Qualcomm Technologies International, Ltd.

Confidential and Proprietary – Qualcomm Technologies International, Ltd.

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or websites to: DocCtrlAgent@qualcomm.com.

Restricted Distribution: Not to be distributed to anyone who is not an employee of either Qualcomm Technologies International, Ltd. or its affiliated companies without the express approval of Qualcomm Configuration Management.

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies International, Ltd.

µEnergy is a product of Qualcomm Technologies International, Ltd. Other Qualcomm products referenced herein are products of Qualcomm Technologies International, Ltd. or Qualcomm Technologies, Inc. or its other subsidiaries.

Qualcomm is a trademark of Qualcomm Incorporated, registered in the United States and other countries. µEnergy is a trademark of Qualcomm Technologies International, Ltd., registered in the United States and other countries. Other product and brand names may be trademarks or registered trademarks of their respective owners.

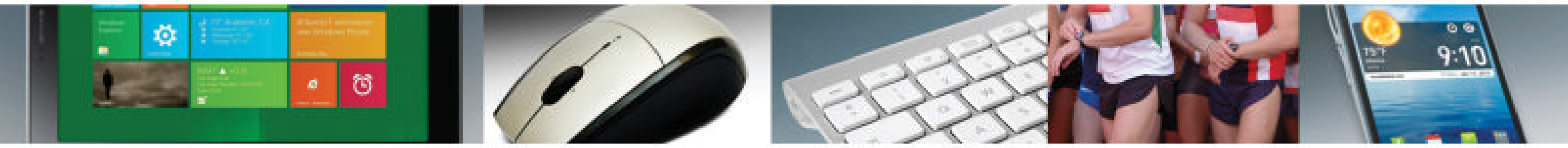
This technical data may be subject to U.S. and international export, re-export, or transfer (“export”) laws. Diversion contrary to U.S. and international law is strictly prohibited.

Qualcomm Technologies International, Ltd. (formerly known as Cambridge Silicon Radio Limited) is a company registered in England and Wales with a registered office at: Churchill House, Cambridge Business Park, Cowley Road, Cambridge, CB4 0WZ, United Kingdom. Registered Number: 3665875 | VAT number: GB787433096.

© 2016 Qualcomm Technologies International, Ltd. All rights reserved.

Push every boundary.™

CSR µEnergy™



Firmware Library Documentation

Issue 3

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Document History

Revision	Date	Change Reason
1	01 JUL 14	Original publication of this document.
2	12 JAN 16	Updated to SDK 2.6.
3	16 JUN 16	Updated to SDK 2.6.1.

Trademarks, Patents and Licenses

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by QTIL and/or its affiliates.

Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to QTIL.

Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by QTIL or its affiliates.

QTIL reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, QTIL cannot accept responsibility for any errors.

Use of this document is permissible only in accordance with the applicable QTIL licence agreement.

Safety-critical Applications

QTIL's products are not designed for use in safety-critical devices or systems such as those relating to: (i) life support; (ii) nuclear power; and/or (iii) civil aviation applications, or other applications where injury or loss of life could be reasonably foreseeable as a result of the failure of a product. The customer agrees not to use QTIL's products (or supply QTIL's products for use) in such devices or systems.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Contents

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

25.1.10 I2cRawRead 145
25.1.11 I2cRawReadByte 146
25.1.12 I2cRawTerminate 146
25.1.13 I2cRawWrite 147
25.1.14 I2cRawWriteByte 147
25.1.15 I2cReady 147
25.1.16 I2cReset 148
25.2 Enumerations 148
25.2.1 enum i2c_command 148
25.3 Defines 149
25.3.1 I2C_EEPROM_POLLED_WRITE_CYCLE 149
25.3.2 I2C_POWER_PIO_UNDEFINED 149
25.3.3 I2C_RESERVED_PIO 150
25.3.4 I2C_SCL_100KBPS_HIGH_PERIOD 150
25.3.5 I2C_SCL_100KBPS_LOW_PERIOD 150
25.3.6 I2C_SCL_400KBPS_HIGH_PERIOD 150
25.3.7 I2C_SCL_400KBPS_LOW_PERIOD 150
25.3.8 I2cRawRestart 151
25.3.9 I2cRawSendAck 151
25.3.10 I2cRawSendNack 151
25.3.11 I2cRawStart 151
25.3.12 I2cRawStop 151
25.3.13 I2cRawWaitAck 152
26 SPI Serial Interface 153
26.1 Functions 153
26.1.1 SpiConfigReadRegisterDelay 153
26.1.2 SpiConfigWriteIntervalDelay 153
26.1.3 SpiConfigWriteTerminationDelay 154
26.1.4 SpiFlashEraseBlock 154
26.1.5 SpiFlashEraseWaitComplete 155
26.1.6 SpiFlashInit 155
26.1.7 SpiFlashRead 156
26.1.8 SpiFlashWrite 156
26.1.9 SpiInit 157
26.1.10 SpiRead 158
26.1.11 SpiReadByte 158
26.1.12 SpiReadRegister 159
26.1.13 SpiReadRegisterBurst 159
26.1.14 SpiWrite 160
26.1.15 SpiWriteByte 161
26.1.16 SpiWriteRegister 161
26.2 Enumerations 162
26.2.1 enum spi_erase_size 162
26.3 Defines 162
26.3.1 SPI_FLASH_DEFAULT_PIO 162
26.3.2 SPI_FLASH_POWER_PIO_UNDEFINED 162
26.3.3 SPI_NCS_PIO_UNDEFINED 162
27 UART 164
27.1 Functions 164
27.1.1 UartConfig 164
27.1.2 UartEnable 164
27.1.3 UartInit 165

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

1 Introduction

This document describes CSR µEnergy® firmware library APIs:

- Modules are described in Section 2 to Section 45.
- Data structures are described in Section 46.
- Reference files are described in Section 47.
- Fault codes are described in Appendix A.

1.1 Chip Support

The majority of APIs described in this document are supported by all chips in the CSR µEnergy product range. In cases where there are differences in support this is stated in the API.

When building a project, it is important to select the correct chip.

To select a chip:

1. Open the project in xIDE.
2. Go to the **Project** menu and select **Properties**.
3. In the left pane, select **Configuration Properties** > **Build System** > **General**.
4. Look for target hardware in the right pane.

Note:

Options include each supported chip family (e.g. CSR1011) and an additional option **Auto Detect**.

Auto Detect attempts to detect the chip of an attached development board. It works only if a device is available.

CSR µEnergy product differences include:

- CSR1000 and CSR1001
  - CSR 1001 has 11 PIOs and CSR1001 has 32 PIOs.
  - Both chips support the same set of APIs. Accessing unavailable PIOs on CSR1000 is treated as a null operation (no error is returned).
- CSR1010 and CSR1011
  - CSR1010 has 11 PIOs and CSR1011 has 32 PIOs.
  - APIs that access these PIOs are identical between the two chips and accessing unavailable PIOs on the CSR1010 does not result in an error.
  - CSR101x chips include two hardware quadrature decoders. There are APIs to configure and use these decoders that are not available on CSR100x chips.
  - CSR101x chips contain substantially more of the CSR µEnergy firmware library in ROM. This has freed up a large amount of extra RAM for application use (code and/or data).

