.

- [cryerror_t cry_lld_SHA512_init](#) (CRYDriver *cryp, [SHA512Context](#) *sha512ctxp)

Hash initialization using SHA512.

- [cryerror_t cry_lld_SHA512_update](#) (CRYDriver *cryp, [SHA512Context](#) *sha512ctxp, size_t size, const uint8_t *in)

Hash update using SHA512.

- [cryerror_t cry_lld_SHA512_final](#) (CRYDriver *cryp, [SHA512Context](#) *sha512ctxp, uint8_t *out)

Hash finalization using SHA512.

- [cryerror_t cry_lld_hmac_loadkey](#) (CRYDriver *cryp, size_t size, const uint8_t *keyp)

Initializes the HMAC transient key.

- [cryerror_t cry_lld_HMACSHA256_init](#) (CRYDriver *cryp, [HMACSHA256Context](#) *hmacsha256ctxp)

Hash initialization using HMAC_SHA256.

- [cryerror_t cry_lld_HMACSHA256_update](#) (CRYDriver *cryp, [HMACSHA256Context](#) *hmacsha256ctxp, size_t size, const uint8_t *in)

Hash update using HMAC.

- [cryerror_t cry_lld_HMACSHA256_final](#) (CRYDriver *cryp, [HMACSHA256Context](#) *hmacsha256ctxp, uint8_t *out)

Hash finalization using HMAC.

- [cryerror_t cry_lld_HMACSHA512_init](#) (CRYDriver *cryp, [HMACSHA512Context](#) *hmacsha512ctxp)

Hash initialization using HMAC_SHA512.

- `cryerror_t cry_llid_HMACSHA512_update` (`CRYDriver *cryp`, `HMACSHA512Context *hmacsha512ctxp`, `size_t size`, `const uint8_t *in`)
Hash update using HMAC.
- `cryerror_t cry_llid_HMACSHA512_final` (`CRYDriver *cryp`, `HMACSHA512Context *hmacsha512ctxp`, `uint8_t *out`)
Hash finalization using HMAC.

9.153.1 Detailed Description

PLATFORM cryptographic subsystem low level driver header.

9.154 hal_dac_llid.c File Reference

PLATFORM DAC subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void `dac_llid_init` (void)
Low level DAC driver initialization.
- void `dac_llid_start` (`DACDriver *dacp`)
Configures and activates the DAC peripheral.
- void `dac_llid_stop` (`DACDriver *dacp`)
Deactivates the DAC peripheral.
- void `dac_llid_put_channel` (`DACDriver *dacp`, `dacchannel_t channel`, `dacsample_t sample`)
Outputs a value directly on a DAC channel.
- void `dac_llid_start_conversion` (`DACDriver *dacp`)
Starts a DAC conversion.
- void `dac_llid_stop_conversion` (`DACDriver *dacp`)
Stops an ongoing conversion.

Variables

- `DACDriver DACD1`
DAC1 driver identifier.

9.154.1 Detailed Description

PLATFORM DAC subsystem low level driver source.

9.155 hal_dac_llid.h File Reference

PLATFORM DAC subsystem low level driver header.

Macros

- #define `DAC_MAX_CHANNELS` 2
Maximum number of DAC channels per unit.
- #define `dac_ild_driver_fields`
Low level fields of the DAC driver structure.
- #define `dac_ild_config_fields`
Low level fields of the DAC configuration structure.
- #define `dac_ild_conversion_group_fields`
Low level fields of the DAC group configuration structure.

Configuration options

- #define `PLATFORM_DAC_USE_DAC1` FALSE
DAC1 CH1 driver enable switch.

Typedefs

- typedef uint32_t `dacchannel_t`
Type of a DAC channel index.
- typedef uint16_t `dacsample_t`
Type representing a DAC sample.

Enumerations

- enum `dacerror_t` { `DAC_ERR_DMAFAILURE` = 0 , `DAC_ERR_UNDERFLOW` = 1 }
Possible DAC failure causes.

Functions

- void `dac_ild_init` (void)
Low level DAC driver initialization.
- void `dac_ild_start` (DACDriver *dcp)
Configures and activates the DAC peripheral.
- void `dac_ild_stop` (DACDriver *dcp)
Deactivates the DAC peripheral.
- void `dac_ild_put_channel` (DACDriver *dcp, `dacchannel_t` channel, `dacsample_t` sample)
Outputs a value directly on a DAC channel.
- void `dac_ild_start_conversion` (DACDriver *dcp)
Starts a DAC conversion.
- void `dac_ild_stop_conversion` (DACDriver *dcp)
Stops an ongoing conversion.

9.155.1 Detailed Description

PLATFORM DAC subsystem low level driver header.

9.156 hal_efl_lld.c File Reference

PLATFORM Embedded Flash subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [efl_lld_init](#) (void)
Low level Embedded Flash driver initialization.
- void [efl_lld_start](#) ([EFlashDriver](#) *eflp)
Configures and activates the Embedded Flash peripheral.
- void [efl_lld_stop](#) ([EFlashDriver](#) *eflp)
Deactivates the Embedded Flash peripheral.
- const [flash_descriptor_t](#) * [efl_lld_get_descriptor](#) (void *instance)
Gets the flash descriptor structure.
- [flash_error_t](#) [efl_lld_read](#) (void *instance, [flash_offset_t](#) offset, size_t n, uint8_t *rp)
Read operation.
- [flash_error_t](#) [efl_lld_program](#) (void *instance, [flash_offset_t](#) offset, size_t n, const uint8_t *pp)
Program operation.
- [flash_error_t](#) [efl_lld_start_erase_all](#) (void *instance)
Starts a whole-device erase operation.
- [flash_error_t](#) [efl_lld_start_erase_sector](#) (void *instance, [flash_sector_t](#) sector)
Starts an sector erase operation.
- [flash_error_t](#) [efl_lld_query_erase](#) (void *instance, uint32_t *msec)
Queries the driver for erase operation progress.
- [flash_error_t](#) [efl_lld_verify_erase](#) (void *instance, [flash_sector_t](#) sector)
Returns the erase state of a sector.

Variables

- [EFlashDriver](#) [EFLD1](#)
EFL1 driver identifier.
- static const [flash_descriptor_t](#) [efl_lld_descriptor](#)

9.156.1 Detailed Description

PLATFORM Embedded Flash subsystem low level driver source.

9.157 hal_efl_lld.h File Reference

PLATFORM Embedded Flash subsystem low level driver header.

Macros

- `#define efl_ild_driver_fields`
Low level fields of the embedded flash driver structure.
- `#define efl_ild_config_fields`
Low level fields of the embedded flash configuration structure.

PLATFORM configuration options

- `#define PLATFORM_EFL_USE_EFL1 FALSE`
EFL1 driver enable switch.

Functions

- `void efl_ild_init (void)`
Low level Embedded Flash driver initialization.
- `void efl_ild_start (EFlashDriver *eflp)`
Configures and activates the Embedded Flash peripheral.
- `void efl_ild_stop (EFlashDriver *eflp)`
Deactivates the Embedded Flash peripheral.
- `const flash_descriptor_t * efl_ild_get_descriptor (void *instance)`
Gets the flash descriptor structure.
- `flash_error_t efl_ild_read (void *instance, flash_offset_t offset, size_t n, uint8_t *rp)`
Read operation.
- `flash_error_t efl_ild_program (void *instance, flash_offset_t offset, size_t n, const uint8_t *pp)`
Program operation.
- `flash_error_t efl_ild_start_erase_all (void *instance)`
Starts a whole-device erase operation.
- `flash_error_t efl_ild_start_erase_sector (void *instance, flash_sector_t sector)`
Starts an sector erase operation.
- `flash_error_t efl_ild_query_erase (void *instance, uint32_t *msec)`
Queries the driver for erase operation progress.
- `flash_error_t efl_ild_verify_erase (void *instance, flash_sector_t sector)`
Returns the erase state of a sector.

9.157.1 Detailed Description

PLATFORM Embedded Flash subsystem low level driver header.

9.158 hal_gpt_ild.c File Reference

PLATFORM GPT subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [gpt_lld_init](#) (void)
Low level GPT driver initialization.
- void [gpt_lld_start](#) (GPTDriver *gtp)
Configures and activates the GPT peripheral.
- void [gpt_lld_stop](#) (GPTDriver *gtp)
Deactivates the GPT peripheral.
- void [gpt_lld_start_timer](#) (GPTDriver *gtp, gptcnt_t interval)
Starts the timer in continuous mode.
- void [gpt_lld_stop_timer](#) (GPTDriver *gtp)
Stops the timer.
- void [gpt_lld_polled_delay](#) (GPTDriver *gtp, gptcnt_t interval)
Starts the timer in one shot mode and waits for completion.

Variables

- [GPTDriver GPTD1](#)
GPTD1 driver identifier.

9.158.1 Detailed Description

PLATFORM GPT subsystem low level driver source.

9.159 hal_gpt_lld.h File Reference

PLATFORM GPT subsystem low level driver header.

Data Structures

- struct [GPTConfig](#)
Driver configuration structure.
- struct [GPTDriver](#)
Structure representing a GPT driver.

Macros

- #define [gpt_lld_change_interval](#)(gtp, interval)
Changes the interval of GPT peripheral.

PLATFORM configuration options

- #define [PLATFORM_GPT_USE_GPT1](#) FALSE
GPTD1 driver enable switch.

Typedefs

- typedef uint32_t [gptfreq_t](#)
GPT frequency type.
- typedef uint16_t [gptcnt_t](#)
GPT counter type.

Functions

- void [gpt_lld_init](#) (void)
Low level GPT driver initialization.
- void [gpt_lld_start](#) (GPTDriver *gptp)
Configures and activates the GPT peripheral.
- void [gpt_lld_stop](#) (GPTDriver *gptp)
Deactivates the GPT peripheral.
- void [gpt_lld_start_timer](#) (GPTDriver *gptp, [gptcnt_t](#) interval)
Starts the timer in continuous mode.
- void [gpt_lld_stop_timer](#) (GPTDriver *gptp)
Stops the timer.
- void [gpt_lld_polled_delay](#) (GPTDriver *gptp, [gptcnt_t](#) interval)
Starts the timer in one shot mode and waits for completion.

9.159.1 Detailed Description

PLATFORM GPT subsystem low level driver header.

9.160 hal_i2c_lld.c File Reference

PLATFORM I2C subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [i2c_lld_init](#) (void)
Low level I2C driver initialization.
- void [i2c_lld_start](#) (I2CDriver *i2cp)
Configures and activates the I2C peripheral.
- void [i2c_lld_stop](#) (I2CDriver *i2cp)
Deactivates the I2C peripheral.
- [msg_t i2c_lld_master_receive_timeout](#) (I2CDriver *i2cp, [i2caddr_t](#) addr, uint8_t *rxbuf, size_t rxbytes, [sysinterval_t](#) timeout)
Receives data via the I2C bus as master.
- [msg_t i2c_lld_master_transmit_timeout](#) (I2CDriver *i2cp, [i2caddr_t](#) addr, const uint8_t *txbuf, size_t txbytes, uint8_t *rxbuf, size_t rxbytes, [sysinterval_t](#) timeout)
Transmits data via the I2C bus as master.

Variables

- [I2CDriver I2CD1](#)
I2C1 driver identifier.

9.160.1 Detailed Description

PLATFORM I2C subsystem low level driver source.

9.161 hal_i2c_ild.h File Reference

PLATFORM I2C subsystem low level driver header.

Data Structures

- struct [hal_i2c_config](#)
I2C driver configuration structure.
- struct [hal_i2c_driver](#)
Structure representing an I2C driver.

Macros

- #define [i2c_ild_get_errors\(i2cp\)](#)
Get errors from I2C driver.

PLATFORM configuration options

- #define [PLATFORM_I2C_USE_I2C1](#) FALSE
I2C1 driver enable switch.

Typedefs

- typedef uint16_t [i2caddr_t](#)
Type representing an I2C address.
- typedef uint32_t [i2cflags_t](#)
Type of I2C Driver condition flags.
- typedef struct [hal_i2c_config](#) [I2CConfig](#)
Type of a structure representing an I2C configuration.
- typedef struct [hal_i2c_driver](#) [I2CDriver](#)
Type of a structure representing an I2C driver.

Functions

- void `i2c_lld_init` (void)
Low level I2C driver initialization.
- void `i2c_lld_start` (I2CDriver *i2cp)
Configures and activates the I2C peripheral.
- void `i2c_lld_stop` (I2CDriver *i2cp)
Deactivates the I2C peripheral.
- `msg_t i2c_lld_master_transmit_timeout` (I2CDriver *i2cp, i2caddr_t addr, const uint8_t *txbuf, size_t txbytes, uint8_t *rxbuf, size_t rxbytes, sysinterval_t timeout)
Transmits data via the I2C bus as master.
- `msg_t i2c_lld_master_receive_timeout` (I2CDriver *i2cp, i2caddr_t addr, uint8_t *rxbuf, size_t rxbytes, sysinterval_t timeout)
Receives data via the I2C bus as master.

9.161.1 Detailed Description

PLATFORM I2C subsystem low level driver header.

9.162 hal_i2s_lld.c File Reference

PLATFORM I2S subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void `i2s_lld_init` (void)
Low level I2S driver initialization.
- void `i2s_lld_start` (I2SDriver *i2sp)
Configures and activates the I2S peripheral.
- void `i2s_lld_stop` (I2SDriver *i2sp)
Deactivates the I2S peripheral.
- void `i2s_lld_start_exchange` (I2SDriver *i2sp)
Starts a I2S data exchange.
- void `i2s_lld_stop_exchange` (I2SDriver *i2sp)
Stops the ongoing data exchange.

Variables

- `I2SDriver I2SD1`
I2S2 driver identifier.

9.162.1 Detailed Description

PLATFORM I2S subsystem low level driver source.

9.163 hal_i2s_lld.h File Reference

PLATFORM I2S subsystem low level driver header.

Macros

- `#define i2s_lld_driver_fields`
Low level fields of the I2S driver structure.
- `#define i2s_lld_config_fields`
Low level fields of the I2S configuration structure.

PLATFORM configuration options

- `#define PLATFORM_I2S_USE_I2S1 FALSE`
I2SD1 driver enable switch.

Functions

- `void i2s_lld_init (void)`
Low level I2S driver initialization.
- `void i2s_lld_start (I2SDriver *i2sp)`
Configures and activates the I2S peripheral.
- `void i2s_lld_stop (I2SDriver *i2sp)`
Deactivates the I2S peripheral.
- `void i2s_lld_start_exchange (I2SDriver *i2sp)`
Starts a I2S data exchange.
- `void i2s_lld_stop_exchange (I2SDriver *i2sp)`
Stops the ongoing data exchange.

9.163.1 Detailed Description

PLATFORM I2S subsystem low level driver header.

9.164 hal_icu_lld.c File Reference

PLATFORM ADC subsystem low level driver source.

```
#include "hal.h"
```


Functions

- void [icu_ild_init](#) (void)
Low level ICU driver initialization.
- void [icu_ild_start](#) (ICUDriver *icup)
Configures and activates the ICU peripheral.
- void [icu_ild_stop](#) (ICUDriver *icup)
Deactivates the ICU peripheral.
- void [icu_ild_start_capture](#) (ICUDriver *icup)
Starts the input capture.
- bool [icu_ild_wait_capture](#) (ICUDriver *icup)
Waits for a completed capture.
- void [icu_ild_stop_capture](#) (ICUDriver *icup)
Stops the input capture.
- void [icu_ild_enable_notifications](#) (ICUDriver *icup)
Enables notifications.
- void [icu_ild_disable_notifications](#) (ICUDriver *icup)
Disables notifications.

Variables

- [ICUDriver ICUD1](#)
ICUD1 driver identifier.

9.164.1 Detailed Description

PLATFORM ADC subsystem low level driver source.

9.165 hal_icu_ild.h File Reference

PLATFORM ICU subsystem low level driver header.

Data Structures

- struct [ICUConfig](#)
Driver configuration structure.
- struct [ICUDriver](#)
Structure representing an ICU driver.

Macros

- #define [icu_ild_get_width](#)(icup)
Returns the width of the latest pulse.
- #define [icu_ild_get_period](#)(icup)
Returns the width of the latest cycle.
- #define [icu_ild_are_notifications_enabled](#)(icup)
Check on notifications status.

PLATFORM configuration options

- #define [PLATFORM_ICU_USE_ICU1](#) FALSE
ICUD1 driver enable switch.

Typedefs

- typedef uint32_t [icufreq_t](#)
ICU frequency type.
- typedef uint32_t [icucnt_t](#)
ICU counter type.

Enumerations

- enum [icumode_t](#) { [ICU_INPUT_ACTIVE_HIGH](#) = 0 , [ICU_INPUT_ACTIVE_LOW](#) = 1 }
ICU driver mode.

Functions

- void [icu_lld_init](#) (void)
Low level ICU driver initialization.
- void [icu_lld_start](#) (ICUDriver *icup)
Configures and activates the ICU peripheral.
- void [icu_lld_stop](#) (ICUDriver *icup)
Deactivates the ICU peripheral.
- void [icu_lld_start_capture](#) (ICUDriver *icup)
Starts the input capture.
- bool [icu_lld_wait_capture](#) (ICUDriver *icup)
Waits for a completed capture.
- void [icu_lld_stop_capture](#) (ICUDriver *icup)
Stops the input capture.
- void [icu_lld_enable_notifications](#) (ICUDriver *icup)
Enables notifications.
- void [icu_lld_disable_notifications](#) (ICUDriver *icup)
Disables notifications.

9.165.1 Detailed Description

PLATFORM ICU subsystem low level driver header.

9.166 hal_lld.c File Reference

PLATFORM HAL subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [hal_lld_init](#) (void)
Low level HAL driver initialization.
- bool [hal_lld_clock_switch_mode](#) (const [halclkcfg_t](#) *ccp)
Switches to a different clock configuration.
- [halfreq_t](#) [hal_lld_get_clock_point](#) ([halclkpt_t](#) clkpt)
Returns the frequency of a clock point in Hz.

9.166.1 Detailed Description

PLATFORM HAL subsystem low level driver source.

9.167 hal_ild.h File Reference

PLATFORM HAL subsystem low level driver header.

Data Structures

- struct [halclkcfg_t](#)
Type of a clock configuration structure.

Macros

- #define [HAL_LLD_SELECT_SPI_V2](#) TRUE
Requires use of SPIv2 driver model.
- #define [HAL_LLD_USE_CLOCK_MANAGEMENT](#)
Specifies implementation of dynamic clock management.

Platform identification macros

- #define [PLATFORM_NAME](#) "templates"

Functions

- void [hal_ild_init](#) (void)
Low level HAL driver initialization.
- bool [hal_ild_clock_switch_mode](#) (const [halclkcfg_t](#) *ccp)
Switches to a different clock configuration.
- [halfreq_t](#) [hal_ild_get_clock_point](#) ([halclkpt_t](#) clkpt)
Returns the frequency of a clock point in Hz.

9.167.1 Detailed Description

PLATFORM HAL subsystem low level driver header.

9.168 hal_mac_ild.c File Reference

PLATFORM MAC subsystem low level driver source.

```
#include <string.h>
#include "hal.h"
#include "hal_mii.h"
```

Functions

- void `mac_lld_init` (void)
Low level MAC initialization.
- void `mac_lld_start` (MACDriver *macp)
Configures and activates the MAC peripheral.
- void `mac_lld_stop` (MACDriver *macp)
Deactivates the MAC peripheral.
- msg_t `mac_lld_get_transmit_descriptor` (MACDriver *macp, MACTransmitDescriptor *tdp)
Returns a transmission descriptor.
- void `mac_lld_release_transmit_descriptor` (MACTransmitDescriptor *tdp)
Releases a transmit descriptor and starts the transmission of the enqueued data as a single frame.
- msg_t `mac_lld_get_receive_descriptor` (MACDriver *macp, MACReceiveDescriptor *rdp)
Returns a receive descriptor.
- void `mac_lld_release_receive_descriptor` (MACReceiveDescriptor *rdp)
Releases a receive descriptor.
- bool `mac_lld_poll_link_status` (MACDriver *macp)
Updates and returns the link status.
- size_t `mac_lld_write_transmit_descriptor` (MACTransmitDescriptor *tdp, uint8_t *buf, size_t size)
Writes to a transmit descriptor's stream.
- size_t `mac_lld_read_receive_descriptor` (MACReceiveDescriptor *rdp, uint8_t *buf, size_t size)
Reads from a receive descriptor's stream.
- uint8_t * `mac_lld_get_next_transmit_buffer` (MACTransmitDescriptor *tdp, size_t size, size_t *sizep)
Returns a pointer to the next transmit buffer in the descriptor chain.
- const uint8_t * `mac_lld_get_next_receive_buffer` (MACReceiveDescriptor *rdp, size_t *sizep)
Returns a pointer to the next receive buffer in the descriptor chain.

Variables

- `MACDriver ETHD1`
MAC1 driver identifier.

9.168.1 Detailed Description

PLATFORM MAC subsystem low level driver source.

9.169 hal_mac_lld.h File Reference

PLATFORM MAC subsystem low level driver header.

Macros

- #define `MAC_SUPPORTS_ZERO_COPY` `TRUE`
This implementation supports the zero-copy mode API.
- #define `mac_lld_driver_fields`
Low level fields of the MAC driver structure.
- #define `mac_lld_config_fields`
Low level fields of the MAC configuration structure.
- #define `mac_lld_transmit_descriptor_fields`
Low level fields of the MAC transmit descriptor structure.
- #define `mac_lld_receive_descriptor_fields`
Low level fields of the MAC receive descriptor structure.

PLATFORM configuration options

- #define `PLATFORM_MAC_USE_MAC1` `FALSE`
MAC driver enable switch.

Functions

- void `mac_lld_init` (void)
Low level MAC initialization.
- void `mac_lld_start` (MACDriver *macp)
Configures and activates the MAC peripheral.
- void `mac_lld_stop` (MACDriver *macp)
Deactivates the MAC peripheral.
- msg_t `mac_lld_get_transmit_descriptor` (MACDriver *macp, MACTransmitDescriptor *tdp)
Returns a transmission descriptor.
- void `mac_lld_release_transmit_descriptor` (MACTransmitDescriptor *tdp)
Releases a transmit descriptor and starts the transmission of the enqueued data as a single frame.
- msg_t `mac_lld_get_receive_descriptor` (MACDriver *macp, MACReceiveDescriptor *rdp)
Returns a receive descriptor.
- void `mac_lld_release_receive_descriptor` (MACReceiveDescriptor *rdp)
Releases a receive descriptor.
- bool `mac_lld_poll_link_status` (MACDriver *macp)
Updates and returns the link status.
- size_t `mac_lld_write_transmit_descriptor` (MACTransmitDescriptor *tdp, uint8_t *buf, size_t size)
Writes to a transmit descriptor's stream.
- size_t `mac_lld_read_receive_descriptor` (MACReceiveDescriptor *rdp, uint8_t *buf, size_t size)
Reads from a receive descriptor's stream.
- uint8_t * `mac_lld_get_next_transmit_buffer` (MACTransmitDescriptor *tdp, size_t size, size_t *sizep)
Returns a pointer to the next transmit buffer in the descriptor chain.
- const uint8_t * `mac_lld_get_next_receive_buffer` (MACReceiveDescriptor *rdp, size_t *sizep)
Returns a pointer to the next receive buffer in the descriptor chain.

9.169.1 Detailed Description

PLATFORM MAC subsystem low level driver header.

9.170 hal_pal_lld.c File Reference

PLATFORM PAL subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [_pal_lld_init](#) (void)
STM32 I/O ports configuration.
- void [_pal_lld_setgroupmode](#) (ioportid_t port, ioportmask_t mask, iomode_t mode)
Pads mode setup.

9.170.1 Detailed Description

PLATFORM PAL subsystem low level driver source.

9.171 hal_pal_lld.h File Reference

PLATFORM PAL subsystem low level driver header.

Data Structures

- struct [PALConfig](#)
Generic I/O ports static initializer.

Macros

- #define [PAL_NEW_INIT](#)
- #define [IOPORT1](#) 0
First I/O port identifier.
- #define [pal_lld_init](#)()
Low level PAL subsystem initialization.
- #define [pal_lld_readport](#)(port)
Reads the physical I/O port states.
- #define [pal_lld_readlatch](#)(port)
Reads the output latch.
- #define [pal_lld_writeport](#)(port, bits)
Writes a bits mask on a I/O port.
- #define [pal_lld_setport](#)(port, bits)
Sets a bits mask on a I/O port.
- #define [pal_lld_clearport](#)(port, bits)
Clears a bits mask on a I/O port.
- #define [pal_lld_toggleport](#)(port, bits)
Toggles a bits mask on a I/O port.

- #define `pal_ild_readgroup`(port, mask, offset)
Reads a group of bits.
- #define `pal_ild_writegroup`(port, mask, offset, bits)
Writes a group of bits.
- #define `pal_ild_setgroupmode`(port, mask, offset, mode)
Pads group mode setup.
- #define `pal_ild_readpad`(port, pad)
Reads a logical state from an I/O pad.
- #define `pal_ild_writepad`(port, pad, bit)
Writes a logical state on an output pad.
- #define `pal_ild_setpad`(port, pad)
Sets a pad logical state to `PAL_HIGH`.
- #define `pal_ild_clearpad`(port, pad)
Clears a pad logical state to `PAL_LOW`.
- #define `pal_ild_togglepad`(port, pad)
Toggles a pad logical state.
- #define `pal_ild_setpadmode`(port, pad, mode)
Pad mode setup.
- #define `pal_ild_get_pad_event`(port, pad)
Returns a PAL event structure associated to a pad.
- #define `pal_ild_get_line_event`(line)
Returns a PAL event structure associated to a line.

Port related definitions

- #define `PAL_IOPORTS_WIDTH` 16U
Width, in bits, of an I/O port.
- #define `PAL_WHOLE_PORT` ((`ioportmask_t`)0xFFFFU)
Whole port mask.

Line handling macros

- #define `PAL_LINE`(port, pad)
Forms a line identifier.
- #define `PAL_PORT`(line)
Decodes a port identifier from a line identifier.
- #define `PAL_PAD`(line)
Decodes a pad identifier from a line identifier.
- #define `PAL_NOLINE` 0U
Value identifying an invalid line.

Typedefs

- typedef uint32_t `ioportmask_t`
Digital I/O port sized unsigned type.
- typedef uint32_t `iomode_t`
Digital I/O modes.
- typedef uint32_t `ioline_t`
Type of an I/O line.
- typedef uint32_t `ioportid_t`
Port Identifier.
- typedef uint32_t `iopadid_t`
Type of an pad identifier.

Functions

- void `_pal_lld_init` (void)
STM32 I/O ports configuration.
- void `_pal_lld_setgroupmode` (ioportid_t port, ioportmask_t mask, iomode_t mode)
Pads mode setup.

9.171.1 Detailed Description

PLATFORM PAL subsystem low level driver header.

9.172 hal_pwm_lld.c File Reference

PLATFORM PWM subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void `pwm_lld_init` (void)
Low level PWM driver initialization.
- void `pwm_lld_start` (PWMDriver *pwmp)
Configures and activates the PWM peripheral.
- void `pwm_lld_stop` (PWMDriver *pwmp)
Deactivates the PWM peripheral.
- void `pwm_lld_enable_channel` (PWMDriver *pwmp, pwmchannel_t channel, pwmcnt_t width)
Enables a PWM channel.
- void `pwm_lld_disable_channel` (PWMDriver *pwmp, pwmchannel_t channel)
Disables a PWM channel and its notification.
- void `pwm_lld_enable_periodic_notification` (PWMDriver *pwmp)
Enables the periodic activation edge notification.
- void `pwm_lld_disable_periodic_notification` (PWMDriver *pwmp)
Disables the periodic activation edge notification.
- void `pwm_lld_enable_channel_notification` (PWMDriver *pwmp, pwmchannel_t channel)
Enables a channel de-activation edge notification.
- void `pwm_lld_disable_channel_notification` (PWMDriver *pwmp, pwmchannel_t channel)
Disables a channel de-activation edge notification.

Variables

- `PWMDriver PWMD1`
PWMD1 driver identifier.

9.172.1 Detailed Description

PLATFORM PWM subsystem low level driver source.

9.173 hal_pwm_ild.h File Reference

PLATFORM PWM subsystem low level driver header.

Data Structures

- struct [PWMChannelConfig](#)
Type of a PWM driver channel configuration structure.
- struct [PWMConfig](#)
Type of a PWM driver configuration structure.
- struct [PWMDriver](#)
Structure representing a PWM driver.

Macros

- #define [PWM_CHANNELS](#) 4
Number of PWM channels per PWM driver.
- #define [pwm_ild_change_period](#)(pwmp, period)
Changes the period the PWM peripheral.

PLATFORM configuration options

- #define [PLATFORM_PWM_USE_PWM1](#) FALSE
PWMD1 driver enable switch.

Typedefs

- typedef uint32_t [pwmmode_t](#)
Type of a PWM mode.
- typedef uint8_t [pwmchannel_t](#)
Type of a PWM channel.
- typedef uint32_t [pwmchnmsk_t](#)
Type of a channels mask.
- typedef uint32_t [pwmcnt_t](#)
Type of a PWM counter.

Functions

- void [pwm_ild_init](#) (void)
Low level PWM driver initialization.
- void [pwm_ild_start](#) (PWMDriver *pwmp)
Configures and activates the PWM peripheral.
- void [pwm_ild_stop](#) (PWMDriver *pwmp)
Deactivates the PWM peripheral.
- void [pwm_ild_enable_channel](#) (PWMDriver *pwmp, [pwmchannel_t](#) channel, [pwmcnt_t](#) width)
Enables a PWM channel.
- void [pwm_ild_disable_channel](#) (PWMDriver *pwmp, [pwmchannel_t](#) channel)
Disables a PWM channel and its notification.
- void [pwm_ild_enable_periodic_notification](#) (PWMDriver *pwmp)
Enables the periodic activation edge notification.
- void [pwm_ild_disable_periodic_notification](#) (PWMDriver *pwmp)
Disables the periodic activation edge notification.
- void [pwm_ild_enable_channel_notification](#) (PWMDriver *pwmp, [pwmchannel_t](#) channel)
Enables a channel de-activation edge notification.
- void [pwm_ild_disable_channel_notification](#) (PWMDriver *pwmp, [pwmchannel_t](#) channel)
Disables a channel de-activation edge notification.

9.173.1 Detailed Description

PLATFORM PWM subsystem low level driver header.

9.174 hal_rtc_lld.c File Reference

PLATFORM RTC subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [rtc_lld_init](#) (void)
RTC driver identifier.
- void [rtc_lld_set_time](#) (RTCDriver *rtcp, const RTCDatetime *timespec)
Set current time.
- void [rtc_lld_get_time](#) (RTCDriver *rtcp, RTCDatetime *timespec)
Get current time.
- void [rtc_lld_set_alarm](#) (RTCDriver *rtcp, rtcalarm_t alarm, const RTCAalarm *alarmspec)
Set alarm time.
- void [rtc_lld_get_alarm](#) (RTCDriver *rtcp, rtcalarm_t alarm, RTCAalarm *alarmspec)
Get alarm time.

9.174.1 Detailed Description

PLATFORM RTC subsystem low level driver source.

9.175 hal_rtc_lld.h File Reference

PLATFORM RTC subsystem low level driver header.

Data Structures

- struct [RTCAalarm](#)
Type of a structure representing an RTC alarm time stamp.

Macros

- #define `rtc_ild_driver_fields` `uint32_t dummy;`
Implementation-specific `RTCDriver` fields.

Implementation capabilities

- #define `RTC_SUPPORTS_CALLBACKS` `TRUE`
Callback support int the driver.
- #define `RTC_ALARMS` `2`
Number of alarms available.
- #define `RTC_HAS_STORAGE` `FALSE`
Presence of a local persistent storage.

PLATFORM configuration options

- #define `PLATFORM_RTC_USE_RTC1` `FALSE`
RTCD1 driver enable switch.

Typedefs

- typedef void(* `rtccb_t`) (`RTCDriver *rtcp`, `rtcevent_t event`)
Type of a generic RTC callback.

Enumerations

- enum `rtcevent_t` { `RTC_EVENT_SECOND` = 0 }
Type of an RTC event.

Functions

- void `rtc_ild_init` (void)
RTC driver identifier.
- void `rtc_ild_set_time` (`RTCDriver *rtcp`, const `RTCDateTime *timespec`)
Set current time.
- void `rtc_ild_get_time` (`RTCDriver *rtcp`, `RTCDateTime *timespec`)
Get current time.
- void `rtc_ild_set_callback` (`RTCDriver *rtcp`, `rtccb_t callback`)

9.175.1 Detailed Description

PLATFORM RTC subsystem low level driver header.

9.176 hal_sdc_ild.c File Reference

PLATFORM SDC subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void `sdcd_ild_init` (void)
Low level SDC driver initialization.
- void `sdcd_ild_start` (SDCDriver *sdcp)
Configures and activates the SDC peripheral.
- void `sdcd_ild_stop` (SDCDriver *sdcp)
Deactivates the SDC peripheral.
- void `sdcd_ild_start_clk` (SDCDriver *sdcp)
Starts the SDIO clock and sets it to init mode (400kHz or less).
- void `sdcd_ild_set_data_clk` (SDCDriver *sdcp, sdcbusclk_t clk)
Sets the SDIO clock to data mode (25MHz or less).
- void `sdcd_ild_stop_clk` (SDCDriver *sdcp)
Stops the SDIO clock.
- void `sdcd_ild_set_bus_mode` (SDCDriver *sdcp, sdcbusmode_t mode)
Switches the bus to 4 bits mode.
- void `sdcd_ild_send_cmd_none` (SDCDriver *sdcp, uint8_t cmd, uint32_t arg)
Sends an SDIO command with no response expected.
- bool `sdcd_ild_send_cmd_short` (SDCDriver *sdcp, uint8_t cmd, uint32_t arg, uint32_t *resp)
Sends an SDIO command with a short response expected.
- bool `sdcd_ild_send_cmd_short_crc` (SDCDriver *sdcp, uint8_t cmd, uint32_t arg, uint32_t *resp)
Sends an SDIO command with a short response expected and CRC.
- bool `sdcd_ild_send_cmd_long_crc` (SDCDriver *sdcp, uint8_t cmd, uint32_t arg, uint32_t *resp)
Sends an SDIO command with a long response expected and CRC.
- bool `sdcd_ild_read` (SDCDriver *sdcp, uint32_t startblk, uint8_t *buf, uint32_t n)
Reads one or more blocks.
- bool `sdcd_ild_write` (SDCDriver *sdcp, uint32_t startblk, const uint8_t *buf, uint32_t n)
Writes one or more blocks.
- bool `sdcd_ild_sync` (SDCDriver *sdcp)
Waits for card idle condition.

Variables

- `SDCDriver SDCD1`
SDCD1 driver identifier.

9.176.1 Detailed Description

PLATFORM SDC subsystem low level driver source.

9.177 hal_sdcd_ild.h File Reference

PLATFORM SDC subsystem low level driver header.

Data Structures

- struct [SDCConfig](#)
Driver configuration structure.
- struct [SDCDriverVMT](#)
[SDCDriver](#) virtual methods table.
- struct [SDCDriver](#)
Structure representing an SDC driver.

Macros

- #define [_sdc_driver_methods _mmc_sdc_block_device_methods](#)
[SDCDriver](#) specific methods.

PLATFORM configuration options

- #define [PLATFORM_SDC_USE_SDC1](#) FALSE
PWMD1 driver enable switch.

Typedefs

- typedef uint32_t [sdcmode_t](#)
Type of card flags.
- typedef uint32_t [sdcflags_t](#)
SDC Driver condition flags type.
- typedef struct SDCDriver [SDCDriver](#)
Type of a structure representing an SDC driver.

Functions

- void [sdc_ild_init](#) (void)
Low level SDC driver initialization.
- void [sdc_ild_start](#) ([SDCDriver](#) *sdcp)
Configures and activates the SDC peripheral.
- void [sdc_ild_stop](#) ([SDCDriver](#) *sdcp)
Deactivates the SDC peripheral.
- void [sdc_ild_start_clk](#) ([SDCDriver](#) *sdcp)
Starts the SDIO clock and sets it to init mode (400kHz or less).
- void [sdc_ild_set_data_clk](#) ([SDCDriver](#) *sdcp, [sdcbusclk_t](#) clk)
Sets the SDIO clock to data mode (25MHz or less).
- void [sdc_ild_stop_clk](#) ([SDCDriver](#) *sdcp)
Stops the SDIO clock.
- void [sdc_ild_set_bus_mode](#) ([SDCDriver](#) *sdcp, [sdcbusmode_t](#) mode)
Switches the bus to 4 bits mode.
- void [sdc_ild_send_cmd_none](#) ([SDCDriver](#) *sdcp, uint8_t cmd, uint32_t arg)
Sends an SDIO command with no response expected.
- bool [sdc_ild_send_cmd_short](#) ([SDCDriver](#) *sdcp, uint8_t cmd, uint32_t arg, uint32_t *resp)
Sends an SDIO command with a short response expected.
- bool [sdc_ild_send_cmd_short_crc](#) ([SDCDriver](#) *sdcp, uint8_t cmd, uint32_t arg, uint32_t *resp)

Sends an SDIO command with a short response expected and CRC.

- bool [sd_c_lld_send_cmd_long_crc](#) ([SDCDriver](#) *sdcp, uint8_t cmd, uint32_t arg, uint32_t *resp)

Sends an SDIO command with a long response expected and CRC.

- bool [sd_c_lld_read_special](#) ([SDCDriver](#) *sdcp, uint8_t *buf, size_t bytes, uint8_t cmd, uint32_t argument)
- bool [sd_c_lld_read](#) ([SDCDriver](#) *sdcp, uint32_t startblk, uint8_t *buf, uint32_t n)

Reads one or more blocks.

- bool [sd_c_lld_write](#) ([SDCDriver](#) *sdcp, uint32_t startblk, const uint8_t *buf, uint32_t n)

Writes one or more blocks.

- bool [sd_c_lld_sync](#) ([SDCDriver](#) *sdcp)

Waits for card idle condition.

- bool [sd_c_lld_is_card_inserted](#) ([SDCDriver](#) *sdcp)
- bool [sd_c_lld_is_write_protected](#) ([SDCDriver](#) *sdcp)

9.177.1 Detailed Description

PLATFORM SDC subsystem low level driver header.

9.178 hal_serial_lld.c File Reference

PLATFORM serial subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [sd_lld_init](#) (void)
Low level serial driver initialization.
- void [sd_lld_start](#) ([SerialDriver](#) *sdp, const [SerialConfig](#) *config)
Low level serial driver configuration and (re)start.
- void [sd_lld_stop](#) ([SerialDriver](#) *sdp)
Low level serial driver stop.

Variables

- [SerialDriver](#) [SD1](#)
USART1 serial driver identifier.
- static const [SerialConfig](#) [default_config](#)
Driver default configuration.

9.178.1 Detailed Description

PLATFORM serial subsystem low level driver source.

9.179 hal_serial_lld.h File Reference

PLATFORM serial subsystem low level driver header.

Data Structures

- struct [hal_serial_config](#)
PLATFORM Serial Driver configuration structure.

Macros

- #define [_serial_driver_data](#)
SerialDriver specific data.

PLATFORM configuration options

- #define [PLATFORM_SERIAL_USE_USART1](#) FALSE
USART1 driver enable switch.

Typedefs

- typedef struct [hal_serial_config](#) SerialConfig
PLATFORM Serial Driver configuration structure.

Functions

- void [sd_lld_init](#) (void)
Low level serial driver initialization.
- void [sd_lld_start](#) ([SerialDriver](#) *sdp, const [SerialConfig](#) *config)
Low level serial driver configuration and (re)start.
- void [sd_lld_stop](#) ([SerialDriver](#) *sdp)
Low level serial driver stop.

9.179.1 Detailed Description

PLATFORM serial subsystem low level driver header.

9.180 hal_sio_lld.c File Reference

PLATFORM SIO subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [sio_lld_init](#) (void)
Low level SIO driver initialization.
- [msg_t sio_lld_start](#) (SIODriver *siop)
Configures and activates the SIO peripheral.
- void [sio_lld_stop](#) (SIODriver *siop)
Deactivates the SIO peripheral.
- void [sio_lld_update_enable_flags](#) (SIODriver *siop)
Enable flags change notification.
- [sioevents_t sio_lld_get_and_clear_errors](#) (SIODriver *siop)
Get and clears SIO error event flags.
- [sioevents_t sio_lld_get_and_clear_events](#) (SIODriver *siop)
Get and clears SIO event flags.
- [sioevents_t sio_lld_get_events](#) (SIODriver *siop)
Returns the pending SIO event flags.
- [size_t sio_lld_read](#) (SIODriver *siop, uint8_t *buffer, size_t n)
Reads data from the RX FIFO.
- [size_t sio_lld_write](#) (SIODriver *siop, const uint8_t *buffer, size_t n)
Writes data into the TX FIFO.
- [msg_t sio_lld_get](#) (SIODriver *siop)
Returns one frame from the RX FIFO.
- void [sio_lld_put](#) (SIODriver *siop, uint_fast16_t data)
Pushes one frame into the TX FIFO.
- [msg_t sio_lld_control](#) (SIODriver *siop, unsigned int operation, void *arg)
Control operation on a serial port.
- void [sio_lld_serve_interrupt](#) (SIODriver *siop)
Serves an UART interrupt.

Variables

- [SIODriver SIOD1](#)
SIO1 driver identifier.

9.180.1 Detailed Description

PLATFORM SIO subsystem low level driver source.

9.181 hal_sio_lld.h File Reference

PLATFORM SIO subsystem low level driver header.

Macros

- `#define sio_ild_driver_fields uint32_t dummy;`
Low level fields of the SIO driver structure.
- `#define sio_ild_config_fields uint32_t dummy;`
Low level fields of the SIO configuration structure.
- `#define sio_ild_is_rx_empty(siop)`
Determines the state of the RX FIFO.
- `#define sio_ild_is_rx_idle(siop)`
Determines the activity state of the receiver.
- `#define sio_ild_has_rx_errors(siop)`
Determines if RX has pending error events to be read and cleared.
- `#define sio_ild_is_tx_full(siop)`
Determines the state of the TX FIFO.
- `#define sio_ild_is_tx_ongoing(siop)`
Determines the transmission state.

PLATFORM configuration options

- `#define PLATFORM_SIO_USE_SIO1 FALSE`
SIO driver enable switch.

Functions

- `void sio_ild_init (void)`
Low level SIO driver initialization.
- `msg_t sio_ild_start (SIODriver *siop)`
Configures and activates the SIO peripheral.
- `void sio_ild_stop (SIODriver *siop)`
Deactivates the SIO peripheral.
- `void sio_ild_update_enable_flags (SIODriver *siop)`
Enable flags change notification.
- `sioevents_t sio_ild_get_and_clear_errors (SIODriver *siop)`
Get and clears SIO error event flags.
- `sioevents_t sio_ild_get_and_clear_events (SIODriver *siop)`
Get and clears SIO event flags.
- `sioevents_t sio_ild_get_events (SIODriver *siop)`
Returns the pending SIO event flags.
- `size_t sio_ild_read (SIODriver *siop, uint8_t *buffer, size_t n)`
Reads data from the RX FIFO.
- `size_t sio_ild_write (SIODriver *siop, const uint8_t *buffer, size_t n)`
Writes data into the TX FIFO.
- `msg_t sio_ild_get (SIODriver *siop)`
Returns one frame from the RX FIFO.
- `void sio_ild_put (SIODriver *siop, uint_fast16_t data)`
Pushes one frame into the TX FIFO.
- `msg_t sio_ild_control (SIODriver *siop, unsigned int operation, void *arg)`
Control operation on a serial port.
- `void sio_ild_serve_interrupt (SIODriver *siop)`
Serves an UART interrupt.

9.181.1 Detailed Description

PLATFORM SIO subsystem low level driver header.

9.182 hal_spi_ild.c File Reference

PLATFORM SPI (v1) subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [spi_ild_init](#) (void)
Low level SPI driver initialization.
- void [spi_ild_start](#) (SPIDriver *spip)
Configures and activates the SPI peripheral.
- void [spi_ild_stop](#) (SPIDriver *spip)
Deactivates the SPI peripheral.
- void [spi_ild_select](#) (SPIDriver *spip)
Asserts the slave select signal and prepares for transfers.
- void [spi_ild_unselect](#) (SPIDriver *spip)
Deasserts the slave select signal.
- void [spi_ild_ignore](#) (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- void [spi_ild_exchange](#) (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- void [spi_ild_send](#) (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- void [spi_ild_receive](#) (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- void [spi_ild_abort](#) (SPIDriver *spip)
Aborts the ongoing SPI operation, if any.
- uint16_t [spi_ild_polled_exchange](#) (SPIDriver *spip, uint16_t frame)
Exchanges one frame using a polled wait.

Variables

- [SPIDriver SPID1](#)
SPI1 driver identifier.

9.182.1 Detailed Description

PLATFORM SPI (v1) subsystem low level driver source.

9.183 hal_spi_ild.h File Reference

PLATFORM SPI (v1) subsystem low level driver header.

Macros

- #define `SPI_SUPPORTS_CIRCULAR TRUE`
Circular mode support flag.
- #define `spi_ild_driver_fields`
Low level fields of the SPI driver structure.
- #define `spi_ild_config_fields`
Low level fields of the SPI configuration structure.

PLATFORM configuration options

- #define `PLATFORM_SPI_USE_SPI1 FALSE`
SPI1 driver enable switch.

Functions

- void `spi_ild_init` (void)
Low level SPI driver initialization.
- void `spi_ild_start` (SPIDriver *spip)
Configures and activates the SPI peripheral.
- void `spi_ild_stop` (SPIDriver *spip)
Deactivates the SPI peripheral.
- void `spi_ild_select` (SPIDriver *spip)
Asserts the slave select signal and prepares for transfers.
- void `spi_ild_unselect` (SPIDriver *spip)
Deasserts the slave select signal.
- void `spi_ild_ignore` (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- void `spi_ild_exchange` (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- void `spi_ild_send` (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- void `spi_ild_receive` (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- void `spi_ild_abort` (SPIDriver *spip)
Aborts the ongoing SPI operation, if any.
- uint16_t `spi_ild_polled_exchange` (SPIDriver *spip, uint16_t frame)
Exchanges one frame using a polled wait.

9.183.1 Detailed Description

PLATFORM SPI (v1) subsystem low level driver header.

9.184 hal_spi_v2_lld.c File Reference

PLATFORM SPI (v2) subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [spi_lld_init](#) (void)
Low level SPI driver initialization.
- [msg_t spi_lld_start](#) (SPIDriver *spip)
Configures and activates the SPI peripheral.
- void [spi_lld_stop](#) (SPIDriver *spip)
Deactivates the SPI peripheral.
- void [spi_lld_select](#) (SPIDriver *spip)
Asserts the slave select signal and prepares for transfers.
- void [spi_lld_unselect](#) (SPIDriver *spip)
Deasserts the slave select signal.
- [msg_t spi_lld_ignore](#) (SPIDriver *spip, size_t n)
Ignores data on the SPI bus.
- [msg_t spi_lld_exchange](#) (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)
Exchanges data on the SPI bus.
- [msg_t spi_lld_send](#) (SPIDriver *spip, size_t n, const void *txbuf)
Sends data over the SPI bus.
- [msg_t spi_lld_receive](#) (SPIDriver *spip, size_t n, void *rxbuf)
Receives data from the SPI bus.
- [msg_t spi_lld_stop_transfer](#) (SPIDriver *spip, size_t *sizep)
Aborts the ongoing SPI operation, if any.
- uint16_t [spi_lld_polled_exchange](#) (SPIDriver *spip, uint16_t frame)
Exchanges one frame using a polled wait.

Variables

- [SPIDriver SPID1](#)
SPI1 driver identifier.

9.184.1 Detailed Description

PLATFORM SPI (v2) subsystem low level driver source.

9.185 hal_spi_v2_lld.h File Reference

PLATFORM SPI (v2) subsystem low level driver header.

Macros

- `#define SPI_SUPPORTS_CIRCULAR TRUE`
Circular mode support flag.
- `#define SPI_SUPPORTS_SLAVE_MODE TRUE`
Slave mode support flag.
- `#define spi_ild_driver_fields`
Low level fields of the SPI driver structure.
- `#define spi_ild_config_fields`
Low level fields of the SPI configuration structure.

PLATFORM configuration options

- `#define PLATFORM_SPI_USE_SPI1 FALSE`
SPI1 driver enable switch.

Functions

- `void spi_ild_init (void)`
Low level SPI driver initialization.
- `msg_t spi_ild_start (SPIDriver *spip)`
Configures and activates the SPI peripheral.
- `void spi_ild_stop (SPIDriver *spip)`
Deactivates the SPI peripheral.
- `void spi_ild_select (SPIDriver *spip)`
Asserts the slave select signal and prepares for transfers.
- `void spi_ild_unselect (SPIDriver *spip)`
Deasserts the slave select signal.
- `msg_t spi_ild_ignore (SPIDriver *spip, size_t n)`
Ignores data on the SPI bus.
- `msg_t spi_ild_exchange (SPIDriver *spip, size_t n, const void *txbuf, void *rxbuf)`
Exchanges data on the SPI bus.
- `msg_t spi_ild_send (SPIDriver *spip, size_t n, const void *txbuf)`
Sends data over the SPI bus.
- `msg_t spi_ild_receive (SPIDriver *spip, size_t n, void *rxbuf)`
Receives data from the SPI bus.
- `msg_t spi_ild_stop_transfer (SPIDriver *spip, size_t *sizep)`
Aborts the ongoing SPI operation, if any.
- `uint16_t spi_ild_polled_exchange (SPIDriver *spip, uint16_t frame)`
Exchanges one frame using a polled wait.

9.185.1 Detailed Description

PLATFORM SPI (v2) subsystem low level driver header.

9.186 hal_st_ild.c File Reference

PLATFORM ST subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [st_lld_init](#) (void)
Low level ST driver initialization.

9.186.1 Detailed Description

PLATFORM ST subsystem low level driver source.

9.187 hal_st_lld.h File Reference

PLATFORM ST subsystem low level driver header.

Functions

- void [st_lld_init](#) (void)
Low level ST driver initialization.
- static [systime_t st_lld_get_counter](#) (void)
Returns the time counter value.
- static void [st_lld_start_alarm](#) ([systime_t](#) abstime)
Starts the alarm.
- static void [st_lld_stop_alarm](#) (void)
Stops the alarm interrupt.
- static void [st_lld_set_alarm](#) ([systime_t](#) abstime)
Sets the alarm time.
- static [systime_t st_lld_get_alarm](#) (void)
Returns the current alarm time.
- static bool [st_lld_is_alarm_active](#) (void)
Determines if the alarm is active.

9.187.1 Detailed Description

PLATFORM ST subsystem low level driver header.

This header is designed to be include-able without having to include other files from the HAL.

9.188 hal_trng_lld.c File Reference

PLATFORM TRNG subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [trng_lld_init](#) (void)
Low level TRNG driver initialization.
- void [trng_lld_start](#) (TRNGDriver *trngp)
Configures and activates the TRNG peripheral.
- void [trng_lld_stop](#) (TRNGDriver *trngp)
Deactivates the TRNG peripheral.
- bool [trng_lld_generate](#) (TRNGDriver *trngp, size_t size, uint8_t *out)
True random numbers generator.

Variables

- [TRNGDriver TRNGD1](#)
TRNGD1 driver identifier.

9.188.1 Detailed Description

PLATFORM TRNG subsystem low level driver source.

9.189 hal_trng_lld.h File Reference

PLATFORM TRNG subsystem low level driver header.

Macros

- #define [trng_lld_driver_fields](#)
Low level fields of the TRNG driver structure.
- #define [trng_lld_config_fields](#)
Low level fields of the TRNG configuration structure.

PLATFORM configuration options

- #define [PLATFORM_TRNG_USE_TRNG1](#) FALSE
TRNGD1 driver enable switch.

Functions

- void [trng_lld_init](#) (void)
Low level TRNG driver initialization.
- void [trng_lld_start](#) (TRNGDriver *trngp)
Configures and activates the TRNG peripheral.
- void [trng_lld_stop](#) (TRNGDriver *trngp)
Deactivates the TRNG peripheral.
- bool [trng_lld_generate](#) (TRNGDriver *trngp, size_t size, uint8_t *out)
True random numbers generator.

9.189.1 Detailed Description

PLATFORM TRNG subsystem low level driver header.

9.190 hal_uart_lld.c File Reference

PLATFORM UART subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [uart_lld_init](#) (void)
Low level UART driver initialization.
- void [uart_lld_start](#) (UARTDriver *uartp)
Configures and activates the UART peripheral.
- void [uart_lld_stop](#) (UARTDriver *uartp)
Deactivates the UART peripheral.
- void [uart_lld_start_send](#) (UARTDriver *uartp, size_t n, const void *txbuf)
Starts a transmission on the UART peripheral.
- size_t [uart_lld_stop_send](#) (UARTDriver *uartp)
Stops any ongoing transmission.
- void [uart_lld_start_receive](#) (UARTDriver *uartp, size_t n, void *rxbuf)
Starts a receive operation on the UART peripheral.
- size_t [uart_lld_stop_receive](#) (UARTDriver *uartp)
Stops any ongoing receive operation.

Variables

- [UARTDriver UARTD1](#)
UART1 driver identifier.

9.190.1 Detailed Description

PLATFORM UART subsystem low level driver source.

9.191 hal_uart_lld.h File Reference

PLATFORM UART subsystem low level driver header.

Data Structures

- struct [hal_uart_config](#)
Driver configuration structure.
- struct [hal_uart_driver](#)
Structure representing an UART driver.

Macros

PLATFORM configuration options

- #define [PLATFORM_UART_USE_UART1](#) FALSE
UART driver enable switch.

Typedefs

- typedef uint32_t [uartflags_t](#)
UART driver condition flags type.
- typedef struct [hal_uart_driver](#) UARTDriver
Type of structure representing an UART driver.
- typedef void(* [uartcb_t](#)) (UARTDriver *uartp)
Generic UART notification callback type.
- typedef void(* [uartccb_t](#)) (UARTDriver *uartp, uint16_t c)
Character received UART notification callback type.
- typedef void(* [uartecb_t](#)) (UARTDriver *uartp, [uartflags_t](#) e)
Receive error UART notification callback type.
- typedef struct [hal_uart_config](#) UARTConfig
Driver configuration structure.

Functions

- void [uart_ild_init](#) (void)
Low level UART driver initialization.
- void [uart_ild_start](#) (UARTDriver *uartp)
Configures and activates the UART peripheral.
- void [uart_ild_stop](#) (UARTDriver *uartp)
Deactivates the UART peripheral.
- void [uart_ild_start_send](#) (UARTDriver *uartp, size_t n, const void *txbuf)
Starts a transmission on the UART peripheral.
- size_t [uart_ild_stop_send](#) (UARTDriver *uartp)
Stops any ongoing transmission.
- void [uart_ild_start_receive](#) (UARTDriver *uartp, size_t n, void *rxbuf)
Starts a receive operation on the UART peripheral.
- size_t [uart_ild_stop_receive](#) (UARTDriver *uartp)
Stops any ongoing receive operation.

9.191.1 Detailed Description

PLATFORM UART subsystem low level driver header.

9.192 hal_usb_lld.c File Reference

PLATFORM USB subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void [usb_lld_init](#) (void)
Low level USB driver initialization.
- void [usb_lld_start](#) (USBDriver *usbp)
Configures and activates the USB peripheral.
- void [usb_lld_stop](#) (USBDriver *usbp)
Deactivates the USB peripheral.
- void [usb_lld_reset](#) (USBDriver *usbp)
USB low level reset routine.
- void [usb_lld_set_address](#) (USBDriver *usbp)
Sets the USB address.
- void [usb_lld_init_endpoint](#) (USBDriver *usbp, [usbep_t](#) ep)
Enables an endpoint.
- void [usb_lld_disable_endpoints](#) (USBDriver *usbp)
Disables all the active endpoints except the endpoint zero.
- [usbepstatus_t](#) [usb_lld_get_status_out](#) (USBDriver *usbp, [usbep_t](#) ep)
Returns the status of an OUT endpoint.
- [usbepstatus_t](#) [usb_lld_get_status_in](#) (USBDriver *usbp, [usbep_t](#) ep)
Returns the status of an IN endpoint.
- void [usb_lld_read_setup](#) (USBDriver *usbp, [usbep_t](#) ep, uint8_t *buf)
Reads a setup packet from the dedicated packet buffer.
- void [usb_lld_prepare_receive](#) (USBDriver *usbp, [usbep_t](#) ep)
Prepares for a receive operation.
- void [usb_lld_prepare_transmit](#) (USBDriver *usbp, [usbep_t](#) ep)
Prepares for a transmit operation.
- void [usb_lld_start_out](#) (USBDriver *usbp, [usbep_t](#) ep)
Starts a receive operation on an OUT endpoint.
- void [usb_lld_start_in](#) (USBDriver *usbp, [usbep_t](#) ep)
Starts a transmit operation on an IN endpoint.
- void [usb_lld_stall_out](#) (USBDriver *usbp, [usbep_t](#) ep)
Brings an OUT endpoint in the stalled state.
- void [usb_lld_stall_in](#) (USBDriver *usbp, [usbep_t](#) ep)
Brings an IN endpoint in the stalled state.
- void [usb_lld_clear_out](#) (USBDriver *usbp, [usbep_t](#) ep)
Brings an OUT endpoint in the active state.
- void [usb_lld_clear_in](#) (USBDriver *usbp, [usbep_t](#) ep)
Brings an IN endpoint in the active state.

Variables

- [USBDriver USBD1](#)
USB1 driver identifier.
- union {
 [USBInEndpointState in](#)
 IN EP0 state.
 [USBOutEndpointState out](#)
 OUT EP0 state.
} [ep0_state](#)

 EP0 state.
- static const [USBEndpointConfig ep0config](#)
 EP0 initialization structure.

9.192.1 Detailed Description

PLATFORM USB subsystem low level driver source.

9.192.2 Variable Documentation

9.192.2.1 in

[USBInEndpointState in](#)

IN EP0 state.

9.192.2.2 out

[USBOutEndpointState out](#)

OUT EP0 state.

9.193 hal_usb_llid.h File Reference

PLATFORM USB subsystem low level driver header.

Data Structures

- struct [USBInEndpointState](#)
 Type of an IN endpoint state structure.
- struct [USBOutEndpointState](#)
 Type of an OUT endpoint state structure.
- struct [USBEndpointConfig](#)
 Type of an USB endpoint configuration structure.
- struct [USBConfig](#)
 Type of an USB driver configuration structure.
- struct [USBDriver](#)
 Structure representing an USB driver.

