# MAX32655 Cordio Platform

Generated by Doxygen 1.9.5

# Chapter 1

# Module Index

## 1.1 Modules

Here is a list of all modules:

# Chapter 2

# Class Index

## 2.1 Class List

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

# Chapter 3

# File Index

## 3.1 File List

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

# 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.*
- 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)

### 4.1.1 Detailed Description

### 4.1.2 Enumeration Type Documentation

#### 4.1.2.1 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. |

#### 4.1.2.2 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. |

#### 4.1.2.3 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_CODEC

### Classes

- struct AudioStdCodecInfo_t

  *Standard codec info block.*
- struct AudioVsCodecInfo_t

  *VS codec info block.*
- struct PalCodecSreamParam_t

  *Codec.*

### Typedefs

- typedef void(∗ **PalCodecDataReady_t**) (uint16_t id)

  *Buffer available call signature.*

### Enumerations

- enum PalAudioDir_t { PAL_CODEC_DIR_INPUT = 0 , PAL_CODEC_DIR_OUTPUT = 1 }

  *Audio data path direction.*
- enum PalAudioChan_t { PAL_CODEC_CH_LEFT = 0 , PAL_CODEC_CH_RIGHT = 1 , AUDIO_NUM_CH }

  *Audio Channel.*
- enum { **PAL_CODEC_CH_LEFT_BIT** = (1 << PAL_CODEC_CH_LEFT) , **PAL_CODEC_CH_RIGHT_BIT** = (1 << PAL_CODEC_CH_RIGHT) }

  *Audio Channel mask.*

### Functions

- void **PalCodecReadLocalSupportedCodecs** (uint8_t ∗pNumStd, AudioStdCodecInfo_t stdCodecs[ ], uint8_t ∗pNumVs, AudioVsCodecInfo_t vsCodecs[ ])
- bool_t **PalCodecReadLocalSupportedCodecCapabilities** (uint8_t codingFmt, uint16_t compId, uint16_t vsCodecId, PalAudioDir_t dir)
- bool_t **PalCodecReadLocalSupportedControllerDelay** (uint8_t codingFmt, uint16_t compId, uint16_t vs↩ CodecId, PalAudioDir_t dir, uint32_t ∗pMinDly, uint32_t ∗pMaxDly)
- bool_t **PalCodecConfigureDataPath** (PalAudioDir_t dir, uint8_t dataPathId)
- void **PalCodecAmpInit** (void)
- uint8_t **PalCodecAmpGetVol** (void)
- void **PalCodecAmpVolumeUp** (void)
- void **PalCodecAmpVolumeDown** (void)
- void **PalCodecAmpMute** (void)
- void **PalCodecAmpUnmute** (void)
- void **PalCodecDataInit** (void)
- bool_t **PalCodecDataStartStream** (uint16_t id, PalCodecSreamParam_t ∗pParam)
- void **PalCodecDataStopStream** (uint16_t id)
- uint16_t **PalCodecDataStreamIn** (uint16_t id, uint8_t ∗pBuf, uint16_t len, uint32_t ∗pPktCtr)
- void **PalCodecDataStreamOut** (uint16_t id, const uint8_t ∗pBuf, uint16_t len, uint32_t pktCtr)

### 4.4.1 Detailed Description

### 4.4.2 Enumeration Type Documentation

#### 4.4.2.1 PalAudioChan_t

enum PalAudioChan_t

Audio Channel.

**Enumerator**

| | |
|---|---|
| PAL_CODEC_CH_LEFT | Left channel. |
| PAL_CODEC_CH_RIGHT | Right channel. |
| AUDIO_NUM_CH | Right channel. |

#### 4.4.2.2 PalAudioDir_t

enum PalAudioDir_t

Audio data path direction.

**Enumerator**

| | |
|---|---|
| PAL_CODEC_DIR_INPUT | Input data path. |
| PAL_CODEC_DIR_OUTPUT | Output data path. |

## 4.5 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 textLen, 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)

### 4.5.1 Detailed Description

### 4.5.2 Macro Definition Documentation

#### 4.5.2.1 PAL_CRYPTO_LL_DATA_MIC_LEN

```
#define PAL_CRYPTO_LL_DATA_MIC_LEN 4
```

Data channel PDU MIC length.

#### 4.5.2.2 PAL_CRYPTO_LL_IV_LEN

```
#define PAL_CRYPTO_LL_IV_LEN 8
```

Initialization vector length.

#### 4.5.2.3 PAL_CRYPTO_LL_KEY_LEN

```
#define PAL_CRYPTO_LL_KEY_LEN 16
```

Encryption key length.

### 4.5.3 Enumeration Type Documentation

#### 4.5.3.1 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. |

## 4.6 PAL_BB_BLE_CHAN

### Classes

- struct PalBbBleChan_t

  *BLE channelization parameters.*

### 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

### Classes

- struct PalBbBleDataParam_t
  *BLE data transfer parameters.*
- struct PalBbBleOpParam_t
  *Operation parameters.*
- struct PalBbBleTxBufDesc_t
  *Transmit buffer descriptor.*

### 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 rx↩ PhyOptions)
  *Receive complete ISR callback signature.*

### Enumerations

- enum PalBbIfsMode_t { PAL_BB_IFS_MODE_CLR , PAL_BB_IFS_MODE_TOGGLE_TIFS , PAL_BB_IFS_MODE_SAME_ABS }
  *IFS modes.*

## 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.*

### 4.7.1 Detailed Description

This section contains driver routines used for packet transmission.

### 4.7.2 Enumeration Type Documentation

#### 4.7.2.1 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. |

### 4.7.3 Function Documentation

#### 4.7.3.1 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.

#### 4.7.3.2 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.

#### 4.7.3.3 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.

#### 4.7.3.4 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.

#### 4.7.3.5 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.

#### 4.7.3.6 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.

#### 4.7.3.7 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.

#### 4.7.3.8 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_BB_BLE_INIT

### Functions

- void PalBbBleInit (void)
  *Initialize the BLE baseband driver.*
- void PalBbBleEnable (void)
  *Enable the BB hardware.*
- void PalBbBleDisable (void)
  *Disable the BB hardware.*

### 4.8.1 Detailed Description

This section contains driver routines which initialize as well as enable the BLE mode of the BB hardware.

### 4.8.2 Function Documentation

#### 4.8.2.1 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.

#### 4.8.2.2 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.

#### 4.8.2.3 PalBbBleInit()

```
void PalBbBleInit (
             void  )
```

Initialize the BLE baseband driver.

One-time initialization of BLE baseband driver.

# 4.9 PAL_BB_BLE_TEST

## Functions

- 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.*

## 4.9.1 Detailed Description

This section contains driver routines used for test modes.

## 4.9.2 Function Documentation

### 4.9.2.1 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.

### 4.9.2.2 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.

#### 4.9.2.3 PalBbBleInlineEncryptDecryptSetDirection()

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

Set inline encryption/decryption direction bit.

