

MAX32655 Cordio Platform

Generated by Doxygen 1.9.5



<b>1 Module Index</b>	<b>1</b>
1.1 Modules	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Module Documentation</b>	<b>7</b>
4.1 PAL_BUTTON	7
4.1.1 Detailed Description	7
4.1.2 Enumeration Type Documentation	7
4.1.2.1 anonymous enum	7
4.1.2.2 PalBtnPos_t	8
4.1.2.3 PalBtnState_t	8
4.2 PAL_CFG	8
4.2.1 Detailed Description	9
4.2.2 Enumeration Type Documentation	9
4.2.2.1 PalCfgId_t	9
4.3 PAL_SYS	9
4.3.1 Detailed Description	10
4.4 PAL_CODEC	10
4.4.1 Detailed Description	11
4.4.2 Enumeration Type Documentation	11
4.4.2.1 PalAudioChan_t	11
4.4.2.2 PalAudioDir_t	11
4.5 PAL_CRYPTO	11
4.5.1 Detailed Description	12
4.5.2 Macro Definition Documentation	12
4.5.2.1 PAL_CRYPTO_LL_DATA_MIC_LEN	12
4.5.2.2 PAL_CRYPTO_LL_IV_LEN	13
4.5.2.3 PAL_CRYPTO_LL_KEY_LEN	13
4.5.3 Enumeration Type Documentation	13
4.5.3.1 PalCryptoState_t	13
4.6 PAL_BB_BLE_CHAN	13
4.6.1 Detailed Description	13
4.6.2 Function Documentation	14
4.6.2.1 PalBbBleSetChannelParam()	14
4.7 PAL_BB_BLE_DATA	14
4.7.1 Detailed Description	15
4.7.2 Enumeration Type Documentation	15
4.7.2.1 PalBbBleMode_t	15

4.7.3 Function Documentation	15
4.7.3.1 PalBbBleCancelData()	15
4.7.3.2 PalBbBleCancelTifs()	16
4.7.3.3 PalBbBleRxData()	16
4.7.3.4 PalBbBleRxTifsData()	16
4.7.3.5 PalBbBleSetDataParams()	16
4.7.3.6 PalBbBleSetOpParams()	17
4.7.3.7 PalBbBleTxData()	17
4.7.3.8 PalBbBleTxTifsData()	17
4.8 PAL_BB_BLE_INIT	18
4.8.1 Detailed Description	18
4.8.2 Function Documentation	18
4.8.2.1 PalBbBleDisable()	18
4.8.2.2 PalBbBleEnable()	18
4.8.2.3 PalBbBleInit()	18
4.9 PAL_BB_BLE_TEST	19
4.9.1 Detailed Description	19
4.9.2 Function Documentation	19
4.9.2.1 PalBbBleEnableDataWhitening()	19
4.9.2.2 PalBbBleEnablePrbs15()	19
4.9.2.3 PalBbBleInlineEncryptDecryptSetDirection()	20
4.9.2.4 PalBbBleInlineEncryptSetPacketCount()	20
4.9.2.5 PalBbBleLowPower()	20
4.10 PAL_TIMER	20
4.10.1 Detailed Description	21
4.10.2 Enumeration Type Documentation	21
4.10.2.1 PalTimerState_t	21
4.11 PAL_LED	21
4.11.1 Detailed Description	22
4.11.2 Enumeration Type Documentation	22
4.11.2.1 anonymous enum	22
4.12 PAL_RTC	22
4.12.1 Detailed Description	23
4.12.2 Enumeration Type Documentation	23
4.12.2.1 PalRtcState_t	23
4.13 PAL_BB_INIT	23
4.13.1 Detailed Description	23
4.13.2 Function Documentation	24
4.13.2.1 PalBbDisable()	24
4.13.2.2 PalBbEnable()	24
4.13.2.3 PalBbInit()	24
4.13.2.4 PalBbLoadCfg()	24

4.13.2.5 PalBbRestore()	25
4.14 PAL_BB_CLOCK	25
4.14.1 Detailed Description	25
4.14.2 Function Documentation	25
4.14.2.1 PalBbGetCurrentTime()	25
4.14.2.2 PalBbGetTimestamp()	25
4.14.2.3 PalBbRegisterProtIrq()	26
4.14.2.4 PalBbSetProtId()	26
4.15 PAL_UART	27
4.15.1 Detailed Description	27
4.15.2 Enumeration Type Documentation	27
4.15.2.1 PalUartId_t	27
4.15.2.2 PalUartState_t	28
<b>5 Class Documentation</b>	<b>29</b>
5.1 AudioStdCodecInfo_t Struct Reference	29
5.1.1 Detailed Description	29
5.1.2 Member Data Documentation	29
5.1.2.1 codecId	29
5.2 AudioVsCodecInfo_t Struct Reference	30
5.2.1 Detailed Description	30
5.2.2 Member Data Documentation	30
5.2.2.1 codecId	30
5.2.2.2 compId	30
5.3 PalBbBleChan_t Struct Reference	30
5.3.1 Detailed Description	31
5.3.2 Member Data Documentation	31
5.3.2.1 accAddr	31
5.3.2.2 chanIdx	31
5.3.2.3 crcInit	31
5.3.2.4 enc	31
5.3.2.5 initTxPhyOptions	32
5.3.2.6 opType	32
5.3.2.7 peerRxStableModIdx	32
5.3.2.8 peerTxStableModIdx	32
5.3.2.9 rxPhy	32
5.3.2.10 tifsTxPhyOptions	32
5.3.2.11 txPhy	32
5.3.2.12 txPower	33
5.4 PalBbBleDataParam_t Struct Reference	33
5.4.1 Detailed Description	33
5.4.2 Member Data Documentation	33

5.4.2.1 dueUsec . . . . .	33
5.4.2.2 rxCback . . . . .	33
5.4.2.3 rxTimeoutUsec . . . . .	34
5.4.2.4 txCback . . . . .	34
5.5 PalBbBleOpParam_t Struct Reference . . . . .	34
5.5.1 Detailed Description . . . . .	34
5.5.2 Member Data Documentation . . . . .	34
5.5.2.1 ifsMode . . . . .	34
5.5.2.2 ifsTime . . . . .	35
5.5.2.3 plfsChan . . . . .	35
5.6 PalBbBleTxBufDesc_t Struct Reference . . . . .	35
5.6.1 Detailed Description . . . . .	35
5.6.2 Member Data Documentation . . . . .	35
5.6.2.1 len . . . . .	35
5.6.2.2 pBuf . . . . .	36
5.7 PalBbCfg_t Struct Reference . . . . .	36
5.7.1 Detailed Description . . . . .	36
5.7.2 Member Data Documentation . . . . .	36
5.7.2.1 BbTimerBoundaryUsec . . . . .	36
5.7.2.2 clkPpm . . . . .	36
5.7.2.3 maxScanPeriodMsec . . . . .	37
5.7.2.4 rfSetupDelayUsec . . . . .	37
5.7.2.5 schSetupDelayUsec . . . . .	37
5.8 PalCodecStreamParam_t Struct Reference . . . . .	37
5.8.1 Detailed Description . . . . .	37
5.8.2 Member Data Documentation . . . . .	37
5.8.2.1 chMask . . . . .	38
5.8.2.2 dir . . . . .	38
5.8.2.3 intervalUsec . . . . .	38
5.8.2.4 pktCtr . . . . .	38
5.8.2.5 rdyCback . . . . .	38
5.9 PalCryptoEnc_t Struct Reference . . . . .	38
5.9.1 Detailed Description . . . . .	39
5.9.2 Member Data Documentation . . . . .	39
5.9.2.1 dir . . . . .	39
5.9.2.2 enaAuth . . . . .	39
5.9.2.3 enaDecrypt . . . . .	39
5.9.2.4 enaEncrypt . . . . .	39
5.9.2.5 iv . . . . .	40
5.9.2.6 nonceMode . . . . .	40
5.9.2.7 pDecryptCtx . . . . .	40
5.9.2.8 pEncryptCtx . . . . .	40