Macros

- #define `USB_MAX_ENDPOINTS` 4
Maximum endpoint address.
- #define `USB_EP0_STATUS_STAGE` `USB_EP0_STATUS_STAGE_SW`
Status stage handling method.
- #define `USB_SET_ADDRESS_MODE` `USB_LATE_SET_ADDRESS`
The address can be changed immediately upon packet reception.
- #define `USB_SET_ADDRESS_ACK_HANDLING` `USB_SET_ADDRESS_ACK_SW`
Method for set address acknowledge.
- #define `usb_lld_get_frame_number`(usbp)
Returns the current frame number.
- #define `usb_lld_get_transaction_size`(usbp, ep)
Returns the exact size of a receive transaction.
- #define `usb_lld_connect_bus`(usbp)
Connects the USB device.
- #define `usb_lld_disconnect_bus`(usbp)
Disconnect the USB device.
- #define `usb_lld_wakeup_host`(usbp)
Start of host wake-up procedure.

PLATFORM configuration options

- #define `PLATFORM_USB_USE_USB1` `FALSE`
USB driver enable switch.

Functions

- void `usb_lld_init` (void)
Low level USB driver initialization.
- void `usb_lld_start` (USBDriver *usbp)
Configures and activates the USB peripheral.
- void `usb_lld_stop` (USBDriver *usbp)
Deactivates the USB peripheral.
- void `usb_lld_reset` (USBDriver *usbp)
USB low level reset routine.
- void `usb_lld_set_address` (USBDriver *usbp)
Sets the USB address.
- void `usb_lld_init_endpoint` (USBDriver *usbp, usbep_t ep)
Enables an endpoint.
- void `usb_lld_disable_endpoints` (USBDriver *usbp)
Disables all the active endpoints except the endpoint zero.
- `usbepstatus_t` `usb_lld_get_status_in` (USBDriver *usbp, usbep_t ep)
Returns the status of an IN endpoint.
- `usbepstatus_t` `usb_lld_get_status_out` (USBDriver *usbp, usbep_t ep)
Returns the status of an OUT endpoint.
- void `usb_lld_read_setup` (USBDriver *usbp, usbep_t ep, uint8_t *buf)
Reads a setup packet from the dedicated packet buffer.
- void `usb_lld_prepare_receive` (USBDriver *usbp, usbep_t ep)
Prepares for a receive operation.
- void `usb_lld_prepare_transmit` (USBDriver *usbp, usbep_t ep)

- void `usb_lld_start_out` (`USBDriver *usbp`, `usbep_t ep`)
Prepares for a transmit operation.
- void `usb_lld_start_in` (`USBDriver *usbp`, `usbep_t ep`)
Starts a receive operation on an OUT endpoint.
- void `usb_lld_stall_out` (`USBDriver *usbp`, `usbep_t ep`)
Starts a transmit operation on an IN endpoint.
- void `usb_lld_stall_in` (`USBDriver *usbp`, `usbep_t ep`)
Brings an OUT endpoint in the stalled state.
- void `usb_lld_clear_out` (`USBDriver *usbp`, `usbep_t ep`)
Brings an IN endpoint in the stalled state.
- void `usb_lld_clear_in` (`USBDriver *usbp`, `usbep_t ep`)
Brings an OUT endpoint in the active state.
- void `usb_lld_clear_in` (`USBDriver *usbp`, `usbep_t ep`)
Brings an IN endpoint in the active state.

9.193.1 Detailed Description

PLATFORM USB subsystem low level driver header.

9.194 hal_wdg_lld.c File Reference

WDG Driver subsystem low level driver source template.

```
#include "hal.h"
```

Functions

- void `wdg_lld_init` (void)
Low level WDG driver initialization.
- void `wdg_lld_start` (`WDGDriver *wdgp`)
Configures and activates the WDG peripheral.
- void `wdg_lld_stop` (`WDGDriver *wdgp`)
Deactivates the WDG peripheral.
- void `wdg_lld_reset` (`WDGDriver *wdgp`)
Reloads WDG's counter.

Variables

- `WDGDriver WGD1`

9.194.1 Detailed Description

WDG Driver subsystem low level driver source template.

9.195 hal_wdg_lld.h File Reference

WDG Driver subsystem low level driver header template.

Data Structures

- struct [WDGConfig](#)
Driver configuration structure.
- struct [WDGDriver](#)
Structure representing an WDG driver.

Macros

Configuration options

- #define [PLATFORM_WDG_USE_WDG1](#) FALSE
WDG1 driver enable switch.

Typedefs

- typedef struct WDGDriver [WDGDriver](#)
Type of a structure representing an WDG driver.

Functions

- void [wdg_lld_init](#) (void)
Low level WDG driver initialization.
- void [wdg_lld_start](#) ([WDGDriver](#) *wdgp)
Configures and activates the WDG peripheral.
- void [wdg_lld_stop](#) ([WDGDriver](#) *wdgp)
Deactivates the WDG peripheral.
- void [wdg_lld_reset](#) ([WDGDriver](#) *wdgp)
Reloads WDG's counter.

9.195.1 Detailed Description

WDG Driver subsystem low level driver header template.

9.196 hal_wspi_lld.c File Reference

PLATFORM WSPI subsystem low level driver source.

```
#include "hal.h"
```

Functions

- void `wspi_lld_init` (void)
Low level WSPI driver initialization.
- void `wspi_lld_start` (WSPIDriver *wspip)
Configures and activates the WSPI peripheral.
- void `wspi_lld_stop` (WSPIDriver *wspip)
Deactivates the WSPI peripheral.
- void `wspi_lld_command` (WSPIDriver *wspip, const `wspi_command_t` *cmdp)
Sends a command without data phase.
- void `wspi_lld_send` (WSPIDriver *wspip, const `wspi_command_t` *cmdp, size_t n, const uint8_t *txbuf)
Sends a command with data over the WSPI bus.
- void `wspi_lld_receive` (WSPIDriver *wspip, const `wspi_command_t` *cmdp, size_t n, uint8_t *rxbuf)
Sends a command then receives data over the WSPI bus.
- void `wspi_lld_map_flash` (WSPIDriver *wspip, const `wspi_command_t` *cmdp, uint8_t **addrp)
Maps in memory space a WSPI flash device.
- void `wspi_lld_unmap_flash` (WSPIDriver *wspip)
Unmaps from memory space a WSPI flash device.

Variables

- `WSPIDriver WSPID1`
WSPID1 driver identifier.

9.196.1 Detailed Description

PLATFORM WSPI subsystem low level driver source.

9.197 hal_wspi_lld.h File Reference

PLATFORM WSPI subsystem low level driver header.

Macros

- `#define wspi_lld_driver_fields`
Low level fields of the WSPI driver structure.
- `#define wspi_lld_config_fields`
Low level fields of the WSPI configuration structure.

WSPI implementation capabilities

- `#define WSPI_SUPPORTS_MEMMAP TRUE`
- `#define WSPI_DEFAULT_CFG_MASKS TRUE`

Configuration options

- `#define PLATFORM_WSPI_USE_WSPID1 FALSE`
WSPID1 driver enable switch.

Functions

- void `wspi_lld_init` (void)
Low level WSPI driver initialization.
- void `wspi_lld_start` (WSPIDriver *wspip)
Configures and activates the WSPI peripheral.
- void `wspi_lld_stop` (WSPIDriver *wspip)
Deactivates the WSPI peripheral.
- void `wspi_lld_command` (WSPIDriver *wspip, const `wspi_command_t` *cmdp)
Sends a command without data phase.
- void `wspi_lld_send` (WSPIDriver *wspip, const `wspi_command_t` *cmdp, size_t n, const uint8_t *txbuf)
Sends a command with data over the WSPI bus.
- void `wspi_lld_receive` (WSPIDriver *wspip, const `wspi_command_t` *cmdp, size_t n, uint8_t *rxbuf)
Sends a command then receives data over the WSPI bus.
- void `wspi_lld_map_flash` (WSPIDriver *wspip, const `wspi_command_t` *cmdp, uint8_t **addrp)
Maps in memory space a WSPI flash device.
- void `wspi_lld_unmap_flash` (WSPIDriver *wspip)
Unmaps from memory space a WSPI flash device.

9.197.1 Detailed Description

PLATFORM WSPI subsystem low level driver header.

9.198 halconf.h File Reference

HAL configuration header.

```
#include "mcuconf.h"
```

Macros

- #define `_CHIBIOS_HAL_CONF_`
- #define `_CHIBIOS_HAL_CONF_VER_9_0_`
- #define `HAL_USE_PAL` TRUE
Enables the PAL subsystem.
- #define `HAL_USE_ADC` TRUE
Enables the ADC subsystem.
- #define `HAL_USE_CAN` TRUE
Enables the CAN subsystem.
- #define `HAL_USE_CRY` TRUE
Enables the cryptographic subsystem.
- #define `HAL_USE_DAC` TRUE
Enables the DAC subsystem.
- #define `HAL_USE_EFL` TRUE
Enables the EFlash subsystem.
- #define `HAL_USE_GPT` TRUE
Enables the GPT subsystem.

- `#define HAL_USE_I2C TRUE`
Enables the I2C subsystem.
- `#define HAL_USE_I2S TRUE`
Enables the I2S subsystem.
- `#define HAL_USE_ICU TRUE`
Enables the ICU subsystem.
- `#define HAL_USE_MAC TRUE`
Enables the MAC subsystem.
- `#define HAL_USE_MMC_SPI TRUE`
Enables the MMC_SPI subsystem.
- `#define HAL_USE_PWM TRUE`
Enables the PWM subsystem.
- `#define HAL_USE_RTC TRUE`
Enables the RTC subsystem.
- `#define HAL_USE_SDC TRUE`
Enables the SDC subsystem.
- `#define HAL_USE_SERIAL TRUE`
Enables the SERIAL subsystem.
- `#define HAL_USE_SERIAL_USB TRUE`
Enables the SERIAL over USB subsystem.
- `#define HAL_USE_SIO TRUE`
Enables the SIO subsystem.
- `#define HAL_USE_SPI TRUE`
Enables the SPI subsystem.
- `#define HAL_USE_TRNG TRUE`
Enables the TRNG subsystem.
- `#define HAL_USE_UART TRUE`
Enables the UART subsystem.
- `#define HAL_USE_USB TRUE`
Enables the USB subsystem.
- `#define HAL_USE_WDG TRUE`
Enables the WDG subsystem.
- `#define HAL_USE_WSPI TRUE`
Enables the WSPI subsystem.
- `#define PAL_USE_CALLBACKS FALSE`
Enables synchronous APIs.
- `#define PAL_USE_WAIT FALSE`
Enables synchronous APIs.
- `#define ADC_USE_WAIT TRUE`
Enables synchronous APIs.
- `#define ADC_USE_MUTUAL_EXCLUSION TRUE`
Enables the `adcAcquireBus()` and `adcReleaseBus()` APIs.
- `#define CAN_USE_SLEEP_MODE TRUE`
Sleep mode related APIs inclusion switch.
- `#define CAN_ENFORCE_USE_CALLBACKS FALSE`
Enforces the driver to use direct callbacks rather than OSAL events.
- `#define HAL_CRY_USE_FALLBACK FALSE`
Enables the SW fall-back of the cryptographic driver.
- `#define HAL_CRY_ENFORCE_FALLBACK FALSE`
Makes the driver forcibly use the fall-back implementations.
- `#define DAC_USE_WAIT TRUE`

- Enables synchronous APIs.*

 - #define `DAC_USE_MUTUAL_EXCLUSION` `TRUE`

Enables the `dacAcquireBus()` and `dacReleaseBus()` APIs.
- #define `I2C_ENABLE_SLAVE_MODE` `FALSE`

Slave mode API enable switch.
- #define `I2C_USE_MUTUAL_EXCLUSION` `TRUE`

Enables the mutual exclusion APIs on the I2C bus.
- #define `MAC_USE_ZERO_COPY` `TRUE`

Enables the zero-copy API.
- #define `MAC_USE_EVENTS` `TRUE`

Enables an event sources for incoming packets.
- #define `MMC_IDLE_TIMEOUT_MS` `1000`

Timeout before assuming a failure while waiting for card idle.
- #define `MMC_USE_MUTUAL_EXCLUSION` `TRUE`

Mutual exclusion on the SPI bus.
- #define `SDC_INIT_RETRY` `100`

Number of initialization attempts before rejecting the card.
- #define `SDC_MMC_SUPPORT` `TRUE`

Include support for MMC cards.
- #define `SDC_NICE_WAITING` `TRUE`

Delays insertions.
- #define `SDC_INIT_OCR_V20` `0x50FF8000U`

OCR initialization constant for V20 cards.
- #define `SDC_INIT_OCR` `0x80100000U`

OCR initialization constant for non-V20 cards.
- #define `SERIAL_DEFAULT_BITRATE` `38400`

Default bit rate.
- #define `SERIAL_BUFFERS_SIZE` `16`

Serial buffers size.
- #define `SIO_DEFAULT_BITRATE` `38400`

Default bit rate.
- #define `SIO_USE_SYNCHRONIZATION` `TRUE`

Support for thread synchronization API.
- #define `SERIAL_USB_BUFFERS_SIZE` `256`

Serial over USB buffers size.
- #define `SERIAL_USB_BUFFERS_NUMBER` `2`

Serial over USB number of buffers.
- #define `SPI_USE_WAIT` `TRUE`

Enables synchronous APIs.
- #define `SPI_USE_ASSERT_ON_ERROR` `TRUE`

Inserts an assertion on function errors before returning.
- #define `SPI_USE_MUTUAL_EXCLUSION` `TRUE`

Enables the `spiAcquireBus()` and `spiReleaseBus()` APIs.
- #define `SPI_SELECT_MODE` `SPI_SELECT_MODE_PAD`

Handling method for SPI CS line.
- #define `UART_USE_WAIT` `TRUE`

Enables synchronous APIs.
- #define `UART_USE_MUTUAL_EXCLUSION` `TRUE`

Enables the `uartAcquireBus()` and `uartReleaseBus()` APIs.
- #define `USB_USE_WAIT` `TRUE`

Enables synchronous APIs.

- `#define WSPi_USE_WAIT TRUE`
Enables synchronous APIs.
- `#define WSPi_USE_MUTUAL_EXCLUSION TRUE`
Enables the `wspiAcquireBus()` and `wspiReleaseBus()` APIs.

9.198.1 Detailed Description

HAL configuration header.

HAL configuration file, this file allows to enable or disable the various device drivers from your application. You may also use this file in order to override the device drivers default settings.

9.199 mcuconf.h File Reference

Macros

- `#define PLATFORM_MCUCONF`

9.199.1 Macro Definition Documentation

9.199.1.1 PLATFORM_MCUCONF

```
#define PLATFORM_MCUCONF
```

9.200 osal.c File Reference

OSAL module code.

```
#include "osal.h"
```

Functions

- void `osalInit` (void)
OSAL module initialization.
- void `osalSysHalt` (const char *reason)
System halt with error message.
- void `osalSysPolledDelayX` (rtcnt_t cycles)
Polled delay.
- void `osalOsTimerHandlerI` (void)
System timer handler.
- void `osalOsRescheduleS` (void)
Checks if a reschedule is required and performs it.
- `system_time_t` `osalOsGetSystemTimeX` (void)

- Current system time.*

 - void `osalThreadSleepS` (`sysinterval_t` time)

Suspends the invoking thread for the specified time.
- void `osalThreadSleep` (`sysinterval_t` time)

Suspends the invoking thread for the specified time.
- `msg_t` `osalThreadSuspendS` (`thread_reference_t` *trp)

Sends the current thread sleeping and sets a reference variable.
- `msg_t` `osalThreadSuspendTimeoutS` (`thread_reference_t` *trp, `sysinterval_t` timeout)

Sends the current thread sleeping and sets a reference variable.
- void `osalThreadResumeI` (`thread_reference_t` *trp, `msg_t` msg)

Wakes up a thread waiting on a thread reference object.
- void `osalThreadResumeS` (`thread_reference_t` *trp, `msg_t` msg)

Wakes up a thread waiting on a thread reference object.
- `msg_t` `osalThreadEnqueueTimeoutS` (`threads_queue_t` *tqp, `sysinterval_t` timeout)

Enqueues the caller thread.
- void `osalThreadDequeueNextI` (`threads_queue_t` *tqp, `msg_t` msg)

Dequeues and wakes up one thread from the queue, if any.
- void `osalThreadDequeueAllI` (`threads_queue_t` *tqp, `msg_t` msg)

Dequeues and wakes up all threads from the queue.
- void `osalEventBroadcastFlagsI` (`event_source_t` *esp, `eventflags_t` flags)

Add flags to an event source object.
- void `osalEventBroadcastFlags` (`event_source_t` *esp, `eventflags_t` flags)

Add flags to an event source object.
- void `osalEventSetCallback` (`event_source_t` *esp, `eventcallback_t` cb, void *param)

Event callback setup.
- void `osalMutexLock` (`mutex_t` *mp)

Locks the specified mutex.
- void `osalMutexUnlock` (`mutex_t` *mp)

Unlocks the specified mutex.

Variables

- const char * `osal_halt_msg`
- Pointer to a halt error message.*

9.200.1 Detailed Description

OSAL module code.

9.201 osal.h File Reference

OSAL module header.