2 Modules

The CSR µEnergy firmware library contains the following API modules:

- Application
- Bluetooth v4.1 (Bluetooth Smart) Protocol Stack
  - Generic Attribute Profile (GATT)
    - GATT Common
      - GATT UUIDs
      - GATT Flags for Write Request
    - GATT Client
    - GATT Server
      - GATT Database Helper Macros
  - Security Manager
  - Generic Access Profile (GAP)
  - Link Supervisor
    - LS Advscan Interface
    - LS to APP Interface
- Programmable Input/Output
  - Analogue IO
  - Digital PIO
  - Edge Capture Mode
  - Pulse Width Modulation
  - PIO Controller
  - Quadrature Decoders
- Serial Interfaces
  - I²C Serial Interface
  - SPI Serial Interface
  - UART
    - UART Baudrate
    - UART Configuration
- Memory Management
  - C Standard Library APIs
  - Persistent Memory
  - Configuration Store
    - Configuration Store Keys
- Non-Volatile Memory
- System
  - Battery
  - Build Identifier
  - Panic
    - Firmware Library Fault Codes
  - Power Management
  - Random Numbers
  - Reset
  - Thermometer
  - Time
  - Timers
  - System-wide Constants
  - System-wide Status Codes
- Debug
- Production Test

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

3 Application

3.1 Functions

3.1.1 AppInit

Syntax

```
void AppInit (sleep_state last_sleep_state)
```

Description

Application function called after a power-on reset etc.

This user application function is called after a power-on reset (including after a firmware panic), after a wakeup from Hibernate or Dormant sleep states or after an HCI Reset has been requested.

The last sleep state is provided to the application in the parameter.

Note:

In the case of a power-on reset, this function is called after `AppPowerOnReset()`.

Parameters

Parameter	Description
`last_sleep_state`	-

Returns

Nothing.

3.1.2 AppPowerOnReset

Syntax

```
void AppPowerOnReset (void)
```

Description

Application function called just after a power-on reset etc.

This user application function is called just after a power-on reset (including after a firmware panic), after a wakeup from Hibernate or Dormant sleep states.

At the time this function is called, the last sleep state is not yet known.

Note:

This function should only contain code to be executed after a power-on reset or panic.

Parameters

Parameter	Description
`None`	-

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

Nothing.

3.1.3 AppProcessLmEvent

Syntax

```
bool AppProcessLmEvent (lm_event_code event_code, LM_EVENT_T *
event_data)
```

Description

Called whenever an LM-specific event is received by the system.

Parameters

Parameter	Description
`event_code`	Identifying which member of `LM_EVENT_T` union should be used to decode `event_data`
`event_data`	Pointer to event datastructure

Returns

Applications should invariably return TRUE.

Note:

A return of FALSE indicates that this event cannot be processed at this time and should stall the event queue. The Application will next be offered the same event when another event is added to the queue, which may be an indefinite time later. The event queue depth is relatively small so stalling the queue for an extended period may cause a fault condition to be raised, usually resulting in reboot.

3.1.4 AppProcessSystemEvent

Syntax

```
void AppProcessSystemEvent (sys_event_id id, void * data)
```

Description

Application function called on system event.

This user application function is called whenever a system event, such as a battery low notification, is received by the system. If the 'data' parameter is not NULL it points to a structure containing information relevant to the wakeup event (see individual events in the `sys_event_id` definition for more information).

Note:

Warning: This 'data' pointer is only valid for the duration of the call to `AppProcessSystemEvent()`. After the application returns from this function the data structure will become invalid. Therefore if the application needs to save any information for later processing it must copy it to a local variable.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
id	-
data	-

Returns

Nothing.

3.2 Defines

3.2.1 AppProcessLmEventPtr

Syntax

```
#define AppProcessLmEventPtr(x, y)
AppProcessLmEvent(x,y)
```

4 Bluetooth v4.1 (Bluetooth Smart) Protocol Stack

- Generic Attribute Profile (GATT)
- Security Manager
- Generic Access Profile (GAP)
- Link Supervisor

5 Generic Attribute Profile (GATT)

- GATT Common
- GATT Client
- GATT Server

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

6 GATT Common

- GATT UUIDs
- GATT Flags for Write Request

Aligned to the flags defined by ATT.

6.1 Functions

6.1.1 GattCancelConnectReq

Syntax

```
void GattCancelConnectReq (void)
```

Description

Cancel a connect request.

This function is used to cancel on-going BLE-U connection setup in master role or to stop connectable directed or undirected advertisements in Slave role.

Parameters

Parameter	Description
None	-

Returns

Nothing.

6.1.2 GattConnectReq

Syntax

```
void GattConnectReq (TYPED_BD_ADDR_T * bd_addr, uint16 flags)
```

Description

This function is used to map ATT fixed channel for BLE-U connection, for master role and slave role connections.

- **Master Role**

  In the master role, the device will attempt to establish a BLE-U connection towards the remote device if a connection doesn't already exist. The flags parameter is used to map onto the corresponding GAP Connection Procedures (see Vol.3 Part C Section 9.3 of the Bluetooth Specification) as follows:

  Directed Connection Establishment Procedure: 'flags' must be set to L2CAP_CONNECTION_MASTER_DIRECTED. 'bd_addr' is used to specify the address of the slave device to connect to.

  Auto Connection Establishment Procedure: 'flags' must be set to L2CAP_CONNECTION_MASTER_WHITELIST. 'bd_addr' is not used in this procedure and can be set to NULL. Prior to starting this procedure the application must have initialised the device whitelist using `LsAddWhiteListDevice()` etc.

  Selective Connection Establishment Procedure: the application currently needs to implement this procedure manually, by: (a) populating the whitelist; (b) scanning for devices in the whitelist using `LsStartStopScan()`; (c) stopping the scan when it gets an advert; (d) configuring connection setup parameters for that device using `LsSetNewConnectionParamReq()`; and then (e) performing the Directed Connection Establishment Procedure as described above.

  General Connection Establishment Procedure: the application currently needs to implement this procedure manually, by: (a) scanning for devices using `LsStartStopScan()`; (b) stopping the scan when it gets an advert from a device it wishes to connect to (which could be the first device, or only after a user has confirmed a connection); and then (c) performing the Directed Connection Establishment Procedure as described above.

  For all master connection setup procedures, the application can control the connection parameters by calling `LsSetNewConnectionParamReq()` prior to calling `GattConnectReq()`, even for GAP procedures that don't explicitly require such control.

- **Slave Role**

  In the slave role, the device will start Directed or Undirected Connectable advertising (subject to GAP connectable mode set by `GapSetMode()`) if a connection doesn't already exist. 'flags' must be set to L2CAP_CONNECTION_SLAVE. When using Undirected Connectable advertising, the application can set up the whitelist prior to calling this function and then set 'flags' to L2CAP_CONNECTION_SLAVE_WHITELIST instead.

  For the slave role in Directed Connectable mode, the peer (master) address to direct the advertising at must be set by calling `GapSetAdvAddress()` before calling `GattConnectReq()`. In this case 'bd_addr' must be set to NULL.

Parameters

Parameter	Description
`bd_addr`	BD_ADDR of the remote device, for the master role Directed Connection Establishment Procedure only
`flags`	Specify the connection type (master or slave, master procedure, use of whitelisting)

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

6.1.3 GattDisconnectReasonReq

Syntax

```
void GattDisconnectReasonReq (uint16 cid, ls_err
disconnect_reason)
```

Description

Terminates BLE-U connection, specifying a non-default reason code.

Un-maps ATT fixed channel and terminates BLE-U connection with remote device if not used for any other purpose.

This function terminates the connection with the HCI reason code supplied in the disconnect_reason parameter. It is the responsibility of the application to ensure that a valid reason code is used. In most cases it will be sufficient to use `GattDisconnectReq()` instead, for the default disconnect reason "Remote User Terminate Connection". However, this function can be used, for example, if the application wishes to disconnect after a request to a remote master to update connection parameters has timed out. In that case the error code "Unacceptable Connection Interval" must be used.