**Parameters**

| | |
|---|---|
| *dir* | 0=slave, non-zero=master |

#### 4.9.2.4 PalBbBleInlineEncryptSetPacketCount()

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

Set the inline encryption packet count for transmit.

**Parameters**

| | |
|---|---|
| *count* | Packet counter value, a 39-bit value |

#### 4.9.2.5 PalBbBleLowPower()

```
void PalBbBleLowPower (
            void  )
```

Low power operation.

**Note**

Called by upper baseband code.

## 4.10 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)

## 4.10.1 Detailed Description

## 4.10.2 Enumeration Type Documentation

#### 4.10.2.1 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. |

# 4.11 PAL_LED

### Enumerations

- enum { 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)

### 4.11.1 Detailed Description

### 4.11.2 Enumeration Type Documentation

#### 4.11.2.1 anonymous enum

`anonymous enum`

Reserved LED IDs.

**Enumerator**

| | |
|---|---|
| PAL_LED_ID_CPU_ACTIVE | CPU active LED ID. |
| PAL_LED_ID_ERROR | Error LED ID. |

## 4.12 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)

### 4.12.1 Detailed Description

### 4.12.2 Enumeration Type Documentation

#### 4.12.2.1 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. |

## 4.13 PAL_BB_INIT

### 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.*

### 4.13.1 Detailed Description

This section contains driver routines which initialize as well as enable the sleep mode of the BB hardware.

### 4.13.2 Function Documentation

#### 4.13.2.1 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.

#### 4.13.2.2 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.

#### 4.13.2.3 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.

#### 4.13.2.4 PalBbLoadCfg()

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

Load BB timing configuration.

**Parameters**

| | |
|---|---|
| *pCfg* | Return configuration values. |

#### 4.13.2.5 PalBbRestore()

```
void PalBbRestore (
            void  )
```

Restore the baseband driver.

This routine restores BB hardware state after deep sleep event.

## 4.14 PAL_BB_CLOCK

### Functions

- 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.*

### 4.14.1 Detailed Description

This section contains driver routines related to the BB clock.

### 4.14.2 Function Documentation

#### 4.14.2.1 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.

#### 4.14.2.2 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.

#### 4.14.2.3 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. |

#### 4.14.2.4 PalBbSetProtId()

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

Set protocol ID.

**Parameters**

| | |
|---|---|
| *protId* | Protocol ID. |

# 4.15 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)

### 4.15.1 Detailed Description

### 4.15.2 Enumeration Type Documentation

#### 4.15.2.1 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. |

#### 4.15.2.2 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 AudioStdCodecInfo_t Struct Reference

Standard codec info block.

`#include <pal_codec.h>`

### Public Attributes

- uint8_t codecId

### 5.1.1 Detailed Description

Standard codec info block.

### 5.1.2 Member Data Documentation

#### 5.1.2.1 codecId

`uint8_t AudioStdCodecInfo_t::codecId`

Codec ID.

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

- /github/workspace/Libraries/Cordio/platform/include/pal_codec.h

## 5.2 AudioVsCodecInfo_t Struct Reference

VS codec info block.

`#include <pal_codec.h>`

### Public Attributes

- uint16_t compId
- uint16_t codecId

### 5.2.1 Detailed Description

VS codec info block.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 codecId

`uint16_t AudioVsCodecInfo_t::codecId`

Codec ID.

#### 5.2.2.2 compId

`uint16_t AudioVsCodecInfo_t::compId`

Company ID.

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

- /github/workspace/Libraries/Cordio/platform/include/pal_codec.h

## 5.3 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

### 5.3.1 Detailed Description

BLE channelization parameters.

### 5.3.2 Member Data Documentation

#### 5.3.2.1 accAddr

```
uint32_t PalBbBleChan_t::accAddr
```

Access address.

#### 5.3.2.2 chanIdx

```
uint8_t PalBbBleChan_t::chanIdx
```

Channel index.

#### 5.3.2.3 crcInit

```
uint32_t PalBbBleChan_t::crcInit
```

CRC initialization value.

#### 5.3.2.4 enc

```
PalCryptoEnc_t PalBbBleChan_t::enc
```

Encryption parameters (NULL if disabled).

#### 5.3.2.5 initTxPhyOptions

```
uint8_t PalBbBleChan_t::initTxPhyOptions
```

Initial Tx PHY options.

#### 5.3.2.6 opType

```
uint8_t PalBbBleChan_t::opType
```

Operation type.

#### 5.3.2.7 peerRxStableModIdx

```
bool_t PalBbBleChan_t::peerRxStableModIdx
```

Peer uses stable modulation index on receiver.

#### 5.3.2.8 peerTxStableModIdx

```
bool_t PalBbBleChan_t::peerTxStableModIdx
```

Peer uses stable modulation index on transmitter.

#### 5.3.2.9 rxPhy

```
uint8_t PalBbBleChan_t::rxPhy
```

Receiver PHY.

#### 5.3.2.10 tifsTxPhyOptions

```
uint8_t PalBbBleChan_t::tifsTxPhyOptions
```

TIFS Tx PHY options.

#### 5.3.2.11 txPhy

```
uint8_t PalBbBleChan_t::txPhy
```

Transmitter PHY.

#### 5.3.2.12 txPower

```
int8_t PalBbBleChan_t::txPower
```

Active transmit power, unit is dBm.

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

- /github/workspace/Libraries/Cordio/platform/include/pal_bb_ble.h

## 5.4 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

### 5.4.1 Detailed Description

BLE data transfer parameters.

### 5.4.2 Member Data Documentation

#### 5.4.2.1 dueUsec

```
uint32_t PalBbBleDataParam_t::dueUsec
```

Due time of the first packet in microseconds.

#### 5.4.2.2 rxCback

```
PalBbBleRxIsr_t PalBbBleDataParam_t::rxCback
```

Receive completion callback.

#### 5.4.2.3 rxTimeoutUsec

```
uint32_t PalBbBleDataParam_t::rxTimeoutUsec
```

Receive timeout in microseconds.

#### 5.4.2.4 txCback

```
PalBbBleTxIsr_t PalBbBleDataParam_t::txCback
```

Transmit completion callback.

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

- /github/workspace/Libraries/Cordio/platform/include/pal_bb_ble.h

## 5.5 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

### 5.5.1 Detailed Description

Operation parameters.

### 5.5.2 Member Data Documentation

#### 5.5.2.1 ifsMode

```
PalBbIfsMode_t PalBbBleOpParam_t::ifsMode
```

IFS mode for next operation.

#### 5.5.2.2 ifsTime

```
uint32_t PalBbBleOpParam_t::ifsTime
```

Absolute time of next PDU.

#### 5.5.2.3 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:

- /github/workspace/Libraries/Cordio/platform/include/pal_bb_ble.h

## 5.6 PalBbBleTxBufDesc_t Struct Reference

Transmit buffer descriptor.

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

### Public Attributes

- uint16_t len
- uint8_t * pBuf

### 5.6.1 Detailed Description

Transmit buffer descriptor.

