Cordio Platform Documentation

Generated by Doxygen 1.8.17

1 Module Index 1

1.1 Modules . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Module Documentation 7

4.1 PAL_BUTTON . . . 7

4.1.1 Detailed Description . . . 7

4.1.2 Enumeration Type Documentation . . . 7

4.1.2.1 PalBtnPos_t . . . 7

4.1.2.2 PalBtnState_t . . . 8

4.2 PAL_CFG . . . 9

4.2.1 Detailed Description . . . 9

4.2.2 Enumeration Type Documentation . . . 9

4.2.2.1 PalCfgId_t . . . 9

4.3 PAL_SYS . . . 10

4.3.1 Detailed Description . . . 10

4.4 PAL_CRYPTO . . . 11

4.4.1 Detailed Description . . . 12

4.4.2 Macro Definition Documentation . . . 12

4.4.2.1 PAL_CRYPTO_LL_DATA_MIC_LEN . . . 12

4.4.2.2 PAL_CRYPTO_LL_IV_LEN . . . 12

4.4.2.3 PAL_CRYPTO_LL_KEY_LEN . . . 12

4.4.3 Enumeration Type Documentation . . . 12

4.4.3.1 PalCryptoState_t . . . 12

4.5 PAL_BB_BLE . . . 13

4.5.1 Detailed Description . . . 14

4.5.2 Macro Definition Documentation . . . 14

4.5.2.1 LL_ENABLE_TESTER . . . 14

4.5.3 Enumeration Type Documentation . . . 14

4.5.3.1 PalBbBleConn_t . . . 14

4.5.3.2 PalBbBleNonce_m . . . 14

4.5.3.3 PalBbIfsMode_t . . . 15

4.5.4 Function Documentation . . . 15

4.5.4.1 PalBbBleDisable() . . . 15

4.5.4.2 PalBbBleEnable() . . . 15

4.5.4.3 PalBbBleInit() . . . 15

4.6 PAL_BB_BLE_CHAN . . . 16

4.6.1 Detailed Description . . . 16

4.6.2 Function Documentation . . . 16
4.6.2.1 PalBbBleSetChannelParam() . . . 16
4.7 PAL_BB_BLE_DATA . . . 17
4.7.1 Detailed Description . . . 17
4.7.2 Function Documentation . . . 17
4.7.2.1 PalBbBleCancelData() . . . 17
4.7.2.2 PalBbBleCancelTifs() . . . 18
4.7.2.3 PalBbBleEnableDataWhitening() . . . 18
4.7.2.4 PalBbBleEnablePrbs15() . . . 18
4.7.2.5 PalBbBleInlineEncryptDecryptSetDirection() . . . 18
4.7.2.6 PalBbBleInlineEncryptSetPacketCount() . . . 19
4.7.2.7 PalBbBleLowPower() . . . 19
4.7.2.8 PalBbBleRxData() . . . 19
4.7.2.9 PalBbBleRxTifsData() . . . 19
4.7.2.10 PalBbBleSetDataParams() . . . 20
4.7.2.11 PalBbBleSetOpParams() . . . 20
4.7.2.12 PalBbBleTxData() . . . 20
4.7.2.13 PalBbBleTxTifsData() . . . 21
4.8 PAL_TIMER . . . 22
4.8.1 Detailed Description . . . 22
4.8.2 Enumeration Type Documentation . . . 22
4.8.2.1 PalTimerState_t . . . 22
4.9 PAL_LED . . . 23
4.9.1 Detailed Description . . . 23
4.9.2 Enumeration Type Documentation . . . 23
4.9.2.1 PalLedReserved_id . . . 23
4.10 PAL_RTC . . . 24
4.10.1 Detailed Description . . . 24
4.10.2 Enumeration Type Documentation . . . 24
4.10.2.1 PalRtcState_t . . . 24
4.11 PAL_BB . . . 25
4.11.1 Detailed Description . . . 26
4.11.2 Enumeration Type Documentation . . . 26
4.11.2.1 PalBbPhy_op . . . 26
4.11.2.2 PalBbPhy_t . . . 26
4.11.2.3 PalBbProt_t . . . 27
4.11.2.4 PalBbStat_c . . . 27
4.11.3 Function Documentation . . . 28
4.11.3.1 PalBbDisable() . . . 28
4.11.3.2 PalBbEnable() . . . 28
4.11.3.3 PalBbGetCurrentTime() . . . 28
4.11.3.4 PalBbGetTimestamp() . . . 28

4.11.3.5 PalBbInit() 29
4.11.3.6 PalBbLoadCfg() 29
4.11.3.7 PalBbRegisterProtIrq() 29
4.11.3.8 PalBbRestore() 30
4.11.3.9 PalBbSetProtId() 30
4.12 PAL_UART 31
4.12.1 Detailed Description 31
4.12.2 Enumeration Type Documentation 31
4.12.2.1 PalUartId_t 31
4.12.2.2 PalUartState_t 32

5 Class Documentation **33**
5.1 PalBbBleChan_t Struct Reference 33
5.1.1 Detailed Description 34
5.1.2 Member Data Documentation 34
5.1.2.1 accAddr 34
5.1.2.2 chanIdx 34
5.1.2.3 crcInit 34
5.1.2.4 enc 34
5.1.2.5 initTxPhyOptions 34
5.1.2.6 opType 34
5.1.2.7 peerRxStableModIdx 35
5.1.2.8 peerTxStableModIdx 35
5.1.2.9 rxPhy 35
5.1.2.10 tifsTxPhyOptions 35
5.1.2.11 txPhy 35
5.1.2.12 txPower 35
5.2 PalBbBleDataParam_t Struct Reference 35
5.2.1 Detailed Description 36
5.2.2 Member Data Documentation 36
5.2.2.1 dueUsec 36
5.2.2.2 rxCback 36
5.2.2.3 rxTimeoutUsec 36
5.2.2.4 txCback 36
5.3 PalBbBleOpParam_t Struct Reference 37
5.3.1 Detailed Description 37
5.3.2 Member Data Documentation 37
5.3.2.1 ifsMode 37
5.3.2.2 ifsTime 38
5.3.2.3 pIfsChan 38
5.4 PalBbBleTxBufDesc_t Struct Reference 38
5.4.1 Detailed Description 38

5.4.2 Member Data Documentation 38
5.4.2.1 len 38
5.4.2.2 pBuf 39
5.5 PalBbCfg_t Struct Reference 39
5.5.1 Detailed Description 39
5.5.2 Member Data Documentation 39
5.5.2.1 BbTimerBoundaryUsec 39
5.5.2.2 clkPpm 39
5.5.2.3 maxScanPeriodMsec 40
5.5.2.4 rfSetupDelayUsec 40
5.5.2.5 schSetupDelayUsec 40
5.6 PalCryptoEnc_t Struct Reference 40
5.6.1 Detailed Description 40
5.6.2 Member Data Documentation 41
5.6.2.1 dir 41
5.6.2.2 enaAuth 41
5.6.2.3 enaDecrypt 41
5.6.2.4 enaEncrypt 41
5.6.2.5 iv 41
5.6.2.6 nonceMode 41
5.6.2.7 pDecryptCtx 41
5.6.2.8 pEncryptCtx 42
5.6.2.9 pEventCounter 42
5.6.2.10 pRxPktCounter 42
5.6.2.11 pTxPktCounter 42
5.6.2.12 sk 42
5.6.2.13 type 42
5.7 PalUartConfig_t Struct Reference 42
5.7.1 Detailed Description 43
5.7.2 Member Data Documentation 43
5.7.2.1 baud 43
5.7.2.2 hwFlow 43
5.7.2.3 rdCback 43
5.7.2.4 wrCback 43

6 File Documentation **45**
6.1 /home/aw/msdk/Libraries/Cordio/platform/include/pal_bb.h File Reference 45
6.1.1 Detailed Description 47
6.2 /home/aw/msdk/Libraries/Cordio/platform/include/pal_bb_ble.h File Reference 47
6.2.1 Detailed Description 50
6.3 /home/aw/msdk/Libraries/Cordio/platform/include/pal_btn.h File Reference 50
6.3.1 Detailed Description 51

6.3.2 Enumeration Type Documentation 51
6.3.2.1 anonymous enum 51
6.4 /home/aw/msdk/Libraries/Cordio/platform/include/pal_cfg.h File Reference 52
6.4.1 Detailed Description 52
6.5 /home/aw/msdk/Libraries/Cordio/platform/include/pal_crypto.h File Reference 53
6.5.1 Detailed Description 54
6.6 /home/aw/msdk/Libraries/Cordio/platform/include/pal_led.h File Reference 54
6.6.1 Detailed Description 55
6.7 /home/aw/msdk/Libraries/Cordio/platform/include/pal_rtc.h File Reference 56
6.7.1 Detailed Description 57
6.8 /home/aw/msdk/Libraries/Cordio/platform/include/pal_sys.h File Reference 57
6.8.1 Detailed Description 58
6.9 /home/aw/msdk/Libraries/Cordio/platform/include/pal_timer.h File Reference 58
6.9.1 Detailed Description 59
6.10 /home/aw/msdk/Libraries/Cordio/platform/include/pal_uart.h File Reference 60
6.10.1 Detailed Description 61

Index **63**

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

PAL_BUTTON 7
PAL_CFG 9
PAL_SYS 10
PAL_CRYPTO 11
PAL_BB_BLE 13
PAL_BB_BLE_CHAN 16
PAL_BB_BLE_DATA 17
PAL_TIMER 22
PAL_LED 23
PAL_RTC 24
PAL_BB 25
PAL_UART 31

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

PalBbBleChan_t
BLE channelization parameters 33
PalBbBleDataParam_t
BLE data transfer parameters 35
PalBbBleOpParam_t
Operation parameters 37
PalBbBleTxBufDesc_t
Transmit buffer descriptor 38
PalBbCfg_t
BB configuration 39
PalCryptoEnc_t
Encryption data 40
PalUartConfig_t
Peripheral configuration 42

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

/home/aw/msdk/Libraries/Cordio/platform/include/pal_bb.h
Baseband interface file 45
/home/aw/msdk/Libraries/Cordio/platform/include/pal_bb_ble.h
BLE Baseband interface file 47
/home/aw/msdk/Libraries/Cordio/platform/include/pal_btn.h
Button driver definition 50
/home/aw/msdk/Libraries/Cordio/platform/include/pal_cfg.h
System configuration definition 52
/home/aw/msdk/Libraries/Cordio/platform/include/pal_crypto.h
Crypto driver definition 53
/home/aw/msdk/Libraries/Cordio/platform/include/pal_led.h
LED driver definition 54
/home/aw/msdk/Libraries/Cordio/platform/include/pal_rtc.h
RTC timer interface file 56
/home/aw/msdk/Libraries/Cordio/platform/include/pal_sys.h
System hooks 57
/home/aw/msdk/Libraries/Cordio/platform/include/pal_timer.h
Timer interface file 58
/home/aw/msdk/Libraries/Cordio/platform/include/pal_uart.h
UART driver definition 60

Chapter 4

Module Documentation

4.1 PAL_BUTTON

Typedefs

- typedef void(∗ PalBtnActionCback_t) (uint8_t btnId, PalBtnPos_t state)

  Action callback signature.

Enumerations

- enum PalBtnState_t { PAL_BTN_STATE_UNINIT = 0, PAL_BTN_STATE_ERROR = 0, PAL_BTN_STATE_READY }

  Operational states.
- enum PalBtnPos_t { PAL_BTN_POS_INVALID, PAL_BTN_POS_DOWN, PAL_BTN_POS_UP }

  Button position.

Functions

- void **PalBtnInit** (PalBtnActionCback_t actCback)
- void **PalBtnDeInit** (void)
- PalBtnState_t **PalBtnGetState** (void)
- PalBtnPos_t **PalBtnGetPosition** (uint8_t id)

4.1.1 Detailed Description

4.1.2 Enumeration Type Documentation

4.1.2.1 PalBtnPos_t

```
enum PalBtnPos_t
```

Button position.