This function will generate a GATT_DISCONNECT_CFM event when it completes.

Parameters

Parameter	Description
`cid`	Connection identifier as received in `GATT_CONNECT_CFM_T` for established BLE-U connection.
`disconnect_reason`	HCI reason code

Returns

Nothing.

6.1.4 GattDisconnectReq

Syntax

```
void GattDisconnectReq (uint16 cid)
```

Description

Terminates BLE-U connection.

Un-maps ATT fixed channel and terminates BLE-U connection with remote device if not used for any other purpose.

This function terminates the connection with the HCI reason code "Remote User Terminate Connection". If the application needs to use an alternative reason code then it should call `GattDisconnectReasonReq()` instead.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Parameters

Parameter	Description
`cid`	Connection identifier as received in `GATT_CONNECT_CFM_T` for established BLE-U connection.

Returns

Nothing.

6.1.5 GattInit

Syntax

```
void GattInit (void)
```

Description

Initialisation function for GATT module.

Parameters

Parameter	Description
`None`	-

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

7 GATT UUIDs

These UUIDs are defined by the Bluetooth SIG. See the Bluetooth Assigned Numbers website (Generic Attribute Profile section) for more information see: https://www.bluetooth.org/technical/assignednumbers/home.htm

7.1 Defines

7.1.1 UUID_GAP

Definition

```
#define UUID_GAP 0x1800
```

Description

Generic Access Profile.

7.1.2 UUID_GATT

Definition

```
#define UUID_GATT 0x1801
```

Description

Generic Attribute Profile.

7.1.3 UUID_PRIMARY_SERVICE

Definition

```
#define UUID_PRIMARY_SERVICE 0x2800
```

Description

Primary Service.

7.1.4 UUID_SECONDARY_SERVICE

Definition

```
#define UUID_SECONDARY_SERVICE 0x2801
```

Description

Secondary Service.

7.1.5 UUID_INCLUDE

Definition

```
#define UUID_INCLUDE 0x2802
```

Description

Include.

7.1.6 UUID_CHARACTERISTIC

Definition

```
#define UUID_CHARACTERISTIC 0x2803
```

Description

Characteristic.

7.1.7 UUID_CHAR_EXT_PROPS

Definition

```
#define UUID_CHAR_EXT_PROPS 0x2900
```

Description

Characteristic Extended Properties.

7.1.8 UUID_CHAR_USER_DESC

Definition

```
#define UUID_CHAR_USER_DESC 0x2901
```

Description

Characteristic User Description.

7.1.9 UUID_CLIENT_CHAR_CFG

Definition

```
#define UUID_CLIENT_CHAR_CFG 0x2902
```

Description

Client Characteristic Configuration.

7.1.10 UUID_SERVER_CHAR_CFG

Definition

```
#define UUID_SERVER_CHAR_CFG 0x2903
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Server Characteristic Configuration.

7.1.11 UUID_CHAR_FORMAT

Definition

```
#define UUID_CHAR_FORMAT 0x2904
```

Description

Characteristic Format.

7.1.12 UUID_CHAR_AGG_FORMAT

Definition

```
#define UUID_CHAR_AGG_FORMAT 0x2905
```

Description

Characteristic Aggregate Format.

7.1.13 UUID_DEVICE_NAME

Definition

```
#define UUID_DEVICE_NAME 0x2A00
```

Description

Device Name.

7.1.14 UUID_APPEARANCE

Definition

```
#define UUID_APPEARANCE 0x2A01
```

Description

Appearance.

7.1.15 UUID_PER_PRIV_FLAG

Definition

```
#define UUID_PER_PRIV_FLAG 0x2A02
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Peripheral Privacy Flag.

7.1.16 UUID_RECONNECTION_ADDR

Definition

```
#define UUID_RECONNECTION_ADDR 0x2A03
```

Description

Reconnection Address.

7.1.17 UUID_PER_PREF_CONN_PARAMS

Definition

```
#define UUID_PER_PREF_CONN_PARAMS 0x2A04
```

Description

Peripheral Preferred Connection Parameters.

7.1.18 UUID_SERVICE_CHANGED

Definition

```
#define UUID_SERVICE_CHANGED 0x2A05
```

Description

Service Changed.

7.1.19 ATT_ATTR_NO_FLAG

Definition

```
#define ATT_ATTR_NO_FLAG 0x0000
```

Description

No attribute flag defined.

7.1.20 ATT_ATTR_NO_ADD_FLAG

Definition

```
#define ATT_ATTR_NO_ADD_FLAG 0x0000
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

No additional attribute flag defined.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

8 GATT Flags for Write Request

Description

Aligned to the flags defined by ATT.

8.1 Defines

8.1.1 GATT_WRITE_COMMAND

Definition

```
#define GATT_WRITE_COMMAND ATT_WRITE_COMMAND
```

Description

Send Write Command to the server.

8.1.2 GATT_WRITE_REQUEST

Definition

```
#define GATT_WRITE_REQUEST ATT_WRITE_REQUEST
```

Description

Send Write Request to the server.

8.1.3 GATT_WRITE_SIGNED

Definition

```
#define GATT_WRITE_SIGNED ATT_WRITE_SIGNED
```

Description

Send Signed Write to the server. Only Write Command can be signed.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

9 GATT Client

9.1 Functions

9.1.1 GattDiscoverAllCharDescriptors

Syntax

```
 sys_status GattDiscoverAllCharDescriptors (uint16 cid, uint16
start_handle, uint16 end_handle)
```

Description

As client, find characteristic descriptors by handle range.
Find all the characteristic descriptor's attribute handles and attribute types within a characteristic definition when only the characteristic handle range is known.

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection
start_handle	Start handle of the specified characteristic
end_handle	End handle of the specified characteristic

Returns

sys_status_success if successful or else an error code.

9.1.2 GattDiscoverAllPrimaryServices

Syntax

```
 sys_status GattDiscoverAllPrimaryServices (uint16 cid)
```

Description

As client, discover all primary services on a server.

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection

Returns

sys_status_success if successful or else an error code.

CSR

9.1.3 GattDiscoverPrimaryServiceByUuid

Syntax

```
sys_status GattDiscoverPrimaryServiceByUuid (uint16 cid,
GATT_UUID_T uuid_type, uint16 * uuid)
```

Description

As client, discover a specific primary service on a server when only the Service UUID is known.
The specific primary service may exist multiple times on a server.

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection
uuid_type	16-bit (GATT_UUID16) or 128-bit (GATT_UUID128) UUID
uuid	Pointer to an array containing UUID value in big-endian format

Returns

`sys_status_success` if successful or else an error code.

9.1.4 GattDiscoverServiceChar

Syntax

```
sys_status GattDiscoverServiceChar (uint16 cid, uint16
start_handle, uint16 end_handle, GATT_UUID_T uuid_type, uint16 * uuid)
```

Description

As client, discover all service characteristics or discover characteristics by UUID.
If the uuid_type is GATT_UUID_NONE and uuid is NULL, then discover all service characteristics within the handle range.
Otherwise, discover characteristics by UUID and service handle range.

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection
start_handle	Start handle of the specified service
end_handle	End handle of the specified service
uuid_type	16-bit (GATT_UUID16) or 128-bit (GATT_UUID128) UUID
uuid	Pointer to an array containing UUID value in big-endian format

Returns

`sys_status_success` if successful or else an error code.

9.1.5 GattExchangeMtuReq

Syntax