5.9.2.9 pEventCounter . . . . .	40
5.9.2.10 pRxPktCounter . . . . .	40
5.9.2.11 pTxPktCounter . . . . .	40
5.9.2.12 sk . . . . .	40
5.9.2.13 type . . . . .	41
5.10 PalUartConfig_t Struct Reference . . . . .	41
5.10.1 Detailed Description . . . . .	41
5.10.2 Member Data Documentation . . . . .	41
5.10.2.1 baud . . . . .	41
5.10.2.2 hwFlow . . . . .	41
5.10.2.3 rdCback . . . . .	41
5.10.2.4 wrCback . . . . .	41
<b>6 File Documentation</b> . . . . .	<b>43</b>
6.1 /github/workspace/Libraries/Cordio/platform/include/pal_bb.h File Reference . . . . .	43
6.2 pal_bb.h . . . . .	43
6.3 /github/workspace/Libraries/Cordio/platform/include/pal_bb_ble.h File Reference . . . . .	45
6.3.1 Detailed Description . . . . .	47
6.3.2 Macro Definition Documentation . . . . .	47
6.3.2.1 LL_ENABLE_TESTER . . . . .	47
6.3.3 Enumeration Type Documentation . . . . .	47
6.3.3.1 anonymous enum . . . . .	47
6.3.3.2 anonymous enum . . . . .	48
6.4 pal_bb_ble.h . . . . .	48
6.5 /github/workspace/Libraries/Cordio/platform/include/pal_btn.h File Reference . . . . .	50
6.5.1 Detailed Description . . . . .	51
6.6 pal_btn.h . . . . .	51
6.7 /github/workspace/Libraries/Cordio/platform/include/pal_cfg.h File Reference . . . . .	52
6.7.1 Detailed Description . . . . .	52
6.8 pal_cfg.h . . . . .	53
6.9 /github/workspace/Libraries/Cordio/platform/include/pal_codec.h File Reference . . . . .	53
6.9.1 Detailed Description . . . . .	54
6.10 pal_codec.h . . . . .	55
6.11 /github/workspace/Libraries/Cordio/platform/include/pal_crypto.h File Reference . . . . .	56
6.11.1 Detailed Description . . . . .	57
6.12 pal_crypto.h . . . . .	57
6.13 /github/workspace/Libraries/Cordio/platform/include/pal_led.h File Reference . . . . .	59
6.13.1 Detailed Description . . . . .	59
6.14 pal_led.h . . . . .	60
6.15 /github/workspace/Libraries/Cordio/platform/include/pal_rtc.h File Reference . . . . .	60
6.15.1 Detailed Description . . . . .	61
6.16 pal_rtc.h . . . . .	61

6.17 /github/workspace/Libraries/Cordio/platform/include/pal_sys.h File Reference . . . . .	62
6.17.1 Detailed Description . . . . .	63
6.18 pal_sys.h . . . . .	63
6.19 /github/workspace/Libraries/Cordio/platform/include/pal_timer.h File Reference . . . . .	64
6.19.1 Detailed Description . . . . .	64
6.20 pal_timer.h . . . . .	65
6.21 /github/workspace/Libraries/Cordio/platform/include/pal_uart.h File Reference . . . . .	65
6.21.1 Detailed Description . . . . .	66
6.22 pal_uart.h . . . . .	67
<b>Index</b>	<b>69</b>



# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

PAL_BUTTON . . . . .	7
PAL_CFG . . . . .	8
PAL_SYS . . . . .	9
PAL_CODEC . . . . .	10
PAL_CRYPTO . . . . .	11
PAL_BB_BLE_CHAN . . . . .	13
PAL_BB_BLE_DATA . . . . .	14
PAL_BB_BLE_INIT . . . . .	18
PAL_BB_BLE_TEST . . . . .	19
PAL_TIMER . . . . .	20
PAL_LED . . . . .	21
PAL_RTC . . . . .	22
PAL_BB_INIT . . . . .	23
PAL_BB_CLOCK . . . . .	25
PAL_UART . . . . .	27



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AudioStdCodecInfo_t</a>	
Standard codec info block . . . . .	29
<a href="#">AudioVsCodecInfo_t</a>	
VS codec info block . . . . .	30
<a href="#">PalBbBleChan_t</a>	
BLE channelization parameters . . . . .	30
<a href="#">PalBbBleDataParam_t</a>	
BLE data transfer parameters . . . . .	33
<a href="#">PalBbBleOpParam_t</a>	
Operation parameters . . . . .	34
<a href="#">PalBbBleTxBufDesc_t</a>	
Transmit buffer descriptor . . . . .	35
<a href="#">PalBbCfg_t</a>	
BB configuration . . . . .	36
<a href="#">PalCodecStreamParam_t</a>	
Codec . . . . .	37
<a href="#">PalCryptoEnc_t</a>	
Encryption data . . . . .	38
<a href="#">PalUartConfig_t</a>	
Peripheral configuration . . . . .	41



## Chapter 3

# File Index

### 3.1 File List

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

/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_bb.h</a>	
Baseband interface file	43
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_bb_ble.h</a>	
BLE Baseband interface file	45
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_btn.h</a>	
Button driver definition	50
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_cfg.h</a>	
System configuration definition	52
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_codec.h</a>	
Hardware audio codec interface file	53
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_crypto.h</a>	
Crypto driver definition	56
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_led.h</a>	
LED driver definition	59
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_rtc.h</a>	
RTC timer interface file	60
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_sys.h</a>	
System hooks	62
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_timer.h</a>	
Timer interface file	64
/github/workspace/Libraries/Cordio/platform/include/ <a href="#">pal_uart.h</a>	
UART driver definition	65



## 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 [PalCodecStreamParam\\_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](#) }
- 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 compld, uint16\_t vsCodecId, [PalAudioDir\\_t](#) dir)
- bool\_t [PalCodecReadLocalSupportedControllerDelay](#) (uint8\_t codingFmt, uint16\_t compld, uint16\_t vsCodecId, [PalAudioDir\\_t](#) dir, uint32\_t \*pMinDly, uint32\_t \*pMaxDly)
- bool\_t [PalCodecConfigureDataPath](#) ([PalAudioDir\\_t](#) dir, uint8\_t dataPathId)
- void [PalCodecAmplInit](#) (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, [PalCodecStreamParam\\_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\_CRYPTOLL\_IV\_LEN

```
#define PAL_CRYPTOLL_IV_LEN 8
```

Initialization vector length.