### 5.6.2 Member Data Documentation

#### 5.6.2.1 len

```
uint16_t PalBbBleTxBufDesc_t::len
```

Length of buffer.

#### 5.6.2.2 pBuf

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

Pointer to buffer.

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

- /github/workspace/Libraries/Cordio/platform/include/pal_bb_ble.h

## 5.7 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

### 5.7.1 Detailed Description

BB configuration.

### 5.7.2 Member Data Documentation

#### 5.7.2.1 BbTimerBoundaryUsec

```
uint32_t PalBbCfg_t::BbTimerBoundaryUsec
```

BB timer boundary translated in microseconds before wraparound.

#### 5.7.2.2 clkPpm

```
uint16_t PalBbCfg_t::clkPpm
```

Clock accuracy in PPM.

#### 5.7.2.3 maxScanPeriodMsec

`uint16_t PalBbCfg_t::maxScanPeriodMsec`

Maximum scan period in milliseconds.

#### 5.7.2.4 rfSetupDelayUsec

`uint8_t PalBbCfg_t::rfSetupDelayUsec`

RF setup delay in microseconds.

#### 5.7.2.5 schSetupDelayUsec

`uint16_t PalBbCfg_t::schSetupDelayUsec`

Schedule setup delay in microseconds.

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

- /github/workspace/Libraries/Cordio/platform/include/pal_bb.h

## 5.8 PalCodecSreamParam_t Struct Reference

Codec.

`#include <pal_codec.h>`

### Public Attributes

- PalAudioDir_t dir
- uint16_t chMask
- uint32_t intervalUsec
- uint32_t pktCtr
- PalCodecDataReady_t rdyCback

### 5.8.1 Detailed Description

Codec.

### 5.8.2 Member Data Documentation

#### 5.8.2.1 chMask

```
uint16_t PalCodecSreamParam_t::chMask
```

Audio channel mask.

#### 5.8.2.2 dir

```
PalAudioDir_t PalCodecSreamParam_t::dir
```

Stream data direction.

#### 5.8.2.3 intervalUsec

```
uint32_t PalCodecSreamParam_t::intervalUsec
```

SDU interval in microseconds.

#### 5.8.2.4 pktCtr

```
uint32_t PalCodecSreamParam_t::pktCtr
```

Initial packet counter value.

#### 5.8.2.5 rdyCback

```
PalCodecDataReady_t PalCodecSreamParam_t::rdyCback
```

Data ready callback.

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

- /github/workspace/Libraries/Cordio/platform/include/pal_codec.h

## 5.9 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

### 5.9.1 Detailed Description

Encryption data.

### 5.9.2 Member Data Documentation

#### 5.9.2.1 dir

```
uint8_t PalCryptoEnc_t::dir
```

Direction value.

#### 5.9.2.2 enaAuth

```
bool_t PalCryptoEnc_t::enaAuth
```

Enable authentication.

#### 5.9.2.3 enaDecrypt

```
bool_t PalCryptoEnc_t::enaDecrypt
```

Rx/Decryption enabled flag.

#### 5.9.2.4 enaEncrypt

```
bool_t PalCryptoEnc_t::enaEncrypt
```

Tx/Encryption enabled flag.

#### 5.9.2.5 iv

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

Initialization vector.

#### 5.9.2.6 nonceMode

```
uint8_t PalCryptoEnc_t::nonceMode
```

Nonce mode.

#### 5.9.2.7 pDecryptCtx

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

Rx/Decryption context.

#### 5.9.2.8 pEncryptCtx

```
void* PalCryptoEnc_t::pEncryptCtx
```

Tx/Encryption context.

#### 5.9.2.9 pEventCounter

```
uint16_t* PalCryptoEnc_t::pEventCounter
```

Connection event counter.

#### 5.9.2.10 pRxPktCounter

```
uint64_t* PalCryptoEnc_t::pRxPktCounter
```

Rx packet counter. Set when nonceMode = PAL_BB_NONCE_MODE_EXT64_CNTR.

#### 5.9.2.11 pTxPktCounter

```
uint64_t* PalCryptoEnc_t::pTxPktCounter
```

Tx packet counter. Set when nonceMode = PAL_BB_NONCE_MODE_EXT64_CNTR.

#### 5.9.2.12 sk

```
uint8_t PalCryptoEnc_t::sk[PAL_CRYPTO_LL_KEY_LEN]
```

Session/Encryption key.

#### 5.9.2.13 type

```
uint8_t PalCryptoEnc_t::type
```

Type, ACL, CIS, BIS

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

- /github/workspace/Libraries/Cordio/platform/include/pal_crypto.h

## 5.10 PalUartConfig_t Struct Reference

Peripheral configuration.

```
#include <pal_uart.h>
```

### Public Attributes

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

### 5.10.1 Detailed Description

Peripheral configuration.

### 5.10.2 Member Data Documentation

#### 5.10.2.1 baud

```
uint32_t PalUartConfig_t::baud
```

Baud rate.

#### 5.10.2.2 hwFlow

```
bool_t PalUartConfig_t::hwFlow
```

Use HW Flow control

#### 5.10.2.3 rdCback

```
PalUartCompCback_t PalUartConfig_t::rdCback
```

Read data completion callback.

#### 5.10.2.4 wrCback

```
PalUartCompCback_t PalUartConfig_t::wrCback
```

Write data completion callback.

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

- /github/workspace/Libraries/Cordio/platform/include/pal_uart.h

# Chapter 6

# File Documentation

## 6.1 /github/workspace/Libraries/Cordio/platform/include/pal_bb.h File Reference

Baseband interface file.

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

Include dependency graph for pal_bb.h:

## 6.2 pal_bb.h

Go to the documentation of this file.

```
1 /*************************************************************************************************/
23 /*************************************************************************************************/
24
25 #ifndef PAL_BB_H
26 #define PAL_BB_H
27
28 #include "pal_types.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
34 /*************************************************************************************************
35   Macros
36 *************************************************************************************************/
37
39 typedef enum
40 {
41   BB_PROT_NONE,
42   BB_PROT_BLE,
43   BB_PROT_BLE_DTM,
44   BB_PROT_PRBS15,
45   BB_PROT_15P4,
46   BB_PROT_NUM
47 } PalBbProt_t;
48
50 enum
51 {
52   BB_STATUS_SUCCESS,
53   BB_STATUS_FAILED,
54   BB_STATUS_CANCELED,
55   BB_STATUS_RX_TIMEOUT,
56   BB_STATUS_CRC_FAILED,
57   BB_STATUS_FRAME_FAILED,
58   BB_STATUS_ACK_FAILED,
59   BB_STATUS_ACK_TIMEOUT,
60   BB_STATUS_TX_CCA_FAILED,
61   BB_STATUS_TX_FAILED
```

```
62 };
63
65 typedef enum
66 {
67   BB_PHY_BLE_1M     = 1,
68   BB_PHY_BLE_2M     = 2,
69   BB_PHY_BLE_CODED  = 3,
70   BB_PHY_15P4       = 4,
71 } PalBbPhy_t;
72
74 enum
75 {
76   BB_PHY_OPTIONS_DEFAULT          = 0,
77   BB_PHY_OPTIONS_BLE_S2           = 1,
78   BB_PHY_OPTIONS_BLE_S8           = 2
79 };
80
81 #ifndef BB_CLK_RATE_HZ
83 #define BB_CLK_RATE_HZ              1000000
84 #endif
85
87 #define BB_MATH_DIV_10E6(n)         ((uint32_t)(((uint64_t)(n) * UINT64_C(4295)) » 32))
88
89 #if (BB_CLK_RATE_HZ == 1000000)
91 #define BB_US_TO_BB_TICKS(us)       (us)
92 #elif (BB_CLK_RATE_HZ == 8000000)
94 #define BB_US_TO_BB_TICKS(us)       ((uint32_t)((us) « 3))
95 #elif (BB_CLK_RATE_HZ == 32768)
97 #define BB_US_TO_BB_TICKS(us)       ((uint32_t)(((uint64_t)(us) * (uint64_t)(70368745)) » 31))   /*
      calculated value may be one tick low */
98 #else
100 #define BB_US_TO_BB_TICKS(us)       BB_MATH_DIV_10E6((uint64_t)(us) * (uint64_t)(BB_CLK_RATE_HZ))
101 #endif
102
103 #define RTC_CLOCK_RATE              32768
104 #define USE_RTC_BB_CLK              (BB_CLK_RATE_HZ == RTC_CLOCK_RATE)
105
106 #if (BB_CLK_RATE_HZ == 1000000)
108 #define BB_TICKS_TO_US(n)           (n)
109 #elif (BB_CLK_RATE_HZ == 8000000)
111 #define BB_TICKS_TO_US(n)           ((n) » 3)
112 #elif (BB_CLK_RATE_HZ == 32768)
114 #define BB_TICKS_TO_US(n)           (uint32_t)(((uint64_t)(n) * 15625) » 9)
115 #else
117 #define BB_TICKS_TO_US(n)           (uint32_t)((uint64_t)(n) * 1000000 / BB_CLK_RATE_HZ)
118 #endif
119
121 #define BB_MAX_SCAN_PERIOD_MS       1000
122
124 #define BB_RF_SETUP_DELAY_US        150
125
127 #define BB_SCH_SETUP_DELAY_US       500
128
130 #define BB_TIMER_1MHZ_MAX_VALUE_US  0xFFFFFFFF  /* 2^32 - 1 = 0xFFFFFFFF. */
131
133 #define BB_TIMER_8MHZ_MAX_VALUE_US  0x1FFFFFFF  /* 2^29 - 1 = 0x1FFFFFFF. */
134
136 #define BB_RTC_MAX_VALUE_US         511999999   /* 2^24 / 32768 * 10^6 - 1 = 512 * 10^6 - 1 = 511999999.
      */
137
138 /**************************************************************************************************
139   Type Definitions
140 **************************************************************************************************/
141
143 typedef void (*bbDrvIrqCback_t)(void);
144
146 typedef struct
147 {
148   uint16_t clkPpm;
149   uint8_t  rfSetupDelayUsec;
150   uint16_t maxScanPeriodMsec;
151   uint16_t schSetupDelayUsec;
152   uint32_t BbTimerBoundaryUsec;
153 } PalBbCfg_t;
154
155 /**************************************************************************************************
156   Function Declarations
157 **************************************************************************************************/
158
165 /*************************************************************************************************/
175 /*************************************************************************************************/
176 void PalBbInit(void);
177
178 /*************************************************************************************************/
184 /*************************************************************************************************/
185 void PalBbRestore(void);
186
```

```
187 /*************************************************************************************************/
194 /*************************************************************************************************/
195 void PalBbEnable(void);
196
197 /*************************************************************************************************/
204 /*************************************************************************************************/
205 void PalBbDisable(void);
206
207 /*************************************************************************************************/
213 /*************************************************************************************************/
214 void PalBbLoadCfg(PalBbCfg_t *pCfg);
215     /* PAL_BB_INIT */
217
223 /*************************************************************************************************/
231 /*************************************************************************************************/
232 uint32_t PalBbGetCurrentTime(void);
233
234 /*************************************************************************************************/
247 /*************************************************************************************************/
248 bool_t PalBbGetTimestamp(uint32_t *pTime);
249
250 /*************************************************************************************************/
258 /*************************************************************************************************/
259 void PalBbRegisterProtIrq(uint8_t protId, bbDrvIrqCback_t timerCback, bbDrvIrqCback_t radioCback);
260
261 /*************************************************************************************************/
267 /*************************************************************************************************/
268 void PalBbSetProtId(uint8_t protId);
269     /* PAL_BB_CLOCK */
271
272 #ifdef __cplusplus
273 };
274 #endif
275
276 #endif /* PAL_BB_H */
```

## 6.3 /github/workspace/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 { PAL_BB_NONCE_MODE_PKT_CNTR , PAL_BB_NONCE_MODE_EXT16_CNTR , PAL_BB_NONCE_MODE_EXT64_ }

  *Nonce modes.*
- enum { 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.*

### 6.3.1 Detailed Description

BLE Baseband interface file.



### 6.3.2 Macro Definition Documentation

#### 6.3.2.1 LL_ENABLE_TESTER

```
#define LL_ENABLE_TESTER 0
```

Enable LL tester extensions.

### 6.3.3 Enumeration Type Documentation

#### 6.3.3.1 anonymous enum

```
anonymous enum
```

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. |

#### 6.3.3.2 anonymous enum

anonymous enum

Connection type.