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
```

Data Structures

- struct [event_source](#)
Events source object.
- struct [threads_queue_t](#)
Type of a thread queue.

Macros

- #define [OSAL_DBG_ENABLE_ASSERTS](#) FALSE
Enables OSAL assertions.
- #define [OSAL_DBG_ENABLE_CHECKS](#) FALSE
Enables OSAL functions parameters checks.

Common constants

- #define [FALSE](#) 0
- #define [TRUE](#) 1
- #define [OSAL_SUCCESS](#) false
- #define [OSAL_FAILED](#) true

Messages

- #define [MSG_OK](#) (msg_t)0
- #define [MSG_TIMEOUT](#) (msg_t)-1
- #define [MSG_RESET](#) (msg_t)-2

Special time constants

- #define [TIME_IMMEDIATE](#) ((sysinterval_t)0)
- #define [TIME_INFINITE](#) ((sysinterval_t)-1)

Systick modes.

- #define [OSAL_ST_MODE_NONE](#) 0
- #define [OSAL_ST_MODE_PERIODIC](#) 1
- #define [OSAL_ST_MODE_FREERUNNING](#) 2

Systick parameters.

- #define [OSAL_ST_RESOLUTION](#) 32
Size in bits of the `systick_t` type.
- #define [OSAL_ST_FREQUENCY](#) 1000
Required systick frequency or resolution.
- #define [OSAL_ST_MODE](#) [OSAL_ST_MODE_PERIODIC](#)
Systick mode required by the underlying OS.

IRQ-related constants

- #define [OSAL_IRQ_PRIORITY_LEVELS](#) 16U
Total priority levels.
- #define [OSAL_IRQ_MAXIMUM_PRIORITY](#) 0U
Highest IRQ priority for HAL drivers.

Debug related macros

- #define `osalDbgAssert(c, remark)`
Condition assertion.
- #define `osalDbgCheck(c)`
Function parameters check.
- #define `osalDbgCheckClassI()`
I-Class state check.
- #define `osalDbgCheckClassS()`
S-Class state check.

IRQ service routines wrappers

- #define `OSAL_IRQ_IS_VALID_PRIORITY(n)`
Priority level verification macro.
- #define `OSAL_IRQ_PROLOGUE()`
IRQ prologue code.
- #define `OSAL_IRQ_EPILOGUE()`
IRQ epilogue code.
- #define `OSAL_IRQ_HANDLER(id)`
IRQ handler function declaration.

Time conversion utilities

- #define `OSAL_S2I(secs)`
Seconds to time interval.
- #define `OSAL_MS2I(msecs)`
Milliseconds to time interval.
- #define `OSAL_US2I(usecs)`
Microseconds to time interval.
- #define `OSAL_I2S(interval)`
Time interval to seconds.
- #define `OSAL_I2MS(interval)`
Time interval to milliseconds.
- #define `OSAL_I2US(interval)`
Time interval to microseconds.

Time conversion utilities for the realtime counter

- #define `OSAL_S2RTC(freq, sec)`
Seconds to realtime counter.
- #define `OSAL_MS2RTC(freq, msec)`
Milliseconds to realtime counter.
- #define `OSAL_US2RTC(freq, usec)`
Microseconds to realtime counter.

Sleep macros using absolute time

- #define `osalThreadSleepSeconds(secs)`
Delays the invoking thread for the specified number of seconds.
- #define `osalThreadSleepMilliseconds(msecs)`
Delays the invoking thread for the specified number of milliseconds.
- #define `osalThreadSleepMicroseconds(usecs)`
Delays the invoking thread for the specified number of microseconds.

Typedefs

- typedef uint32_t [syssts_t](#)
Type of a system status word.
- typedef int32_t [msg_t](#)
Type of a message.
- typedef uint32_t [systime_t](#)
Type of system time counter.
- typedef uint32_t [sysinterval_t](#)
Type of system time interval.
- typedef uint64_t [time_conv_t](#)
Type of time conversion variable.
- typedef uint32_t [rtcnt_t](#)
Type of realtime counter.
- typedef void * [thread_reference_t](#)
Type of a thread reference.
- typedef uint32_t [eventflags_t](#)
Type of an event flags mask.
- typedef struct [event_source](#) [event_source_t](#)
Type of an event flags object.
- typedef void(* [eventcallback_t](#)) ([event_source_t](#) *esp)
Type of an event source callback.
- typedef uint32_t [mutex_t](#)
Type of a mutex.

Functions

- void [osalInit](#) (void)
OSAL module initialization.
- void [osalSysHalt](#) (const char *reason)
System halt with error message.
- void [osalSysPolledDelayX](#) ([rtcnt_t](#) cycles)
Polled delay.
- void [osalOsTimerHandlerI](#) (void)
System timer handler.
- void [osalOsRescheduleS](#) (void)
Checks if a reschedule is required and performs it.
- [systime_t](#) [osalOsGetSystemTimeX](#) (void)
Current system time.
- void [osalThreadSleepS](#) ([sysinterval_t](#) time)
Suspends the invoking thread for the specified time.
- void [osalThreadSleep](#) ([sysinterval_t](#) time)
Suspends the invoking thread for the specified time.
- [msg_t](#) [osalThreadSuspendS](#) ([thread_reference_t](#) *trp)
Sends the current thread sleeping and sets a reference variable.
- [msg_t](#) [osalThreadSuspendTimeoutS](#) ([thread_reference_t](#) *trp, [sysinterval_t](#) timeout)
Sends the current thread sleeping and sets a reference variable.
- void [osalThreadResumeI](#) ([thread_reference_t](#) *trp, [msg_t](#) msg)
Wakes up a thread waiting on a thread reference object.
- void [osalThreadResumeS](#) ([thread_reference_t](#) *trp, [msg_t](#) msg)

- Wakes up a thread waiting on a thread reference object.*

 - `msg_t osalThreadEnqueueTimeoutS (threads_queue_t *tqp, sysinterval_t timeout)`

Enqueues the caller thread.
- `void osalThreadDequeueNextI (threads_queue_t *tqp, msg_t msg)`

Dequeues and wakes up one thread from the queue, if any.
- `void osalThreadDequeueAllI (threads_queue_t *tqp, msg_t msg)`

Dequeues and wakes up all threads from the queue.
- `void osalEventBroadcastFlagsI (event_source_t *esp, eventflags_t flags)`

Add flags to an event source object.
- `void osalEventBroadcastFlags (event_source_t *esp, eventflags_t flags)`

Add flags to an event source object.
- `void osalEventSetCallback (event_source_t *esp, eventcallback_t cb, void *param)`

Event callback setup.
- `void osalMutexLock (mutex_t *mp)`

Locks the specified mutex.
- `void osalMutexUnlock (mutex_t *mp)`

Unlocks the specified mutex.
- `static void osalSysDisable (void)`

Disables interrupts globally.
- `static void osalSysEnable (void)`

Enables interrupts globally.
- `static void osalSysLock (void)`

Enters a critical zone from thread context.
- `static void osalSysUnlock (void)`

Leaves a critical zone from thread context.
- `static void osalSysLockFromISR (void)`

Enters a critical zone from ISR context.
- `static void osalSysUnlockFromISR (void)`

Leaves a critical zone from ISR context.
- `static syssts_t osalSysGetStatusAndLockX (void)`

Returns the execution status and enters a critical zone.
- `static void osalSysRestoreStatusX (syssts_t sts)`

Restores the specified execution status and leaves a critical zone.
- `static systime_t osalTimeAddX (systime_t systime, sysinterval_t interval)`

Adds an interval to a system time returning a system time.
- `static sysinterval_t osalTimeDiffX (systime_t start, systime_t end)`

Subtracts two system times returning an interval.
- `static bool osalTimeInRangeX (systime_t time, systime_t start, systime_t end)`

Checks if the specified time is within the specified time window.
- `static void osalThreadQueueObjectInit (threads_queue_t *tqp)`

Initializes a threads queue object.
- `static void osalEventObjectInit (event_source_t *esp)`

Initializes an event source object.
- `static void osalMutexObjectInit (mutex_t *mp)`

Initializes a `mutex_t` object.

Variables

- `const char * osal_halt_msg`

Pointer to a halt error message.

9.201.1 Detailed Description

OSAL module header.

Index

- [__CHIBIOS_HAL_CONF_](#)
 - Configuration, [45](#)
- [__CHIBIOS_HAL_CONF_VER_9_0_](#)
 - Configuration, [45](#)
- [__IOBUS_DATA](#)
 - PAL Driver, [317](#)
- [__CHIBIOS_HAL_](#)
 - HAL Driver, [59](#)
- [__bsio_ctl](#)
 - Buffered SIO Driver, [734](#)
- [__bsio_default_cb](#)
 - Buffered SIO Driver, [731](#)
- [__bsio_get](#)
 - Buffered SIO Driver, [732](#)
- [__bsio_gett](#)
 - Buffered SIO Driver, [733](#)
- [__bsio_onotify](#)
 - Buffered SIO Driver, [731](#)
- [__bsio_pop_data](#)
 - Buffered SIO Driver, [730](#)
- [__bsio_push_data](#)
 - Buffered SIO Driver, [730](#)
- [__bsio_put](#)
 - Buffered SIO Driver, [732](#)
- [__bsio_putt](#)
 - Buffered SIO Driver, [733](#)
- [__bsio_read](#)
 - Buffered SIO Driver, [732](#)
- [__bsio_readt](#)
 - Buffered SIO Driver, [734](#)
- [__bsio_write](#)
 - Buffered SIO Driver, [731](#)
- [__bsio_writet](#)
 - Buffered SIO Driver, [733](#)
- [__buffered_serial_ctl_impl](#)
 - Buffered Serial Class, [964](#)
- [__buffered_serial_data](#)
 - Buffered Serial Class, [958](#)
- [__buffered_serial_get_impl](#)
 - Buffered Serial Class, [962](#)
- [__buffered_serial_get_timeout_impl](#)
 - Buffered Serial Class, [963](#)
- [__buffered_serial_methods](#)
 - Buffered Serial Class, [958](#)
- [__buffered_serial_objinit_impl](#)
 - Buffered Serial Class, [960](#)
- [__buffered_serial_put_impl](#)
 - Buffered Serial Class, [962](#)
- [__buffered_serial_put_timeout_impl](#)
 - Buffered Serial Class, [963](#)
- [__buffered_serial_read_impl](#)
 - Buffered Serial Class, [961](#)
- [__buffered_serial_read_timeout_impl](#)
 - Buffered Serial Class, [964](#)
- [__buffered_serial_write_impl](#)
 - Buffered Serial Class, [961](#)
- [__buffered_serial_write_timeout_impl](#)
 - Buffered Serial Class, [963](#)
- [__buffered_sio_driver_data](#)
 - Buffered SIO Driver, [724](#)
- [__buffered_sio_driver_methods](#)
 - Buffered SIO Driver, [724](#)
- [__ctl](#)
 - SIO Driver, [491](#)
- [__get](#)
 - SIO Driver, [489](#)
- [__gett](#)
 - SIO Driver, [490](#)
- [__hal_xsnor_macronix_mx25_vmt](#)
 - Macronix MX25 SNOR Driver, [893](#)
- [__hal_xsnor_micron_n25q_vmt](#)
 - Micron N25Q Driver, [916](#)
- [__mac_callback](#)
 - MAC Driver, [287](#)
- [__mac_rx_wakeup](#)
 - MAC Driver, [288](#)
- [__mac_tx_wakeup](#)
 - MAC Driver, [288](#)
- [__mmc_driver_methods](#)
 - MMC over SPI Driver, [768](#)
- [__mx25_dispose_impl](#)
 - Macronix MX25 SNOR Driver, [882](#)
- [__mx25_init_impl](#)
 - Macronix MX25 SNOR Driver, [883](#)
- [__mx25_mmap_off_impl](#)
 - Macronix MX25 SNOR Driver, [888](#)
- [__mx25_mmap_on_impl](#)
 - Macronix MX25 SNOR Driver, [888](#)
- [__mx25_objinit_impl](#)
 - Macronix MX25 SNOR Driver, [881](#)
- [__mx25_program_impl](#)
 - Macronix MX25 SNOR Driver, [884](#)
- [__mx25_query_erase_impl](#)
 - Macronix MX25 SNOR Driver, [886](#)
- [__mx25_read_impl](#)
 - Macronix MX25 SNOR Driver, [883](#)
- [__mx25_start_erase_all_impl](#)
 - Macronix MX25 SNOR Driver, [885](#)

- __mx25_start_erase_sector_impl
 - Macronix MX25 SNOR Driver, [886](#)
- __mx25_verify_erase_impl
 - Macronix MX25 SNOR Driver, [887](#)
- __n25q_dispose_impl
 - Micron N25Q Driver, [903](#)
- __n25q_init_impl
 - Micron N25Q Driver, [904](#)
- __n25q_mmap_off_impl
 - Micron N25Q Driver, [909](#)
- __n25q_mmap_on_impl
 - Micron N25Q Driver, [909](#)
- __n25q_objinit_impl
 - Micron N25Q Driver, [902](#)
- __n25q_program_impl
 - Micron N25Q Driver, [905](#)
- __n25q_query_erase_impl
 - Micron N25Q Driver, [907](#)
- __n25q_read_impl
 - Micron N25Q Driver, [904](#)
- __n25q_start_erase_all_impl
 - Micron N25Q Driver, [906](#)
- __n25q_start_erase_sector_impl
 - Micron N25Q Driver, [907](#)
- __n25q_verify_erase_impl
 - Micron N25Q Driver, [908](#)
- __put
 - SIO Driver, [488](#)
- __putt
 - SIO Driver, [489](#)
- __read
 - SIO Driver, [488](#)
- __readt
 - SIO Driver, [491](#)
- __sio_callback
 - SIO Driver, [479](#)
- __sio_reloc_field
 - SIO Driver, [482](#)
- __sio_wakeup_errors
 - SIO Driver, [480](#)
- __sio_wakeup_rx
 - SIO Driver, [480](#)
- __sio_wakeup_rxidle
 - SIO Driver, [480](#)
- __sio_wakeup_tx
 - SIO Driver, [481](#)
- __sio_wakeup_txend
 - SIO Driver, [481](#)
- __spi_isr_complete_code
 - SPIv2 Driver, [551](#)
- __spi_isr_error_code
 - SPIv2 Driver, [553](#)
- __spi_isr_full_code
 - SPIv2 Driver, [553](#)
- __spi_isr_half_code
 - SPIv2 Driver, [552](#)
- __spi_wakeup_isr
 - SPIv2 Driver, [551](#)
- __write
 - SIO Driver, [487](#)
- __writet
 - SIO Driver, [490](#)
- __xsnor_bus_acquire
 - Base Serial NOR Driver class, [837](#), [839](#)
- __xsnor_bus_cmd
 - Base Serial NOR Driver class, [840](#)
- __xsnor_bus_cmd_addr
 - Base Serial NOR Driver class, [842](#)
- __xsnor_bus_cmd_addr_dummy_receive
 - Base Serial NOR Driver class, [845](#)
- __xsnor_bus_cmd_addr_receive
 - Base Serial NOR Driver class, [843](#)
- __xsnor_bus_cmd_addr_send
 - Base Serial NOR Driver class, [843](#)
- __xsnor_bus_cmd_dummy_receive
 - Base Serial NOR Driver class, [844](#)
- __xsnor_bus_cmd_receive
 - Base Serial NOR Driver class, [842](#)
- __xsnor_bus_cmd_send
 - Base Serial NOR Driver class, [841](#)
- __xsnor_bus_release
 - Base Serial NOR Driver class, [837](#), [840](#)
- __xsnor_dispose_impl
 - Base Serial NOR Driver class, [838](#)
- __xsnor_fls_acquire_exclusive_impl
 - Base Serial NOR Driver class, [856](#)
- __xsnor_fls_get_descriptor_impl
 - Base Serial NOR Driver class, [852](#)
- __xsnor_fls_program_impl
 - Base Serial NOR Driver class, [853](#)
- __xsnor_fls_query_erase_impl
 - Base Serial NOR Driver class, [855](#)
- __xsnor_fls_read_impl
 - Base Serial NOR Driver class, [852](#)
- __xsnor_fls_release_exclusive_impl
 - Base Serial NOR Driver class, [856](#)
- __xsnor_fls_start_erase_all_impl
 - Base Serial NOR Driver class, [854](#)
- __xsnor_fls_start_erase_sector_impl
 - Base Serial NOR Driver class, [854](#)
- __xsnor_fls_verify_erase_impl
 - Base Serial NOR Driver class, [855](#)
- __xsnor_objinit_impl
 - Base Serial NOR Driver class, [837](#)
- __xsnor_spi_cmd_addr
 - Base Serial NOR Driver class, [839](#)
- _adc_isr_error_code
 - ADC Driver, [75](#)
- _adc_isr_full_code
 - ADC Driver, [75](#)
- _adc_isr_half_code
 - ADC Driver, [74](#)
- _adc_reset_i
 - ADC Driver, [73](#)
- _adc_reset_s
 - ADC Driver, [73](#)

- `_adc_timeout_isr`
 - ADC Driver, [74](#)
- `_adc_wakeup_isr`
 - ADC Driver, [73](#)
- `_align1`
 - CANRxFrame, [1136](#)
 - CANTxFrame, [1139](#)
- `_base_asynchronous_channel_data`
 - Abstract I/O Channel Interface, [940](#)
- `_base_asynchronous_channel_methods`
 - Abstract I/O Channel Interface, [940](#)
- `_base_block_device_data`
 - Abstract I/O Block Device, [918](#)
- `_base_block_device_methods`
 - Abstract I/O Block Device, [918](#)
- `_base_buffered_stream_data`
 - Abstract Streams Interface, [929](#)
- `_base_buffered_stream_methods`
 - Abstract Streams Interface, [929](#)
- `_base_channel_data`
 - Abstract I/O Channel Interface, [933](#)
- `_base_channel_methods`
 - Abstract I/O Channel Interface, [933](#)
- `_base_flash_data`
 - Generic Flash Interface, [34](#)
- `_base_flash_methods`
 - Generic Flash Interface, [33](#)
- `_base_flash_methods_alone`
 - Generic Flash Interface, [33](#)
- `_base_object_data`
 - Base Object, [979](#)
- `_base_object_methods`
 - Base Object, [979](#)
- `_base_pers_storage_methods`
 - Abstract Persistent Storage Interface, [950](#)
- `_base_pers_storage_methods_alone`
 - Abstract Persistent Storage Interface, [950](#)
- `_base_persistent_storage_data`
 - Abstract Persistent Storage Interface, [950](#)
- `_base_sequential_stream_data`
 - Abstract Streams Interface, [927](#)
- `_base_sequential_stream_methods`
 - Abstract Streams Interface, [927](#)
- `_buffered_stream_adapter_data`
 - Buffered Streams Class, [965](#)
- `_can_error_isr`
 - CAN Driver, [95](#)
- `_can_rx_full_isr`
 - CAN Driver, [95](#)
- `_can_tx_empty_isr`
 - CAN Driver, [94](#)
- `_can_wakeup_isr`
 - CAN Driver, [95](#)
- `_ctl`
 - Serial Driver, [454](#)
 - Serial over USB Driver, [821](#)
- `_dac_isr_error_code`
 - DAC Driver, [189](#)
- `_dac_isr_full_code`
 - DAC Driver, [188](#)
- `_dac_isr_half_code`
 - DAC Driver, [187](#)
- `_dac_reset_i`
 - DAC Driver, [186](#)
- `_dac_reset_s`
 - DAC Driver, [186](#)
- `_dac_timeout_isr`
 - DAC Driver, [187](#)
- `_dac_wait_s`
 - DAC Driver, [185](#)
- `_dac_wakeup_isr`
 - DAC Driver, [187](#)
- `_efl_driver_data`
 - EFL Driver, [207](#)
- `_efl_flash_methods`
 - EFL Driver, [206](#)
- `_efl_flash_methods_alone`
 - EFL Driver, [206](#)
- `_file_stream_data`
 - Abstract Files Interface, [943](#)
- `_file_stream_methods`
 - Abstract Files Interface, [943](#)
- `_get`
 - Buffered Streams Class, [966](#)
 - Memory Streams Class, [954](#)
 - Serial Driver, [451](#)
 - Serial over USB Driver, [819](#)
- `_gett`
 - Serial Driver, [452](#)
 - Serial over USB Driver, [820](#)
- `_gpt_isr_invoke_cb`
 - GPT Driver, [221](#)
- `_i2c_wakeup_error_isr`
 - I2C Driver, [238](#)
- `_i2c_wakeup_isr`
 - I2C Driver, [238](#)
- `_i2s_isr_full_code`
 - I2S Driver, [257](#)
- `_i2s_isr_half_code`
 - I2S Driver, [256](#)
- `_icu_isr_invoke_overflow_cb`
 - ICU Driver, [272](#)
- `_icu_isr_invoke_period_cb`
 - ICU Driver, [271](#)
- `_icu_isr_invoke_width_cb`
 - ICU Driver, [271](#)
- `_memory_stream_data`
 - Memory Streams Class, [953](#)
- `_mmc_sd_block_device_data`
 - MMC/SD Block Device, [1024](#)
- `_mmc_sd_block_device_methods`
 - MMC/SD Block Device, [1024](#)
- `_mmc_sd_get_capacity`
 - MMC/SD Block Device, [1026](#)
- `_mmc_sd_get_capacity_ext`
 - MMC/SD Block Device, [1027](#)

- `_mmcscd_get_slice`
 - MMC/SD Block Device, [1026](#)
- `_mmcscd_unpack_csd_mmc`
 - MMC/SD Block Device, [1028](#)
- `_mmcscd_unpack_csd_v10`
 - MMC/SD Block Device, [1029](#)
- `_mmcscd_unpack_csd_v20`
 - MMC/SD Block Device, [1029](#)
- `_mmcscd_unpack_mmc_cid`
 - MMC/SD Block Device, [1028](#)
- `_mmcscd_unpack_sdc_cid`
 - MMC/SD Block Device, [1027](#)
- `_null_stream_data`
 - Null Streams Class, [955](#)
- `_pal_clear_event`
 - PAL Driver, [319](#), [320](#)
- `_pal_init_event`
 - PAL Driver, [318](#), [319](#)
- `_pal_isr_code`
 - PAL Driver, [320](#), [321](#)
- `_pal_ild_init`
 - PAL Driver, [357](#)
- `_pal_ild_setgroupmode`
 - PAL Driver, [357](#)
- `_put`
 - Buffered Streams Class, [966](#)
 - Memory Streams Class, [954](#)
 - Serial Driver, [451](#)
 - Serial over USB Driver, [818](#)
- `_putt`
 - Serial Driver, [452](#)
 - Serial over USB Driver, [819](#)
- `_read`
 - Serial Driver, [450](#)
 - Serial over USB Driver, [818](#)
- `_reads`
 - Buffered Streams Class, [966](#)
 - Memory Streams Class, [954](#)
- `_readt`
 - Serial Driver, [453](#)
 - Serial over USB Driver, [821](#)
- `_rtc_driver_methods`
 - RTC Driver, [389](#)
- `_sdc_driver_methods`
 - SDC Driver, [407](#)
- `_sdc_wait_for_transfer_state`
 - SDC Driver, [418](#)
- `_sdc_wait_for_transfer_state_internal`
 - SDC Driver, [417](#)
- `_sdc_wait_for_transfer_state_nocrc`
 - SDC Driver, [418](#)
- `_serial_driver_data`
 - Serial Driver, [449](#)
- `_serial_driver_methods`
 - Serial Driver, [440](#)
- `_serial_usb_driver_data`
 - Serial over USB Driver, [816](#)
- `_serial_usb_driver_methods`
 - Serial over USB Driver, [816](#)
- `_sio_driver_methods`
 - SIO Driver, [471](#)
- `_snor_flash_methods`
 - Serial NOR Flash Driver, [797](#)
- `_snor_flash_methods_alone`
 - Serial NOR Flash Driver, [796](#)
- `_spi_isr_code`
 - SPIv1 Driver, [521](#)
- `_spi_isr_full_code`
 - SPIv1 Driver, [522](#)
- `_spi_isr_half_code`
 - SPIv1 Driver, [522](#)
- `_spi_wakeup_isr`
 - SPIv1 Driver, [521](#)
- `_uart_rx_char_match_isr_code`
 - UART Driver, [613](#)
- `_uart_rx_complete_isr_code`
 - UART Driver, [611](#)
- `_uart_rx_error_isr_code`
 - UART Driver, [611](#)
- `_uart_rx_idle_code`
 - UART Driver, [612](#)
- `_uart_timeout_isr_code`
 - UART Driver, [613](#)
- `_uart_tx1_isr_code`
 - UART Driver, [609](#)
- `_uart_tx2_isr_code`
 - UART Driver, [610](#)
- `_uart_wakeup_rx_cm_isr`
 - UART Driver, [609](#)
- `_uart_wakeup_rx_complete_isr`
 - UART Driver, [608](#)
- `_uart_wakeup_rx_error_isr`
 - UART Driver, [608](#)
- `_uart_wakeup_rx_timeout_isr`
 - UART Driver, [609](#)
- `_uart_wakeup_tx1_isr`
 - UART Driver, [607](#)
- `_uart_wakeup_tx2_isr`
 - UART Driver, [607](#)
- `_unget`
 - Buffered Streams Class, [966](#)
 - Memory Streams Class, [954](#)
- `_usb_ep0in`
 - USB Driver, [675](#)
- `_usb_ep0out`
 - USB Driver, [676](#)
- `_usb_ep0setup`
 - USB Driver, [674](#)
- `_usb_isr_invoke_event_cb`
 - USB Driver, [654](#)
- `_usb_isr_invoke_in_cb`
 - USB Driver, [655](#)
- `_usb_isr_invoke_out_cb`
 - USB Driver, [656](#)
- `_usb_isr_invoke_setup_cb`
 - USB Driver, [655](#)

- [_usb_isr_invoke_sof_cb](#)
 - [USB Driver, 654](#)
- [_usb_reset](#)
 - [USB Driver, 672](#)
- [_usb_suspend](#)
 - [USB Driver, 673](#)
- [_usb_wakeup](#)
 - [USB Driver, 674](#)
- [_write](#)
 - [Serial Driver, 450](#)
 - [Serial over USB Driver, 817](#)
- [_writes](#)
 - [Buffered Streams Class, 966](#)
 - [Memory Streams Class, 954](#)
- [_writet](#)
 - [Serial Driver, 453](#)
 - [Serial over USB Driver, 820](#)
- [_wspi_error_code](#)
 - [WSPI Driver, 705](#)
- [_wspi_isr_code](#)
 - [WSPI Driver, 704](#)
- [_wspi_wakeup_isr](#)
 - [WSPI Driver, 703](#)
- [Abstract Files Interface, 941](#)
 - [_file_stream_data, 943](#)
 - [_file_stream_methods, 943](#)
 - [FILE_EOF, 943](#)
 - [FILE_ERROR, 943](#)
 - [FILE_OK, 943](#)
 - [fileoffset_t, 949](#)
 - [fileStreamClose, 946](#)
 - [fileStreamGet, 945](#)
 - [fileStreamGetError, 946](#)
 - [fileStreamGetPosition, 947](#)
 - [fileStreamGetSize, 947](#)
 - [fileStreamPut, 944](#)
 - [fileStreamRead, 944](#)
 - [fileStreamSetPosition, 948](#)
 - [fileStreamWrite, 943](#)
- [Abstract I/O Block Device, 917](#)
 - [_base_block_device_data, 918](#)
 - [_base_block_device_methods, 918](#)
 - [BLK_ACTIVE, 925](#)
 - [BLK_CONNECTING, 925](#)
 - [BLK_DISCONNECTING, 925](#)
 - [BLK_READING, 925](#)
 - [BLK_READY, 925](#)
 - [BLK_STOP, 925](#)
 - [BLK_SYNCING, 925](#)
 - [BLK_UNINIT, 925](#)
 - [BLK_WRITING, 925](#)
 - [blkConnect, 921](#)
 - [blkDisconnect, 921](#)
 - [blkGetDriverState, 919](#)
 - [blkGetInfo, 924](#)
 - [blkIsInserted, 920](#)
 - [blkIsTransferring, 919](#)
 - [blkIsWriteProtected, 920](#)
 - [blkRead, 922](#)
 - [blkstate_t, 924](#)
 - [blkSync, 923](#)
 - [blkWrite, 922](#)
- [Abstract I/O Channel Interface, 931](#)
 - [_base_asynchronous_channel_data, 940](#)
 - [_base_asynchronous_channel_methods, 940](#)
 - [_base_channel_data, 933](#)
 - [_base_channel_methods, 933](#)
 - [CHN_BREAK_DETECTED, 940](#)
 - [CHN_BUFFER_FULL_ERROR, 940](#)
 - [CHN_CONNECTED, 939](#)
 - [CHN_CTL_INVALID, 933](#)
 - [CHN_CTL_NOP, 933](#)
 - [CHN_CTL_TX_WAIT, 933](#)
 - [CHN_DISCONNECTED, 939](#)
 - [CHN_FRAMING_ERROR, 939](#)
 - [CHN_IDLE_DETECTED, 940](#)
 - [CHN_INPUT_AVAILABLE, 939](#)
 - [CHN_NO_ERROR, 938](#)
 - [CHN_NOISE_ERROR, 939](#)
 - [CHN_OUTPUT_EMPTY, 939](#)
 - [CHN_OVERRUN_ERROR, 940](#)
 - [CHN_PARITY_ERROR, 939](#)
 - [CHN_TRANSMISSION_END, 939](#)
 - [chnAddFlagsI, 941](#)
 - [chnControl, 938](#)
 - [chnGetEventSource, 940](#)
 - [chnGetTimeout, 934](#)
 - [chnPutTimeout, 933](#)
 - [chnRead, 936](#)
 - [chnReadTimeout, 936](#)
 - [chnWrite, 935](#)
 - [chnWriteTimeout, 935](#)
- [Abstract Persistent Storage Interface, 949](#)
 - [_base_pers_storage_methods, 950](#)
 - [_base_pers_storage_methods_alone, 950](#)
 - [_base_persistent_storage_data, 950](#)
 - [getBasePersistentStorage, 950](#)
 - [PS_ERROR_HW_FAILURE, 952](#)
 - [PS_ERROR_READ, 952](#)
 - [ps_error_t, 952](#)
 - [PS_ERROR_VERIFY, 952](#)
 - [PS_ERROR_WRITE, 952](#)
 - [PS_NO_ERROR, 952](#)
 - [ps_offset_t, 952](#)
 - [psGetStorageSize, 950](#)
 - [psRead, 951](#)
 - [psWrite, 951](#)
- [Abstract Streams Interface, 925](#)
 - [_base_buffered_stream_data, 929](#)
 - [_base_buffered_stream_methods, 929](#)
 - [_base_sequential_stream_data, 927](#)
 - [_base_sequential_stream_methods, 927](#)
 - [STM_OK, 926](#)
 - [STM_RESET, 926](#)
 - [STM_TIMEOUT, 926](#)
 - [streamGet, 929](#)

- streamPut, 928
- streamRead, 927
- streamUnget, 930
- streamWrite, 927
- accu_month_len
 - RTC Driver, 398
- acquire_exclusive
 - flash_interface_vmt, 1155
- active_status
 - USB Driver, 684
- ADC Driver, 68
 - _adc_isr_error_code, 75
 - _adc_isr_full_code, 75
 - _adc_isr_half_code, 74
 - _adc_reset_i, 73
 - _adc_reset_s, 73
 - _adc_timeout_isr, 74
 - _adc_wakeup_isr, 73
 - ADC_ACTIVE, 79
 - adc_channels_num_t, 79
 - ADC_COMPLETE, 79
 - ADC_ERR_AWD, 76
 - ADC_ERR_DMAFAILURE, 76
 - ADC_ERR_OVERFLOW, 76
 - ADC_ERROR, 79
 - adc_llc_config_fields, 77
 - adc_llc_configuration_group_fields, 77
 - adc_llc_driver_fields, 77
 - adc_llc_init, 87
 - adc_llc_start, 88
 - adc_llc_start_conversion, 88
 - adc_llc_stop, 88
 - adc_llc_stop_conversion, 89
 - ADC_READY, 79
 - ADC_STOP, 79
 - ADC_UNINIT, 79
 - ADC_USE_MUTUAL_EXCLUSION, 72
 - ADC_USE_WAIT, 72
 - adcAcquireBus, 86
 - adccallback_t, 78
 - ADCCConfig, 78
 - ADCCConversionGroup, 78
 - adcConvert, 85
 - ADCD1, 89
 - ADCDriver, 78
 - adcerror_t, 79
 - adcerrorcallback_t, 78
 - adclnit, 79
 - adclsBufferComplete, 72
 - adcObjectInit, 80
 - adcReleaseBus, 87
 - adcsample_t, 78
 - adcStart, 80
 - adcStartConversion, 82
 - adcStartConversionI, 83
 - adcstate_t, 79
 - adcStop, 81
 - adcStopConversion, 84
 - adcStopConversionI, 84
 - PLATFORM_ADC_USE_ADC1, 77
- ADC_ACTIVE
 - ADC Driver, 79
- adc_channels_num_t
 - ADC Driver, 79
- ADC_COMPLETE
 - ADC Driver, 79
- ADC_ERR_AWD
 - ADC Driver, 76
- ADC_ERR_DMAFAILURE
 - ADC Driver, 76
- ADC_ERR_OVERFLOW
 - ADC Driver, 76
- ADC_ERROR
 - ADC Driver, 79
- adc_llc_config_fields
 - ADC Driver, 77
- adc_llc_configuration_group_fields
 - ADC Driver, 77
- adc_llc_driver_fields
 - ADC Driver, 77
- adc_llc_init
 - ADC Driver, 87
- adc_llc_start
 - ADC Driver, 88
- adc_llc_start_conversion
 - ADC Driver, 88
- adc_llc_stop
 - ADC Driver, 88
- adc_llc_stop_conversion
 - ADC Driver, 89
- ADC_READY
 - ADC Driver, 79
- ADC_STOP
 - ADC Driver, 79
- ADC_UNINIT
 - ADC Driver, 79
- ADC_USE_MUTUAL_EXCLUSION
 - ADC Driver, 72
 - Configuration, 49
- ADC_USE_WAIT
 - ADC Driver, 72
 - Configuration, 49
- adcAcquireBus
 - ADC Driver, 86
- adccallback_t
 - ADC Driver, 78
- ADCCConfig
 - ADC Driver, 78
- ADCCConversionGroup
 - ADC Driver, 78
- adcConvert
 - ADC Driver, 85
- ADCD1
 - ADC Driver, 89
- ADCDriver
 - ADC Driver, 78

- adccerror_t
 - ADC Driver, [79](#)
- adccerrorcallback_t
 - ADC Driver, [78](#)
- adccInit
 - ADC Driver, [79](#)
- adccIsBufferComplete
 - ADC Driver, [72](#)
- adccObjectInit
 - ADC Driver, [80](#)
- adccReleaseBus
 - ADC Driver, [87](#)
- adccsample_t
 - ADC Driver, [78](#)
- adccStart
 - ADC Driver, [80](#)
- adccStartConversion
 - ADC Driver, [82](#)
- adccStartConversionI
 - ADC Driver, [83](#)
- adccstate_t
 - ADC Driver, [79](#)
- adccStop
 - ADC Driver, [81](#)
- adccStopConversion
 - ADC Driver, [84](#)
- adccStopConversionI
 - ADC Driver, [84](#)
- addr
 - wspi_command_t, [1331](#)
- address
 - flash_descriptor_t, [1149](#)
 - USBDriver, [1322](#)
- ADVERTISE_100BASE4
 - MII/RMII Header, [1063](#)
- ADVERTISE_100FULL
 - MII/RMII Header, [1063](#)
- ADVERTISE_100HALF
 - MII/RMII Header, [1062](#)
- ADVERTISE_10FULL
 - MII/RMII Header, [1062](#)
- ADVERTISE_10HALF
 - MII/RMII Header, [1062](#)
- ADVERTISE_ALL
 - MII/RMII Header, [1064](#)
- ADVERTISE_CSMA
 - MII/RMII Header, [1062](#)
- ADVERTISE_FULL
 - MII/RMII Header, [1064](#)
- ADVERTISE_LPACK
 - MII/RMII Header, [1063](#)
- ADVERTISE_NPAGE
 - MII/RMII Header, [1063](#)
- ADVERTISE_PAUSE_ASYM
 - MII/RMII Header, [1063](#)
- ADVERTISE_PAUSE_CAP
 - MII/RMII Header, [1063](#)
- ADVERTISE_RESV
 - MII/RMII Header, [1063](#)
- ADVERTISE_RFAULT
 - MII/RMII Header, [1063](#)
- ADVERTISE_SLCT
 - MII/RMII Header, [1062](#)
- ALIGNED_DHDR_SIZE
 - Managed Flash Storage Driver, [741](#)
- ALIGNED_REC_SIZE
 - Managed Flash Storage Driver, [741](#)
- ALIGNED_SIZEOF
 - Managed Flash Storage Driver, [741](#)
- alt
 - wspi_command_t, [1331](#)
- arg
 - hal_mac_driver, [1185](#)
 - hal_sio_driver, [1193](#)
 - palevent_t, [1267](#)
- attributes
 - flash_descriptor_t, [1149](#)
- bank0_sectors
 - MFSCConfig, [1249](#)
- bank0_start
 - MFSCConfig, [1249](#)
- bank1_sectors
 - MFSCConfig, [1249](#)
- bank1_start
 - MFSCConfig, [1249](#)
- bank_size
 - MFSCConfig, [1249](#)
- Base Flash Interface, [857](#)
 - flash_interface_i, [858](#)
 - flsAcquireExclusive, [862](#)
 - flsGetDescriptor, [858](#)
 - flsProgram, [859](#)
 - flsQueryErase, [861](#)
 - flsRead, [859](#)
 - flsReleaseExclusive, [862](#)
 - flsStartEraseAll, [860](#)
 - flsStartEraseSector, [860](#)
 - flsVerifyErase, [861](#)
- Base Object, [978](#)
 - _base_object_data, [979](#)
 - _base_object_methods, [979](#)
 - objGetInstance, [979](#)
- Base Serial NOR Driver class, [831](#)
 - __xsnor_bus_acquire, [837](#), [839](#)
 - __xsnor_bus_cmd, [840](#)
 - __xsnor_bus_cmd_addr, [842](#)
 - __xsnor_bus_cmd_addr_dummy_receive, [845](#)
 - __xsnor_bus_cmd_addr_receive, [843](#)
 - __xsnor_bus_cmd_addr_send, [843](#)
 - __xsnor_bus_cmd_dummy_receive, [844](#)
 - __xsnor_bus_cmd_receive, [842](#)
 - __xsnor_bus_cmd_send, [841](#)
 - __xsnor_bus_release, [837](#), [840](#)
 - __xsnor_dispose_impl, [838](#)
 - __xsnor_fls_acquire_exclusive_impl, [856](#)
 - __xsnor_fls_get_descriptor_impl, [852](#)

- [__xsnor_fls_program_impl, 853](#)
- [__xsnor_fls_query_erase_impl, 855](#)
- [__xsnor_fls_read_impl, 852](#)
- [__xsnor_fls_release_exclusive_impl, 856](#)
- [__xsnor_fls_start_erase_all_impl, 854](#)
- [__xsnor_fls_start_erase_sector_impl, 854](#)
- [__xsnor_fls_verify_erase_impl, 855](#)
- [__xsnor_objinit_impl, 837](#)
- [__xsnor_spi_cmd_addr, 839](#)
- [FLASH_ATTR_SPI_4BYTES_ADDR_HINT, 836](#)
- [hal_xsnor_base_c, 837](#)
- [XSNOR_BUFFER_SIZE, 836](#)
- [xsnor_buffers_t, 837](#)
- [XSNOR_BUS_MODE_SPI, 835](#)
- [XSNOR_BUS_MODE_WSPI_1LINE, 835](#)
- [XSNOR_BUS_MODE_WSPI_2LINES, 835](#)
- [XSNOR_BUS_MODE_WSPI_4LINES, 836](#)
- [XSNOR_BUS_MODE_WSPI_8LINES, 836](#)
- [xsnor_commands_t, 837](#)
- [xsnor_config_t, 837](#)
- [xsnor_device_init, 849](#)
- [xsnor_device_mmap_off, 852](#)
- [xsnor_device_mmap_on, 852](#)
- [xsnor_device_program, 849](#)
- [xsnor_device_query_erase, 851](#)
- [xsnor_device_read, 849](#)
- [xsnor_device_start_erase_all, 850](#)
- [xsnor_device_start_erase_sector, 850](#)
- [xsnor_device_verify_erase, 851](#)
- [XSNOR_SHARED_BUS, 836](#)
- [XSNOR_USE_BOTH, 836](#)
- [XSNOR_USE_SPI, 836](#)
- [XSNOR_USE_WSPI, 836](#)
- [xsnorMemoryMap, 847](#)
- [xsnorMemoryUnmap, 848](#)
- [xsnorStart, 846](#)
- [xsnorStop, 847](#)
- [BaseAsynchronousChannel, 1105](#)
 - [vmt, 1108](#)
- [BaseAsynchronousChannelVMT, 1108](#)
- [BaseBlockDevice, 1109](#)
 - [vmt, 1111](#)
- [BaseBlockDeviceVMT, 1112](#)
- [BaseBufferedStream, 1113](#)
 - [vmt, 1115](#)
- [BaseBufferedStreamVMT, 1115](#)
- [BaseChannel, 1116](#)
 - [vmt, 1118](#)
- [BaseChannelVMT, 1118](#)
- [BaseFlash, 1119](#)
 - [vmt, 1121](#)
- [BaseFlashVMT, 1122](#)
- [BaseObject, 1122](#)
 - [vmt, 1124](#)
- [BaseObjectVMT, 1124](#)
- [BasePersistentStorage, 1124](#)
 - [vmt, 1125](#)
- [BasePersistentStorageVMT, 1126](#)
- [BaseSequentialStream, 1126](#)
 - [vmt, 1127](#)
- [BaseSequentialStreamVMT, 1128](#)
- [bCharFormat](#)
 - [cdc_linecoding_t, 1140](#)
- [bcounter](#)
 - [io_buffers_queue, 1231](#)
- [bDataBits](#)
 - [cdc_linecoding_t, 1141](#)
- [bhdr](#)
 - [mfs_nocache_buffer, 1245](#)
- [bitsize_t](#)
 - [Crypto Driver, 118](#)
- [BLK_ACTIVE](#)
 - [Abstract I/O Block Device, 925](#)
- [BLK_CONNECTING](#)
 - [Abstract I/O Block Device, 925](#)
- [BLK_DISCONNECTING](#)
 - [Abstract I/O Block Device, 925](#)
- [blk_num](#)
 - [BlockDeviceInfo, 1129](#)
- [BLK_READING](#)
 - [Abstract I/O Block Device, 925](#)
- [BLK_READY](#)
 - [Abstract I/O Block Device, 925](#)
- [blk_size](#)
 - [BlockDeviceInfo, 1129](#)
- [BLK_STOP](#)
 - [Abstract I/O Block Device, 925](#)
- [BLK_SYNCING](#)
 - [Abstract I/O Block Device, 925](#)
- [BLK_UNINIT](#)
 - [Abstract I/O Block Device, 925](#)
- [BLK_WRITING](#)
 - [Abstract I/O Block Device, 925](#)
- [blkConnect](#)
 - [Abstract I/O Block Device, 921](#)
- [blkDisconnect](#)
 - [Abstract I/O Block Device, 921](#)
- [blkGetDriverState](#)
 - [Abstract I/O Block Device, 919](#)
- [blkGetInfo](#)
 - [Abstract I/O Block Device, 924](#)
- [blkIsInserted](#)
 - [Abstract I/O Block Device, 920](#)
- [blkIsTransferring](#)
 - [Abstract I/O Block Device, 919](#)
- [blkIsWriteProtected](#)
 - [Abstract I/O Block Device, 920](#)
- [blkRead](#)
 - [Abstract I/O Block Device, 922](#)
- [blkstate_t](#)
 - [Abstract I/O Block Device, 924](#)
- [blkSync](#)
 - [Abstract I/O Block Device, 923](#)
- [blkWrite](#)
 - [Abstract I/O Block Device, 922](#)
- [block_addresses](#)

- [mmc_spi_driver_t](#), [1257](#)
- [BlockDeviceInfo](#), [1129](#)
 - [blk_num](#), [1129](#)
 - [blk_size](#), [1129](#)
- [Blocks](#), [916](#)
- [BMCR_ANENABLE](#)
 - [MII/RMII Header](#), [1060](#)
- [BMCR_ANRESTART](#)
 - [MII/RMII Header](#), [1059](#)
- [BMCR_CTST](#)
 - [MII/RMII Header](#), [1059](#)
- [BMCR_FULLDPLX](#)
 - [MII/RMII Header](#), [1059](#)
- [BMCR_ISOLATE](#)
 - [MII/RMII Header](#), [1060](#)
- [BMCR_LOOPBACK](#)
 - [MII/RMII Header](#), [1060](#)
- [BMCR_PDOWN](#)
 - [MII/RMII Header](#), [1060](#)
- [BMCR_RESET](#)
 - [MII/RMII Header](#), [1060](#)
- [BMCR_RESV](#)
 - [MII/RMII Header](#), [1059](#)
- [BMCR_SPEED100](#)
 - [MII/RMII Header](#), [1060](#)
- [BMSR_100BASE4](#)
 - [MII/RMII Header](#), [1062](#)
- [BMSR_100FULL](#)
 - [MII/RMII Header](#), [1062](#)
- [BMSR_100HALF](#)
 - [MII/RMII Header](#), [1062](#)
- [BMSR_10FULL](#)
 - [MII/RMII Header](#), [1061](#)
- [BMSR_10HALF](#)
 - [MII/RMII Header](#), [1061](#)
- [BMSR_ANEGCAPABLE](#)
 - [MII/RMII Header](#), [1061](#)
- [BMSR_ANEGCOMPLETE](#)
 - [MII/RMII Header](#), [1061](#)
- [BMSR_ERCAP](#)
 - [MII/RMII Header](#), [1060](#)
- [BMSR_JCD](#)
 - [MII/RMII Header](#), [1060](#)
- [BMSR_LSTATUS](#)
 - [MII/RMII Header](#), [1061](#)
- [BMSR_MFPRESUPPCAP](#)
 - [MII/RMII Header](#), [1061](#)
- [BMSR_RESV](#)
 - [MII/RMII Header](#), [1061](#)
- [BMSR_RFAULT](#)
 - [MII/RMII Header](#), [1061](#)
- [bn](#)
 - [io_buffers_queue](#), [1232](#)
- [bParityType](#)
 - [cdc_linecoding_t](#), [1140](#)
- [BQ_BUFFER_SIZE](#)
 - [I/O Buffers Queues](#), [982](#)
- [bqGetLinkX](#)
 - [I/O Buffers Queues](#), [983](#)
- [bqlsSuspendedX](#)
 - [I/O Buffers Queues](#), [984](#)
- [bqnotify_t](#)
 - [I/O Buffers Queues](#), [988](#)
- [bqResumeX](#)
 - [I/O Buffers Queues](#), [985](#)
- [bqSetLinkX](#)
 - [I/O Buffers Queues](#), [984](#)
- [bqSizeX](#)
 - [I/O Buffers Queues](#), [983](#)
- [bqSpacel](#)
 - [I/O Buffers Queues](#), [983](#)
- [bqSuspendI](#)
 - [I/O Buffers Queues](#), [985](#)
- [brdptr](#)
 - [io_buffers_queue](#), [1231](#)
- [BS_READY](#)
 - [Buffered Serial Class](#), [959](#)
- [BS_STOP](#)
 - [Buffered Serial Class](#), [959](#)
- [BS_UNINIT](#)
 - [Buffered Serial Class](#), [959](#)
- [bsaObjectInit](#)
 - [Buffered Streams Class](#), [966](#)
- [bsIncomingDataI](#)
 - [Buffered Serial Class](#), [959](#)
- [bsioAsynchronousRead](#)
 - [Buffered SIO Driver](#), [729](#)
- [bsioAsynchronousWrite](#)
 - [Buffered SIO Driver](#), [727](#)
- [bsioGet](#)
 - [Buffered SIO Driver](#), [725](#)
- [bsioGetI](#)
 - [Buffered SIO Driver](#), [725](#)
- [bsioGetTimeout](#)
 - [Buffered SIO Driver](#), [726](#)
- [bsioObjectInit](#)
 - [Buffered SIO Driver](#), [734](#)
- [bsioPut](#)
 - [Buffered SIO Driver](#), [724](#)
- [bsioPutI](#)
 - [Buffered SIO Driver](#), [724](#)
- [bsioPutTimeout](#)
 - [Buffered SIO Driver](#), [725](#)
- [bsioRead](#)
 - [Buffered SIO Driver](#), [728](#)
- [bsioReadI](#)
 - [Buffered SIO Driver](#), [728](#)
- [bsioReadTimeout](#)
 - [Buffered SIO Driver](#), [729](#)
- [bsioStart](#)
 - [Buffered SIO Driver](#), [735](#)
- [bsioStop](#)
 - [Buffered SIO Driver](#), [736](#)
- [bsioWrite](#)
 - [Buffered SIO Driver](#), [726](#)
- [bsioWriteI](#)

- Buffered SIO Driver, 726
- bsioWriteTimeout
 - Buffered SIO Driver, 727
- bsize
 - io_buffers_queue, 1232
- bsRequestDataI
 - Buffered Serial Class, 959
- bsstate_t
 - Buffered Serial Class, 958
- btop
 - io_buffers_queue, 1231
- buf
 - SDCDriver, 1280
 - snor_nocache_buffer, 1293
- buffer
 - mmc_spi_driver_t, 1257
- Buffered Serial Class, 957
 - __buffered_serial_ctl_impl, 964
 - __buffered_serial_data, 958
 - __buffered_serial_get_impl, 962
 - __buffered_serial_get_timeout_impl, 963
 - __buffered_serial_methods, 958
 - __buffered_serial_objinit_impl, 960
 - __buffered_serial_put_impl, 962
 - __buffered_serial_put_timeout_impl, 963
 - __buffered_serial_read_impl, 961
 - __buffered_serial_read_timeout_impl, 964
 - __buffered_serial_write_impl, 961
 - __buffered_serial_write_timeout_impl, 963
 - BS_READY, 959
 - BS_STOP, 959
 - BS_UNINIT, 959
 - bsIncomingDataI, 959
 - bsRequestDataI, 959
 - bsstate_t, 958
 - BufferedSerial, 958
- Buffered SIO Driver, 722
 - __bsio_ctl, 734
 - __bsio_default_cb, 731
 - __bsio_get, 732
 - __bsio_gett, 733
 - __bsio_onotify, 731
 - __bsio_pop_data, 730
 - __bsio_push_data, 730
 - __bsio_put, 732
 - __bsio_putt, 733
 - __bsio_read, 732
 - __bsio_readt, 734
 - __bsio_write, 731
 - __bsio_writet, 733
 - __buffered_sio_driver_data, 724
 - __buffered_sio_driver_methods, 724
 - bsioAsynchronousRead, 729
 - bsioAsynchronousWrite, 727
 - bsioGet, 725
 - bsioGetI, 725
 - bsioGetTimeout, 726
 - bsioObjectInit, 734
 - bsioPut, 724
 - bsioPutI, 724
 - bsioPutTimeout, 725
 - bsioRead, 728
 - bsioReadI, 728
 - bsioReadTimeout, 729
 - bsioStart, 735
 - bsioStop, 736
 - bsioWrite, 726
 - bsioWritel, 726
 - bsioWriteTimeout, 727
 - BufferedSIOConfig, 730
 - BufferedSIODriver, 730
 - vmt, 737
- Buffered Streams Class, 965
 - __buffered_stream_adapter_data, 965
 - __get, 966
 - __put, 966
 - __reads, 966
 - __unget, 966
 - __writes, 966
 - bsaObjectInit, 966
 - BUFSTREAM_BUFFER_SIZE, 965
 - vmt, 967
- BufferedSerial
 - Buffered Serial Class, 958
- BufferedSerialVMT, 1130
- BufferedSIOConfig
 - Buffered SIO Driver, 730
- BufferedSIODriver
 - Buffered SIO Driver, 730
- BufferedSIODriverVMT, 1131
- BufferedStreamAdapter, 1133
 - vmt, 1133
- BufferedStreamAdapterVMT, 1134
- buffers
 - io_buffers_queue, 1232
 - xsnor_config, 1340
- BUFFERS_CHUNKS_SIZE
 - I/O Buffers Queues, 982
- BUFSTREAM_BUFFER_SIZE
 - Buffered Streams Class, 965
- bufstreams.c, 1443
- bufstreams.h, 1443
- bulk_in
 - SerialUSBConfig, 1285
- bulk_out
 - SerialUSBConfig, 1285
- bus
 - xsnor_config, 1340
- bus_acquire
 - Serial NOR Flash Driver, 797, 800
- bus_cmd
 - Serial NOR Flash Driver, 802
- bus_cmd_addr
 - Serial NOR Flash Driver, 805
- bus_cmd_addr_dummy_receive
 - Serial NOR Flash Driver, 808

- bus_cmd_addr_receive
 - Serial NOR Flash Driver, [806](#)
- bus_cmd_addr_send
 - Serial NOR Flash Driver, [805](#)
- bus_cmd_dummy_receive
 - Serial NOR Flash Driver, [807](#)
- bus_cmd_receive
 - Serial NOR Flash Driver, [804](#)
- bus_cmd_send
 - Serial NOR Flash Driver, [803](#)
- bus_release
 - Serial NOR Flash Driver, [797](#), [800](#)
- bus_stop
 - Serial NOR Flash Driver, [801](#)
- bus_type
 - xsnor_config, [1340](#)
- bus_width
 - SDCConfig, [1278](#)
- buscfg
 - SNORConfig, [1294](#)
- BUSConfig
 - Serial NOR Flash Driver, [796](#)
- BUSDriver
 - Serial NOR Flash Driver, [796](#)
- busp
 - SNORConfig, [1294](#)
- bwrptr
 - io_buffers_queue, [1231](#)
- c_size
 - unpacked_mmc_csd_t, [1304](#)
 - unpacked_sdc_csd_10_t, [1310](#)
 - unpacked_sdc_csd_20_t, [1314](#)
- c_size_mult
 - unpacked_mmc_csd_t, [1304](#)
 - unpacked_sdc_csd_10_t, [1311](#)
- callback
 - GPTConfig, [1157](#)
 - PWMChannelConfig, [1268](#)
 - PWMConfig, [1270](#)
- CAN Driver, [89](#)
 - _can_error_isr, [95](#)
 - _can_rx_full_isr, [95](#)
 - _can_tx_empty_isr, [94](#)
 - _can_wakeup_isr, [95](#)
 - CAN_ANY_MAILBOX, [93](#)
 - CAN_BUS_OFF_ERROR, [93](#)
 - can_callback_t, [96](#)
 - CAN_ENFORCE_USE_CALLBACKS, [94](#)
 - CAN_FRAMING_ERROR, [93](#)
 - CAN_LIMIT_ERROR, [93](#)
 - CAN_LIMIT_WARNING, [93](#)
 - can_ild_abort, [108](#)
 - can_ild_init, [106](#)
 - can_ild_is_rx_nonempty, [108](#)
 - can_ild_is_tx_empty, [107](#)
 - can_ild_receive, [108](#)
 - can_ild_sleep, [109](#)
 - can_ild_start, [106](#)
 - can_ild_stop, [107](#)
 - can_ild_transmit, [107](#)
 - can_ild_wakeup, [109](#)
 - CAN_MAILBOX_TO_MASK, [94](#)
 - CAN_OVERFLOW_ERROR, [93](#)
 - CAN_READY, [97](#)
 - CAN_RX_MAILBOXES, [96](#)
 - CAN_SLEEP, [97](#)
 - CAN_STARTING, [97](#)
 - CAN_STOP, [97](#)
 - CAN_STOPPING, [97](#)
 - CAN_TX_MAILBOXES, [95](#)
 - CAN_UNINIT, [97](#)
 - CAN_USE_SLEEP_MODE, [93](#)
 - CANConfig, [97](#)
 - CAND1, [110](#)
 - CANDriver, [96](#)
 - canInit, [97](#)
 - canmbx_t, [96](#)
 - canObjectInit, [97](#)
 - canReceive, [94](#)
 - canReceiveTimeout, [103](#)
 - canSleep, [104](#)
 - canStart, [98](#)
 - canstate_t, [97](#)
 - canStop, [99](#)
 - canTransmit, [94](#)
 - canTransmitTimeout, [102](#)
 - canTryAbortX, [101](#)
 - canTryReceiveI, [100](#)
 - canTryTransmitI, [100](#)
 - canWakeup, [105](#)
 - PLATFORM_CAN_USE_CAN1, [96](#)
- CAN_ANY_MAILBOX
 - CAN Driver, [93](#)
- CAN_BUS_OFF_ERROR
 - CAN Driver, [93](#)
- can_callback_t
 - CAN Driver, [96](#)
- CAN_ENFORCE_USE_CALLBACKS
 - CAN Driver, [94](#)
 - Configuration, [49](#)
- CAN_FRAMING_ERROR
 - CAN Driver, [93](#)
- CAN_LIMIT_ERROR
 - CAN Driver, [93](#)
- CAN_LIMIT_WARNING
 - CAN Driver, [93](#)
- can_ild_abort
 - CAN Driver, [108](#)
- can_ild_init
 - CAN Driver, [106](#)
- can_ild_is_rx_nonempty
 - CAN Driver, [108](#)
- can_ild_is_tx_empty
 - CAN Driver, [107](#)
- can_ild_receive
 - CAN Driver, [108](#)

- can_ild_sleep
 - CAN Driver, [109](#)
- can_ild_start
 - CAN Driver, [106](#)
- can_ild_stop
 - CAN Driver, [107](#)
- can_ild_transmit
 - CAN Driver, [107](#)
- can_ild_wakeup
 - CAN Driver, [109](#)
- CAN_MAILBOX_TO_MASK
 - CAN Driver, [94](#)
- CAN_OVERFLOW_ERROR
 - CAN Driver, [93](#)
- CAN_READY
 - CAN Driver, [97](#)
- CAN_RX_MAILBOXES
 - CAN Driver, [96](#)
- CAN_SLEEP
 - CAN Driver, [97](#)
- CAN_STARTING
 - CAN Driver, [97](#)
- CAN_STOP
 - CAN Driver, [97](#)
- CAN_STOPPING
 - CAN Driver, [97](#)
- CAN_TX_MAILBOXES
 - CAN Driver, [95](#)
- CAN_UNINIT
 - CAN Driver, [97](#)
- CAN_USE_SLEEP_MODE
 - CAN Driver, [93](#)
 - Configuration, [49](#)
- CANConfig
 - CAN Driver, [97](#)
- CAND1
 - CAN Driver, [110](#)
- CANDriver
 - CAN Driver, [96](#)
- canInit
 - CAN Driver, [97](#)
- canmbx_t
 - CAN Driver, [96](#)
- canObjectInit
 - CAN Driver, [97](#)
- canReceive
 - CAN Driver, [94](#)
- canReceiveTimeout
 - CAN Driver, [103](#)
- CANRxFrame, [1134](#)
 - _align1, [1136](#)
 - data16, [1136](#)
 - data32, [1137](#)
 - data8, [1136](#)
 - DLC, [1135](#)
 - EID, [1136](#)
 - FMI, [1135](#)
 - IDE, [1136](#)
 - RTR, [1136](#)
 - SID, [1136](#)
 - TIME, [1135](#)
- canSleep
 - CAN Driver, [104](#)
- canStart
 - CAN Driver, [98](#)
- canstate_t
 - CAN Driver, [97](#)
- canStop
 - CAN Driver, [99](#)
- canTransmit
 - CAN Driver, [94](#)
- canTransmitTimeout
 - CAN Driver, [102](#)
- canTryAbortX
 - CAN Driver, [101](#)
- canTryReceiveI
 - CAN Driver, [100](#)
- canTryTransmitI
 - CAN Driver, [100](#)
- CANTxFrame, [1137](#)
 - _align1, [1139](#)
 - data16, [1139](#)
 - data32, [1139](#)
 - data8, [1139](#)
 - DLC, [1138](#)
 - EID, [1139](#)
 - IDE, [1138](#)
 - RTR, [1138](#)
 - SID, [1139](#)
- canWakeup
 - CAN Driver, [105](#)
- cardmode
 - SDCDriver, [1280](#)
- cb
 - event_source, [1144](#)
 - hal_mac_driver, [1184](#)
 - hal_sio_driver, [1192](#)
 - palevent_t, [1267](#)
- CC_ALIGN_CODE
 - Compiler portability., [24](#)
- CC_ALIGN_DATA
 - Compiler portability., [24](#)
- CC_FORCE_INLINE
 - Compiler portability., [25](#)
- CC_LIKELY
 - Compiler portability., [26](#)
- CC_NO_INLINE
 - Compiler portability., [25](#)
- CC_NO_RETURN
 - Compiler portability., [25](#)
- CC_PACK
 - Compiler portability., [25](#)
- CC_ROMCONST
 - Compiler portability., [26](#)
- CC_SECTION
 - Compiler portability., [24](#)

- CC_UNLIKELY
 - Compiler portability., [26](#)
- CC_USED