#### 4.5.2.3 PAL\_CRYPTOLL\_KEY\_LEN

```
#define PAL_CRYPTOLL_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_CRYPTOLL_STATE_UNINIT	Uninitialized state.
PAL_CRYPTOLL_STATE_ERROR	Error state.
PAL_CRYPTOLL_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 channelization parameters.

## 4.6.2 Function Documentation

### 4.6.2.1 PalBbBleSetChannelParam()

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

Set channelization parameters.

#### Parameters

<i>pChan</i>	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

<code>PAL_BB_IFS_MODE_CLR</code>	Clear IFS (last packet).
<code>PAL_BB_IFS_MODE_TOGGLE_TIFS</code>	Toggle operation with TIFS timing.
<code>PAL_BB_IFS_MODE_SAME_ABS</code>	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

<i>pBuf</i>	Receive data buffer.
<i>len</i>	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

<i>pBuf</i>	Receive data buffer.
<i>len</i>	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

<i>pParam</i>	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

<i>pOpParam</i>	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

<i>descs</i>	Array of transmit buffer descriptors.
<i>cnt</i>	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

<i>descs</i>	Array of transmit buffer descriptor.
<i>cnt</i>	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

<i>enable</i>	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

<i>enable</i>	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

<i>dir</i>	0=slave, non-zero=master
------------	--------------------------

#### 4.9.2.4 PalBbBleInlineEncryptSetPacketCount()

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

Set the inline encryption packet count for transmit.

##### Parameters

<i>count</i>	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

<code>PAL_TIMER_STATE_UNINIT</code>	Uninitialized state.
<code>PAL_TIMER_STATE_ERROR</code>	Error state.
<code>PAL_TIMER_STATE_READY</code>	Ready state.
<code>PAL_TIMER_STATE_BUSY</code>	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

<i>pCfg</i>	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**

<i>pTime</i>	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,
    bbDrvIrqCbck_t timerCbck,
    bbDrvIrqCbck_t radioCbck )
```

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

**Parameters**

<i>protId</i>	Protocol ID.
<i>timerCbck</i>	Timer IRQ callback.
<i>radioCbck</i>	Timer IRQ callback.

**4.14.2.4 PalBbSetProtId()**

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

Set protocol ID.

**Parameters**

<i>protId</i>	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 }
  - enum [PalUartId\\_t](#) { [PAL\\_UART\\_ID\\_USER](#) = 0 , [PAL\\_UART\\_ID\\_CHCI](#) = 1 , [PAL\\_UART\\_ID\\_TERMINAL](#) = 2 , [PAL\\_UART\\_ID\\_MAX](#) }
- Operational states.*
- 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

<a href="#">PAL_UART_ID_USER</a>	UART 0.
<a href="#">PAL_UART_ID_CHCI</a>	UART CHCI.
<a href="#">PAL_UART_ID_TERMINAL</a>	UART TERMINAL.
<a href="#">PAL_UART_ID_MAX</a>	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 [compld](#)
- uint16\_t [codeclId](#)

### 5.2.1 Detailed Description

VS codec info block.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 codeclId

```
uint16_t AudioVsCodecInfo_t::codeclId
```

Codec ID.

#### 5.2.2.2 compld

```
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 \\* plfsChan](#)

### 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 plfsChan

```
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](/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
38 typedef enum
39 {
40     BB_PROT_NONE,
41     BB_PROT_BLE,
42     BB_PROT_BLE_DTM,
43     BB_PROT_PRBS15,
44     BB_PROT_15P4,
45     BB_PROT_NUM
46 } PalBbProt_t;
47
48 enum
49 {
50     BB_STATUS_SUCCESS,
51     BB_STATUS_FAILED,
52     BB_STATUS_CANCELED,
53     BB_STATUS_RX_TIMEOUT,
54     BB_STATUS_CRC_FAILED,
55     BB_STATUS_FRAME_FAILED,
56     BB_STATUS_ACK_FAILED,
57     BB_STATUS_ACK_TIMEOUT,
58     BB_STATUS_TX_CCA_FAILED,
59     BB_STATUS_TX_FAILED
```

```

62 };
63
64 typedef enum
65 {
66     BB_PHY_BLE_1M      = 1,
67     BB_PHY_BLE_2M      = 2,
68     BB_PHY_BLE_CODED   = 3,
69     BB_PHY_15P4        = 4,
70 } PalBbPhy_t;
71
72
73 enum
74 {
75     BB_PHY_OPTIONS_DEFAULT      = 0,
76     BB_PHY_OPTIONS_BLE_S2      = 1,
77     BB_PHY_OPTIONS_BLE_S8      = 2
78 };
79
80
81 #ifndef BB_CLK_RATE_HZ
82 #define BB_CLK_RATE_HZ          1000000
83 #endif
84
85
86 #define BB_MATH_DIV_10E6(n)     ((uint32_t)((uint64_t)(n) * UINT64_C(4295)) >> 32))
87
88
89 #if (BB_CLK_RATE_HZ == 1000000)
90 #define BB_US_TO_BB_TICKS(us)   (us)
91 #elif (BB_CLK_RATE_HZ == 8000000)
92 #define BB_US_TO_BB_TICKS(us)   ((uint32_t)((us) << 3))
93 #elif (BB_CLK_RATE_HZ == 32768)
94 #define BB_US_TO_BB_TICKS(us)   ((uint32_t)((uint64_t)(us) * (uint64_t)(70368745)) >> 31)) /*
95                                     calculated value may be one tick low */
96 #else
97 #define BB_US_TO_BB_TICKS(us)   BB_MATH_DIV_10E6((uint64_t)(us) * (uint64_t)(BB_CLK_RATE_HZ))
98 #endif
99
100 #define RTC_CLOCK_RATE          32768
101 #define USE_RTC_BB_CLK          (BB_CLK_RATE_HZ == RTC_CLOCK_RATE)
102
103 #if (BB_CLK_RATE_HZ == 1000000)
104 #define BB_TICKS_TO_US(n)       (n)
105 #elif (BB_CLK_RATE_HZ == 8000000)
106 #define BB_TICKS_TO_US(n)       ((n) >> 3)
107 #elif (BB_CLK_RATE_HZ == 32768)
108 #define BB_TICKS_TO_US(n)       ((uint32_t)((uint64_t)(n) * 15625) >> 9)
109 #else
110 #define BB_TICKS_TO_US(n)       ((uint32_t)((uint64_t)(n) * 1000000 / BB_CLK_RATE_HZ))
111 #endif
112
113 #define BB_MAX_SCAN_PERIOD_MS   1000
114
115 #define BB_RF_SETUP_DELAY_US     150
116
117 #define BB_SCH_SETUP_DELAY_US    500
118
119 #define BB_TIMER_1MHZ_MAX_VALUE_US 0xFFFFFFFF /* 2^32 - 1 = 0xFFFFFFFF. */
120
121 #define BB_TIMER_8MHZ_MAX_VALUE_US 0x1FFFFFFF /* 2^29 - 1 = 0x1FFFFFFF. */
122
123 #define BB_RTC_MAX_VALUE_US      511999999 /* 2^24 / 32768 * 10^6 - 1 = 512 * 10^6 - 1 = 511999999.
124                                             */
125
126
127 /*****
128  Type Definitions
129 *****/
130
131 typedef void (*bbDrvIrqCbck_t)(void);
132
133
134 typedef struct
135 {
136     uint16_t clkPpm;
137     uint8_t rfSetupDelayUsec;
138     uint16_t maxScanPeriodMsec;
139     uint16_t schSetupDelayUsec;
140     uint32_t BbTimerBoundaryUsec;
141 } PalBbCfg_t;
142
143
144 /*****
145  Function Declarations
146 *****/
147
148 void PalBbInit(void);
149
150
151 void PalBbRestore(void);
152
153

```

```

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 */
216
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, bbDrvIrqCbck_t timerCbck, bbDrvIrqCbck_t radioCbck);
260
261 /*****
267 /*****
268 void PalBbSetProtId(uint8_t protId);
269     /* PAL_BB_CLOCK */
270
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\_CNTR** }  
*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** descscs[ ], uint8\_t cnt)  
*Transmit a packet.*
- void **PalBbBleTxTifsData** (**PalBbBleTxBufDesc\_t** descscs[ ], 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.