**Enumerator**

| | |
|---|---|
| PAL_BB_TYPE_ACL | ACL. |
| PAL_BB_TYPE_CIS | CIS. |
| PAL_BB_TYPE_BIS | BIS. |

## 6.4 pal_bb_ble.h

Go to the documentation of this file.

```
1 /*************************************************************************************************/
23 /*************************************************************************************************/
24
25 #ifndef PAL_BB_BLE_H
26 #define PAL_BB_BLE_H
27
28 #include "pal_bb.h"
29 #include "pal_crypto.h"
30
31 #ifdef __cplusplus
32 extern "C" {
33 #endif
34
35 /*************************************************************************************************
36   Macros
37 *************************************************************************************************/
38
39 #ifndef LL_ENABLE_TESTER
40 #define LL_ENABLE_TESTER        0
41 #endif
42
43 /*************************************************************************************************
44   Data Types
45 *************************************************************************************************/
46
48 enum
49 {
50   PAL_BB_NONCE_MODE_PKT_CNTR,
51   PAL_BB_NONCE_MODE_EXT16_CNTR,
52   PAL_BB_NONCE_MODE_EXT64_CNTR
53 };
54
56 enum
57 {
58   PAL_BB_TYPE_ACL,
59   PAL_BB_TYPE_CIS,
60   PAL_BB_TYPE_BIS
61 };
62
67 typedef struct
68 {
69   uint8_t       opType;
70   uint8_t       chanIdx;
71   int8_t        txPower;
72   uint32_t      accAddr;
73   uint32_t      crcInit;
74   uint8_t       txPhy;
75   uint8_t       rxPhy;
76   uint8_t       initTxPhyOptions;
77   uint8_t       tifsTxPhyOptions;
78   bool_t        peerTxStableModIdx;
79   bool_t        peerRxStableModIdx;
80   PalCryptoEnc_t enc;
82 #if (LL_ENABLE_TESTER)
83   uint32_t      accAddrRx;
84   uint32_t      accAddrTx;
85   uint32_t      crcInitRx;
86   uint32_t      crcInitTx;
```

```
87   int8_t        txPwrOffset;
88 #endif
89 } PalBbBleChan_t;
90     /* PAL_BB_BLE_CHAN */
92
99 typedef void (*PalBbBleTxIsr_t)(uint8_t status);
100
102 typedef void (*PalBbBleRxIsr_t)(uint8_t status, int8_t rssi, uint32_t crc, uint32_t timestamp, uint8_t
      rxPhyOptions);
103
105 typedef enum
106 {
107   PAL_BB_IFS_MODE_CLR,
108   PAL_BB_IFS_MODE_TOGGLE_TIFS,
109   PAL_BB_IFS_MODE_SAME_ABS
110 } PalBbIfsMode_t;
111
113 typedef struct
114 {
115   PalBbBleTxIsr_t   txCback;
116   PalBbBleRxIsr_t   rxCback;
118   uint32_t          dueUsec;
119   uint32_t          rxTimeoutUsec;
120 } PalBbBleDataParam_t;
121
123 typedef struct
124 {
125   PalBbIfsMode_t    ifsMode:8;
126   uint32_t          ifsTime;
127   PalBbBleChan_t    *pIfsChan;
128 } PalBbBleOpParam_t;
129
131 typedef struct
132 {
133   uint16_t          len;
134   uint8_t           *pBuf;
135 } PalBbBleTxBufDesc_t;
136     /* PAL_BB_BLE_DATA */
138
139 /*************************************************************************************************
140   Function Declarations
141 *************************************************************************************************/
142
149 /*************************************************************************************************/
155 /*************************************************************************************************/
156 void PalBbBleInit(void);
157
158 /*************************************************************************************************/
165 /*************************************************************************************************/
166 void PalBbBleEnable(void);
167
168 /*************************************************************************************************/
175 /*************************************************************************************************/
176 void PalBbBleDisable(void);
177     /* PAL_BB_BLE_INIT */
179
185 /*************************************************************************************************/
202 /*************************************************************************************************/
203 void PalBbBleSetChannelParam(PalBbBleChan_t *pChan);
204     /* PAL_BB_BLE_CHAN */
206
212 /*************************************************************************************************/
221 /*************************************************************************************************/
222 void PalBbBleSetDataParams(const PalBbBleDataParam_t *pParam);
223
224 /*************************************************************************************************/
232 /*************************************************************************************************/
233 void PalBbBleSetOpParams(const PalBbBleOpParam_t *pOpParam);
234
235 /*************************************************************************************************/
244 /*************************************************************************************************/
245 void PalBbBleTxData(PalBbBleTxBufDesc_t descs[], uint8_t cnt);
246
247 /*************************************************************************************************/
257 /*************************************************************************************************/
258 void PalBbBleTxTifsData(PalBbBleTxBufDesc_t descs[], uint8_t cnt);
259
260 /*************************************************************************************************/
269 /*************************************************************************************************/
270 void PalBbBleRxData(uint8_t *pBuf, uint16_t len);
271
272 /*************************************************************************************************/
282 /*************************************************************************************************/
283 void PalBbBleRxTifsData(uint8_t *pBuf, uint16_t len);
284
285 /*************************************************************************************************/
292 /*************************************************************************************************/
```

```
293 void PalBbBleCancelTifs(void);
294
295 /*************************************************************************************************/
302 /*************************************************************************************************/
303 void PalBbBleCancelData(void);
304     /* PAL_BB_BLE_DATA */
306
312 /*************************************************************************************************/
320 /*************************************************************************************************/
321 void PalBbBleEnableDataWhitening(bool_t enable);
322
323 /*************************************************************************************************/
334 /*************************************************************************************************/
335 void PalBbBleEnablePrbs15(bool_t enable);
336
337 /*************************************************************************************************/
344 /*************************************************************************************************/
345 void PalBbBleInlineEncryptDecryptSetDirection(uint8_t dir);
346
347 /*************************************************************************************************/
354 /*************************************************************************************************/
355 void PalBbBleInlineEncryptSetPacketCount(uint64_t count);
356
357 /*************************************************************************************************/
363 /*************************************************************************************************/
364 void PalBbBleLowPower(void);
365     /* PAL_BB_BLE_TEST */
367
368 #ifdef __cplusplus
369 };
370 #endif
371
372 #endif /* PAL_BB_BLE_H */
```

## 6.5 /github/workspace/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)

### 6.5.1 Detailed Description

Button driver definition.



## 6.6 pal_btn.h

Go to the documentation of this file.

```
1 /*************************************************************************************************/
23 /*************************************************************************************************/
24
25 #ifndef PAL_BTN_H
26 #define PAL_BTN_H
27
28 #include "pal_types.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
37 /*************************************************************************************************
38   Data Types
39 *************************************************************************************************/
40
42 typedef enum
43 {
44   PAL_BTN_STATE_UNINIT = 0,
45   PAL_BTN_STATE_ERROR  = 0,
46   PAL_BTN_STATE_READY
47 } PalBtnState_t;
48
50 typedef enum
51 {
52   PAL_BTN_POS_INVALID,
53   PAL_BTN_POS_DOWN,
54   PAL_BTN_POS_UP
55 } PalBtnPos_t;
56
58 typedef void (*PalBtnActionCback_t)(uint8_t btnId, PalBtnPos_t state);
59
61 enum
62 {
63   PAL_BTN_AUDIO_PLAY = 0x80,
64   PAL_BTN_AUDIO_FWD,
65   PAL_BTN_AUDIO_RWD,
66   PAL_BTN_AUDIO_VOL_UP,
```

```
67   PAL_BTN_AUDIO_VOL_DN,
68   PAL_BTN_AUDIO_MUTE
69 };
70
71 /**************************************************************************************************
72   Function Declarations
73 **************************************************************************************************/
74
75 /* Initialization */
76 void PalBtnInit(PalBtnActionCback_t actCback);
77 void PalBtnDeInit(void);
78
79 /* Control and Status */
80 PalBtnState_t PalBtnGetState(void);
81 PalBtnPos_t PalBtnGetPosition(uint8_t id);
82     /* PAL_BUTTON */
84
85 #ifdef __cplusplus
86 };
87 #endif
88
89 #endif /* PAL_BTN_H */
```

## 6.7 /github/workspace/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)

### 6.7.1 Detailed Description

System configuration definition.