Enumerator

PAL_BTN_POS_INVALID	Button position is invalid.
PAL_BTN_POS_DOWN	Button position is depressed.
PAL_BTN_POS_UP	Button position is released.

4.1.2.2 PalBtnState_t

```
enum PalBtnState_t
```

Operational states.

Enumerator

PAL_BTN_STATE_UNINIT	Uninitialized state.
PAL_BTN_STATE_ERROR	Error state.
PAL_BTN_STATE_READY	Ready state.

4.2 PAL_CFG

Enumerations

- enum PalCfgId_t {
  PAL_CFG_ID_BD_ADDR, PAL_CFG_ID_BLE_PHY, PAL_CFG_ID_LL_PARAM, PAL_CFG_ID_MAC_ADDR,
  PAL_CFG_ID_UUID }

  Configuration ID.

Functions

- void **PalCfgLoadData** (uint8_t cfgId, uint8_t ∗pBuf, uint32_t len)
- void **PalCfgSetDeviceUuid** (uint8_t ∗pBuf)

4.2.1 Detailed Description

4.2.2 Enumeration Type Documentation

4.2.2.1 PalCfgId_t

```
enum PalCfgId_t
```

Configuration ID.

Enumerator

PAL_CFG_ID_BD_ADDR	BD address.
PAL_CFG_ID_BLE_PHY	Ble PHY.
PAL_CFG_ID_LL_PARAM	LL parameters.
PAL_CFG_ID_MAC_ADDR	MAC address.
PAL_CFG_ID_UUID	UUID.

4.3 PAL_SYS

Macros

- #define PAL_SYS_ASSERT(expr)

  Parameter check (disabled).

Functions

- void **PalSysInit** (void)
- void **PalSysAssertTrap** (void)
- void **PalSysSetTrap** (bool_t enable)
- uint32_t **PalSysGetAssertCount** (void)
- uint32_t **PalSysGetStackUsage** (void)
- void **PalSysSleep** (void)
- bool_t **PalSysIsBusy** (void)
- void **PalSysSetBusy** (void)
- void **PalSysSetIdle** (void)
- void **PalEnterCs** (void)
- void **PalExitCs** (void)

4.3.1 Detailed Description

4.4 PAL_CRYPTO

Classes

- struct PalCryptoEnc_t

  Encryption data.

Macros

- #define PAL_CRYPTO_AES_BLOCK_SIZE 16

  AES block size.
- #define PAL_CRYPTO_LL_KEY_LEN 16
- #define PAL_CRYPTO_LL_IV_LEN 8
- #define PAL_CRYPTO_LL_DATA_MIC_LEN 4
- #define SEC_CCM_KEY_LEN 16

  CCM-Mode algorithm lengths.
- #define SEC_CCM_MAX_ADDITIONAL_LEN ((1<<16) - (1<<8))

  CCM-Mode algorithm maximum additional length.
- #define SEC_CCM_L 2

  CCM-Mode algorithm length.
- #define SEC_CCM_NONCE_LEN (15-SEC_CCM_L)

  CCM-Mode algorithm nonce length.

Enumerations

- enum PalCryptoState_t { PAL_CRYPTO_STATE_UNINIT = 0, PAL_CRYPTO_STATE_ERROR = 0, PAL_CRYPTO_STATE_READY }

  Operational states.

Functions

- void **PalCryptoInit** (void)
- void **PalCryptoDeInit** (void)
- void **PalCryptoGenerateP256KeyPair** (const uint8_t ∗pPrivKey, uint8_t ∗pPubKey)
- void **PalCryptoGenerateDhKey** (const uint8_t ∗pPubKey, const uint8_t ∗pPrivKey, uint8_t ∗pDhKey)
- bool_t **PalCryptoValidatePublicKey** (const uint8_t ∗pPubKey, bool_t generateKey)
- void **PalCryptoGenerateRandomNumber** (uint8_t ∗pBuf, uint8_t len)
- uint32_t **PalCryptoCcmDec** (const uint8_t ∗pKey, uint8_t ∗pNonce, uint8_t ∗pCypherText, uint16_t text↩
Len, uint8_t ∗pClear, uint16_t clearLen, uint8_t ∗pMic, uint8_t micLen, uint8_t ∗pResult, uint8_t handlerId, uint16_t param, uint8_t event)
- void **PalCryptoCcmEnc** (const uint8_t ∗pKey, uint8_t ∗pNonce, uint8_t ∗pPlainText, uint16_t textLen, uint8↩
_t ∗pClear, uint16_t clearLen, uint8_t micLen, uint8_t ∗pResult, uint8_t handlerId, uint16_t param, uint8_t event)
- void **PalCryptoAesEcb** (const uint8_t ∗pKey, uint8_t ∗pOut, const uint8_t ∗pIn)
- void **PalCryptoAesCmac** (const uint8_t ∗pKey, uint8_t ∗pOut, const uint8_t ∗pIn, uint16_t len)
- void **PalCryptoAesEnable** (PalCryptoEnc_t ∗pEnc, uint8_t id, uint8_t localDir)
- bool_t **PalCryptoAesCcmEncrypt** (PalCryptoEnc_t ∗pEnc, uint8_t ∗pHdr, uint8_t ∗pBuf, uint8_t ∗pMic)
- bool_t **PalCryptoAesCcmDecrypt** (PalCryptoEnc_t ∗pEnc, uint8_t ∗pBuf)
- void **PalCryptoSetEncryptPacketCount** (PalCryptoEnc_t ∗pEnc, uint64_t pktCnt)
- void **PalCryptoSetDecryptPacketCount** (PalCryptoEnc_t ∗pEnc, uint64_t pktCnt)

4.4.1 Detailed Description

4.4.2 Macro Definition Documentation

4.4.2.1 PAL_CRYPTO_LL_DATA_MIC_LEN

```
#define PAL_CRYPTO_LL_DATA_MIC_LEN 4
```

Data channel PDU MIC length.

4.4.2.2 PAL_CRYPTO_LL_IV_LEN

```
#define PAL_CRYPTO_LL_IV_LEN 8
```

Initialization vector length.

4.4.2.3 PAL_CRYPTO_LL_KEY_LEN

```
#define PAL_CRYPTO_LL_KEY_LEN 16
```

Encryption key length.

4.4.3 Enumeration Type Documentation

4.4.3.1 PalCryptoState_t

```
enum PalCryptoState_t
```

Operational states.

Enumerator

PAL_CRYPTO_STATE_UNINIT	Uninitialized state.
PAL_CRYPTO_STATE_ERROR	Error state.
PAL_CRYPTO_STATE_READY	Ready state.

4.5 PAL_BB_BLE

Classes

- struct PalBbBleChan_t

  BLE channelization parameters.
- struct PalBbBleDataParam_t

  BLE data transfer parameters.
- struct PalBbBleOpParam_t

  Operation parameters.
- struct PalBbBleTxBufDesc_t

  Transmit buffer descriptor.

Macros

- #define LL_ENABLE_TESTER 0

Typedefs

- typedef void(∗ PalBbBleTxIsr_t) (uint8_t status)

  Transmit complete ISR callback signature.
- typedef void(∗ PalBbBleRxIsr_t) (uint8_t status, int8_t rssi, uint32_t crc, uint32_t timestamp, uint8_t rxPhy↩ Options)

  Receive complete ISR callback signature.

Enumerations

- enum PalBbBleNonce_m { PAL_BB_NONCE_MODE_PKT_CNTR, PAL_BB_NONCE_MODE_EXT16_CNTR, PAL_BB_NONCE_MODE_EXT64_CNTR }

  Nonce modes.
- enum PalBbBleConn_t { PAL_BB_TYPE_ACL, PAL_BB_TYPE_CIS, PAL_BB_TYPE_BIS }

  Connection type.
- enum PalBbIfsMode_t { PAL_BB_IFS_MODE_CLR, PAL_BB_IFS_MODE_TOGGLE_TIFS, PAL_BB_IFS_MODE_SAME_ABS }

  IFS modes.

Functions

- void PalBbBleInit (void)

  Initialize the BLE baseband driver.
- void PalBbBleEnable (void)

  Enable the BB hardware.
- void PalBbBleDisable (void)

  Disable the BB hardware.

4.5.1 Detailed Description

4.5.2 Macro Definition Documentation

4.5.2.1 LL_ENABLE_TESTER

```
#define LL_ENABLE_TESTER 0
```

Enable LL tester extensions.

4.5.3 Enumeration Type Documentation

4.5.3.1 PalBbBleConn_t

```
enum PalBbBleConn_t
```

Connection type.

Enumerator

PAL_BB_TYPE_ACL	ACL.
PAL_BB_TYPE_CIS	CIS.
PAL_BB_TYPE_BIS	BIS.

4.5.3.2 PalBbBleNonce_m

```
enum PalBbBleNonce_m
```

Nonce modes.

Enumerator

PAL_BB_NONCE_MODE_PKT_CNTR	Packet counter mode (default).
PAL_BB_NONCE_MODE_EXT16_CNTR	16-bit counter mode, PalCryptoEnc_t::pEventCounter must be non-NULL.
PAL_BB_NONCE_MODE_EXT64_CNTR	64-bit counter mode, PalCryptoEnc_t::pTxPktCounter/pRxPktCounter must be non-NULL.

4.5.3.3 PalBbIfsMode_t

```
enum PalBbIfsMode_t
```

IFS modes.

Enumerator

PAL_BB_IFS_MODE_CLR	Clear IFS (last packet).
PAL_BB_IFS_MODE_TOGGLE_TIFS	Toggle operation with TIFS timing.
PAL_BB_IFS_MODE_SAME_ABS	Same operation with absolute timing.

4.5.4 Function Documentation

4.5.4.1 PalBbBleDisable()

```
void PalBbBleDisable (
             void  )
```

Disable the BB hardware.

Disable the baseband and put radio hardware to sleep. Must be called from an idle state. A radio operation cannot be in progress.

4.5.4.2 PalBbBleEnable()

```
void PalBbBleEnable (
             void  )
```

Enable the BB hardware.

Wake the BB hardware out of sleep and enable for operation. All BB functionality is available when this routine completes. BB clock is set to zero and started.

4.5.4.3 PalBbBleInit()

```
void PalBbBleInit (
             void  )
```

Initialize the BLE baseband driver.

One-time initialization of BLE baseband driver.

4.6 PAL_BB_BLE_CHAN

Functions

- void PalBbBleSetChannelParam (PalBbBleChan_t *pChan)

  Set channelization parameters.

4.6.1 Detailed Description

This section contains the driver routine used to set the chanelization parameters.

4.6.2 Function Documentation

4.6.2.1 PalBbBleSetChannelParam()

```
void PalBbBleSetChannelParam (
            PalBbBleChan_t * pChan )
```

Set channelization parameters.

Parameters

pChan	Channelization parameters.

Calling this routine will set parameters for all future transmit and receive operations until this routine is called again providing new parameters.

The setting of channelization parameters influence the operations of the following listed routines. Therefore, this routine is called to set the channel characteristics before the use of data routines described in *PAL_BB_BLE_DATA*.

Note

The *pParam* contents are not guaranteed to be static and is only valid in the context of the call to this routine. Therefore parameters requiring persistence should be copied.