Copyright (c) 2013-2019 Arm Ltd. All Rights Reserved.

Copyright (c) 2019-2020 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

### 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, <a href="#">PalCryptoEnc_t::pEventCounter</a> must be non-NULL.
PAL_BB_NONCE_MODE_EXT64_CNTR	64-bit counter mode, <a href="#">PalCryptoEnc_t::pTxPktCounter</a> / <a href="#">pRxPktCounter</a> 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
47 enum
48 {
49     PAL_BB_NONCE_MODE_PKT_CNTR,
50     PAL_BB_NONCE_MODE_EXT16_CNTR,
51     PAL_BB_NONCE_MODE_EXT64_CNTR
52 };
53
54 enum
55 {
56     PAL_BB_TYPE_ACL,
57     PAL_BB_TYPE_CIS,
58     PAL_BB_TYPE_BIS
59 };
60
61 typedef struct
62 {
63     uint8_t      opType;
64     uint8_t      chanIdx;
65     int8_t       txPower;
66     uint32_t     accAddr;
67     uint32_t     crcInit;
68     uint8_t      txPhy;
69     uint8_t      rxPhy;
70     uint8_t      initTxPhyOptions;
71     uint8_t      tifsTxPhyOptions;
72     bool_t       peerTxStableModIdx;
73     bool_t       peerRxStableModIdx;
74     PalCryptoEnc_t enc;
75 #if (LL_ENABLE_TESTER)
76     uint32_t     accAddrRx;
77     uint32_t     accAddrTx;
78     uint32_t     crcInitRx;
79     uint32_t     crcInitTx;
80

```



```

87  int8_t          txPwrOffset;
88 #endif
89 } PalBbBleChan_t;
90 /* PAL_BB_BLE_CHAN */
91
92 typedef void (*PalBbBleTxIsr_t)(uint8_t status);
93
94 typedef void (*PalBbBleRxIsr_t)(uint8_t status, int8_t rssi, uint32_t crc, uint32_t timestamp, uint8_t
rxPhyOptions);
95
96 typedef enum
97 {
98     PAL_BB_IFS_MODE_CLR,
99     PAL_BB_IFS_MODE_TOGGLE_TIFS,
100    PAL_BB_IFS_MODE_SAME_ABS
101 } PalBbIifsMode_t;
102
103 typedef struct
104 {
105     PalBbBleTxIsr_t    txCback;
106     PalBbBleRxIsr_t    rxCback;
107     uint32_t            dueUsec;
108     uint32_t            rxTimeoutUsec;
109 } PalBbBleDataParam_t;
110
111 typedef struct
112 {
113     PalBbIifsMode_t     ifsMode:8;
114     uint32_t             ifsTime;
115     PalBbBleChan_t      *pIifsChan;
116 } PalBbBleOpParam_t;
117
118 typedef struct
119 {
120     uint16_t             len;
121     uint8_t              *pBuf;
122 } PalBbBleTxBufDesc_t;
123 /* PAL_BB_BLE_DATA */
124
125 /*****
126  * Function Declarations
127  *****/
128
129 /*****
130  * void PalBbBleInit(void);
131  *****/
132 void PalBbBleInit(void);
133
134 /*****
135  * void PalBbBleEnable(void);
136  *****/
137 void PalBbBleEnable(void);
138
139 /*****
140  * void PalBbBleDisable(void);
141  *****/
142 void PalBbBleDisable(void);
143 /* PAL_BB_BLE_INIT */
144
145 /*****
146  * void PalBbBleSetChannelParam(PalBbBleChan_t *pChan);
147  *****/
148 void PalBbBleSetChannelParam(PalBbBleChan_t *pChan);
149 /* PAL_BB_BLE_CHAN */
150
151 /*****
152  * void PalBbBleSetDataParams(const PalBbBleDataParam_t *pParam);
153  *****/
154 void PalBbBleSetDataParams(const PalBbBleDataParam_t *pParam);
155
156 /*****
157  * void PalBbBleSetOpParams(const PalBbBleOpParam_t *pOpParam);
158  *****/
159 void PalBbBleSetOpParams(const PalBbBleOpParam_t *pOpParam);
160
161 /*****
162  * void PalBbBleTxData(PalBbBleTxBufDesc_t desc[], uint8_t cnt);
163  *****/
164 void PalBbBleTxData(PalBbBleTxBufDesc_t desc[], uint8_t cnt);
165
166 /*****
167  * void PalBbBleTxTifsData(PalBbBleTxBufDesc_t desc[], uint8_t cnt);
168  *****/
169 void PalBbBleTxTifsData(PalBbBleTxBufDesc_t desc[], uint8_t cnt);
170
171 /*****
172  * void PalBbBleRxData(uint8_t *pBuf, uint16_t len);
173  *****/
174 void PalBbBleRxData(uint8_t *pBuf, uint16_t len);
175
176 /*****
177  * void PalBbBleRxTifsData(uint8_t *pBuf, uint16_t len);
178  *****/
179 void PalBbBleRxTifsData(uint8_t *pBuf, uint16_t len);
180
181 /*****
182  *
183  *****/
184
185 /*****
186  *
187  *****/

```

```

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(\* **PalBtnActionCodeback\_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.

Copyright (c) 2018-2019 ARM Ltd. All Rights Reserved.

Copyright (c) 2019-2020 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 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
34  /*****
35  Data Types
36  *****/
37
38  typedef enum
39  {
40      PAL_BTN_STATE_UNINIT = 0,
41      PAL_BTN_STATE_ERROR = 0,
42      PAL_BTN_STATE_READY
43  } PalBtnState_t;
44
45  typedef enum
46  {
47      PAL_BTN_POS_INVALID,
48      PAL_BTN_POS_DOWN,
49      PAL_BTN_POS_UP
50  } PalBtnPos_t;
51
52  typedef void (*PalBtnActionCback_t)(uint8_t btnId, PalBtnPos_t state);
53
54  enum
55  {
56      PAL_BTN_AUDIO_PLAY = 0x80,
57      PAL_BTN_AUDIO_FWD,
58      PAL_BTN_AUDIO_RWD,
59      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(PalBtnActionCode_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 */
83
84 #ifdef __cplusplus
85 };
86 #endif
87
88 #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.

Copyright (c) 2018-2019 Arm Ltd. All Rights Reserved.

Copyright (c) 2019 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 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
34
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 */
57
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 [PalCodecStreamParam\\_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 compld, uint16\_t vsCodecId, `PalAudioDir_t` dir)
- bool\_t **PalCodecReadLocalSupportedControllerDelay** (uint8\_t codingFmt, uint16\_t compld, uint16\_t vsCodecId, `PalAudioDir_t` dir, uint32\_t \*pMinDly, uint32\_t \*pMaxDly)
- bool\_t **PalCodecConfigureDataPath** (`PalAudioDir_t` dir, uint8\_t dataPathId)
- void **PalCodecAmplnit** (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.