## 6.8 pal_cfg.h

Go to the documentation of this file.

```
1  /*************************************************************************************************/
23 /*************************************************************************************************/
24
25 #ifndef PAL_CFG_H
26 #define PAL_CFG_H
27
28 #include "pal_types.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
37 /*************************************************************************************************
38   Data Types
39 *************************************************************************************************/
40
42 typedef enum
43 {
44   PAL_CFG_ID_BD_ADDR,
45   PAL_CFG_ID_BLE_PHY,
46   PAL_CFG_ID_LL_PARAM,
47   PAL_CFG_ID_MAC_ADDR,
48   PAL_CFG_ID_UUID,
49 } PalCfgId_t;
50
51 /*************************************************************************************************
52   Function Declarations
53 *************************************************************************************************/
54 void PalCfgLoadData(uint8_t cfgId, uint8_t *pBuf, uint32_t len);
55 void PalCfgSetDeviceUuid(uint8_t *pBuf);
56     /* PAL_CFG */
58
59 #ifdef __cplusplus
60 };
61 #endif
62
63 #endif /* PAL_CFG_H */
```

## 6.9 /github/workspace/Libraries/Cordio/platform/include/pal_codec.h File Reference

Hardware audio codec interface file.

```
#include "wsf_types.h"
#include "wsf_os.h"
```

Include dependency graph for pal_codec.h:

### Classes

- struct AudioStdCodecInfo_t

  *Standard codec info block.*
- struct AudioVsCodecInfo_t

  *VS codec info block.*
- struct PalCodecSreamParam_t

  *Codec.*

### Typedefs

- typedef void(∗ **PalCodecDataReady_t**) (uint16_t id)

  *Buffer available call signature.*

## Enumerations

- enum PalAudioDir_t { PAL_CODEC_DIR_INPUT = 0 , PAL_CODEC_DIR_OUTPUT = 1 }

  *Audio data path direction.*
- enum PalAudioChan_t { PAL_CODEC_CH_LEFT = 0 , PAL_CODEC_CH_RIGHT = 1 , AUDIO_NUM_CH }

  *Audio Channel.*
- enum { **PAL_CODEC_CH_LEFT_BIT** = (1 << PAL_CODEC_CH_LEFT) , **PAL_CODEC_CH_RIGHT_BIT** = (1 << PAL_CODEC_CH_RIGHT) }

  *Audio Channel mask.*

## Functions

- void **PalCodecReadLocalSupportedCodecs** (uint8_t *pNumStd, AudioStdCodecInfo_t stdCodecs[ ], uint8_t *pNumVs, AudioVsCodecInfo_t vsCodecs[ ])
- bool_t **PalCodecReadLocalSupportedCodecCapabilities** (uint8_t codingFmt, uint16_t compId, uint16_t vsCodecId, PalAudioDir_t dir)
- bool_t **PalCodecReadLocalSupportedControllerDelay** (uint8_t codingFmt, uint16_t compId, uint16_t vsCodecId, PalAudioDir_t dir, uint32_t *pMinDly, uint32_t *pMaxDly)
- bool_t **PalCodecConfigureDataPath** (PalAudioDir_t dir, uint8_t dataPathId)
- void **PalCodecAmpInit** (void)
- uint8_t **PalCodecAmpGetVol** (void)
- void **PalCodecAmpVolumeUp** (void)
- void **PalCodecAmpVolumeDown** (void)
- void **PalCodecAmpMute** (void)
- void **PalCodecAmpUnmute** (void)
- void **PalCodecDataInit** (void)
- bool_t **PalCodecDataStartStream** (uint16_t id, PalCodecSreamParam_t *pParam)
- void **PalCodecDataStopStream** (uint16_t id)
- uint16_t **PalCodecDataStreamIn** (uint16_t id, uint8_t *pBuf, uint16_t len, uint32_t *pPktCtr)
- void **PalCodecDataStreamOut** (uint16_t id, const uint8_t *pBuf, uint16_t len, uint32_t pktCtr)

### 6.9.1 Detailed Description

Hardware audio codec interface file.

# 6.10 pal_codec.h

Go to the documentation of this file.

```
1 /*************************************************************************************************/
21 /*************************************************************************************************/
22
23 #ifndef PAL_CODEC_H
24 #define PAL_CODEC_H
25
26 #include "wsf_types.h"
27 #include "wsf_os.h"
28
29 #ifdef __cplusplus
30 extern "C" {
31 #endif
32
38 /*************************************************************************************************'
39   Data Types
40 *************************************************************************************************/
41
43 typedef enum
44 {
45   PAL_CODEC_DIR_INPUT   = 0,
46   PAL_CODEC_DIR_OUTPUT  = 1
47 } PalAudioDir_t;
48
50 typedef enum
51 {
52   PAL_CODEC_CH_LEFT     = 0,
53   PAL_CODEC_CH_RIGHT    = 1,
54   AUDIO_NUM_CH
55 } PalAudioChan_t;
56
58 enum
59 {
60   PAL_CODEC_CH_LEFT_BIT  = (1 « PAL_CODEC_CH_LEFT),
61   PAL_CODEC_CH_RIGHT_BIT = (1 « PAL_CODEC_CH_RIGHT)
62 };
63
65 typedef struct
66 {
67   uint8_t codecId;
68 } AudioStdCodecInfo_t;
69
71 typedef struct
72 {
73   uint16_t compId;
74   uint16_t codecId;
75 } AudioVsCodecInfo_t;
76
78 typedef void (*PalCodecDataReady_t)(uint16_t id);
79
81 typedef struct
82 {
83   PalAudioDir_t dir;
84   uint16_t      chMask;
85   uint32_t      intervalUsec;
86   uint32_t      pktCtr;
87   PalCodecDataReady_t rdyCback;
88 } PalCodecSreamParam_t;
89
90 /*************************************************************************************************
91   Function Declarations
92 *************************************************************************************************/
93
94 /* Codec Information */
95 void PalCodecReadLocalSupportedCodecs(uint8_t *pNumStd, AudioStdCodecInfo_t stdCodecs[],
96                                       uint8_t *pNumVs, AudioVsCodecInfo_t vsCodecs[]);
97 bool_t PalCodecReadLocalSupportedCodecCapabilities(uint8_t codingFmt, uint16_t compId, uint16_t
       vsCodecId, PalAudioDir_t dir);
98 bool_t PalCodecReadLocalSupportedControllerDelay(uint8_t codingFmt, uint16_t compId, uint16_t vsCodecId,
       PalAudioDir_t dir,
99                                                  uint32_t *pMinDly, uint32_t *pMaxDly);
100 bool_t PalCodecConfigureDataPath(PalAudioDir_t dir, uint8_t dataPathId);
101
102 /* Audio Amplifier */
103 void PalCodecAmpInit(void);
104 uint8_t PalCodecAmpGetVol(void);
105 void PalCodecAmpVolumeUp(void);
106 void PalCodecAmpVolumeDown(void);
107 void PalCodecAmpMute(void);
108 void PalCodecAmpUnmute(void);
109
110 /* Data Path */
111 void PalCodecDataInit(void);
```

```
112 bool_t PalCodecDataStartStream(uint16_t id, PalCodecSreamParam_t *pParam);
113 void PalCodecDataStopStream(uint16_t id);
114 uint16_t PalCodecDataStreamIn(uint16_t id, uint8_t *pBuf, uint16_t len, uint32_t *pPktCtr);
115 void PalCodecDataStreamOut(uint16_t id, const uint8_t *pBuf, uint16_t len, uint32_t pktCtr);
116     /* PAL_CODEC */
118
119 #ifdef __cplusplus
120 };
121 #endif
122
123 #endif /* PAL_CODEC_H */
```