4.7 PAL_BB_BLE_DATA

Functions

- void PalBbBleSetDataParams (const PalBbBleDataParam_t ∗pParam)

  Set the data packet exchange parameters.
- void PalBbBleSetOpParams (const PalBbBleOpParam_t ∗pOpParam)

  Set the operation parameters.
- void PalBbBleTxData (PalBbBleTxBufDesc_t descs[ ], uint8_t cnt)

  Transmit a packet.
- void PalBbBleTxTifsData (PalBbBleTxBufDesc_t descs[ ], uint8_t cnt)

  Transmit packet at TIFS after the last packet received.
- void PalBbBleRxData (uint8_t ∗pBuf, uint16_t len)

  Receive packet.
- void PalBbBleRxTifsData (uint8_t ∗pBuf, uint16_t len)

  Receive packet at TIFS after the last packet transmitted.
- void PalBbBleCancelTifs (void)

  Cancel TIFS timer.
- void PalBbBleCancelData (void)

  Cancel a pending transmit or receive.
- void PalBbBleEnableDataWhitening (bool_t enable)

  Enable or disable data whitening.
- void PalBbBleEnablePrbs15 (bool_t enable)

  Enable or disable PRBS15.
- void PalBbBleInlineEncryptDecryptSetDirection (uint8_t dir)

  Set inline encryption/decryption direction bit.
- void PalBbBleInlineEncryptSetPacketCount (uint64_t count)

  Set the inline encryption packet count for transmit.
- void PalBbBleLowPower (void)

  Low power operation.

4.7.1 Detailed Description

This section contains driver routines used for packet transmission.

4.7.2 Function Documentation

4.7.2.1 PalBbBleCancelData()

```
void PalBbBleCancelData (
            void  )
```

Cancel a pending transmit or receive.

This stops any active radio operation. This routine is never called in the callback (i.e. ISR) context.

4.7.2.2 PalBbBleCancelTifs()

```
void PalBbBleCancelTifs (
            void  )
```

Cancel TIFS timer.

This stops any active TIFS timer operation. This routine is always called in the callback (i.e. ISR) context.

4.7.2.3 PalBbBleEnableDataWhitening()

```
void PalBbBleEnableDataWhitening (
            bool_t enable )
```

Enable or disable data whitening.

Parameters

enable	Flag to indicate data whitening.

Sets an internal variable that indicates if data whitening is enabled or not.

4.7.2.4 PalBbBleEnablePrbs15()

```
void PalBbBleEnablePrbs15 (
            bool_t enable )
```

Enable or disable PRBS15.

Parameters

enable	Flag to indicate PRBS15.

Immediately enable or disable continuous PRBS15 bitstream. Setting the channelization parameters with *PalBbBleSetChannelParam()* must precede enabling PRBS15.

Use of *PAL_BB_BLE_DATA* routines is not allowed while PRBS15 is enabled.

4.7.2.5 PalBbBleInlineEncryptDecryptSetDirection()

```
void PalBbBleInlineEncryptDecryptSetDirection (
            uint8_t dir )
```

Set inline encryption/decryption direction bit.

Parameters

dir	0=slave, non-zero=master

4.7.2.6 PalBbBleInlineEncryptSetPacketCount()

```
void PalBbBleInlineEncryptSetPacketCount (
            uint64_t count )
```

Set the inline encryption packet count for transmit.

Parameters

count	Packet counter value, a 39-bit value

4.7.2.7 PalBbBleLowPower()

```
void PalBbBleLowPower (
            void  )
```

Low power operation.

Note

Called by upper baseband code.

4.7.2.8 PalBbBleRxData()

```
void PalBbBleRxData (
            uint8_t * pBuf,
            uint16_t len )
```

Receive packet.

Parameters

pBuf	Receive data buffer.
len	Length of data buffer.

Set the first data buffer for the first packet of an alternating Rx-Tx data exchange cycle.

4.7.2.9 PalBbBleRxTifsData()

```
void PalBbBleRxTifsData (
            uint8_t * pBuf,
            uint16_t len )
```

Receive packet at TIFS after the last packet transmitted.

Parameters

pBuf	Receive data buffer.
len	Length of data buffer.

If possible, the receive will occur on the TIFS timing. If not possible, the callback status will indicate this.

4.7.2.10 PalBbBleSetDataParams()

```
void PalBbBleSetDataParams (
            const PalBbBleDataParam_t * pParam )
```

Set the data packet exchange parameters.

Parameters

pParam	Data exchange parameters.

Calling this routine will set parameters for all future transmit and receive operations until this routine is called again providing new parameters.

4.7.2.11 PalBbBleSetOpParams()

```
void PalBbBleSetOpParams (
            const PalBbBleOpParam_t * pOpParam )
```

Set the operation parameters.

Parameters

pOpParam	Operations parameters.

Calling this routine will set parameters for the next transmit or receive operations.

4.7.2.12 PalBbBleTxData()

```
void PalBbBleTxData (
            PalBbBleTxBufDesc_t descs[],
            uint8_t cnt )
```

Transmit a packet.

Parameters

descs	Array of transmit buffer descriptors.
cnt	Number of descriptors.

Set the first data buffer for the first packet of an alternating Tx-Rx data exchange cycle.

4.7.2.13 PalBbBleTxTifsData()

```
void PalBbBleTxTifsData (
            PalBbBleTxBufDesc_t descs[],
            uint8_t cnt )
```

Transmit packet at TIFS after the last packet received.

Parameters

descs	Array of transmit buffer descriptor.
cnt	Number of descriptors.

If possible, the transmit will occur at the TIFS timing. If not possible, the callback status will indicate this.

4.8 PAL_TIMER

Typedefs

- typedef void(∗ PalTimerCompCback_t) (void)

  Completion callback.

Enumerations

- enum PalTimerState_t { PAL_TIMER_STATE_UNINIT = 0, PAL_TIMER_STATE_ERROR = 0, PAL_TIMER_STATE_READY,
  PAL_TIMER_STATE_BUSY }

  Operational states.

Functions

- void **PalTimerInit** (PalTimerCompCback_t expCback)
- void **PalTimerDeInit** (void)
- PalTimerState_t **PalTimerGetState** (void)
- void **PalTimerStart** (uint32_t expUsec)
- void **PalTimerStop** (void)
- uint32_t **PalTimerGetCurrentTime** (void)
- uint32_t **PalTimerGetExpTime** (void)
- void **PalTimerSleep** (uint32_t expUsec)
- void **PalTimerRestore** (uint32_t schTime)
- void **PalTimerSetIRQPriority** (uint32_t priority)

4.8.1 Detailed Description

4.8.2 Enumeration Type Documentation

4.8.2.1 PalTimerState_t

```
enum PalTimerState_t
```

Operational states.

Enumerator

PAL_TIMER_STATE_UNINIT	Uninitialized state.
PAL_TIMER_STATE_ERROR	Error state.
PAL_TIMER_STATE_READY	Ready state.
PAL_TIMER_STATE_BUSY	Busy state.

4.9 PAL_LED

Enumerations

- enum PalLedReserved_id { PAL_LED_ID_CPU_ACTIVE = 0x30, PAL_LED_ID_ERROR = 0x31 }

  Reserved LED IDs.

Functions

- void **PalLedInit** (void)
- void **PalLedDeInit** (void)
- void **PalLedOn** (uint8_t id)
- void **PalLedOff** (uint8_t id)

4.9.1 Detailed Description

4.9.2 Enumeration Type Documentation

4.9.2.1 PalLedReserved_id

```
enum PalLedReserved_id
```

Reserved LED IDs.

Enumerator

PAL_LED_ID_CPU_ACTIVE	CPU active LED ID.
PAL_LED_ID_ERROR	Error LED ID.

4.10 PAL_RTC

Macros

- #define PAL_MAX_RTC_COUNTER_VAL (0x00FFFFFF)

  Max value of RTC.
- #define PAL_RTC_TICKS_PER_SEC (32768) /∗ RTC ticks per second (with prescaler) ∗/

  Clock frequency of the RTC timer used.

Typedefs

- typedef void(∗ palRtcIrqCback_t) (void)

  Platform RTC callback.

Enumerations

- enum PalRtcState_t { PAL_RTC_STATE_UNINIT = 0, PAL_RTC_STATE_ERROR = 0, PAL_RTC_STATE_READY = 1 }

  Operational states.

Functions

- void **PalRtcInit** (void)
- void **PalRtcEnableCompareIrq** (uint8_t channelId)
- void **PalRtcDisableCompareIrq** (uint8_t channelId)
- uint32_t **PalRtcCounterGet** (void)
- void **PalRtcCompareSet** (uint8_t channelId, uint32_t value)
- PalRtcState_t **PalRtcGetState** (void)

4.10.1 Detailed Description

4.10.2 Enumeration Type Documentation

4.10.2.1 PalRtcState_t

```
enum PalRtcState_t
```

Operational states.

Enumerator

PAL_RTC_STATE_UNINIT	Uninitialized state.
PAL_RTC_STATE_ERROR	Error state.
PAL_RTC_STATE_READY	Ready state.

4.11 PAL_BB

Classes

- struct PalBbCfg_t

  BB configuration.

Macros

- #define BB_CLK_RATE_HZ 1000000

  BB clock rate in hertz.
- #define BB_MATH_DIV_10E6(n) ((uint32_t)(((uint64_t)(n) ∗ UINT64_C(4295)) >> 32))

  Binary divide with 1,000,000 divisor (n[max]=0xFFFFFFFF).
- #define BB_US_TO_BB_TICKS(us) (us)

  Return microseconds (no conversion required).
- #define **RTC_CLOCK_RATE** 32768
- #define **USE_RTC_BB_CLK** (BB_CLK_RATE_HZ == RTC_CLOCK_RATE)
- #define BB_TICKS_TO_US(n) (n)

  BB ticks to microseconds (no conversion required).
- #define BB_MAX_SCAN_PERIOD_MS 1000

  Typical maximum duration to scan in a scan interval (BbRtCfg_t::maxScanPeriodMs).
- #define BB_RF_SETUP_DELAY_US 150

  Typical RF setup delay (BbRtCfg_t::rfSetupDelayUs).
- #define BB_SCH_SETUP_DELAY_US 500

  Typical operation setup delay in microseconds (BbRtCfg_t::schSetupDelayUs).
- #define BB_TIMER_1MHZ_MAX_VALUE_US 0xFFFFFFFF /∗ 2^32 - 1 = 0xFFFFFFFF. ∗/

  Maximum time tick for 32 bit timer(1MHz) in microseconds (BbRtCfg_t::schSetupDelayUs).
- #define BB_TIMER_8MHZ_MAX_VALUE_US 0x1FFFFFFF /∗ 2^29 - 1 = 0x1FFFFFFF. ∗/

  Maximum time tick for 32 bit timer(8MHz) in microseconds (BbRtCfg_t::schSetupDelayUs).
- #define BB_RTC_MAX_VALUE_US 511999999 /∗ 2^24 / 32768 ∗ 10^6 - 1 = 512 ∗ 10^6 - 1 = 511999999. ∗/

  Maximum time tick for 24 bit RTC counter(32768Hz) in microseconds. (BbRtCfg_t::BbTimerBoundaryUs)

Typedefs

- typedef void(∗ bbDrvIrqCback_t) (void)

  IRQ callback datatypes.

Enumerations

- enum PalBbProt_t {
  BB_PROT_NONE, BB_PROT_BLE, BB_PROT_BLE_DTM, BB_PROT_PRBS15,
  BB_PROT_15P4, BB_PROT_NUM }

  Protocol types.
- enum PalBbStat_c {
  BB_STATUS_SUCCESS, BB_STATUS_FAILED, BB_STATUS_CANCELED, BB_STATUS_RX_TIMEOUT,
  BB_STATUS_CRC_FAILED, BB_STATUS_FRAME_FAILED, BB_STATUS_ACK_FAILED, BB_STATUS_ACK_TIMEOUT,
  BB_STATUS_TX_CCA_FAILED, BB_STATUS_TX_FAILED }

  Status codes.
- enum PalBbPhy_t { BB_PHY_BLE_1M = 1, BB_PHY_BLE_2M = 2, BB_PHY_BLE_CODED = 3,
  BB_PHY_15P4 = 4 }

  PHY types.
- enum PalBbPhy_op { BB_PHY_OPTIONS_DEFAULT = 0, BB_PHY_OPTIONS_BLE_S2 = 1, BB_PHY_OPTIONS_BLE_S8 = 2 }

  PHY options.