Copyright (c) 2019-2020 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 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
33 /*****
34 Data Types
35 *****/
36
37 typedef enum
38 {
39     PAL_CODEC_DIR_INPUT    = 0,
40     PAL_CODEC_DIR_OUTPUT   = 1
41 } PalAudioDir_t;
42
43 typedef enum
44 {
45     PAL_CODEC_CH_LEFT      = 0,
46     PAL_CODEC_CH_RIGHT     = 1,
47     AUDIO_NUM_CH
48 } PalAudioChan_t;
49
50 enum
51 {
52     PAL_CODEC_CH_LEFT_BIT  = (1 << PAL_CODEC_CH_LEFT),
53     PAL_CODEC_CH_RIGHT_BIT = (1 << PAL_CODEC_CH_RIGHT)
54 };
55
56 typedef struct
57 {
58     uint8_t codecId;
59 } AudioStdCodecInfo_t;
60
61 typedef struct
62 {
63     uint16_t compId;
64     uint16_t codecId;
65 } AudioVsCodecInfo_t;
66
67 typedef void (*PalCodecDataReady_t)(uint16_t id);
68
69 typedef struct
70 {
71     PalAudioDir_t dir;
72     uint16_t chMask;
73     uint32_t intervalUsec;
74     uint32_t pktCtr;
75     PalCodecDataReady_t rdyCbck;
76 } PalCodecStreamParam_t;
77
78 /*****
79 Function Declarations
80 *****/
81
82 /* Codec Information */
83 void PalCodecReadLocalSupportedCodecs(uint8_t *pNumStd, AudioStdCodecInfo_t stdCodecs[],
84                                         uint8_t *pNumVs, AudioVsCodecInfo_t vsCodecs[]);
85 bool_t PalCodecReadLocalSupportedCodecCapabilities(uint8_t codingFmt, uint16_t compId, uint16_t
86 vsCodecId, PalAudioDir_t dir);
87 bool_t PalCodecReadLocalSupportedControllerDelay(uint8_t codingFmt, uint16_t compId, uint16_t vsCodecId,
88 PalAudioDir_t dir,
89                                         uint32_t *pMinDly, uint32_t *pMaxDly);
90 bool_t PalCodecConfigureDataPath(PalAudioDir_t dir, uint8_t dataPathId);
91
92 /* Audio Amplifier */
93 void PalCodecAmpInit(void);
94 uint8_t PalCodecAmpGetVol(void);
95 void PalCodecAmpVolumeUp(void);
96 void PalCodecAmpVolumeDown(void);
97 void PalCodecAmpMute(void);
98 void PalCodecAmpUnmute(void);
99
100 /* Data Path */
101 void PalCodecDataInit(void);

```

```

112 bool_t PalCodecDataStartStream(uint16_t id, PalCodecStreamParam_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 */
117
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.

Copyright (c) 2018-2019 ARM Ltd. All Rights Reserved.

Copyright (c) 2019-2020 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 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
64 typedef enum
65 {
66     PAL_CRYPTO_STATE_UNINIT = 0,
67     PAL_CRYPTO_STATE_ERROR  = 0,
68     PAL_CRYPTO_STATE_READY
69 } PalCryptoState_t;
70
71 typedef struct
72 {
73     /* SK placed here for 32-bit alignment. */
74     uint8_t      sk[PAL_CRYPTO_LL_KEY_LEN];
75     uint8_t      iv[PAL_CRYPTO_LL_IV_LEN];
76     bool_t       enaEncrypt;
77     bool_t       enaDecrypt;
78     bool_t       enaAuth;
79     uint8_t      nonceMode;
80     uint16_t     *pEventCounter;
81     uint64_t     *pTxPktCounter;
82     uint64_t     *pRxPktCounter;
83     uint8_t      dir;
84     uint8_t      type;
85     void         *pEncryptCtx;
86     void         *pDecryptCtx;
87 } PalCryptoEnc_t;
88
89 /*****
90  Function Declarations
91 *****/
92
93 /* Initialization */
94 void PalCryptoInit(void);
95 void PalCryptoDeInit(void);
96
97 /* Key generation */
98 void PalCryptoGenerateP256KeyPair(const uint8_t *pPrivKey, uint8_t *pPubKey);
99 void PalCryptoGenerateDhKey(const uint8_t *pPubKey, const uint8_t *pPrivKey, uint8_t *pDhKey);
100 bool_t PalCryptoValidatePublicKey(const uint8_t *pPubKey, bool_t generateKey);
101 void PalCryptoGenerateRandomNumber(uint8_t *pBuf, uint8_t len);
102
103 /* CCM */
104 uint32_t PalCryptoCcmDec(const uint8_t *pKey, uint8_t *pNonce, uint8_t *pCypherText, uint16_t textLen,
105                         uint8_t *pClear, uint16_t clearLen, uint8_t *pMic, uint8_t micLen,
106                         uint8_t *pResult, uint8_t handlerId, uint16_t param, uint8_t event);
107 void PalCryptoCcmEnc(const uint8_t *pKey, uint8_t *pNonce, uint8_t *pPlainText, uint16_t textLen,
108                     uint8_t *pClear, uint16_t clearLen, uint8_t micLen, uint8_t *pResult,
109                     uint8_t handlerId, uint16_t param, uint8_t event);
110
111 /* Crypto AES */
112 void PalCryptoAesEcb(const uint8_t *pKey, uint8_t *pOut, const uint8_t *pIn);
113 void PalCryptoAesCmac(const uint8_t *pKey, uint8_t *pOut, const uint8_t *pIn, uint16_t len);
114 void PalCryptoAesEnable(PalCryptoEnc_t *pEnc, uint8_t id, uint8_t localDir);
115 bool_t PalCryptoAesCcmEncrypt(PalCryptoEnc_t *pEnc, uint8_t *pHdr, uint8_t *pBuf, uint8_t *pMic);
116 bool_t PalCryptoAesCcmDecrypt(PalCryptoEnc_t *pEnc, uint8_t *pBuf);
117 void PalCryptoSetEncryptPacketCount(PalCryptoEnc_t *pEnc, uint64_t pktCnt);
118 void PalCryptoSetDecryptPacketCount(PalCryptoEnc_t *pEnc, uint64_t pktCnt);
119
120 /* PAL_CRYPTO */
121
122 #ifdef __cplusplus
123 };
124 #endif
125
126 #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.

Copyright (c) 2019-2020 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 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
32 /*****
33 Data Types
34 *****/
35
36 enum
37 {
38     /* System signals. */
39     PAL_LED_ID_CPU_ACTIVE    = 0x30,
40     PAL_LED_ID_ERROR        = 0x31,
41 };
42
43 /*****
44 Function Declarations
45 *****/
46
47 /* Initialization */
48 void PalLedInit(void);
49 void PalLedDeInit(void);
50
51 /* Control and Status */
52 void PalLedOn(uint8_t id);
53 void PalLedOff(uint8_t id);
54     /* PAL_LED */
55
56 #ifdef __cplusplus
57 };
58 #endif
59
60 #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.

Copyright (c) 2018 Arm Ltd. All Rights Reserved.

Copyright (c) 2019-2020 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 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
34 /*****
35 Macros
36 *****/
37
38 #ifndef PAL_MAX_RTC_COUNTER_VAL
39 #define PAL_MAX_RTC_COUNTER_VAL (0x00FFFFFF)
40 #endif
41
42 #define PAL_RTC_TICKS_PER_SEC (32768) /* RTC ticks per second (with prescaler) */
43
44 typedef void (*palRtcIrqCback_t) (void);
45

```

```

52 /*****
53  Type Definitions
54 *****/
55
56 typedef enum
57 {
58     PAL_RTC_STATE_UNINIT = 0,
59     PAL_RTC_STATE_ERROR  = 0,
60     PAL_RTC_STATE_READY  = 1
61 } PalRtcState_t;
62
63 /*****
64  Function Declarations
65 *****/
66
67 /* Initialization */
68 void PalRtcInit(void);
69
70 /* Control and Status */
71 void PalRtcEnableCompareIrq(uint8_t channelId);
72 void PalRtcDisableCompareIrq(uint8_t channelId);
73 uint32_t PalRtcCounterGet(void);
74 void PalRtcCompareSet(uint8_t channelId, uint32_t value);
75 PalRtcState_t PalRtcGetState(void);
76
77 /* PAL_RTC */
78
79 #ifdef __cplusplus
80 };
81 #endif
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.