```
sys_status GattExchangeMtuReq (uint16 cid, uint16 client_rx_mtu)
```

Description

As client, exchange the maximum RX MTU supported by the device.

This function should preferably be used when the GATT client supports an ATT_MTU value greater than default the ATT_MTU (23 octets) for Attribute Protocol data. This procedure shall only be initiated once during a connection.

Note:

CSR1000 currently only supports default ATT_MTU (23 octets) value.

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection
client_rx_mtu	Maximum RX MTU supported by GATT client

Returns

`sys_status_success` if successful or else an error code.

9.1.6 GattFindIncludedServices

Syntax

```
sys_status GattFindIncludedServices (uint16 cid, uint16
start_handle, uint16 end_handle)
```

Description

As client, find included service declarations within a service definition on a server.

The service specified is identified by the service handle range.

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection
start_handle	Start handle of the specified service
end_handle	End handle of the specified service

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

`sys_status_success` if successful or else an error code.

9.1.7 GattInstallClientRole

Syntax

```
void GattInstallClientRole (void)
```

Description

Install GATT Client functionality.
This will install all mandatory GATT Client procedures. It should be called after `GattInit()` in any application that operates as a GATT Client. This function must be called before any other GATT Client functions are called to guarantee correct operation.

Parameters

Parameter	Description
None	-

Returns

Nothing

9.1.8 GattReadCharUsingUuid

Syntax

```
sys_status GattReadCharUsingUuid (uint16 cid, uint16
start_handle, uint16 end_handle, GATT_UUID_T uuid_type, uint16 * uuid)
```

Description

As client, read a Characteristic Value by UUID.

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection
start_handle	Start handle of the specified service to which characteristic belongs
end_handle	End handle of the specified service to which characteristic belongs
uuid_type	16-bit (GATT_UUID16) or 128-bit (GATT_UUID128) UUID
uuid	Pointer to an array containing the UUID value in big-endian format

Returns

`sys_status_success` if successful or else an error code.

9.1.9 GattReadCharValue

Syntax

```
sys_status GattReadCharValue (uint16 cid, uint16 handle)
```

Description

As client, read a Characteristic from a server by Handle.

May be used to read a characteristic value or descriptor from a server when the client knows the characteristic's Attribute handle.

Parameters

Parameter	Description
`cid`	Connection identifier for established BLE-U connection
`handle`	Characteristic value handle (to read Characteristic Value) OR Characteristic descriptor handle (to read Characteristic Descriptor)

Returns

`sys_status_success` if successful or else an error code.

9.1.10 GattReadLongCharValue

Syntax

```
sys_status GattReadLongCharValue (uint16 cid, uint16 handle,
uint16 value_offset)
```

Description

As client, read a long Characteristic by handle.

May be used to read a characteristic value or descriptor from a server when the client knows the characteristic's Attribute handle.

This procedure should be used if the length of the Characteristic Value or Descriptor to be read is longer than can be sent in a single Read Response Attribute Protocol message.

Parameters

Parameter	Description
`cid`	Connection identifier for established BLE-U connection
`handle`	Characteristic value handle (to read Characteristic Value) OR Characteristic descriptor handle (to read Characteristic Descriptor)
`value_offset`	Offset within the characteristic to be read. To read the complete Characteristic set value_offset to `0x00`.

Returns

`sys_status_success` if successful or else an error code.

9.1.11 GattReadMultipleCharValues

Syntax

```
 sys_status GattReadMultipleCharValues ( uint16 cid, uint16
size_handles, uint16 * handles )
```

Description

As client, read multiple Characteristic Values by handles.

Note:

The characteristic values to be read should have a known fixed length with the exception of the last value that can have variable length.

Note:

A client should not request multiple Characteristic Values when the response's Set Of Values parameter is equal to (ATT_MTU - 1) octets in length since it is not possible to determine if the last Characteristic Value was read or additional Characteristic Values exist but were truncated.

Parameters

Parameter	Description
`cid`	Connection identifier for established BLE-U connection
`size_handles`	Number of elements in the 'handles' array
`handles`	Pointer to an array of characteristic value handles, in the order that the values are required in response

Returns

sys_status_success if successful or else an error code.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

9.1.12 GattStopCurrentProcCmd

Syntax

```
void GattStopCurrentProcCmd ( uint16 cid)
```

Description

As client, stop a procedure.
Can be used to stop any of the following on-going procedures before completion.

- GattDiscoverAllPrimaryServices
- GattDiscoverPrimaryServiceByUuid
- GattFindIncludedServices
- GattDiscoverServiceChar
- GattDiscoverAllCharDescriptors
- GattReadCharUsingUuid

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection

Returns

Nothing

9.1.13 GattWriteCharValueReq

Syntax

```
 sys_status GattWriteCharValueReq ( uint16 cid, uint16 flags,
uint16 handle, uint16 size_value, uint8 * value )
```

Description

As client, write a Characteristic by handle.
May be used to write a characteristic value or descriptor on a server when the client knows the characteristic's handle.
Used in following GATT procedures. These procedures only write the first (ATT_MTU - 3) octets of a characteristic value or descriptor.

- Write Without Response - used when the client does not need an acknowledgement that the write was successfully performed.
- Signed Write Without Response - used when the ATT bearer is not encrypted. This procedure shall only be used if the Characteristic Properties authenticated bit is enabled and the client and server device share a bond - NOT CURRENTLY SUPPORTED.
- Write Characteristic Value - This function is used when the client needs an acknowledgement that the write was successfully performed.

If the application wishes to stream data to the server it can use the Write Without Response procedure. This procedure checks to ensure that internal buffer exhaustion cannot occur as a direct result of the streaming Write Commands. If this check fails then the function will return status code gatt_status_busy.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
`cid`	Connection identifier for established BLE-U connection
`flags`	The following flags are used to distinguish the sub-procedures GATT_WRITE_COMMAND - Write Without Response GATT_WRITE_SIGNED - Signed Write Without Response GATT_WRITE_REQUEST - Write Characteristic Value or Descriptor
`handle`	Characteristic handle for value or descriptor
`size_value`	Size of characteristic value or descriptor in octets
`value`	Pointer to an array containing characteristic in LITTLE ENDIAN format

Returns

sys_status_success if successful or else an error code.

9.1.14 GattWriteLongCharValueReq

Syntax

```
 sys_status GattWriteLongCharValueReq ( uint16 cid, uint16
handle, uint16 offset, uint16 size_value, uint8 * value, bool reliable )
```

Description

As client, write a long Characteristic value or descriptor by handle.
Used when the client knows the Characteristic value handle but the length of the Characteristic value is longer than (ATT_MTU - 3) octets.
This function is also used for reliable writes of a long Characteristic value or descriptor.

Parameters

Parameter	Description
`cid`	Connection identifier for established BLE-U connection
`handle`	Characteristic value handle (to write Characteristic Value) OR Characteristic descriptor handle (to write Characteristic Descriptor)
`offset`	Characteristic offset to which the value is written
`size_value`	Size of characteristic value or descriptor in octets
`value`	Pointer to an array containing characteristic value or descriptor in LITTLE ENDIAN format. The application must preserve the characteristic value memory until the procedure is completed.
`reliable`	TRUE, if assurance is required that the correct Characteristic value is going to be written before the write is performed

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

sys_status_success if successful or else an error code.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

10 GATT Server

- GATT Database Helper Macros

10.1 Functions

10.1.1 GattAccessRsp

Syntax