 - Compiler portability., [24](#)
- CC_WEAK
 - Compiler portability., [24](#)
- ccc
 - unpacked_mmc_csd_t, [1303](#)
 - unpacked_sdc_csd_10_t, [1310](#)
 - unpacked_sdc_csd_20_t, [1313](#)
- ccportab.h, [1341](#)
- CDC_ABSTRACT_CONTROL_MANAGEMENT
 - USB CDC Header, [1072](#)
- CDC_ABSTRACT_CONTROL_MODEL
 - USB CDC Header, [1071](#)
- CDC_CALL_MANAGEMENT
 - USB CDC Header, [1072](#)
- CDC_CLEAR_COMM_FEATURE
 - USB CDC Header, [1070](#)
- CDC_COMMUNICATION_INTERFACE_CLASS
 - USB CDC Header, [1071](#)
- CDC_CS_INTERFACE
 - USB CDC Header, [1071](#)
- CDC_DATA_INTERFACE_CLASS
 - USB CDC Header, [1071](#)
- CDC_GET_COMM_FEATURE
 - USB CDC Header, [1070](#)
- CDC_GET_ENCAPSULATED_RESPONSE
 - USB CDC Header, [1069](#)
- CDC_GET_LINE_CODING
 - USB CDC Header, [1070](#)
- CDC_GET_OPERATION_PARMS
 - USB CDC Header, [1071](#)
- CDC_GET_RINGER_PARMS
 - USB CDC Header, [1071](#)
- CDC_HEADER
 - USB CDC Header, [1072](#)
- cdc_linecoding_t, [1140](#)
 - bCharFormat, [1140](#)
 - bDataBits, [1141](#)
 - bParityType, [1140](#)
 - dwDTERate, [1140](#)
- CDC_PULSE_SETUP
 - USB CDC Header, [1070](#)
- CDC_RING_AUX_JACK
 - USB CDC Header, [1070](#)
- CDC_SEND_BREAK
 - USB CDC Header, [1071](#)
- CDC_SEND_ENCAPSULATED_COMMAND
 - USB CDC Header, [1069](#)
- CDC_SEND_PULSE
 - USB CDC Header, [1070](#)
- CDC_SET_AUX_LINE_STATE
 - USB CDC Header, [1070](#)
- CDC_SET_COMM_FEATURE
 - USB CDC Header, [1069](#)
- CDC_SET_CONTROL_LINE_STATE
 - USB CDC Header, [1071](#)
- CDC_SET_HOOK_STATE
 - USB CDC Header, [1070](#)
- CDC_SET_LINE_CODING
 - USB CDC Header, [1070](#)
- CDC_SET_OPERATION_PARMS
 - USB CDC Header, [1071](#)
- CDC_SET_PULSE_TIME
 - USB CDC Header, [1070](#)
- CDC_SET_RINGER_PARMS
 - USB CDC Header, [1071](#)
- CDC_UNION
 - USB CDC Header, [1072](#)
- cfg
 - wsapi_command_t, [1331](#)
 - xsnor_bus_spi, [1336](#)
 - xsnor_bus_wsapi, [1337](#)
- CFG_C16_8DTR
 - Macronix MX25 SNOR Driver, [878](#)
- CFG_C16_8STR
 - Macronix MX25 SNOR Driver, [878](#)
- CFG_C16_A32_8DTR
 - Macronix MX25 SNOR Driver, [879](#)
- CFG_C16_A32_8STR
 - Macronix MX25 SNOR Driver, [879](#)
- CFG_C16_A32_DATA_8DTR
 - Macronix MX25 SNOR Driver, [880](#)
- CFG_C16_A32_DATA_8STR
 - Macronix MX25 SNOR Driver, [880](#)
- CFG_C16_DATA_8DTR
 - Macronix MX25 SNOR Driver, [880](#)
- CFG_C16_DATA_8STR
 - Macronix MX25 SNOR Driver, [879](#)
- CFG_C8_A32_DATA_SPI
 - Macronix MX25 SNOR Driver, [880](#)
- CFG_C8_A32_SPI
 - Macronix MX25 SNOR Driver, [879](#)
- CFG_C8_DATA_SPI
 - Macronix MX25 SNOR Driver, [879](#)
- CFG_C8_SPI
 - Macronix MX25 SNOR Driver, [878](#)
- CH_HAL_MAJOR
 - HAL Driver, [59](#)
- CH_HAL_MINOR
 - HAL Driver, [59](#)
- CH_HAL_PATCH
 - HAL Driver, [59](#)
- CH_HAL_STABLE
 - HAL Driver, [59](#)
- CH_HAL_VERSION
 - HAL Driver, [59](#)
- ch_ltoa
 - Output Formatter Utility, [968](#)
- channels
 - PWMConfig, [1270](#)
 - PWMDriver, [1272](#)
- CHN_BREAK_DETECTED
 - Abstract I/O Channel Interface, [940](#)
- CHN_BUFFER_FULL_ERROR

- Abstract I/O Channel Interface, [940](#)
- CHN_CONNECTED
 - Abstract I/O Channel Interface, [939](#)
- CHN_CTL_INVALID
 - Abstract I/O Channel Interface, [933](#)
- CHN_CTL_NOP
 - Abstract I/O Channel Interface, [933](#)
- CHN_CTL_TX_WAIT
 - Abstract I/O Channel Interface, [933](#)
- CHN_DISCONNECTED
 - Abstract I/O Channel Interface, [939](#)
- CHN_FRAMING_ERROR
 - Abstract I/O Channel Interface, [939](#)
- CHN_IDLE_DETECTED
 - Abstract I/O Channel Interface, [940](#)
- CHN_INPUT_AVAILABLE
 - Abstract I/O Channel Interface, [939](#)
- CHN_NO_ERROR
 - Abstract I/O Channel Interface, [938](#)
- CHN_NOISE_ERROR
 - Abstract I/O Channel Interface, [939](#)
- CHN_OUTPUT_EMPTY
 - Abstract I/O Channel Interface, [939](#)
- CHN_OVERRUN_ERROR
 - Abstract I/O Channel Interface, [940](#)
- CHN_PARITY_ERROR
 - Abstract I/O Channel Interface, [939](#)
- CHN_TRANSMISSION_END
 - Abstract I/O Channel Interface, [939](#)
- chnAddFlagsl
 - Abstract I/O Channel Interface, [941](#)
- chnControl
 - Abstract I/O Channel Interface, [938](#)
- chnGetEventSource
 - Abstract I/O Channel Interface, [940](#)
- chnGetTimeout
 - Abstract I/O Channel Interface, [934](#)
- chnPutTimeout
 - Abstract I/O Channel Interface, [933](#)
- chnRead
 - Abstract I/O Channel Interface, [936](#)
- chnReadTimeout
 - Abstract I/O Channel Interface, [936](#)
- chnWrite
 - Abstract I/O Channel Interface, [935](#)
- chnWriteTimeout
 - Abstract I/O Channel Interface, [935](#)
- chprintf
 - Output Formatter Utility, [969](#)
- chprintf.c, [1444](#)
- chprintf.h, [1445](#)
- CHPRINTF_USE_FLOAT
 - Output Formatter Utility, [968](#)
- chscanf
 - Input Formatter Utility, [975](#)
- chscanf.c, [1445](#)
- chscanf.h, [1446](#)
- CHSCANF_USE_FLOAT
 - Input Formatter Utility, [974](#)
- chsnprintf
 - Output Formatter Utility, [970](#)
- chsnscanf
 - Input Formatter Utility, [976](#)
- chvprintf
 - Output Formatter Utility, [968](#)
- chvscanf
 - Input Formatter Utility, [974](#)
- chvsnprintf
 - Output Formatter Utility, [972](#)
- chvsnsnscanf
 - Input Formatter Utility, [977](#)
- circular
 - hal_adc_configuration_group, [1160](#)
 - hal_spi_config, [1195](#)
- cmd
 - snor_nocache_buffer, [1293](#)
 - wsapi_command_t, [1331](#)
 - xsnor_commands, [1338](#)
- cmd1l
 - Micron N25Q Driver, [913](#)
- cmd2l
 - Micron N25Q Driver, [913](#)
- cmd4l
 - Micron N25Q Driver, [914](#)
- cmd_8dtr
 - Macronix MX25 SNOR Driver, [892](#)
- cmd_8str
 - Macronix MX25 SNOR Driver, [891](#)
- cmd_addr
 - xsnor_commands, [1338](#)
- cmd_addr_data
 - xsnor_commands, [1339](#)
- CMD_BULK_ERASE
 - Micron N25Q Driver, [901](#)
- CMD_CLEAR_FLAG_STATUS_REGISTER
 - Micron N25Q Driver, [900](#)
- cmd_data
 - xsnor_commands, [1339](#)
- CMD_FAST_READ
 - Micron N25Q Driver, [899](#)
- CMD_MULTIPLE_IO_READ_ID
 - Micron N25Q Driver, [899](#)
- CMD_OPI_8DTRD
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_OPI_8READ
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_OPI_BE
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_CE
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_DP
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_ENSO
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_ESFBR
 - Macronix MX25 SNOR Driver, [876](#)

- CMD_OPI_ESSPB
 - Macronix MX25 SNOR Driver, [876](#)
- CMD_OPI_EXSO
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_OPI_GBLK
 - Macronix MX25 SNOR Driver, [877](#)
- CMD_OPI_GBULK
 - Macronix MX25 SNOR Driver, [877](#)
- CMD_OPI_NOP
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_OPI_PASSULK
 - Macronix MX25 SNOR Driver, [877](#)
- CMD_OPI_PE_RESUME
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_PE_SUSPEND
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_PP
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_OPI_RDCCR
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_OPI_RDCCR2
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_OPI_RDDPB
 - Macronix MX25 SNOR Driver, [877](#)
- CMD_OPI_RDFBR
 - Macronix MX25 SNOR Driver, [876](#)
- CMD_OPI_RDID
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_OPI_RDLR
 - Macronix MX25 SNOR Driver, [876](#)
- CMD_OPI_RDPASS
 - Macronix MX25 SNOR Driver, [877](#)
- CMD_OPI_RDSCUR
 - Macronix MX25 SNOR Driver, [876](#)
- CMD_OPI_RDSFDP
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_OPI_RDSPB
 - Macronix MX25 SNOR Driver, [876](#)
- CMD_OPI_RDSR
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_OPI_RST
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_OPI_RSTEN
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_OPI_SBL
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_SE
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_WPSEL
 - Macronix MX25 SNOR Driver, [877](#)
- CMD_OPI_WRCR
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_OPI_WRCR2
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_OPI_WRDI
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_WRPDPB
 - Macronix MX25 SNOR Driver, [877](#)
- CMD_OPI_WREN
 - Macronix MX25 SNOR Driver, [874](#)
- CMD_OPI_WRFBR
 - Macronix MX25 SNOR Driver, [876](#)
- CMD_OPI_WRLR
 - Macronix MX25 SNOR Driver, [876](#)
- CMD_OPI_WRPASS
 - Macronix MX25 SNOR Driver, [877](#)
- CMD_OPI_WRSCUR
 - Macronix MX25 SNOR Driver, [876](#)
- CMD_OPI_WRSPB
 - Macronix MX25 SNOR Driver, [876](#)
- CMD_OPI_WRSR
 - Macronix MX25 SNOR Driver, [875](#)
- CMD_PAGE_PROGRAM
 - Micron N25Q Driver, [900](#)
- CMD_PROGRAM_ERASE_RESUME
 - Micron N25Q Driver, [901](#)
- CMD_PROGRAM_ERASE_SUSPEND
 - Micron N25Q Driver, [901](#)
- CMD_PROGRAM_OTP_ARRAY
 - Micron N25Q Driver, [901](#)
- CMD_READ
 - Micron N25Q Driver, [899](#)
- CMD_READ_DISCOVERY_PARAMETER
 - Micron N25Q Driver, [899](#)
- CMD_READ_ENHANCED_V_CONF_REGISTER
 - Micron N25Q Driver, [900](#)
- CMD_READ_FLAG_STATUS_REGISTER
 - Micron N25Q Driver, [900](#)
- CMD_READ_ID
 - Micron N25Q Driver, [899](#)
- CMD_READ_LOCK_REGISTER
 - Micron N25Q Driver, [899](#)
- CMD_READ_NV_CONFIGURATION_REGISTER
 - Micron N25Q Driver, [900](#)
- CMD_READ_OTP_ARRAY
 - Micron N25Q Driver, [901](#)
- CMD_READ_STATUS_REGISTER
 - Micron N25Q Driver, [899](#)
- CMD_READ_V_CONF_REGISTER
 - Micron N25Q Driver, [900](#)
- CMD_RESET_ENABLE
 - Micron N25Q Driver, [898](#)
- cmd_reset_enable_1
 - Micron N25Q Driver, [914](#)
- cmd_reset_enable_2
 - Micron N25Q Driver, [914](#)
- cmd_reset_enable_4
 - Micron N25Q Driver, [915](#)
- cmd_reset_enable_8dtr
 - Macronix MX25 SNOR Driver, [893](#)
- cmd_reset_enable_8str
 - Macronix MX25 SNOR Driver, [892](#)
- cmd_reset_enable_spi
 - Macronix MX25 SNOR Driver, [892](#)
- CMD_RESET_MEMORY
 - Micron N25Q Driver, [898](#)

- cmd_reset_memory_1
 - Micron N25Q Driver, [914](#)
- cmd_reset_memory_2
 - Micron N25Q Driver, [915](#)
- cmd_reset_memory_4
 - Micron N25Q Driver, [915](#)
- cmd_reset_memory_8dtr
 - Macronix MX25 SNOR Driver, [893](#)
- cmd_reset_memory_8str
 - Macronix MX25 SNOR Driver, [892](#)
- cmd_reset_memory_spi
 - Macronix MX25 SNOR Driver, [892](#)
- CMD_SECTOR_ERASE
 - Micron N25Q Driver, [901](#)
- cmd_spi
 - Macronix MX25 SNOR Driver, [891](#)
- CMD_SPI_BE3B
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_BE4B
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_CE
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_DP
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_ENSO
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_ESFBR
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SPI_ESSPB
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_EXSO
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_FAST_READ3B
 - Macronix MX25 SNOR Driver, [868](#)
- CMD_SPI_FAST_READ4B
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_GBLK
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_SPI_GBULK
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_SPI_NOP
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_PASSULK
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_SPI_PE_RESUME
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_PE_SUSPEND
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_PP3B
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_PP4B
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_RD CR
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SPI_RD CR2
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SPI_RDDPB
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_RDFBR
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SPI_RDID
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SPI_RDLR
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_RDPASS
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_SPI_RDSCUR
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_RDSFDP
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SPI_RDSPB
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_RDSR
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SPI_READ3B
 - Macronix MX25 SNOR Driver, [868](#)
- CMD_SPI_READ4B
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_RST
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_RSTEN
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_SBL
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_SE3B
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_SE4B
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_WPSEL
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_WRCR2
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SPI_WRDI
 - Macronix MX25 SNOR Driver, [870](#)
- CMD_SPI_WRPB
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_WREN
 - Macronix MX25 SNOR Driver, [869](#)
- CMD_SPI_WRFBR
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SPI_WRLR
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_WRPASS
 - Macronix MX25 SNOR Driver, [873](#)
- CMD_SPI_WRSCUR
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_WRSPB
 - Macronix MX25 SNOR Driver, [872](#)
- CMD_SPI_WRSR
 - Macronix MX25 SNOR Driver, [871](#)
- CMD_SUBSECTOR_ERASE
 - Micron N25Q Driver, [901](#)
- CMD_WRITE_DISABLE
 - Micron N25Q Driver, [899](#)
- CMD_WRITE_ENABLE
 - Micron N25Q Driver, [899](#)

- CMD_WRITE_ENHANCED_V_CONF_REGISTER
 - Micron N25Q Driver, [900](#)
- CMD_WRITE_LOCK_REGISTER
 - Micron N25Q Driver, [900](#)
- CMD_WRITE_NV_CONFIGURATION_REGISTER
 - Micron N25Q Driver, [900](#)
- CMD_WRITE_STATUS_REGISTER
 - Micron N25Q Driver, [899](#)
- CMD_WRITE_V_CONF_REGISTER
 - Micron N25Q Driver, [900](#)
- cmdbuf
 - xsnor_buffers, [1332](#)
- commands
 - hal_xsnor_base, [1208](#)
 - hal_xsnor_macronix_mx25, [1214](#)
 - hal_xsnor_micron_n25q, [1220](#)
- Compiler portability, [23](#)
 - CC_ALIGN_CODE, [24](#)
 - CC_ALIGN_DATA, [24](#)
 - CC_FORCE_INLINE, [25](#)
 - CC_LIKELY, [26](#)
 - CC_NO_INLINE, [25](#)
 - CC_NO_RETURN, [25](#)
 - CC_PACK, [25](#)
 - CC_ROMCONST, [26](#)
 - CC_SECTION, [24](#)
 - CC_UNLIKELY, [26](#)
 - CC_USED, [24](#)
 - CC_WEAK, [24](#)
- Complex Drivers, [721](#)
- config
 - CRYDriver, [1142](#)
 - GPTDriver, [1158](#)
 - hal_adc_driver, [1162](#)
 - hal_can_driver, [1169](#)
 - hal_dac_driver, [1174](#)
 - hal_i2c_driver, [1179](#)
 - hal_i2s_driver, [1182](#)
 - hal_mac_driver, [1184](#)
 - hal_sio_driver, [1192](#)
 - hal_spi_driver, [1198](#)
 - hal_trng_driver, [1199](#)
 - hal_uart_driver, [1203](#)
 - hal_wspi_driver, [1206](#)
 - hal_xsnor_base, [1208](#)
 - hal_xsnor_macronix_mx25, [1214](#)
 - hal_xsnor_micron_n25q, [1220](#)
 - ICUDriver, [1229](#)
 - MFSDriver, [1251](#)
 - mmc_spi_driver_t, [1257](#)
 - PWMDriver, [1272](#)
 - SDCDriver, [1280](#)
 - SNORDriver, [1297](#)
 - USBDriver, [1320](#)
 - WDGDriver, [1330](#)
- Configuration, [43](#)
 - _CHIBIOS_HAL_CONF_, [45](#)
 - _CHIBIOS_HAL_CONF_VER_9_0_, [45](#)
- ADC_USE_MUTUAL_EXCLUSION, [49](#)
- ADC_USE_WAIT, [49](#)
- CAN_ENFORCE_USE_CALLBACKS, [49](#)
- CAN_USE_SLEEP_MODE, [49](#)
- DAC_USE_MUTUAL_EXCLUSION, [50](#)
- DAC_USE_WAIT, [50](#)
- HAL_CRY_ENFORCE_FALLBACK, [50](#)
- HAL_CRY_USE_FALLBACK, [50](#)
- HAL_USE_ADC, [46](#)
- HAL_USE_CAN, [46](#)
- HAL_USE_CRY, [46](#)
- HAL_USE_DAC, [46](#)
- HAL_USE_EFL, [46](#)
- HAL_USE_GPT, [46](#)
- HAL_USE_I2C, [46](#)
- HAL_USE_I2S, [46](#)
- HAL_USE_ICU, [47](#)
- HAL_USE_MAC, [47](#)
- HAL_USE_MMC_SPI, [47](#)
- HAL_USE_PAL, [45](#)
- HAL_USE_PWM, [47](#)
- HAL_USE_RTC, [47](#)
- HAL_USE_SDC, [47](#)
- HAL_USE_SERIAL, [47](#)
- HAL_USE_SERIAL_USB, [47](#)
- HAL_USE_SIO, [48](#)
- HAL_USE_SPI, [48](#)
- HAL_USE_TRNG, [48](#)
- HAL_USE_UART, [48](#)
- HAL_USE_USB, [48](#)
- HAL_USE_WDG, [48](#)
- HAL_USE_WSPI, [48](#)
- I2C_ENABLE_SLAVE_MODE, [50](#)
- I2C_USE_MUTUAL_EXCLUSION, [51](#)
- MAC_USE_EVENTS, [51](#)
- MAC_USE_ZERO_COPY, [51](#)
- MMC_IDLE_TIMEOUT_MS, [51](#)
- MMC_USE_MUTUAL_EXCLUSION, [51](#)
- PAL_USE_CALLBACKS, [48](#)
- PAL_USE_WAIT, [49](#)
- SDC_INIT_OCR, [52](#)
- SDC_INIT_OCR_V20, [52](#)
- SDC_INIT_RETRY, [51](#)
- SDC_MMC_SUPPORT, [52](#)
- SDC_NICE_WAITING, [52](#)
- SERIAL_BUFFERS_SIZE, [52](#)
- SERIAL_DEFAULT_BITRATE, [52](#)
- SERIAL_USB_BUFFERS_NUMBER, [53](#)
- SERIAL_USB_BUFFERS_SIZE, [53](#)
- SIO_DEFAULT_BITRATE, [53](#)
- SIO_USE_SYNCHRONIZATION, [53](#)
- SPI_SELECT_MODE, [54](#)
- SPI_USE_ASSERT_ON_ERROR, [54](#)
- SPI_USE_MUTUAL_EXCLUSION, [54](#)
- SPI_USE_WAIT, [53](#)
- UART_USE_MUTUAL_EXCLUSION, [54](#)
- UART_USE_WAIT, [54](#)
- USB_USE_WAIT, [55](#)

- WSPI_USE_MUTUAL_EXCLUSION, [55](#)
- WSPI_USE_WAIT, [55](#)
- configuration
 - USBDriver, [1322](#)
- content_prot_app
 - unpacked_mmc_csd_t, [1305](#)
- copy
 - unpacked_mmc_csd_t, [1305](#)
 - unpacked_sdc_csd_10_t, [1312](#)
 - unpacked_sdc_csd_20_t, [1315](#)
- counter
 - mfs_bank_header_t, [1241](#)
- crc
 - mfs_bank_header_t, [1241](#)
 - mfs_data_header_t, [1243](#)
 - unpacked_mmc_cid_t, [1301](#)
 - unpacked_mmc_csd_t, [1306](#)
 - unpacked_sdc_cid_t, [1308](#)
 - unpacked_sdc_csd_10_t, [1312](#)
 - unpacked_sdc_csd_20_t, [1315](#)
- crc16
 - Managed Flash Storage Driver, [746](#)
- crc16_table
 - Managed Flash Storage Driver, [764](#)
- cry_algo_aes
 - Crypto Driver, [119](#)
- cry_algo_des
 - Crypto Driver, [119](#)
- cry_algo_hmac
 - Crypto Driver, [119](#)
- cry_algo_none
 - Crypto Driver, [119](#)
- CRY_ERR_AUTH_FAILED
 - Crypto Driver, [119](#)
- CRY_ERR_INV_ALGO
 - Crypto Driver, [119](#)
- CRY_ERR_INV_KEY_ID
 - Crypto Driver, [119](#)
- CRY_ERR_INV_KEY_SIZE
 - Crypto Driver, [119](#)
- CRY_ERR_INV_KEY_TYPE
 - Crypto Driver, [119](#)
- CRY_ERR_OP_FAILURE
 - Crypto Driver, [119](#)
- cry_llc_aes_loadkey
 - Crypto Driver, [156](#)
- cry_llc_decrypt_AES
 - Crypto Driver, [157](#)
- cry_llc_decrypt_AES_CBC
 - Crypto Driver, [160](#)
- cry_llc_decrypt_AES_CFB
 - Crypto Driver, [163](#)
- cry_llc_decrypt_AES_CTR
 - Crypto Driver, [164](#)
- cry_llc_decrypt_AES_ECB
 - Crypto Driver, [159](#)
- cry_llc_decrypt_AES_GCM
 - Crypto Driver, [166](#)
- cry_llc_decrypt_DES
 - Crypto Driver, [168](#)
- cry_llc_decrypt_DES_CBC
 - Crypto Driver, [171](#)
- cry_llc_decrypt_DES_ECB
 - Crypto Driver, [170](#)
- cry_llc_des_loadkey
 - Crypto Driver, [167](#)
- cry_llc_encrypt_AES
 - Crypto Driver, [157](#)
- cry_llc_encrypt_AES_CBC
 - Crypto Driver, [160](#)
- cry_llc_encrypt_AES_CFB
 - Crypto Driver, [161](#)
- cry_llc_encrypt_AES_CTR
 - Crypto Driver, [164](#)
- cry_llc_encrypt_AES_ECB
 - Crypto Driver, [158](#)
- cry_llc_encrypt_AES_GCM
 - Crypto Driver, [165](#)
- cry_llc_encrypt_DES
 - Crypto Driver, [168](#)
- cry_llc_encrypt_DES_CBC
 - Crypto Driver, [171](#)
- cry_llc_encrypt_DES_ECB
 - Crypto Driver, [169](#)
- cry_llc_hmac_loadkey
 - Crypto Driver, [177](#)
- cry_llc_HMACSHA256_final
 - Crypto Driver, [179](#)
- cry_llc_HMACSHA256_init
 - Crypto Driver, [178](#)
- cry_llc_HMACSHA256_update
 - Crypto Driver, [179](#)
- cry_llc_HMACSHA512_final
 - Crypto Driver, [181](#)
- cry_llc_HMACSHA512_init
 - Crypto Driver, [180](#)
- cry_llc_HMACSHA512_update
 - Crypto Driver, [180](#)
- cry_llc_init
 - Crypto Driver, [155](#)
- cry_llc_SHA1_final
 - Crypto Driver, [173](#)
- cry_llc_SHA1_init
 - Crypto Driver, [172](#)
- cry_llc_SHA1_update
 - Crypto Driver, [173](#)
- cry_llc_SHA256_final
 - Crypto Driver, [175](#)
- cry_llc_SHA256_init
 - Crypto Driver, [174](#)
- cry_llc_SHA256_update
 - Crypto Driver, [175](#)
- cry_llc_SHA512_final
 - Crypto Driver, [177](#)
- cry_llc_SHA512_init
 - Crypto Driver, [176](#)

- cry_llc_SHA512_update
 - Crypto Driver, [176](#)
- cry_llc_start
 - Crypto Driver, [155](#)
- cry_llc_stop
 - Crypto Driver, [156](#)
- CRY_LLD_SUPPORTS_AES
 - Crypto Driver, [116](#)
- CRY_LLD_SUPPORTS_AES_CBC
 - Crypto Driver, [116](#)
- CRY_LLD_SUPPORTS_AES_CFB
 - Crypto Driver, [117](#)
- CRY_LLD_SUPPORTS_AES_CTR
 - Crypto Driver, [117](#)
- CRY_LLD_SUPPORTS_AES_ECB
 - Crypto Driver, [116](#)
- CRY_LLD_SUPPORTS_AES_GCM
 - Crypto Driver, [117](#)
- CRY_LLD_SUPPORTS_DES
 - Crypto Driver, [117](#)
- CRY_LLD_SUPPORTS_DES_CBC
 - Crypto Driver, [117](#)
- CRY_LLD_SUPPORTS_DES_ECB
 - Crypto Driver, [117](#)
- CRY_LLD_SUPPORTS_HMAC_SHA256
 - Crypto Driver, [117](#)
- CRY_LLD_SUPPORTS_HMAC_SHA512
 - Crypto Driver, [118](#)
- CRY_LLD_SUPPORTS_SHA1
 - Crypto Driver, [117](#)
- CRY_LLD_SUPPORTS_SHA256
 - Crypto Driver, [117](#)
- CRY_LLD_SUPPORTS_SHA512
 - Crypto Driver, [117](#)
- CRY_NOERROR
 - Crypto Driver, [119](#)
- CRY_READY
 - Crypto Driver, [119](#)
- CRY_STOP
 - Crypto Driver, [119](#)
- CRY_UNINIT
 - Crypto Driver, [119](#)
- cryalgorithm_t
 - Crypto Driver, [119](#)
- CRYConfig, [1141](#)
 - dummy, [1141](#)
- CRYD1
 - Crypto Driver, [182](#)
- cryDecryptAES
 - Crypto Driver, [124](#)
- cryDecryptAES_CBC
 - Crypto Driver, [128](#)
- cryDecryptAES_CFB
 - Crypto Driver, [130](#)
- cryDecryptAES_CTR
 - Crypto Driver, [132](#)
- cryDecryptAES_ECB
 - Crypto Driver, [126](#)
- cryDecryptAES_GCM
 - Crypto Driver, [134](#)
- cryDecryptDES
 - Crypto Driver, [137](#)
- cryDecryptDES_CBC
 - Crypto Driver, [141](#)
- cryDecryptDES_ECB
 - Crypto Driver, [139](#)
- CRYDriver, [1142](#)
 - config, [1142](#)
 - Crypto Driver, [118](#)
 - state, [1142](#)
- cryEncryptAES
 - Crypto Driver, [123](#)
- cryEncryptAES_CBC
 - Crypto Driver, [127](#)
- cryEncryptAES_CFB
 - Crypto Driver, [129](#)
- cryEncryptAES_CTR
 - Crypto Driver, [131](#)
- cryEncryptAES_ECB
 - Crypto Driver, [125](#)
- cryEncryptAES_GCM
 - Crypto Driver, [133](#)
- cryEncryptDES
 - Crypto Driver, [136](#)
- cryEncryptDES_CBC
 - Crypto Driver, [140](#)
- cryEncryptDES_ECB
 - Crypto Driver, [138](#)
- cryerror_t
 - Crypto Driver, [119](#)
- cryHMACSHA256Final
 - Crypto Driver, [152](#)
- cryHMACSHA256Init
 - Crypto Driver, [150](#)
- cryHMACSHA256Update
 - Crypto Driver, [151](#)
- cryHMACSHA512Final
 - Crypto Driver, [154](#)
- cryHMACSHA512Init
 - Crypto Driver, [153](#)
- cryHMACSHA512Update
 - Crypto Driver, [153](#)
- cryInit
 - Crypto Driver, [120](#)
- crykey_t
 - Crypto Driver, [118](#)
- cryLoadAESTransientKey
 - Crypto Driver, [122](#)
- cryLoadDESTransientKey
 - Crypto Driver, [135](#)
- cryLoadHMACTransientKey
 - Crypto Driver, [149](#)
- cryObjectInit
 - Crypto Driver, [120](#)
- Crypto Driver, [110](#)
 - bitsize_t, [118](#)

cry_algo_aes, 119
 cry_algo_des, 119
 cry_algo_hmac, 119
 cry_algo_none, 119
 CRY_ERR_AUTH_FAILED, 119
 CRY_ERR_INV_ALGO, 119
 CRY_ERR_INV_KEY_ID, 119
 CRY_ERR_INV_KEY_SIZE, 119
 CRY_ERR_INV_KEY_TYPE, 119
 CRY_ERR_OP_FAILURE, 119
 cry_llc_aes_loadkey, 156
 cry_llc_decrypt_AES, 157
 cry_llc_decrypt_AES_CBC, 160
 cry_llc_decrypt_AES_CFB, 163
 cry_llc_decrypt_AES_CTR, 164
 cry_llc_decrypt_AES_ECB, 159
 cry_llc_decrypt_AES_GCM, 166
 cry_llc_decrypt_DES, 168
 cry_llc_decrypt_DES_CBC, 171
 cry_llc_decrypt_DES_ECB, 170
 cry_llc_des_loadkey, 167
 cry_llc_encrypt_AES, 157
 cry_llc_encrypt_AES_CBC, 160
 cry_llc_encrypt_AES_CFB, 161
 cry_llc_encrypt_AES_CTR, 164
 cry_llc_encrypt_AES_ECB, 158
 cry_llc_encrypt_AES_GCM, 165
 cry_llc_encrypt_DES, 168
 cry_llc_encrypt_DES_CBC, 171
 cry_llc_encrypt_DES_ECB, 169
 cry_llc_hmac_loadkey, 177
 cry_llc_HMACSHA256_final, 179
 cry_llc_HMACSHA256_init, 178
 cry_llc_HMACSHA256_update, 179
 cry_llc_HMACSHA512_final, 181
 cry_llc_HMACSHA512_init, 180
 cry_llc_HMACSHA512_update, 180
 cry_llc_init, 155
 cry_llc_SHA1_final, 173
 cry_llc_SHA1_init, 172
 cry_llc_SHA1_update, 173
 cry_llc_SHA256_final, 175
 cry_llc_SHA256_init, 174
 cry_llc_SHA256_update, 175
 cry_llc_SHA512_final, 177
 cry_llc_SHA512_init, 176
 cry_llc_SHA512_update, 176
 cry_llc_start, 155
 cry_llc_stop, 156
 CRY_LLD_SUPPORTS_AES, 116
 CRY_LLD_SUPPORTS_AES_CBC, 116
 CRY_LLD_SUPPORTS_AES_CFB, 117
 CRY_LLD_SUPPORTS_AES_CTR, 117
 CRY_LLD_SUPPORTS_AES_ECB, 116
 CRY_LLD_SUPPORTS_AES_GCM, 117
 CRY_LLD_SUPPORTS_DES, 117
 CRY_LLD_SUPPORTS_DES_CBC, 117
 CRY_LLD_SUPPORTS_DES_ECB, 117
 CRY_LLD_SUPPORTS_HMAC_SHA256, 117
 CRY_LLD_SUPPORTS_HMAC_SHA512, 118
 CRY_LLD_SUPPORTS_SHA1, 117
 CRY_LLD_SUPPORTS_SHA256, 117
 CRY_LLD_SUPPORTS_SHA512, 117
 CRY_NOERROR, 119
 CRY_READY, 119
 CRY_STOP, 119
 CRY_UNINIT, 119
 cryalgorithm_t, 119
 CRYD1, 182
 cryDecryptAES, 124
 cryDecryptAES_CBC, 128
 cryDecryptAES_CFB, 130
 cryDecryptAES_CTR, 132
 cryDecryptAES_ECB, 126
 cryDecryptAES_GCM, 134
 cryDecryptDES, 137
 cryDecryptDES_CBC, 141
 cryDecryptDES_ECB, 139
 CRYDriver, 118
 cryEncryptAES, 123
 cryEncryptAES_CBC, 127
 cryEncryptAES_CFB, 129
 cryEncryptAES_CTR, 131
 cryEncryptAES_ECB, 125
 cryEncryptAES_GCM, 133
 cryEncryptDES, 136
 cryEncryptDES_CBC, 140
 cryEncryptDES_ECB, 138
 cryerror_t, 119
 cryHMACSHA256Final, 152
 cryHMACSHA256Init, 150
 cryHMACSHA256Update, 151
 cryHMACSHA512Final, 154
 cryHMACSHA512Init, 153
 cryHMACSHA512Update, 153
 cryInit, 120
 crykey_t, 118
 cryLoadAESTransientKey, 122
 cryLoadDESTransientKey, 135
 cryLoadHMACTransientKey, 149
 cryObjectInit, 120
 crySHA1Final, 144
 crySHA1Init, 142
 crySHA1Update, 143
 crySHA256Final, 146
 crySHA256Init, 145
 crySHA256Update, 145
 crySHA512Final, 149
 crySHA512Init, 147
 crySHA512Update, 148
 cryStart, 120
 crystate_t, 118
 cryStop, 121
 HAL_CRY_ENFORCE_FALLBACK, 116
 HAL_CRY_USE_FALLBACK, 116
 PLATFORM_CRY_USE_CRY1, 118

- crySHA1Final
 - Crypto Driver, [144](#)
- crySHA1Init
 - Crypto Driver, [142](#)
- crySHA1Update
 - Crypto Driver, [143](#)
- crySHA256Final
 - Crypto Driver, [146](#)
- crySHA256Init
 - Crypto Driver, [145](#)
- crySHA256Update
 - Crypto Driver, [145](#)
- crySHA512Final
 - Crypto Driver, [149](#)
- crySHA512Init
 - Crypto Driver, [147](#)
- crySHA512Update
 - Crypto Driver, [148](#)
- cryStart
 - Crypto Driver, [120](#)
- crystate_t
 - Crypto Driver, [118](#)
- cryStop
 - Crypto Driver, [121](#)
- csd_structure
 - unpacked_mmc_csd_t, [1303](#)
 - unpacked_sdc_csd_10_t, [1309](#)
 - unpacked_sdc_csd_20_t, [1313](#)
- current_bank
 - MFSDriver, [1251](#)
- current_counter
 - MFSDriver, [1252](#)
- DAC Driver, [182](#)
 - _dac_isr_error_code, [189](#)
 - _dac_isr_full_code, [188](#)
 - _dac_isr_half_code, [187](#)
 - _dac_reset_i, [186](#)
 - _dac_reset_s, [186](#)
 - _dac_timeout_isr, [187](#)
 - _dac_wait_s, [185](#)
 - _dac_wakeup_isr, [187](#)
 - DAC_ACTIVE, [192](#)
 - DAC_COMPLETE, [192](#)
 - DAC_ERR_DMAFAILURE, [192](#)
 - DAC_ERR_UNDERFLOW, [192](#)
 - DAC_ERROR, [192](#)
 - dac_llid_config_fields, [190](#)
 - dac_llid_conversion_group_fields, [190](#)
 - dac_llid_driver_fields, [190](#)
 - dac_llid_init, [202](#)
 - dac_llid_put_channel, [203](#)
 - dac_llid_start, [202](#)
 - dac_llid_start_conversion, [203](#)
 - dac_llid_stop, [203](#)
 - dac_llid_stop_conversion, [204](#)
 - DAC_MAX_CHANNELS, [189](#)
 - DAC_READY, [192](#)
 - DAC_STOP, [192](#)
 - DAC_UNINIT, [192](#)
 - DAC_USE_MUTUAL_EXCLUSION, [185](#)
 - DAC_USE_SYNCHRONIZATION, [185](#)
 - dacAcquireBus, [201](#)
 - daccallback_t, [191](#)
 - dacchannel_t, [191](#)
 - DACConfig, [190](#)
 - DACConversionGroup, [191](#)
 - dacConvert, [198](#)
 - DACD1, [204](#)
 - DACDriver, [190](#)
 - dacerror_t, [192](#)
 - dacerrorcallback_t, [191](#)
 - dacInit, [192](#)
 - dacIsBufferComplete, [185](#)
 - dacObjectInit, [192](#)
 - dacPutChannelX, [194](#)
 - dacReleaseBus, [201](#)
 - dacsample_t, [191](#)
 - dacStart, [193](#)
 - dacStartConversion, [195](#)
 - dacStartConversionI, [196](#)
 - dacstate_t, [191](#)
 - dacStop, [194](#)
 - dacStopConversion, [197](#)
 - dacStopConversionI, [197](#)
 - dacSynchronize, [200](#)
 - dacSynchronizeS, [199](#)
 - PLATFORM_DAC_USE_DAC1, [189](#)
- DAC_ACTIVE
 - DAC Driver, [192](#)
- DAC_COMPLETE
 - DAC Driver, [192](#)
- DAC_ERR_DMAFAILURE
 - DAC Driver, [192](#)
- DAC_ERR_UNDERFLOW
 - DAC Driver, [192](#)
- DAC_ERROR
 - DAC Driver, [192](#)
- dac_llid_config_fields
 - DAC Driver, [190](#)
- dac_llid_conversion_group_fields
 - DAC Driver, [190](#)
- dac_llid_driver_fields
 - DAC Driver, [190](#)
- dac_llid_init
 - DAC Driver, [202](#)
- dac_llid_put_channel
 - DAC Driver, [203](#)
- dac_llid_start
 - DAC Driver, [202](#)
- dac_llid_start_conversion
 - DAC Driver, [203](#)
- dac_llid_stop
 - DAC Driver, [203](#)
- dac_llid_stop_conversion
 - DAC Driver, [204](#)
- DAC_MAX_CHANNELS

- DAC Driver, [189](#)
- DAC_READY
 - DAC Driver, [192](#)
- DAC_STOP
 - DAC Driver, [192](#)
- DAC_UNINIT
 - DAC Driver, [192](#)
- DAC_USE_MUTUAL_EXCLUSION
 - Configuration, [50](#)
 - DAC Driver, [185](#)
- DAC_USE_SYNCHRONIZATION
 - DAC Driver, [185](#)
- DAC_USE_WAIT
 - Configuration, [50](#)
- dacAcquireBus
 - DAC Driver, [201](#)
- daccallback_t
 - DAC Driver, [191](#)
- dacchannel_t
 - DAC Driver, [191](#)
- DACConfig
 - DAC Driver, [190](#)
- DACConversionGroup
 - DAC Driver, [191](#)
- dacConvert
 - DAC Driver, [198](#)
- DACD1
 - DAC Driver, [204](#)
- DACDriver
 - DAC Driver, [190](#)
- dacerror_t
 - DAC Driver, [192](#)
- dacerrorcallback_t
 - DAC Driver, [191](#)
- daclnit
 - DAC Driver, [192](#)
- daclsBufferComplete
 - DAC Driver, [185](#)
- dacObjectInit
 - DAC Driver, [192](#)
- dacPutChannelX
 - DAC Driver, [194](#)
- dacReleaseBus
 - DAC Driver, [201](#)
- dacsample_t
 - DAC Driver, [191](#)
- dacStart
 - DAC Driver, [193](#)
- dacStartConversion
 - DAC Driver, [195](#)
- dacStartConversionI
 - DAC Driver, [196](#)
- dacstate_t
 - DAC Driver, [191](#)
- dacStop
 - DAC Driver, [194](#)
- dacStopConversion
 - DAC Driver, [197](#)
- dacStopConversionI
 - DAC Driver, [197](#)
- dacSynchronize
 - DAC Driver, [200](#)
- dacSynchronizeS
 - DAC Driver, [199](#)
- data16
 - CANRxFrame, [1136](#)
 - CANTxFrame, [1139](#)
 - mfs_nocache_buffer, [1245](#)
- data32
 - CANRxFrame, [1137](#)
 - CANTxFrame, [1139](#)
 - mfs_nocache_buffer, [1245](#)
- data8
 - CANRxFrame, [1136](#)
 - CANTxFrame, [1139](#)
 - mfs_nocache_buffer, [1245](#)
- data_cb
 - hal_spi_config, [1196](#)
- databuf
 - xsnor_buffers, [1332](#)
- day
 - RTCDateTime, [1274](#)
- dayofweek
 - RTCDateTime, [1274](#)
- default_config
 - Serial Driver, [462](#)
- default_ecc
 - unpacked_mmc_csd_t, [1305](#)
- default_handler
 - USB Driver, [662](#)
- Deprecated List, [3](#)
- depth
 - hal_adc_driver, [1162](#)
 - hal_dac_driver, [1174](#)
- descriptor
 - hal_xsnor_base, [1209](#)
 - hal_xsnor_macronix_mx25, [1215](#)
 - hal_xsnor_micron_n25q, [1221](#)
- descriptors
 - MFSDriver, [1252](#)
- detect_bus_clk
 - SDC Driver, [414](#)
- dhdr
 - mfs_nocache_buffer, [1245](#)
- dispose
 - hal_xsnor_base_vmt, [1211](#)
 - hal_xsnor_macronix_mx25_vmt, [1217](#)
 - hal_xsnor_micron_n25q_vmt, [1223](#)
- DLC
 - CANRxFrame, [1135](#)
 - CANTxFrame, [1138](#)
- drv
 - xsnor_bus_spi, [1336](#)
 - xsnor_bus_wspi, [1337](#)
- dsr_imp
 - unpacked_mmc_csd_t, [1304](#)

- unpacked_sdc_csd_10_t, [1310](#)
- unpacked_sdc_csd_20_t, [1314](#)
- dstflag
 - RTCDateTime, [1274](#)
- dummy
 - CRYConfig, [1141](#)
 - hal_can_config, [1168](#)
 - hal_i2c_config, [1178](#)
 - halclkcfg_t, [1225](#)
 - HMACSHA256Context, [1225](#)
 - HMACSHA512Context, [1226](#)
 - RTCAAlarm, [1273](#)
 - SHA1Context, [1290](#)
 - SHA256Context, [1290](#)
 - SHA512Context, [1291](#)
 - wsapi_command_t, [1331](#)
- dwDTERate
 - cdc_linecoding_t, [1140](#)
- early
 - hal_uart_driver, [1203](#)
- ecc
 - unpacked_mmc_csd_t, [1306](#)
- EFL Driver, [204](#)
 - _efl_driver_data, [207](#)
 - _efl_flash_methods, [206](#)
 - _efl_flash_methods_alone, [206](#)
 - efl_acquire_exclusive, [208](#)
 - efl_ild_config_fields, [207](#)
 - efl_ild_descriptor, [216](#)
 - efl_ild_driver_fields, [207](#)
 - efl_ild_get_descriptor, [212](#)
 - efl_ild_init, [211](#)
 - efl_ild_program, [213](#)
 - efl_ild_query_erase, [215](#)
 - efl_ild_read, [212](#)
 - efl_ild_start, [211](#)
 - efl_ild_start_erase_all, [214](#)
 - efl_ild_start_erase_sector, [214](#)
 - efl_ild_stop, [212](#)
 - efl_ild_verify_erase, [215](#)
 - efl_release_exclusive, [208](#)
 - EFL_USE_MUTUAL_EXCLUSION, [206](#)
 - EFlashConfig, [208](#)
 - EFlashDriver, [208](#)
 - EFLD1, [216](#)
 - eflInit, [208](#)
 - eflObjectInit, [209](#)
 - eflStart, [209](#)
 - eflStop, [210](#)
 - PLATFORM_EFL_USE_EFL1, [207](#)
 - vmt, [216](#)
- efl_acquire_exclusive
 - EFL Driver, [208](#)
- efl_ild_config_fields
 - EFL Driver, [207](#)
- efl_ild_descriptor
 - EFL Driver, [216](#)
- efl_ild_driver_fields
 - EFL Driver, [207](#)
- efl_ild_get_descriptor
 - EFL Driver, [212](#)
- efl_ild_init
 - EFL Driver, [211](#)
- efl_ild_program
 - EFL Driver, [213](#)
- efl_ild_query_erase
 - EFL Driver, [215](#)
- efl_ild_read
 - EFL Driver, [212](#)
- efl_ild_start
 - EFL Driver, [211](#)
- efl_ild_start_erase_all
 - EFL Driver, [214](#)
- efl_ild_start_erase_sector
 - EFL Driver, [214](#)
- efl_ild_stop
 - EFL Driver, [212](#)
- efl_ild_verify_erase
 - EFL Driver, [215](#)
- efl_release_exclusive
 - EFL Driver, [208](#)
- EFL_USE_MUTUAL_EXCLUSION
 - EFL Driver, [206](#)
- EFlashConfig
 - EFL Driver, [208](#)
- EFlashDriver
 - EFL Driver, [208](#)
- EFlashDriverVMT, [1143](#)
- EFLD1
 - EFL Driver, [216](#)
- eflInit
 - EFL Driver, [208](#)
- eflObjectInit
 - EFL Driver, [209](#)
- eflStart
 - EFL Driver, [209](#)
- eflStop
 - EFL Driver, [210](#)
- EID
 - CANRxFrame, [1136](#)
 - CANTxFrame, [1139](#)
- enabled
 - hal_sio_driver, [1192](#)
 - PWMDriver, [1272](#)
- end_cb
 - hal_adc_configuration_group, [1160](#)
 - hal_dac_conversion_group, [1172](#)
 - hal_i2s_config, [1181](#)
 - hal_spi_config, [1195](#)
 - hal_wsapi_config, [1205](#)
- ep0_state
 - USB Driver, [684](#)
- ep0config
 - USB Driver, [685](#)
- ep0endcb
 - USBDriver, [1321](#)

- ep0n
 - USBDriver, [1321](#)
- ep0next
 - USBDriver, [1321](#)
- ep0state
 - USBDriver, [1321](#)
- ep_mode
 - USBEndpointConfig, [1324](#)
- EP_STATUS_ACTIVE
 - USB Driver, [660](#)
- EP_STATUS_DISABLED
 - USB Driver, [660](#)
- EP_STATUS_STALLED
 - USB Driver, [660](#)
- epc
 - USBDriver, [1320](#)
- erase_blk_en
 - unpacked_sdc_csd_10_t, [1311](#)
 - unpacked_sdc_csd_20_t, [1314](#)
- erase_grp_mult
 - unpacked_mmc_csd_t, [1305](#)
- erase_grp_size
 - unpacked_mmc_csd_t, [1304](#)
- erase_sector_size
 - unpacked_sdc_csd_10_t, [1311](#)
 - unpacked_sdc_csd_20_t, [1314](#)
- erased
 - MFSCConfig, [1249](#)
- error_cb
 - hal_adc_configuration_group, [1160](#)
 - hal_dac_conversion_group, [1172](#)
 - hal_spi_config, [1196](#)
 - hal_wspi_config, [1205](#)
- error_event
 - hal_can_driver, [1170](#)
- errors
 - hal_i2c_driver, [1179](#)
 - SDCDriver, [1280](#)
- es
 - hal_mac_driver, [1184](#)
- ETHD1
 - MAC Driver, [307](#)
- event_cb
 - USBConfig, [1317](#)
- event_source, [1144](#)
 - cb, [1144](#)
 - flags, [1144](#)
 - param, [1145](#)
- event_source_t
 - OSAL, [1090](#)
- eventcallback_t
 - OSAL, [1090](#)
- eventflags_t
 - OSAL, [1090](#)
- EXPANSION_ENABLENPAGE
 - MII/RMII Header, [1066](#)
- EXPANSION_LCWP
 - MII/RMII Header, [1066](#)
- EXPANSION_MFAULTS
 - MII/RMII Header, [1066](#)
- EXPANSION_NPCAPABLE
 - MII/RMII Header, [1066](#)
- EXPANSION_NWAY
 - MII/RMII Header, [1066](#)
- EXPANSION_RESV
 - MII/RMII Header, [1066](#)
- FALSE
 - OSAL, [1079](#)
- fields
 - mfs_bank_header_t, [1241](#)
 - mfs_data_header_t, [1243](#)
- FILE_EOF
 - Abstract Files Interface, [943](#)
- FILE_ERROR
 - Abstract Files Interface, [943](#)
- file_format
 - unpacked_mmc_csd_t, [1306](#)
 - unpacked_sdc_csd_10_t, [1312](#)
 - unpacked_sdc_csd_20_t, [1315](#)
- file_format_grp
 - unpacked_mmc_csd_t, [1305](#)
 - unpacked_sdc_csd_10_t, [1311](#)
 - unpacked_sdc_csd_20_t, [1315](#)
- FILE_OK
 - Abstract Files Interface, [943](#)
- fileoffset_t
 - Abstract Files Interface, [949](#)
- FileStream, [1145](#)
 - vmt, [1147](#)
- fileStreamClose
 - Abstract Files Interface, [946](#)
- fileStreamGet
 - Abstract Files Interface, [945](#)
- fileStreamGetError
 - Abstract Files Interface, [946](#)
- fileStreamGetPosition
 - Abstract Files Interface, [947](#)
- fileStreamGetSize
 - Abstract Files Interface, [947](#)
- fileStreamPut
 - Abstract Files Interface, [944](#)
- fileStreamRead
 - Abstract Files Interface, [944](#)
- fileStreamSetPosition
 - Abstract Files Interface, [948](#)
- FileStreamVMT, [1147](#)
- fileStreamWrite
 - Abstract Files Interface, [943](#)
- flags
 - event_source, [1144](#)
 - hal_mac_driver, [1184](#)
- FLAGS_ALL_ERRORS
 - Macronix MX25 SNOR Driver, [878](#)
 - Micron N25Q Driver, [902](#)
- FLAGS_E_FAIL
 - Macronix MX25 SNOR Driver, [877](#)

- FLAGS_ERASE_ERROR
 - Micron N25Q Driver, [901](#)
- FLAGS_ERASE_SUSPEND
 - Micron N25Q Driver, [901](#)
- FLAGS_ESB
 - Macronix MX25 SNOR Driver, [878](#)
- FLAGS_LDSD
 - Macronix MX25 SNOR Driver, [878](#)
- FLAGS_P_FAIL
 - Macronix MX25 SNOR Driver, [878](#)
- FLAGS_PROGRAM_ERASE
 - Micron N25Q Driver, [901](#)
- FLAGS_PROGRAM_ERROR
 - Micron N25Q Driver, [902](#)
- FLAGS_PROGRAM_SUSPEND
 - Micron N25Q Driver, [902](#)
- FLAGS_PROTECTION_ERROR
 - Micron N25Q Driver, [902](#)
- FLAGS_PSB
 - Macronix MX25 SNOR Driver, [878](#)
- FLAGS_RESERVED
 - Micron N25Q Driver, [902](#)
- FLAGS_SECURED_OTP
 - Macronix MX25 SNOR Driver, [878](#)
- FLAGS_VPP_ERROR
 - Micron N25Q Driver, [902](#)
- FLAGS_WPSEL
 - Macronix MX25 SNOR Driver, [877](#)
- FLASH_ATTR_ECC_CAPABLE
 - Generic Flash Interface, [32](#)
- FLASH_ATTR_ECC_ZERO_LINE_CAPABLE
 - Generic Flash Interface, [33](#)
- FLASH_ATTR_ERASED_IS_ONE
 - Generic Flash Interface, [32](#)
- FLASH_ATTR_MEMORY_MAPPED
 - Generic Flash Interface, [32](#)
- FLASH_ATTR_REWRITABLE
 - Generic Flash Interface, [32](#)
- FLASH_ATTR_SPI_4BYTES_ADDR_HINT
 - Base Serial NOR Driver class, [836](#)
- FLASH_ATTR_SUSPEND_ERASE_CAPABLE
 - Generic Flash Interface, [33](#)
- FLASH_BUSY_ERASING
 - Generic Flash Interface, [40](#)
- flash_descriptor_t, [1148](#)
 - address, [1149](#)
 - attributes, [1149](#)
 - page_size, [1149](#)
 - sectors, [1149](#)
 - sectors_count, [1149](#)
 - sectors_size, [1149](#)
 - size, [1150](#)
- FLASH_ERASE
 - Generic Flash Interface, [40](#)
- FLASH_ERROR_ERASE
 - Generic Flash Interface, [40](#)
- FLASH_ERROR_HW_FAILURE
 - Generic Flash Interface, [40](#)
- FLASH_ERROR_PROGRAM
 - Generic Flash Interface, [40](#)
- FLASH_ERROR_READ
 - Generic Flash Interface, [40](#)
- flash_error_t
 - Generic Flash Interface, [40](#)
- FLASH_ERROR_UNIMPLEMENTED
 - Generic Flash Interface, [40](#)
- FLASH_ERROR_VERIFY
 - Generic Flash Interface, [40](#)
- flash_interface, [1150](#)
 - vmt, [1152](#)
- flash_interface_i, [1152](#)
 - Base Flash Interface, [858](#)
- flash_interface_vmt, [1153](#)
 - acquire_exclusive, [1155](#)
 - get_descriptor, [1154](#)
 - instance_offset, [1154](#)
 - program, [1154](#)
 - query_erase, [1154](#)
 - read, [1154](#)
 - release_exclusive, [1155](#)
 - start_erase_all, [1154](#)
 - start_erase_sector, [1154](#)
 - verify_erase, [1155](#)
- FLASH_NO_ERROR
 - Generic Flash Interface, [40](#)
- flash_offset_t
 - Generic Flash Interface, [39](#)
- FLASH_PGM
 - Generic Flash Interface, [40](#)
- FLASH_READ
 - Generic Flash Interface, [40](#)
- FLASH_READY
 - Generic Flash Interface, [40](#)
- flash_sector_descriptor_t, [1155](#)
 - offset, [1156](#)
 - size, [1156](#)
- flash_sector_t
 - Generic Flash Interface, [39](#)
- flash_state_t
 - Generic Flash Interface, [39](#)
- FLASH_STOP
 - Generic Flash Interface, [40](#)
- FLASH_UNINIT
 - Generic Flash Interface, [40](#)
- flashAcquireExclusive
 - Generic Flash Interface, [38](#)
- flashGetAddressOffset
 - Generic Flash Interface, [42](#)
- flashGetDescriptor
 - Generic Flash Interface, [34](#)
- flashGetOffsetAddress
 - Generic Flash Interface, [42](#)
- flashGetOffsetSector
 - Generic Flash Interface, [41](#)
- flashGetSectorOffset
 - Generic Flash Interface, [40](#)

- flashGetSectorSize
 - Generic Flash Interface, [41](#)
- flashp
 - MFSCfg, [1249](#)
- flashProgram
 - Generic Flash Interface, [35](#)
- flashQueryErase
 - Generic Flash Interface, [37](#)
- flashRead
 - Generic Flash Interface, [34](#)
- flashReleaseExclusive
 - Generic Flash Interface, [39](#)
- flashStartEraseAll
 - Generic Flash Interface, [36](#)
- flashStartEraseSector
 - Generic Flash Interface, [36](#)
- flashVerifyErase
 - Generic Flash Interface, [37](#)
- flashWaitErase
 - Generic Flash Interface, [40](#)
- FLOAT_PRECISION
 - Output Formatter Utility, [967](#)
- fls
 - hal_xsnor_base, [1208](#)
 - hal_xsnor_macronix_mx25, [1214](#)
 - hal_xsnor_micron_n25q, [1220](#)
- flsAcquireExclusive
 - Base Flash Interface, [862](#)
- flsGetDescriptor
 - Base Flash Interface, [858](#)
- flsProgram
 - Base Flash Interface, [859](#)
- flsQueryErase
 - Base Flash Interface, [861](#)
- flsRead
 - Base Flash Interface, [859](#)
- flsReleaseExclusive
 - Base Flash Interface, [862](#)
- flsStartEraseAll
 - Base Flash Interface, [860](#)
- flsStartEraseSector
 - Base Flash Interface, [860](#)
- flsVerifyErase
 - Base Flash Interface, [861](#)
- FMI
 - CANRxFrame, [1135](#)
- frequency
 - GPTConfig, [1157](#)
 - ICUConfig, [1227](#)
 - PWMConfig, [1270](#)
- Generic Flash Interface, [29](#)
 - _base_flash_data, [34](#)
 - _base_flash_methods, [33](#)
 - _base_flash_methods_alone, [33](#)
 - FLASH_ATTR_ECC_CAPABLE, [32](#)
 - FLASH_ATTR_ECC_ZERO_LINE_CAPABLE, [33](#)
 - FLASH_ATTR_ERASED_IS_ONE, [32](#)
 - FLASH_ATTR_MEMORY_MAPPED, [32](#)
 - FLASH_ATTR_REWRITABLE, [32](#)
 - FLASH_ATTR_SUSPEND_ERASE_CAPABLE, [33](#)
 - FLASH_BUSY_ERASING, [40](#)
 - FLASH_ERASE, [40](#)
 - FLASH_ERROR_ERASE, [40](#)
 - FLASH_ERROR_HW_FAILURE, [40](#)
 - FLASH_ERROR_PROGRAM, [40](#)
 - FLASH_ERROR_READ, [40](#)
 - flash_error_t, [40](#)
 - FLASH_ERROR_UNIMPLEMENTED, [40](#)
 - FLASH_ERROR_VERIFY, [40](#)
 - FLASH_NO_ERROR, [40](#)
 - flash_offset_t, [39](#)
 - FLASH_PGM, [40](#)
 - FLASH_READ, [40](#)
 - FLASH_READY, [40](#)
 - flash_sector_t, [39](#)
 - flash_state_t, [39](#)
 - FLASH_STOP, [40](#)
 - FLASH_UNINIT, [40](#)
 - flashAcquireExclusive, [38](#)
 - flashGetAddressOffset, [42](#)
 - flashGetDescriptor, [34](#)
 - flashGetOffsetAddress, [42](#)
 - flashGetOffsetSector, [41](#)
 - flashGetSectorOffset, [40](#)
 - flashGetSectorSize, [41](#)
 - flashProgram, [35](#)
 - flashQueryErase, [37](#)
 - flashRead, [34](#)
 - flashReleaseExclusive, [39](#)
 - flashStartEraseAll, [36](#)
 - flashStartEraseSector, [36](#)
 - flashVerifyErase, [37](#)
 - flashWaitErase, [40](#)
 - getBaseFlash, [34](#)
- get
 - Null Streams Class, [956](#)
- get_descriptor
 - flash_interface_vmt, [1154](#)
- get_descriptor_cb
 - USBConfig, [1317](#)
- get_hword
 - USB Driver, [661](#)
- getBaseFlash
 - Generic Flash Interface, [34](#)
- getBasePersistentStorage
 - Abstract Persistent Storage Interface, [950](#)
- GPT Driver, [217](#)
 - _gpt_isr_invoke_cb, [221](#)
 - GPT_CONTINUOUS, [224](#)
 - gpt_lld_change_interval, [222](#)
 - gpt_lld_init, [231](#)
 - gpt_lld_polled_delay, [233](#)
 - gpt_lld_start, [231](#)
 - gpt_lld_start_timer, [232](#)
 - gpt_lld_stop, [232](#)
 - gpt_lld_stop_timer, [232](#)

- GPT_ONESHOT, [224](#)
- GPT_READY, [224](#)
- GPT_STOP, [224](#)
- GPT_UNINIT, [224](#)
- gptcallback_t, [223](#)
- gptChangeInterval, [226](#)
- gptChangeIntervall, [220](#)
- gptcnt_t, [223](#)
- GPTD1, [233](#)
- GPTDriver, [223](#)
- gptfreq_t, [223](#)
- gptGetCounterX, [221](#)
- gptGetIntervalX, [220](#)
- gptInit, [224](#)
- gptObjectInit, [224](#)
- gptPolledDelay, [230](#)
- gptStart, [224](#)
- gptStartContinuous, [227](#)
- gptStartContinuousl, [227](#)
- gptStartOneShot, [228](#)
- gptStartOneShotl, [228](#)
- gptstate_t, [223](#)
- gptStop, [225](#)
- gptStopTimer, [229](#)
- gptStopTimerl, [230](#)
- PLATFORM_GPT_USE_GPT1, [222](#)
- GPT_CONTINUOUS
 - GPT Driver, [224](#)
- gpt_lld_change_interval
 - GPT Driver, [222](#)
- gpt_lld_init
 - GPT Driver, [231](#)
- gpt_lld_polled_delay
 - GPT Driver, [233](#)
- gpt_lld_start
 - GPT Driver, [231](#)