Copyright (c) 2016-2019 Arm Ltd. All Rights Reserved.

Copyright (c) 2019-2020 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 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 */
78
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.

Copyright (c) 2016-2019 Arm Ltd. All Rights Reserved.

Copyright (c) 2019-2020 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.



## 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
34  /*****
35  Macros
36  *****/
37
38  typedef enum
39  {
40      PAL_TIMER_STATE_UNINIT = 0,
41      PAL_TIMER_STATE_ERROR = 0,
42      PAL_TIMER_STATE_READY,
43      PAL_TIMER_STATE_BUSY
44  } PalTimerState_t;
45
46  /*****
47  Data Types
48  *****/
49
50  typedef void (*PalTimerCompCbcbk_t)(void);
51
52  /*****
53  Function Declarations
54  *****/
55
56  /* Initialization */
57  void PalTimerInit(PalTimerCompCbcbk_t expCbcbk);
58  void PalTimerDeInit(void);
59
60  /* Control and Status */
61  PalTimerState_t PalTimerGetState(void);
62  void PalTimerStart(uint32_t expUsec);
63  void PalTimerStop(void);
64  uint32_t PalTimerGetCurrentTime(void);
65  uint32_t PalTimerGetExpTime(void);
66  void PalTimerSleep(uint32_t expUsec);
67  void PalTimerRestore(uint32_t schTime);
68  /* PAL_TIMER */
69
70  #ifdef __cplusplus
71  };
72  #endif
73
74  #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(\* **PalUartCompCbback\_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.

Copyright (c) 2018 ARM Ltd. All Rights Reserved.