```
 void GattAccessRsp (uint16 cid, uint16 handle, sys_status rc,
uint16 size_value, uint8 * value)
```

Description

As server, respond to `GATT_ACCESS_IND` event if the `ATT_ACCESS_PERMISSION` and/or `ATT_ACCESS_WRITE_COMPLETE` flags have been set.

Note:

Warning: If the application needs to return a GATT Application Error Code in the range `0x80` to `0xFF` (see Bluetooth Specification v4.0 Vol.3 Part F Section 3.4.1) then it *must* OR the value in that range with `gatt_status_app_mask`. This ensures that the status code is located within the correct block of status codes within the firmware sys_status enumerated type. If the application error code is not ORed with this value then the resulting status code sent to tbe peer device will be "Unlikely Error".

Parameters

Parameter	Description
`cid`	Connection identifier for established BLE-U connection
`handle`	Attribute handle
`rc`	GATT error code or success (`sys_status_success`, `gatt_status_irq_proceed`, or Application Error code as described in the warning above)
`size_value`	Length of the value
`value`	Pointer to an array containing Attribute value in LITTLE ENDIAN format

Returns

Nothing.

Note:

Applications that handle the database access themselves should return `sys_status_success`, whereas applications that want the ATT stack to handle database access should return `gatt_status_irq_proceed`. Either way, behaviour is undefined if different success codes are returned for different phases of a single transaction - for instance in a Write Long request.

The application receiving a `GATT_ACCESS_IND` event shall treat it as an atomic event and return `GattAccessRsp()` immediately without any context switch or other GATT calls.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

10.1.2 GattAddDatabaseReq

Syntax

```
void GattAddDatabaseReq (uint16 size_db, uint16 * db)
```

Description

As server, add a complete attribute database for supported services.

Used in GATT server role to add a complete attribute database for supported services. Once the attribute database is registered, the application shall not try to update the attribute database directly.

Calling this function will instantiate all mandatory GATT Server procedures.

Note:

The application shall preserve the attribute database memory. This memory shall not be used for any other purpose.

Parameters

Parameter	Description
size_db	Size of data base in words
db	Pointer to a data base array of service attributes in LITTLE ENDIAN format

Returns

Nothing.

10.1.3 GattCharValueIndication

Syntax

```
void GattCharValueIndication (uint16 cid, uint16 handle, uint16
size_value, uint8 * value)
```

Description

As server, indicate a characteristic value to a client, expecting an ACK.

Used in GATT server role to indicate a characteristic value to a client and expects an Attribute Protocol layer acknowledgement.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection
handle	Characteristic value handle
size_value	Size of characteristic value in octets
value	Pointer to an array containing characteristic value in LITTLE ENDIAN format

Returns

Nothing.

10.1.4 GattCharValueNotification

Syntax

```
 void GattCharValueNotification (uint16 cid, uint16 handle,
uint16 size_value, uint8 * value)
```

Description

As server, notify a characteristic value to a client, not expecting an ACK.

Used in GATT server role to notify a Characteristic Value to a client without expecting any Attribute Protocol layer acknowledgement.

Data streaming using Notifications is supported. The server will check that each request will not result in buffer exhaustion. If the check fails then the corresponding CFM message will return `gatt_status_busy`.

Parameters

Parameter	Description
cid	Connection identifier for established BLE-U connection
handle	Characteristic value handle
size_value	Size of characteristic value in octets
value	Pointer to an array containing characteristic value in LITTLE ENDIAN format

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

10.1.5 GattExchangeMtuRsp

Syntax

```
void GattExchangeMtuRsp (uint16 cid, uint16 server_rx_mtu)
```

Description

As server, respond to Exchange MTU.

Used in GATT server role to respond to an Exchange MTU indication with the maximum MTU supported by the server.

Note:

CSR1000 currently only supports the default ATT_MTU (23 octets) value.

Parameters

Parameter	Description
`cid`	Connection identifier for established BLE-U connection
`server_rx_mtu`	Maximum RX MTU supported by the GATT server

Returns

Nothing.

10.1.6 GattInstallServerExchangeMtu

Syntax

```
void GattInstallServerExchangeMtu (void)
```

Description

Install GATT Server support for the optional Exchange MTU procedure.

When installed, the application will receive a `GATT_EXCHANGE_MTU_IND` message when a remote client starts an MTU negotiation. If the procedure is not installed then the GATT layer will automatically negotiate the MTU based on the firmware default of ATT_MTU_DEFAULT.

Parameters

Parameter	Description
`None`	-

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

10.1.7 GattInstallServerReadMultiple

Syntax

```
void GattInstallServerReadMultiple (void)
```

Description

Install GATT Server support for the optional Read Multiple Characteristic Values procedure.
If the procedure is not installed then the GATT layer will reject any attempt by a remote client to initiate it.

Parameters

Parameter	Description
None	-

Returns

Nothing.

10.1.8 GattInstallServerWrite

Syntax

```
void GattInstallServerWrite (void)
```

Description

Install GATT Server support for the optional Write Characteristic Value, Write Without Response, Signed Write Without Response and Write Characteristic Descriptors procedures.
This function does not need to be called if the application calls `GattInstallServerWriteLongReliable()`.
If the procedures are not installed then the GATT layer will reject any attempt by a remote client to initiate them.

Parameters

Parameter	Description
None	-

Returns

Nothing.

10.1.9 GattInstallServerWriteLongReliable

Syntax

```
void GattInstallServerWriteLongReliable (void)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Install GATT Server support for the optional Write Long Characteristic Value, Write Long Characteristic Descriptor and Characteristic Value Reliable Write procedures.

If Write Long Characteristic Value is supported then it is mandatory to also support Write Characteristic Value on the server. Therefore calling this function will automatically install support for the other Write procedures.

If the procedures are not installed then the GATT layer will reject any attempt by a remote client to initiate them.

Parameters

Parameter	Description
`None`	-

Returns

Nothing.

CSR µEnergy Firmware Library Documentation

11 GATT Database Helper Macros

Description

Applications should supply the attribute type and overall length in bytes. These macros are used by further macros below to build complete attributes including all data fields.

The length is either a 6-bit or 8-bit field, depending on whether the attribute type is or is not a Characteristic Declaration (as that attribute type needs an extra 2 bits for the Additional Flags field). Rather than try to cope with both cases in one macro we simply provide two related macros and expect the caller to use the right one.

The macros in this section are a set of generic and GATT-specific macros which together build a library of macros that can be used when constructing a static GATT server database. Using the macros should remove some of the complexity in building standard attributes, particularly in cases where 16-bit values such as a UUID or handle are not word-aligned and therefore need to be byte-swapped and split across two 16-bit integers.

As an example of how to use the macros, here is the start of the database definition of a simple GATT server. We want to declare a GAP service, putting the handle numbers of each attribute in braces {like this}:

```
{0001}   Primary Service Declaration (GAP)
{0002}   Characteristic, prop=[READ], handle=0x0003, uuid=DEVICE NAME
{0003}   Characteristic value: "Example Server"
{0004}   Characteristic, prop=[READ], handle=0x0005, uuid=APPEARANCE
{0005}   0x0011
{0006}   Characteristic, prop=[READ|SGWR], handle=0x0007, uuid=RECONNECTION ADDR
{0007}   0x0000 0x0000 0x0000
{0008}   Characteristic, prop=[SGWR], handle=0x0009, uuid=PERIPHERAL PRIVACY FLAG
{0009}   0x00
{000a}   Characteristic, prop=[READ|WREQ|WCMD], handle=0x000b, uuid=PREF.
          CONNECTION PARAMS
{000b}   min=6, max=6, latency=0, to=2000 (0x07D0)
{000c}   Characteristic Extended props (Reliable Write)
```

Note:

While the final Characteristic is followed by a Characteristic Extended Properties attribute, the char property bit XTND is not set, therefore the extended properties will not be used.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