- gpt_lld_start_timer
 - GPT Driver, [232](#)
- gpt_lld_stop
 - GPT Driver, [232](#)
- gpt_lld_stop_timer
 - GPT Driver, [232](#)
- GPT_ONESHOT
 - GPT Driver, [224](#)
- GPT_READY
 - GPT Driver, [224](#)
- GPT_STOP
 - GPT Driver, [224](#)
- GPT_UNINIT
 - GPT Driver, [224](#)
- gptcallback_t
 - GPT Driver, [223](#)
- gptChangeInterval
 - GPT Driver, [226](#)
- gptChangeIntervall
 - GPT Driver, [220](#)
- gptcnt_t
 - GPT Driver, [223](#)
- GPTConfig, [1156](#)
 - callback, [1157](#)
 - frequency, [1157](#)
- GPTD1
 - GPT Driver, [233](#)
- GPTDriver, [1157](#)
 - config, [1158](#)
 - GPT Driver, [223](#)
 - state, [1158](#)
- gptfreq_t
 - GPT Driver, [223](#)
- gptGetCounterX
 - GPT Driver, [221](#)
- gptGetIntervalX
 - GPT Driver, [220](#)
- gptInit
 - GPT Driver, [224](#)
- gptObjectInit
 - GPT Driver, [224](#)
- gptPolledDelay
 - GPT Driver, [230](#)
- gptStart
 - GPT Driver, [224](#)
- gptStartContinuous
 - GPT Driver, [227](#)
- gptStartContinuousl
 - GPT Driver, [227](#)
- gptStartOneShot
 - GPT Driver, [228](#)
- gptStartOneShotl
 - GPT Driver, [228](#)
- gptstate_t
 - GPT Driver, [223](#)
- gptStop
 - GPT Driver, [225](#)
- gptStopTimer
 - GPT Driver, [229](#)
- gptStopTimerl
 - GPT Driver, [230](#)
- grpp
 - hal_adc_driver, [1162](#)
 - hal_dac_driver, [1174](#)
- HAL, [27](#)
- HAL Driver, [57](#)
 - __CHIBIOS_HAL__, [59](#)
 - CH_HAL_MAJOR, [59](#)
 - CH_HAL_MINOR, [59](#)
 - CH_HAL_PATCH, [59](#)
 - CH_HAL_STABLE, [59](#)
 - CH_HAL_VERSION, [59](#)
 - HAL_FAILED, [60](#)
 - hal_lld_clock_switch_mode, [67](#)
 - hal_lld_get_clock_point, [67](#)
 - hal_lld_init, [67](#)
 - HAL_LLD_SELECT_SPI_V2, [63](#)
 - HAL_LLD_USE_CLOCK_MANAGEMENT, [63](#)
 - HAL_RET_CONFIG_ERROR, [60](#)
 - HAL_RET_HW_BUSY, [60](#)

- HAL_RET_HW_FAILURE, 61
- HAL_RET_IS_INVALID, 61
- HAL_RET_NO_RESOURCE, 60
- HAL_RET_SUCCESS, 60
- HAL_RET_UNKNOWN_CTL, 61
- HAL_SUCCESS, 60
- HAL_USE_ADC, 61
- HAL_USE_CAN, 61
- HAL_USE_CRY, 61
- HAL_USE_DAC, 61
- HAL_USE_EFL, 62
- HAL_USE_GPT, 62
- HAL_USE_I2C, 62
- HAL_USE_I2S, 62
- HAL_USE_ICU, 62
- HAL_USE_MAC, 62
- HAL_USE_PAL, 61
- HAL_USE_PWM, 62
- HAL_USE_RTC, 62
- HAL_USE_SDC, 62
- HAL_USE_SERIAL, 62
- HAL_USE_SIO, 63
- HAL_USE_SPI, 63
- HAL_USE_TRNG, 63
- HAL_USE_UART, 63
- HAL_USE_USB, 63
- HAL_USE_WDG, 63
- HAL_USE_WSPI, 63
- halclkpt_t, 64
- halClockGetPointX, 65
- halClockSwitchMode, 66
- halfreq_t, 64
- hallnit, 64
- PLATFORM_NAME, 63
- hal.c, 1449
- hal.dox, 1344
- hal.h, 1344
- hal_adc.c, 1449
- hal_adc.dox, 1344
- hal_adc.h, 1347
- hal_adc_config, 1158
- hal_adc_configuration_group, 1159
 - circular, 1160
 - end_cb, 1160
 - error_cb, 1160
 - num_channels, 1160
- hal_adc_driver, 1160
 - config, 1162
 - depth, 1162
 - grpp, 1162
 - mutex, 1162
 - samples, 1162
 - state, 1162
 - thread, 1162
- hal_adc_lld.c, 1479
- hal_adc_lld.h, 1479
- hal_buffered_serial, 1163
 - vmt, 1165
- hal_buffered_serial.c, 1450
- hal_buffered_serial.h, 1349
- hal_buffered_sio.c, 1419
- hal_buffered_sio.h, 1420
- hal_buffered_siol_driver, 1165
 - vmt, 1167
- hal_buffererd_sio.dox, 1344
- hal_buffers.c, 1450
- hal_buffers.dox, 1344
- hal_buffers.h, 1350
- hal_can.c, 1451
- hal_can.dox, 1344
- hal_can.h, 1352
- hal_can_config, 1167
 - dummy, 1168
- hal_can_driver, 1168
 - config, 1169
 - error_event, 1170
 - rxfull_event, 1169
 - rxqueue, 1169
 - sleep_event, 1170
 - state, 1169
 - txempty_event, 1170
 - txqueue, 1169
 - wakeup_event, 1170
- hal_can_lld.c, 1480
- hal_can_lld.h, 1481
- hal_channels.h, 1354
- HAL_CRY_ENFORCE_FALLBACK
 - Configuration, 50
 - Crypto Driver, 116
- HAL_CRY_USE_FALLBACK
 - Configuration, 50
 - Crypto Driver, 116
- hal_crypto.c, 1452
- hal_crypto.dox, 1344
- hal_crypto.h, 1355
- hal_crypto_lld.c, 1482
- hal_crypto_lld.h, 1485
- hal_dac.c, 1454
- hal_dac.dox, 1344
- hal_dac.h, 1359
- hal_dac_config, 1171
- hal_dac_conversion_group, 1171
 - end_cb, 1172
 - error_cb, 1172
 - num_channels, 1172
- hal_dac_driver, 1173
 - config, 1174
 - depth, 1174
 - grpp, 1174
 - mutex, 1174
 - samples, 1174
 - state, 1174
 - thread, 1174
- hal_dac_lld.c, 1488
- hal_dac_lld.h, 1488
- hal_efl.c, 1455

- hal_efl.dox, [1344](#)
- hal_efl.h, [1360](#)
- hal_efl_config, [1175](#)
- hal_efl_driver, [1175](#)
 - mutex, [1177](#)
 - vmt, [1177](#)
- hal_efl_lld.c, [1490](#)
- hal_efl_lld.h, [1490](#)
- HAL_FAILED
 - HAL Driver, [60](#)
- hal_files.h, [1361](#)
- hal_flash.c, [1456](#)
- hal_flash.dox, [1344](#)
- hal_flash.h, [1362](#)
- hal_flash_interface.h, [1438](#)
- hal_gpt.c, [1456](#)
- hal_gpt.dox, [1344](#)
- hal_gpt.h, [1364](#)
- hal_gpt_lld.c, [1491](#)
- hal_gpt_lld.h, [1492](#)
- hal_i2c.c, [1457](#)
- hal_i2c.dox, [1344](#)
- hal_i2c.h, [1366](#)
- hal_i2c_config, [1177](#)
 - dummy, [1178](#)
- hal_i2c_driver, [1178](#)
 - config, [1179](#)
 - errors, [1179](#)
 - mutex, [1179](#)
 - state, [1179](#)
- hal_i2c_lld.c, [1493](#)
- hal_i2c_lld.h, [1494](#)
- hal_i2s.c, [1458](#)
- hal_i2s.dox, [1344](#)
- hal_i2s.h, [1367](#)
- hal_i2s_config, [1180](#)
 - end_cb, [1181](#)
 - rx_buffer, [1180](#)
 - size, [1181](#)
 - tx_buffer, [1180](#)
- hal_i2s_driver, [1181](#)
 - config, [1182](#)
 - state, [1182](#)
- hal_i2s_lld.c, [1495](#)
- hal_i2s_lld.h, [1496](#)
- hal_icu.c, [1459](#)
- hal_icu.dox, [1344](#)
- hal_icu.h, [1369](#)
- hal_icu_lld.c, [1496](#)
- hal_icu_lld.h, [1497](#)
- hal_interfaces.dox, [1344](#)
- hal_ioblock.dox, [1344](#)
- hal_ioblock.h, [1370](#)
- hal_lld.c, [1498](#)
- hal_lld.h, [1499](#)
- hal_lld_clock_switch_mode
 - HAL Driver, [67](#)
- hal_lld_get_clock_point
 - HAL Driver, [67](#)
- hal_lld_init
 - HAL Driver, [67](#)
- HAL_LLD_SELECT_SPI_V2
 - HAL Driver, [63](#)
- HAL_LLD_USE_CLOCK_MANAGEMENT
 - HAL Driver, [63](#)
- hal_mac.c, [1459](#)
- hal_mac.dox, [1344](#)
- hal_mac.h, [1371](#)
- hal_mac_config, [1182](#)
- hal_mac_driver, [1183](#)
 - arg, [1185](#)
 - cb, [1184](#)
 - config, [1184](#)
 - es, [1184](#)
 - flags, [1184](#)
 - rdqueue, [1184](#)
 - state, [1184](#)
 - tdqueue, [1184](#)
- hal_mac_lld.c, [1499](#)
- hal_mac_lld.h, [1500](#)
- hal_mac_receive_descriptor, [1185](#)
 - offset, [1186](#)
 - size, [1186](#)
- hal_mac_transmit_descriptor, [1186](#)
 - offset, [1187](#)
 - size, [1187](#)
- hal_mfs.c, [1421](#)
- hal_mfs.dox, [1344](#)
- hal_mfs.h, [1423](#)
- hal_mii.dox, [1344](#)
- hal_mii.h, [1373](#)
- hal_mmc_spi.c, [1460](#)
- hal_mmc_spi.dox, [1344](#)
- hal_mmc_spi.h, [1376](#)
- hal_mmcsd.c, [1462](#)
- hal_mmcsd.dox, [1344](#)
- hal_mmcsd.h, [1377](#)
- hal_norflash.dox, [1344](#)
- hal_objects.dox, [1344](#)
- hal_objects.h, [1382](#)
- hal_pal.c, [1462](#)
- hal_pal.dox, [1344](#)
- hal_pal.h, [1382](#)
- hal_pal_lld.c, [1502](#)
- hal_pal_lld.h, [1502](#)
- hal_peripherals.dox, [1344](#)
- hal_persistent.h, [1386](#)
- hal_pwm.c, [1463](#)
- hal_pwm.dox, [1344](#)
- hal_pwm.h, [1387](#)
- hal_pwm_lld.c, [1504](#)
- hal_pwm_lld.h, [1505](#)
- hal_queues.c, [1464](#)
- hal_queues.dox, [1344](#)
- hal_queues.h, [1388](#)
- HAL_RET_CONFIG_ERROR

- HAL Driver, 60
- HAL_RET_HW_BUSY
 - HAL Driver, 60
- HAL_RET_HW_FAILURE
 - HAL Driver, 61
- HAL_RET_IS_INVALID
 - HAL Driver, 61
- HAL_RET_NO_RESOURCE
 - HAL Driver, 60
- HAL_RET_SUCCESS
 - HAL Driver, 60
- HAL_RET_UNKNOWN_CTL
 - HAL Driver, 61
- hal_rtc.c, 1465
- hal_rtc.dox, 1344
- hal_rtc.h, 1390
- hal_rtc_lld.c, 1506
- hal_rtc_lld.h, 1506
- hal_sdc.c, 1466
- hal_sdc.dox, 1344
- hal_sdc.h, 1392
- hal_sdc_lld.c, 1507
- hal_sdc_lld.h, 1508
- hal_serial.c, 1467
- hal_serial.dox, 1344
- hal_serial.h, 1393
- hal_serial_config, 1187
 - speed, 1188
- hal_serial_driver, 1188
 - vmt, 1190
- hal_serial_lld.c, 1510
- hal_serial_lld.h, 1511
- hal_serial_nor.c, 1425
- hal_serial_nor.h, 1426
- hal_serial_usb.c, 1468
- hal_serial_usb.dox, 1344
- hal_serial_usb.h, 1395
- hal_sio.c, 1470
- hal_sio.dox, 1344
- hal_sio.h, 1397
- hal_sio_config, 1190
- hal_sio_driver, 1191
 - arg, 1193
 - cb, 1192
 - config, 1192
 - enabled, 1192
 - state, 1192
 - sync_rx, 1193
 - sync_rxidle, 1193
 - sync_tx, 1193
 - sync_txend, 1193
 - vmt, 1192
- hal_sio_lld.c, 1511
- hal_sio_lld.h, 1512
- hal_spi.c, 1471
- hal_spi.dox, 1344
- hal_spi.h, 1400
- hal_spi_config, 1194
 - circular, 1195
 - data_cb, 1196
 - end_cb, 1195
 - error_cb, 1196
 - slave, 1196
 - ssline, 1195
 - ssmask, 1195
 - sspad, 1195
 - ssport, 1195
- hal_spi_driver, 1196
 - config, 1198
 - mutex, 1198
 - state, 1198
 - sync_transfer, 1198
 - thread, 1198
- hal_spi_lld.c, 1514
- hal_spi_lld.h, 1515
- hal_spi_v1.dox, 1344
- hal_spi_v1.h, 1400
- hal_spi_v1.inc, 1471
- hal_spi_v2.dox, 1344
- hal_spi_v2.h, 1403
- hal_spi_v2.inc, 1472
- hal_spi_v2_lld.c, 1516
- hal_spi_v2_lld.h, 1516
- hal_st.c, 1473
- hal_st.dox, 1344
- hal_st.h, 1405
- hal_st_lld.c, 1517
- hal_st_lld.h, 1518
- hal_streams.h, 1406
- HAL_SUCCESS
 - HAL Driver, 60
- hal_trng.c, 1474
- hal_trng.dox, 1344
- hal_trng.h, 1407
- hal_trng_config, 1198
- hal_trng_driver, 1199
 - config, 1199
 - state, 1199
- hal_trng_lld.c, 1518
- hal_trng_lld.h, 1519
- hal_uart.c, 1475
- hal_uart.dox, 1344
- hal_uart.h, 1408
- hal_uart_config, 1200
 - rxchar_cb, 1201
 - rxend_cb, 1201
 - rxerr_cb, 1201
 - txend1_cb, 1201
 - txend2_cb, 1201
- hal_uart_driver, 1202
 - config, 1203
 - early, 1203
 - mutex, 1204
 - rxstate, 1203
 - state, 1203
 - threadrx, 1203

- threadtx, [1203](#)
- txstate, [1203](#)
- hal_uart_lld.c, [1520](#)
- hal_uart_lld.h, [1520](#)
- hal_usb.c, [1476](#)
- hal_usb.dox, [1344](#)
- hal_usb.h, [1410](#)
- hal_usb_cdc.dox, [1344](#)
- hal_usb_cdc.h, [1414](#)
- hal_usb_lld.c, [1522](#)
 - in, [1523](#)
 - out, [1523](#)
- hal_usb_lld.h, [1523](#)
- HAL_USE_ADC
 - Configuration, [46](#)
 - HAL Driver, [61](#)
- HAL_USE_CAN
 - Configuration, [46](#)
 - HAL Driver, [61](#)
- HAL_USE_CRY
 - Configuration, [46](#)
 - HAL Driver, [61](#)
- HAL_USE_DAC
 - Configuration, [46](#)
 - HAL Driver, [61](#)
- HAL_USE_EFL
 - Configuration, [46](#)
 - HAL Driver, [62](#)
- HAL_USE_GPT
 - Configuration, [46](#)
 - HAL Driver, [62](#)
- HAL_USE_I2C
 - Configuration, [46](#)
 - HAL Driver, [62](#)
- HAL_USE_I2S
 - Configuration, [46](#)
 - HAL Driver, [62](#)
- HAL_USE_ICU
 - Configuration, [47](#)
 - HAL Driver, [62](#)
- HAL_USE_MAC
 - Configuration, [47](#)
 - HAL Driver, [62](#)
- HAL_USE_MMC_SPI
 - Configuration, [47](#)
- HAL_USE_PAL
 - Configuration, [45](#)
 - HAL Driver, [61](#)
- HAL_USE_PWM
 - Configuration, [47](#)
 - HAL Driver, [62](#)
- HAL_USE_RTC
 - Configuration, [47](#)
 - HAL Driver, [62](#)
- HAL_USE_SDC
 - Configuration, [47](#)
 - HAL Driver, [62](#)
- HAL_USE_SERIAL
 - Configuration, [47](#)
 - HAL Driver, [62](#)
- HAL_USE_SERIAL_USB
 - Configuration, [47](#)
- HAL_USE_SIO
 - Configuration, [48](#)
 - HAL Driver, [63](#)
- HAL_USE_SPI
 - Configuration, [48](#)
 - HAL Driver, [63](#)
- HAL_USE_TRNG
 - Configuration, [48](#)
 - HAL Driver, [63](#)
- HAL_USE_UART
 - Configuration, [48](#)
 - HAL Driver, [63](#)
- HAL_USE_USB
 - Configuration, [48](#)
 - HAL Driver, [63](#)
- HAL_USE_WDG
 - Configuration, [48](#)
 - HAL Driver, [63](#)
- HAL_USE_WSPI
 - Configuration, [48](#)
 - HAL Driver, [63](#)
- hal_wdg.c, [1477](#)
- hal_wdg.dox, [1344](#)
- hal_wdg.h, [1416](#)
- hal_wdg_lld.c, [1525](#)
- hal_wdg_lld.h, [1526](#)
- hal_wspi.c, [1478](#)
- hal_wspi.dox, [1344](#)
- hal_wspi.h, [1416](#)
- hal_wspi_config, [1204](#)
 - end_cb, [1205](#)
 - error_cb, [1205](#)
- hal_wspi_driver, [1205](#)
 - config, [1206](#)
 - mutex, [1206](#)
 - state, [1206](#)
 - thread, [1206](#)
- hal_wspi_lld.c, [1526](#)
- hal_wspi_lld.h, [1527](#)
- hal_xsnor_base, [1207](#)
 - commands, [1208](#)
 - config, [1208](#)
 - descriptor, [1209](#)
 - fls, [1208](#)
 - mutex, [1209](#)
 - state, [1208](#)
 - vmt, [1208](#)
- hal_xsnor_base.c, [1441](#)
- hal_xsnor_base.h, [1439](#)
- hal_xsnor_base_c, [1209](#)
 - Base Serial NOR Driver class, [837](#)
- hal_xsnor_base_vmt, [1210](#)
 - dispose, [1211](#)
 - init, [1211](#)

- mmap_off, [1212](#)
 - mmap_on, [1212](#)
 - program, [1211](#)
 - query_erase, [1211](#)
 - read, [1211](#)
 - start_erase_all, [1211](#)
 - start_erase_sector, [1211](#)
 - verify_erase, [1212](#)
- hal_xsnor_macronix_mx25, [1212](#)
 - commands, [1214](#)
 - config, [1214](#)
 - descriptor, [1215](#)
 - fls, [1214](#)
 - mutex, [1214](#)
 - state, [1214](#)
 - vmt, [1214](#)
- hal_xsnor_macronix_mx25.c, [1428](#)
- hal_xsnor_macronix_mx25.h, [1429](#)
- hal_xsnor_macronix_mx25_c, [1215](#)
 - Macronix MX25 SNOR Driver, [881](#)
- hal_xsnor_macronix_mx25_impl.inc, [1430](#)
- hal_xsnor_macronix_mx25_vmt, [1216](#)
 - dispose, [1217](#)
 - init, [1217](#)
 - mmap_off, [1218](#)
 - mmap_on, [1218](#)
 - program, [1217](#)
 - query_erase, [1217](#)
 - read, [1217](#)
 - start_erase_all, [1217](#)
 - start_erase_sector, [1217](#)
 - verify_erase, [1218](#)
- hal_xsnor_micron_n25q, [1218](#)
 - commands, [1220](#)
 - config, [1220](#)
 - descriptor, [1221](#)
 - fls, [1220](#)
 - mutex, [1220](#)
 - state, [1220](#)
 - vmt, [1220](#)
- hal_xsnor_micron_n25q.c, [1434](#)
- hal_xsnor_micron_n25q.h, [1434](#)
- hal_xsnor_micron_n25q_c, [1221](#)
 - Micron N25Q Driver, [902](#)
- hal_xsnor_micron_n25q_impl.inc, [1435](#)
- hal_xsnor_micron_n25q_vmt, [1222](#)
 - dispose, [1223](#)
 - init, [1223](#)
 - mmap_off, [1224](#)
 - mmap_on, [1224](#)
 - program, [1223](#)
 - query_erase, [1223](#)
 - read, [1223](#)
 - start_erase_all, [1223](#)
 - start_erase_sector, [1223](#)
 - verify_erase, [1224](#)
- halclkcfg_t, [1224](#)
 - dummy, [1225](#)
- halclkpt_t
 - HAL Driver, [64](#)
- halClockGetPointX
 - HAL Driver, [65](#)
- halClockSwitchMode
 - HAL Driver, [66](#)
- halconf.h, [1528](#)
- halfreq_t
 - HAL Driver, [64](#)
- hallnit
 - HAL Driver, [64](#)
- halted_status
 - USB Driver, [684](#)
- hdr32
 - mfs_bank_header_t, [1241](#)
 - mfs_data_header_t, [1243](#)
- hdr8
 - mfs_bank_header_t, [1241](#)
 - mfs_data_header_t, [1243](#)
- HMACSHA256Context, [1225](#)
 - dummy, [1225](#)
- HMACSHA512Context, [1226](#)
 - dummy, [1226](#)
- hscfg
 - mmc_spi_config_t, [1255](#)
- I/O Buffers Queues, [980](#)
 - BQ_BUFFER_SIZE, [982](#)
 - bqGetLinkX, [983](#)
 - bqIsSuspendedX, [984](#)
 - bqnotify_t, [988](#)
 - bqResumeX, [985](#)
 - bqSetLinkX, [984](#)
 - bqSizeX, [983](#)
 - bqSpaceI, [983](#)
 - bqSuspendI, [985](#)
 - BUFFERS_CHUNKS_SIZE, [982](#)
 - ibqGetEmptyBufferI, [990](#)
 - ibqGetFullBufferTimeout, [991](#)
 - ibqGetFullBufferTimeoutS, [992](#)
 - ibqGetTimeout, [994](#)
 - ibqIsEmptyI, [986](#)
 - ibqIsFullI, [986](#)
 - ibqObjectInit, [989](#)
 - ibqPostFullBufferI, [990](#)
 - ibqReadTimeout, [995](#)
 - ibqReleaseEmptyBuffer, [993](#)
 - ibqReleaseEmptyBufferS, [993](#)
 - ibqResetI, [989](#)
 - input_buffers_queue_t, [988](#)
 - io_buffers_queue_t, [988](#)
 - obqFlush, [1004](#)
 - obqGetEmptyBufferTimeout, [998](#)
 - obqGetEmptyBufferTimeoutS, [999](#)
 - obqGetFullBufferI, [997](#)
 - obqIsEmptyI, [987](#)
 - obqIsFullI, [987](#)
 - obqObjectInit, [996](#)
 - obqPostFullBuffer, [1000](#)

- obqPostFullBufferS, [1001](#)
- obqPutTimeout, [1001](#)
- obqReleaseEmptyBufferI, [998](#)
- obqResetI, [997](#)
- obqTryFlushI, [1003](#)
- obqWriteTimeout, [1002](#)
- output_buffers_queue_t, [988](#)
- I/O Bytes Queues, [1030](#)
 - input_queue_t, [1040](#)
 - io_queue_t, [1040](#)
 - iq_read, [1041](#)
 - iqGet, [1037](#)
 - iqGetEmptyI, [1035](#)
 - iqGetFullI, [1035](#)
 - iqGetI, [1044](#)
 - iqGetTimeout, [1044](#)
 - iqIsEmptyI, [1036](#)
 - iqIsFullI, [1036](#)
 - iqObjectInit, [1042](#)
 - iqPutI, [1043](#)
 - iqReadI, [1045](#)
 - iqReadTimeout, [1046](#)
 - iqResetI, [1042](#)
 - oq_write, [1041](#)
 - oqGetEmptyI, [1038](#)
 - oqGetFullI, [1037](#)
 - oqGetI, [1050](#)
 - oqIsEmptyI, [1038](#)
 - oqIsFullI, [1039](#)
 - oqObjectInit, [1047](#)
 - oqPut, [1039](#)
 - oqPutI, [1049](#)
 - oqPutTimeout, [1049](#)
 - oqResetI, [1048](#)
 - oqWriteI, [1051](#)
 - oqWriteTimeout, [1052](#)
 - output_queue_t, [1040](#)
 - Q_EMPTY, [1033](#)
 - Q_FULL, [1033](#)
 - Q_OK, [1032](#)
 - Q_RESET, [1032](#)
 - Q_TIMEOUT, [1032](#)
 - qGetLink, [1034](#)
 - qnotify_t, [1040](#)
 - qSetLink, [1034](#)
 - qSizeX, [1033](#)
 - qSpacel, [1033](#)
- I2C Driver, [233](#)
 - _i2c_wakeup_error_isr, [238](#)
 - _i2c_wakeup_isr, [238](#)
 - I2C_ACK_FAILURE, [237](#)
 - I2C_ACTIVE_RX, [241](#)
 - I2C_ACTIVE_TX, [241](#)
 - I2C_ARBITRATION_LOST, [237](#)
 - I2C_BUS_ERROR, [237](#)
 - I2C_ENABLE_SLAVE_MODE, [238](#)
 - i2c_llid_get_errors, [240](#)
 - i2c_llid_init, [250](#)
 - i2c_llid_master_receive_timeout, [251](#)
 - i2c_llid_master_transmit_timeout, [252](#)
 - i2c_llid_start, [250](#)
 - i2c_llid_stop, [251](#)
 - I2C_LOCKED, [241](#)
 - I2C_NO_ERROR, [237](#)
 - I2C_OVERRUN, [237](#)
 - I2C_PEC_ERROR, [237](#)
 - I2C_READY, [241](#)
 - I2C_SMB_ALERT, [237](#)
 - I2C_STOP, [241](#)
 - I2C_SUPPORTS_SLAVE_MODE, [238](#)
 - I2C_TIMEOUT, [237](#)
 - I2C_UNINIT, [241](#)
 - I2C_USE_MUTUAL_EXCLUSION, [238](#)
 - i2cAcquireBus, [246](#)
 - i2caddr_t, [241](#)
 - I2CConfig, [241](#)
 - I2CD1, [253](#)
 - I2CDriver, [241](#)
 - i2cflags_t, [241](#)
 - i2cGetErrors, [244](#)
 - i2cInit, [242](#)
 - i2cMasterReceive, [239](#)
 - i2cMasterReceiveTimeout, [245](#)
 - i2cMasterTransmit, [239](#)
 - i2cMasterTransmitTimeout, [244](#)
 - i2cObjectInit, [242](#)
 - i2cReleaseBus, [247](#)
 - i2cSlavesAnswerRequired, [239](#)
 - i2cSlaveMatchAddress, [247](#)
 - i2cSlaveReceiveTimeout, [248](#)
 - i2cSlaveTransmitTimeout, [249](#)
 - i2cStart, [242](#)
 - i2cstate_t, [241](#)
 - i2cStop, [243](#)
 - PLATFORM_I2C_USE_I2C1, [240](#)
- I2C_ACK_FAILURE
 - I2C Driver, [237](#)
- I2C_ACTIVE_RX
 - I2C Driver, [241](#)
- I2C_ACTIVE_TX
 - I2C Driver, [241](#)
- I2C_ARBITRATION_LOST
 - I2C Driver, [237](#)
- I2C_BUS_ERROR
 - I2C Driver, [237](#)
- I2C_ENABLE_SLAVE_MODE
 - Configuration, [50](#)
 - I2C Driver, [238](#)
- i2c_llid_get_errors
 - I2C Driver, [240](#)
- i2c_llid_init
 - I2C Driver, [250](#)
- i2c_llid_master_receive_timeout
 - I2C Driver, [251](#)
- i2c_llid_master_transmit_timeout
 - I2C Driver, [252](#)

- i2c_llid_start
 - I2C Driver, [250](#)
- i2c_llid_stop
 - I2C Driver, [251](#)
- I2C_LOCKED
 - I2C Driver, [241](#)
- I2C_NO_ERROR
 - I2C Driver, [237](#)
- I2C_OVERRUN
 - I2C Driver, [237](#)
- I2C_PEC_ERROR
 - I2C Driver, [237](#)
- I2C_READY
 - I2C Driver, [241](#)
- I2C_SMB_ALERT
 - I2C Driver, [237](#)
- I2C_STOP
 - I2C Driver, [241](#)
- I2C_SUPPORTS_SLAVE_MODE
 - I2C Driver, [238](#)
- I2C_TIMEOUT
 - I2C Driver, [237](#)
- I2C_UNINIT
 - I2C Driver, [241](#)
- I2C_USE_MUTUAL_EXCLUSION
 - Configuration, [51](#)
 - I2C Driver, [238](#)
- i2cAcquireBus
 - I2C Driver, [246](#)
- i2caddr_t
 - I2C Driver, [241](#)
- I2CConfig
 - I2C Driver, [241](#)
- I2CD1
 - I2C Driver, [253](#)
- I2CDriver
 - I2C Driver, [241](#)
- i2cflags_t
 - I2C Driver, [241](#)
- i2cGetErrors
 - I2C Driver, [244](#)
- i2cInit
 - I2C Driver, [242](#)
- i2cMasterReceive
 - I2C Driver, [239](#)
- i2cMasterReceiveTimeout
 - I2C Driver, [245](#)
- i2cMasterTransmit
 - I2C Driver, [239](#)
- i2cMasterTransmitTimeout
 - I2C Driver, [244](#)
- i2cObjectInit
 - I2C Driver, [242](#)
- i2cReleaseBus
 - I2C Driver, [247](#)
- i2cSlaveIsAnswerRequired
 - I2C Driver, [239](#)
- i2cSlaveMatchAddress
 - I2C Driver, [247](#)
- i2cSlaveReceiveTimeout
 - I2C Driver, [248](#)
- i2cSlaveTransmitTimeout
 - I2C Driver, [249](#)
- i2cStart
 - I2C Driver, [242](#)
- i2cstate_t
 - I2C Driver, [241](#)
- i2cStop
 - I2C Driver, [243](#)
- I2S Driver, [253](#)
 - _i2s_isr_full_code, [257](#)
 - _i2s_isr_half_code, [256](#)
 - I2S_ACTIVE, [259](#)
 - I2S_COMPLETE, [259](#)
 - i2s_llid_config_fields, [258](#)
 - i2s_llid_driver_fields, [257](#)
 - i2s_llid_init, [262](#)
 - i2s_llid_start, [262](#)
 - i2s_llid_start_exchange, [263](#)
 - i2s_llid_stop, [263](#)
 - i2s_llid_stop_exchange, [263](#)
 - I2S_MODE_MASTER, [255](#)
 - I2S_MODE_SLAVE, [255](#)
 - I2S_READY, [259](#)
 - I2S_STOP, [259](#)
 - I2S_UNINIT, [259](#)
 - i2scallback_t, [258](#)
 - I2SConfig, [258](#)
 - I2SD1, [264](#)
 - I2SDriver, [258](#)
 - i2sInit, [259](#)
 - i2sIsBufferComplete, [255](#)
 - i2sObjectInit, [259](#)
 - i2sStart, [259](#)
 - i2sStartExchange, [261](#)
 - i2sStartExchangeI, [255](#)
 - i2sstate_t, [258](#)
 - i2sStop, [260](#)
 - i2sStopExchange, [261](#)
 - i2sStopExchangeI, [256](#)
 - PLATFORM_I2S_USE_I2S1, [257](#)
- I2S_ACTIVE
 - I2S Driver, [259](#)
- I2S_COMPLETE
 - I2S Driver, [259](#)
- i2s_llid_config_fields
 - I2S Driver, [258](#)
- i2s_llid_driver_fields
 - I2S Driver, [257](#)
- i2s_llid_init
 - I2S Driver, [262](#)
- i2s_llid_start
 - I2S Driver, [262](#)
- i2s_llid_start_exchange
 - I2S Driver, [263](#)
- i2s_llid_stop

- I2S Driver, [263](#)
- i2s_llid_stop_exchange
 - I2S Driver, [263](#)
- I2S_MODE_MASTER
 - I2S Driver, [255](#)
- I2S_MODE_SLAVE
 - I2S Driver, [255](#)
- I2S_READY
 - I2S Driver, [259](#)
- I2S_STOP
 - I2S Driver, [259](#)
- I2S_UNINIT
 - I2S Driver, [259](#)
- i2scallback_t
 - I2S Driver, [258](#)
- I2SConfig
 - I2S Driver, [258](#)
- I2SD1
 - I2S Driver, [264](#)
- I2SDriver
 - I2S Driver, [258](#)
- i2sInit
 - I2S Driver, [259](#)
- i2sIsBufferComplete
 - I2S Driver, [255](#)
- i2sObjectInit
 - I2S Driver, [259](#)
- i2sStart
 - I2S Driver, [259](#)
- i2sStartExchange
 - I2S Driver, [261](#)
- i2sStartExchangel
 - I2S Driver, [255](#)
- i2sstate_t
 - I2S Driver, [258](#)
- i2sStop
 - I2S Driver, [260](#)
- i2sStopExchange
 - I2S Driver, [261](#)
- i2sStopExchangel
 - I2S Driver, [256](#)
- ibnotify
 - Serial over USB Driver, [822](#)
- ibqGetEmptyBufferI
 - I/O Buffers Queues, [990](#)
- ibqGetFullBufferTimeout
 - I/O Buffers Queues, [991](#)
- ibqGetFullBufferTimeoutS
 - I/O Buffers Queues, [992](#)
- ibqGetTimeout
 - I/O Buffers Queues, [994](#)
- ibqIsEmptyI
 - I/O Buffers Queues, [986](#)
- ibqIsFullI
 - I/O Buffers Queues, [986](#)
- ibqObjectInit
 - I/O Buffers Queues, [989](#)
- ibqPostFullBufferI
 - I/O Buffers Queues, [990](#)
- ibqReadTimeout
 - I/O Buffers Queues, [995](#)
- ibqReleaseEmptyBuffer
 - I/O Buffers Queues, [993](#)
- ibqReleaseEmptyBufferS
 - I/O Buffers Queues, [993](#)
- ibqResetI
 - I/O Buffers Queues, [989](#)
- ICU Driver, [264](#)
 - _icu_isr_invoke_overflow_cb, [272](#)
 - _icu_isr_invoke_period_cb, [271](#)
 - _icu_isr_invoke_width_cb, [271](#)
 - ICU_ACTIVE, [275](#)
 - ICU_INPUT_ACTIVE_HIGH, [275](#)
 - ICU_INPUT_ACTIVE_LOW, [275](#)
 - icu_llid_are_notifications_enabled, [273](#)
 - icu_llid_disable_notifications, [283](#)
 - icu_llid_enable_notifications, [282](#)
 - icu_llid_get_period, [273](#)
 - icu_llid_get_width, [272](#)
 - icu_llid_init, [280](#)
 - icu_llid_start, [281](#)
 - icu_llid_start_capture, [281](#)
 - icu_llid_stop, [281](#)
 - icu_llid_stop_capture, [282](#)
 - icu_llid_wait_capture, [282](#)
 - ICU_READY, [275](#)
 - ICU_STOP, [275](#)
 - ICU_UNINIT, [275](#)
 - ICU_USE_OVERFLOW_SCALING, [267](#)
 - ICU_WAITING, [275](#)
 - icuAreNotificationsEnabledX, [269](#)
 - icucallback_t, [274](#)
 - icucnt_t, [274](#)
 - ICUD1, [284](#)
 - icuDisableNotifications, [280](#)
 - icuDisableNotificationsI, [269](#)
 - ICUDriver, [274](#)
 - icuEnableNotifications, [279](#)
 - icuEnableNotificationsI, [268](#)
 - icufreq_t, [274](#)
 - icuGetPeriodX, [270](#)
 - icuGetWidthX, [270](#)
 - iculnit, [275](#)
 - icumode_t, [275](#)
 - icuObjectInit, [275](#)
 - icuStart, [276](#)
 - icuStartCapture, [277](#)
 - icuStartCaptureI, [267](#)
 - icustate_t, [274](#)
 - icuStop, [276](#)
 - icuStopCapture, [278](#)
 - icuStopCaptureI, [268](#)
 - icuWaitCapture, [277](#)
 - PLATFORM_ICU_USE_ICU1, [272](#)
- ICU_ACTIVE
 - ICU Driver, [275](#)

- ICU_INPUT_ACTIVE_HIGH
 - ICU Driver, [275](#)
- ICU_INPUT_ACTIVE_LOW
 - ICU Driver, [275](#)
- icu_ild_are_notifications_enabled
 - ICU Driver, [273](#)
- icu_ild_disable_notifications
 - ICU Driver, [283](#)
- icu_ild_enable_notifications
 - ICU Driver, [282](#)
- icu_ild_get_period
 - ICU Driver, [273](#)
- icu_ild_get_width
 - ICU Driver, [272](#)
- icu_ild_init
 - ICU Driver, [280](#)
- icu_ild_start
 - ICU Driver, [281](#)
- icu_ild_start_capture
 - ICU Driver, [281](#)
- icu_ild_stop
 - ICU Driver, [281](#)
- icu_ild_stop_capture
 - ICU Driver, [282](#)
- icu_ild_wait_capture
 - ICU Driver, [282](#)
- ICU_READY
 - ICU Driver, [275](#)
- ICU_STOP
 - ICU Driver, [275](#)
- ICU_UNINIT
 - ICU Driver, [275](#)
- ICU_USE_OVERFLOW_SCALING
 - ICU Driver, [267](#)
- ICU_WAITING
 - ICU Driver, [275](#)
- icuAreNotificationsEnabledX
 - ICU Driver, [269](#)
- icucallback_t
 - ICU Driver, [274](#)
- icucnt_t
 - ICU Driver, [274](#)
- ICUConfig, [1226](#)
 - frequency, [1227](#)
 - mode, [1227](#)
 - overflow_cb, [1228](#)
 - period_cb, [1228](#)
 - width_cb, [1228](#)
- ICUD1
 - ICU Driver, [284](#)
- icuDisableNotifications
 - ICU Driver, [280](#)
- icuDisableNotificationsI
 - ICU Driver, [269](#)
- ICUDriver, [1228](#)
 - config, [1229](#)
 - ICU Driver, [274](#)
 - state, [1229](#)
- icuEnableNotifications
 - ICU Driver, [279](#)
- icuEnableNotificationsI
 - ICU Driver, [268](#)
- icufreq_t
 - ICU Driver, [274](#)
- icuGetPeriodX
 - ICU Driver, [270](#)
- icuGetWidthX
 - ICU Driver, [270](#)
- icuinitt
 - ICU Driver, [275](#)
- icumode_t
 - ICU Driver, [275](#)
- icuObjectInit
 - ICU Driver, [275](#)
- icuStart
 - ICU Driver, [276](#)
- icuStartCapture
 - ICU Driver, [277](#)
- icuStartCaptureI
 - ICU Driver, [267](#)
- icustate_t
 - ICU Driver, [274](#)
- icuStop
 - ICU Driver, [276](#)
- icuStopCapture
 - ICU Driver, [278](#)
- icuStopCaptureI
 - ICU Driver, [268](#)
- icuWaitCapture
 - ICU Driver, [277](#)
- id
 - mfs_data_header_t, [1243](#)
 - mfs_transaction_op_t, [1247](#)
- IDE
 - CANRxFrame, [1136](#)
 - CANTxFrame, [1138](#)
- Improved Serial NOR Driver, [831](#)
- in
 - hal_usb_ild.c, [1523](#)
 - USB Driver, [684](#)
- in_cb
 - USBEndpointConfig, [1324](#)
- in_maxsize
 - USBEndpointConfig, [1325](#)
- in_params
 - USBDriver, [1320](#)
- in_state
 - USBEndpointConfig, [1325](#)
- init
 - hal_xsnor_base_vmt, [1211](#)
 - hal_xsnor_macronix_mx25_vmt, [1217](#)
 - hal_xsnor_micron_n25q_vmt, [1223](#)
- Inner Code, [980](#)
- Input Formatter Utility, [973](#)
 - chscanf, [975](#)
 - CHSCANF_USE_FLOAT, [974](#)

- chsnscanf, [976](#)
- chvscanf, [974](#)
- chvsnscanf, [977](#)
- sym_to_val, [974](#)
- input_buffers_queue_t
 - I/O Buffers Queues, [988](#)
- input_queue_t
 - I/O Bytes Queues, [1040](#)
- instance_offset
 - flash_interface_vmt, [1154](#)
- int_in
 - SerialUSBConfig, [1285](#)
- Interfaces and Classes, [916](#)
- Introduction, [1](#)
- io_buffers_queue, [1229](#)
 - bcounter, [1231](#)
 - bn, [1232](#)
 - brdptr, [1231](#)
 - bsize, [1232](#)
 - btpr, [1231](#)
 - buffers, [1232](#)
 - bwrptr, [1231](#)
 - link, [1233](#)
 - notify, [1232](#)
 - ptr, [1232](#)
 - suspended, [1231](#)
 - top, [1232](#)
 - waiting, [1231](#)
- io_buffers_queue_t
 - I/O Buffers Queues, [988](#)
- io_queue, [1233](#)
 - q_buffer, [1234](#)
 - q_counter, [1234](#)
 - q_link, [1235](#)
 - q_notify, [1235](#)
 - q_rdprr, [1235](#)
 - q_top, [1234](#)
 - q_waiting, [1234](#)
 - q_wrptr, [1235](#)
- io_queue_t
 - I/O Bytes Queues, [1040](#)
- IOBus, [1235](#)
 - mask, [1236](#)
 - offset, [1236](#)
 - portid, [1236](#)
- IOBUS_DECL
 - PAL Driver, [317](#)
- ioline_t
 - PAL Driver, [351](#)
- iomode_t
 - PAL Driver, [351](#)
- iopadid_t
 - PAL Driver, [351](#)
- IOPORT1
 - PAL Driver, [340](#)
- ioportid_t
 - PAL Driver, [351](#)
- ioportmask_t
 - PAL Driver, [351](#)
- iq_read
 - I/O Bytes Queues, [1041](#)
- iqGet
 - I/O Bytes Queues, [1037](#)
- iqGetEmptyI
 - I/O Bytes Queues, [1035](#)
- iqGetFullI
 - I/O Bytes Queues, [1035](#)
- iqGetI
 - I/O Bytes Queues, [1044](#)
- iqGetTimeout
 - I/O Bytes Queues, [1044](#)
- iqIsEmptyI
 - I/O Bytes Queues, [1036](#)
- iqIsFullI
 - I/O Bytes Queues, [1036](#)
- iqObjectInit
 - I/O Bytes Queues, [1042](#)
- iqPutI
 - I/O Bytes Queues, [1043](#)
- iqReadI
 - I/O Bytes Queues, [1045](#)
- iqReadTimeout
 - I/O Bytes Queues, [1046](#)
- iqResetI
 - I/O Bytes Queues, [1042](#)
- LC_PARITY_EVEN
 - USB CDC Header, [1072](#)
- LC_PARITY_MARK
 - USB CDC Header, [1073](#)
- LC_PARITY_NONE
 - USB CDC Header, [1072](#)
- LC_PARITY_ODD
 - USB CDC Header, [1072](#)
- LC_PARITY_SPACE
 - USB CDC Header, [1073](#)
- LC_STOP_1
 - USB CDC Header, [1072](#)
- LC_STOP_1P5
 - USB CDC Header, [1072](#)
- LC_STOP_2
 - USB CDC Header, [1072](#)
- linecoding
 - Serial over USB Driver, [830](#)
- link
 - io_buffers_queue, [1233](#)
- long_to_string_with_divisor
 - Output Formatter Utility, [968](#)
- LPA_100
 - MII/RMII Header, [1066](#)
- LPA_100BASE4
 - MII/RMII Header, [1065](#)
- LPA_100FULL
 - MII/RMII Header, [1064](#)
- LPA_100HALF
 - MII/RMII Header, [1064](#)
- LPA_10FULL

- MII/RMII Header, [1064](#)
- LPA_10HALF
 - MII/RMII Header, [1064](#)
- LPA_DUPLEX
 - MII/RMII Header, [1065](#)
- LPA_LPACK
 - MII/RMII Header, [1065](#)
- LPA_NPAGE
 - MII/RMII Header, [1065](#)
- LPA_PAUSE_ASYM
 - MII/RMII Header, [1065](#)
- LPA_PAUSE_CAP
 - MII/RMII Header, [1065](#)
- LPA_RESV
 - MII/RMII Header, [1065](#)
- LPA_RFAULT
 - MII/RMII Header, [1065](#)
- LPA_SLCT
 - MII/RMII Header, [1064](#)
- lscfg
 - mmc_spi_config_t, [1255](#)
- MAC Driver, [284](#)
 - __mac_callback, [287](#)
 - __mac_rx_wakeup, [288](#)
 - __mac_tx_wakeup, [288](#)
 - ETHD1, [307](#)
 - MAC_ACTIVE, [295](#)
 - MAC_FLAGS_RX, [287](#)
 - MAC_FLAGS_TX, [287](#)
 - mac_llid_config_fields, [294](#)
 - mac_llid_driver_fields, [294](#)
 - mac_llid_get_next_receive_buffer, [307](#)
 - mac_llid_get_next_transmit_buffer, [306](#)
 - mac_llid_get_receive_descriptor, [303](#)
 - mac_llid_get_transmit_descriptor, [302](#)
 - mac_llid_init, [301](#)
 - mac_llid_poll_link_status, [305](#)
 - mac_llid_read_receive_descriptor, [305](#)
 - mac_llid_receive_descriptor_fields, [294](#)
 - mac_llid_release_receive_descriptor, [303](#)
 - mac_llid_release_transmit_descriptor, [303](#)
 - mac_llid_start, [301](#)
 - mac_llid_stop, [302](#)
 - mac_llid_transmit_descriptor_fields, [294](#)
 - mac_llid_write_transmit_descriptor, [305](#)
 - MAC_STOP, [295](#)
 - MAC_SUPPORTS_ZERO_COPY, [293](#)
 - MAC_UNINIT, [295](#)
 - MAC_USE_EVENTS, [287](#)
 - MAC_USE_ZERO_COPY, [287](#)
 - maccb_t, [295](#)
 - MACConfig, [294](#)
 - MACDriver, [294](#)
 - macGetAndClearEventsI, [298](#)
 - macGetEventSource, [289](#)
 - macGetNextReceiveBuffer, [293](#)
 - macGetNextTransmitBuffer, [292](#)
 - macGetReceiveDescriptorX, [290](#)
 - macGetTransmitDescriptorX, [289](#)
 - macInit, [296](#)
 - macObjectInit, [296](#)
 - macPollLinkStatus, [300](#)
 - macReadReceiveDescriptor, [292](#)
 - MACReceiveDescriptor, [295](#)
 - macReleaseReceiveDescriptor, [301](#)
 - macReleaseReceiveDescriptorX, [291](#)
 - macReleaseTransmitDescriptor, [301](#)
 - macReleaseTransmitDescriptorX, [290](#)
 - macSetCallbackX, [288](#)
 - macStart, [296](#)
 - macstate_t, [295](#)
 - macStop, [297](#)
 - MACTransmitDescriptor, [295](#)
 - macWaitReceiveDescriptor, [299](#)
 - macWaitTransmitDescriptor, [298](#)
 - macWriteTransmitDescriptor, [291](#)
 - PLATFORM_MAC_USE_MAC1, [294](#)
- MAC_ACTIVE
 - MAC Driver, [295](#)
- MAC_FLAGS_RX
 - MAC Driver, [287](#)
- MAC_FLAGS_TX
 - MAC Driver, [287](#)
- mac_llid_config_fields
 - MAC Driver, [294](#)
- mac_llid_driver_fields
 - MAC Driver, [294](#)
- mac_llid_get_next_receive_buffer
 - MAC Driver, [307](#)
- mac_llid_get_next_transmit_buffer
 - MAC Driver, [306](#)
- mac_llid_get_receive_descriptor
 - MAC Driver, [303](#)
- mac_llid_get_transmit_descriptor
 - MAC Driver, [302](#)
- mac_llid_init
 - MAC Driver, [301](#)
- mac_llid_poll_link_status
 - MAC Driver, [305](#)
- mac_llid_read_receive_descriptor
 - MAC Driver, [305](#)
- mac_llid_receive_descriptor_fields
 - MAC Driver, [294](#)
- mac_llid_release_receive_descriptor
 - MAC Driver, [303](#)
- mac_llid_release_transmit_descriptor
 - MAC Driver, [303](#)
- mac_llid_start
 - MAC Driver, [301](#)
- mac_llid_stop
 - MAC Driver, [302](#)
- mac_llid_transmit_descriptor_fields
 - MAC Driver, [294](#)
- mac_llid_write_transmit_descriptor
 - MAC Driver, [305](#)
- MAC_STOP

- MAC Driver, [295](#)
- MAC_SUPPORTS_ZERO_COPY
 - MAC Driver, [293](#)
- MAC_UNINIT
 - MAC Driver, [295](#)
- MAC_USE_EVENTS
 - Configuration, [51](#)
 - MAC Driver, [287](#)
- MAC_USE_ZERO_COPY
 - Configuration, [51](#)
 - MAC Driver, [287](#)
- maccb_t
 - MAC Driver, [295](#)
- MACConfig
 - MAC Driver, [294](#)
- MACDriver
 - MAC Driver, [294](#)
- macGetAndClearEventsI
 - MAC Driver, [298](#)
- macGetEventSource
 - MAC Driver, [289](#)
- macGetNextReceiveBuffer
 - MAC Driver, [293](#)
- macGetNextTransmitBuffer
 - MAC Driver, [292](#)
- macGetReceiveDescriptorX
 - MAC Driver, [290](#)
- macGetTransmitDescriptorX
 - MAC Driver, [289](#)
- macInit
 - MAC Driver, [296](#)
- macObjectInit
 - MAC Driver, [296](#)
- macPollLinkStatus
 - MAC Driver, [300](#)
- macReadReceiveDescriptor
 - MAC Driver, [292](#)
- MACReceiveDescriptor
 - MAC Driver, [295](#)
- macReleaseReceiveDescriptor
 - MAC Driver, [301](#)
- macReleaseReceiveDescriptorX
 - MAC Driver, [291](#)
- macReleaseTransmitDescriptor
 - MAC Driver, [301](#)
- macReleaseTransmitDescriptorX
 - MAC Driver, [290](#)
- Macronix MX25 SNOR Driver, [863](#)
 - __hal_xsnor_macronix_mx25_vmt, [893](#)
 - __mx25_dispose_impl, [882](#)
 - __mx25_init_impl, [883](#)
 - __mx25_mmap_off_impl, [888](#)
 - __mx25_mmap_on_impl, [888](#)
 - __mx25_objinit_impl, [881](#)
 - __mx25_program_impl, [884](#)
 - __mx25_query_erase_impl, [886](#)
 - __mx25_read_impl, [883](#)
 - __mx25_start_erase_all_impl, [885](#)
 - __mx25_start_erase_sector_impl, [886](#)
 - __mx25_verify_erase_impl, [887](#)
- CFG_C16_8DTR, [878](#)
- CFG_C16_8STR, [878](#)
- CFG_C16_A32_8DTR, [879](#)
- CFG_C16_A32_8STR, [879](#)
- CFG_C16_A32_DATA_8DTR, [880](#)
- CFG_C16_A32_DATA_8STR, [880](#)
- CFG_C16_DATA_8DTR, [880](#)
- CFG_C16_DATA_8STR, [879](#)
- CFG_C8_A32_DATA_SPI, [880](#)
- CFG_C8_A32_SPI, [879](#)
- CFG_C8_DATA_SPI, [879](#)
- CFG_C8_SPI, [878](#)
- cmd_8dtr, [892](#)
- cmd_8str, [891](#)
- CMD_OPI_8DTRD, [873](#)
- CMD_OPI_8READ, [873](#)
- CMD_OPI_BE, [874](#)
- CMD_OPI_CE, [874](#)
- CMD_OPI_DP, [874](#)
- CMD_OPI_ENSO, [874](#)
- CMD_OPI_ESFBR, [876](#)
- CMD_OPI_ESSPB, [876](#)
- CMD_OPI_EXSO, [875](#)
- CMD_OPI_GBLK, [877](#)
- CMD_OPI_GBULK, [877](#)
- CMD_OPI_NOP, [875](#)
- CMD_OPI_PASSULK, [877](#)
- CMD_OPI_PE_RESUME, [874](#)
- CMD_OPI_PE_SUSPEND, [874](#)
- CMD_OPI_PP, [873](#)
- CMD_OPI_RDCR, [875](#)
- CMD_OPI_RDCR2, [875](#)
- CMD_OPI_RDDPB, [877](#)
- CMD_OPI_RDFBR, [876](#)
- CMD_OPI_RDID, [873](#)
- CMD_OPI_RDLR, [876](#)
- CMD_OPI_RDPASS, [877](#)
- CMD_OPI_RDSCUR, [876](#)
- CMD_OPI_RDSFDP, [873](#)
- CMD_OPI_RDSPB, [876](#)
- CMD_OPI_RDSR, [875](#)
- CMD_OPI_RST, [875](#)
- CMD_OPI_RSTEN, [875](#)
- CMD_OPI_SBL, [874](#)
- CMD_OPI_SE, [874](#)
- CMD_OPI_WPSEL, [877](#)
- CMD_OPI_WRCR, [875](#)
- CMD_OPI_WRCR2, [875](#)
- CMD_OPI_WRDI, [874](#)
- CMD_OPI_WRDPB, [877](#)
- CMD_OPI_WREN, [874](#)
- CMD_OPI_WRFBR, [876](#)
- CMD_OPI_WRLR, [876](#)
- CMD_OPI_WRPASS, [877](#)
- CMD_OPI_WRSCUR, [876](#)
- CMD_OPI_WRSPB, [876](#)

CMD_OPI_WRSR, [875](#)
 cmd_reset_enable_8dtr, [893](#)
 cmd_reset_enable_8str, [892](#)
 cmd_reset_enable_spi, [892](#)
 cmd_reset_memory_8dtr, [893](#)
 cmd_reset_memory_8str, [892](#)
 cmd_reset_memory_spi, [892](#)
 cmd_spi, [891](#)
 CMD_SPI_BE3B, [869](#)
 CMD_SPI_BE4B, [869](#)
 CMD_SPI_CE, [869](#)
 CMD_SPI_DP, [870](#)
 CMD_SPI_ENSO, [870](#)
 CMD_SPI_ESFBR, [871](#)
 CMD_SPI_ESSPB, [872](#)
 CMD_SPI_EXSO, [870](#)
 CMD_SPI_FAST_READ3B, [868](#)
 CMD_SPI_FAST_READ4B, [869](#)
 CMD_SPI_GBLK, [873](#)
 CMD_SPI_GBULK, [873](#)
 CMD_SPI_NOP, [870](#)
 CMD_SPI_PASSULK, [873](#)
 CMD_SPI_PE_RESUME, [870](#)
 CMD_SPI_PE_SUSPEND, [870](#)
 CMD_SPI_PP3B, [869](#)
 CMD_SPI_PP4B, [869](#)
 CMD_SPI_RDCCR, [871](#)
 CMD_SPI_RDCCR2, [871](#)
 CMD_SPI_RDDPB, [872](#)
 CMD_SPI_RDFBR, [871](#)
 CMD_SPI_RDID, [871](#)
 CMD_SPI_RDLR, [872](#)
 CMD_SPI_RDPASS, [873](#)
 CMD_SPI_RDSCUR, [872](#)
 CMD_SPI_RDSFDP, [871](#)
 CMD_SPI_RDSPB, [872](#)
 CMD_SPI_RDSR, [871](#)
 CMD_SPI_READ3B, [868](#)
 CMD_SPI_READ4B, [869](#)
 CMD_SPI_RST, [870](#)
 CMD_SPI_RSTEN, [870](#)
 CMD_SPI_SBL, [870](#)
 CMD_SPI_SE3B, [869](#)
 CMD_SPI_SE4B, [869](#)
 CMD_SPI_WPSEL, [872](#)
 CMD_SPI_WRCR2, [871](#)
 CMD_SPI_WRDI, [870](#)
 CMD_SPI_WRPDPB, [872](#)
 CMD_SPI_WREN, [869](#)
 CMD_SPI_WRFBR, [871](#)
 CMD_SPI_WRLR, [872](#)
 CMD_SPI_WRPASS, [873](#)
 CMD_SPI_WRSCUR, [872](#)
 CMD_SPI_WRSPB, [872](#)
 CMD_SPI_WRSR, [871](#)
 FLAGS_ALL_ERRORS, [878](#)
 FLAGS_E_FAIL, [877](#)
 FLAGS_ESB, [878](#)
 FLAGS_LDSD, [878](#)
 FLAGS_P_FAIL, [878](#)
 FLAGS_PSB, [878](#)
 FLAGS_SECURED_OTP, [878](#)
 FLAGS_WPSEL, [877](#)
 hal_xsnor_macronix_mx25_c, [881](#)
 mx25_find_id, [889](#)
 mx25_manufacturer_ids, [893](#)
 mx25_memory_type_ids, [893](#)
 MX25_OPT_DUMMY_CYCLES, [867](#)
 MX25_OPT_DUMMY_CYCLES_MASK, [867](#)
 MX25_OPT_NICE_WAITING, [868](#)
 MX25_OPT_NO_WIDTH_SWITCH, [867](#)
 MX25_OPT_USE_DTR, [868](#)
 MX25_OPT_USE_SUBSECTORS, [868](#)
 mx25_poll_status, [889](#)
 mx25_read_id, [890](#)
 mx25_reset_memory, [890](#)
 mx25_write_cr2, [891](#)
 mx25ObjectInit, [889](#)
 PAGE_MASK, [868](#)
 PAGE_SIZE, [868](#)
 macSetCallbackX
 MAC Driver, [288](#)
 macStart
 MAC Driver, [296](#)
 macstate_t
 MAC Driver, [295](#)
 macStop
 MAC Driver, [297](#)
 MACTransmitDescriptor
 MAC Driver, [295](#)
 macWaitReceiveDescriptor
 MAC Driver, [299](#)
 macWaitTransmitDescriptor
 MAC Driver, [298](#)
 macWriteTransmitDescriptor
 MAC Driver, [291](#)
 magic1
 mfs_bank_header_t, [1240](#)
 mfs_data_header_t, [1243](#)
 magic2
 mfs_bank_header_t, [1240](#)
 mfs_data_header_t, [1243](#)
 main.dox, [1341](#)
 Managed Flash Storage Driver, [737](#)
 ALIGNED_DHDR_SIZE, [741](#)
 ALIGNED_REC_SIZE, [741](#)
 ALIGNED_SIZEOF, [741](#)
 crc16, [746](#)
 crc16_table, [764](#)
 MFS_ALIGN_MASK, [744](#)
 MFS_ALIGN_NEXT, [744](#)
 MFS_ALIGN_PREV, [744](#)
 MFS_BANK_0, [745](#)
 MFS_BANK_1, [745](#)
 mfs_bank_check_header, [750](#)
 mfs_bank_erase, [749](#)

- MFS_BANK_ERASED, [746](#)
- MFS_BANK_GARBAGE, [746](#)
- mfs_bank_get_state, [752](#)
- MFS_BANK_MAGIC_1, [742](#)
- MFS_BANK_MAGIC_2, [742](#)
- MFS_BANK_OK, [746](#)
- mfs_bank_scan_records, [751](#)
- mfs_bank_state_t, [746](#)
- mfs_bank_t, [745](#)
- mfs_bank_verify_erase, [749](#)
- mfs_bank_write_header, [750](#)
- MFS_CFG_BUFFER_SIZE, [743](#)
- MFS_CFG_MAX_RECORDS, [742](#)
- MFS_CFG_MAX_REPAIR_ATTEMPTS, [742](#)
- MFS_CFG_MEMORY_ALIGNMENT, [743](#)
- MFS_CFG_STRONG_CHECKING, [743](#)
- MFS_CFG_TRANSACTION_MAX, [743](#)
- MFS_CFG_WRITE_VERIFY, [743](#)
- MFS_ERR_FLASH_FAILURE, [746](#)
- MFS_ERR_INTERNAL, [746](#)
- MFS_ERR_INV_SIZE, [746](#)
- MFS_ERR_INV_STATE, [746](#)
- MFS_ERR_NOT_ERASED, [746](#)
- MFS_ERR_NOT_FOUND, [746](#)
- MFS_ERR_OUT_OF_MEM, [746](#)
- MFS_ERR_TRANSACTION_NUM, [746](#)
- MFS_ERR_TRANSACTION_SIZE, [746](#)
- MFS_ERROR, [745](#)
- mfs_error_t, [745](#)
- mfs_flash_copy, [748](#)
- mfs_flash_get_bank_offset, [746](#)
- mfs_flash_read, [747](#)
- mfs_flash_write, [747](#)
- mfs_garbage_collect, [753](#)
- MFS_HEADER_MAGIC_1, [742](#)
- MFS_HEADER_MAGIC_2, [742](#)
- mfs_id_t, [745](#)
- MFS_IS_ALIGNED, [744](#)
- MFS_IS_ERROR, [743](#)
- MFS_IS_WARNING, [744](#)
- mfs_mount, [754](#)
- MFS_NO_ERROR, [746](#)
- mfs_nocache_buffer_t, [745](#)
- MFS_READY, [745](#)
- mfs_state_reset, [746](#)
- mfs_state_t, [745](#)
- MFS_STOP, [745](#)
- MFS_TRANSACTION, [745](#)
- mfs_try_mount, [753](#)
- MFS_UNINIT, [745](#)
- MFS_WARN_GC, [746](#)
- MFS_WARN_REPAIR, [746](#)
- mfsCommitTransaction, [762](#)
- mfsErase, [757](#)
- mfsEraseRecord, [759](#)
- mfsObjectInit, [755](#)
- mfsPerformGarbageCollection, [760](#)
- mfsReadRecord, [757](#)
- mfsRollbackTransaction, [763](#)
- mfsStart, [756](#)
- mfsStartTransaction, [761](#)
- mfsStop, [756](#)
- mfsWriteRecord, [758](#)
- PAIR, [741](#)
- RET_ON_ERROR, [742](#)
- mask
 - IOBus, [1236](#)
- MAX_FILLER
 - Output Formatter Utility, [967](#)
- mcuconf.h, [1531](#)
 - PLATFORM_MCUCONF, [1531](#)
- mdt_m
 - unpacked_mmc_cid_t, [1301](#)
 - unpacked_sdc_cid_t, [1307](#)
- mdt_y
 - unpacked_mmc_cid_t, [1301](#)
 - unpacked_sdc_cid_t, [1308](#)
- Memory Streams Class, [953](#)
 - _get, [954](#)
 - _memory_stream_data, [953](#)
 - _put, [954](#)
 - _reads, [954](#)
 - _unget, [954](#)
 - _writes, [954](#)
 - msObjectInit, [954](#)
 - vmt, [955](#)
- MemoryStream, [1237](#)
 - vmt, [1239](#)
- memstreams.c, [1446](#)
- memstreams.h, [1447](#)
- MemStreamVMT, [1239](#)
- MFS_ALIGN_MASK
 - Managed Flash Storage Driver, [744](#)
- MFS_ALIGN_NEXT
 - Managed Flash Storage Driver, [744](#)
- MFS_ALIGN_PREV
 - Managed Flash Storage Driver, [744](#)
- MFS_BANK_0
 - Managed Flash Storage Driver, [745](#)
- MFS_BANK_1
 - Managed Flash Storage Driver, [745](#)
- mfs_bank_check_header
 - Managed Flash Storage Driver, [750](#)
- mfs_bank_erase
 - Managed Flash Storage Driver, [749](#)
- MFS_BANK_ERASED
 - Managed Flash Storage Driver, [746](#)
- MFS_BANK_GARBAGE
 - Managed Flash Storage Driver, [746](#)
- mfs_bank_get_state
 - Managed Flash Storage Driver, [752](#)
- mfs_bank_header_t, [1239](#)
 - counter, [1241](#)
 - crc, [1241](#)
 - fields, [1241](#)
 - hdr32, [1241](#)

- hdr8, [1241](#)
- magic1, [1240](#)
- magic2, [1240](#)
- reserved1, [1241](#)
- MFS_BANK_MAGIC_1
 - Managed Flash Storage Driver, [742](#)
- MFS_BANK_MAGIC_2
 - Managed Flash Storage Driver, [742](#)
- MFS_BANK_OK
 - Managed Flash Storage Driver, [746](#)
- mfs_bank_scan_records
 - Managed Flash Storage Driver, [751](#)
- mfs_bank_state_t
 - Managed Flash Storage Driver, [746](#)
- mfs_bank_t
 - Managed Flash Storage Driver, [745](#)
- mfs_bank_verify_erase
 - Managed Flash Storage Driver, [749](#)
- mfs_bank_write_header
 - Managed Flash Storage Driver, [750](#)
- MFS_CFG_BUFFER_SIZE
 - Managed Flash Storage Driver, [743](#)
- MFS_CFG_MAX_RECORDS
 - Managed Flash Storage Driver, [742](#)
- MFS_CFG_MAX_REPAIR_ATTEMPTS
 - Managed Flash Storage Driver, [742](#)
- MFS_CFG_MEMORY_ALIGNMENT
 - Managed Flash Storage Driver, [743](#)
- MFS_CFG_STRONG_CHECKING
 - Managed Flash Storage Driver, [743](#)
- MFS_CFG_TRANSACTION_MAX
 - Managed Flash Storage Driver, [743](#)
- MFS_CFG_WRITE_VERIFY
 - Managed Flash Storage Driver, [743](#)
- mfs_data_header_t, [1242](#)
 - crc, [1243](#)
 - fields, [1243](#)
 - hdr32, [1243](#)
 - hdr8, [1243](#)
 - id, [1243](#)
 - magic1, [1243](#)
 - magic2, [1243](#)
 - size, [1243](#)
- MFS_ERR_FLASH_FAILURE
 - Managed Flash Storage Driver, [746](#)
- MFS_ERR_INTERNAL
 - Managed Flash Storage Driver, [746](#)
- MFS_ERR_INV_SIZE
 - Managed Flash Storage Driver, [746](#)
- MFS_ERR_INV_STATE
 - Managed Flash Storage Driver, [746](#)
- MFS_ERR_NOT_ERASED
 - Managed Flash Storage Driver, [746](#)
- MFS_ERR_NOT_FOUND
 - Managed Flash Storage Driver, [746](#)
- MFS_ERR_OUT_OF_MEM
 - Managed Flash Storage Driver, [746](#)
- MFS_ERR_TRANSACTION_NUM
 - Managed Flash Storage Driver, [746](#)
- MFS_ERR_TRANSACTION_SIZE
 - Managed Flash Storage Driver, [746](#)
- MFS_ERROR
 - Managed Flash Storage Driver, [745](#)
- mfs_error_t
 - Managed Flash Storage Driver, [745](#)
- mfs_flash_copy
 - Managed Flash Storage Driver, [748](#)
- mfs_flash_get_bank_offset