Copyright (c) 2019-2020 Packetcraft, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 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
34 /*****
35     Data Types
36 *****/
37
38 typedef void (*PalUartCompCback_t)(void);
39
40 typedef struct
41 {
42     PalUartCompCback_t rdCback;
43     PalUartCompCback_t wrCback;
44     uint32_t baud;
45     bool_t hwFlow;
46 } PalUartConfig_t;
47
48 typedef enum
49 {
50     PAL_UART_STATE_UNINIT = 0,
51     PAL_UART_STATE_ERROR = 0,
52     PAL_UART_STATE_READY = 1,
53     PAL_UART_STATE_BUSY = 2,
54 } PalUartState_t;
55
56 typedef enum
57 {
58     PAL_UART_ID_USER = 0,
59     PAL_UART_ID_CHCI = 1,
60     PAL_UART_ID_TERMINAL = 2,
61     PAL_UART_ID_MAX
62 } PalUartId_t;
63
64 /*****
65     Function Declarations
66 *****/
67
68 /* Initialization */
69 void PalUartInit(PalUartId_t id, const PalUartConfig_t *pCfg);
70 void PalUartDeInit(PalUartId_t id);
71
72 /* Control and Status */
73 PalUartState_t PalUartGetState(PalUartId_t id);
74
75 /* Data Transfer */
76 void PalUartReadData(PalUartId_t id, uint8_t *pData, uint16_t len);
77 void PalUartWriteData(PalUartId_t id, const uint8_t *pData, uint16_t len);
78
79 /* PAL_UART */
80
81 #ifdef __cplusplus
82 };
83 #endif
84
85 #endif /* PAL_UART_H */

```



# Index

/github/workspace/Libraries/Cordio/platform/include/pal\_bb.h, PalBbBleChan\_t, 31  
43  
/github/workspace/Libraries/Cordio/platform/include/pal\_bb\_ble.h,  
45, 48 PalCodecSreamParam\_t, 38  
/github/workspace/Libraries/Cordio/platform/include/pal\_btn.h, PalCryptoEnc\_t, 39  
50, 51 dueUsec  
/github/workspace/Libraries/Cordio/platform/include/pal\_cfg.h, PalBbBleDataParam\_t, 33  
52, 53  
/github/workspace/Libraries/Cordio/platform/include/pal\_codec.h, enaAuth  
53, 55 PalCryptoEnc\_t, 39  
/github/workspace/Libraries/Cordio/platform/include/pal\_crypto.h, enaDecrypt  
56, 57 PalCryptoEnc\_t, 39  
/github/workspace/Libraries/Cordio/platform/include/pal\_led.h, enaEncrypt  
59, 60 PalCryptoEnc\_t, 39  
/github/workspace/Libraries/Cordio/platform/include/pal\_rtc.h, enc  
60, 61 PalBbBleChan\_t, 31  
/github/workspace/Libraries/Cordio/platform/include/pal\_sys.h, hwFlow  
62, 63 PalUartConfig\_t, 41  
/github/workspace/Libraries/Cordio/platform/include/pal\_timer.h,  
64, 65  
/github/workspace/Libraries/Cordio/platform/include/pal\_uart.h, ifsMode  
65, 67 PalBbBleOpParam\_t, 34  
ifsTime  
PalBbBleOpParam\_t, 34  
initTxPhyOptions  
PalBbBleChan\_t, 31  
intervalUsec  
PalCodecSreamParam\_t, 38  
iv  
PalCryptoEnc\_t, 39  
len  
PalBbBleTxBufDesc\_t, 35  
LL\_ENABLE\_TESTER  
pal\_bb\_ble.h, 47  
maxScanPeriodMsec  
PalBbCfg\_t, 36  
nonceMode  
PalCryptoEnc\_t, 40  
opType  
PalBbBleChan\_t, 32  
pal\_bb\_ble.h  
LL\_ENABLE\_TESTER, 47  
PAL\_BB\_NONCE\_MODE\_EXT16\_CNTR, 47  
PAL\_BB\_NONCE\_MODE\_EXT64\_CNTR, 47  
PAL\_BB\_NONCE\_MODE\_PKT\_CNTR, 47  
PAL\_BB\_TYPE\_ACL, 48  
accAddr  
PalBbBleChan\_t, 31  
AUDIO\_NUM\_CH  
PAL\_CODEC, 11  
AudioStdCodecInfo\_t, 29  
codecId, 29  
AudioVsCodecInfo\_t, 30  
codecId, 30  
compld, 30  
baud  
PalUartConfig\_t, 41  
BbTimerBoundaryUsec  
PalBbCfg\_t, 36  
chanIdx  
PalBbBleChan\_t, 31  
chMask  
PalCodecSreamParam\_t, 37  
clkPpm  
PalBbCfg\_t, 36  
codecId  
AudioStdCodecInfo\_t, 29  
AudioVsCodecInfo\_t, 30  
compld  
AudioVsCodecInfo\_t, 30  
crclInit

PAL\_BB\_TYPE\_BIS, 48  
 PAL\_BB\_TYPE\_CIS, 48  
 PAL\_BB\_BLE\_CHAN, 13  
   PalBbBleSetChannelParam, 14  
 PAL\_BB\_BLE\_DATA, 14  
   PAL\_BB\_IFS\_MODE\_CLR, 15  
   PAL\_BB\_IFS\_MODE\_SAME\_ABS, 15  
   PAL\_BB\_IFS\_MODE\_TOGGLE\_TIFS, 15  
   PalBbBleCancelData, 15  
   PalBbBleCancelTifs, 16  
   PalBbBleRxData, 16  
   PalBbBleRxTifsData, 16  
   PalBbBleSetDataParams, 16  
   PalBbBleSetOpParams, 17  
   PalBbBleTxData, 17  
   PalBbBleTxTifsData, 17  
   PalBbIfsMode\_t, 15  
 PAL\_BB\_BLE\_INIT, 18  
   PalBbBleDisable, 18  
   PalBbBleEnable, 18  
   PalBbBleInit, 18  
 PAL\_BB\_BLE\_TEST, 19  
   PalBbBleEnableDataWhitening, 19  
   PalBbBleEnablePrbs15, 19  
   PalBbBleInlineEncryptDecryptSetDirection, 20  
   PalBbBleInlineEncryptSetPacketCount, 20  
   PalBbBleLowPower, 20  
 PAL\_BB\_CLOCK, 25  
   PalBbGetCurrentTime, 25  
   PalBbGetTimestamp, 25  
   PalBbRegisterProtIrq, 26  
   PalBbSetProtId, 26  
 PAL\_BB\_IFS\_MODE\_CLR  
   PAL\_BB\_BLE\_DATA, 15  
 PAL\_BB\_IFS\_MODE\_SAME\_ABS  
   PAL\_BB\_BLE\_DATA, 15  
 PAL\_BB\_IFS\_MODE\_TOGGLE\_TIFS  
   PAL\_BB\_BLE\_DATA, 15  
 PAL\_BB\_INIT, 23  
   PalBbDisable, 24  
   PalBbEnable, 24  
   PalBbInit, 24  
   PalBbLoadCfg, 24  
   PalBbRestore, 24  
 PAL\_BB\_NONCE\_MODE\_EXT16\_CNTR  
   pal\_bb\_ble.h, 47  
 PAL\_BB\_NONCE\_MODE\_EXT64\_CNTR  
   pal\_bb\_ble.h, 47  
 PAL\_BB\_NONCE\_MODE\_PKT\_CNTR  
   pal\_bb\_ble.h, 47  
 PAL\_BB\_TYPE\_ACL  
   pal\_bb\_ble.h, 48  
 PAL\_BB\_TYPE\_BIS  
   pal\_bb\_ble.h, 48  
 PAL\_BB\_TYPE\_CIS  
   pal\_bb\_ble.h, 48  
 PAL\_BTN\_AUDIO\_FWD  
   PAL\_BUTTON, 8  
 PAL\_BTN\_AUDIO\_MUTE  
   PAL\_BUTTON, 8  
 PAL\_BTN\_AUDIO\_PLAY  
   PAL\_BUTTON, 8  
 PAL\_BTN\_AUDIO\_RWD  
   PAL\_BUTTON, 8  
 PAL\_BTN\_AUDIO\_VOL\_DN  
   PAL\_BUTTON, 8  
 PAL\_BTN\_AUDIO\_VOL\_UP  
   PAL\_BUTTON, 8  
 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\_AUDIO\_FWD, 8  
   PAL\_BTN\_AUDIO\_MUTE, 8  
   PAL\_BTN\_AUDIO\_PLAY, 8  
   PAL\_BTN\_AUDIO\_RWD, 8  
   PAL\_BTN\_AUDIO\_VOL\_DN, 8  
   PAL\_BTN\_AUDIO\_VOL\_UP, 8  
   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, 8  
   PalBtnState\_t, 8  
 PAL\_CFG, 8  
   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\_CODEC, 10  
   AUDIO\_NUM\_CH, 11  
   PAL\_CODEC\_CH\_LEFT, 11  
   PAL\_CODEC\_CH\_RIGHT, 11

PAL\_CODEC\_DIR\_INPUT, [11](#)  
PAL\_CODEC\_DIR\_OUTPUT, [11](#)  
PalAudioChan\_t, [11](#)  
PalAudioDir\_t, [11](#)  
PAL\_CODEC\_CH\_LEFT  
PAL\_CODEC, [11](#)  
PAL\_CODEC\_CH\_RIGHT  
PAL\_CODEC, [11](#)  
PAL\_CODEC\_DIR\_INPUT  
PAL\_CODEC, [11](#)  
PAL\_CODEC\_DIR\_OUTPUT  