This translates to the following C code:

```
uint16 example_db[] =
{
    GATT_DECL_PRIM_SERV_UUID16(ATT_ATTR_NO_FLAG, UUID_GAP),
    GATT_DECL_CHAR16(ATT_ATTR_NO_FLAG, ATT_ATTR_NO_ADD_FLAG,
                     ATT_PERM_READ,
                     0x0003, UUID_DEVICE_NAME),
    GATT_DECL_CHAR_VALUE16(ATT_ATTR_NO_FLAG,
                           0x0E,
                           BYTE_JOIN_16('E','x'), BYTE_JOIN_16('a','m'),
                           BYTE_JOIN_16('p','l'), BYTE_JOIN_16('e',' '),
                           BYTE_JOIN_16('S','e'), BYTE_JOIN_16('r','v'),
                           BYTE_JOIN_16('e','r')),
    GATT_DECL_CHAR16(ATT_ATTR_NO_FLAG, ATT_ATTR_NO_ADD_FLAG,
                     ATT_PERM_READ,
                     0x0005, UUID_APPEARANCE),
    GATT_DECL_CHAR_VALUE16(ATT_ATTR_NO_FLAG,
                           0x02, BYTE_SWAP_16(0x0011)),
    GATT_DECL_CHAR16(ATT_ATTR_NO_FLAG, ATT_ATTR_NO_ADD_FLAG,
                     ATT_PERM_READ|ATT_PERM_WRITE_SIGNED,
                     0x0007, UUID_RECONNECTION_ADDR),
    GATT_DECL_CHAR_VALUE16(ATT_ATTR_NO_FLAG,
                           0x0006, 0x0000, 0x0000, 0x0000),
    GATT_DECL_CHAR16(ATT_ATTR_NO_FLAG, ATT_ATTR_NO_ADD_FLAG,
                     ATT_PERM_WRITE_SIGNED, 0x0009, UUID_PER_PRIV_FLAG),
    GATT_DECL_CHAR_VALUE16(ATT_ATTR_NO_FLAG, 0x0001, 0x0000),
    GATT_DECL_CHAR16(ATT_ATTR_NO_FLAG, ATT_ATTR_NO_ADD_FLAG,
                     ATT_PERM_READ|ATT_PERM_WRITE_REQ|ATT_PERM_WRITE_CMD,
                     0x000b, UUID_PER_PREF_CONN_PARAMS),
    GATT_DECL_CHAR_VALUE16(ATT_ATTR_NO_FLAG,
                           0x0008,
                           BYTE_SWAP_16(0x0006), BYTE_SWAP_16(0x0006),
                           0x0000, BYTE_SWAP_16(0x07d0)),
    GATT_DECL_CHAR_EXT_PROPS(ATT_ATTR_NO_FLAG,
                             ATT_PERM_RELIABLE_WRITE),
};
```

In a real application this would be the first of many services would be declared and the example_db array would be considerably longer.

11.1 Defines

11.1.1 GATT_DECL_ATTR

Definition

```
#define GATT_DECL_ATTR (_attr, _flags, _len)
```

Description

Value: `( (((uint16)(_attr) & 0x000f) << 12) | \`
`(((uint16)(_flags) & 0x000f) << 8) | ((_len) & 0x00ff) )`

Declare an attribute.

11.1.2 GATT_DECL_ATTR_ADD

Definition

```
#define GATT_DECL_ATTR_ADD (_attr, _flags, _add_flags, _len)
```

Description

Value:
```
( (((uint16)(_attr) & 0x000f) << 12) | \
(((uint16)(_flags) & 0x000f) << 8) | \
(((uint16)(_add_flags) & 0x0003) << 6) | ((_len) & 0x003f) )
```

Declare a characteristic.

11.1.3 GATT_DECL_CHAR128

Definition

```
#define GATT_DECL_CHAR128 ( _flags, _add_flags, _prop, _hndl, _uuid1,
_uuid2, _uuid3, _uuid4, _uuid5, _uuid6, _uuid7, _uuid8 )
```

Description

Value:
```
GATT_DECL_ATTR_ADD(att_type_declaration, _flags, _add_flags, 19), \
( ((uint16)WORD_LSB(_prop) << 8) | WORD_LSB(_hndl) ), \
( ((uint16)WORD_MSB(_hndl) << 8) | WORD_LSB(_uuid8) ), \
( ((uint16)WORD_MSB(_uuid8) << 8) | WORD_LSB(_uuid7) ), \
( ((uint16)WORD_MSB(_uuid7) << 8) | WORD_LSB(_uuid6) ), \
( ((uint16)WORD_MSB(_uuid6) << 8) | WORD_LSB(_uuid5) ), \
( ((uint16)WORD_MSB(_uuid5) << 8) | WORD_LSB(_uuid4) ), \
( ((uint16)WORD_MSB(_uuid4) << 8) | WORD_LSB(_uuid3) ), \
( ((uint16)WORD_MSB(_uuid3) << 8) | WORD_LSB(_uuid2) ), \
( ((uint16)WORD_MSB(_uuid2) << 8) | WORD_LSB(_uuid1) ), \
( ((uint16)WORD_MSB(_uuid1) << 8) /* Trailing byte */)
```

Characteristic Declaration 128-bit UUID (fixed length): 1 byte properties, 2 bytes handle, 16 bytes UUID.

Note:

Warning: This macro is not intended to be used directly within the database definition. It is called from the GATT_DECL_CHAR128 macro, which is used to create the Characteristic Declaration and Characteristic Value attributes in one step.

11.1.4 GATT_DECL_CHAR128_FULL

Definition

```
#define GATT_DECL_CHAR128_FULL ( _flags, _read_sec, _write_sec, _perm, _hndl,
_uuid1, _uuid2, _uuid3, _uuid4, _uuid5, _uuid6, _uuid7, _uuid8, _len, ... )
```

Description

Value:
```
GATT_DECL_CHAR128(ATT_ATTR_NO_FLAG, _read_sec, _perm, _hndl, \
_uuid1, _uuid2, _uuid3, _uuid4, _uuid5, _uuid6, _uuid7, _uuid8), \
GATT_DECL_CHAR_VALUE128((_flags|_write_sec), _len , ##__VA_ARGS__ )
```

Characteristic Definition with 128bits UUID: At minimum a Characteristic definition includes a Characteristic Declaration immediately followed by a Characteristic Value. This macro creates both attributes in one go (and will thus "consume" two consecutive attribute handles).

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Variable	Description
`_flags`	Flags for the Characteristic Value: ATT_ATTR_IRQ
`_read_sec`	Read Security for the Characteristic Value: ATT_ATTR_SEC_NONE, ATT_ATTR_SEC_ENCRYPTION, ATT_ATTR_SEC_AUTHENTICATION
`_write_sec`	Write Security for the Characteristic Value: ATT_ATTR_SEC_NONE, ATT_ATTR_SEC_ENCRYPTION, ATT_ATTR_SEC_AUTHENTICATION
`_perm`	GATT permissions for the Characteristic Value: ATT_PERM_CONFIGURE_BROADCAST, ATT_PERM_READ, ATT_PERM_WRITE_CMD, ATT_PERM_WRITE_REQ, ATT_PERM_NOTIFY, ATT_PERM_INDICATE, ATT_PERM_WRITE_SIGNED, ATT_PERM_EXTENDED
`_hndl`	The handle of the Characteristic Value (16 bits). Shall be the Characteristic Declaration handle + 1.
`_uuid1-8`	Characteristic Value 128-bit UUID. The UUID should be provided as 8 16-bit comma-separated values, with the Most Significant Word of the UUID placed first. The whole UUID will then be word- and byte-swapped, and properly aligned (because the length is odd, the UUID starts half-way through a 16-bit word)
`_len`	Length of the Characteristic Value (in bytes).
`value`	The application is responsible for converting the 'value' byte sequence to a set of 16-bit integers, e.g. using the BYTE_JOIN_16 macro.