 - Managed Flash Storage Driver, [746](#)
- mfs_flash_read
 - Managed Flash Storage Driver, [747](#)
- mfs_flash_write
 - Managed Flash Storage Driver, [747](#)
- mfs_garbage_collect
 - Managed Flash Storage Driver, [753](#)
- MFS_HEADER_MAGIC_1
 - Managed Flash Storage Driver, [742](#)
- MFS_HEADER_MAGIC_2
 - Managed Flash Storage Driver, [742](#)
- mfs_id_t
 - Managed Flash Storage Driver, [745](#)
- MFS_IS_ALIGNED
 - Managed Flash Storage Driver, [744](#)
- MFS_IS_ERROR
 - Managed Flash Storage Driver, [743](#)
- MFS_IS_WARNING
 - Managed Flash Storage Driver, [744](#)
- mfs_mount
 - Managed Flash Storage Driver, [754](#)
- MFS_NO_ERROR
 - Managed Flash Storage Driver, [746](#)
- mfs_nocache_buffer, [1244](#)
 - bhdr, [1245](#)
 - data16, [1245](#)
 - data32, [1245](#)
 - data8, [1245](#)
 - dhdr, [1245](#)
- mfs_nocache_buffer_t
 - Managed Flash Storage Driver, [745](#)
- MFS_READY
 - Managed Flash Storage Driver, [745](#)
- mfs_record_descriptor_t, [1245](#)
 - offset, [1246](#)
 - size, [1246](#)
- mfs_state_reset
 - Managed Flash Storage Driver, [746](#)
- mfs_state_t
 - Managed Flash Storage Driver, [745](#)
- MFS_STOP
 - Managed Flash Storage Driver, [745](#)
- MFS_TRANSACTION
 - Managed Flash Storage Driver, [745](#)
- mfs_transaction_op_t, [1246](#)
 - id, [1247](#)
 - offset, [1247](#)
 - size, [1247](#)

- mfs_try_mount
 - Managed Flash Storage Driver, [753](#)
- MFS_UNINIT
 - Managed Flash Storage Driver, [745](#)
- MFS_WARN_GC
 - Managed Flash Storage Driver, [746](#)
- MFS_WARN_REPAIR
 - Managed Flash Storage Driver, [746](#)
- mfsCommitTransaction
 - Managed Flash Storage Driver, [762](#)
- MFSConfig, [1247](#)
 - bank0_sectors, [1249](#)
 - bank0_start, [1249](#)
 - bank1_sectors, [1249](#)
 - bank1_start, [1249](#)
 - bank_size, [1249](#)
 - erased, [1249](#)
 - flashp, [1249](#)
- MFSDriver, [1250](#)
 - config, [1251](#)
 - current_bank, [1251](#)
 - current_counter, [1252](#)
 - descriptors, [1252](#)
 - ncbuf, [1253](#)
 - next_offset, [1252](#)
 - state, [1251](#)
 - tr_limit_offset, [1252](#)
 - tr_next_offset, [1252](#)
 - tr_nops, [1252](#)
 - tr_ops, [1253](#)
 - used_space, [1252](#)
- mfsErase
 - Managed Flash Storage Driver, [757](#)
- mfsEraseRecord
 - Managed Flash Storage Driver, [759](#)
- mfsObjectInit
 - Managed Flash Storage Driver, [755](#)
- mfsPerformGarbageCollection
 - Managed Flash Storage Driver, [760](#)
- mfsReadRecord
 - Managed Flash Storage Driver, [757](#)
- mfsRollbackTransaction
 - Managed Flash Storage Driver, [763](#)
- mfsStart
 - Managed Flash Storage Driver, [756](#)
- mfsStartTransaction
 - Managed Flash Storage Driver, [761](#)
- mfsStop
 - Managed Flash Storage Driver, [756](#)
- mfsWriteRecord
 - Managed Flash Storage Driver, [758](#)
- Micron N25Q Driver, [894](#)
 - __hal_xsnor_micron_n25q_vmt, [916](#)
 - __n25q_dispose_impl, [903](#)
 - __n25q_init_impl, [904](#)
 - __n25q_mmap_off_impl, [909](#)
 - __n25q_mmap_on_impl, [909](#)
 - __n25q_objinit_impl, [902](#)
 - __n25q_program_impl, [905](#)
 - __n25q_query_erase_impl, [907](#)
 - __n25q_read_impl, [904](#)
 - __n25q_start_erase_all_impl, [906](#)
 - __n25q_start_erase_sector_impl, [907](#)
 - __n25q_verify_erase_impl, [908](#)
 - cmd1l, [913](#)
 - cmd2l, [913](#)
 - cmd4l, [914](#)
 - CMD_BULK_ERASE, [901](#)
 - CMD_CLEAR_FLAG_STATUS_REGISTER, [900](#)
 - CMD_FAST_READ, [899](#)
 - CMD_MULTIPLE_IO_READ_ID, [899](#)
 - CMD_PAGE_PROGRAM, [900](#)
 - CMD_PROGRAM_ERASE_RESUME, [901](#)
 - CMD_PROGRAM_ERASE_SUSPEND, [901](#)
 - CMD_PROGRAM_OTP_ARRAY, [901](#)
 - CMD_READ, [899](#)
 - CMD_READ_DISCOVERY_PARAMETER, [899](#)
 - CMD_READ_ENHANCED_V_CONF_REGISTER, [900](#)
 - CMD_READ_FLAG_STATUS_REGISTER, [900](#)
 - CMD_READ_ID, [899](#)
 - CMD_READ_LOCK_REGISTER, [899](#)
 - CMD_READ_NV_CONFIGURATION_REGISTER, [900](#)
 - CMD_READ_OTP_ARRAY, [901](#)
 - CMD_READ_STATUS_REGISTER, [899](#)
 - CMD_READ_V_CONF_REGISTER, [900](#)
 - CMD_RESET_ENABLE, [898](#)
 - cmd_reset_enable_1, [914](#)
 - cmd_reset_enable_2, [914](#)
 - cmd_reset_enable_4, [915](#)
 - CMD_RESET_MEMORY, [898](#)
 - cmd_reset_memory_1, [914](#)
 - cmd_reset_memory_2, [915](#)
 - cmd_reset_memory_4, [915](#)
 - CMD_SECTOR_ERASE, [901](#)
 - CMD_SUBSECTOR_ERASE, [901](#)
 - CMD_WRITE_DISABLE, [899](#)
 - CMD_WRITE_ENABLE, [899](#)
 - CMD_WRITE_ENHANCED_V_CONF_REGISTER, [900](#)
 - CMD_WRITE_LOCK_REGISTER, [900](#)
 - CMD_WRITE_NV_CONFIGURATION_REGISTER, [900](#)
 - CMD_WRITE_STATUS_REGISTER, [899](#)
 - CMD_WRITE_V_CONF_REGISTER, [900](#)
 - FLAGS_ALL_ERRORS, [902](#)
 - FLAGS_ERASE_ERROR, [901](#)
 - FLAGS_ERASE_SUSPEND, [901](#)
 - FLAGS_PROGRAM_ERASE, [901](#)
 - FLAGS_PROGRAM_ERROR, [902](#)
 - FLAGS_PROGRAM_SUSPEND, [902](#)
 - FLAGS_PROTECTION_ERROR, [902](#)
 - FLAGS_RESERVED, [902](#)
 - FLAGS_VPP_ERROR, [902](#)
 - hal_xsnor_micron_n25q_c, [902](#)

- n25q_activate_xip, 911
- n25q_find_id, 910
- n25q_manufacturer_ids, 915
- n25q_memory_type_ids, 915
- N25Q_OPT_DUMMY_CYCLES, 897
- N25Q_OPT_DUMMY_CYCLES_MASK, 897
- N25Q_OPT_NICE_WAITING, 898
- N25Q_OPT_NO_WIDTH_SWITCH, 898
- N25Q_OPT_USE_SUBSECTORS, 898
- n25q_poll_status, 911
- n25q_read_id, 912
- n25q_reset_memory, 912
- n25q_reset_xip, 912
- n25qObjectInit, 910
- PAGE_MASK, 898
- PAGE_SIZE, 898
- mid
 - unpacked_mmc_cid_t, 1300
 - unpacked_sdc_cid_t, 1307
- MII/RMII Header, 1053
 - ADVERTISE_100BASE4, 1063
 - ADVERTISE_100FULL, 1063
 - ADVERTISE_100HALF, 1062
 - ADVERTISE_10FULL, 1062
 - ADVERTISE_10HALF, 1062
 - ADVERTISE_ALL, 1064
 - ADVERTISE_CSMA, 1062
 - ADVERTISE_FULL, 1064
 - ADVERTISE_LPACK, 1063
 - ADVERTISE_NPAGE, 1063
 - ADVERTISE_PAUSE_ASYM, 1063
 - ADVERTISE_PAUSE_CAP, 1063
 - ADVERTISE_RESV, 1063
 - ADVERTISE_RFAULT, 1063
 - ADVERTISE_SLCT, 1062
 - BMCR_ANENABLE, 1060
 - BMCR_ANRESTART, 1059
 - BMCR_CTST, 1059
 - BMCR_FULLDPLX, 1059
 - BMCR_ISOLATE, 1060
 - BMCR_LOOPBACK, 1060
 - BMCR_PDOWN, 1060
 - BMCR_RESET, 1060
 - BMCR_RESV, 1059
 - BMCR_SPEED100, 1060
 - BMSR_100BASE4, 1062
 - BMSR_100FULL, 1062
 - BMSR_100HALF, 1062
 - BMSR_10FULL, 1061
 - BMSR_10HALF, 1061
 - BMSR_ANEGCAPABLE, 1061
 - BMSR_ANEGCOMPLETE, 1061
 - BMSR_ERCAP, 1060
 - BMSR_JCD, 1060
 - BMSR_LSTATUS, 1061
 - BMSR_MFPRESUPPCAP, 1061
 - BMSR_RESV, 1061
 - BMSR_RFAULT, 1061
 - EXPANSION_ENABLENPAGE, 1066
 - EXPANSION_LCWP, 1066
 - EXPANSION_MFAULTS, 1066
 - EXPANSION_NPCAPABLE, 1066
 - EXPANSION_NWAY, 1066
 - EXPANSION_RESV, 1066
 - LPA_100, 1066
 - LPA_100BASE4, 1065
 - LPA_100FULL, 1064
 - LPA_100HALF, 1064
 - LPA_10FULL, 1064
 - LPA_10HALF, 1064
 - LPA_DUPLEX, 1065
 - LPA_LPACK, 1065
 - LPA_NPAGE, 1065
 - LPA_PAUSE_ASYM, 1065
 - LPA_PAUSE_CAP, 1065
 - LPA_RESV, 1065
 - LPA_RFAULT, 1065
 - LPA_SLCT, 1064
 - MII_ADVERTISE, 1057
 - MII_AM79C875_ID, 1067
 - MII_ANNPTR, 1057
 - MII_BMCR, 1056
 - MII_BMSR, 1056
 - MII_CTRL1000, 1057
 - MII_DCOUNTER, 1058
 - MII_DM9161_ID, 1067
 - MII_DP83848I_ID, 1067
 - MII_ESTATUS, 1057
 - MII_EXPANSION, 1057
 - MII_FCSCOUNTER, 1058
 - MII_KS8721_ID, 1067
 - MII_KSZ8081_ID, 1067
 - MII_LAN8710A_ID, 1067
 - MII_LAN8720_ID, 1068
 - MII_LAN8740A_ID, 1068
 - MII_LAN8742A_ID, 1068
 - MII_LBRERROR, 1058
 - MII_LPA, 1057
 - MII_MICR, 1058
 - MII_NCONFIG, 1059
 - MII_NWAYTEST, 1058
 - MII_PHYADDR, 1059
 - MII_PHYSID1, 1056
 - MII_PHYSID2, 1056
 - MII_PHYSTS, 1057
 - MII_RERRCOUNTER, 1058
 - MII_RESV1, 1058
 - MII_RESV2, 1059
 - MII_SREVISION, 1058
 - MII_STAT1000, 1057
 - MII_STE101P_ID, 1067
 - MII_TPISTATUS, 1059
 - NWAYTEST_LOOPBACK, 1067
 - NWAYTEST_RESV1, 1066
 - NWAYTEST_RESV2, 1067
 - MII_ADVERTISE

- MII/RMII Header, [1057](#)
- MII_AM79C875_ID
 - MII/RMII Header, [1067](#)
- MII_ANNPTR
 - MII/RMII Header, [1057](#)
- MII_BMCR
 - MII/RMII Header, [1056](#)
- MII_BMSR
 - MII/RMII Header, [1056](#)
- MII_CTRL1000
 - MII/RMII Header, [1057](#)
- MII_DCOUNTER
 - MII/RMII Header, [1058](#)
- MII_DM9161_ID
 - MII/RMII Header, [1067](#)
- MII_DP83848I_ID
 - MII/RMII Header, [1067](#)
- MII_ESTATUS
 - MII/RMII Header, [1057](#)
- MII_EXPANSION
 - MII/RMII Header, [1057](#)
- MII_FCSCOUNTER
 - MII/RMII Header, [1058](#)
- MII_KS8721_ID
 - MII/RMII Header, [1067](#)
- MII_KSZ8081_ID
 - MII/RMII Header, [1067](#)
- MII_LAN8710A_ID
 - MII/RMII Header, [1067](#)
- MII_LAN8720_ID
 - MII/RMII Header, [1068](#)
- MII_LAN8740A_ID
 - MII/RMII Header, [1068](#)
- MII_LAN8742A_ID
 - MII/RMII Header, [1068](#)
- MII_LBRERROR
 - MII/RMII Header, [1058](#)
- MII_LPA
 - MII/RMII Header, [1057](#)
- MII_MICR
 - MII/RMII Header, [1058](#)
- MII_NCONFIG
 - MII/RMII Header, [1059](#)
- MII_NWAYTEST
 - MII/RMII Header, [1058](#)
- MII_PHYADDR
 - MII/RMII Header, [1059](#)
- MII_PHYSID1
 - MII/RMII Header, [1056](#)
- MII_PHYSID2
 - MII/RMII Header, [1056](#)
- MII_PHYSTS
 - MII/RMII Header, [1057](#)
- MII_RERRCOUNTER
 - MII/RMII Header, [1058](#)
- MII_RESV1
 - MII/RMII Header, [1058](#)
- MII_RESV2
 - MII/RMII Header, [1059](#)
- MII_SREVISION
 - MII/RMII Header, [1058](#)
- MII_STAT1000
 - MII/RMII Header, [1057](#)
- MII_STE101P_ID
 - MII/RMII Header, [1067](#)
- MII_TPISTATUS
 - MII/RMII Header, [1059](#)
- millisecond
 - RTCDatetime, [1274](#)
- mmap_off
 - hal_xsnor_base_vmt, [1212](#)
 - hal_xsnor_macronix_mx25_vmt, [1218](#)
 - hal_xsnor_micron_n25q_vmt, [1224](#)
- mmap_on
 - hal_xsnor_base_vmt, [1212](#)
 - hal_xsnor_macronix_mx25_vmt, [1218](#)
 - hal_xsnor_micron_n25q_vmt, [1224](#)
- MMC over SPI Driver, [764](#)
 - __mmc_driver_methods, [768](#)
 - MMC_ACMD41_RETRY, [767](#)
 - MMC_BUFFER_SIZE, [767](#)
 - MMC_CMD0_RETRY, [767](#)
 - MMC_CMD1_RETRY, [767](#)
 - mmc_connect, [770](#)
 - mmc_crc7, [773](#)
 - mmc_crc7_lookup_table, [792](#)
 - mmc_disconnect, [770](#)
 - mmc_get_info, [773](#)
 - MMC_IDLE_TIMEOUT_MS, [767](#)
 - mmc_is_card_inserted, [770](#)
 - mmc_is_write_protected, [770](#)
 - mmc_lld_is_card_inserted, [791](#)
 - mmc_lld_is_write_protected, [791](#)
 - mmc_read, [771](#)
 - mmc_read_CxD, [779](#)
 - mmc_recvr1, [775](#)
 - mmc_recvr3, [776](#)
 - mmc_send_command_R1, [777](#)
 - mmc_send_command_R3, [778](#)
 - mmc_send_hdr, [774](#)
 - mmc_sync, [772](#)
 - MMC_USE_MUTUAL_EXCLUSION, [768](#)
 - mmc_vmt, [792](#)
 - MMC_WAIT_DATA, [767](#)
 - mmc_wait_idle, [774](#)
 - mmc_write, [772](#)
 - MMConfig, [769](#)
 - mmcConnect, [782](#)
 - mmcDisconnect, [783](#)
 - MMCDriver, [769](#)
 - mmcErase, [790](#)
 - mmcGetInfo, [790](#)
 - mmclnit, [780](#)
 - mmclsCardInserted, [768](#)
 - mmclsWriteProtected, [768](#)
 - mmcObjectInit, [780](#)

- mmcSequentialRead, [785](#)
- mmcSequentialWrite, [787](#)
- mmcStart, [780](#)
- mmcStartSequentialRead, [784](#)
- mmcStartSequentialWrite, [786](#)
- mmcStop, [782](#)
- mmcStopSequentialRead, [786](#)
- mmcStopSequentialWrite, [788](#)
- mmcSync, [789](#)
- MMC/SD Block Device, [1005](#)
 - _mmcscd_block_device_data, [1024](#)
 - _mmcscd_block_device_methods, [1024](#)
 - _mmcscd_get_capacity, [1026](#)
 - _mmcscd_get_capacity_ext, [1027](#)
 - _mmcscd_get_slice, [1026](#)
 - _mmcscd_unpack_csd_mmc, [1028](#)
 - _mmcscd_unpack_csd_v10, [1029](#)
 - _mmcscd_unpack_csd_v20, [1029](#)
 - _mmcscd_unpack_mmc_cid, [1028](#)
 - _mmcscd_unpack_sdc_cid, [1027](#)
 - MMCSDBLOCK_SIZE, [1009](#)
 - MMCSDCID_MMC_CRC_SLICE, [1023](#)
 - MMCSDCID_MMC_MDT_M_SLICE, [1023](#)
 - MMCSDCID_MMC_MDT_Y_SLICE, [1023](#)
 - MMCSDCID_MMC_MID_SLICE, [1024](#)
 - MMCSDCID_MMC_OID_SLICE, [1024](#)
 - MMCSDCID_MMC_PNM0_SLICE, [1023](#)
 - MMCSDCID_MMC_PNM1_SLICE, [1023](#)
 - MMCSDCID_MMC_PNM2_SLICE, [1024](#)
 - MMCSDCID_MMC_PNM3_SLICE, [1024](#)
 - MMCSDCID_MMC_PNM4_SLICE, [1024](#)
 - MMCSDCID_MMC_PNM5_SLICE, [1024](#)
 - MMCSDCID_MMC_PRV_M_SLICE, [1023](#)
 - MMCSDCID_MMC_PRV_N_SLICE, [1023](#)
 - MMCSDCID_MMC_PSN_SLICE, [1023](#)
 - MMCSDCID_SDC_CRC_SLICE, [1021](#)
 - MMCSDCID_SDC_MDT_M_SLICE, [1022](#)
 - MMCSDCID_SDC_MDT_Y_SLICE, [1022](#)
 - MMCSDCID_SDC_MID_SLICE, [1023](#)
 - MMCSDCID_SDC_OID_SLICE, [1023](#)
 - MMCSDCID_SDC_PNM0_SLICE, [1022](#)
 - MMCSDCID_SDC_PNM1_SLICE, [1022](#)
 - MMCSDCID_SDC_PNM2_SLICE, [1022](#)
 - MMCSDCID_SDC_PNM3_SLICE, [1022](#)
 - MMCSDCID_SDC_PNM4_SLICE, [1022](#)
 - MMCSDCID_SDC_PRV_M_SLICE, [1022](#)
 - MMCSDCID_SDC_PRV_N_SLICE, [1022](#)
 - MMCSDCID_SDC_PSN_SLICE, [1022](#)
 - MMCSDCMD8_PATTERN, [1009](#)
 - MMCSDCMD_ALL_SEND_CID, [1010](#)
 - MMCSDCMD_APP_CMD, [1013](#)
 - MMCSDCMD_APP_OP_COND, [1012](#)
 - MMCSDCMD_ERASE, [1012](#)
 - MMCSDCMD_ERASE_RW_BLK_END, [1012](#)
 - MMCSDCMD_ERASE_RW_BLK_START, [1012](#)
 - MMCSDCMD_GO_IDLE_STATE, [1010](#)
 - MMCSDCMD_INIT, [1010](#)
 - MMCSDCMD_LOCK_UNLOCK, [1013](#)
 - MMCSDCMD_READ_MULTIPLE_BLOCK, [1012](#)
 - MMCSDCMD_READ_OCR, [1013](#)
 - MMCSDCMD_READ_SINGLE_BLOCK, [1012](#)
 - MMCSDCMD_SEL_DESEL_CARD, [1011](#)
 - MMCSDCMD_SEND_CID, [1011](#)
 - MMCSDCMD_SEND_CSD, [1011](#)
 - MMCSDCMD_SEND_EXT_CSD, [1011](#)
 - MMCSDCMD_SEND_IF_COND, [1011](#)
 - MMCSDCMD_SEND_RELATIVE_ADDR, [1010](#)
 - MMCSDCMD_SEND_STATUS, [1011](#)
 - MMCSDCMD_SET_BLOCK_COUNT, [1012](#)
 - MMCSDCMD_SET_BLOCKLEN, [1012](#)
 - MMCSDCMD_SET_BUS_WIDTH, [1011](#)
 - MMCSDCMD_SET_RELATIVE_ADDR, [1011](#)
 - MMCSDCMD_STOP_TRANSMISSION, [1011](#)
 - MMCSDCMD_SWITCH, [1011](#)
 - MMCSDCMD_WRITE_BLOCK, [1012](#)
 - MMCSDCMD_WRITE_MULTIPLE_BLOCK, [1012](#)
 - MMCSDCSD_10_C_SIZE_MULT_SLICE, [1020](#)
 - MMCSDCSD_10_C_SIZE_SLICE, [1020](#)
 - MMCSDCSD_10_CCC_SLICE, [1021](#)
 - MMCSDCSD_10_COPY_SLICE, [1019](#)
 - MMCSDCSD_10_CRC_SLICE, [1019](#)
 - MMCSDCSD_10_CSD_STRUCTURE_SLICE, [1021](#)
 - MMCSDCSD_10_DSR_IMP_SLICE, [1020](#)
 - MMCSDCSD_10_ERASE_BLK_EN_SLICE, [1020](#)
 - MMCSDCSD_10_ERASE_SECTOR_SIZE_SLICE, [1020](#)
 - MMCSDCSD_10_FILE_FORMAT_GRP_SLICE, [1019](#)
 - MMCSDCSD_10_FILE_FORMAT_SLICE, [1019](#)
 - MMCSDCSD_10_NSAC_SLICE, [1021](#)
 - MMCSDCSD_10_PERM_WRITE_PROTECT_SLICE, [1019](#)
 - MMCSDCSD_10_R2W_FACTOR_SLICE, [1019](#)
 - MMCSDCSD_10_READ_BL_LEN_SLICE, [1021](#)
 - MMCSDCSD_10_READ_BL_PARTIAL_SLICE, [1021](#)
 - MMCSDCSD_10_READ_BLK_MISALIGN_SLICE, [1021](#)
 - MMCSDCSD_10_TAAC_SLICE, [1021](#)
 - MMCSDCSD_10_TMP_WRITE_PROTECT_SLICE, [1019](#)
 - MMCSDCSD_10_TRANS_SPEED_SLICE, [1021](#)
 - MMCSDCSD_10_VDD_R_CURR_MAX_SLICE, [1020](#)
 - MMCSDCSD_10_VDD_R_CURR_MIX_SLICE, [1020](#)
 - MMCSDCSD_10_VDD_W_CURR_MAX_SLICE, [1020](#)
 - MMCSDCSD_10_VDD_W_CURR_MIN_SLICE, [1020](#)
 - MMCSDCSD_10_WP_GRP_ENABLE_SLICE, [1019](#)
 - MMCSDCSD_10_WP_GRP_SIZE_SLICE, [1020](#)
 - MMCSDCSD_10_WRITE_BL_LEN_SLICE, [1019](#)

- MMCS_D_CSD_10_WRITE_BLK_PARTIAL_SLICE, [1019](#)
- MMCS_D_CSD_10_WRITE_BLK_MISALIGN_SLICE, [1021](#)
- MMCS_D_CSD_20_C_SIZE_SLICE, [1017](#)
- MMCS_D_CSD_20_CCC_SLICE, [1018](#)
- MMCS_D_CSD_20_COPY_SLICE, [1017](#)
- MMCS_D_CSD_20_CRC_SLICE, [1016](#)
- MMCS_D_CSD_20_CSD_STRUCTURE_SLICE, [1018](#)
- MMCS_D_CSD_20_DSR_IMP_SLICE, [1018](#)
- MMCS_D_CSD_20_ERASE_BLK_EN_SLICE, [1017](#)
- MMCS_D_CSD_20_ERASE_SECTOR_SIZE_SLICE, [1017](#)
- MMCS_D_CSD_20_FILE_FORMAT_GRP_SLICE, [1017](#)
- MMCS_D_CSD_20_FILE_FORMAT_SLICE, [1016](#)
- MMCS_D_CSD_20_NSAC_SLICE, [1018](#)
- MMCS_D_CSD_20_PERM_WRITE_PROTECT_SLICE, [1016](#)
- MMCS_D_CSD_20_R2W_FACTOR_SLICE, [1017](#)
- MMCS_D_CSD_20_READ_BLK_LEN_SLICE, [1018](#)
- MMCS_D_CSD_20_READ_BLK_PARTIAL_SLICE, [1018](#)
- MMCS_D_CSD_20_READ_BLK_MISALIGN_SLICE, [1018](#)
- MMCS_D_CSD_20_TAAC_SLICE, [1018](#)
- MMCS_D_CSD_20_TMP_WRITE_PROTECT_SLICE, [1016](#)
- MMCS_D_CSD_20_TRANS_SPEED_SLICE, [1018](#)
- MMCS_D_CSD_20_WP_GRP_ENABLE_SLICE, [1017](#)
- MMCS_D_CSD_20_WP_GRP_SIZE_SLICE, [1017](#)
- MMCS_D_CSD_20_WRITE_BLK_LEN_SLICE, [1017](#)
- MMCS_D_CSD_20_WRITE_BLK_PARTIAL_SLICE, [1017](#)
- MMCS_D_CSD_20_WRITE_BLK_MISALIGN_SLICE, [1018](#)
- MMCS_D_CSD_MMC_C_SIZE_MULT_SLICE, [1014](#)
- MMCS_D_CSD_MMC_C_SIZE_SLICE, [1014](#)
- MMCS_D_CSD_MMC_CCC_SLICE, [1013](#)
- MMCS_D_CSD_MMC_CONTENT_PROT_APP_SLICE, [1015](#)
- MMCS_D_CSD_MMC_COPY_SLICE, [1016](#)
- MMCS_D_CSD_MMC_CRC_SLICE, [1016](#)
- MMCS_D_CSD_MMC_CSD_STRUCTURE_SLICE, [1013](#)
- MMCS_D_CSD_MMC_DEFAULT_ECC_SLICE, [1015](#)
- MMCS_D_CSD_MMC_DSR_IMP_SLICE, [1014](#)
- MMCS_D_CSD_MMC_ECC_SLICE, [1016](#)
- MMCS_D_CSD_MMC_ERASE_GRP_MULT_SLICE, [1015](#)
- MMCS_D_CSD_MMC_ERASE_GRP_SIZE_SLICE, [1015](#)
- MMCS_D_CSD_MMC_FILE_FORMAT_GRP_SLICE, [1015](#)
- MMCS_D_CSD_MMC_FILE_FORMAT_SLICE, [1016](#)
- MMCS_D_CSD_MMC_NSAC_SLICE, [1013](#)
- MMCS_D_CSD_MMC_PERM_WRITE_PROTECT_SLICE, [1016](#)
- MMCS_D_CSD_MMC_R2W_FACTOR_SLICE, [1015](#)
- MMCS_D_CSD_MMC_READ_BLK_LEN_SLICE, [1013](#)
- MMCS_D_CSD_MMC_READ_BLK_PARTIAL_SLICE, [1014](#)
- MMCS_D_CSD_MMC_READ_BLK_MISALIGN_SLICE, [1014](#)
- MMCS_D_CSD_MMC_SPEC_VERS_SLICE, [1013](#)
- MMCS_D_CSD_MMC_TAAC_SLICE, [1013](#)
- MMCS_D_CSD_MMC_TMP_WRITE_PROTECT_SLICE, [1016](#)
- MMCS_D_CSD_MMC_TRAN_SPEED_SLICE, [1013](#)
- MMCS_D_CSD_MMC_VDD_R_CURR_MAX_SLICE, [1014](#)
- MMCS_D_CSD_MMC_VDD_R_CURR_MIN_SLICE, [1014](#)
- MMCS_D_CSD_MMC_VDD_W_CURR_MAX_SLICE, [1014](#)
- MMCS_D_CSD_MMC_VDD_W_CURR_MIN_SLICE, [1014](#)
- MMCS_D_CSD_MMC_WP_GRP_ENABLE_SLICE, [1015](#)
- MMCS_D_CSD_MMC_WP_GRP_SIZE_SLICE, [1015](#)
- MMCS_D_CSD_MMC_WRITE_BLK_LEN_SLICE, [1015](#)
- MMCS_D_CSD_MMC_WRITE_BLK_PARTIAL_SLICE, [1015](#)
- MMCS_D_CSD_MMC_WRITE_BLK_MISALIGN_SLICE, [1014](#)
- MMCS_D_R1_ERROR, [1024](#)
- MMCS_D_R1_ERROR_MASK, [1009](#)
- MMCS_D_R1_IS_CARD_LOCKED, [1025](#)
- MMCS_D_R1_STS, [1025](#)
- MMCS_D_STS_DATA, [1010](#)
- MMCS_D_STS_DIS, [1010](#)
- MMCS_D_STS_IDENT, [1009](#)
- MMCS_D_STS_IDLE, [1009](#)
- MMCS_D_STS_PRG, [1010](#)
- MMCS_D_STS_RCV, [1010](#)
- MMCS_D_STS_READY, [1009](#)
- MMCS_D_STS_STBY, [1010](#)
- MMCS_D_STS_TRAN, [1010](#)
- mmcGetCardCapacity, [1025](#)
- MMC_ACMD41_RETRY
- MMC over SPI Driver, [767](#)
- MMC_BUFFER_SIZE
- MMC over SPI Driver, [767](#)
- MMC_CMD0_RETRY
- MMC over SPI Driver, [767](#)

MMC_CMD1_RETRY
 MMC over SPI Driver, [767](#)
 mmc_cmd6_construct
 SDC Driver, [411](#)
 mmc_connect
 MMC over SPI Driver, [770](#)
 mmc_crc7
 MMC over SPI Driver, [773](#)
 mmc_crc7_lookup_table
 MMC over SPI Driver, [792](#)
 mmc_detect_bus_clk
 SDC Driver, [414](#)
 mmc_disconnect
 MMC over SPI Driver, [770](#)
 mmc_get_info
 MMC over SPI Driver, [773](#)
 MMC_IDLE_TIMEOUT_MS
 Configuration, [51](#)
 MMC over SPI Driver, [767](#)
 mmc_init
 SDC Driver, [409](#)
 mmc_is_card_inserted
 MMC over SPI Driver, [770](#)
 mmc_is_write_protected
 MMC over SPI Driver, [770](#)
 mmc_ild_is_card_inserted
 MMC over SPI Driver, [791](#)
 mmc_ild_is_write_protected
 MMC over SPI Driver, [791](#)
 mmc_read
 MMC over SPI Driver, [771](#)
 mmc_read_CxD
 MMC over SPI Driver, [779](#)
 mmc_recvr1
 MMC over SPI Driver, [775](#)
 mmc_recvr3
 MMC over SPI Driver, [776](#)
 mmc_send_command_R1
 MMC over SPI Driver, [777](#)
 mmc_send_command_R3
 MMC over SPI Driver, [778](#)
 mmc_send_hdr
 MMC over SPI Driver, [774](#)
 mmc_set_bus_width
 SDC Driver, [416](#)
 mmc_spi_config_t, [1253](#)
 hscfg, [1255](#)
 lscfg, [1255](#)
 spip, [1255](#)
 mmc_spi_driver_t, [1255](#)
 block_addresses, [1257](#)
 buffer, [1257](#)
 config, [1257](#)
 vmt, [1257](#)
 mmc_spi_driver_vmt, [1258](#)
 MMC_SWITCH_CLEAR_BITS
 SDC Driver, [408](#)
 MMC_SWITCH_COMMAND_SET
 SDC Driver, [408](#)
 MMC_SWITCH_SET_BITS
 SDC Driver, [408](#)
 mmc_switch_t
 SDC Driver, [407](#)
 MMC_SWITCH_WRITE_BYTE
 SDC Driver, [408](#)
 mmc_sync
 MMC over SPI Driver, [772](#)
 MMC_USE_MUTUAL_EXCLUSION
 Configuration, [51](#)
 MMC over SPI Driver, [768](#)
 mmc_vmt
 MMC over SPI Driver, [792](#)
 MMC_WAIT_DATA
 MMC over SPI Driver, [767](#)
 mmc_wait_idle
 MMC over SPI Driver, [774](#)
 mmc_write
 MMC over SPI Driver, [772](#)
 MMCConfig
 MMC over SPI Driver, [769](#)
 mmcConnect
 MMC over SPI Driver, [782](#)
 mmcDisconnect
 MMC over SPI Driver, [783](#)
 MMCDriver
 MMC over SPI Driver, [769](#)
 mmcErase
 MMC over SPI Driver, [790](#)
 mmcGetInfo
 MMC over SPI Driver, [790](#)
 mmclnit
 MMC over SPI Driver, [780](#)
 mmclsCardInserted
 MMC over SPI Driver, [768](#)
 mmclsWriteProtected
 MMC over SPI Driver, [768](#)
 mmcObjectInit
 MMC over SPI Driver, [780](#)
 MMCSD_BLOCK_SIZE
 MMC/SD Block Device, [1009](#)
 MMCSD_CID_MMC_CRC_SLICE
 MMC/SD Block Device, [1023](#)
 MMCSD_CID_MMC_MDT_M_SLICE
 MMC/SD Block Device, [1023](#)
 MMCSD_CID_MMC_MDT_Y_SLICE
 MMC/SD Block Device, [1023](#)
 MMCSD_CID_MMC_MID_SLICE
 MMC/SD Block Device, [1024](#)
 MMCSD_CID_MMC_OID_SLICE
 MMC/SD Block Device, [1024](#)
 MMCSD_CID_MMC_PNM0_SLICE
 MMC/SD Block Device, [1023](#)
 MMCSD_CID_MMC_PNM1_SLICE
 MMC/SD Block Device, [1023](#)
 MMCSD_CID_MMC_PNM2_SLICE
 MMC/SD Block Device, [1024](#)

- MMCSD_CID_MMC_PNM3_SLICE
 - MMC/SD Block Device, [1024](#)
- MMCSD_CID_MMC_PNM4_SLICE
 - MMC/SD Block Device, [1024](#)
- MMCSD_CID_MMC_PNM5_SLICE
 - MMC/SD Block Device, [1024](#)
- MMCSD_CID_MMC_PRV_M_SLICE
 - MMC/SD Block Device, [1023](#)
- MMCSD_CID_MMC_PRV_N_SLICE
 - MMC/SD Block Device, [1023](#)
- MMCSD_CID_MMC_PSN_SLICE
 - MMC/SD Block Device, [1023](#)
- MMCSD_CID_SDC_CRC_SLICE
 - MMC/SD Block Device, [1021](#)
- MMCSD_CID_SDC_MDT_M_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CID_SDC_MDT_Y_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CID_SDC_MID_SLICE
 - MMC/SD Block Device, [1023](#)
- MMCSD_CID_SDC_OID_SLICE
 - MMC/SD Block Device, [1023](#)
- MMCSD_CID_SDC_PNM0_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CID_SDC_PNM1_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CID_SDC_PNM2_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CID_SDC_PNM3_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CID_SDC_PNM4_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CID_SDC_PRV_M_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CID_SDC_PRV_N_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CID_SDC_PSN_SLICE
 - MMC/SD Block Device, [1022](#)
- MMCSD_CMD8_PATTERN
 - MMC/SD Block Device, [1009](#)
- MMCSD_CMD_ALL_SEND_CID
 - MMC/SD Block Device, [1010](#)
- MMCSD_CMD_APP_CMD
 - MMC/SD Block Device, [1013](#)
- MMCSD_CMD_APP_OP_COND
 - MMC/SD Block Device, [1012](#)
- MMCSD_CMD_ERASE
 - MMC/SD Block Device, [1012](#)
- MMCSD_CMD_ERASE_RW_BLK_END
 - MMC/SD Block Device, [1012](#)
- MMCSD_CMD_ERASE_RW_BLK_START
 - MMC/SD Block Device, [1012](#)
- MMCSD_CMD_GO_IDLE_STATE
 - MMC/SD Block Device, [1010](#)
- MMCSD_CMD_INIT
 - MMC/SD Block Device, [1010](#)
- MMCSD_CMD_LOCK_UNLOCK
 - MMC/SD Block Device, [1013](#)
- MMCSD_CMD_READ_MULTIPLE_BLOCK
 - MMC/SD Block Device, [1012](#)
- MMCSD_CMD_READ_OCR
 - MMC/SD Block Device, [1013](#)
- MMCSD_CMD_READ_SINGLE_BLOCK
 - MMC/SD Block Device, [1012](#)
- MMCSD_CMD_SEL_DESEL_CARD
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_SEND_CID
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_SEND_CSD
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_SEND_EXT_CSD
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_SEND_IF_COND
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_SEND_RELATIVE_ADDR
 - MMC/SD Block Device, [1010](#)
- MMCSD_CMD_SEND_STATUS
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_SET_BLOCK_COUNT
 - MMC/SD Block Device, [1012](#)
- MMCSD_CMD_SET_BLOCKLEN
 - MMC/SD Block Device, [1012](#)
- MMCSD_CMD_SET_BUS_WIDTH
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_SET_RELATIVE_ADDR
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_STOP_TRANSMISSION
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_SWITCH
 - MMC/SD Block Device, [1011](#)
- MMCSD_CMD_WRITE_BLOCK
 - MMC/SD Block Device, [1012](#)
- MMCSD_CMD_WRITE_MULTIPLE_BLOCK
 - MMC/SD Block Device, [1012](#)
- MMCSD_CSD_10_C_SIZE_MULT_SLICE
 - MMC/SD Block Device, [1020](#)
- MMCSD_CSD_10_C_SIZE_SLICE
 - MMC/SD Block Device, [1020](#)
- MMCSD_CSD_10_CCC_SLICE
 - MMC/SD Block Device, [1021](#)
- MMCSD_CSD_10_COPY_SLICE
 - MMC/SD Block Device, [1019](#)
- MMCSD_CSD_10_CRC_SLICE
 - MMC/SD Block Device, [1019](#)
- MMCSD_CSD_10_CSD_STRUCTURE_SLICE
 - MMC/SD Block Device, [1021](#)
- MMCSD_CSD_10_DSR_IMP_SLICE
 - MMC/SD Block Device, [1020](#)
- MMCSD_CSD_10_ERASE_BLK_EN_SLICE
 - MMC/SD Block Device, [1020](#)
- MMCSD_CSD_10_ERASE_SECTOR_SIZE_SLICE
 - MMC/SD Block Device, [1020](#)
- MMCSD_CSD_10_FILE_FORMAT_GRP_SLICE
 - MMC/SD Block Device, [1019](#)
- MMCSD_CSD_10_FILE_FORMAT_SLICE
 - MMC/SD Block Device, [1019](#)

- MMCSDB_CSD_10_NSAC_SLICE
MMC/SD Block Device, [1021](#)
- MMCSDB_CSD_10_PERM_WRITE_PROTECT_SLICE
MMC/SD Block Device, [1019](#)
- MMCSDB_CSD_10_R2W_FACTOR_SLICE
MMC/SD Block Device, [1019](#)
- MMCSDB_CSD_10_READ_BL_LEN_SLICE
MMC/SD Block Device, [1021](#)
- MMCSDB_CSD_10_READ_BL_PARTIAL_SLICE
MMC/SD Block Device, [1021](#)
- MMCSDB_CSD_10_READ_BLK_MISALIGN_SLICE
MMC/SD Block Device, [1021](#)
- MMCSDB_CSD_10_TAAC_SLICE
MMC/SD Block Device, [1021](#)
- MMCSDB_CSD_10_TMP_WRITE_PROTECT_SLICE
MMC/SD Block Device, [1019](#)
- MMCSDB_CSD_10_TRANS_SPEED_SLICE
MMC/SD Block Device, [1021](#)
- MMCSDB_CSD_10_VDD_R_CURR_MAX_SLICE
MMC/SD Block Device, [1020](#)
- MMCSDB_CSD_10_VDD_R_CURR_MIX_SLICE
MMC/SD Block Device, [1020](#)
- MMCSDB_CSD_10_VDD_W_CURR_MAX_SLICE
MMC/SD Block Device, [1020](#)
- MMCSDB_CSD_10_VDD_W_CURR_MIN_SLICE
MMC/SD Block Device, [1020](#)
- MMCSDB_CSD_10_WP_GRP_ENABLE_SLICE
MMC/SD Block Device, [1019](#)
- MMCSDB_CSD_10_WP_GRP_SIZE_SLICE
MMC/SD Block Device, [1020](#)
- MMCSDB_CSD_10_WRITE_BL_LEN_SLICE
MMC/SD Block Device, [1019](#)
- MMCSDB_CSD_10_WRITE_BL_PARTIAL_SLICE
MMC/SD Block Device, [1019](#)
- MMCSDB_CSD_10_WRITE_BLK_MISALIGN_SLICE
MMC/SD Block Device, [1021](#)
- MMCSDB_CSD_20_C_SIZE_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_CCC_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_20_COPY_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_CRC_SLICE
MMC/SD Block Device, [1016](#)
- MMCSDB_CSD_20_CSD_STRUCTURE_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_20_DSR_IMP_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_20_ERASE_BLK_EN_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_ERASE_SECTOR_SIZE_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_FILE_FORMAT_GRP_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_FILE_FORMAT_SLICE
MMC/SD Block Device, [1016](#)
- MMCSDB_CSD_20_NSAC_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_20_PERM_WRITE_PROTECT_SLICE
MMC/SD Block Device, [1016](#)
- MMCSDB_CSD_20_R2W_FACTOR_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_READ_BL_LEN_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_20_READ_BL_PARTIAL_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_20_READ_BLK_MISALIGN_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_20_TAAC_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_20_TMP_WRITE_PROTECT_SLICE
MMC/SD Block Device, [1016](#)
- MMCSDB_CSD_20_TRANS_SPEED_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_20_WP_GRP_ENABLE_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_WP_GRP_SIZE_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_WRITE_BL_LEN_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_WRITE_BL_PARTIAL_SLICE
MMC/SD Block Device, [1017](#)
- MMCSDB_CSD_20_WRITE_BLK_MISALIGN_SLICE
MMC/SD Block Device, [1018](#)
- MMCSDB_CSD_MMC_C_SIZE_MULT_SLICE
MMC/SD Block Device, [1014](#)
- MMCSDB_CSD_MMC_C_SIZE_SLICE
MMC/SD Block Device, [1014](#)
- MMCSDB_CSD_MMC_CCC_SLICE
MMC/SD Block Device, [1013](#)
- MMCSDB_CSD_MMC_CONTENT_PROT_APP_SLICE
MMC/SD Block Device, [1015](#)
- MMCSDB_CSD_MMC_COPY_SLICE
MMC/SD Block Device, [1016](#)
- MMCSDB_CSD_MMC_CRC_SLICE
MMC/SD Block Device, [1016](#)
- MMCSDB_CSD_MMC_CSD_STRUCTURE_SLICE
MMC/SD Block Device, [1013](#)
- MMCSDB_CSD_MMC_DEFAULT_ECC_SLICE
MMC/SD Block Device, [1015](#)
- MMCSDB_CSD_MMC_DSR_IMP_SLICE
MMC/SD Block Device, [1014](#)
- MMCSDB_CSD_MMC_ECC_SLICE
MMC/SD Block Device, [1016](#)
- MMCSDB_CSD_MMC_ERASE_GRP_MULT_SLICE
MMC/SD Block Device, [1015](#)
- MMCSDB_CSD_MMC_ERASE_GRP_SIZE_SLICE
MMC/SD Block Device, [1015](#)
- MMCSDB_CSD_MMC_FILE_FORMAT_GRP_SLICE
MMC/SD Block Device, [1015](#)
- MMCSDB_CSD_MMC_FILE_FORMAT_SLICE
MMC/SD Block Device, [1016](#)
- MMCSDB_CSD_MMC_NSAC_SLICE
MMC/SD Block Device, [1013](#)
- MMCSDB_CSD_MMC_PERM_WRITE_PROTECT_SLICE
MMC/SD Block Device, [1016](#)

MMCSD_CSD_MMC_R2W_FACTOR_SLICE
 MMC/SD Block Device, [1015](#)
 MMCSD_CSD_MMC_READ_BL_LEN_SLICE
 MMC/SD Block Device, [1013](#)
 MMCSD_CSD_MMC_READ_BL_PARTIAL_SLICE
 MMC/SD Block Device, [1014](#)
 MMCSD_CSD_MMC_READ_BLK_MISALIGN_SLICE
 MMC/SD Block Device, [1014](#)
 MMCSD_CSD_MMC_SPEC_VERS_SLICE
 MMC/SD Block Device, [1013](#)
 MMCSD_CSD_MMC_TAAC_SLICE
 MMC/SD Block Device, [1013](#)
 MMCSD_CSD_MMC_TMP_WRITE_PROTECT_SLICE
 MMC/SD Block Device, [1016](#)
 MMCSD_CSD_MMC_TRAN_SPEED_SLICE
 MMC/SD Block Device, [1013](#)
 MMCSD_CSD_MMC_VDD_R_CURR_MAX_SLICE
 MMC/SD Block Device, [1014](#)
 MMCSD_CSD_MMC_VDD_R_CURR_MIN_SLICE
 MMC/SD Block Device, [1014](#)
 MMCSD_CSD_MMC_VDD_W_CURR_MAX_SLICE
 MMC/SD Block Device, [1014](#)
 MMCSD_CSD_MMC_VDD_W_CURR_MIN_SLICE
 MMC/SD Block Device, [1014](#)
 MMCSD_CSD_MMC_WP_GRP_ENABLE_SLICE
 MMC/SD Block Device, [1015](#)
 MMCSD_CSD_MMC_WP_GRP_SIZE_SLICE
 MMC/SD Block Device, [1015](#)
 MMCSD_CSD_MMC_WRITE_BL_LEN_SLICE
 MMC/SD Block Device, [1015](#)
 MMCSD_CSD_MMC_WRITE_BL_PARTIAL_SLICE
 MMC/SD Block Device, [1015](#)
 MMCSD_CSD_MMC_WRITE_BLK_MISALIGN_SLICE
 MMC/SD Block Device, [1014](#)
 MMCSD_R1_ERROR
 MMC/SD Block Device, [1024](#)
 MMCSD_R1_ERROR_MASK
 MMC/SD Block Device, [1009](#)
 MMCSD_R1_IS_CARD_LOCKED
 MMC/SD Block Device, [1025](#)
 MMCSD_R1_STS
 MMC/SD Block Device, [1025](#)
 MMCSD_STS_DATA
 MMC/SD Block Device, [1010](#)
 MMCSD_STS_DIS
 MMC/SD Block Device, [1010](#)
 MMCSD_STS_IDENT
 MMC/SD Block Device, [1009](#)
 MMCSD_STS_IDLE
 MMC/SD Block Device, [1009](#)
 MMCSD_STS_PRG
 MMC/SD Block Device, [1010](#)
 MMCSD_STS_RCV
 MMC/SD Block Device, [1010](#)
 MMCSD_STS_READY
 MMC/SD Block Device, [1009](#)
 MMCSD_STS_STBY
 MMC/SD Block Device, [1010](#)
 MMCSD_STS_TRAN
 MMC/SD Block Device, [1010](#)
 MMCSDBlockDevice, [1259](#)
 vmt, [1262](#)
 MMCSDBlockDeviceVMT, [1262](#)
 mmcCsdGetCardCapacity
 MMC/SD Block Device, [1025](#)
 mmcSequentialRead
 MMC over SPI Driver, [785](#)
 mmcSequentialWrite
 MMC over SPI Driver, [787](#)
 mmcStart
 MMC over SPI Driver, [780](#)
 mmcStartSequentialRead
 MMC over SPI Driver, [784](#)
 mmcStartSequentialWrite
 MMC over SPI Driver, [786](#)
 mmcStop
 MMC over SPI Driver, [782](#)
 mmcStopSequentialRead
 MMC over SPI Driver, [786](#)
 mmcStopSequentialWrite
 MMC over SPI Driver, [788](#)
 mmcSync
 MMC over SPI Driver, [789](#)
 mode
 ICUConfig, [1227](#)
 PWMChannelConfig, [1268](#)
 mode_detect
 SDC Driver, [409](#)
 month
 RTCDatetime, [1274](#)
 month_len
 RTC Driver, [398](#)
 MSG_OK
 OSAL, [1079](#)
 MSG_RESET
 OSAL, [1079](#)
 msg_t
 OSAL, [1089](#)
 MSG_TIMEOUT
 OSAL, [1079](#)
 msObjectInit
 Memory Streams Class, [954](#)
 mutex
 hal_adc_driver, [1162](#)
 hal_dac_driver, [1174](#)
 hal_efl_driver, [1177](#)
 hal_i2c_driver, [1179](#)
 hal_spi_driver, [1198](#)
 hal_uart_driver, [1204](#)
 hal_wspi_driver, [1206](#)
 hal_xsnor_base, [1209](#)
 hal_xsnor_macronix_mx25, [1214](#)
 hal_xsnor_micron_n25q, [1220](#)
 SNORDriver, [1297](#)
 mutex_t
 OSAL, [1091](#)

- mx25_find_id
 - Macronix MX25 SNOR Driver, [889](#)
- mx25_manufacturer_ids
 - Macronix MX25 SNOR Driver, [893](#)
- mx25_memory_type_ids
 - Macronix MX25 SNOR Driver, [893](#)
- MX25_OPT_DUMMY_CYCLES
 - Macronix MX25 SNOR Driver, [867](#)
- MX25_OPT_DUMMY_CYCLES_MASK
 - Macronix MX25 SNOR Driver, [867](#)
- MX25_OPT_NICE_WAITING
 - Macronix MX25 SNOR Driver, [868](#)
- MX25_OPT_NO_WIDTH_SWITCH
 - Macronix MX25 SNOR Driver, [867](#)
- MX25_OPT_USE_DTR
 - Macronix MX25 SNOR Driver, [868](#)
- MX25_OPT_USE_SUBSECTORS
 - Macronix MX25 SNOR Driver, [868](#)
- mx25_poll_status
 - Macronix MX25 SNOR Driver, [889](#)
- mx25_read_id
 - Macronix MX25 SNOR Driver, [890](#)
- mx25_reset_memory
 - Macronix MX25 SNOR Driver, [890](#)
- mx25_write_cr2
 - Macronix MX25 SNOR Driver, [891](#)
- mx25ObjectInit
 - Macronix MX25 SNOR Driver, [889](#)
- n25q_activate_xip
 - Micron N25Q Driver, [911](#)
- n25q_find_id
 - Micron N25Q Driver, [910](#)
- n25q_manufacturer_ids
 - Micron N25Q Driver, [915](#)
- n25q_memory_type_ids
 - Micron N25Q Driver, [915](#)
- N25Q_OPT_DUMMY_CYCLES
 - Micron N25Q Driver, [897](#)
- N25Q_OPT_DUMMY_CYCLES_MASK
 - Micron N25Q Driver, [897](#)
- N25Q_OPT_NICE_WAITING
 - Micron N25Q Driver, [898](#)
- N25Q_OPT_NO_WIDTH_SWITCH
 - Micron N25Q Driver, [898](#)
- N25Q_OPT_USE_SUBSECTORS
 - Micron N25Q Driver, [898](#)
- n25q_poll_status
 - Micron N25Q Driver, [911](#)
- n25q_read_id
 - Micron N25Q Driver, [912](#)
- n25q_reset_memory
 - Micron N25Q Driver, [912](#)
- n25q_reset_xip
 - Micron N25Q Driver, [912](#)
- n25qObjectInit
 - Micron N25Q Driver, [910](#)
- ncbuf
 - MFSDriver, [1253](#)
- next_offset
 - MFSDriver, [1252](#)
- nocache
 - SNORDriver, [1297](#)
- Normal Drivers, [55](#)
- notify
 - io_buffers_queue, [1232](#)
- nsac
 - unpacked_mmc_csd_t, [1303](#)
 - unpacked_sdc_csd_10_t, [1309](#)
 - unpacked_sdc_csd_20_t, [1313](#)
- Null Streams Class, [955](#)
 - _null_stream_data, [955](#)
 - get, [956](#)
 - nullObjectInit, [956](#)
 - put, [956](#)
 - reads, [956](#)
 - unget, [956](#)
 - vmt, [957](#)
 - writes, [956](#)
- nullObjectInit
 - Null Streams Class, [956](#)
- NullStream, [1263](#)
 - vmt, [1265](#)
- nullstreams.c, [1448](#)
- nullstreams.h, [1448](#)
- NullStreamVMT, [1265](#)
- num_channels
 - hal_adc_configuration_group, [1160](#)
 - hal_dac_conversion_group, [1172](#)
- NWAYTEST_LOOPBACK
 - MII/RMII Header, [1067](#)
- NWAYTEST_RESV1
 - MII/RMII Header, [1066](#)
- NWAYTEST_RESV2
 - MII/RMII Header, [1067](#)
- objGetInstance
 - Base Object, [979](#)
- obnotify
 - Serial over USB Driver, [822](#)
- obqFlush
 - I/O Buffers Queues, [1004](#)
- obqGetEmptyBufferTimeout
 - I/O Buffers Queues, [998](#)
- obqGetEmptyBufferTimeoutS
 - I/O Buffers Queues, [999](#)
- obqGetFullBufferI
 - I/O Buffers Queues, [997](#)
- obqIsEmptyI
 - I/O Buffers Queues, [987](#)
- obqIsFullI
 - I/O Buffers Queues, [987](#)
- obqObjectInit
 - I/O Buffers Queues, [996](#)
- obqPostFullBuffer
 - I/O Buffers Queues, [1000](#)
- obqPostFullBufferS
 - I/O Buffers Queues, [1001](#)

- obqPutTimeout
 - I/O Buffers Queues, [1001](#)
- obqReleaseEmptyBufferI
 - I/O Buffers Queues, [998](#)
- obqResetI
 - I/O Buffers Queues, [997](#)
- obqTryFlushI
 - I/O Buffers Queues, [1003](#)
- obqWriteTimeout
 - I/O Buffers Queues, [1002](#)
- offset
 - flash_sector_descriptor_t, [1156](#)
 - hal_mac_receive_descriptor, [1186](#)
 - hal_mac_transmit_descriptor, [1187](#)
 - IOBus, [1236](#)
 - mfs_record_descriptor_t, [1246](#)
 - mfs_transaction_op_t, [1247](#)
- oid
 - unpacked_mmc_cid_t, [1300](#)
 - unpacked_sdc_cid_t, [1307](#)
- options
 - xsnor_config, [1340](#)
- oq_write
 - I/O Bytes Queues, [1041](#)
- oqGetEmptyI
 - I/O Bytes Queues, [1038](#)
- oqGetFullI
 - I/O Bytes Queues, [1037](#)
- oqGetI
 - I/O Bytes Queues, [1050](#)
- oqlsEmptyI
 - I/O Bytes Queues, [1038](#)
- oqlsFullI
 - I/O Bytes Queues, [1039](#)
- oqObjectInit
 - I/O Bytes Queues, [1047](#)
- oqPut
 - I/O Bytes Queues, [1039](#)
- oqPutI
 - I/O Bytes Queues, [1049](#)
- oqPutTimeout
 - I/O Bytes Queues, [1049](#)
- oqResetI
 - I/O Bytes Queues, [1048](#)
- oqWritel
 - I/O Bytes Queues, [1051](#)
- oqWriteTimeout
 - I/O Bytes Queues, [1052](#)
- OSAL, [1073](#)
 - event_source_t, [1090](#)
 - eventcallback_t, [1090](#)
 - eventflags_t, [1090](#)
 - FALSE, [1079](#)
 - MSG_OK, [1079](#)
 - MSG_RESET, [1079](#)
 - msg_t, [1089](#)
 - MSG_TIMEOUT, [1079](#)
 - mutex_t, [1091](#)
 - OSAL_DBG_ENABLE_ASSERTS, [1080](#)
 - OSAL_DBG_ENABLE_CHECKS, [1080](#)
 - OSAL_FAILED, [1079](#)
 - osal_halt_msg, [1104](#)
 - OSAL_I2MS, [1085](#)
 - OSAL_I2S, [1085](#)
 - OSAL_I2US, [1086](#)
 - OSAL_IRQ_EPILOGUE, [1082](#)
 - OSAL_IRQ_HANDLER, [1083](#)
 - OSAL_IRQ_IS_VALID_PRIORITY, [1082](#)
 - OSAL_IRQ_MAXIMUM_PRIORITY, [1080](#)
 - OSAL_IRQ_PRIORITY_LEVELS, [1080](#)
 - OSAL_IRQ_PROLOGUE, [1082](#)
 - OSAL_MS2I, [1083](#)
 - OSAL_MS2RTC, [1087](#)
 - OSAL_S2I, [1083](#)
 - OSAL_S2RTC, [1086](#)
 - OSAL_ST_FREQUENCY, [1080](#)
 - OSAL_ST_MODE, [1080](#)
 - OSAL_ST_MODE_FREERUNNING, [1080](#)
 - OSAL_ST_MODE_NONE, [1079](#)
 - OSAL_ST_MODE_PERIODIC, [1079](#)
 - OSAL_ST_RESOLUTION, [1080](#)
 - OSAL_SUCCESS, [1079](#)
 - OSAL_US2I, [1084](#)
 - OSAL_US2RTC, [1087](#)
 - osalDbgAssert, [1081](#)
 - osalDbgCheck, [1081](#)
 - osalDbgCheckClassI, [1082](#)
 - osalDbgCheckClassS, [1082](#)
 - osalEventBroadcastFlags, [1097](#)
 - osalEventBroadcastFlagsI, [1097](#)
 - osalEventObjectInit, [1103](#)
 - osalEventSetCallback, [1098](#)
 - osalInit, [1091](#)
 - osalMutexLock, [1098](#)
 - osalMutexObjectInit, [1104](#)
 - osalMutexUnlock, [1099](#)
 - osalOsGetSystemTimeX, [1092](#)
 - osalOsRescheduleS, [1092](#)
 - osalOsTimerHandlerI, [1092](#)
 - osalSysDisable, [1099](#)
 - osalSysEnable, [1099](#)
 - osalSysGetStatusAndLockX, [1101](#)
 - osalSysHalt, [1091](#)
 - osalSysLock, [1100](#)
 - osalSysLockFromISR, [1100](#)
 - osalSysPolledDelayX, [1091](#)
 - osalSysRestoreStatusX, [1101](#)
 - osalSysUnlock, [1100](#)
 - osalSysUnlockFromISR, [1100](#)
 - osalThreadDequeueAllI, [1097](#)
 - osalThreadDequeueNextI, [1096](#)
 - osalThreadEnqueueTimeoutS, [1096](#)
 - osalThreadQueueObjectInit, [1103](#)
 - osalThreadResumeI, [1095](#)
 - osalThreadResumeS, [1095](#)
 - osalThreadSleep, [1093](#)

- osalThreadSleepMicroseconds, [1089](#)
- osalThreadSleepMilliseconds, [1088](#)
- osalThreadSleepS, [1093](#)
- osalThreadSleepSeconds, [1088](#)
- osalThreadSuspendS, [1094](#)
- osalThreadSuspendTimeoutS, [1094](#)
- osalTimeAddX, [1102](#)
- osalTimeDiffX, [1102](#)
- osalTimeInRangeX, [1102](#)
- rtcnt_t, [1090](#)
- sysinterval_t, [1090](#)
- syssts_t, [1089](#)
- systime_t, [1089](#)
- thread_reference_t, [1090](#)
- time_conv_t, [1090](#)
- TIME_IMMEDIATE, [1079](#)
- TIME_INFINITE, [1079](#)
- TRUE, [1079](#)
- osal.c, [1531](#)
- osal.h, [1532](#)
- OSAL_DBG_ENABLE_ASSERTS
 - OSAL, [1080](#)
- OSAL_DBG_ENABLE_CHECKS
 - OSAL, [1080](#)
- OSAL_FAILED
 - OSAL, [1079](#)
- osal_halt_msg
 - OSAL, [1104](#)
- OSAL_I2MS
 - OSAL, [1085](#)
- OSAL_I2S
 - OSAL, [1085](#)
- OSAL_I2US
 - OSAL, [1086](#)
- OSAL_IRQ_EPILOGUE
 - OSAL, [1082](#)
- OSAL_IRQ_HANDLER
 - OSAL, [1083](#)
- OSAL_IRQ_IS_VALID_PRIORITY
 - OSAL, [1082](#)
- OSAL_IRQ_MAXIMUM_PRIORITY
 - OSAL, [1080](#)
- OSAL_IRQ_PRIORITY_LEVELS
 - OSAL, [1080](#)
- OSAL_IRQ_PROLOGUE
 - OSAL, [1082](#)
- OSAL_MS2I
 - OSAL, [1083](#)
- OSAL_MS2RTC
 - OSAL, [1087](#)
- OSAL_S2I
 - OSAL, [1083](#)
- OSAL_S2RTC
 - OSAL, [1086](#)
- OSAL_ST_FREQUENCY
 - OSAL, [1080](#)
- OSAL_ST_MODE
 - OSAL, [1080](#)
- OSAL_ST_MODE_FREERUNNING
 - OSAL, [1080](#)
- OSAL_ST_MODE_NONE
 - OSAL, [1079](#)
- OSAL_ST_MODE_PERIODIC
 - OSAL, [1079](#)
- OSAL_ST_RESOLUTION
 - OSAL, [1080](#)
- OSAL_SUCCESS
 - OSAL, [1079](#)
- OSAL_US2I
 - OSAL, [1084](#)
- OSAL_US2RTC
 - OSAL, [1087](#)
- osalDbgAssert
 - OSAL, [1081](#)
- osalDbgCheck
 - OSAL, [1081](#)
- osalDbgCheckClassI
 - OSAL, [1082](#)
- osalDbgCheckClassS
 - OSAL, [1082](#)
- osalEventBroadcastFlags
 - OSAL, [1097](#)
- osalEventBroadcastFlagsI
 - OSAL, [1097](#)
- osalEventObjectInit
 - OSAL, [1103](#)
- osalEventSetCallback
 - OSAL, [1098](#)
- osalInit
 - OSAL, [1091](#)
- osalMutexLock
 - OSAL, [1098](#)
- osalMutexObjectInit
 - OSAL, [1104](#)
- osalMutexUnlock
 - OSAL, [1099](#)
- osalOsGetSystemTimeX
 - OSAL, [1092](#)
- osalOsRescheduleS
 - OSAL, [1092](#)
- osalOsTimerHandlerI
 - OSAL, [1092](#)
- osalSysDisable
 - OSAL, [1099](#)
- osalSysEnable
 - OSAL, [1099](#)
- osalSysGetStatusAndLockX
 - OSAL, [1101](#)
- osalSysHalt
 - OSAL, [1091](#)
- osalSysLock
 - OSAL, [1100](#)
- osalSysLockFromISR
 - OSAL, [1100](#)
- osalSysPolledDelayX
 - OSAL, [1091](#)

- osalSysRestoreStatusX
 - OSAL, [1101](#)
- osalSysUnlock
 - OSAL, [1100](#)
- osalSysUnlockFromISR
 - OSAL, [1100](#)
- osalThreadDequeueAll
 - OSAL, [1097](#)
- osalThreadDequeueNextI
 - OSAL, [1096](#)
- osalThreadEnqueueTimeoutS
 - OSAL, [1096](#)
- osalThreadQueueObjectInit
 - OSAL, [1103](#)
- osalThreadResumeI
 - OSAL, [1095](#)
- osalThreadResumeS
 - OSAL, [1095](#)
- osalThreadSleep
 - OSAL, [1093](#)
- osalThreadSleepMicroseconds
 - OSAL, [1089](#)
- osalThreadSleepMilliseconds
 - OSAL, [1088](#)
- osalThreadSleepS
 - OSAL, [1093](#)
- osalThreadSleepSeconds
 - OSAL, [1088](#)
- osalThreadSuspendS
 - OSAL, [1094](#)
- osalThreadSuspendTimeoutS
 - OSAL, [1094](#)
- osalTimeAddX
 - OSAL, [1102](#)
- osalTimeDiffX
 - OSAL, [1102](#)
- osalTimeIsInRangeX
 - OSAL, [1102](#)
- out
 - hal_usb_lld.c, [1523](#)
 - USB Driver, [684](#)
- out_cb
 - USBEndpointConfig, [1325](#)
- out_maxsize
 - USBEndpointConfig, [1325](#)
- out_params
 - USBDriver, [1321](#)
- out_state
 - USBEndpointConfig, [1325](#)
- Output Formatter Utility, [967](#)
 - ch_ltoa, [968](#)
 - chprintf, [969](#)
 - CHPRINTF_USE_FLOAT, [968](#)
 - chsnprintf, [970](#)
 - chvprintf, [968](#)
 - chvsnprintf, [972](#)
 - FLOAT_PRECISION, [967](#)
 - long_to_string_with_divisor, [968](#)
 - MAX_FILLER, [967](#)
- output_buffers_queue_t
 - I/O Buffers Queues, [988](#)
- output_queue_t
 - I/O Bytes Queues, [1040](#)
- overflow_cb
 - ICUConfig, [1228](#)
- PAGE_MASK
 - Macronix MX25 SNOR Driver, [868](#)
 - Micron N25Q Driver, [898](#)
- PAGE_SIZE
 - Macronix MX25 SNOR Driver, [868](#)
 - Micron N25Q Driver, [898](#)
- page_size
 - flash_descriptor_t, [1149](#)
- PAIR
 - Managed Flash Storage Driver, [741](#)
- PAL Driver, [308](#)
 - _IOBUS_DATA, [317](#)
 - _pal_clear_event, [319](#), [320](#)
 - _pal_init_event, [318](#), [319](#)
 - _pal_isr_code, [320](#), [321](#)
 - _pal_lld_init, [357](#)
 - _pal_lld_setgroupmode, [357](#)
 - IOBUS_DECL, [317](#)
 - ioline_t, [351](#)
 - iomode_t, [351](#)
 - iopadid_t, [351](#)
 - IOPORT1, [340](#)
 - ioportid_t, [351](#)
 - ioportmask_t, [351](#)
 - PAL_EVENT_MODE_BOTH_EDGES, [316](#)
 - PAL_EVENT_MODE_DISABLED, [315](#)
 - PAL_EVENT_MODE_EDGES_MASK, [315](#)
 - PAL_EVENT_MODE_FALLING_EDGE, [316](#)
 - PAL_EVENT_MODE_RISING_EDGE, [316](#)
 - PAL_GROUP_MASK, [317](#)
 - PAL_HIGH, [315](#)
 - PAL_IOPORTS_WIDTH, [339](#)
 - PAL_LINE, [340](#)
 - pal_lld_clearpad, [348](#)
 - pal_lld_clearport, [343](#)
 - pal_lld_get_line_event, [350](#)
 - pal_lld_get_pad_event, [350](#)
 - pal_lld_init, [341](#)
 - pal_lld_readgroup, [344](#)
 - pal_lld_readlatch, [341](#)
 - pal_lld_readpad, [346](#)
 - pal_lld_readport, [341](#)
 - pal_lld_setgroupmode, [346](#)
 - pal_lld_setpad, [347](#)
 - pal_lld_setpadmode, [349](#)
 - pal_lld_setport, [342](#)
 - pal_lld_togglepad, [348](#)
 - pal_lld_toggleport, [343](#)
 - pal_lld_writegroup, [344](#)
 - pal_lld_writepad, [347](#)
 - pal_lld_writeport, [342](#)

PAL_LOW, 315
 PAL_MODE_INPUT, 314
 PAL_MODE_INPUT_ANALOG, 315
 PAL_MODE_INPUT_PULLDOWN, 315
 PAL_MODE_INPUT_PULLUP, 314
 PAL_MODE_OUTPUT_OPENDRAIN, 315
 PAL_MODE_OUTPUT_PUSHPULL, 315
 PAL_MODE_RESET, 314
 PAL_MODE_UNCONNECTED, 314
 PAL_NEW_INIT, 339
 PAL_NOLINE, 340
 PAL_PAD, 340
 PAL_PORT, 340
 PAL_PORT_BIT, 316
 PAL_USE_CALLBACKS, 316
 PAL_USE_WAIT, 316
 PAL_WHOLE_PORT, 339
 palcallback_t, 351
 palClearLine, 332
 palClearPad, 329
 palClearPort, 324
 palDisableLineEvent, 337
 palDisableLineEventI, 336
 palDisablePadEvent, 335
 palDisablePadEventI, 334
 palEnableLineEvent, 336
 palEnableLineEventI, 335
 palEnablePadEvent, 334
 palEnablePadEventI, 334