Functions

- void PalBbInit (void)

  Initialize the baseband driver.
- void PalBbRestore (void)

  Restore the baseband driver.
- void PalBbEnable (void)

  Enable the BB hardware.
- void PalBbDisable (void)

  Disable the BB hardware.
- void PalBbLoadCfg (PalBbCfg_t ∗pCfg)

  Load BB timing configuration.
- uint32_t PalBbGetCurrentTime (void)

  Get the current BB clock value in microseconds.
- bool_t PalBbGetTimestamp (uint32_t ∗pTime)

  Get the current FRC time.
- void PalBbRegisterProtIrq (uint8_t protId, bbDrvIrqCback_t timerCback, bbDrvIrqCback_t radioCback)

  Called to register a protocol's Radio and Timer IRQ callback functions.
- void PalBbSetProtId (uint8_t protId)

  Set protocol ID.

4.11.1 Detailed Description

4.11.2 Enumeration Type Documentation

4.11.2.1 PalBbPhy_op

```
enum PalBbPhy_op
```

PHY options.

Enumerator

BB_PHY_OPTIONS_DEFAULT	BB defined PHY Options behavior.
BB_PHY_OPTIONS_BLE_S2	Always use S=2 coding when transmitting on LE Coded PHY.
BB_PHY_OPTIONS_BLE_S8	Always use S=8 coding when transmitting on LE Coded PHY.

4.11.2.2 PalBbPhy_t

```
enum PalBbPhy_t
```

PHY types.

Enumerator

BB_PHY_BLE_1M	Bluetooth Low Energy 1Mbps PHY.
BB_PHY_BLE_2M	Bluetooth Low Energy 2Mbps PHY.
BB_PHY_BLE_CODED	Bluetooth Low Energy Coded PHY (data coding unspecified).
BB_PHY_15P4	802.15.4 PHY.

4.11.2.3 PalBbProt_t

```
enum PalBbProt_t
```

Protocol types.

Enumerator

BB_PROT_NONE	Non-protocol specific operation.
BB_PROT_BLE	Bluetooth Low Energy normal mode.
BB_PROT_BLE_DTM	Bluetooth Low Energy direct test mode.
BB_PROT_PRBS15	Enable the continuous PRBS15 transmit sequence.
BB_PROT_15P4	802.15.4.
BB_PROT_NUM	Number of protocols.

4.11.2.4 PalBbStat_c

```
enum PalBbStat_c
```

Status codes.

Enumerator

BB_STATUS_SUCCESS	Operation successful.
BB_STATUS_FAILED	General failure.
BB_STATUS_CANCELED	Receive canceled.
BB_STATUS_RX_TIMEOUT	Receive packet timeout.
BB_STATUS_CRC_FAILED	Receive packet with CRC verification failed.
BB_STATUS_FRAME_FAILED	Receive packet with frame verification failed.
BB_STATUS_ACK_FAILED	ACK packet failure.
BB_STATUS_ACK_TIMEOUT	ACK packet timeout.
BB_STATUS_TX_CCA_FAILED	Transmit CCA failure.
BB_STATUS_TX_FAILED	Transmit failure.

4.11.3 Function Documentation

4.11.3.1 PalBbDisable()

```
void PalBbDisable (
             void  )
```

Disable the BB hardware.

This routine signals the BB hardware to go into low power (disable power and clocks) after all BB operations have been disabled.

4.11.3.2 PalBbEnable()

```
void PalBbEnable (
             void  )
```

Enable the BB hardware.

This routine brings the BB hardware out of low power (enable power and clocks) just before a first BB operation is executed.

4.11.3.3 PalBbGetCurrentTime()

```
uint32_t PalBbGetCurrentTime (
             void  )
```

Get the current BB clock value in microseconds.

Returns

Current BB clock value, units are microseconds.

This routine reads the current value from the BB clock and returns its value.

4.11.3.4 PalBbGetTimestamp()

```
bool_t PalBbGetTimestamp (
             uint32_t * pTime )
```

Get the current FRC time.

Parameters

pTime	Pointer to return the current time.

Returns

TRUE if time is valid, FALSE otherwise.

Get the current FRC time.

Note

FRC is limited to the same bit-width as the BB clock. Return value is available only when the BB is active.

4.11.3.5 PalBbInit()

```
void PalBbInit (
            void   )
```

Initialize the baseband driver.

One-time initialization of baseband resources. This routine can be used to setup baseband resources, load RF trim parameters and execute RF calibrations and seed the random number generator.

This routine should block until the BB hardware is completely initialized.

4.11.3.6 PalBbLoadCfg()

```
void PalBbLoadCfg (
            PalBbCfg_t * pCfg )
```

Load BB timing configuration.

Parameters

pCfg	Return configuration values.

4.11.3.7 PalBbRegisterProtIrq()

```
void PalBbRegisterProtIrq (
            uint8_t protId,
            bbDrvIrqCback_t timerCback,
            bbDrvIrqCback_t radioCback )
```

Called to register a protocol's Radio and Timer IRQ callback functions.

Parameters

protId	Protocol ID.
timerCback	Timer IRQ callback.
radioCback	Timer IRQ callback.

4.11.3.8 PalBbRestore()

```
void PalBbRestore (
            void  )
```

Restore the baseband driver.

This routine restores BB hardware state after deep sleep event.

4.11.3.9 PalBbSetProtId()

```
void PalBbSetProtId (
            uint8_t protId )
```

Set protocol ID.

Parameters

prot↔ Id	Protocol ID.

4.12 PAL_UART

Classes

- struct PalUartConfig_t

  Peripheral configuration.

Typedefs

- typedef void(∗ PalUartCompCback_t) (void)

  Completion callback.

Enumerations

- enum PalUartState_t { PAL_UART_STATE_UNINIT = 0, PAL_UART_STATE_ERROR = 0, PAL_UART_STATE_READY = 1, PAL_UART_STATE_BUSY = 2 }

  Operational states.
- enum PalUartId_t { PAL_UART_ID_USER = 0, PAL_UART_ID_CHCI = 1, PAL_UART_ID_TERMINAL = 2, PAL_UART_ID_MAX }

  Reserved UART ID.

Functions

- void **PalUartInit** (PalUartId_t id, const PalUartConfig_t ∗pCfg)
- void **PalUartDeInit** (PalUartId_t id)
- PalUartState_t **PalUartGetState** (PalUartId_t id)
- void **PalUartReadData** (PalUartId_t id, uint8_t ∗pData, uint16_t len)
- void **PalUartWriteData** (PalUartId_t id, const uint8_t ∗pData, uint16_t len)

4.12.1 Detailed Description

4.12.2 Enumeration Type Documentation

4.12.2.1 PalUartId_t

```
enum PalUartId_t
```

Reserved UART ID.

Enumerator

PAL_UART_ID_USER	UART 0.
PAL_UART_ID_CHCI	UART CHCI.
PAL_UART_ID_TERMINAL	UART TERMINAL.
PAL_UART_ID_MAX	Number of UART instances.

4.12.2.2 PalUartState_t

```
enum PalUartState_t
```

Operational states.

Enumerator

PAL_UART_STATE_UNINIT	Uninitialized state.
PAL_UART_STATE_ERROR	Error state.
PAL_UART_STATE_READY	Ready state.
PAL_UART_STATE_BUSY	Busy state.

Chapter 5

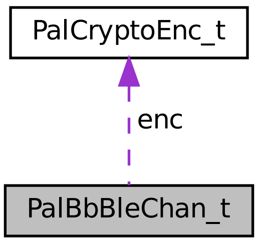
Class Documentation

5.1 PalBbBleChan_t Struct Reference

BLE channelization parameters.

```
#include <pal_bb_ble.h>
```

Collaboration diagram for PalBbBleChan_t:



Public Attributes

- uint8_t opType
- uint8_t chanIdx
- int8_t txPower
- uint32_t accAddr
- uint32_t crcInit
- uint8_t txPhy
- uint8_t rxPhy
- uint8_t initTxPhyOptions
- uint8_t tifsTxPhyOptions
- bool_t peerTxStableModIdx
- bool_t peerRxStableModIdx
- PalCryptoEnc_t enc

5.1.1 Detailed Description

BLE channelization parameters.

5.1.2 Member Data Documentation

5.1.2.1 accAddr

`uint32_t PalBbBleChan_t::accAddr`

Access address.

5.1.2.2 chanIdx

`uint8_t PalBbBleChan_t::chanIdx`

Channel index.

5.1.2.3 crcInit

`uint32_t PalBbBleChan_t::crcInit`

CRC initialization value.

5.1.2.4 enc

`PalCryptoEnc_t PalBbBleChan_t::enc`

Encryption parameters (NULL if disabled).

5.1.2.5 initTxPhyOptions

`uint8_t PalBbBleChan_t::initTxPhyOptions`

Initial Tx PHY options.

5.1.2.6 opType

`uint8_t PalBbBleChan_t::opType`

Operation type.

5.1.2.7 peerRxStableModIdx

```
bool_t PalBbBleChan_t::peerRxStableModIdx
```

Peer uses stable modulation index on receiver.

5.1.2.8 peerTxStableModIdx

```
bool_t PalBbBleChan_t::peerTxStableModIdx
```

Peer uses stable modulation index on transmitter.

5.1.2.9 rxPhy

```
uint8_t PalBbBleChan_t::rxPhy
```

Receiver PHY.

5.1.2.10 tifsTxPhyOptions

```
uint8_t PalBbBleChan_t::tifsTxPhyOptions
```

TIFS Tx PHY options.

5.1.2.11 txPhy

```
uint8_t PalBbBleChan_t::txPhy
```

Transmitter PHY.

5.1.2.12 txPower

```
int8_t PalBbBleChan_t::txPower
```

Active transmit power, unit is dBm.

The documentation for this struct was generated from the following file:

- /home/aw/msdk/Libraries/Cordio/platform/include/pal_bb_ble.h

5.2 PalBbBleDataParam_t Struct Reference

BLE data transfer parameters.

```
#include <pal_bb_ble.h>
```

Public Attributes

- PalBbBleTxIsr_t txCback
- PalBbBleRxIsr_t rxCback
- uint32_t dueUsec
- uint32_t rxTimeoutUsec

5.2.1 Detailed Description

BLE data transfer parameters.

5.2.2 Member Data Documentation

5.2.2.1 dueUsec

```
uint32_t PalBbBleDataParam_t::dueUsec
```

Due time of the first packet in microseconds.

5.2.2.2 rxCback

```
PalBbBleRxIsr_t PalBbBleDataParam_t::rxCback
```

Receive completion callback.

5.2.2.3 rxTimeoutUsec

```
uint32_t PalBbBleDataParam_t::rxTimeoutUsec
```

Receive timeout in microseconds.

5.2.2.4 txCback

```
PalBbBleTxIsr_t PalBbBleDataParam_t::txCback
```

Transmit completion callback.

The documentation for this struct was generated from the following file:

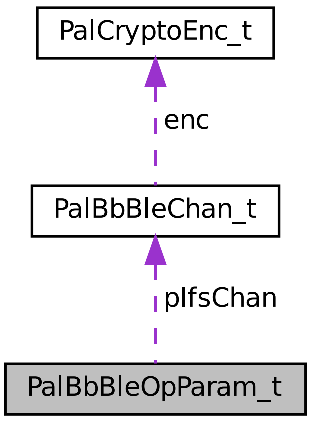
- /home/aw/msdk/Libraries/Cordio/platform/include/pal_bb_ble.h

5.3 PalBbBleOpParam_t Struct Reference

Operation parameters.

```
#include <pal_bb_ble.h>
```

Collaboration diagram for PalBbBleOpParam_t:



Public Attributes

- PalBbIfsMode_t ifsMode:8
- uint32_t ifsTime
- PalBbBleChan_t ∗ pIfsChan

5.3.1 Detailed Description

Operation parameters.

5.3.2 Member Data Documentation

5.3.2.1 ifsMode

```
PalBbIfsMode_t PalBbBleOpParam_t::ifsMode
```

IFS mode for next operation.