11.1.5 GATT_DECL_CHAR16

Definition

```
#define GATT_DECL_CHAR16 ( _flags, _add_flags, _prop, _hndl, _uuid )
```

Description

Value:
```
GATT_DECL_ATTR_ADD(att_type_declaration, _flags, _add_flags, 5), \
 ( ((uint16)WORD_LSB(_prop) << 8) | WORD_LSB(_hndl) ), \
 ( ((uint16)WORD_MSB(_hndl) << 8) | WORD_LSB(_uuid) ), \
 ( ((uint16)WORD_MSB(_uuid) << 8) )
```

Characteristic Declaration 16-bit UUID (fixed length): 1 byte properties, 2 bytes handle, 2 bytes UUID.

Warning: This macro is not intended to be used directly within the database definition. It is called from the GATT_DECL_CHAR16_FULL macro, which is used to create the Characteristic Declaration and Characteristic Value attributes in one step.

11.1.6 GATT_DECL_CHAR16_FULL

Definition

```
#define GATT_DECL_CHAR16_FULL ( _flags, _read_sec, _write_sec, _perm, _hndl,
_uuid, _len, ... )
```

Description

```
Value: GATT_DECL_CHAR16(ATT_ATTR_NO_FLAG, _read_sec, _perm, _hndl, _uuid), \
 GATT_DECL_CHAR_VALUE16((_flags|_write_sec), _len , ##__VA_ARGS__ )
```

Characteristic Definition with 16bit UUID: At minimum a Characteristic definition includes a Characteristic Declaration immediately followed by a Characteristic Value. This macro creates both attributes in one go (and will thus "consume" two consecutive attribute handles).

Parameters

Variable	Description
`_flags`	Flags for the Characteristic Value: ATT_ATTR_IRQ
`_read_sec`	Read Security for the Characteristic Value: ATT_ATTR_SEC_NONE, ATT_ATTR_SEC_ENCRYPTION, ATT_ATTR_SEC_AUTHENTICATION
`_write_sec`	Write Security for the Characteristic Value: ATT_ATTR_SEC_NONE, ATT_ATTR_SEC_ENCRYPTION, ATT_ATTR_SEC_AUTHENTICATION
`_perm`	GATT permissions for the Characteristic Value: ATT_PERM_CONFIGURE_BROADCAST, ATT_PERM_READ, ATT_PERM_WRITE_CMD, ATT_PERM_WRITE_REQ, ATT_PERM_NOTIFY, ATT_PERM_INDICATE, ATT_PERM_WRITE_SIGNED, ATT_PERM_EXTENDED
`_hndl`	The handle of the Characteristic Value (16 bits). Shall be the Characteristic Declaration handle + 1.
`_uuid`	Characteristic Value 16-bit UUID.
`_len`	Length of the Characteristic Value (in bytes).
`value`	The application is responsible for converting the 'value' byte sequence to a set of 16-bit integers, e.g. using the BYTE_JOIN_16 macro.

11.1.7 GATT_DECL_CHAR_AGG_FORMAT

Definition

```
#define GATT_DECL_CHAR_AGG_FORMAT ( _flags, _num, ... )
GATT_DECL_ATTR(att_type_ch_agg, _flags, _num*2) , ##__VA_ARGS__
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

Characteristic Aggregated Format:

Parameter

Variable	Description
_flags	Flags for the Characteristic Value: ATT_ATTR_IRQ *
_num	Number of handles in the handleList
handleList	List of handles of which the aggregated format is constructed

11.1.8 GATT_DECL_CHAR_EXT_PROPS

Definition

```
#define GATT_DECL_CHAR_EXT_PROPS ( _flags, _props )
```

Description

```
Value: GATT_DECL_ATTR(att_type_ch_extended, _flags, 2), \
 BYTE_SWAP_16(_props)
```

Characteristic Extended Properties (fixed length): 2 bytes extended properties.

11.1.9 GATT_DECL_CHAR_FORMAT

Definition

```
#define GATT_DECL_CHAR_FORMAT ( _flags, _fmt, _exp, _unit, _ns, _desc )
```

Description

```
Value: GATT_DECL_ATTR(att_type_ch_format, _flags, 7), \
 BYTE_JOIN_16(_fmt, _exp), BYTE_SWAP_16(_unit), \
 BYTE_JOIN_16(_ns, WORD_LSB(_desc)), BYTE_JOIN_16(WORD_MSB(_desc), 0)
```

Characteristic Format (fixed length): 1 byte format, 1 byte exponent, 2 bytes unit, 1 byte name space, 2 bytes description.

11.1.10 GATT_DECL_CHAR_VALUE128

Definition

```
#define GATT_DECL_CHAR_VALUE128 ( _flags, _len, ... )
GATT_DECL_ATTR(att_type_value128, _flags, _len) , ##__VA_ARGS__
```

Description

Characteristic Value for 128-bit UUID (variable length): the application should supply the length, and is responsible for converting the following byte sequence to a set of 16-bit integers, e.g. using the BYTE_JOIN_16 macro. This macro is defined using a GCC variadic macro.

Warning: This macro is not intended to be used directly within the database definition. It is called from the GATT_DECL_CHAR128 macro, which is used to create the Characteristic Declaration and Characteristic Value attributes in one step.

11.1.11 GATT_DECL_CHAR_VALUE16

Definition

```
#define GATT_DECL_CHAR_VALUE16 ( _flags, _len, ... )
GATT_DECL_ATTR(att_type_value, _flags, _len) , ##__VA_ARGS__
```

Description

Characteristic Value for 16-bit UUID (variable length): the application should supply the length, and is responsible for converting the following byte sequence to a set of 16-bit integers, e.g. using the BYTE_JOIN_16 macro. This macro is defined using a GCC variadic macro.

Warning:This macro is not intended to be used directly within the database definition. It is called from the GATT_DECL_CHAR16_FULL macro, which is used to create the Characteristic Declaration and Characteristic Value attributes in one step.

11.1.12 GATT_DECL_FULL

Definition

```
#define GATT_DECL_FULL ( _flags, _read_sec, _write_sec, _uuid, _perm, _len, ... )
```

Description

```
Value: GATT_DECL_ATTR(att_type_full, (_flags | _write_sec), _len), \
 _uuid, \
 (_read_sec<<14 | _perm) , ##__VA_ARGS__
```

Generic Attribute (16-bit UUID): 2 bytes UUID, 1 word permissions, variable-length data.

Warning:It is not advised to use this macro to declare database attributes that can be specified using the macros above, as the generated attribute will take up more space in the ATT database.

CSR µEnergy Firmware Library Documentation

Parameters

Variable	Description
_flags	Flags for the Attribute: ATT_ATTR_IRQ
_read_sec	Read Security for the Attribute: ATT_ATTR_SEC_NONE, ATT_ATTR_SEC_ENCRYPTION, ATT_ATTR_SEC_AUTHENTICATION
_write_sec	Write Security for the Attribute: ATT_ATTR_SEC_NONE, ATT_ATTR_SEC_ENCRYPTION, ATT_ATTR_SEC_AUTHENTICATION
_perm	GATT permissions for the Attribute: ATT_PERM_CONFIGURE_BROADCAST, ATT_PERM_READ, ATT_PERM_WRITE_CMD, ATT_PERM_WRITE_REQ, ATT_PERM_NOTIFY, ATT_PERM_INDICATE, ATT_PERM_WRITE_SIGNED, ATT_PERM_EXTENDED
_uuid	Attribute 16-bit UUID.
_len	Length of the Attribute data (in bytes).
value	The application is responsible for converting the 'value' byte sequence to a set of 16-bit integers, e.g. using the BYTE_JOIN_16 macro.

11.1.13 GATT_DECL_FULL128

Definition