PAL\_CODEC, [11](#)  
PAL\_CRYPTO, [11](#)  
PAL\_CRYPTO\_LL\_DATA\_MIC\_LEN, [12](#)  
PAL\_CRYPTO\_LL\_IV\_LEN, [12](#)  
PAL\_CRYPTO\_LL\_KEY\_LEN, [13](#)  
PAL\_CRYPTO\_STATE\_ERROR, [13](#)  
PAL\_CRYPTO\_STATE\_READY, [13](#)  
PAL\_CRYPTO\_STATE\_UNINIT, [13](#)  
PalCryptoState\_t, [13](#)  
PAL\_CRYPTO\_LL\_DATA\_MIC\_LEN  
PAL\_CRYPTO, [12](#)  
PAL\_CRYPTO\_LL\_IV\_LEN  
PAL\_CRYPTO, [12](#)  
PAL\_CRYPTO\_LL\_KEY\_LEN  
PAL\_CRYPTO, [13](#)  
PAL\_CRYPTO\_STATE\_ERROR  
PAL\_CRYPTO, [13](#)  
PAL\_CRYPTO\_STATE\_READY  
PAL\_CRYPTO, [13](#)  
PAL\_CRYPTO\_STATE\_UNINIT  
PAL\_CRYPTO, [13](#)  
PAL\_LED, [21](#)  
PAL\_LED\_ID\_CPU\_ACTIVE, [22](#)  
PAL\_LED\_ID\_ERROR, [22](#)  
PAL\_LED\_ID\_CPU\_ACTIVE  
PAL\_LED, [22](#)  
PAL\_LED\_ID\_ERROR  
PAL\_LED, [22](#)  
PAL\_RTC, [22](#)  
PAL\_RTC\_STATE\_ERROR, [23](#)  
PAL\_RTC\_STATE\_READY, [23](#)  
PAL\_RTC\_STATE\_UNINIT, [23](#)  
PalRtcState\_t, [23](#)  
PAL\_RTC\_STATE\_ERROR  
PAL\_RTC, [23](#)  
PAL\_RTC\_STATE\_READY  
PAL\_RTC, [23](#)  
PAL\_RTC\_STATE\_UNINIT  
PAL\_RTC, [23](#)  
PAL\_SYS, [9](#)  
PAL\_TIMER, [20](#)  
PAL\_TIMER\_STATE\_BUSY, [21](#)  
PAL\_TIMER\_STATE\_ERROR, [21](#)  
PAL\_TIMER\_STATE\_READY, [21](#)  
PAL\_TIMER\_STATE\_UNINIT, [21](#)  
PalTimerState\_t, [21](#)  
PAL\_TIMER\_STATE\_BUSY  
PAL\_TIMER, [21](#)  
PAL\_TIMER\_STATE\_ERROR  
PAL\_TIMER, [21](#)  
PAL\_TIMER\_STATE\_READY  
PAL\_TIMER, [21](#)  
PAL\_TIMER\_STATE\_UNINIT  
PAL\_TIMER, [21](#)  
PAL\_UART, [27](#)  
PAL\_UART\_ID\_CHCI, [27](#)  
PAL\_UART\_ID\_MAX, [27](#)  
PAL\_UART\_ID\_TERMINAL, [27](#)  
PAL\_UART\_ID\_USER, [27](#)  
PAL\_UART\_STATE\_BUSY, [28](#)  
PAL\_UART\_STATE\_ERROR, [28](#)  
PAL\_UART\_STATE\_READY, [28](#)  
PAL\_UART\_STATE\_UNINIT, [28](#)  
PalUartId\_t, [27](#)  
PalUartState\_t, [28](#)  
PAL\_UART\_ID\_CHCI  
PAL\_UART, [27](#)  
PAL\_UART\_ID\_MAX  
PAL\_UART, [27](#)  
PAL\_UART\_ID\_TERMINAL  
PAL\_UART, [27](#)  
PAL\_UART\_ID\_USER  
PAL\_UART, [27](#)  
PAL\_UART\_STATE\_BUSY  
PAL\_UART, [28](#)  
PAL\_UART\_STATE\_ERROR  
PAL\_UART, [28](#)  
PAL\_UART\_STATE\_READY  
PAL\_UART, [28](#)  
PAL\_UART\_STATE\_UNINIT  
PAL\_UART, [28](#)  
PalAudioChan\_t  
PAL\_CODEC, [11](#)  
PalAudioDir\_t  
PAL\_CODEC, [11](#)  
PalBbBleCancelData  
PAL\_BB\_BLE\_DATA, [15](#)  
PalBbBleCancelTifs  
PAL\_BB\_BLE\_DATA, [16](#)  
PalBbBleChan\_t, [30](#)  
accAddr, [31](#)  
chanIdx, [31](#)  
crclnit, [31](#)  
enc, [31](#)  
initTxPhyOptions, [31](#)  
opType, [32](#)  
peerRxStableModIdx, [32](#)  
peerTxStableModIdx, [32](#)  
rxPhy, [32](#)  
tifsTxPhyOptions, [32](#)  
txPhy, [32](#)  
txPower, [32](#)  
PalBbBleDataParam\_t, [33](#)  
dueUsec, [33](#)  
rxCback, [33](#)

- rxTimeoutUsec, 33
- txCback, 34
- PalBbBleDisable
  - PAL\_BB\_BLE\_INIT, 18
- PalBbBleEnable
  - PAL\_BB\_BLE\_INIT, 18
- PalBbBleEnableDataWhitening
  - PAL\_BB\_BLE\_TEST, 19
- PalBbBleEnablePrbs15
  - PAL\_BB\_BLE\_TEST, 19
- PalBbBleInit
  - PAL\_BB\_BLE\_INIT, 18
- PalBbBleInlineEncryptDecryptSetDirection
  - PAL\_BB\_BLE\_TEST, 20
- PalBbBleInlineEncryptSetPacketCount
  - PAL\_BB\_BLE\_TEST, 20
- PalBbBleLowPower
  - PAL\_BB\_BLE\_TEST, 20
- PalBbBleOpParam\_t, 34
  - ifsMode, 34
  - ifsTime, 34
  - plfsChan, 35
- PalBbBleRxData
  - PAL\_BB\_BLE\_DATA, 16
- PalBbBleRxTifsData
  - PAL\_BB\_BLE\_DATA, 16
- PalBbBleSetChannelParam
  - PAL\_BB\_BLE\_CHAN, 14
- PalBbBleSetDataParams
  - PAL\_BB\_BLE\_DATA, 16
- PalBbBleSetOpParams
  - PAL\_BB\_BLE\_DATA, 17
- PalBbBleTxBufDesc\_t, 35
  - len, 35
  - pBuf, 35
- PalBbBleTxData
  - PAL\_BB\_BLE\_DATA, 17
- PalBbBleTxTifsData
  - PAL\_BB\_BLE\_DATA, 17
- PalBbCfg\_t, 36
  - BbTimerBoundaryUsec, 36
  - clkPpm, 36
  - maxScanPeriodMsec, 36
  - rfSetupDelayUsec, 37
  - schSetupDelayUsec, 37
- PalBbDisable
  - PAL\_BB\_INIT, 24
- PalBbEnable
  - PAL\_BB\_INIT, 24
- PalBbGetCurrentTime
  - PAL\_BB\_CLOCK, 25
- PalBbGetTimestamp
  - PAL\_BB\_CLOCK, 25
- PalBbIfsMode\_t
  - PAL\_BB\_BLE\_DATA, 15
- PalBbInit
  - PAL\_BB\_INIT, 24
- PalBbLoadCfg
  - PAL\_BB\_INIT, 24
- PalBbRegisterProtIrq
  - PAL\_BB\_CLOCK, 26
- PalBbRestore
  - PAL\_BB\_INIT, 24
- PalBbSetProtId
  - PAL\_BB\_CLOCK, 26
- PalBtnPos\_t
  - PAL\_BUTTON, 8
- PalBtnState\_t
  - PAL\_BUTTON, 8
- PalCfgId\_t
  - PAL\_CFG, 9
- PalCodecStreamParam\_t, 37
  - chMask, 37
  - dir, 38
  - intervalUsec, 38
  - pktCtr, 38
  - rdyCback, 38
- PalCryptoEnc\_t, 38
  - dir, 39
  - enaAuth, 39
  - enaDecrypt, 39
  - enaEncrypt, 39
  - iv, 39
  - nonceMode, 40
  - pDecryptCtx, 40
  - pEncryptCtx, 40
  - pEventCounter, 40
  - pRxPktCounter, 40
  - pTxPktCounter, 40
  - sk, 40
  - type, 40
- PalCryptoState\_t
  - PAL\_CRYPT, 13
- PalRtcState\_t
  - PAL\_RTC, 23
- PalTimerState\_t
  - PAL\_TIMER, 21
- PalUartConfig\_t, 41
  - baud, 41
  - hwFlow, 41
  - rdCback, 41
  - wrCback, 41
- PalUartId\_t
  - PAL\_UART, 27
- PalUartState\_t
  - PAL\_UART, 28
- pBuf
  - PalBbBleTxBufDesc\_t, 35
- pDecryptCtx
  - PalCryptoEnc\_t, 40
- peerRxStableModIdx
  - PalBbBleChan\_t, 32
- peerTxStableModIdx
  - PalBbBleChan\_t, 32
- pEncryptCtx
  - PalCryptoEnc\_t, 40



- pEventCounter
  - PalCryptoEnc\_t, [40](#)
- plfsChan
  - PalBbBleOpParam\_t, [35](#)
- pktCtr
  - PalCodecSreamParam\_t, [38](#)
- pRxPktCounter
  - PalCryptoEnc\_t, [40](#)
- pTxPktCounter
  - PalCryptoEnc\_t, [40](#)
- rdCback
  - PalUartConfig\_t, [41](#)
- rdyCback
  - PalCodecSreamParam\_t, [38](#)
- rfSetupDelayUsec
  - PalBbCfg\_t, [37](#)
- rxCback
  - PalBbBleDataParam\_t, [33](#)
- rxPhy
  - PalBbBleChan\_t, [32](#)
- rxTimeoutUsec
  - PalBbBleDataParam\_t, [33](#)
- schSetupDelayUsec
  - PalBbCfg\_t, [37](#)
- sk
  - PalCryptoEnc\_t, [40](#)
- tifsTxPhyOptions
  - PalBbBleChan\_t, [32](#)
- txCback
  - PalBbBleDataParam\_t, [34](#)
- txPhy
  - PalBbBleChan\_t, [32](#)
- txPower
  - PalBbBleChan\_t, [32](#)
- type
  - PalCryptoEnc\_t, [40](#)
- wrCback
  - PalUartConfig\_t, [41](#)