5.3.2.2 ifsTime

```
uint32_t PalBbBleOpParam_t::ifsTime
```

Absolute time of next PDU.

5.3.2.3 pIfsChan

```
PalBbBleChan_t* PalBbBleOpParam_t::pIfsChan
```

Channel of next PDU, NULL for no change.

The documentation for this struct was generated from the following file:

- /home/aw/msdk/Libraries/Cordio/platform/include/pal_bb_ble.h

5.4 PalBbBleTxBufDesc_t Struct Reference

Transmit buffer descriptor.

```
#include <pal_bb_ble.h>
```

Public Attributes

- uint16_t len
- uint8_t * pBuf

5.4.1 Detailed Description

Transmit buffer descriptor.

5.4.2 Member Data Documentation

5.4.2.1 len

```
uint16_t PalBbBleTxBufDesc_t::len
```

Length of buffer.

5.4.2.2 pBuf

```
uint8_t* PalBbBleTxBufDesc_t::pBuf
```

Pointer to buffer.

The documentation for this struct was generated from the following file:

- /home/aw/msdk/Libraries/Cordio/platform/include/pal_bb_ble.h

5.5 PalBbCfg_t Struct Reference

BB configuration.

```
#include <pal_bb.h>
```

Public Attributes

- uint16_t clkPpm
- uint8_t rfSetupDelayUsec
- uint16_t maxScanPeriodMsec
- uint16_t schSetupDelayUsec
- uint32_t BbTimerBoundaryUsec

5.5.1 Detailed Description

BB configuration.

5.5.2 Member Data Documentation

5.5.2.1 BbTimerBoundaryUsec

```
uint32_t PalBbCfg_t::BbTimerBoundaryUsec
```

BB timer boundary translated in microseconds before wraparound.

5.5.2.2 clkPpm

```
uint16_t PalBbCfg_t::clkPpm
```

Clock accuracy in PPM.

5.5.2.3 maxScanPeriodMsec

```
uint16_t PalBbCfg_t::maxScanPeriodMsec
```

Maximum scan period in milliseconds.

5.5.2.4 rfSetupDelayUsec

```
uint8_t PalBbCfg_t::rfSetupDelayUsec
```

RF setup delay in microseconds.

5.5.2.5 schSetupDelayUsec

```
uint16_t PalBbCfg_t::schSetupDelayUsec
```

Schedule setup delay in microseconds.

The documentation for this struct was generated from the following file:

- /home/aw/msdk/Libraries/Cordio/platform/include/pal_bb.h

5.6 PalCryptoEnc_t Struct Reference

Encryption data.

```
#include <pal_crypto.h>
```

Public Attributes

- uint8_t sk [PAL_CRYPTO_LL_KEY_LEN]
- uint8_t iv [PAL_CRYPTO_LL_IV_LEN]
- bool_t enaEncrypt
- bool_t enaDecrypt
- bool_t enaAuth
- uint8_t nonceMode
- uint16_t ∗ pEventCounter
- uint64_t ∗ pTxPktCounter
- uint64_t ∗ pRxPktCounter
- uint8_t dir
- uint8_t type
- void ∗ pEncryptCtx
- void ∗ pDecryptCtx

5.6.1 Detailed Description

Encryption data.

5.6.2 Member Data Documentation

5.6.2.1 dir

```
uint8_t PalCryptoEnc_t::dir
```

Direction value.

5.6.2.2 enaAuth

```
bool_t PalCryptoEnc_t::enaAuth
```

Enable authentication.

5.6.2.3 enaDecrypt

```
bool_t PalCryptoEnc_t::enaDecrypt
```

Rx/Decryption enabled flag.

5.6.2.4 enaEncrypt

```
bool_t PalCryptoEnc_t::enaEncrypt
```

Tx/Encryption enabled flag.

5.6.2.5 iv

```
uint8_t PalCryptoEnc_t::iv[PAL_CRYPTO_LL_IV_LEN]
```

Initialization vector.

5.6.2.6 nonceMode

```
uint8_t PalCryptoEnc_t::nonceMode
```

Nonce mode.

5.6.2.7 pDecryptCtx

```
void* PalCryptoEnc_t::pDecryptCtx
```

Rx/Decryption context.

5.6.2.8 pEncryptCtx

`void* PalCryptoEnc_t::pEncryptCtx`

Tx/Encryption context.

5.6.2.9 pEventCounter

`uint16_t* PalCryptoEnc_t::pEventCounter`

Connection event counter.

5.6.2.10 pRxPktCounter

`uint64_t* PalCryptoEnc_t::pRxPktCounter`

Rx packet counter. Set when nonceMode = PAL_BB_NONCE_MODE_EXT64_CNTR.

5.6.2.11 pTxPktCounter

`uint64_t* PalCryptoEnc_t::pTxPktCounter`

Tx packet counter. Set when nonceMode = PAL_BB_NONCE_MODE_EXT64_CNTR.

5.6.2.12 sk

`uint8_t PalCryptoEnc_t::sk[PAL_CRYPTO_LL_KEY_LEN]`

Session/Encryption key.

5.6.2.13 type

`uint8_t PalCryptoEnc_t::type`

Type, ACL, CIS, BIS

The documentation for this struct was generated from the following file:

- /home/aw/msdk/Libraries/Cordio/platform/include/pal_crypto.h

5.7 PalUartConfig_t Struct Reference

Peripheral configuration.

`#include <pal_uart.h>`

Public Attributes

- PalUartCompCback_t rdCback
- PalUartCompCback_t wrCback
- uint32_t baud
- bool_t hwFlow

5.7.1 Detailed Description

Peripheral configuration.

5.7.2 Member Data Documentation

5.7.2.1 baud

```
uint32_t PalUartConfig_t::baud
```

Baud rate.

5.7.2.2 hwFlow

```
bool_t PalUartConfig_t::hwFlow
```

Use HW Flow control

5.7.2.3 rdCback

```
PalUartCompCback_t PalUartConfig_t::rdCback
```

Read data completion callback.

5.7.2.4 wrCback

```
PalUartCompCback_t PalUartConfig_t::wrCback
```

Write data completion callback.

The documentation for this struct was generated from the following file:

- /home/aw/msdk/Libraries/Cordio/platform/include/pal_uart.h

Chapter 6

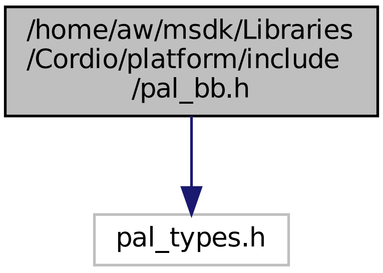
File Documentation

6.1 /home/aw/msdk/Libraries/Cordio/platform/include/pal_bb.h File Reference

Baseband interface file.

```
#include "pal_types.h"
```

Include dependency graph for pal_bb.h:



Classes

- struct PalBbCfg_t

  BB configuration.

Macros

- #define BB_CLK_RATE_HZ 1000000

  BB clock rate in hertz.
- #define BB_MATH_DIV_10E6(n) ((uint32_t)(((uint64_t)(n) $*$ UINT64_C(4295)) $>>$ 32))

Binary divide with 1,000,000 divisor (n[max]=0xFFFFFFFF).

- #define BB_US_TO_BB_TICKS(us) (us)

    Return microseconds (no conversion required).

- #define **RTC_CLOCK_RATE** 32768
- #define **USE_RTC_BB_CLK** (BB_CLK_RATE_HZ == RTC_CLOCK_RATE)
- #define BB_TICKS_TO_US(n) (n)

    BB ticks to microseconds (no conversion required).

- #define BB_MAX_SCAN_PERIOD_MS 1000

    Typical maximum duration to scan in a scan interval (BbRtCfg_t::maxScanPeriodMs).

- #define BB_RF_SETUP_DELAY_US 150

    Typical RF setup delay (BbRtCfg_t::rfSetupDelayUs).

- #define BB_SCH_SETUP_DELAY_US 500

    Typical operation setup delay in microseconds (BbRtCfg_t::schSetupDelayUs).

- #define BB_TIMER_1MHZ_MAX_VALUE_US 0xFFFFFFFF /∗ 2^32 - 1 = 0xFFFFFFFF. ∗/

    Maximum time tick for 32 bit timer(1MHz) in microseconds (BbRtCfg_t::schSetupDelayUs).

- #define BB_TIMER_8MHZ_MAX_VALUE_US 0x1FFFFFFF /∗ 2^29 - 1 = 0x1FFFFFFF. ∗/

    Maximum time tick for 32 bit timer(8MHz) in microseconds (BbRtCfg_t::schSetupDelayUs).

- #define BB_RTC_MAX_VALUE_US 511999999 /∗ 2^24 / 32768 ∗ 10^6 - 1 = 512 ∗ 10^6 - 1 = 511999999. ∗/

    Maximum time tick for 24 bit RTC counter(32768Hz) in microseconds. (BbRtCfg_t::BbTimerBoundaryUs)

Typedefs

- typedef void(∗ bbDrvIrqCback_t) (void)

    IRQ callback datatypes.

Enumerations

- enum PalBbProt_t {
  BB_PROT_NONE, BB_PROT_BLE, BB_PROT_BLE_DTM, BB_PROT_PRBS15,
  BB_PROT_15P4, BB_PROT_NUM }

    Protocol types.

- enum PalBbStat_c {
  BB_STATUS_SUCCESS, BB_STATUS_FAILED, BB_STATUS_CANCELED, BB_STATUS_RX_TIMEOUT,
  BB_STATUS_CRC_FAILED, BB_STATUS_FRAME_FAILED, BB_STATUS_ACK_FAILED, BB_STATUS_ACK_TIMEOUT,
  BB_STATUS_TX_CCA_FAILED, BB_STATUS_TX_FAILED }

    Status codes.

- enum PalBbPhy_t { BB_PHY_BLE_1M = 1, BB_PHY_BLE_2M = 2, BB_PHY_BLE_CODED = 3,
  BB_PHY_15P4 = 4 }

    PHY types.

- enum PalBbPhy_op { BB_PHY_OPTIONS_DEFAULT = 0, BB_PHY_OPTIONS_BLE_S2 = 1, BB_PHY_OPTIONS_BLE_S8 = 2 }

    PHY options.

Functions

- void PalBbInit (void)

  Initialize the baseband driver.
- void PalBbRestore (void)

  Restore the baseband driver.
- void PalBbEnable (void)

  Enable the BB hardware.
- void PalBbDisable (void)

  Disable the BB hardware.
- void PalBbLoadCfg (PalBbCfg_t ∗pCfg)

  Load BB timing configuration.
- uint32_t PalBbGetCurrentTime (void)

  Get the current BB clock value in microseconds.
- bool_t PalBbGetTimestamp (uint32_t ∗pTime)

  Get the current FRC time.
- void PalBbRegisterProtIrq (uint8_t protId, bbDrvIrqCback_t timerCback, bbDrvIrqCback_t radioCback)

  Called to register a protocol's Radio and Timer IRQ callback functions.
- void PalBbSetProtId (uint8_t protId)

  Set protocol ID.

6.1.1 Detailed Description

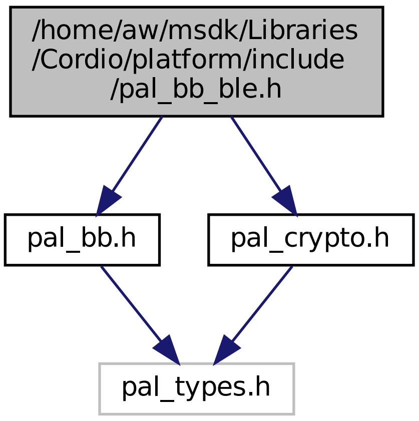
Baseband interface file.