 palInit, 321
 palIsLineEventEnabledX, 338
 palIsPadEventEnabledX, 337
 palReadBus, 351
 palReadGroup, 325
 palReadGroupLatch, 325
 palReadLatch, 322
 palReadLine, 330
 palReadPad, 327
 palReadPort, 322
 palSetBusMode, 352
 palSetGroupMode, 326
 palSetLine, 332
 palSetLineCallback, 339
 palSetLineCallbackI, 353
 palSetLineMode, 333
 palSetPad, 328
 palSetPadCallback, 338
 palSetPadCallbackI, 353
 palSetPadMode, 330
 palSetPort, 323
 palToggleLine, 333
 palTogglePad, 329
 palTogglePort, 324
 palWaitLineTimeout, 356
 palWaitLineTimeoutS, 355
 palWaitPadTimeout, 354
 palWaitPadTimeoutS, 353
 palWriteBus, 352
 palWriteGroup, 326
 palWriteLine, 331
 palWritePad, 328
 palWritePort, 323
 PAL_EVENT_MODE_BOTH_EDGES
 PAL Driver, 316
 PAL_EVENT_MODE_DISABLED
 PAL Driver, 315
 PAL_EVENT_MODE_EDGES_MASK
 PAL Driver, 315
 PAL_EVENT_MODE_FALLING_EDGE
 PAL Driver, 316
 PAL_EVENT_MODE_RISING_EDGE
 PAL Driver, 316
 PAL_GROUP_MASK
 PAL Driver, 317
 PAL_HIGH
 PAL Driver, 315
 PAL_IOPORTS_WIDTH
 PAL Driver, 339
 PAL_LINE
 PAL Driver, 340
 pal_lld_clearpad
 PAL Driver, 348
 pal_lld_clearport
 PAL Driver, 343
 pal_lld_get_line_event
 PAL Driver, 350
 pal_lld_get_pad_event
 PAL Driver, 350
 pal_lld_init
 PAL Driver, 341
 pal_lld_readgroup
 PAL Driver, 344
 pal_lld_readlatch
 PAL Driver, 341
 pal_lld_readpad
 PAL Driver, 346
 pal_lld_readport
 PAL Driver, 341
 pal_lld_setgroupmode
 PAL Driver, 346
 pal_lld_setpad
 PAL Driver, 347
 pal_lld_setpadmode
 PAL Driver, 349
 pal_lld_setport
 PAL Driver, 342
 pal_lld_togglepad
 PAL Driver, 348
 pal_lld_toggleport
 PAL Driver, 343
 pal_lld_writgroup
 PAL Driver, 344
 pal_lld_writepad
 PAL Driver, 347
 pal_lld_writport
 PAL Driver, 342

- PAL_LOW
 - PAL Driver, [315](#)
- PAL_MODE_INPUT
 - PAL Driver, [314](#)
- PAL_MODE_INPUT_ANALOG
 - PAL Driver, [315](#)
- PAL_MODE_INPUT_PULLDOWN
 - PAL Driver, [315](#)
- PAL_MODE_INPUT_PULLUP
 - PAL Driver, [314](#)
- PAL_MODE_OUTPUT_OPENDRAIN
 - PAL Driver, [315](#)
- PAL_MODE_OUTPUT_PUSHPULL
 - PAL Driver, [315](#)
- PAL_MODE_RESET
 - PAL Driver, [314](#)
- PAL_MODE_UNCONNECTED
 - PAL Driver, [314](#)
- PAL_NEW_INIT
 - PAL Driver, [339](#)
- PAL_NOLINE
 - PAL Driver, [340](#)
- PAL_PAD
 - PAL Driver, [340](#)
- PAL_PORT
 - PAL Driver, [340](#)
- PAL_PORT_BIT
 - PAL Driver, [316](#)
- PAL_USE_CALLBACKS
 - Configuration, [48](#)
 - PAL Driver, [316](#)
- PAL_USE_WAIT
 - Configuration, [49](#)
 - PAL Driver, [316](#)
- PAL_WHOLE_PORT
 - PAL Driver, [339](#)
- palcallback_t
 - PAL Driver, [351](#)
- palClearLine
 - PAL Driver, [332](#)
- palClearPad
 - PAL Driver, [329](#)
- palClearPort
 - PAL Driver, [324](#)
- PALConfig, [1265](#)
- palDisableLineEvent
 - PAL Driver, [337](#)
- palDisableLineEventI
 - PAL Driver, [336](#)
- palDisablePadEvent
 - PAL Driver, [335](#)
- palDisablePadEventI
 - PAL Driver, [334](#)
- palEnableLineEvent
 - PAL Driver, [336](#)
- palEnableLineEventI
 - PAL Driver, [335](#)
- palEnablePadEvent
 - PAL Driver, [334](#)
- palEnablePadEventI
 - PAL Driver, [334](#)
- palevent_t, [1266](#)
 - arg, [1267](#)
 - cb, [1267](#)
 - threads, [1267](#)
- palInit
 - PAL Driver, [321](#)
- pallsLineEventEnabledX
 - PAL Driver, [338](#)
- pallsPadEventEnabledX
 - PAL Driver, [337](#)
- palReadBus
 - PAL Driver, [351](#)
- palReadGroup
 - PAL Driver, [325](#)
- palReadGroupLatch
 - PAL Driver, [325](#)
- palReadLatch
 - PAL Driver, [322](#)
- palReadLine
 - PAL Driver, [330](#)
- palReadPad
 - PAL Driver, [327](#)
- palReadPort
 - PAL Driver, [322](#)
- palSetBusMode
 - PAL Driver, [352](#)
- palSetGroupMode
 - PAL Driver, [326](#)
- palSetLine
 - PAL Driver, [332](#)
- palSetLineCallback
 - PAL Driver, [339](#)
- palSetLineCallbackI
 - PAL Driver, [353](#)
- palSetLineMode
 - PAL Driver, [333](#)
- palSetPad
 - PAL Driver, [328](#)
- palSetPadCallback
 - PAL Driver, [338](#)
- palSetPadCallbackI
 - PAL Driver, [353](#)
- palSetPadMode
 - PAL Driver, [330](#)
- palSetPort
 - PAL Driver, [323](#)
- palToggleLine
 - PAL Driver, [333](#)
- palTogglePad
 - PAL Driver, [329](#)
- palTogglePort
 - PAL Driver, [324](#)
- palWaitLineTimeout
 - PAL Driver, [356](#)
- palWaitLineTimeoutS

- PAL Driver, [355](#)
- palWaitPadTimeout
 - PAL Driver, [354](#)
- palWaitPadTimeoutS
 - PAL Driver, [353](#)
- palWriteBus
 - PAL Driver, [352](#)
- palWriteGroup
 - PAL Driver, [326](#)
- palWriteLine
 - PAL Driver, [331](#)
- palWritePad
 - PAL Driver, [328](#)
- palWritePort
 - PAL Driver, [323](#)
- param
 - event_source, [1145](#)
- period
 - PWMConfig, [1270](#)
 - PWMDriver, [1272](#)
- period_cb
 - ICUConfig, [1228](#)
- Peripheral Interfaces, [29](#)
- perm_write_protect
 - unpacked_mmc_csd_t, [1306](#)
 - unpacked_sdc_csd_10_t, [1312](#)
 - unpacked_sdc_csd_20_t, [1315](#)
- PLATFORM_ADC_USE_ADC1
 - ADC Driver, [77](#)
- PLATFORM_CAN_USE_CAN1
 - CAN Driver, [96](#)
- PLATFORM_CRY_USE_CRY1
 - Crypto Driver, [118](#)
- PLATFORM_DAC_USE_DAC1
 - DAC Driver, [189](#)
- PLATFORM_EFL_USE_EFL1
 - EFL Driver, [207](#)
- PLATFORM_GPT_USE_GPT1
 - GPT Driver, [222](#)
- PLATFORM_I2C_USE_I2C1
 - I2C Driver, [240](#)
- PLATFORM_I2S_USE_I2S1
 - I2S Driver, [257](#)
- PLATFORM_ICU_USE_ICU1
 - ICU Driver, [272](#)
- PLATFORM_MAC_USE_MAC1
 - MAC Driver, [294](#)
- PLATFORM_MCUCONF
 - mcuconf.h, [1531](#)
- PLATFORM_NAME
 - HAL Driver, [63](#)
- PLATFORM_PWM_USE_PWM1
 - PWM Driver, [369](#)
- PLATFORM_RTC_USE_RTC1
 - RTC Driver, [389](#)
- PLATFORM_SDC_USE_SDC1
 - SDC Driver, [406](#)
- PLATFORM_SERIAL_USE_USART1
 - Serial Driver, [448](#)
- PLATFORM_SIO_USE_SIO1
 - SIO Driver, [482](#)
- PLATFORM_SPI_USE_SPI1
 - SPIv1 Driver, [523](#)
 - SPIv2 Driver, [554](#)
- PLATFORM_TRNG_USE_TRNG1
 - TRNG Driver, [594](#)
- PLATFORM_UART_USE_UART1
 - UART Driver, [614](#)
- PLATFORM_USB_USE_USB1
 - USB Driver, [657](#)
- PLATFORM_WDG_USE_WDG1
 - WDG Driver, [687](#)
- PLATFORM_WSPI_USE_WSPI1
 - WSPI Driver, [706](#)
- pnm
 - unpacked_mmc_cid_t, [1300](#)
 - unpacked_sdc_cid_t, [1307](#)
- portid
 - IOBus, [1236](#)
- program
 - flash_interface_vmt, [1154](#)
 - hal_xsnor_base_vmt, [1211](#)
 - hal_xsnor_macronix_mx25_vmt, [1217](#)
 - hal_xsnor_micron_n25q_vmt, [1223](#)
- prv_m
 - unpacked_mmc_cid_t, [1301](#)
 - unpacked_sdc_cid_t, [1307](#)
- prv_n
 - unpacked_mmc_cid_t, [1301](#)
 - unpacked_sdc_cid_t, [1307](#)
- PS_ERROR_HW_FAILURE
 - Abstract Persistent Storage Interface, [952](#)
- PS_ERROR_READ
 - Abstract Persistent Storage Interface, [952](#)
- ps_error_t
 - Abstract Persistent Storage Interface, [952](#)
- PS_ERROR_VERIFY
 - Abstract Persistent Storage Interface, [952](#)
- PS_ERROR_WRITE
 - Abstract Persistent Storage Interface, [952](#)
- PS_NO_ERROR
 - Abstract Persistent Storage Interface, [952](#)
- ps_offset_t
 - Abstract Persistent Storage Interface, [952](#)
- psGetStorageSize
 - Abstract Persistent Storage Interface, [950](#)
- psn
 - unpacked_mmc_cid_t, [1301](#)
 - unpacked_sdc_cid_t, [1307](#)
- psRead
 - Abstract Persistent Storage Interface, [951](#)
- psWrite
 - Abstract Persistent Storage Interface, [951](#)
- ptr
 - io_buffers_queue, [1232](#)
- put

- Null Streams Class, 956
- PWM Driver, 358
 - PLATFORM_PWM_USE_PWM1, 369
 - PWM_CHANNELS, 369
 - PWM_DEGREES_TO_WIDTH, 362
 - PWM_FRACTION_TO_WIDTH, 362
 - pwm_lld_change_period, 370
 - pwm_lld_disable_channel, 382
 - pwm_lld_disable_channel_notification, 384
 - pwm_lld_disable_periodic_notification, 383
 - pwm_lld_enable_channel, 381
 - pwm_lld_enable_channel_notification, 383
 - pwm_lld_enable_periodic_notification, 382
 - pwm_lld_init, 380
 - pwm_lld_start, 380
 - pwm_lld_stop, 381
 - PWM_OUTPUT_ACTIVE_HIGH, 361
 - PWM_OUTPUT_ACTIVE_LOW, 361
 - PWM_OUTPUT_DISABLED, 361
 - PWM_OUTPUT_MASK, 361
 - PWM_PERCENTAGE_TO_WIDTH, 363
 - PWM_READY, 371
 - PWM_STOP, 371
 - PWM_UNINIT, 371
 - pwmcallback_t, 370
 - pwmChangePeriod, 374
 - pwmChangePeriodI, 363
 - pwmchannel_t, 371
 - pwmchnmsk_t, 371
 - pwmcnt_t, 371
 - PWMD1, 384
 - pwmDisableChannel, 376
 - pwmDisableChannelI, 366
 - pwmDisableChannelNotification, 379
 - pwmDisableChannelNotificationI, 369
 - pwmDisablePeriodicNotification, 377
 - pwmDisablePeriodicNotificationI, 367
 - PWMDriver, 370
 - pwmEnableChannel, 375
 - pwmEnableChannelI, 364
 - pwmEnableChannelNotification, 378
 - pwmEnableChannelNotificationI, 368
 - pwmEnablePeriodicNotification, 377
 - pwmEnablePeriodicNotificationI, 367
 - pwmInit, 372
 - pwmIsChannelEnabledI, 366
 - pwmmode_t, 371
 - pwmObjectInit, 372
 - pwmStart, 372
 - pwmstate_t, 371
 - pwmStop, 373
- PWM_CHANNELS
 - PWM Driver, 369
- PWM_DEGREES_TO_WIDTH
 - PWM Driver, 362
- PWM_FRACTION_TO_WIDTH
 - PWM Driver, 362
- pwm_lld_change_period
 - PWM Driver, 370
- pwm_lld_disable_channel
 - PWM Driver, 382
- pwm_lld_disable_channel_notification
 - PWM Driver, 384
- pwm_lld_disable_periodic_notification
 - PWM Driver, 383
- pwm_lld_enable_channel
 - PWM Driver, 381
- pwm_lld_enable_channel_notification
 - PWM Driver, 383
- pwm_lld_enable_periodic_notification
 - PWM Driver, 382
- pwm_lld_init
 - PWM Driver, 380
- pwm_lld_start
 - PWM Driver, 380
- pwm_lld_stop
 - PWM Driver, 381
- PWM_OUTPUT_ACTIVE_HIGH
 - PWM Driver, 361
- PWM_OUTPUT_ACTIVE_LOW
 - PWM Driver, 361
- PWM_OUTPUT_DISABLED
 - PWM Driver, 361
- PWM_OUTPUT_MASK
 - PWM Driver, 361
- PWM_PERCENTAGE_TO_WIDTH
 - PWM Driver, 363
- PWM_READY
 - PWM Driver, 371
- PWM_STOP
 - PWM Driver, 371
- PWM_UNINIT
 - PWM Driver, 371
- pwmcallback_t
 - PWM Driver, 370
- pwmChangePeriod
 - PWM Driver, 374
- pwmChangePeriodI
 - PWM Driver, 363
- pwmchannel_t
 - PWM Driver, 371
- PWMChannelConfig, 1267
 - callback, 1268
 - mode, 1268
- pwmchnmsk_t
 - PWM Driver, 371
- pwmcnt_t
 - PWM Driver, 371
- PWMConfig, 1268
 - callback, 1270
 - channels, 1270
 - frequency, 1270
 - period, 1270
- PWMD1
 - PWM Driver, 384
- pwmDisableChannel

- PWM Driver, [376](#)
- pwmDisableChannel
 - PWM Driver, [366](#)
- pwmDisableChannelNotification
 - PWM Driver, [379](#)
- pwmDisableChannelNotificationI
 - PWM Driver, [369](#)
- pwmDisablePeriodicNotification
 - PWM Driver, [377](#)
- pwmDisablePeriodicNotificationI
 - PWM Driver, [367](#)
- PWMDriver, [1271](#)
 - channels, [1272](#)
 - config, [1272](#)
 - enabled, [1272](#)
 - period, [1272](#)
 - PWM Driver, [370](#)
 - state, [1272](#)
- pwmEnableChannel
 - PWM Driver, [375](#)
- pwmEnableChannelI
 - PWM Driver, [364](#)
- pwmEnableChannelNotification
 - PWM Driver, [378](#)
- pwmEnableChannelNotificationI
 - PWM Driver, [368](#)
- pwmEnablePeriodicNotification
 - PWM Driver, [377](#)
- pwmEnablePeriodicNotificationI
 - PWM Driver, [367](#)
- pwmInit
 - PWM Driver, [372](#)
- pwmIsChannelEnabledI
 - PWM Driver, [366](#)
- pwmmode_t
 - PWM Driver, [371](#)
- pwmObjectInit
 - PWM Driver, [372](#)
- pwmStart
 - PWM Driver, [372](#)
- pwmstate_t
 - PWM Driver, [371](#)
- pwmStop
 - PWM Driver, [373](#)
- q_buffer
 - io_queue, [1234](#)
- q_counter
 - io_queue, [1234](#)
- Q_EMPTY
 - I/O Bytes Queues, [1033](#)
- Q_FULL
 - I/O Bytes Queues, [1033](#)
- q_link
 - io_queue, [1235](#)
- q_notify
 - io_queue, [1235](#)
- Q_OK
 - I/O Bytes Queues, [1032](#)
- q_rdptr
 - io_queue, [1235](#)
- Q_RESET
 - I/O Bytes Queues, [1032](#)
- Q_TIMEOUT
 - I/O Bytes Queues, [1032](#)
- q_top
 - io_queue, [1234](#)
- q_waiting
 - io_queue, [1234](#)
- q_wrptr
 - io_queue, [1235](#)
- qGetLink
 - I/O Bytes Queues, [1034](#)
- qnotify_t
 - I/O Bytes Queues, [1040](#)
- qSetLink
 - I/O Bytes Queues, [1034](#)
- qSizeX
 - I/O Bytes Queues, [1033](#)
- qSpacel
 - I/O Bytes Queues, [1033](#)
- query_erase
 - flash_interface_vmt, [1154](#)
 - hal_xsnor_base_vmt, [1211](#)
 - hal_xsnor_macronix_mx25_vmt, [1217](#)
 - hal_xsnor_micron_n25q_vmt, [1223](#)
- r2w_factor
 - unpacked_mmc_csd_t, [1305](#)
 - unpacked_sdc_csd_10_t, [1311](#)
 - unpacked_sdc_csd_20_t, [1315](#)
- rca
 - SDCDriver, [1280](#)
- rdqueue
 - hal_mac_driver, [1184](#)
- read
 - flash_interface_vmt, [1154](#)
 - hal_xsnor_base_vmt, [1211](#)
 - hal_xsnor_macronix_mx25_vmt, [1217](#)
 - hal_xsnor_micron_n25q_vmt, [1223](#)
- read_bl_len
 - unpacked_mmc_csd_t, [1303](#)
 - unpacked_sdc_csd_10_t, [1310](#)
 - unpacked_sdc_csd_20_t, [1314](#)
- read_bl_partial
 - unpacked_mmc_csd_t, [1303](#)
 - unpacked_sdc_csd_10_t, [1310](#)
 - unpacked_sdc_csd_20_t, [1314](#)
- read_blk_misalign
 - unpacked_mmc_csd_t, [1304](#)
 - unpacked_sdc_csd_10_t, [1310](#)
 - unpacked_sdc_csd_20_t, [1314](#)
- reads
 - Null Streams Class, [956](#)
- receiving
 - USBDriver, [1320](#)
- release_exclusive
 - flash_interface_vmt, [1155](#)

- requests_hook_cb
 - USBConfig, [1317](#)
- reserved1
 - mfs_bank_header_t, [1241](#)
- RET_ON_ERROR
 - Managed Flash Storage Driver, [742](#)
- RTC Driver, [385](#)
 - _rtc_driver_methods, [389](#)
 - accu_month_len, [398](#)
 - month_len, [398](#)
 - PLATFORM_RTC_USE_RTC1, [389](#)
 - RTC_ALARMS, [389](#)
 - RTC_BASE_YEAR, [387](#)
 - RTC_DAY_CATURDAY, [388](#)
 - RTC_DAY_FRIDAY, [388](#)
 - RTC_DAY_MONDAY, [388](#)
 - RTC_DAY_SATURDAY, [388](#)
 - RTC_DAY_SUNDAY, [388](#)
 - RTC_DAY_THURSDAY, [388](#)
 - RTC_DAY_TUESDAY, [388](#)
 - RTC_DAY_WEDNESDAY, [388](#)
 - RTC_EVENT_SECOND, [390](#)
 - RTC_FAT_DATE_DAYS_MASK, [387](#)
 - RTC_FAT_DATE_MONTHS_MASK, [388](#)
 - RTC_FAT_DATE_YEARS_MASK, [388](#)
 - RTC_FAT_TIME_HOURS_MASK, [387](#)
 - RTC_FAT_TIME_MINUTES_MASK, [387](#)
 - RTC_FAT_TIME_SECONDS_MASK, [387](#)
 - RTC_HAS_STORAGE, [389](#)
 - rtc_lld_driver_fields, [389](#)
 - rtc_lld_get_alarm, [397](#)
 - rtc_lld_get_time, [397](#)
 - rtc_lld_init, [396](#)
 - rtc_lld_set_alarm, [397](#)
 - rtc_lld_set_callback, [398](#)
 - rtc_lld_set_time, [396](#)
 - RTC_SUPPORTS_CALLBACKS, [389](#)
 - rtcalarm_t, [390](#)
 - rtccb_t, [390](#)
 - rtcConvertDateTimeToFAT, [395](#)
 - rtcConvertDateTimeToStructTm, [394](#)
 - rtcConvertStructTmToDateTime, [395](#)
 - RTCDriver, [390](#)
 - rtcevent_t, [390](#)
 - rtcGetAlarm, [393](#)
 - rtcGetTime, [391](#)
 - rtcInit, [390](#)
 - rtcObjectInit, [390](#)
 - rtcSetAlarm, [392](#)
 - rtcSetCallback, [394](#)
 - rtcSetTime, [391](#)
- RTC_ALARMS
 - RTC Driver, [389](#)
- RTC_BASE_YEAR
 - RTC Driver, [387](#)
- RTC_DAY_CATURDAY
 - RTC Driver, [388](#)
- RTC_DAY_FRIDAY
 - RTC Driver, [388](#)
- RTC_DAY_MONDAY
 - RTC Driver, [388](#)
- RTC_DAY_SATURDAY
 - RTC Driver, [388](#)
- RTC_DAY_SUNDAY
 - RTC Driver, [388](#)
- RTC_DAY_THURSDAY
 - RTC Driver, [388](#)
- RTC_DAY_TUESDAY
 - RTC Driver, [388](#)
- RTC_DAY_WEDNESDAY
 - RTC Driver, [388](#)
- RTC_EVENT_SECOND
 - RTC Driver, [390](#)
- RTC_FAT_DATE_DAYS_MASK
 - RTC Driver, [387](#)
- RTC_FAT_DATE_MONTHS_MASK
 - RTC Driver, [388](#)
- RTC_FAT_DATE_YEARS_MASK
 - RTC Driver, [388](#)
- RTC_FAT_TIME_HOURS_MASK
 - RTC Driver, [387](#)
- RTC_FAT_TIME_MINUTES_MASK
 - RTC Driver, [387](#)
- RTC_FAT_TIME_SECONDS_MASK
 - RTC Driver, [387](#)
- RTC_HAS_STORAGE
 - RTC Driver, [389](#)
- rtc_lld_driver_fields
 - RTC Driver, [389](#)
- rtc_lld_get_alarm
 - RTC Driver, [397](#)
- rtc_lld_get_time
 - RTC Driver, [397](#)
- rtc_lld_init
 - RTC Driver, [396](#)
- rtc_lld_set_alarm
 - RTC Driver, [397](#)
- rtc_lld_set_callback
 - RTC Driver, [398](#)
- rtc_lld_set_time
 - RTC Driver, [396](#)
- RTC_SUPPORTS_CALLBACKS
 - RTC Driver, [389](#)
- RTCArm, [1272](#)
 - dummy, [1273](#)
- rtcalarm_t
 - RTC Driver, [390](#)
- rtccb_t
 - RTC Driver, [390](#)
- rtcConvertDateTimeToFAT
 - RTC Driver, [395](#)
- rtcConvertDateTimeToStructTm
 - RTC Driver, [394](#)
- rtcConvertStructTmToDateTime
 - RTC Driver, [395](#)
- RTCDateTime, [1273](#)

- day, [1274](#)
- dayofweek, [1274](#)
- dstflag, [1274](#)
- millisecond, [1274](#)
- month, [1274](#)
- year, [1274](#)
- RTCDriver, [1275](#)
 - RTC Driver, [390](#)
 - vmt, [1275](#)
- RTCDriverVMT, [1276](#)
- rtcevent_t
 - RTC Driver, [390](#)
- rtcGetAlarm
 - RTC Driver, [393](#)
- rtcGetTime
 - RTC Driver, [391](#)
- rtclnit
 - RTC Driver, [390](#)
- rtcnt_t
 - OSAL, [1090](#)
- rtcObjectInit
 - RTC Driver, [390](#)
- rtcSetAlarm
 - RTC Driver, [392](#)
- rtcSetCallback
 - RTC Driver, [394](#)
- rtcSetTime
 - RTC Driver, [391](#)
- RTR
 - CANRxFrame, [1136](#)
 - CANTxFrame, [1138](#)
- rx_buffer
 - hal_i2s_config, [1180](#)
- rxbuf
 - USBOutEndpointState, [1328](#)
- rxchar_cb
 - hal_uart_config, [1201](#)
- rxcnt
 - USBOutEndpointState, [1328](#)
- rxend_cb
 - hal_uart_config, [1201](#)
- rxerr_cb
 - hal_uart_config, [1201](#)
- rxfull_event
 - hal_can_driver, [1169](#)
- rxqueue
 - hal_can_driver, [1169](#)
- rxsize
 - USBOutEndpointState, [1328](#)
- rxstate
 - hal_uart_driver, [1203](#)
- samples
 - hal_adc_driver, [1162](#)
 - hal_dac_driver, [1174](#)
- saved_state
 - USBDriver, [1322](#)
- SD1
 - Serial Driver, [462](#)
- SD_BREAK_DETECTED
 - Serial Driver, [439](#)
- SD_FRAMING_ERROR
 - Serial Driver, [439](#)
- sd_lld_init
 - Serial Driver, [461](#)
- sd_lld_start
 - Serial Driver, [461](#)
- sd_lld_stop
 - Serial Driver, [462](#)
- SD_NOISE_ERROR
 - Serial Driver, [439](#)
- SD_OVERRUN_ERROR
 - Serial Driver, [439](#)
- SD_PARITY_ERROR
 - Serial Driver, [439](#)
- SD_QUEUE_FULL_ERROR
 - Serial Driver, [439](#)
- SD_READY
 - Serial Driver, [450](#)
- SD_STOP
 - Serial Driver, [450](#)
- SD_SWITCH_CHECK
 - SDC Driver, [408](#)
- SD_SWITCH_FUNCTION_CMD_SYSTEM
 - SDC Driver, [408](#)
- SD_SWITCH_FUNCTION_CURRENT_LIMIT
 - SDC Driver, [408](#)
- SD_SWITCH_FUNCTION_DRIVER_STRENGTH
 - SDC Driver, [408](#)
- SD_SWITCH_FUNCTION_SPEED
 - SDC Driver, [408](#)
- sd_switch_function_t
 - SDC Driver, [408](#)
- SD_SWITCH_SET
 - SDC Driver, [408](#)
- sd_switch_t
 - SDC Driver, [408](#)
- SD_UNINIT
 - Serial Driver, [450](#)
- sdAsynchronousRead
 - Serial Driver, [448](#)
- sdAsynchronousWrite
 - Serial Driver, [446](#)
- SDC Driver, [399](#)
 - _sdc_driver_methods, [407](#)
 - _sdc_wait_for_transfer_state, [418](#)
 - _sdc_wait_for_transfer_state_internal, [417](#)
 - _sdc_wait_for_transfer_state_nocrc, [418](#)
 - detect_bus_clk, [414](#)
 - mmc_cmd6_construct, [411](#)
 - mmc_detect_bus_clk, [414](#)
 - mmc_init, [409](#)
 - mmc_set_bus_width, [416](#)
 - MMC_SWITCH_CLEAR_BITS, [408](#)
 - MMC_SWITCH_COMMAND_SET, [408](#)
 - MMC_SWITCH_SET_BITS, [408](#)
 - mmc_switch_t, [407](#)

MMC_SWITCH_WRITE_BYTE, 408
mode_detect, 409
PLATFORM_SDC_USE_SDC1, 406
SD_SWITCH_CHECK, 408
SD_SWITCH_FUNCTION_CMD_SYSTEM, 408
SD_SWITCH_FUNCTION_CURRENT_LIMIT, 408
SD_SWITCH_FUNCTION_DRIVER_STRENGTH, 408
SD_SWITCH_FUNCTION_SPEED, 408
sd_switch_function_t, 408
SD_SWITCH_SET, 408
sd_switch_t, 408
SDC_CLK_25MHz, 409
SDC_CLK_50MHz, 409
sd_cmd6_check_status, 412
sd_cmd6_construct, 411
sd_cmd6_extract_info, 412
SDC_CMD_CRC_ERROR, 403
SDC_COMMAND_TIMEOUT, 404
SDC_DATA_CRC_ERROR, 403
SDC_DATA_TIMEOUT, 403
sd_detect_bus_clk, 413
sdc_init, 410
SDC_INIT_OCR, 405
SDC_INIT_OCR_V20, 405
SDC_INIT_RETRY, 404
sd_c_lld_init, 428
sd_c_lld_is_card_inserted, 435
sd_c_lld_is_write_protected, 435
sd_c_lld_read, 433
sd_c_lld_read_special, 434
sd_c_lld_send_cmd_long_crc, 432
sd_c_lld_send_cmd_none, 431
sd_c_lld_send_cmd_short, 431
sd_c_lld_send_cmd_short_crc, 432
sd_c_lld_set_bus_mode, 430
sd_c_lld_set_data_clk, 430
sd_c_lld_start, 429
sd_c_lld_start_clk, 429
sd_c_lld_stop, 429
sd_c_lld_stop_clk, 430
sd_c_lld_sync, 434
sd_c_lld_write, 433
SDC_MMC_SUPPORT, 404
SDC_MODE_1BIT, 408
SDC_MODE_4BIT, 408
SDC_MODE_8BIT, 408
SDC_MODE_CARDTYPE_MASK, 403
SDC_MODE_CARDTYPE_MMC, 403
SDC_MODE_CARDTYPE_SDV11, 403
SDC_MODE_CARDTYPE_SDV20, 403
SDC_MODE_HIGH_CAPACITY, 403
SDC_NICE_WAITING, 405
SDC_NO_ERROR, 403
SDC_OVERFLOW_ERROR, 404
SDC_RX_OVERRUN, 404
sd_set_bus_width, 415
SDC_STARTBIT_ERROR, 404
SDC_TX_UNDERRUN, 404
SDC_UNHANDLED_ERROR, 404
sd_c_vmt, 435
sdcbusclk_t, 408
sdcbusmode_t, 408
sdConnect, 421
SDCD1, 435
sdDisconnect, 423
SDC_Driver, 407
sdErase, 427
sdflags_t, 407
sdGetAndClearErrors, 426
sdGetInfo, 427
sdInit, 419
sdIsCardInserted, 405
sdIsWriteProtected, 406
sdmode_t, 407
sdObjectInit, 420
sdRead, 424
sdStart, 420
sdStop, 421
sdSync, 426
sdWrite, 425
SDC_CLK_25MHz
 SDC Driver, 409
SDC_CLK_50MHz
 SDC Driver, 409
sd_cmd6_check_status
 SDC Driver, 412
sd_cmd6_construct
 SDC Driver, 411
sd_cmd6_extract_info
 SDC Driver, 412
SDC_CMD_CRC_ERROR
 SDC Driver, 403
SDC_COMMAND_TIMEOUT
 SDC Driver, 404
SDC_DATA_CRC_ERROR
 SDC Driver, 403
SDC_DATA_TIMEOUT
 SDC Driver, 403
sd_detect_bus_clk
 SDC Driver, 413
sd_c_init
 SDC Driver, 410
SDC_INIT_OCR
 Configuration, 52
 SDC Driver, 405
SDC_INIT_OCR_V20
 Configuration, 52
 SDC Driver, 405
SDC_INIT_RETRY
 Configuration, 51
 SDC Driver, 404
sd_c_lld_init
 SDC Driver, 428
sd_c_lld_is_card_inserted
 SDC Driver, 435

sdc_lla_is_write_protected
 SDC Driver, [435](#)
 sdc_lla_read
 SDC Driver, [433](#)
 sdc_lla_read_special
 SDC Driver, [434](#)
 sdc_lla_send_cmd_long_crc
 SDC Driver, [432](#)
 sdc_lla_send_cmd_none
 SDC Driver, [431](#)
 sdc_lla_send_cmd_short
 SDC Driver, [431](#)
 sdc_lla_send_cmd_short_crc
 SDC Driver, [432](#)
 sdc_lla_set_bus_mode
 SDC Driver, [430](#)
 sdc_lla_set_data_clk
 SDC Driver, [430](#)
 sdc_lla_start
 SDC Driver, [429](#)
 sdc_lla_start_clk
 SDC Driver, [429](#)
 sdc_lla_stop
 SDC Driver, [429](#)
 sdc_lla_stop_clk
 SDC Driver, [430](#)
 sdc_lla_sync
 SDC Driver, [434](#)
 sdc_lla_write
 SDC Driver, [433](#)
 SDC_MMC_SUPPORT
 Configuration, [52](#)
 SDC Driver, [404](#)
 SDC_MODE_1BIT
 SDC Driver, [408](#)
 SDC_MODE_4BIT
 SDC Driver, [408](#)
 SDC_MODE_8BIT
 SDC Driver, [408](#)
 SDC_MODE_CARDTYPE_MASK
 SDC Driver, [403](#)
 SDC_MODE_CARDTYPE_MMC
 SDC Driver, [403](#)
 SDC_MODE_CARDTYPE_SDV11
 SDC Driver, [403](#)
 SDC_MODE_CARDTYPE_SDV20
 SDC Driver, [403](#)
 SDC_MODE_HIGH_CAPACITY
 SDC Driver, [403](#)
 SDC_NICE_WAITING
 Configuration, [52](#)
 SDC Driver, [405](#)
 SDC_NO_ERROR
 SDC Driver, [403](#)
 SDC_OVERFLOW_ERROR
 SDC Driver, [404](#)
 SDC_RX_OVERRUN
 SDC Driver, [404](#)
 sdc_set_bus_width
 SDC Driver, [415](#)
 SDC_STARTBIT_ERROR
 SDC Driver, [404](#)
 SDC_TX_UNDERRUN
 SDC Driver, [404](#)
 SDC_UNHANDLED_ERROR
 SDC Driver, [404](#)
 sdc_vmt
 SDC Driver, [435](#)
 sdcbusclk_t
 SDC Driver, [408](#)
 sdcbusmode_t
 SDC Driver, [408](#)
 SDCConfig, [1278](#)
 bus_width, [1278](#)
 sdcConnect
 SDC Driver, [421](#)
 SDCD1
 SDC Driver, [435](#)
 sdcDisconnect
 SDC Driver, [423](#)
 SDCDriver, [1279](#)
 buf, [1280](#)
 cardmode, [1280](#)
 config, [1280](#)
 errors, [1280](#)
 rca, [1280](#)
 SDC Driver, [407](#)
 vmt, [1280](#)
 SDCDriverVMT, [1281](#)
 sdcErase
 SDC Driver, [427](#)
 sdcflags_t
 SDC Driver, [407](#)
 sdcGetAndClearErrors
 SDC Driver, [426](#)
 sdcGetInfo
 SDC Driver, [427](#)
 sdclnit
 SDC Driver, [419](#)
 sdclsCardInserted
 SDC Driver, [405](#)
 sdclsWriteProtected
 SDC Driver, [406](#)
 sdcmode_t
 SDC Driver, [407](#)
 sdcObjectInit
 SDC Driver, [420](#)
 sdControl
 Serial Driver, [460](#)
 sdcRead
 SDC Driver, [424](#)
 sdcStart
 SDC Driver, [420](#)
 sdcStop
 SDC Driver, [421](#)
 sdcSync

- SDC Driver, [426](#)
- sdcWrite
 - SDC Driver, [425](#)
- sdGet
 - Serial Driver, [443](#)
- sdGetl
 - Serial Driver, [442](#)
- sdGetTimeout
 - Serial Driver, [443](#)
- sdGetWouldBlock
 - Serial Driver, [459](#)
- sdIncomingDataI
 - Serial Driver, [457](#)
- sdInit
 - Serial Driver, [454](#)
- sdObjectInit
 - Serial Driver, [454](#)
- sdPut
 - Serial Driver, [440](#)
- sdPutl
 - Serial Driver, [440](#)
- sdPutTimeout
 - Serial Driver, [441](#)
- sdPutWouldBlock
 - Serial Driver, [458](#)
- sdRead
 - Serial Driver, [447](#)
- sdReadl
 - Serial Driver, [446](#)
- sdReadTimeout
 - Serial Driver, [447](#)
- sdRequestDataI
 - Serial Driver, [458](#)
- sdStart
 - Serial Driver, [455](#)
- sdstate_t
 - Serial Driver, [449](#)
- sdStop
 - Serial Driver, [456](#)
- SDU_READY
 - Serial over USB Driver, [817](#)
- sdu_start_receive
 - Serial over USB Driver, [817](#)
- SDU_STOP
 - Serial over USB Driver, [817](#)
- SDU_UNINIT
 - Serial over USB Driver, [817](#)
- sduConfigureHookI
 - Serial over USB Driver, [826](#)
- sduControl
 - Serial over USB Driver, [829](#)
- sduDataReceived
 - Serial over USB Driver, [828](#)
- sduDataTransmitted
 - Serial over USB Driver, [828](#)
- sduInit
 - Serial over USB Driver, [822](#)
- sduInterruptTransmitted
 - Serial over USB Driver, [829](#)
- sduObjectInit
 - Serial over USB Driver, [823](#)
- sduRequestsHook
 - Serial over USB Driver, [826](#)
- sduSOFHookI
 - Serial over USB Driver, [827](#)
- sduStart
 - Serial over USB Driver, [823](#)
- sdustate_t
 - Serial over USB Driver, [817](#)
- sduStop
 - Serial over USB Driver, [824](#)
- sduSuspendHookI
 - Serial over USB Driver, [825](#)
- sduWakeupHookI
 - Serial over USB Driver, [825](#)
- sdWrite
 - Serial Driver, [444](#)
- sdWritel
 - Serial Driver, [444](#)
- sdWriteTimeout
 - Serial Driver, [445](#)
- sectors
 - flash_descriptor_t, [1149](#)
- sectors_count
 - flash_descriptor_t, [1149](#)
- sectors_size
 - flash_descriptor_t, [1149](#)
- Serial Driver, [435](#)
 - _ctl, [454](#)
 - _get, [451](#)
 - _gett, [452](#)
 - _put, [451](#)
 - _putt, [452](#)
 - _read, [450](#)
 - _readt, [453](#)
 - _serial_driver_data, [449](#)
 - _serial_driver_methods, [440](#)
 - _write, [450](#)
 - _writel, [453](#)
- default_config, [462](#)
- PLATFORM_SERIAL_USE_USART1, [448](#)
- SD1, [462](#)
- SD_BREAK_DETECTED, [439](#)
- SD_FRAMING_ERROR, [439](#)
- sd_llid_init, [461](#)
- sd_llid_start, [461](#)
- sd_llid_stop, [462](#)
- SD_NOISE_ERROR, [439](#)
- SD_OVERRUN_ERROR, [439](#)
- SD_PARITY_ERROR, [439](#)
- SD_QUEUE_FULL_ERROR, [439](#)
- SD_READY, [450](#)
- SD_STOP, [450](#)
- SD_UNINIT, [450](#)
- sdAsynchronousRead, [448](#)
- sdAsynchronousWrite, [446](#)

- sdControl, [460](#)
- sdGet, [443](#)
- sdGetl, [442](#)
- sdGetTimeout, [443](#)
- sdGetWouldBlock, [459](#)
- sdIncomingDataI, [457](#)
- sdInit, [454](#)
- sdObjectInit, [454](#)
- sdPut, [440](#)
- sdPutl, [440](#)
- sdPutTimeout, [441](#)
- sdPutWouldBlock, [458](#)
- sdRead, [447](#)
- sdReadl, [446](#)
- sdReadTimeout, [447](#)
- sdRequestDataI, [458](#)
- sdStart, [455](#)
- sdstate_t, [449](#)
- sdStop, [456](#)
- sdWrite, [444](#)
- sdWritel, [444](#)
- sdWriteTimeout, [445](#)
- SERIAL_BUFFERS_SIZE, [439](#)
- SERIAL_DEFAULT_BITRATE, [439](#)
- SerialConfig, [449](#)
- SerialDriver, [449](#)
- vmt, [462](#)
- Serial NOR Flash Driver, [792](#)
 - _snor_flash_methods, [797](#)
 - _snor_flash_methods_alone, [796](#)
 - bus_acquire, [797](#), [800](#)
 - bus_cmd, [802](#)
 - bus_cmd_addr, [805](#)
 - bus_cmd_addr_dummy_receive, [808](#)
 - bus_cmd_addr_receive, [806](#)
 - bus_cmd_addr_send, [805](#)
 - bus_cmd_dummy_receive, [807](#)
 - bus_cmd_receive, [804](#)
 - bus_cmd_send, [803](#)
 - bus_release, [797](#), [800](#)
 - bus_stop, [801](#)
 - BUSConfig, [796](#)
 - BUSDriver, [796](#)
 - snor_acquire_exclusive, [798](#)
 - SNOR_BUFFER_SIZE, [795](#)
 - SNOR_BUS_DRIVER, [795](#)
 - SNOR_BUS_DRIVER_SPI, [795](#)
 - SNOR_BUS_DRIVER_WSPI, [795](#)
 - snor_get_descriptor, [797](#)
 - snor_nocache_buffer_t, [797](#)
 - snor_program, [798](#)
 - snor_query_erase, [798](#)
 - snor_read, [798](#)
 - snor_read_sfdp, [799](#)
 - snor_release_exclusive, [799](#)
 - SNOR_SHARED_BUS, [796](#)
 - SNOR_SPI_4BYTES_ADDRESS, [796](#)
 - snor_spi_cmd_addr, [799](#)
 - snor_start_erase_all, [798](#)
 - snor_start_erase_sector, [798](#)
 - SNOR_USE_MUTUAL_EXCLUSION, [796](#)
 - snor_verify_erase, [798](#)
 - snor_vmt, [813](#)
 - snorMemoryMap, [811](#)
 - snorMemoryUnmap, [811](#)
 - snorObjectInit, [809](#)
 - snorStart, [809](#)
 - snorStop, [810](#)
- Serial over USB Driver, [813](#)
 - _ctl, [821](#)
 - _get, [819](#)
 - _gett, [820](#)
 - _put, [818](#)
 - _putt, [819](#)
 - _read, [818](#)
 - _readt, [821](#)
 - _serial_usb_driver_data, [816](#)
 - _serial_usb_driver_methods, [816](#)
 - _write, [817](#)
 - _writet, [820](#)
 - ibnotify, [822](#)
 - linecoding, [830](#)
 - obnotify, [822](#)
 - SDU_READY, [817](#)
 - sdu_start_receive, [817](#)
 - SDU_STOP, [817](#)
 - SDU_UNINIT, [817](#)
 - sduConfigureHookI, [826](#)
 - sduControl, [829](#)
 - sduDataReceived, [828](#)
 - sduDataTransmitted, [828](#)
 - sduInit, [822](#)
 - sduInterruptTransmitted, [829](#)
 - sduObjectInit, [823](#)
 - sduRequestsHook, [826](#)
 - sduSOFHookI, [827](#)
 - sduStart, [823](#)
 - sdustate_t, [817](#)
 - sduStop, [824](#)
 - sduSuspendHookI, [825](#)
 - sduWakeupHookI, [825](#)
 - SERIAL_USB_BUFFERS_NUMBER, [816](#)
 - SERIAL_USB_BUFFERS_SIZE, [816](#)
 - SerialUSBDriver, [817](#)
 - vmt, [830](#)
- SERIAL_BUFFERS_SIZE
 - Configuration, [52](#)
 - Serial Driver, [439](#)
- SERIAL_DEFAULT_BITRATE
 - Configuration, [52](#)
 - Serial Driver, [439](#)
- SERIAL_USB_BUFFERS_NUMBER
 - Configuration, [53](#)
 - Serial over USB Driver, [816](#)
- SERIAL_USB_BUFFERS_SIZE
 - Configuration, [53](#)

- Serial over USB Driver, [816](#)
- SerialConfig
 - Serial Driver, [449](#)
- SerialDriver
 - Serial Driver, [449](#)
- SerialDriverVMT, [1282](#)
- SerialUSBConfig, [1284](#)
 - bulk_in, [1285](#)
 - bulk_out, [1285](#)
 - int_in, [1285](#)
 - usbp, [1285](#)
- SerialUSBDriver, [1285](#)
 - Serial over USB Driver, [817](#)
 - vmt, [1287](#)
- SerialUSBDriverVMT, [1288](#)
- set_address
 - USB Driver, [661](#)
- setup
 - USBDriver, [1321](#)
- setup_cb
 - USBEndpointConfig, [1324](#)
- SHA1Context, [1289](#)
 - dummy, [1290](#)
- SHA256Context, [1290](#)
 - dummy, [1290](#)
- SHA512Context, [1291](#)
 - dummy, [1291](#)
- SID
 - CANRxFrame, [1136](#)
 - CANTxFrame, [1139](#)
- SIO Driver, [463](#)
 - __ctl, [491](#)
 - __get, [489](#)
 - __gett, [490](#)
 - __put, [488](#)
 - __putt, [489](#)
 - __read, [488](#)
 - __readt, [491](#)
 - __sio_callback, [479](#)
 - __sio_reloc_field, [482](#)
 - __sio_wakeup_errors, [480](#)
 - __sio_wakeup_rx, [480](#)
 - __sio_wakeup_rxidle, [480](#)
 - __sio_wakeup_tx, [481](#)
 - __sio_wakeup_txend, [481](#)
 - __write, [487](#)
 - __writet, [490](#)
 - _sio_driver_methods, [471](#)
- PLATFORM_SIO_USE_SIO1, [482](#)
- SIO_DEFAULT_BITRATE, [470](#)
- SIO_EV_ALL_DATA, [469](#)
- SIO_EV_ALL_ERRORS, [470](#)
- SIO_EV_ALL_ERRORS_POS, [468](#)
- SIO_EV_ALL_EVENTS, [470](#)
- SIO_EV_FRAMING_ERR, [469](#)
- SIO_EV_FRAMING_ERR_POS, [468](#)
- SIO_EV_NOISE_ERR, [469](#)
- SIO_EV_NOISE_ERR_POS, [468](#)
- SIO_EV_NONE, [469](#)
- SIO_EV_OVERRUN_ERR, [470](#)
- SIO_EV_OVERRUN_ERR_POS, [468](#)
- SIO_EV_PARITY_ERR, [469](#)
- SIO_EV_PARITY_ERR_POS, [468](#)
- SIO_EV_RXBREAK, [470](#)
- SIO_EV_RXBREAK_POS, [469](#)
- SIO_EV_RXIDLE, [470](#)
- SIO_EV_RXIDLE_POS, [469](#)
- SIO_EV_RXNOTEMPTY, [469](#)
- SIO_EV_RXNOTEMPTY_POS, [468](#)
- SIO_EV_TXDONE, [469](#)
- SIO_EV_TXDONE_POS, [468](#)
- SIO_EV_TXNOTFULL, [469](#)
- SIO_EV_TXNOTFULL_POS, [468](#)
- sio_llc_config_fields, [482](#)
- sio_llc_control, [509](#)
- sio_llc_driver_fields, [482](#)
- sio_llc_get, [508](#)
- sio_llc_get_and_clear_errors, [506](#)
- sio_llc_get_and_clear_events, [507](#)
- sio_llc_get_events, [507](#)
- sio_llc_has_rx_errors, [484](#)
- sio_llc_init, [505](#)
- sio_llc_is_rx_empty, [483](#)
- sio_llc_is_rx_idle, [483](#)
- sio_llc_is_tx_full, [484](#)
- sio_llc_is_tx_ongoing, [485](#)
- sio_llc_put, [509](#)
- sio_llc_read, [507](#)
- sio_llc_serve_interrupt, [510](#)
- sio_llc_start, [505](#)
- sio_llc_stop, [506](#)
- sio_llc_update_enable_flags, [506](#)
- sio_llc_write, [508](#)
- SIO_MSG_ERRORS, [470](#)
- SIO_READY, [486](#)
- SIO_STOP, [486](#)
- SIO_UNINIT, [486](#)
- SIO_USE_STREAMS_INTERFACE, [471](#)
- SIO_USE_SYNCHRONIZATION, [470](#)
- sioAsyncRead, [499](#)
- sioAsyncReadX, [477](#)
- sioAsyncWrite, [500](#)
- sioAsyncWriteX, [478](#)
- siocb_t, [486](#)
- sioClearEnableFlags, [495](#)
- sioClearEnableFlagsX, [475](#)
- SIOConfig, [486](#)
- sioControlX, [479](#)
- SIOD1, [510](#)
- SIODriver, [485](#)
- sioevents_t, [485](#)
- sioGetAndClearErrors, [496](#)
- sioGetAndClearErrorsX, [475](#)
- sioGetAndClearEvents, [496](#)
- sioGetAndClearEventsX, [476](#)
- sioGetEnableFlagsX, [475](#)

- sioGetEvents, [498](#)
- sioGetEventsX, [476](#)
- sioGetX, [476](#)
- sioHasRXErrorsX, [472](#)
- sioInit, [491](#)
- sioIsRXEmptyX, [471](#)
- sioIsRXIdleX, [472](#)
- sioIsTXFullX, [473](#)
- sioIsTXOngoingX, [473](#)
- sioObjectInit, [492](#)
- sioPutX, [477](#)
- sioSetCallbackX, [471](#)
- sioSetEnableFlags, [494](#)
- sioSetEnableFlagsX, [474](#)
- sioStart, [492](#)
- sioState_t, [486](#)
- sioStop, [493](#)
- sioSynchronizeRX, [501](#)
- sioSynchronizeRXIdle, [502](#)
- sioSynchronizeTX, [503](#)
- sioSynchronizeTXEnd, [504](#)
- sioWriteEnableFlags, [494](#)
- sioWriteEnableFlagsX, [474](#)
- sync_read, [487](#)
- sync_write, [487](#)
- vmt, [510](#)
- SIO_DEFAULT_BITRATE
 - Configuration, [53](#)
 - SIO Driver, [470](#)
- sio_driver_vmt, [1291](#)
- SIO_EV_ALL_DATA
 - SIO Driver, [469](#)
- SIO_EV_ALL_ERRORS
 - SIO Driver, [470](#)
- SIO_EV_ALL_ERRORS_POS
 - SIO Driver, [468](#)
- SIO_EV_ALL_EVENTS
 - SIO Driver, [470](#)
- SIO_EV_FRAMING_ERR
 - SIO Driver, [469](#)
- SIO_EV_FRAMING_ERR_POS
 - SIO Driver, [468](#)
- SIO_EV_NOISE_ERR
 - SIO Driver, [469](#)
- SIO_EV_NOISE_ERR_POS
 - SIO Driver, [468](#)
- SIO_EV_NONE
 - SIO Driver, [469](#)
- SIO_EV_OVERRUN_ERR
 - SIO Driver, [470](#)
- SIO_EV_OVERRUN_ERR_POS
 - SIO Driver, [468](#)
- SIO_EV_PARITY_ERR
 - SIO Driver, [469](#)
- SIO_EV_PARITY_ERR_POS
 - SIO Driver, [468](#)
- SIO_EV_RXBREAK
 - SIO Driver, [470](#)
- SIO_EV_RXBREAK_POS
 - SIO Driver, [469](#)
- SIO_EV_RXIDLE
 - SIO Driver, [470](#)
- SIO_EV_RXIDLE_POS
 - SIO Driver, [469](#)
- SIO_EV_RXNOTEMPY
 - SIO Driver, [469](#)
- SIO_EV_RXNOTEMPY_POS
 - SIO Driver, [468](#)
- SIO_EV_TXDONE
 - SIO Driver, [469](#)
- SIO_EV_TXDONE_POS
 - SIO Driver, [468](#)
- SIO_EV_TXNOTFULL
 - SIO Driver, [469](#)
- SIO_EV_TXNOTFULL_POS
 - SIO Driver, [468](#)
- sio_llc_config_fields
 - SIO Driver, [482](#)
- sio_llc_control
 - SIO Driver, [509](#)
- sio_llc_driver_fields
 - SIO Driver, [482](#)
- sio_llc_get
 - SIO Driver, [508](#)
- sio_llc_get_and_clear_errors
 - SIO Driver, [506](#)
- sio_llc_get_and_clear_events
 - SIO Driver, [507](#)
- sio_llc_get_events
 - SIO Driver, [507](#)
- sio_llc_has_rx_errors
 - SIO Driver, [484](#)
- sio_llc_init
 - SIO Driver, [505](#)
- sio_llc_is_rx_empty
 - SIO Driver, [483](#)
- sio_llc_is_rx_idle
 - SIO Driver, [483](#)
- sio_llc_is_tx_full
 - SIO Driver, [484](#)
- sio_llc_is_tx_ongoing
 - SIO Driver, [485](#)
- sio_llc_put
 - SIO Driver, [509](#)
- sio_llc_read
 - SIO Driver, [507](#)
- sio_llc_serve_interrupt
 - SIO Driver, [510](#)
- sio_llc_start
 - SIO Driver, [505](#)
- sio_llc_stop
 - SIO Driver, [506](#)
- sio_llc_update_enable_flags
 - SIO Driver, [506](#)
- sio_llc_write
 - SIO Driver, [508](#)

- SIO_MSG_ERRORS
 - SIO Driver, [470](#)
- SIO_READY
 - SIO Driver, [486](#)
- SIO_STOP
 - SIO Driver, [486](#)
- SIO_UNINIT
 - SIO Driver, [486](#)
- SIO_USE_STREAMS_INTERFACE
 - SIO Driver, [471](#)
- SIO_USE_SYNCHRONIZATION
 - Configuration, [53](#)
 - SIO Driver, [470](#)
- sioAsyncRead
 - SIO Driver, [499](#)
- sioAsyncReadX
 - SIO Driver, [477](#)
- sioAsyncWrite
 - SIO Driver, [500](#)
- sioAsyncWriteX
 - SIO Driver, [478](#)
- siocb_t
 - SIO Driver, [486](#)
- sioClearEnableFlags
 - SIO Driver, [495](#)
- sioClearEnableFlagsX
 - SIO Driver, [475](#)
- SIOConfig
 - SIO Driver, [486](#)
- sioControlX
 - SIO Driver, [479](#)
- SIOD1
 - SIO Driver, [510](#)
- SIODriver
 - SIO Driver, [485](#)
- sioevents_t
 - SIO Driver, [485](#)
- sioGetAndClearErrors
 - SIO Driver, [496](#)
- sioGetAndClearErrorsX
 - SIO Driver, [475](#)
- sioGetAndClearEvents
 - SIO Driver, [496](#)
- sioGetAndClearEventsX
 - SIO Driver, [476](#)
- sioGetEnableFlagsX
 - SIO Driver, [475](#)
- sioGetEvents
 - SIO Driver, [498](#)
- sioGetEventsX
 - SIO Driver, [476](#)
- sioGetX
 - SIO Driver, [476](#)
- sioHasRXErrorsX
 - SIO Driver, [472](#)
- sioInit
 - SIO Driver, [491](#)
- siolsRXEmptyX
 - SIO Driver, [471](#)
- siolsRXIdleX
 - SIO Driver, [472](#)
- siolsTXFullX
 - SIO Driver, [473](#)
- siolsTXOngoingX
 - SIO Driver, [473](#)
- sioObjectInit
 - SIO Driver, [492](#)
- sioPutX
 - SIO Driver, [477](#)
- sioSetCallbackX
 - SIO Driver, [471](#)
- sioSetEnableFlags
 - SIO Driver, [494](#)
- sioSetEnableFlagsX
 - SIO Driver, [474](#)
- sioStart
 - SIO Driver, [492](#)
- siostate_t
 - SIO Driver, [486](#)
- sioStop
 - SIO Driver, [493](#)
- sioSynchronizeRX
 - SIO Driver, [501](#)
- sioSynchronizeRXIdle
 - SIO Driver, [502](#)
- sioSynchronizeTX
 - SIO Driver, [503](#)
- sioSynchronizeTXEnd
 - SIO Driver, [504](#)
- sioWriteEnableFlags
 - SIO Driver, [494](#)
- sioWriteEnableFlagsX
 - SIO Driver, [474](#)
- size
 - flash_descriptor_t, [1150](#)
 - flash_sector_descriptor_t, [1156](#)
 - hal_i2s_config, [1181](#)
 - hal_mac_receive_descriptor, [1186](#)
 - hal_mac_transmit_descriptor, [1187](#)
 - mfs_data_header_t, [1243](#)
 - mfs_record_descriptor_t, [1246](#)
 - mfs_transaction_op_t, [1247](#)
- slave
 - hal_spi_config, [1196](#)
- sleep_event
 - hal_can_driver, [1170](#)
- snor_acquire_exclusive
 - Serial NOR Flash Driver, [798](#)
- SNOR_BUFFER_SIZE
 - Serial NOR Flash Driver, [795](#)
- SNOR_BUS_DRIVER
 - Serial NOR Flash Driver, [795](#)
- SNOR_BUS_DRIVER_SPI
 - Serial NOR Flash Driver, [795](#)
- SNOR_BUS_DRIVER_WSPI
 - Serial NOR Flash Driver, [795](#)

- snor_get_descriptor
 - Serial NOR Flash Driver, [797](#)
- snor_nocache_buffer, [1293](#)
 - buf, [1293](#)
 - cmd, [1293](#)
- snor_nocache_buffer_t
 - Serial NOR Flash Driver, [797](#)
- snor_program
 - Serial NOR Flash Driver, [798](#)
- snor_query_erase
 - Serial NOR Flash Driver, [798](#)
- snor_read
 - Serial NOR Flash Driver, [798](#)
- snor_read_sfdp
 - Serial NOR Flash Driver, [799](#)
- snor_release_exclusive
 - Serial NOR Flash Driver, [799](#)
- SNOR_SHARED_BUS
 - Serial NOR Flash Driver, [796](#)
- SNOR_SPI_4BYTES_ADDRESS
 - Serial NOR Flash Driver, [796](#)
- snor_spi_cmd_addr
 - Serial NOR Flash Driver, [799](#)
- snor_start_erase_all
 - Serial NOR Flash Driver, [798](#)
- snor_start_erase_sector
 - Serial NOR Flash Driver, [798](#)
- SNOR_USE_MUTUAL_EXCLUSION
 - Serial NOR Flash Driver, [796](#)
- snor_verify_erase
 - Serial NOR Flash Driver, [798](#)
- snor_vmt
 - Serial NOR Flash Driver, [813](#)
- SNORConfig, [1294](#)
 - buscfg, [1294](#)
 - busp, [1294](#)
- SNORDriver, [1295](#)
 - config, [1297](#)
 - mutex, [1297](#)
 - nocache, [1297](#)
 - vmt, [1297](#)
- SNORDriverVMT, [1297](#)
- snorMemoryMap
 - Serial NOR Flash Driver, [811](#)
- snorMemoryUnmap
 - Serial NOR Flash Driver, [811](#)
- snorObjectInit
 - Serial NOR Flash Driver, [809](#)
- snorStart
 - Serial NOR Flash Driver, [809](#)
- snorStop
 - Serial NOR Flash Driver, [810](#)
- sof_cb
 - USBConfig, [1317](#)
- spec_vers
 - unpacked_mmc_csd_t, [1303](#)
- speed
 - hal_serial_config, [1188](#)
- spi
 - xsnor_bus_configs, [1335](#)
- SPI Driver Selector, [511](#)
- SPI_ACTIVE
 - SPIv1 Driver, [524](#)
 - SPIv2 Driver, [555](#)
- SPI_COMPLETE
 - SPIv1 Driver, [524](#)
 - SPIv2 Driver, [555](#)
- spi_llc_abort
 - SPIv1 Driver, [542](#)
- spi_llc_config_fields
 - SPIv1 Driver, [523](#)
 - SPIv2 Driver, [554](#)
- spi_llc_driver_fields
 - SPIv1 Driver, [523](#)
 - SPIv2 Driver, [554](#)
- spi_llc_exchange
 - SPIv1 Driver, [540](#)
 - SPIv2 Driver, [577](#)
- spi_llc_ignore
 - SPIv1 Driver, [540](#)
 - SPIv2 Driver, [577](#)
- spi_llc_init
 - SPIv1 Driver, [538](#)
 - SPIv2 Driver, [575](#)
- spi_llc_polled_exchange
 - SPIv1 Driver, [542](#)
 - SPIv2 Driver, [579](#)
- spi_llc_receive
 - SPIv1 Driver, [541](#)
 - SPIv2 Driver, [578](#)
- spi_llc_select
 - SPIv1 Driver, [539](#)
 - SPIv2 Driver, [576](#)
- spi_llc_send
 - SPIv1 Driver, [541](#)
 - SPIv2 Driver, [578](#)
- spi_llc_start
 - SPIv1 Driver, [538](#)
 - SPIv2 Driver, [575](#)
- spi_llc_stop
 - SPIv1 Driver, [539](#)
 - SPIv2 Driver, [576](#)
- spi_llc_stop_transfer
 - SPIv2 Driver, [579](#)
- spi_llc_unselect
 - SPIv1 Driver, [539](#)
 - SPIv2 Driver, [576](#)
- SPI_READY
 - SPIv1 Driver, [524](#)
 - SPIv2 Driver, [555](#)
- SPI_SELECT_MODE
 - Configuration, [54](#)
 - SPIv1 Driver, [516](#)
 - SPIv2 Driver, [548](#)
- SPI_SELECT_MODE_LINE
 - SPIv1 Driver, [515](#)

- SPIv2 Driver, [547](#)
- SPI_SELECT_MODE_LLD
 - SPIv1 Driver, [515](#)
 - SPIv2 Driver, [548](#)
- SPI_SELECT_MODE_NONE
 - SPIv1 Driver, [515](#)
 - SPIv2 Driver, [547](#)
- SPI_SELECT_MODE_PAD
 - SPIv1 Driver, [515](#)
 - SPIv2 Driver, [547](#)
- SPI_SELECT_MODE_PORT
 - SPIv1 Driver, [515](#)
 - SPIv2 Driver, [547](#)
- SPI_STOP
 - SPIv1 Driver, [524](#)
 - SPIv2 Driver, [555](#)
- SPI_SUPPORTS_CIRCULAR
 - SPIv1 Driver, [523](#)
 - SPIv2 Driver, [554](#)
- SPI_SUPPORTS_SLAVE_MODE
 - SPIv2 Driver, [554](#)
- SPI_UNINIT
 - SPIv1 Driver, [524](#)
 - SPIv2 Driver, [555](#)
- SPI_USE_ASSERT_ON_ERROR
 - Configuration, [54](#)
 - SPIv2 Driver, [548](#)
- SPI_USE_MUTUAL_EXCLUSION
 - Configuration, [54](#)
 - SPIv1 Driver, [516](#)
 - SPIv2 Driver, [548](#)
- SPI_USE_SYNCHRONIZATION
 - SPIv2 Driver, [548](#)
- SPI_USE_WAIT
 - Configuration, [53](#)
 - SPIv1 Driver, [515](#)
- spiAbort
 - SPIv1 Driver, [532](#)
 - SPIv2 Driver, [551](#)
- spiAbortI
 - SPIv1 Driver, [532](#)
 - SPIv2 Driver, [550](#)
- spiAcquireBus
 - SPIv1 Driver, [537](#)
 - SPIv2 Driver, [574](#)
- spibuf
 - xsnor_buffers, [1333](#)
- spicallback_t
 - SPIv1 Driver, [524](#)
- spicb_t
 - SPIv2 Driver, [555](#)
- SPIConfig
 - SPIv1 Driver, [524](#)
 - SPIv2 Driver, [555](#)
- SPID1
 - SPIv1 Driver, [543](#)
 - SPIv2 Driver, [580](#)
- SPIDriver
 - SPIv1 Driver, [524](#)
 - SPIv2 Driver, [555](#)
- spiExchange
 - SPIv1 Driver, [534](#)
 - SPIv2 Driver, [571](#)
- spilgnore
 - SPIv1 Driver, [533](#)
 - SPIv2 Driver, [570](#)
- spilnit
 - SPIv1 Driver, [525](#)
 - SPIv2 Driver, [556](#)
- spilsBufferComplete
 - SPIv1 Driver, [516](#)
 - SPIv2 Driver, [548](#)
- spiObjectInit
 - SPIv1 Driver, [525](#)
 - SPIv2 Driver, [556](#)
- spip
 - mmc_spi_config_t, [1255](#)
- spiPolledExchange
 - SPIv1 Driver, [520](#)
 - SPIv2 Driver, [550](#)
- spiReceive
 - SPIv1 Driver, [536](#)
 - SPIv2 Driver, [573](#)
- spiReleaseBus
 - SPIv1 Driver, [537](#)
 - SPIv2 Driver, [574](#)
- spiSelect
 - SPIv1 Driver, [527](#)
 - SPIv2 Driver, [558](#)
- spiSelectI
 - SPIv1 Driver, [517](#)
 - SPIv2 Driver, [549](#)
- spiSend
 - SPIv1 Driver, [535](#)
 - SPIv2 Driver, [572](#)
- spiStart
 - SPIv1 Driver, [525](#)
 - SPIv2 Driver, [556](#)
- spiStartExchange
 - SPIv1 Driver, [529](#)
 - SPIv2 Driver, [562](#)
- spiStartExchangel
 - SPIv1 Driver, [518](#)
 - SPIv2 Driver, [561](#)
- spiStartIgnore
 - SPIv1 Driver, [528](#)
 - SPIv2 Driver, [560](#)
- spiStartIgnoreI
 - SPIv1 Driver, [517](#)
 - SPIv2 Driver, [559](#)
- spiStartReceive
 - SPIv1 Driver, [531](#)
 - SPIv2 Driver, [566](#)
- spiStartReceiveI
 - SPIv1 Driver, [519](#)
 - SPIv2 Driver, [565](#)

- spiStartSend
 - SPIv1 Driver, [530](#)
 - SPIv2 Driver, [564](#)
- spiStartSendI
 - SPIv1 Driver, [519](#)
 - SPIv2 Driver, [563](#)
- spistate_t
 - SPIv1 Driver, [524](#)
 - SPIv2 Driver, [555](#)
- spiStop
 - SPIv1 Driver, [526](#)
 - SPIv2 Driver, [557](#)
- spiStopTransfer
 - SPIv2 Driver, [567](#)
- spiStopTransferI
 - SPIv2 Driver, [567](#)
- spiSynchronize
 - SPIv2 Driver, [569](#)
- spiSynchronizeS
 - SPIv2 Driver, [568](#)
- spiUnselect
 - SPIv1 Driver, [527](#)
 - SPIv2 Driver, [558](#)
- spiUnselectI
 - SPIv1 Driver, [517](#)
 - SPIv2 Driver, [549](#)
- SPIv1 Driver, [511](#)
 - _spi_isr_code, [521](#)
 - _spi_isr_full_code, [522](#)
 - _spi_isr_half_code, [522](#)
 - _spi_wakeup_isr, [521](#)