## 6.11 /github/workspace/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 textLen, 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)

### 6.11.1 Detailed Description

Crypto driver definition.



## 6.12 pal_crypto.h

Go to the documentation of this file.

```
1 /*************************************************************************************************/
23 /*************************************************************************************************/
24
25 #ifndef PAL_CRYPTO_H
26 #define PAL_CRYPTO_H
27
28 #include "pal_types.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
37 /*************************************************************************************************
38   Macros
39 *************************************************************************************************/
```

```
40
42 #define PAL_CRYPTO_AES_BLOCK_SIZE               16
43
44 #define PAL_CRYPTO_LL_KEY_LEN                   16
45 #define PAL_CRYPTO_LL_IV_LEN                    8
46 #define PAL_CRYPTO_LL_DATA_MIC_LEN              4
49 #define SEC_CCM_KEY_LEN                         16
50
52 #define SEC_CCM_MAX_ADDITIONAL_LEN              ((1«16) - (1«8))
53
55 #define SEC_CCM_L                               2
56
58 #define SEC_CCM_NONCE_LEN                       (15-SEC_CCM_L)
59
60 /**************************************************************************************************
61   Data Types
62 **************************************************************************************************/
63
65 typedef enum
66 {
67   PAL_CRYPTO_STATE_UNINIT = 0,
68   PAL_CRYPTO_STATE_ERROR  = 0,
69   PAL_CRYPTO_STATE_READY
70 } PalCryptoState_t;
71
73 typedef struct
74 {
75   /* SK placed here for 32-bit alignment. */
76   uint8_t       sk[PAL_CRYPTO_LL_KEY_LEN];
77   uint8_t       iv[PAL_CRYPTO_LL_IV_LEN];
78   bool_t        enaEncrypt;
79   bool_t        enaDecrypt;
80   bool_t        enaAuth;
81   uint8_t       nonceMode;
82   uint16_t      *pEventCounter;
83   uint64_t      *pTxPktCounter;
84   uint64_t      *pRxPktCounter;
85   uint8_t       dir;
86   uint8_t       type;
87   void          *pEncryptCtx;
88   void          *pDecryptCtx;
89 } PalCryptoEnc_t;
90
91 /**************************************************************************************************
92   Function Declarations
93 **************************************************************************************************/
94
95 /* Initialization */
96 void PalCryptoInit(void);
97 void PalCryptoDeInit(void);
98
99 /* Key generation */
100 void PalCryptoGenerateP256KeyPair(const uint8_t *pPrivKey, uint8_t *pPubKey);
101 void PalCryptoGenerateDhKey(const uint8_t *pPubKey, const uint8_t *pPrivKey, uint8_t *pDhKey);
102 bool_t PalCryptoValidatePublicKey(const uint8_t *pPubKey, bool_t generateKey);
103 void PalCryptoGenerateRandomNumber(uint8_t *pBuf, uint8_t len);
104
105 /* CCM */
106 uint32_t PalCryptoCcmDec(const uint8_t *pKey, uint8_t *pNonce, uint8_t *pCypherText, uint16_t textLen,
107                          uint8_t *pClear, uint16_t clearLen, uint8_t *pMic, uint8_t micLen,
108                          uint8_t *pResult, uint8_t handlerId, uint16_t param, uint8_t event);
109 void PalCryptoCcmEnc(const uint8_t *pKey, uint8_t *pNonce, uint8_t *pPlainText, uint16_t textLen,
110               uint8_t *pClear, uint16_t clearLen, uint8_t micLen, uint8_t *pResult,
111               uint8_t handlerId, uint16_t param, uint8_t event);
112
113 /* Crypto AES */
114 void PalCryptoAesEcb(const uint8_t *pKey, uint8_t *pOut, const uint8_t *pIn);
115 void PalCryptoAesCmac(const uint8_t *pKey, uint8_t *pOut, const uint8_t *pIn, uint16_t len);
116 void PalCryptoAesEnable(PalCryptoEnc_t *pEnc, uint8_t id, uint8_t localDir);
117 bool_t PalCryptoAesCcmEncrypt(PalCryptoEnc_t *pEnc, uint8_t *pHdr, uint8_t *pBuf, uint8_t *pMic);
118 bool_t PalCryptoAesCcmDecrypt(PalCryptoEnc_t *pEnc, uint8_t *pBuf);
119 void PalCryptoSetEncryptPacketCount(PalCryptoEnc_t *pEnc, uint64_t pktCnt);
120 void PalCryptoSetDecryptPacketCount(PalCryptoEnc_t *pEnc, uint64_t pktCnt);
121     /* PAL_CRYPTO */
123
124 #ifdef __cplusplus
125 };
126 #endif
127
128 #endif /* PAL_CRYPTO_H */
```

# 6.13 /github/workspace/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 { 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)

### 6.13.1 Detailed Description

LED driver definition.

## 6.14 pal_led.h

Go to the documentation of this file.

```
1 /*************************************************************************************************/
21 /*************************************************************************************************/
22
23 #ifndef PAL_LED_H
24 #define PAL_LED_H
25
26 #include "pal_types.h"
27
28 #ifdef __cplusplus
29 extern "C" {
30 #endif
31
35 /*************************************************************************************************
36   Data Types
37 *************************************************************************************************/
38
40 enum
41 {
42   /* System signals. */
43   PAL_LED_ID_CPU_ACTIVE     = 0x30,
44   PAL_LED_ID_ERROR          = 0x31,
45 };
46
47 /*************************************************************************************************
48   Function Declarations
49 *************************************************************************************************/
50
51 /* Initialization */
52 void PalLedInit(void);
53 void PalLedDeInit(void);
54
55 /* Control and Status */
56 void PalLedOn(uint8_t id);
57 void PalLedOff(uint8_t id);
58     /* PAL_LED */
60
61 #ifdef __cplusplus
62 };
63 #endif
64
65 #endif /* PAL_LED_H */
```

## 6.15 /github/workspace/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)

### 6.15.1 Detailed Description

RTC timer interface file.