6.2 /home/aw/msdk/Libraries/Cordio/platform/include/pal_bb_ble.h File Reference

BLE Baseband interface file.

```
#include "pal_bb.h"
#include "pal_crypto.h"
```

Include dependency graph for pal_bb_ble.h:



Classes

- struct PalBbBleChan_t

  BLE channelization parameters.
- struct PalBbBleDataParam_t

  BLE data transfer parameters.
- struct PalBbBleOpParam_t

  Operation parameters.
- struct PalBbBleTxBufDesc_t

  Transmit buffer descriptor.

Macros

- #define LL_ENABLE_TESTER 0

Typedefs

- typedef void(∗ PalBbBleTxIsr_t) (uint8_t status)

  Transmit complete ISR callback signature.
- typedef void(∗ PalBbBleRxIsr_t) (uint8_t status, int8_t rssi, uint32_t crc, uint32_t timestamp, uint8_t rxPhy↩ Options)

  Receive complete ISR callback signature.

Enumerations

- enum PalBbBleNonce_m { PAL_BB_NONCE_MODE_PKT_CNTR, PAL_BB_NONCE_MODE_EXT16_CNTR, PAL_BB_NONCE_MODE_EXT64_CNTR }

  Nonce modes.
- enum PalBbBleConn_t { PAL_BB_TYPE_ACL, PAL_BB_TYPE_CIS, PAL_BB_TYPE_BIS }

  Connection type.
- enum PalBbIfsMode_t { PAL_BB_IFS_MODE_CLR, PAL_BB_IFS_MODE_TOGGLE_TIFS, PAL_BB_IFS_MODE_SAME_ABS }

  IFS modes.

Functions

- void PalBbBleInit (void)

  Initialize the BLE baseband driver.
- void PalBbBleEnable (void)

  Enable the BB hardware.
- void PalBbBleDisable (void)

  Disable the BB hardware.
- void PalBbBleSetChannelParam (PalBbBleChan_t ∗pChan)

  Set channelization parameters.
- void PalBbBleSetDataParams (const PalBbBleDataParam_t ∗pParam)

  Set the data packet exchange parameters.
- void PalBbBleSetOpParams (const PalBbBleOpParam_t ∗pOpParam)

  Set the operation parameters.
- void PalBbBleTxData (PalBbBleTxBufDesc_t descs[ ], uint8_t cnt)

  Transmit a packet.
- void PalBbBleTxTifsData (PalBbBleTxBufDesc_t descs[ ], uint8_t cnt)

  Transmit packet at TIFS after the last packet received.
- void PalBbBleRxData (uint8_t ∗pBuf, uint16_t len)

  Receive packet.
- void PalBbBleRxTifsData (uint8_t ∗pBuf, uint16_t len)

  Receive packet at TIFS after the last packet transmitted.
- void PalBbBleCancelTifs (void)

  Cancel TIFS timer.
- void PalBbBleCancelData (void)

  Cancel a pending transmit or receive.
- void PalBbBleEnableDataWhitening (bool_t enable)

  Enable or disable data whitening.
- void PalBbBleEnablePrbs15 (bool_t enable)

  Enable or disable PRBS15.
- void PalBbBleInlineEncryptDecryptSetDirection (uint8_t dir)

  Set inline encryption/decryption direction bit.
- void PalBbBleInlineEncryptSetPacketCount (uint64_t count)

  Set the inline encryption packet count for transmit.
- void PalBbBleLowPower (void)

  Low power operation.

6.2.1 Detailed Description

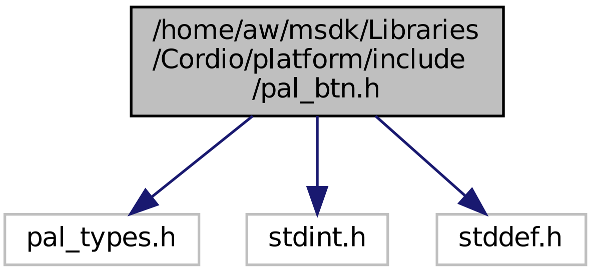
BLE Baseband interface file.

6.3 /home/aw/msdk/Libraries/Cordio/platform/include/pal_btn.h File Reference

Button driver definition.

```
#include "pal_types.h"
```

Include dependency graph for pal_btn.h:



Typedefs

- typedef void(∗ PalBtnActionCback_t) (uint8_t btnId, PalBtnPos_t state)

  Action callback signature.

Enumerations

- enum PalBtnState_t { PAL_BTN_STATE_UNINIT = 0, PAL_BTN_STATE_ERROR = 0, PAL_BTN_STATE_READY }

  Operational states.
- enum PalBtnPos_t { PAL_BTN_POS_INVALID, PAL_BTN_POS_DOWN, PAL_BTN_POS_UP }

  Button position.
- enum {
  PAL_BTN_AUDIO_PLAY = 0x80, PAL_BTN_AUDIO_FWD, PAL_BTN_AUDIO_RWD, PAL_BTN_AUDIO_VOL_UP,
  PAL_BTN_AUDIO_VOL_DN, PAL_BTN_AUDIO_MUTE }

  Audio button assignments (only mapped in audio applications).

Functions

- void **PalBtnInit** (PalBtnActionCback_t actCback)
- void **PalBtnDeInit** (void)
- PalBtnState_t **PalBtnGetState** (void)
- PalBtnPos_t **PalBtnGetPosition** (uint8_t id)

6.3.1 Detailed Description

Button driver definition.

6.3.2 Enumeration Type Documentation

6.3.2.1 anonymous enum

```
anonymous enum
```

Audio button assignments (only mapped in audio applications).

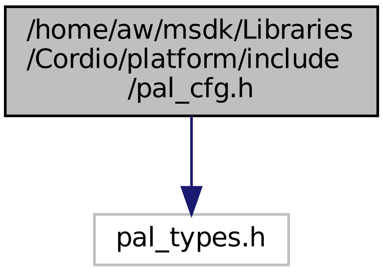
Enumerator

PAL_BTN_AUDIO_PLAY	Play button.
PAL_BTN_AUDIO_FWD	Fast Forward button.
PAL_BTN_AUDIO_RWD	Fast Rewind button.
PAL_BTN_AUDIO_VOL_UP	Volume Up button.
PAL_BTN_AUDIO_VOL_DN	Volume Down button.
PAL_BTN_AUDIO_MUTE	Mute button.

6.4 /home/aw/msdk/Libraries/Cordio/platform/include/pal_cfg.h File Reference

System configuration definition.

`#include "pal_types.h"`
Include dependency graph for pal_cfg.h:



Enumerations

- enum PalCfgId_t {
  PAL_CFG_ID_BD_ADDR, PAL_CFG_ID_BLE_PHY, PAL_CFG_ID_LL_PARAM, PAL_CFG_ID_MAC_ADDR,
  PAL_CFG_ID_UUID }

  Configuration ID.

Functions

- void **PalCfgLoadData** (uint8_t cfgId, uint8_t ∗pBuf, uint32_t len)
- void **PalCfgSetDeviceUuid** (uint8_t ∗pBuf)

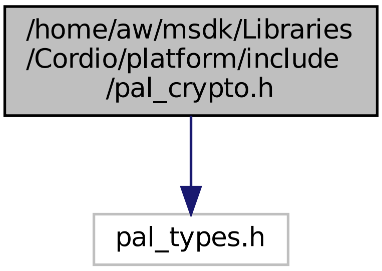
6.4.1 Detailed Description

System configuration definition.

6.5 /home/aw/msdk/Libraries/Cordio/platform/include/pal_crypto.h File Reference

Crypto driver definition.

```
#include "pal_types.h"
```
Include dependency graph for pal_crypto.h:



Classes

- struct PalCryptoEnc_t

    Encryption data.

Macros

- #define PAL_CRYPTO_AES_BLOCK_SIZE 16

    AES block size.
- #define PAL_CRYPTO_LL_KEY_LEN 16
- #define PAL_CRYPTO_LL_IV_LEN 8
- #define PAL_CRYPTO_LL_DATA_MIC_LEN 4
- #define SEC_CCM_KEY_LEN 16

    CCM-Mode algorithm lengths.
- #define SEC_CCM_MAX_ADDITIONAL_LEN ((1<<16) - (1<<8))

    CCM-Mode algorithm maximum additional length.
- #define SEC_CCM_L 2

    CCM-Mode algorithm length.
- #define SEC_CCM_NONCE_LEN (15-SEC_CCM_L)

    CCM-Mode algorithm nonce length.

Enumerations

- enum PalCryptoState_t { PAL_CRYPTO_STATE_UNINIT = 0, PAL_CRYPTO_STATE_ERROR = 0, PAL_CRYPTO_STATE_READY }

    Operational states.

Functions

- void **PalCryptoInit** (void)
- void **PalCryptoDeInit** (void)
- void **PalCryptoGenerateP256KeyPair** (const uint8_t ∗pPrivKey, uint8_t ∗pPubKey)
- void **PalCryptoGenerateDhKey** (const uint8_t ∗pPubKey, const uint8_t ∗pPrivKey, uint8_t ∗pDhKey)
- bool_t **PalCryptoValidatePublicKey** (const uint8_t ∗pPubKey, bool_t generateKey)
- void **PalCryptoGenerateRandomNumber** (uint8_t ∗pBuf, uint8_t len)
- uint32_t **PalCryptoCcmDec** (const uint8_t ∗pKey, uint8_t ∗pNonce, uint8_t ∗pCypherText, uint16_t text↩Len, uint8_t ∗pClear, uint16_t clearLen, uint8_t ∗pMic, uint8_t micLen, uint8_t ∗pResult, uint8_t handlerId, uint16_t param, uint8_t event)
- void **PalCryptoCcmEnc** (const uint8_t ∗pKey, uint8_t ∗pNonce, uint8_t ∗pPlainText, uint16_t textLen, uint8↩_t ∗pClear, uint16_t clearLen, uint8_t micLen, uint8_t ∗pResult, uint8_t handlerId, uint16_t param, uint8_t event)
- void **PalCryptoAesEcb** (const uint8_t ∗pKey, uint8_t ∗pOut, const uint8_t ∗pIn)
- void **PalCryptoAesCmac** (const uint8_t ∗pKey, uint8_t ∗pOut, const uint8_t ∗pIn, uint16_t len)
- void **PalCryptoAesEnable** (PalCryptoEnc_t ∗pEnc, uint8_t id, uint8_t localDir)
- bool_t **PalCryptoAesCcmEncrypt** (PalCryptoEnc_t ∗pEnc, uint8_t ∗pHdr, uint8_t ∗pBuf, uint8_t ∗pMic)
- bool_t **PalCryptoAesCcmDecrypt** (PalCryptoEnc_t ∗pEnc, uint8_t ∗pBuf)
- void **PalCryptoSetEncryptPacketCount** (PalCryptoEnc_t ∗pEnc, uint64_t pktCnt)
- void **PalCryptoSetDecryptPacketCount** (PalCryptoEnc_t ∗pEnc, uint64_t pktCnt)

6.5.1 Detailed Description

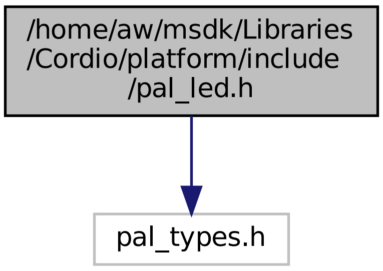
Crypto driver definition.

6.6 /home/aw/msdk/Libraries/Cordio/platform/include/pal_led.h File Reference

LED driver definition.

```
#include "pal_types.h"
```

Include dependency graph for pal_led.h:



Enumerations

- enum PalLedReserved_id { PAL_LED_ID_CPU_ACTIVE = 0x30, PAL_LED_ID_ERROR = 0x31 }

  Reserved LED IDs.