 - PLATFORM_SPI_USE_SPI1, [523](#)
 - SPI_ACTIVE, [524](#)
 - SPI_COMPLETE, [524](#)
 - spi_llc_abort, [542](#)
 - spi_llc_config_fields, [523](#)
 - spi_llc_driver_fields, [523](#)
 - spi_llc_exchange, [540](#)
 - spi_llc_ignore, [540](#)
 - spi_llc_init, [538](#)
 - spi_llc_polled_exchange, [542](#)
 - spi_llc_receive, [541](#)
 - spi_llc_select, [539](#)
 - spi_llc_send, [541](#)
 - spi_llc_start, [538](#)
 - spi_llc_stop, [539](#)
 - spi_llc_unselect, [539](#)
 - SPI_READY, [524](#)
 - SPI_SELECT_MODE, [516](#)
 - SPI_SELECT_MODE_LINE, [515](#)
 - SPI_SELECT_MODE_LLD, [515](#)
 - SPI_SELECT_MODE_NONE, [515](#)
 - SPI_SELECT_MODE_PAD, [515](#)
 - SPI_SELECT_MODE_PORT, [515](#)
 - SPI_STOP, [524](#)
 - SPI_SUPPORTS_CIRCULAR, [523](#)
 - SPI_UNINIT, [524](#)
 - SPI_USE_MUTUAL_EXCLUSION, [516](#)
 - SPI_USE_WAIT, [515](#)
 - spiAbort, [532](#)
 - spiAbortI, [532](#)
 - spiAcquireBus, [537](#)
 - spicallback_t, [524](#)
 - SPIConfig, [524](#)
 - SPID1, [543](#)
 - SPIDriver, [524](#)
 - spiExchange, [534](#)
 - spiIgnore, [533](#)
 - spiInit, [525](#)
 - spisBufferComplete, [516](#)
 - spiObjectInit, [525](#)
 - spiPolledExchange, [520](#)
 - spiReceive, [536](#)
 - spiReleaseBus, [537](#)
 - spiSelect, [527](#)
 - spiSelectI, [517](#)
 - spiSend, [535](#)
 - spiStart, [525](#)
 - spiStartExchange, [529](#)
 - spiStartExchangeI, [518](#)
 - spiStartIgnore, [528](#)
 - spiStartIgnoreI, [517](#)
 - spiStartReceive, [531](#)
 - spiStartReceiveI, [519](#)
 - spiStartSend, [530](#)
 - spiStartSendI, [519](#)
 - spistate_t, [524](#)
 - spiStop, [526](#)
 - spiUnselect, [527](#)
 - spiUnselectI, [517](#)
- SPIv2 Driver, [543](#)
 - __spi_isr_complete_code, [551](#)
 - __spi_isr_error_code, [553](#)
 - __spi_isr_full_code, [553](#)
 - __spi_isr_half_code, [552](#)
 - __spi_wakeup_isr, [551](#)
 - PLATFORM_SPI_USE_SPI1, [554](#)
 - SPI_ACTIVE, [555](#)
 - SPI_COMPLETE, [555](#)
 - spi_llc_config_fields, [554](#)
 - spi_llc_driver_fields, [554](#)
 - spi_llc_exchange, [577](#)
 - spi_llc_ignore, [577](#)
 - spi_llc_init, [575](#)
 - spi_llc_polled_exchange, [579](#)
 - spi_llc_receive, [578](#)
 - spi_llc_select, [576](#)
 - spi_llc_send, [578](#)
 - spi_llc_start, [575](#)
 - spi_llc_stop, [576](#)
 - spi_llc_stop_transfer, [579](#)
 - spi_llc_unselect, [576](#)
 - SPI_READY, [555](#)
 - SPI_SELECT_MODE, [548](#)
 - SPI_SELECT_MODE_LINE, [547](#)
 - SPI_SELECT_MODE_LLD, [548](#)

- SPI_SELECT_MODE_NONE, [547](#)
- SPI_SELECT_MODE_PAD, [547](#)
- SPI_SELECT_MODE_PORT, [547](#)
- SPI_STOP, [555](#)
- SPI_SUPPORTS_CIRCULAR, [554](#)
- SPI_SUPPORTS_SLAVE_MODE, [554](#)
- SPI_UNINIT, [555](#)
- SPI_USE_ASSERT_ON_ERROR, [548](#)
- SPI_USE_MUTUAL_EXCLUSION, [548](#)
- SPI_USE_SYNCHRONIZATION, [548](#)
- spiAbort, [551](#)
- spiAbortI, [550](#)
- spiAcquireBus, [574](#)
- spicb_t, [555](#)
- SPIConfig, [555](#)
- SPID1, [580](#)
- SPIDriver, [555](#)
- spiExchange, [571](#)
- spiIgnore, [570](#)
- spiInit, [556](#)
- spisBufferComplete, [548](#)
- spiObjectInit, [556](#)
- spiPolledExchange, [550](#)
- spiReceive, [573](#)
- spiReleaseBus, [574](#)
- spiSelect, [558](#)
- spiSelectI, [549](#)
- spiSend, [572](#)
- spiStart, [556](#)
- spiStartExchange, [562](#)
- spiStartExchangel, [561](#)
- spiStartIgnore, [560](#)
- spiStartIgnoreI, [559](#)
- spiStartReceive, [566](#)
- spiStartReceiveI, [565](#)
- spiStartSend, [564](#)
- spiStartSendI, [563](#)
- spistate_t, [555](#)
- spiStop, [557](#)
- spiStopTransfer, [567](#)
- spiStopTransferI, [567](#)
- spiSynchronize, [569](#)
- spiSynchronizeS, [568](#)
- spiUnselect, [558](#)
- spiUnselectI, [549](#)
- ssline
 - hal_spi_config, [1195](#)
- ssmask
 - hal_spi_config, [1195](#)
- sspad
 - hal_spi_config, [1195](#)
- ssport
 - hal_spi_config, [1195](#)
- ST Driver, [580](#)
 - st_callback_t, [582](#)
 - st_callbacks, [592](#)
 - st_llc_get_alarm, [591](#)
 - st_llc_get_counter, [590](#)
 - st_llc_init, [590](#)
 - st_llc_is_alarm_active, [591](#)
 - ST_LLC_NUM_ALARMS, [582](#)
 - st_llc_set_alarm, [591](#)
 - st_llc_start_alarm, [590](#)
 - st_llc_stop_alarm, [591](#)
 - stBind, [582](#)
 - stBindAlarmN, [586](#)
 - stGetAlarm, [585](#)
 - stGetAlarmN, [589](#)
 - stGetCounter, [582](#)
 - stInit, [582](#)
 - stIsAlarmActive, [585](#)
 - stIsAlarmActiveN, [589](#)
 - stSetAlarm, [584](#)
 - stSetAlarmN, [588](#)
 - stSetCallback, [586](#)
 - stStartAlarm, [583](#)
 - stStartAlarmN, [587](#)
 - stStopAlarm, [584](#)
 - stStopAlarmN, [588](#)
- st_callback_t
 - ST Driver, [582](#)
- st_callbacks
 - ST Driver, [592](#)
- st_llc_get_alarm
 - ST Driver, [591](#)
- st_llc_get_counter
 - ST Driver, [590](#)
- st_llc_init
 - ST Driver, [590](#)
- st_llc_is_alarm_active
 - ST Driver, [591](#)
- ST_LLC_NUM_ALARMS
 - ST Driver, [582](#)
- st_llc_set_alarm
 - ST Driver, [591](#)
- st_llc_start_alarm
 - ST Driver, [590](#)
- st_llc_stop_alarm
 - ST Driver, [591](#)
- start_erase_all
 - flash_interface_vmt, [1154](#)
 - hal_xsnor_base_vmt, [1211](#)
 - hal_xsnor_macronix_mx25_vmt, [1217](#)
 - hal_xsnor_micron_n25q_vmt, [1223](#)
- start_erase_sector
 - flash_interface_vmt, [1154](#)
 - hal_xsnor_base_vmt, [1211](#)
 - hal_xsnor_macronix_mx25_vmt, [1217](#)
 - hal_xsnor_micron_n25q_vmt, [1223](#)
- state
 - CRYDriver, [1142](#)
 - GPTDriver, [1158](#)
 - hal_adc_driver, [1162](#)
 - hal_can_driver, [1169](#)
 - hal_dac_driver, [1174](#)
 - hal_i2c_driver, [1179](#)

- hal_i2s_driver, [1182](#)
- hal_mac_driver, [1184](#)
- hal_sio_driver, [1192](#)
- hal_spi_driver, [1198](#)
- hal_trng_driver, [1199](#)
- hal_uart_driver, [1203](#)
- hal_wspi_driver, [1206](#)
- hal_xsnor_base, [1208](#)
- hal_xsnor_macronix_mx25, [1214](#)
- hal_xsnor_micron_n25q, [1220](#)
- ICUDriver, [1229](#)
- MFSDriver, [1251](#)
- PWMDriver, [1272](#)
- USBDriver, [1320](#)
- WDGDriver, [1330](#)
- status
 - USBDriver, [1322](#)
- stBind
 - ST Driver, [582](#)
- stBindAlarmN
 - ST Driver, [586](#)
- stGetAlarm
 - ST Driver, [585](#)
- stGetAlarmN
 - ST Driver, [589](#)
- stGetCounter
 - ST Driver, [582](#)
- stInit
 - ST Driver, [582](#)
- stIsAlarmActive
 - ST Driver, [585](#)
- stIsAlarmActiveN
 - ST Driver, [589](#)
- STM_OK
 - Abstract Streams Interface, [926](#)
- STM_RESET
 - Abstract Streams Interface, [926](#)
- STM_TIMEOUT
 - Abstract Streams Interface, [926](#)
- streamGet
 - Abstract Streams Interface, [929](#)
- streamPut
 - Abstract Streams Interface, [928](#)
- streamRead
 - Abstract Streams Interface, [927](#)
- Streams, [925](#)
- streamUnget
 - Abstract Streams Interface, [930](#)
- streamWrite
 - Abstract Streams Interface, [927](#)
- stSetAlarm
 - ST Driver, [584](#)
- stSetAlarmN
 - ST Driver, [588](#)
- stSetCallback
 - ST Driver, [586](#)
- stStartAlarm
 - ST Driver, [583](#)
- stStartAlarmN
 - ST Driver, [587](#)
- stStopAlarm
 - ST Driver, [584](#)
- stStopAlarmN
 - ST Driver, [588](#)
- Support Code, [1053](#)
- suspended
 - io_buffers_queue, [1231](#)
- sym_to_val
 - Input Formatter Utility, [974](#)
- sync_read
 - SIO Driver, [487](#)
- sync_rx
 - hal_sio_driver, [1193](#)
- sync_rxdle
 - hal_sio_driver, [1193](#)
- sync_transfer
 - hal_spi_driver, [1198](#)
- sync_tx
 - hal_sio_driver, [1193](#)
- sync_txend
 - hal_sio_driver, [1193](#)
- sync_write
 - SIO Driver, [487](#)
- sysinterval_t
 - OSAL, [1090](#)
- syssts_t
 - OSAL, [1089](#)
- system_time_t
 - OSAL, [1089](#)
- taac
 - unpacked_mmc_csd_t, [1303](#)
 - unpacked_sdc_csd_10_t, [1309](#)
 - unpacked_sdc_csd_20_t, [1313](#)
- tdqueue
 - hal_mac_driver, [1184](#)
- thread
 - hal_adc_driver, [1162](#)
 - hal_dac_driver, [1174](#)
 - hal_spi_driver, [1198](#)
 - hal_wspi_driver, [1206](#)
 - USBInEndpointState, [1327](#)
 - USBOutEndpointState, [1328](#)
- thread_reference_t
 - OSAL, [1090](#)
- threadrx
 - hal_uart_driver, [1203](#)
- threads
 - palevent_t, [1267](#)
- threads_queue_t, [1298](#)
- tr, [1299](#)
- threadtx
 - hal_uart_driver, [1203](#)
- TIME
 - CANRxFrame, [1135](#)
- time_conv_t
 - OSAL, [1090](#)

- TIME_IMMEDIATE
 - OSAL, [1079](#)
- TIME_INFINITE
 - OSAL, [1079](#)
- tmp_write_protect
 - unpacked_mmc_csd_t, [1306](#)
 - unpacked_sdc_csd_10_t, [1312](#)
 - unpacked_sdc_csd_20_t, [1315](#)
- top
 - io_buffers_queue, [1232](#)
- tr
 - threads_queue_t, [1299](#)
- tr_limit_offset
 - MFSDriver, [1252](#)
- tr_next_offset
 - MFSDriver, [1252](#)
- tr_nops
 - MFSDriver, [1252](#)
- tr_ops
 - MFSDriver, [1253](#)
- tran_speed
 - unpacked_mmc_csd_t, [1303](#)
 - unpacked_sdc_csd_10_t, [1309](#)
 - unpacked_sdc_csd_20_t, [1313](#)
- transmitting
 - USBDriver, [1320](#)
- TRNG Driver, [592](#)
 - PLATFORM_TRNG_USE_TRNG1, [594](#)
 - trng_ild_config_fields, [594](#)
 - trng_ild_driver_fields, [594](#)
 - trng_ild_generate, [600](#)
 - trng_ild_init, [599](#)
 - trng_ild_start, [599](#)
 - trng_ild_stop, [599](#)
 - TRNG_READY, [595](#)
 - TRNG_RUNNING, [595](#)
 - TRNG_STOP, [595](#)
 - TRNG_UNINIT, [595](#)
 - TRNGConfig, [594](#)
 - TRNGD1, [601](#)
 - TRNGDriver, [594](#)
 - trngGenerate, [598](#)
 - trngInit, [595](#)
 - trngObjectInit, [595](#)
 - trngStart, [595](#)
 - trngstate_t, [594](#)
 - trngStop, [597](#)
- trng_ild_config_fields
 - TRNG Driver, [594](#)
- trng_ild_driver_fields
 - TRNG Driver, [594](#)
- trng_ild_generate
 - TRNG Driver, [600](#)
- trng_ild_init
 - TRNG Driver, [599](#)
- trng_ild_start
 - TRNG Driver, [599](#)
- trng_ild_stop
 - TRNG Driver, [599](#)
- TRNG_READY
 - TRNG Driver, [595](#)
- TRNG_RUNNING
 - TRNG Driver, [595](#)
- TRNG_STOP
 - TRNG Driver, [595](#)
- TRNG_UNINIT
 - TRNG Driver, [595](#)
- TRNGConfig
 - TRNG Driver, [594](#)
- TRNGD1
 - TRNG Driver, [601](#)
- TRNGDriver
 - TRNG Driver, [594](#)
- trngGenerate
 - TRNG Driver, [598](#)
- trngInit
 - TRNG Driver, [595](#)
- trngObjectInit
 - TRNG Driver, [595](#)
- trngStart
 - TRNG Driver, [595](#)
- trngstate_t
 - TRNG Driver, [594](#)
- trngStop
 - TRNG Driver, [597](#)
- TRUE
 - OSAL, [1079](#)
- tx_buffer
 - hal_i2s_config, [1180](#)
- txbuf
 - USBInEndpointState, [1326](#)
- txcnt
 - USBInEndpointState, [1326](#)
- txempty_event
 - hal_can_driver, [1170](#)
- txend1_cb
 - hal_uart_config, [1201](#)
- txend2_cb
 - hal_uart_config, [1201](#)
- txqueue
 - hal_can_driver, [1169](#)
- txsize
 - USBInEndpointState, [1326](#)
- txstate
 - hal_uart_driver, [1203](#)
- UART Driver, [601](#)
 - _uart_rx_char_match_isr_code, [613](#)
 - _uart_rx_complete_isr_code, [611](#)
 - _uart_rx_error_isr_code, [611](#)
 - _uart_rx_idle_code, [612](#)
 - _uart_timeout_isr_code, [613](#)
 - _uart_tx1_isr_code, [609](#)
 - _uart_tx2_isr_code, [610](#)
 - _uart_wakeup_rx_cm_isr, [609](#)
 - _uart_wakeup_rx_complete_isr, [608](#)
 - _uart_wakeup_rx_error_isr, [608](#)

- [_uart_wakeup_rx_timeout_isr, 609](#)
- [_uart_wakeup_tx1_isr, 607](#)
- [_uart_wakeup_tx2_isr, 607](#)
- [PLATFORM_UART_USE_UART1, 614](#)
- [UART_BREAK_DETECTED, 606](#)
- [UART_ERR_NOT_ACTIVE, 607](#)
- [UART_FRAMING_ERROR, 606](#)
- [uart_lld_init, 629](#)
- [uart_lld_start, 630](#)
- [uart_lld_start_receive, 631](#)
- [uart_lld_start_send, 630](#)
- [uart_lld_stop, 630](#)
- [uart_lld_stop_receive, 632](#)
- [uart_lld_stop_send, 631](#)
- [UART_NO_ERROR, 606](#)
- [UART_NOISE_ERROR, 606](#)
- [UART_OVERRUN_ERROR, 606](#)
- [UART_PARITY_ERROR, 606](#)
- [UART_READY, 616](#)
- [UART_RX_ACTIVE, 616](#)
- [UART_RX_COMPLETE, 616](#)
- [UART_RX_IDLE, 616](#)
- [UART_STOP, 616](#)
- [UART_TX_ACTIVE, 616](#)
- [UART_TX_COMPLETE, 616](#)
- [UART_TX_IDLE, 616](#)
- [UART_UNINIT, 616](#)
- [UART_USE_MUTUAL_EXCLUSION, 607](#)
- [UART_USE_WAIT, 607](#)
- [uartAcquireBus, 628](#)
- [uartcb_t, 615](#)
- [uartccb_t, 615](#)
- [UARTConfig, 615](#)
- [UARTD1, 632](#)
- [UARTDriver, 614](#)
- [uartecb_t, 615](#)
- [uartflags_t, 614](#)
- [uartInit, 616](#)
- [uartObjectInit, 617](#)
- [uartReceiveTimeout, 627](#)
- [uartReleaseBus, 629](#)
- [uartrxstate_t, 616](#)
- [uartSendFullTimeout, 626](#)
- [uartSendTimeout, 625](#)
- [uartStart, 617](#)
- [uartStartReceive, 622](#)
- [uartStartReceiveI, 623](#)
- [uartStartSend, 619](#)
- [uartStartSendI, 620](#)
- [uartstate_t, 615](#)
- [uartStop, 618](#)
- [uartStopReceive, 623](#)
- [uartStopReceiveI, 624](#)
- [uartStopSend, 620](#)
- [uartStopSendI, 621](#)
- [uarttxstate_t, 616](#)
- [UART_BREAK_DETECTED](#)
- [UART Driver, 606](#)
- [UART_ERR_NOT_ACTIVE](#)
- [UART Driver, 607](#)
- [UART_FRAMING_ERROR](#)
- [UART Driver, 606](#)
- [uart_lld_init](#)
- [UART Driver, 629](#)
- [uart_lld_start](#)
- [UART Driver, 630](#)
- [uart_lld_start_receive](#)
- [UART Driver, 631](#)
- [uart_lld_start_send](#)
- [UART Driver, 630](#)
- [uart_lld_stop](#)
- [UART Driver, 630](#)
- [uart_lld_stop_receive](#)
- [UART Driver, 632](#)
- [uart_lld_stop_send](#)
- [UART Driver, 631](#)
- [UART_NO_ERROR](#)
- [UART Driver, 606](#)
- [UART_NOISE_ERROR](#)
- [UART Driver, 606](#)
- [UART_OVERRUN_ERROR](#)
- [UART Driver, 606](#)
- [UART_PARITY_ERROR](#)
- [UART Driver, 606](#)
- [UART_READY](#)
- [UART Driver, 616](#)
- [UART_RX_ACTIVE](#)
- [UART Driver, 616](#)
- [UART_RX_COMPLETE](#)
- [UART Driver, 616](#)
- [UART_RX_IDLE](#)
- [UART Driver, 616](#)
- [UART_STOP](#)
- [UART Driver, 616](#)
- [UART_TX_ACTIVE](#)
- [UART Driver, 616](#)
- [UART_TX_COMPLETE](#)
- [UART Driver, 616](#)
- [UART_TX_IDLE](#)
- [UART Driver, 616](#)
- [UART_UNINIT](#)
- [UART Driver, 616](#)
- [UART_USE_MUTUAL_EXCLUSION](#)
- [Configuration, 54](#)
- [UART Driver, 607](#)
- [UART_USE_WAIT](#)
- [Configuration, 54](#)
- [UART Driver, 607](#)
- [uartAcquireBus](#)
- [UART Driver, 628](#)
- [uartcb_t](#)
- [UART Driver, 615](#)
- [uartccb_t](#)
- [UART Driver, 615](#)
- [UARTConfig](#)
- [UART Driver, 615](#)

- UARTD1
 - UART Driver, [632](#)
- UARTDriver
 - UART Driver, [614](#)
- uartecb_t
 - UART Driver, [615](#)
- uartflags_t
 - UART Driver, [614](#)
- uartInit
 - UART Driver, [616](#)
- uartObjectInit
 - UART Driver, [617](#)
- uartReceiveTimeout
 - UART Driver, [627](#)
- uartReleaseBus
 - UART Driver, [629](#)
- uartrxstate_t
 - UART Driver, [616](#)
- uartSendFullTimeout
 - UART Driver, [626](#)
- uartSendTimeout
 - UART Driver, [625](#)
- uartStart
 - UART Driver, [617](#)
- uartStartReceive
 - UART Driver, [622](#)
- uartStartReceiveI
 - UART Driver, [623](#)
- uartStartSend
 - UART Driver, [619](#)
- uartStartSendI
 - UART Driver, [620](#)
- uartstate_t
 - UART Driver, [615](#)
- uartStop
 - UART Driver, [618](#)
- uartStopReceive
 - UART Driver, [623](#)
- uartStopReceiveI
 - UART Driver, [624](#)
- uartStopSend
 - UART Driver, [620](#)
- uartStopSendI
 - UART Driver, [621](#)
- uarttxstate_t
 - UART Driver, [616](#)
- ud_size
 - USBDescriptor, [1318](#)
- ud_string
 - USBDescriptor, [1318](#)
- unget
 - Null Streams Class, [956](#)
- unpacked_mmc_cid_t, [1299](#)
 - crc, [1301](#)
 - mdt_m, [1301](#)
 - mdt_y, [1301](#)
 - mid, [1300](#)
 - oid, [1300](#)
 - pnm, [1300](#)
 - prv_m, [1301](#)
 - prv_n, [1301](#)
 - psn, [1301](#)
- unpacked_mmc_csd_t, [1301](#)
 - c_size, [1304](#)
 - c_size_mult, [1304](#)
 - ccc, [1303](#)
 - content_prot_app, [1305](#)
 - copy, [1305](#)
 - crc, [1306](#)
 - csd_structure, [1303](#)
 - default_ecc, [1305](#)
 - dsr_imp, [1304](#)
 - ecc, [1306](#)
 - erase_grp_mult, [1305](#)
 - erase_grp_size, [1304](#)
 - file_format, [1306](#)
 - file_format_grp, [1305](#)
 - nsac, [1303](#)
 - perm_write_protect, [1306](#)
 - r2w_factor, [1305](#)
 - read_bl_len, [1303](#)
 - read_bl_partial, [1303](#)
 - read_blk_misalign, [1304](#)
 - spec_vers, [1303](#)
 - taac, [1303](#)
 - tmp_write_protect, [1306](#)
 - tran_speed, [1303](#)
 - vdd_r_curr_max, [1304](#)
 - vdd_r_curr_min, [1304](#)
 - vdd_w_curr_max, [1304](#)
 - vdd_w_curr_min, [1304](#)
 - wp_grp_enable, [1305](#)
 - wp_grp_size, [1305](#)
 - write_bl_len, [1305](#)
 - write_bl_partial, [1305](#)
 - write_blk_misalign, [1304](#)
- unpacked_sdc_cid_t, [1306](#)
 - crc, [1308](#)
 - mdt_m, [1307](#)
 - mdt_y, [1308](#)
 - mid, [1307](#)
 - oid, [1307](#)
 - pnm, [1307](#)
 - prv_m, [1307](#)
 - prv_n, [1307](#)
 - psn, [1307](#)
- unpacked_sdc_csd_10_t, [1308](#)
 - c_size, [1310](#)
 - c_size_mult, [1311](#)
 - ccc, [1310](#)
 - copy, [1312](#)
 - crc, [1312](#)
 - csd_structure, [1309](#)
 - dsr_imp, [1310](#)
 - erase_blk_en, [1311](#)
 - erase_sector_size, [1311](#)

- file_format, 1312
- file_format_grp, 1311
- nsac, 1309
- perm_write_protect, 1312
- r2w_factor, 1311
- read_bl_len, 1310
- read_bl_partial, 1310
- read_blk_misalign, 1310
- taac, 1309
- tmp_write_protect, 1312
- tran_speed, 1309
- vdd_r_curr_max, 1310
- vdd_r_curr_min, 1310
- vdd_w_curr_max, 1311
- vdd_w_curr_min, 1310
- wp_grp_enable, 1311
- wp_grp_size, 1311
- write_bl_len, 1311
- write_bl_partial, 1311
- write_blk_misalign, 1310
- unpacked_sdc_csd_20_t, 1312
 - c_size, 1314
 - ccc, 1313
 - copy, 1315
 - crc, 1315
 - csd_structure, 1313
 - dsr_imp, 1314
 - erase_blk_en, 1314
 - erase_sector_size, 1314
 - file_format, 1315
 - file_format_grp, 1315
 - nsac, 1313
 - perm_write_protect, 1315
 - r2w_factor, 1315
 - read_bl_len, 1314
 - read_bl_partial, 1314
 - read_blk_misalign, 1314
 - taac, 1313
 - tmp_write_protect, 1315
 - tran_speed, 1313
 - wp_grp_enable, 1314
 - wp_grp_size, 1314
 - write_bl_len, 1315
 - write_bl_partial, 1315
 - write_blk_misalign, 1314
- USB CDC Header, 1068
 - CDC_ABSTRACT_CONTROL_MANAGEMENT, 1072
 - CDC_ABSTRACT_CONTROL_MODEL, 1071
 - CDC_CALL_MANAGEMENT, 1072
 - CDC_CLEAR_COMM_FEATURE, 1070
 - CDC_COMMUNICATION_INTERFACE_CLASS, 1071
 - CDC_CS_INTERFACE, 1071
 - CDC_DATA_INTERFACE_CLASS, 1071
 - CDC_GET_COMM_FEATURE, 1070
 - CDC_GET_ENCAPSULATED_RESPONSE, 1069
 - CDC_GET_LINE_CODING, 1070
 - CDC_GET_OPERATION_PARMS, 1071
 - CDC_GET_RINGER_PARMS, 1071
 - CDC_HEADER, 1072
 - CDC_PULSE_SETUP, 1070
 - CDC_RING_AUX_JACK, 1070
 - CDC_SEND_BREAK, 1071
 - CDC_SEND_ENCAPSULATED_COMMAND, 1069
 - CDC_SEND_PULSE, 1070
 - CDC_SET_AUX_LINE_STATE, 1070
 - CDC_SET_COMM_FEATURE, 1069
 - CDC_SET_CONTROL_LINE_STATE, 1071
 - CDC_SET_HOOK_STATE, 1070
 - CDC_SET_LINE_CODING, 1070
 - CDC_SET_OPERATION_PARMS, 1071
 - CDC_SET_PULSE_TIME, 1070
 - CDC_SET_RINGER_PARMS, 1071
 - CDC_UNION, 1072
 - LC_PARITY_EVEN, 1072
 - LC_PARITY_MARK, 1073
 - LC_PARITY_NONE, 1072
 - LC_PARITY_ODD, 1072
 - LC_PARITY_SPACE, 1073
 - LC_STOP_1, 1072
 - LC_STOP_1P5, 1072
 - LC_STOP_2, 1072
- USB Driver, 632
 - _usb_ep0in, 675
 - _usb_ep0out, 676
 - _usb_ep0setup, 674
 - _usb_isr_invoke_event_cb, 654
 - _usb_isr_invoke_in_cb, 655
 - _usb_isr_invoke_out_cb, 656
 - _usb_isr_invoke_setup_cb, 655
 - _usb_isr_invoke_sof_cb, 654
 - _usb_reset, 672
 - _usb_suspend, 673
 - _usb_wakeup, 674
 - active_status, 684
 - default_handler, 662
 - ep0_state, 684
 - ep0config, 685
 - EP_STATUS_ACTIVE, 660
 - EP_STATUS_DISABLED, 660
 - EP_STATUS_STALLED, 660
 - get_hword, 661
 - halted_status, 684
 - in, 684
 - out, 684
 - PLATFORM_USB_USE_USB1, 657
 - set_address, 661
 - USB_ACTIVE, 660
 - USB_DESC_BCD, 646
 - USB_DESC_BYTE, 645
 - USB_DESC_CONFIGURATION, 647
 - USB_DESC_CONFIGURATION_SIZE, 647
 - USB_DESC_DEVICE, 646
 - USB_DESC_DEVICE_SIZE, 646

USB_DESC_ENDPOINT, 648
USB_DESC_ENDPOINT_SIZE, 648
USB_DESC_INDEX, 645
USB_DESC_INTERFACE, 647
USB_DESC_INTERFACE_ASSOCIATION, 648
USB_DESC_INTERFACE_ASSOCIATION_SIZE, 648
USB_DESC_INTERFACE_SIZE, 647
USB_DESC_WORD, 646
USB_DESCRIPTOR_CONFIGURATION, 644
USB_DESCRIPTOR_DEVICE, 644
USB_DESCRIPTOR_DEVICE_QUALIFIER, 644
USB_DESCRIPTOR_ENDPOINT, 644
USB_DESCRIPTOR_INTERFACE, 644
USB_DESCRIPTOR_INTERFACE_ASSOCIATION, 644
USB_DESCRIPTOR_INTERFACE_POWER, 644
USB_DESCRIPTOR_OTHER_SPEED_CFG, 644
USB_DESCRIPTOR_STRING, 644
USB_EARLY_SET_ADDRESS, 645
USB_ENDPOINT_IN, 641
USB_ENDPOINT_OUT, 641
USB_EP0_ERROR, 661
USB_EP0_IN_SENDING_STS, 661
USB_EP0_IN_TX, 661
USB_EP0_IN_WAITING_TX0, 661
USB_EP0_OUT_RX, 661
USB_EP0_OUT_WAITING_STS, 661
USB_EP0_STATUS_STAGE, 656
USB_EP0_STATUS_STAGE_HW, 645
USB_EP0_STATUS_STAGE_SW, 645
USB_EP0_STP_WAITING, 661
USB_EP_MODE_TYPE, 649
USB_EP_MODE_TYPE_BULK, 649
USB_EP_MODE_TYPE_CTRL, 649
USB_EP_MODE_TYPE_INTR, 649
USB_EP_MODE_TYPE_ISOC, 649
USB_EVENT_ADDRESS, 661
USB_EVENT_CONFIGURED, 661
USB_EVENT_RESET, 661
USB_EVENT_STALLED, 661
USB_EVENT_SUSPEND, 661
USB_EVENT_UNCONFIGURED, 661
USB_EVENT_WAKEUP, 661
USB_FEATURE_DEVICE_REMOTE_WAKEUP, 645
USB_FEATURE_ENDPOINT_HALT, 644
USB_FEATURE_TEST_MODE, 645
USB_IN_STATE, 649
USB_LATE_SET_ADDRESS, 645
usb_lld_clear_in, 683
usb_lld_clear_out, 683
usb_lld_connect_bus, 658
usb_lld_disable_endpoints, 679
usb_lld_disconnect_bus, 658
usb_lld_get_frame_number, 657
usb_lld_get_status_in, 680
usb_lld_get_status_out, 679
usb_lld_get_transaction_size, 657
usb_lld_init, 677
usb_lld_init_endpoint, 679
usb_lld_prepare_receive, 681
usb_lld_prepare_transmit, 681
usb_lld_read_setup, 680
usb_lld_reset, 678
usb_lld_set_address, 678
usb_lld_stall_in, 683
usb_lld_stall_out, 682
usb_lld_start, 677
usb_lld_start_in, 682
usb_lld_start_out, 682
usb_lld_stop, 678
usb_lld_wakeup_host, 658
USB_MAX_ENDPOINTS, 656
USB_OUT_STATE, 650
USB_READY, 660
USB_REQ_CLEAR_FEATURE, 643
USB_REQ_GET_CONFIGURATION, 643
USB_REQ_GET_DESCRIPTOR, 643
USB_REQ_GET_INTERFACE, 643
USB_REQ_GET_STATUS, 642
USB_REQ_SET_ADDRESS, 643
USB_REQ_SET_CONFIGURATION, 643
USB_REQ_SET_DESCRIPTOR, 643
USB_REQ_SET_FEATURE, 643
USB_REQ_SET_INTERFACE, 643
USB_REQ_SYNCH_FRAME, 643
USB_RTYPE_DIR_DEV2HOST, 641
USB_RTYPE_DIR_HOST2DEV, 641
USB_RTYPE_DIR_MASK, 641
USB_RTYPE_RECIPIENT_DEVICE, 642
USB_RTYPE_RECIPIENT_ENDPOINT, 642
USB_RTYPE_RECIPIENT_INTERFACE, 642
USB_RTYPE_RECIPIENT_MASK, 642
USB_RTYPE_RECIPIENT_OTHER, 642
USB_RTYPE_TYPE_CLASS, 642
USB_RTYPE_TYPE_MASK, 641
USB_RTYPE_TYPE_RESERVED, 642
USB_RTYPE_TYPE_STD, 642
USB_RTYPE_TYPE_VENDOR, 642
USB_SELECTED, 660
USB_SET_ADDRESS_ACK_HANDLING, 657
USB_SET_ADDRESS_ACK_HW, 645
USB_SET_ADDRESS_ACK_SW, 645
USB_SET_ADDRESS_MODE, 656
USB_STOP, 660
USB_SUSPENDED, 660
USB_UNINIT, 660
USB_USE_WAIT, 650
usbcallback_t, 659
usbConnectBus, 650
USB_D1, 684
usbDisableEndpointsI, 666
usbDisconnectBus, 651
USB_Driver, 659
usbep0state_t, 660

- usbep_t, 659
- usbepcallback_t, 659
- usbepstatus_t, 660
- usbevent_t, 661
- usbeventcb_t, 659
- usbgetdescriptor_t, 660
- usbGetDriverStatel, 650
- usbGetFrameNumberX, 651
- usbGetReceiveStatusl, 652
- usbGetReceiveTransactionSizeX, 652
- usbGetTransmitStatusl, 651
- usbInit, 663
- usbInitEndpointl, 666
- usbObjectInit, 664
- usbReadSetup, 653
- usbReadSetupl, 677
- usbReceive, 668
- usbreqhandler_t, 659
- usbSetupTransfer, 653
- usbStallReceiveI, 670
- usbStallTransmitI, 671
- usbStart, 664
- usbStartReceiveI, 667
- usbStartTransmitI, 668
- usbstate_t, 660
- usbStop, 665
- usbTransmit, 669
- usbWakeupHost, 672
- zero_status, 684
- USB_ACTIVE
 - USB Driver, 660
- USB_DESC_BCD
 - USB Driver, 646
- USB_DESC_BYTE
 - USB Driver, 645
- USB_DESC_CONFIGURATION
 - USB Driver, 647
- USB_DESC_CONFIGURATION_SIZE
 - USB Driver, 647
- USB_DESC_DEVICE
 - USB Driver, 646
- USB_DESC_DEVICE_SIZE
 - USB Driver, 646
- USB_DESC_ENDPOINT
 - USB Driver, 648
- USB_DESC_ENDPOINT_SIZE
 - USB Driver, 648
- USB_DESC_INDEX
 - USB Driver, 645
- USB_DESC_INTERFACE
 - USB Driver, 647
- USB_DESC_INTERFACE_ASSOCIATION
 - USB Driver, 648
- USB_DESC_INTERFACE_ASSOCIATION_SIZE
 - USB Driver, 648
- USB_DESC_INTERFACE_SIZE
 - USB Driver, 647
- USB_DESC_WORD
 - USB Driver, 646
- USB_DESCRIPTOR_CONFIGURATION
 - USB Driver, 644
- USB_DESCRIPTOR_DEVICE
 - USB Driver, 644
- USB_DESCRIPTOR_DEVICE_QUALIFIER
 - USB Driver, 644
- USB_DESCRIPTOR_ENDPOINT
 - USB Driver, 644
- USB_DESCRIPTOR_INTERFACE
 - USB Driver, 644
- USB_DESCRIPTOR_INTERFACE_ASSOCIATION
 - USB Driver, 644
- USB_DESCRIPTOR_INTERFACE_POWER
 - USB Driver, 644
- USB_DESCRIPTOR_OTHER_SPEED_CFG
 - USB Driver, 644
- USB_DESCRIPTOR_STRING
 - USB Driver, 644
- USB_EARLY_SET_ADDRESS
 - USB Driver, 645
- USB_ENDPOINT_IN
 - USB Driver, 641
- USB_ENDPOINT_OUT
 - USB Driver, 641
- USB_EP0_ERROR
 - USB Driver, 661
- USB_EP0_IN_SENDING_STS
 - USB Driver, 661
- USB_EP0_IN_TX
 - USB Driver, 661
- USB_EP0_IN_WAITING_TX0
 - USB Driver, 661
- USB_EP0_OUT_RX
 - USB Driver, 661
- USB_EP0_OUT_WAITING_STS
 - USB Driver, 661
- USB_EP0_STATUS_STAGE
 - USB Driver, 656
- USB_EP0_STATUS_STAGE_HW
 - USB Driver, 645
- USB_EP0_STATUS_STAGE_SW
 - USB Driver, 645
- USB_EP0_STP_WAITING
 - USB Driver, 661
- USB_EP_MODE_TYPE
 - USB Driver, 649
- USB_EP_MODE_TYPE_BULK
 - USB Driver, 649
- USB_EP_MODE_TYPE_CTRL
 - USB Driver, 649
- USB_EP_MODE_TYPE_INTR
 - USB Driver, 649
- USB_EP_MODE_TYPE_ISOC
 - USB Driver, 649
- USB_EVENT_ADDRESS
 - USB Driver, 661
- USB_EVENT_CONFIGURED

- USB Driver, [661](#)
- USB_EVENT_RESET
 - USB Driver, [661](#)
- USB_EVENT_STALLED
 - USB Driver, [661](#)
- USB_EVENT_SUSPEND
 - USB Driver, [661](#)
- USB_EVENT_UNCONFIGURED
 - USB Driver, [661](#)
- USB_EVENT_WAKEUP
 - USB Driver, [661](#)
- USB_FEATURE_DEVICE_REMOTE_WAKEUP
 - USB Driver, [645](#)
- USB_FEATURE_ENDPOINT_HALT
 - USB Driver, [644](#)
- USB_FEATURE_TEST_MODE
 - USB Driver, [645](#)
- USB_IN_STATE
 - USB Driver, [649](#)
- USB_LATE_SET_ADDRESS
 - USB Driver, [645](#)
- usb_lld_clear_in
 - USB Driver, [683](#)
- usb_lld_clear_out
 - USB Driver, [683](#)
- usb_lld_connect_bus
 - USB Driver, [658](#)
- usb_lld_disable_endpoints
 - USB Driver, [679](#)
- usb_lld_disconnect_bus
 - USB Driver, [658](#)
- usb_lld_get_frame_number
 - USB Driver, [657](#)
- usb_lld_get_status_in
 - USB Driver, [680](#)
- usb_lld_get_status_out
 - USB Driver, [679](#)
- usb_lld_get_transaction_size
 - USB Driver, [657](#)
- usb_lld_init
 - USB Driver, [677](#)
- usb_lld_init_endpoint
 - USB Driver, [679](#)
- usb_lld_prepare_receive
 - USB Driver, [681](#)
- usb_lld_prepare_transmit
 - USB Driver, [681](#)
- usb_lld_read_setup
 - USB Driver, [680](#)
- usb_lld_reset
 - USB Driver, [678](#)
- usb_lld_set_address
 - USB Driver, [678](#)
- usb_lld_stall_in
 - USB Driver, [683](#)
- usb_lld_stall_out
 - USB Driver, [682](#)
- usb_lld_start
 - USB Driver, [677](#)
- usb_lld_start_in
 - USB Driver, [682](#)
- usb_lld_start_out
 - USB Driver, [682](#)
- usb_lld_stop
 - USB Driver, [678](#)
- usb_lld_wakeup_host
 - USB Driver, [658](#)
- USB_MAX_ENDPOINTS
 - USB Driver, [656](#)
- USB_OUT_STATE
 - USB Driver, [650](#)
- USB_READY
 - USB Driver, [660](#)
- USB_REQ_CLEAR_FEATURE
 - USB Driver, [643](#)
- USB_REQ_GET_CONFIGURATION
 - USB Driver, [643](#)
- USB_REQ_GET_DESCRIPTOR
 - USB Driver, [643](#)
- USB_REQ_GET_INTERFACE
 - USB Driver, [643](#)
- USB_REQ_GET_STATUS
 - USB Driver, [642](#)
- USB_REQ_SET_ADDRESS
 - USB Driver, [643](#)
- USB_REQ_SET_CONFIGURATION
 - USB Driver, [643](#)
- USB_REQ_SET_DESCRIPTOR
 - USB Driver, [643](#)
- USB_REQ_SET_FEATURE
 - USB Driver, [643](#)
- USB_REQ_SET_INTERFACE
 - USB Driver, [643](#)
- USB_REQ_SYNCH_FRAME
 - USB Driver, [643](#)
- USB_RTYPE_DIR_DEV2HOST
 - USB Driver, [641](#)
- USB_RTYPE_DIR_HOST2DEV
 - USB Driver, [641](#)
- USB_RTYPE_DIR_MASK
 - USB Driver, [641](#)
- USB_RTYPE_RECIPIENT_DEVICE
 - USB Driver, [642](#)
- USB_RTYPE_RECIPIENT_ENDPOINT
 - USB Driver, [642](#)
- USB_RTYPE_RECIPIENT_INTERFACE
 - USB Driver, [642](#)
- USB_RTYPE_RECIPIENT_MASK
 - USB Driver, [642](#)
- USB_RTYPE_RECIPIENT_OTHER
 - USB Driver, [642](#)
- USB_RTYPE_TYPE_CLASS
 - USB Driver, [642](#)
- USB_RTYPE_TYPE_MASK
 - USB Driver, [641](#)
- USB_RTYPE_TYPE_RESERVED

- USB Driver, [642](#)
- USB_RTYPE_TYPE_STD
 - USB Driver, [642](#)
- USB_RTYPE_TYPE_VENDOR
 - USB Driver, [642](#)
- USB_SELECTED
 - USB Driver, [660](#)
- USB_SET_ADDRESS_ACK_HANDLING
 - USB Driver, [657](#)
- USB_SET_ADDRESS_ACK_HW
 - USB Driver, [645](#)
- USB_SET_ADDRESS_ACK_SW
 - USB Driver, [645](#)
- USB_SET_ADDRESS_MODE
 - USB Driver, [656](#)
- USB_STOP
 - USB Driver, [660](#)
- USB_SUSPENDED
 - USB Driver, [660](#)
- USB_UNINIT
 - USB Driver, [660](#)
- USB_USE_WAIT
 - Configuration, [55](#)
 - USB Driver, [650](#)
- usbcallback_t
 - USB Driver, [659](#)
- USBConfig, [1316](#)
 - event_cb, [1317](#)
 - get_descriptor_cb, [1317](#)
 - requests_hook_cb, [1317](#)
 - sof_cb, [1317](#)
- usbConnectBus
 - USB Driver, [650](#)
- USBID1
 - USB Driver, [684](#)
- USBDescriptor, [1318](#)
 - ud_size, [1318](#)
 - ud_string, [1318](#)
- usbDisableEndpointsI
 - USB Driver, [666](#)
- usbDisconnectBus
 - USB Driver, [651](#)
- USBDriver, [1319](#)
 - address, [1322](#)
 - config, [1320](#)
 - configuration, [1322](#)
 - ep0endcb, [1321](#)
 - ep0n, [1321](#)
 - ep0next, [1321](#)
 - ep0state, [1321](#)
 - epc, [1320](#)
 - in_params, [1320](#)
 - out_params, [1321](#)
 - receiving, [1320](#)
 - saved_state, [1322](#)
 - setup, [1321](#)
 - state, [1320](#)
 - status, [1322](#)
 - transmitting, [1320](#)
 - USB Driver, [659](#)
- USBEndpointConfig, [1322](#)
 - ep_mode, [1324](#)
 - in_cb, [1324](#)
 - in_maxsize, [1325](#)
 - in_state, [1325](#)
 - out_cb, [1325](#)
 - out_maxsize, [1325](#)
 - out_state, [1325](#)
 - setup_cb, [1324](#)
- usbep0state_t
 - USB Driver, [660](#)
- usbep_t
 - USB Driver, [659](#)
- usbepcallback_t
 - USB Driver, [659](#)
- usbepstatus_t
 - USB Driver, [660](#)
- usbevent_t
 - USB Driver, [661](#)
- usbeventcb_t
 - USB Driver, [659](#)
- usbgetdescriptor_t
 - USB Driver, [660](#)
- usbGetDriverStateI
 - USB Driver, [650](#)
- usbGetFrameNumberX
 - USB Driver, [651](#)
- usbGetReceiveStatusI
 - USB Driver, [652](#)
- usbGetReceiveTransactionSizeX
 - USB Driver, [652](#)
- usbGetTransmitStatusI
 - USB Driver, [651](#)
- USBInEndpointState, [1326](#)
 - thread, [1327](#)
 - txbuf, [1326](#)
 - txcnt, [1326](#)
 - txsize, [1326](#)
- usbInit
 - USB Driver, [663](#)
- usbInitEndpointI
 - USB Driver, [666](#)
- usbObjectInit
 - USB Driver, [664](#)
- USBOutEndpointState, [1327](#)
 - rxbuf, [1328](#)
 - rxcnt, [1328](#)
 - rxsize, [1328](#)
 - thread, [1328](#)
- usbp
 - SerialUSBConfig, [1285](#)
- usbReadSetup
 - USB Driver, [653](#)
- usbReadSetupI
 - USB Driver, [677](#)
- usbReceive

- USB Driver, [668](#)
- usbreqhandler_t
 - USB Driver, [659](#)
- usbSetupTransfer
 - USB Driver, [653](#)
- usbStallReceive
 - USB Driver, [670](#)
- usbStallTransmit
 - USB Driver, [671](#)
- usbStart
 - USB Driver, [664](#)
- usbStartReceive
 - USB Driver, [667](#)
- usbStartTransmit
 - USB Driver, [668](#)
- usbstate_t
 - USB Driver, [660](#)
- usbStop
 - USB Driver, [665](#)
- usbTransmit
 - USB Driver, [669](#)
- usbWakeupHost
 - USB Driver, [672](#)
- used_space
 - MFSDriver, [1252](#)
- vdd_r_curr_max
 - unpacked_mmc_csd_t, [1304](#)
 - unpacked_sdc_csd_10_t, [1310](#)
- vdd_r_curr_min
 - unpacked_mmc_csd_t, [1304](#)
 - unpacked_sdc_csd_10_t, [1310](#)
- vdd_w_curr_max
 - unpacked_mmc_csd_t, [1304](#)
 - unpacked_sdc_csd_10_t, [1311](#)
- vdd_w_curr_min
 - unpacked_mmc_csd_t, [1304](#)
 - unpacked_sdc_csd_10_t, [1310](#)
- verify_erase
 - flash_interface_vmt, [1155](#)
 - hal_xsnor_base_vmt, [1212](#)
 - hal_xsnor_macronix_mx25_vmt, [1218](#)
 - hal_xsnor_micron_n25q_vmt, [1224](#)
- vmt
 - BaseAsynchronousChannel, [1108](#)
 - BaseBlockDevice, [1111](#)
 - BaseBufferedStream, [1115](#)
 - BaseChannel, [1118](#)
 - BaseFlash, [1121](#)
 - BaseObject, [1124](#)
 - BasePersistentStorage, [1125](#)
 - BaseSequentialStream, [1127](#)
 - Buffered SIO Driver, [737](#)
 - Buffered Streams Class, [967](#)
 - BufferedStreamAdapter, [1133](#)
 - EFL Driver, [216](#)
 - FileStream, [1147](#)
 - flash_interface, [1152](#)
 - hal_buffered_serial, [1165](#)
 - hal_buffered_siol_driver, [1167](#)
 - hal_efl_driver, [1177](#)
 - hal_serial_driver, [1190](#)
 - hal_sio_driver, [1192](#)
 - hal_xsnor_base, [1208](#)
 - hal_xsnor_macronix_mx25, [1214](#)
 - hal_xsnor_micron_n25q, [1220](#)
 - Memory Streams Class, [955](#)
 - MemoryStream, [1239](#)
 - mmc_spi_driver_t, [1257](#)
 - MMCSDBlockDevice, [1262](#)
 - Null Streams Class, [957](#)
 - NullStream, [1265](#)
 - RTCDriver, [1275](#)
 - SDCDriver, [1280](#)
 - Serial Driver, [462](#)
 - Serial over USB Driver, [830](#)
 - SerialUSBDriver, [1287](#)
 - SIO Driver, [510](#)
 - SNORDriver, [1297](#)
- waiting
 - io_buffers_queue, [1231](#)
- wakeup_event
 - hal_can_driver, [1170](#)
- WDG Driver, [685](#)
 - PLATFORM_WDG_USE_WDG1, [687](#)
 - wdg_ild_init, [690](#)
 - wdg_ild_reset, [691](#)
 - wdg_ild_start, [690](#)
 - wdg_ild_stop, [691](#)
 - WDG_READY, [687](#)
 - WDG_STOP, [687](#)
 - WDG_UNINIT, [687](#)
 - WDGD1, [691](#)
 - WDGDriver, [687](#)
 - wdgInit, [688](#)
 - wdgReset, [689](#)
 - wdgResetI, [686](#)
 - wdgStart, [688](#)
 - wdgstate_t, [687](#)
 - wdgStop, [689](#)
- wdg_ild_init
 - WDG Driver, [690](#)
- wdg_ild_reset
 - WDG Driver, [691](#)
- wdg_ild_start
 - WDG Driver, [690](#)
- wdg_ild_stop
 - WDG Driver, [691](#)
- WDG_READY
 - WDG Driver, [687](#)
- WDG_STOP
 - WDG Driver, [687](#)
- WDG_UNINIT
 - WDG Driver, [687](#)
- WDGConfig, [1328](#)
- WDGD1
 - WDG Driver, [691](#)

- WDGDriver, [1329](#)
 - config, [1330](#)
 - state, [1330](#)
 - WDG Driver, [687](#)
- wdgInit
 - WDG Driver, [688](#)
- wdgReset
 - WDG Driver, [689](#)
- wdgResetl
 - WDG Driver, [686](#)
- wdgStart
 - WDG Driver, [688](#)
- wdgstate_t
 - WDG Driver, [687](#)
- wdgStop
 - WDG Driver, [689](#)
- width_cb
 - ICUConfig, [1228](#)
- wp_grp_enable
 - unpacked_mmc_csd_t, [1305](#)
 - unpacked_sdc_csd_10_t, [1311](#)
 - unpacked_sdc_csd_20_t, [1314](#)
- wp_grp_size
 - unpacked_mmc_csd_t, [1305](#)
 - unpacked_sdc_csd_10_t, [1311](#)
 - unpacked_sdc_csd_20_t, [1314](#)
- write_bl_len
 - unpacked_mmc_csd_t, [1305](#)
 - unpacked_sdc_csd_10_t, [1311](#)
 - unpacked_sdc_csd_20_t, [1315](#)
- write_bl_partial
 - unpacked_mmc_csd_t, [1305](#)
 - unpacked_sdc_csd_10_t, [1311](#)
 - unpacked_sdc_csd_20_t, [1315](#)
- write_blk_misalign
 - unpacked_mmc_csd_t, [1304](#)
 - unpacked_sdc_csd_10_t, [1310](#)
 - unpacked_sdc_csd_20_t, [1314](#)
- writes
 - Null Streams Class, [956](#)
- wspi
 - xsnor_bus_configs, [1335](#)
- WSPI Driver, [692](#)
 - _wspi_error_code, [705](#)
 - _wspi_isr_code, [704](#)
 - _wspi_wakeup_isr, [703](#)
 - PLATFORM_WSPI_USE_WSPI1, [706](#)
 - WSPI_CFG_ADDR_DTR, [698](#)
 - WSPI_CFG_ADDR_MODE_EIGHT_LINES, [698](#)
 - WSPI_CFG_ADDR_MODE_FOUR_LINES, [698](#)
 - WSPI_CFG_ADDR_MODE_MASK, [697](#)
 - WSPI_CFG_ADDR_MODE_NONE, [697](#)
 - WSPI_CFG_ADDR_MODE_ONE_LINE, [697](#)
 - WSPI_CFG_ADDR_MODE_TWO_LINES, [697](#)
 - WSPI_CFG_ADDR_SIZE_16, [698](#)
 - WSPI_CFG_ADDR_SIZE_24, [698](#)
 - WSPI_CFG_ADDR_SIZE_32, [698](#)
 - WSPI_CFG_ADDR_SIZE_8, [698](#)
 - WSPI_CFG_ADDR_SIZE_MASK, [698](#)
 - WSPI_CFG_ALL_DTR, [700](#)
 - WSPI_CFG_ALT_DTR, [699](#)
 - WSPI_CFG_ALT_MODE_EIGHT_LINES, [699](#)
 - WSPI_CFG_ALT_MODE_FOUR_LINES, [699](#)
 - WSPI_CFG_ALT_MODE_MASK, [698](#)
 - WSPI_CFG_ALT_MODE_NONE, [698](#)
 - WSPI_CFG_ALT_MODE_ONE_LINE, [699](#)
 - WSPI_CFG_ALT_MODE_TWO_LINES, [699](#)
 - WSPI_CFG_ALT_SIZE_16, [699](#)
 - WSPI_CFG_ALT_SIZE_24, [699](#)
 - WSPI_CFG_ALT_SIZE_32, [699](#)
 - WSPI_CFG_ALT_SIZE_8, [699](#)
 - WSPI_CFG_ALT_SIZE_MASK, [699](#)
 - WSPI_CFG_CMD_DTR, [697](#)
 - WSPI_CFG_CMD_MODE_EIGHT_LINES, [696](#)
 - WSPI_CFG_CMD_MODE_FOUR_LINES, [696](#)
 - WSPI_CFG_CMD_MODE_MASK, [696](#)
 - WSPI_CFG_CMD_MODE_NONE, [696](#)
 - WSPI_CFG_CMD_MODE_ONE_LINE, [696](#)
 - WSPI_CFG_CMD_MODE_TWO_LINES, [696](#)
 - WSPI_CFG_CMD_SIZE_16, [697](#)
 - WSPI_CFG_CMD_SIZE_24, [697](#)
 - WSPI_CFG_CMD_SIZE_32, [697](#)
 - WSPI_CFG_CMD_SIZE_8, [697](#)
 - WSPI_CFG_CMD_SIZE_MASK, [697](#)
 - WSPI_CFG_DATA_DTR, [700](#)
 - WSPI_CFG_DATA_MODE_EIGHT_LINES, [700](#)
 - WSPI_CFG_DATA_MODE_FOUR_LINES, [700](#)
 - WSPI_CFG_DATA_MODE_MASK, [700](#)
 - WSPI_CFG_DATA_MODE_NONE, [700](#)
 - WSPI_CFG_DATA_MODE_ONE_LINE, [700](#)
 - WSPI_CFG_DATA_MODE_TWO_LINES, [700](#)
 - WSPI_CFG_DQS_ENABLE, [700](#)
 - WSPI_CFG_SIOO, [700](#)
 - WSPI_COMPLETE, [707](#)
 - WSPI_DEFAULT_CFG_MASKS, [705](#)
 - wspi_ild_command, [719](#)
 - wspi_ild_config_fields, [706](#)
 - wspi_ild_driver_fields, [706](#)
 - wspi_ild_init, [718](#)
 - wspi_ild_map_flash, [720](#)
 - wspi_ild_receive, [720](#)
 - wspi_ild_send, [719](#)
 - wspi_ild_start, [718](#)
 - wspi_ild_stop, [719](#)
 - wspi_ild_unmap_flash, [721](#)
 - WSPI_MEMMAP, [707](#)
 - WSPI_READY, [707](#)
 - WSPI_RECEIVE, [707](#)
 - WSPI_SEND, [707](#)
 - WSPI_STOP, [707](#)
 - WSPI_SUPPORTS_MEMMAP, [705](#)
 - WSPI_UNINIT, [707](#)
 - WSPI_USE_MUTUAL_EXCLUSION, [696](#)
 - WSPI_USE_WAIT, [696](#)
 - wspiAcquireBus, [717](#)
 - wspicallback_t, [706](#)

wspiCommand, [712](#)
WSPIDriver, [706](#)
WSPID1, [721](#)
wspiInit, [707](#)
wspiMapFlash, [715](#)
wspiMapFlashI, [702](#)
wspiObjectInit, [707](#)
wspiReceive, [714](#)
wspiReleaseBus, [717](#)
wspiSend, [713](#)
wspiStart, [709](#)
wspiStartCommand, [710](#)
wspiStartCommandI, [701](#)
wspiStartReceive, [712](#)
wspiStartReceiveI, [702](#)
wspiStartSend, [711](#)
wspiStartSendI, [701](#)
wspistate_t, [707](#)
wspiStop, [709](#)
wspiUnmapFlash, [716](#)
wspiUnmapFlashI, [703](#)
WSPI_CFG_ADDR_DTR
WSPI Driver, [698](#)
WSPI_CFG_ADDR_MODE_EIGHT_LINES
WSPI Driver, [698](#)
WSPI_CFG_ADDR_MODE_FOUR_LINES
WSPI Driver, [698](#)
WSPI_CFG_ADDR_MODE_MASK
WSPI Driver, [697](#)
WSPI_CFG_ADDR_MODE_NONE
WSPI Driver, [697](#)
WSPI_CFG_ADDR_MODE_ONE_LINE
WSPI Driver, [697](#)
WSPI_CFG_ADDR_MODE_TWO_LINES
WSPI Driver, [697](#)
WSPI_CFG_ADDR_SIZE_16
WSPI Driver, [698](#)
WSPI_CFG_ADDR_SIZE_24
WSPI Driver, [698](#)
WSPI_CFG_ADDR_SIZE_32
WSPI Driver, [698](#)
WSPI_CFG_ADDR_SIZE_8
WSPI Driver, [698](#)
WSPI_CFG_ADDR_SIZE_MASK
WSPI Driver, [698](#)
WSPI_CFG_ALL_DTR
WSPI Driver, [700](#)
WSPI_CFG_ALT_DTR
WSPI Driver, [699](#)
WSPI_CFG_ALT_MODE_EIGHT_LINES
WSPI Driver, [699](#)
WSPI_CFG_ALT_MODE_FOUR_LINES
WSPI Driver, [699](#)
WSPI_CFG_ALT_MODE_MASK
WSPI Driver, [698](#)
WSPI_CFG_ALT_MODE_NONE
WSPI Driver, [698](#)
WSPI_CFG_ALT_MODE_ONE_LINE
WSPI Driver, [699](#)
WSPI_CFG_ALT_MODE_TWO_LINES
WSPI Driver, [699](#)
WSPI_CFG_ALT_SIZE_16
WSPI Driver, [699](#)
WSPI_CFG_ALT_SIZE_24
WSPI Driver, [699](#)
WSPI_CFG_ALT_SIZE_32
WSPI Driver, [699](#)
WSPI_CFG_ALT_SIZE_8
WSPI Driver, [699](#)
WSPI_CFG_ALT_SIZE_MASK
WSPI Driver, [699](#)
WSPI_CFG_CMD_DTR
WSPI Driver, [697](#)
WSPI_CFG_CMD_MODE_EIGHT_LINES
WSPI Driver, [696](#)
WSPI_CFG_CMD_MODE_FOUR_LINES
WSPI Driver, [696](#)
WSPI_CFG_CMD_MODE_MASK
WSPI Driver, [696](#)
WSPI_CFG_CMD_MODE_NONE
WSPI Driver, [696](#)
WSPI_CFG_CMD_MODE_ONE_LINE
WSPI Driver, [696](#)
WSPI_CFG_CMD_MODE_TWO_LINES
WSPI Driver, [696](#)
WSPI_CFG_CMD_SIZE_16
WSPI Driver, [697](#)
WSPI_CFG_CMD_SIZE_24
WSPI Driver, [697](#)
WSPI_CFG_CMD_SIZE_32
WSPI Driver, [697](#)
WSPI_CFG_CMD_SIZE_8
WSPI Driver, [697](#)
WSPI_CFG_CMD_SIZE_MASK
WSPI Driver, [697](#)
WSPI_CFG_DATA_DTR
WSPI Driver, [700](#)
WSPI_CFG_DATA_MODE_EIGHT_LINES
WSPI Driver, [700](#)
WSPI_CFG_DATA_MODE_FOUR_LINES
WSPI Driver, [700](#)
WSPI_CFG_DATA_MODE_MASK
WSPI Driver, [700](#)
WSPI_CFG_DATA_MODE_NONE
WSPI Driver, [700](#)
WSPI_CFG_DATA_MODE_ONE_LINE
WSPI Driver, [700](#)
WSPI_CFG_DATA_MODE_TWO_LINES
WSPI Driver, [700](#)
WSPI_CFG_DQS_ENABLE
WSPI Driver, [700](#)
WSPI_CFG_SIOO
WSPI Driver, [700](#)
wspi_command_t, [1330](#)
addr, [1331](#)

- alt, [1331](#)
- cfg, [1331](#)
- cmd, [1331](#)
- dummy, [1331](#)
- WSPI_COMPLETE
 - WSPI Driver, [707](#)
- WSPI_DEFAULT_CFG_MASKS
 - WSPI Driver, [705](#)
- wspi_llc_command
 - WSPI Driver, [719](#)
- wspi_llc_config_fields
 - WSPI Driver, [706](#)
- wspi_llc_driver_fields
 - WSPI Driver, [706](#)
- wspi_llc_init
 - WSPI Driver, [718](#)
- wspi_llc_map_flash
 - WSPI Driver, [720](#)
- wspi_llc_receive
 - WSPI Driver, [720](#)
- wspi_llc_send
 - WSPI Driver, [719](#)
- wspi_llc_start
 - WSPI Driver, [718](#)
- wspi_llc_stop
 - WSPI Driver, [719](#)
- wspi_llc_unmap_flash
 - WSPI Driver, [721](#)
- WSPI_MEMMAP
 - WSPI Driver, [707](#)
- WSPI_READY
 - WSPI Driver, [707](#)
- WSPI_RECEIVE
 - WSPI Driver, [707](#)
- WSPI_SEND
 - WSPI Driver, [707](#)
- WSPI_STOP
 - WSPI Driver, [707](#)
- WSPI_SUPPORTS_MEMMAP
 - WSPI Driver, [705](#)
- WSPI_UNINIT
 - WSPI Driver, [707](#)
- WSPI_USE_MUTUAL_EXCLUSION
 - Configuration, [55](#)
 - WSPI Driver, [696](#)
- WSPI_USE_WAIT
 - Configuration, [55](#)
 - WSPI Driver, [696](#)
- wspiAcquireBus
 - WSPI Driver, [717](#)
- wspicallback_t
 - WSPI Driver, [706](#)
- wspiCommand
 - WSPI Driver, [712](#)
- WSPIConfig
 - WSPI Driver, [706](#)
- WSPID1
 - WSPI Driver, [721](#)
- WSPIDriver
 - WSPI Driver, [706](#)
- wspiInit
 - WSPI Driver, [707](#)
- wspiMapFlash
 - WSPI Driver, [715](#)
- wspiMapFlashI
 - WSPI Driver, [702](#)
- wspiObjectInit
 - WSPI Driver, [707](#)
- wspiReceive
 - WSPI Driver, [714](#)
- wspiReleaseBus
 - WSPI Driver, [717](#)
- wspiSend
 - WSPI Driver, [713](#)
- wspiStart
 - WSPI Driver, [709](#)
- wspiStartCommand
 - WSPI Driver, [710](#)
- wspiStartCommandI
 - WSPI Driver, [701](#)
- wspiStartReceive
 - WSPI Driver, [712](#)
- wspiStartReceiveI
 - WSPI Driver, [702](#)
- wspiStartSend
 - WSPI Driver, [711](#)
- wspiStartSendI
 - WSPI Driver, [701](#)
- wspistate_t
 - WSPI Driver, [707](#)
- wspiStop
 - WSPI Driver, [709](#)
- wspiUnmapFlash
 - WSPI Driver, [716](#)
- wspiUnmapFlashI
 - WSPI Driver, [703](#)
- xsno.doc, [1438](#)
- xsno_arch.doc, [1438](#)
- XSNOR_BUFFER_SIZE
 - Base Serial NOR Driver class, [836](#)
- xsno_buffers, [1332](#)
 - cmdbuf, [1332](#)
 - databuf, [1332](#)
 - spibuf, [1333](#)
- xsno_buffers_t
 - Base Serial NOR Driver class, [837](#)
- xsno_bus_configs, [1333](#)
 - spi, [1335](#)
 - wspi, [1335](#)
- XSNOR_BUS_MODE_SPI
 - Base Serial NOR Driver class, [835](#)
- XSNOR_BUS_MODE_WSPI_1LINE
 - Base Serial NOR Driver class, [835](#)
- XSNOR_BUS_MODE_WSPI_2LINES
 - Base Serial NOR Driver class, [835](#)
- XSNOR_BUS_MODE_WSPI_4LINES

- Base Serial NOR Driver class, [836](#)
- XSNOR_BUS_MODE_WSPI_8LINES
 - Base Serial NOR Driver class, [836](#)
- xsnor_bus_spi, [1335](#)
 - cfg, [1336](#)
 - drv, [1336](#)
- xsnor_bus_wspi, [1336](#)
 - cfg, [1337](#)
 - drv, [1337](#)
- xsnor_commands, [1338](#)
 - cmd, [1338](#)
 - cmd_addr, [1338](#)
 - cmd_addr_data, [1339](#)
 - cmd_data, [1339](#)
- xsnor_commands_t
 - Base Serial NOR Driver class, [837](#)
- xsnor_config, [1339](#)
 - buffers, [1340](#)
 - bus, [1340](#)
 - bus_type, [1340](#)
 - options, [1340](#)
- xsnor_config_t
 - Base Serial NOR Driver class, [837](#)
- xsnor_device_init
 - Base Serial NOR Driver class, [849](#)
- xsnor_device_mmap_off
 - Base Serial NOR Driver class, [852](#)
- xsnor_device_mmap_on
 - Base Serial NOR Driver class, [852](#)
- xsnor_device_program
 - Base Serial NOR Driver class, [849](#)
- xsnor_device_query_erase
 - Base Serial NOR Driver class, [851](#)
- xsnor_device_read
 - Base Serial NOR Driver class, [849](#)
- xsnor_device_start_erase_all
 - Base Serial NOR Driver class, [850](#)
- xsnor_device_start_erase_sector
 - Base Serial NOR Driver class, [850](#)
- xsnor_device_verify_erase
 - Base Serial NOR Driver class, [851](#)
- xsnor_macronix_mx25.dox, [1438](#)
- xsnor_micron_n25q.dox, [1438](#)
- XSNOR_SHARED_BUS
 - Base Serial NOR Driver class, [836](#)
- XSNOR_USE_BOTH
 - Base Serial NOR Driver class, [836](#)
- XSNOR_USE_SPI
 - Base Serial NOR Driver class, [836](#)
- XSNOR_USE_WSPI
 - Base Serial NOR Driver class, [836](#)
- xsnorMemoryMap
 - Base Serial NOR Driver class, [847](#)
- xsnorMemoryUnmap
 - Base Serial NOR Driver class, [848](#)
- xsnorStart
 - Base Serial NOR Driver class, [846](#)
- xsnorStop
 - Base Serial NOR Driver class, [847](#)
- year
 - RTCDatetime, [1274](#)
- zero_status
 - USB Driver, [684](#)