## 6.16 pal_rtc.h

Go to the documentation of this file.

```
1 /*************************************************************************************************/
23 /*************************************************************************************************/
24
25 #ifndef PAL_RTC_H
26 #define PAL_RTC_H
27
28 #include "pal_types.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
37 /*************************************************************************************************
38   Macros
39 *************************************************************************************************/
40
42 #ifndef PAL_MAX_RTC_COUNTER_VAL
43 #define PAL_MAX_RTC_COUNTER_VAL     (0x00FFFFFF)
44 #endif
45
47 #define PAL_RTC_TICKS_PER_SEC       (32768) /* RTC ticks per second (with prescaler) */
48
50 typedef void (*palRtcIrqCback_t)(void);
51
```

```
52 /**************************************************************************************************
53   Type Definitions
54 **************************************************************************************************/
55
57 typedef enum
58 {
59   PAL_RTC_STATE_UNINIT = 0,
60   PAL_RTC_STATE_ERROR  = 0,
61   PAL_RTC_STATE_READY  = 1
62 } PalRtcState_t;
63
64 /**************************************************************************************************
65   Function Declarations
66 **************************************************************************************************/
67
68 /* Initialization */
69 void PalRtcInit(void);
70
71 /* Control and Status */
72 void PalRtcEnableCompareIrq(uint8_t channelId);
73 void PalRtcDisableCompareIrq(uint8_t channelId);
74 uint32_t PalRtcCounterGet(void);
75 void PalRtcCompareSet(uint8_t channelId, uint32_t value);
76 PalRtcState_t PalRtcGetState(void);
77     /* PAL_RTC */
79
80 #ifdef __cplusplus
81 };
82 #endif
83
84 #endif
```

## 6.17 /github/workspace/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)

### 6.17.1 Detailed Description

System hooks.



## 6.18 pal_sys.h

Go to the documentation of this file.

```
1 /**************************************************************************************************/
23 /**************************************************************************************************/
24
25 #ifndef PAL_SYS_H
26 #define PAL_SYS_H
27
28 #include "pal_types.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
37 /**************************************************************************************************
38   Macros
39 **************************************************************************************************/
40
41 /* Common error handling routines; for use with PAL implementation only. */
42
43 #ifdef DEBUG
44
46 #define PAL_SYS_ASSERT(expr)    { if (!(expr)) { PalSysAssertTrap(); } }
47
48 #else
49
51 #define PAL_SYS_ASSERT(expr)
52
53 #endif
54
55 /**************************************************************************************************
56   Function Declarations
57 **************************************************************************************************/
58
59 /* Initialization */
60 void PalSysInit(void);
61
62 /* Diagnostics */
63 void PalSysAssertTrap(void);
64 void PalSysSetTrap(bool_t enable);
65 uint32_t PalSysGetAssertCount(void);
66 uint32_t PalSysGetStackUsage(void);
67
68 /* Power Management */
69 void PalSysSleep(void);
70 bool_t PalSysIsBusy(void);
71 void PalSysSetBusy(void);
72 void PalSysSetIdle(void);
73
74 /* Critical Section */
75 void PalEnterCs(void);
76 void PalExitCs(void);
77     /* PAL_SYS */
79
80 #ifdef __cplusplus
81 };
82 #endif
83
84 #endif  /* PAL_SYS_H */
```

## 6.19 /github/workspace/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)

### 6.19.1 Detailed Description

Timer interface file.

## 6.20 pal_timer.h

Go to the documentation of this file.

```
1 /*************************************************************************************************/
23 /*************************************************************************************************/
24
25 #ifndef PAL_TIMER_H
26 #define PAL_TIMER_H
27
28 #include "pal_types.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
37 /*************************************************************************************************
38   Macros
39 *************************************************************************************************/
40
42 typedef enum
43 {
44   PAL_TIMER_STATE_UNINIT = 0,
45   PAL_TIMER_STATE_ERROR  = 0,
46   PAL_TIMER_STATE_READY,
47   PAL_TIMER_STATE_BUSY
48 } PalTimerState_t;
49
50 /*************************************************************************************************
51   Data Types
52 *************************************************************************************************/
53
55 typedef void (*PalTimerCompCback_t)(void);
56
57 /*************************************************************************************************
58   Function Declarations
59 *************************************************************************************************/
60
61 /* Initialization */
62 void PalTimerInit(PalTimerCompCback_t expCback);
63 void PalTimerDeInit(void);
64
65 /* Control and Status */
66 PalTimerState_t PalTimerGetState(void);
67 void PalTimerStart(uint32_t expUsec);
68 void PalTimerStop(void);
69 uint32_t PalTimerGetCurrentTime(void);
70 uint32_t PalTimerGetExpTime(void);
71 void PalTimerSleep(uint32_t expUsec);
72 void PalTimerRestore(uint32_t schTime);
73     /* PAL_TIMER */
75
76 #ifdef __cplusplus
77 };
78 #endif
79
80 #endif
```

## 6.21 /github/workspace/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)

### 6.21.1 Detailed Description

UART driver definition.

# 6.22 pal_uart.h

Go to the documentation of this file.

```
1 /**************************************************************************************************/
23 /**************************************************************************************************/
24
25 #ifndef PAL_UART_H
26 #define PAL_UART_H
27
28 #include "pal_types.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
37 /**************************************************************************************************
38   Data Types
39 **************************************************************************************************/
40
42 typedef void (*PalUartCompCback_t)(void);
43
45 typedef struct
46 {
47   PalUartCompCback_t rdCback;
48   PalUartCompCback_t wrCback;
49   uint32_t baud;
50   bool_t hwFlow;
51 } PalUartConfig_t;
52
54 typedef enum
55 {
56   PAL_UART_STATE_UNINIT = 0,
57   PAL_UART_STATE_ERROR  = 0,
58   PAL_UART_STATE_READY  = 1,
59   PAL_UART_STATE_BUSY   = 2,
60 } PalUartState_t;
61
63 typedef enum
64 {
65   PAL_UART_ID_USER      = 0,
66   PAL_UART_ID_CHCI      = 1,
67   PAL_UART_ID_TERMINAL  = 2,
68   PAL_UART_ID_MAX
69 } PalUartId_t;
70
71 /**************************************************************************************************
72   Function Declarations
73 **************************************************************************************************/
74
75 /* Initialization */
76 void PalUartInit(PalUartId_t id, const PalUartConfig_t *pCfg);
77 void PalUartDeInit(PalUartId_t id);
78
79 /* Control and Status */
80 PalUartState_t PalUartGetState(PalUartId_t id);
81
82 /* Data Transfer */
83 void PalUartReadData(PalUartId_t id, uint8_t *pData, uint16_t len);
84 void PalUartWriteData(PalUartId_t id, const uint8_t *pData, uint16_t len);
85     /* PAL_UART */
87
88 #ifdef __cplusplus
89 };
90 #endif
91
92 #endif /* PAL_UART_H */
```

# Index