Functions

- void **PalLedInit** (void)
- void **PalLedDeInit** (void)
- void **PalLedOn** (uint8_t id)
- void **PalLedOff** (uint8_t id)

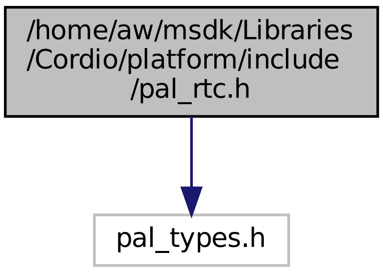
6.6.1 Detailed Description

LED driver definition.

6.7 /home/aw/msdk/Libraries/Cordio/platform/include/pal_rtc.h File Reference

RTC timer interface file.

`#include "pal_types.h"`
Include dependency graph for pal_rtc.h:



Macros

- #define PAL_MAX_RTC_COUNTER_VAL (0x00FFFFFF)

  Max value of RTC.
- #define PAL_RTC_TICKS_PER_SEC (32768) /∗ RTC ticks per second (with prescaler) ∗/

  Clock frequency of the RTC timer used.

Typedefs

- typedef void(∗ palRtcIrqCback_t) (void)

  Platform RTC callback.

Enumerations

- enum PalRtcState_t { PAL_RTC_STATE_UNINIT = 0, PAL_RTC_STATE_ERROR = 0, PAL_RTC_STATE_READY = 1 }

  Operational states.

Functions

- void **PalRtcInit** (void)
- void **PalRtcEnableCompareIrq** (uint8_t channelId)
- void **PalRtcDisableCompareIrq** (uint8_t channelId)
- uint32_t **PalRtcCounterGet** (void)
- void **PalRtcCompareSet** (uint8_t channelId, uint32_t value)
- PalRtcState_t **PalRtcGetState** (void)

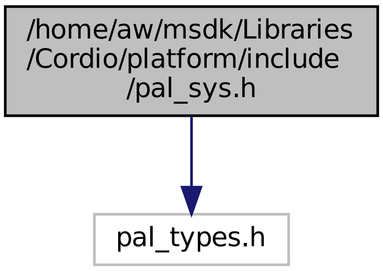
6.7.1 Detailed Description

RTC timer interface file.

6.8 /home/aw/msdk/Libraries/Cordio/platform/include/pal_sys.h File Reference

System hooks.

```
#include "pal_types.h"
```
Include dependency graph for pal_sys.h:



Macros

- #define PAL_SYS_ASSERT(expr)

    Parameter check (disabled).

Functions

- void **PalSysInit** (void)
- void **PalSysAssertTrap** (void)
- void **PalSysSetTrap** (bool_t enable)
- uint32_t **PalSysGetAssertCount** (void)
- uint32_t **PalSysGetStackUsage** (void)
- void **PalSysSleep** (void)
- bool_t **PalSysIsBusy** (void)
- void **PalSysSetBusy** (void)
- void **PalSysSetIdle** (void)
- void **PalEnterCs** (void)
- void **PalExitCs** (void)

6.8.1 Detailed Description

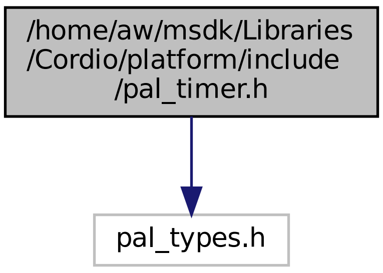
System hooks.

6.9 /home/aw/msdk/Libraries/Cordio/platform/include/pal_timer.h File Reference

Timer interface file.

```
#include "pal_types.h"
```

Include dependency graph for pal_timer.h:



Typedefs

- typedef void(∗ PalTimerCompCback_t) (void)

    Completion callback.

Enumerations

- enum PalTimerState_t { PAL_TIMER_STATE_UNINIT = 0, PAL_TIMER_STATE_ERROR = 0, PAL_TIMER_STATE_READY, PAL_TIMER_STATE_BUSY }

    Operational states.

Functions

- void **PalTimerInit** (PalTimerCompCback_t expCback)
- void **PalTimerDeInit** (void)
- PalTimerState_t **PalTimerGetState** (void)
- void **PalTimerStart** (uint32_t expUsec)
- void **PalTimerStop** (void)
- uint32_t **PalTimerGetCurrentTime** (void)
- uint32_t **PalTimerGetExpTime** (void)
- void **PalTimerSleep** (uint32_t expUsec)
- void **PalTimerRestore** (uint32_t schTime)
- void **PalTimerSetIRQPriority** (uint32_t priority)

6.9.1 Detailed Description

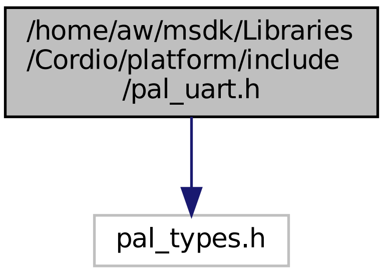
Timer interface file.

6.10 /home/aw/msdk/Libraries/Cordio/platform/include/pal_uart.h File Reference

UART driver definition.

```
#include "pal_types.h"
```

Include dependency graph for pal_uart.h:



Classes

- struct PalUartConfig_t

  Peripheral configuration.

Typedefs

- typedef void(∗ PalUartCompCback_t) (void)

  Completion callback.

Enumerations

- enum PalUartState_t { PAL_UART_STATE_UNINIT = 0, PAL_UART_STATE_ERROR = 0, PAL_UART_STATE_READY = 1, PAL_UART_STATE_BUSY = 2 }

  Operational states.
- enum PalUartId_t { PAL_UART_ID_USER = 0, PAL_UART_ID_CHCI = 1, PAL_UART_ID_TERMINAL = 2, PAL_UART_ID_MAX }

  Reserved UART ID.

Functions

- void **PalUartInit** (PalUartId_t id, const PalUartConfig_t ∗pCfg)
- void **PalUartDeInit** (PalUartId_t id)
- PalUartState_t **PalUartGetState** (PalUartId_t id)
- void **PalUartReadData** (PalUartId_t id, uint8_t ∗pData, uint16_t len)
- void **PalUartWriteData** (PalUartId_t id, const uint8_t ∗pData, uint16_t len)

6.10.1 Detailed Description

UART driver definition.

Index

/home/aw/msdk/Libraries/Cordio/platform/include/pal_bb.h, 45
/home/aw/msdk/Libraries/Cordio/platform/include/pal_bb_ble.h, 47
/home/aw/msdk/Libraries/Cordio/platform/include/pal_btn.h, 50
/home/aw/msdk/Libraries/Cordio/platform/include/pal_cfg.h, 52
/home/aw/msdk/Libraries/Cordio/platform/include/pal_crypto.h, 53
/home/aw/msdk/Libraries/Cordio/platform/include/pal_led.h, 54
/home/aw/msdk/Libraries/Cordio/platform/include/pal_rtc.h, 56
/home/aw/msdk/Libraries/Cordio/platform/include/pal_sys.h, 57
/home/aw/msdk/Libraries/Cordio/platform/include/pal_timer.h, 58
/home/aw/msdk/Libraries/Cordio/platform/include/pal_uart.h, 60

accAddr
    PalBbBleChan_t, 34

baud
    PalUartConfig_t, 43
BB_PHY_15P4
    PAL_BB, 27
BB_PHY_BLE_1M
    PAL_BB, 27
BB_PHY_BLE_2M
    PAL_BB, 27
BB_PHY_BLE_CODED
    PAL_BB, 27
BB_PHY_OPTIONS_BLE_S2
    PAL_BB, 26
BB_PHY_OPTIONS_BLE_S8
    PAL_BB, 26
BB_PHY_OPTIONS_DEFAULT
    PAL_BB, 26
BB_PROT_15P4
    PAL_BB, 27
BB_PROT_BLE
    PAL_BB, 27
BB_PROT_BLE_DTM
    PAL_BB, 27
BB_PROT_NONE
    PAL_BB, 27
BB_PROT_NUM
    PAL_BB, 27
BB_PROT_PRBS15
    PAL_BB, 27
BB_STATUS_ACK_FAILED
    PAL_BB, 27
BB_STATUS_ACK_TIMEOUT
    PAL_BB, 27
BB_STATUS_CANCELED
    PAL_BB, 27
BB_STATUS_CRC_FAILED
    PAL_BB, 27
BB_STATUS_FAILED
    PAL_BB, 27
BB_STATUS_FRAME_FAILED
    PAL_BB, 27
BB_STATUS_RX_TIMEOUT
    PAL_BB, 27
BB_STATUS_SUCCESS
    PAL_BB, 27
BB_STATUS_TX_CCA_FAILED
    PAL_BB, 27
BB_STATUS_TX_FAILED
    PAL_BB, 27
BbTimerBoundaryUsec
    PalBbCfg_t, 39

chanIdx
    PalBbBleChan_t, 34
clkPpm
    PalBbCfg_t, 39
crcInit
    PalBbBleChan_t, 34

dir
    PalCryptoEnc_t, 41
dueUsec
    PalBbBleDataParam_t, 36

enaAuth
    PalCryptoEnc_t, 41
enaDecrypt
    PalCryptoEnc_t, 41
enaEncrypt
    PalCryptoEnc_t, 41
enc
    PalBbBleChan_t, 34

hwFlow
    PalUartConfig_t, 43

ifsMode
    PalBbBleOpParam_t, 37

ifsTime
PalBbBleOpParam_t, 37
initTxPhyOptions
PalBbBleChan_t, 34
iv
PalCryptoEnc_t, 41

len
PalBbBleTxBufDesc_t, 38
LL_ENABLE_TESTER
PAL_BB_BLE, 14

maxScanPeriodMsec
PalBbCfg_t, 39

nonceMode
PalCryptoEnc_t, 41

opType
PalBbBleChan_t, 34

PAL_BB, 25
BB_PHY_15P4, 27
BB_PHY_BLE_1M, 27
BB_PHY_BLE_2M, 27
BB_PHY_BLE_CODED, 27
BB_PHY_OPTIONS_BLE_S2, 26
BB_PHY_OPTIONS_BLE_S8, 26
BB_PHY_OPTIONS_DEFAULT, 26
BB_PROT_15P4, 27
BB_PROT_BLE, 27
BB_PROT_BLE_DTM, 27
BB_PROT_NONE, 27
BB_PROT_NUM, 27
BB_PROT_PRBS15, 27
BB_STATUS_ACK_FAILED, 27
BB_STATUS_ACK_TIMEOUT, 27
BB_STATUS_CANCELED, 27
BB_STATUS_CRC_FAILED, 27
BB_STATUS_FAILED, 27
BB_STATUS_FRAME_FAILED, 27
BB_STATUS_RX_TIMEOUT, 27
BB_STATUS_SUCCESS, 27
BB_STATUS_TX_CCA_FAILED, 27
BB_STATUS_TX_FAILED, 27
PalBbDisable, 28
PalBbEnable, 28
PalBbGetCurrentTime, 28
PalBbGetTimestamp, 28
PalBbInit, 29
PalBbLoadCfg, 29
PalBbPhy_op, 26
PalBbPhy_t, 26
PalBbProt_t, 27
PalBbRegisterProtIrq, 29
PalBbRestore, 30
PalBbSetProtId, 30
PalBbStat_c, 27
PAL_BB_BLE, 13
LL_ENABLE_TESTER, 14
PAL_BB_IFS_MODE_CLR, 15
PAL_BB_IFS_MODE_SAME_ABS, 15
PAL_BB_IFS_MODE_TOGGLE_TIFS, 15
PAL_BB_NONCE_MODE_EXT16_CNTR, 14
PAL_BB_NONCE_MODE_EXT64_CNTR, 14
PAL_BB_NONCE_MODE_PKT_CNTR, 14
PAL_BB_TYPE_ACL, 14
PAL_BB_TYPE_BIS, 14
PAL_BB_TYPE_CIS, 14
PalBbBleConn_t, 14
PalBbBleDisable, 15
PalBbBleEnable, 15
PalBbBleInit, 15
PalBbBleNonce_m, 14
PalBbIfsMode_t, 14
PAL_BB_BLE_CHAN, 16
PalBbBleSetChannelParam, 16
PAL_BB_BLE_DATA, 17
PalBbBleCancelData, 17
PalBbBleCancelTifs, 17
PalBbBleEnableDataWhitening, 18
PalBbBleEnablePrbs15, 18
PalBbBleInlineEncryptDecryptSetDirection, 18
PalBbBleInlineEncryptSetPacketCount, 19
PalBbBleLowPower, 19
PalBbBleRxData, 19
PalBbBleRxTifsData, 19
PalBbBleSetDataParams, 20
PalBbBleSetOpParams, 20
PalBbBleTxData, 20
PalBbBleTxTifsData, 21
PAL_BB_IFS_MODE_CLR
PAL_BB_BLE, 15
PAL_BB_IFS_MODE_SAME_ABS
PAL_BB_BLE, 15
PAL_BB_IFS_MODE_TOGGLE_TIFS
PAL_BB_BLE, 15
PAL_BB_NONCE_MODE_EXT16_CNTR
PAL_BB_BLE, 14
PAL_BB_NONCE_MODE_EXT64_CNTR
PAL_BB_BLE, 14
PAL_BB_NONCE_MODE_PKT_CNTR
PAL_BB_BLE, 14
PAL_BB_TYPE_ACL
PAL_BB_BLE, 14
PAL_BB_TYPE_BIS
PAL_BB_BLE, 14
PAL_BB_TYPE_CIS
PAL_BB_BLE, 14
pal_btn.h
PAL_BTN_AUDIO_FWD, 51
PAL_BTN_AUDIO_MUTE, 51
PAL_BTN_AUDIO_PLAY, 51
PAL_BTN_AUDIO_RWD, 51
PAL_BTN_AUDIO_VOL_DN, 51
PAL_BTN_AUDIO_VOL_UP, 51
PAL_BTN_AUDIO_FWD

pal_btn.h, 51
PAL_BTN_AUDIO_MUTE
pal_btn.h, 51
PAL_BTN_AUDIO_PLAY
pal_btn.h, 51
PAL_BTN_AUDIO_RWD
pal_btn.h, 51
PAL_BTN_AUDIO_VOL_DN
pal_btn.h, 51
PAL_BTN_AUDIO_VOL_UP
pal_btn.h, 51
PAL_BTN_POS_DOWN
PAL_BUTTON, 8
PAL_BTN_POS_INVALID
PAL_BUTTON, 8
PAL_BTN_POS_UP
PAL_BUTTON, 8
PAL_BTN_STATE_ERROR
PAL_BUTTON, 8
PAL_BTN_STATE_READY
PAL_BUTTON, 8
PAL_BTN_STATE_UNINIT
PAL_BUTTON, 8
PAL_BUTTON, 7
PAL_BTN_POS_DOWN, 8
PAL_BTN_POS_INVALID, 8
PAL_BTN_POS_UP, 8
PAL_BTN_STATE_ERROR, 8
PAL_BTN_STATE_READY, 8
PAL_BTN_STATE_UNINIT, 8
PalBtnPos_t, 7
PalBtnState_t, 8
PAL_CFG, 9
PAL_CFG_ID_BD_ADDR, 9
PAL_CFG_ID_BLE_PHY, 9
PAL_CFG_ID_LL_PARAM, 9
PAL_CFG_ID_MAC_ADDR, 9
PAL_CFG_ID_UUID, 9
PalCfgId_t, 9
PAL_CFG_ID_BD_ADDR
PAL_CFG, 9
PAL_CFG_ID_BLE_PHY
PAL_CFG, 9
PAL_CFG_ID_LL_PARAM
PAL_CFG, 9
PAL_CFG_ID_MAC_ADDR
PAL_CFG, 9
PAL_CFG_ID_UUID
PAL_CFG, 9
PAL_CRYPTO, 11
PAL_CRYPTO_LL_DATA_MIC_LEN, 12
PAL_CRYPTO_LL_IV_LEN, 12
PAL_CRYPTO_LL_KEY_LEN, 12
PAL_CRYPTO_STATE_ERROR, 12
PAL_CRYPTO_STATE_READY, 12
PAL_CRYPTO_STATE_UNINIT, 12
PalCryptoState_t, 12
PAL_CRYPTO_LL_DATA_MIC_LEN
PAL_CRYPTO, 12
PAL_CRYPTO_LL_IV_LEN
PAL_CRYPTO, 12
PAL_CRYPTO_LL_KEY_LEN
PAL_CRYPTO, 12
PAL_CRYPTO_STATE_ERROR
PAL_CRYPTO, 12
PAL_CRYPTO_STATE_READY
PAL_CRYPTO, 12
PAL_CRYPTO_STATE_UNINIT
PAL_CRYPTO, 12
PAL_LED, 23
PAL_LED_ID_CPU_ACTIVE, 23
PAL_LED_ID_ERROR, 23
PalLedReserved_id, 23
PAL_LED_ID_CPU_ACTIVE
PAL_LED, 23
PAL_LED_ID_ERROR
PAL_LED, 23
PAL_RTC, 24
PAL_RTC_STATE_ERROR, 24
PAL_RTC_STATE_READY, 24
PAL_RTC_STATE_UNINIT, 24
PalRtcState_t, 24
PAL_RTC_STATE_ERROR
PAL_RTC, 24
PAL_RTC_STATE_READY
PAL_RTC, 24
PAL_RTC_STATE_UNINIT
PAL_RTC, 24
PAL_SYS, 10
PAL_TIMER, 22
PAL_TIMER_STATE_BUSY, 22
PAL_TIMER_STATE_ERROR, 22
PAL_TIMER_STATE_READY, 22
PAL_TIMER_STATE_UNINIT, 22
PalTimerState_t, 22
PAL_TIMER_STATE_BUSY
PAL_TIMER, 22
PAL_TIMER_STATE_ERROR
PAL_TIMER, 22
PAL_TIMER_STATE_READY
PAL_TIMER, 22
PAL_TIMER_STATE_UNINIT
PAL_TIMER, 22
PAL_UART, 31
PAL_UART_ID_CHCI, 31
PAL_UART_ID_MAX, 31
PAL_UART_ID_TERMINAL, 31
PAL_UART_ID_USER, 31
PAL_UART_STATE_BUSY, 32
PAL_UART_STATE_ERROR, 32
PAL_UART_STATE_READY, 32
PAL_UART_STATE_UNINIT, 32
PalUartId_t, 31
PalUartState_t, 32
PAL_UART_ID_CHCI
PAL_UART, 31

PAL_UART_ID_MAX
PAL_UART, 31
PAL_UART_ID_TERMINAL
PAL_UART, 31
PAL_UART_ID_USER
PAL_UART, 31
PAL_UART_STATE_BUSY
PAL_UART, 32
PAL_UART_STATE_ERROR
PAL_UART, 32
PAL_UART_STATE_READY
PAL_UART, 32
PAL_UART_STATE_UNINIT
PAL_UART, 32
PalBbBleCancelData
PAL_BB_BLE_DATA, 17
PalBbBleCancelTifs
PAL_BB_BLE_DATA, 17
PalBbBleChan_t, 33
accAddr, 34
chanIdx, 34
crcInit, 34
enc, 34
initTxPhyOptions, 34
opType, 34
peerRxStableModIdx, 34
peerTxStableModIdx, 35
rxPhy, 35
tifsTxPhyOptions, 35
txPhy, 35
txPower, 35
PalBbBleConn_t
PAL_BB_BLE, 14
PalBbBleDataParam_t, 35
dueUsec, 36
rxCback, 36
rxTimeoutUsec, 36
txCback, 36
PalBbBleDisable
PAL_BB_BLE, 15
PalBbBleEnable
PAL_BB_BLE, 15
PalBbBleEnableDataWhitening
PAL_BB_BLE_DATA, 18
PalBbBleEnablePrbs15
PAL_BB_BLE_DATA, 18
PalBbBleInit
PAL_BB_BLE, 15
PalBbBleInlineEncryptDecryptSetDirection
PAL_BB_BLE_DATA, 18
PalBbBleInlineEncryptSetPacketCount
PAL_BB_BLE_DATA, 19
PalBbBleLowPower
PAL_BB_BLE_DATA, 19
PalBbBleNonce_m
PAL_BB_BLE, 14
PalBbBleOpParam_t, 37
ifsMode, 37
ifsTime, 37
pIfsChan, 38
PalBbBleRxData
PAL_BB_BLE_DATA, 19
PalBbBleRxTifsData
PAL_BB_BLE_DATA, 19
PalBbBleSetChannelParam
PAL_BB_BLE_CHAN, 16
PalBbBleSetDataParams
PAL_BB_BLE_DATA, 20
PalBbBleSetOpParams
PAL_BB_BLE_DATA, 20
PalBbBleTxBufDesc_t, 38
len, 38
pBuf, 38
PalBbBleTxData
PAL_BB_BLE_DATA, 20
PalBbBleTxTifsData
PAL_BB_BLE_DATA, 21
PalBbCfg_t, 39
BbTimerBoundaryUsec, 39
clkPpm, 39
maxScanPeriodMsec, 39
rfSetupDelayUsec, 40
schSetupDelayUsec, 40
PalBbDisable
PAL_BB, 28
PalBbEnable
PAL_BB, 28
PalBbGetCurrentTime
PAL_BB, 28
PalBbGetTimestamp
PAL_BB, 28
PalBbIfsMode_t
PAL_BB_BLE, 14
PalBbInit
PAL_BB, 29
PalBbLoadCfg
PAL_BB, 29
PalBbPhy_op
PAL_BB, 26
PalBbPhy_t
PAL_BB, 26
PalBbProt_t
PAL_BB, 27
PalBbRegisterProtIrq
PAL_BB, 29
PalBbRestore
PAL_BB, 30
PalBbSetProtId
PAL_BB, 30
PalBbStat_c
PAL_BB, 27
PalBtnPos_t
PAL_BUTTON, 7
PalBtnState_t
PAL_BUTTON, 8
PalCfgId_t

PAL_CFG, 9
PalCryptoEnc_t, 40
dir, 41
enaAuth, 41
enaDecrypt, 41
enaEncrypt, 41
iv, 41
nonceMode, 41
pDecryptCtx, 41
pEncryptCtx, 41
pEventCounter, 42
pRxPktCounter, 42
pTxPktCounter, 42
sk, 42
type, 42
PalCryptoState_t
PAL_CRYPTO, 12
PalLedReserved_id
PAL_LED, 23
PalRtcState_t
PAL_RTC, 24
PalTimerState_t
PAL_TIMER, 22
PalUartConfig_t, 42
baud, 43
hwFlow, 43
rdCback, 43
wrCback, 43
PalUartId_t
PAL_UART, 31
PalUartState_t
PAL_UART, 32
pBuf
PalBbBleTxBufDesc_t, 38
pDecryptCtx
PalCryptoEnc_t, 41
peerRxStableModIdx
PalBbBleChan_t, 34
peerTxStableModIdx
PalBbBleChan_t, 35
pEncryptCtx
PalCryptoEnc_t, 41
pEventCounter
PalCryptoEnc_t, 42
pIfsChan
PalBbBleOpParam_t, 38
pRxPktCounter
PalCryptoEnc_t, 42
pTxPktCounter
PalCryptoEnc_t, 42

rdCback
PalUartConfig_t, 43
rfSetupDelayUsec
PalBbCfg_t, 40
rxCback
PalBbBleDataParam_t, 36
rxPhy
PalBbBleChan_t, 35
rxTimeoutUsec
PalBbBleDataParam_t, 36

schSetupDelayUsec
PalBbCfg_t, 40
sk
PalCryptoEnc_t, 42

tifsTxPhyOptions
PalBbBleChan_t, 35
txCback
PalBbBleDataParam_t, 36
txPhy
PalBbBleChan_t, 35
txPower
PalBbBleChan_t, 35
type
PalCryptoEnc_t, 42

wrCback
PalUartConfig_t, 43