```
#define GATT_DECL_FULL128 ( _flags, _read_sec, _write_sec, _uuid1,
_uuid2, _uuid3, _uuid4, _uuid5, _uuid6, _uuid7, _uuid8, _prop, _len, ... )
```

Description

```
Value: GATT_DECL_ATTR(att_type_full128, _flags, _len), \
 _uuid1, _uuid2, _uuid3, _uuid4, _uuid5, _uuid6, _uuid7, _uuid8, \
 (_read_sec<<14 | _prop) , ##__VA_ARGS__
```

Generic Attribute (128-bit UUID): 16 bytes UUID, 1 word permissions, variable-length data.

Warning:It is not advised to use this macro to declare database attributes that can be specified using the macros above, as the generated attribute will take up more space in the ATT database.

Parameters

Variable	Description
_flags	Flags for the Attribute: ATT_ATTR_IRQ
_read_sec	Read Security for the Attribute: ATT_ATTR_SEC_NONE, ATT_ATTR_SEC_ENCRYPTION, ATT_ATTR_SEC_AUTHENTICATION
_write_sec	Write Security for the Attribute: ATT_ATTR_SEC_NONE, ATT_ATTR_SEC_ENCRYPTION, ATT_ATTR_SEC_AUTHENTICATION
_perm	GATT permissions for the Attribute: ATT_PERM_CONFIGURE_BROADCAST, ATT_PERM_READ, ATT_PERM_WRITE_CMD, ATT_PERM_WRITE_REQ, ATT_PERM_NOTIFY, ATT_PERM_INDICATE, ATT_PERM_WRITE_SIGNED, ATT_PERM_EXTENDED
_uuid1-8	Attribute 128-bit UUID. The UUID should be provided as 8 16-bit comma-separated values, with the Most Significant Word of the UUID placed first. The UUID will then be packed into its internal 32-bit representation.
_len	Length of the Attribute (in bytes).
value	The application is responsible for converting the 'value' byte sequence to a set of 16-bit integers, e.g. using the BYTE_JOIN_16 macro.

11.1.14 GATT_DECL_HANDLE_PAD

Definition

```
#define GATT_DECL_HANDLE_PAD ( _pad ) GATT_DECL_ATTR(att_type_handle_padding,
0 /* No flags */, 2), _pad
```

Description

Meta-Attribute to pad handle counter, allowing gaps to be left for expansion in the database during development, or to allow us to use the recommended handle numbering specified for qualification test cases.

Note:

This entry in the database is NOT an attribute and will not be discoverable by remote clients OR by local applications using the APIs to query the database.

Warning:Padding meta-attributes are only intended to be placed *between* services. Placing them within the definition of a service is not supported and may lead to invalid behaviour. 2 bytes pad length

11.1.15 GATT_DECL_INCL_SERV_UUID128

Definition

```
#define GATT_DECL_INCL_SERV_UUID128 ( _flags, _hndl, _end )
```

Description

```
Value: GATT_DECL_ATTR(att_type_include, _flags, 4), \
 BYTE_SWAP_16(_hndl), BYTE_SWAP_16(_end)
```

Include Service 128-bit UUID (fixed length): 2 bytes include service handle, 2 bytes end group handle.

11.1.16 GATT_DECL_INCL_SERV_UUID16

Definition

```
#define GATT_DECL_INCL_SERV_UUID16 ( _flags, _hndl, _end, _uuid )
```

Description

```
Value: GATT_DECL_ATTR(att_type_include, _flags, 6), \
 BYTE_SWAP_16(_hndl), BYTE_SWAP_16(_end), BYTE_SWAP_16(_uuid)
```

Include Service 16-bit UUID (fixed length): 2 bytes include service handle, 2 bytes end group handle, 2 bytes service UUID.

11.1.17 GATT_DECL_PRIM_SERV_UUID128

Definition

```
#define GATT_DECL_PRIM_SERV_UUID128 ( _flags, _uuid1, _uuid2, _uuid3, _uuid4,
_uuid5, _uuid6, _uuid7, _uuid8 )
```

Description

```
Value: GATT_DECL_ATTR(att_type_pri_service, _flags, 16), \
 BYTE_SWAP_16(_uuid8), BYTE_SWAP_16(_uuid7),
 BYTE_SWAP_16(_uuid6), BYTE_SWAP_16(_uuid5), \
 BYTE_SWAP_16(_uuid4), BYTE_SWAP_16(_uuid3),
 BYTE_SWAP_16(_uuid2), BYTE_SWAP_16(_uuid1)
```

Primary Service 128-bit UUID (fixed length): 16 bytes UUID.

Note:

The UUID should be provided as 8 16-bit comma-separated values, with the Most Significant Word of the UUID placed first. The whole UUID will then be word- and byte-swapped

11.1.18 GATT_DECL_PRIM_SERV_UUID16

Definition

```
#define GATT_DECL_PRIM_SERV_UUID16 ( _flags, _uuid )
```

Description

```
Value: GATT_DECL_ATTR(att_type_pri_service, _flags, 2), \
 BYTE_SWAP_16(_uuid)
```

Primary Service 16-bit UUID (fixed length): 2 bytes UUID.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

11.1.19 GATT_DECL_SEC_SERV_UUID16

Definition

```
#define GATT_DECL_SEC_SERV_UUID16 ( _flags, _uuid )
```

Description

```
Value: GATT_DECL_ATTR(att_type_sec_service, _flags, 2), \
 BYTE_SWAP_16(_uuid)
```

Secondary Service (fixed length): 2 bytes UUID.

11.1.20 WORD_MSB

Definition

```
#define WORD_MSB(_val) ( ((_val) & 0xff00) >> 8 )
```

Description

Extract the MSB of a 16-bit integer, shifting it down to the lower 8-bits.

11.1.21 WORD_LSB

Definition

```
#define WORD_LSB(_val) ( ((_val) & 0x00ff) )
```

Description

Extract the LSB of a 16-bit integer.

11.1.22 BYTE_SWAP_16

Definition

```
#define BYTE_SWAP_16(_val) ( ((uint16)WORD_LSB(_val) << 8) | WORD_MSB(_val) )
```

Description

Swap the byte order of a 16-bit word.

11.1.23 BYTE_SPLIT_16

Definition

```
#define BYTE_SPLIT_16(_val) WORD_LSB(_val), WORD_MSB(_val)
```

Description

Split a 16-bit word into two bytes, LSB first.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

11.1.24 BYTE_JOIN_16

Definition

```
#define BYTE_JOIN_16(_msb, _lsb) ( ((uint16)WORD_LSB(_msb) << 8) | WORD_LSB(_lsb) )
```

Description

Combine two 8-bit values into a single 16-bit value, with masking to ensure stray bits don't get set.

11.1.25 GATT_DECL_CHAR_USER_DESC

Definition

```
#define GATT_DECL_CHAR_USER_DESC(_flags, _len,...)
 GATT_DECL_ATTR(att_type_ch_descr, _flags, _len) , ##__VA_ARGS__
```

Description

Characteristic User Description (variable length): the application should supply the length, and is responsible for converting the following byte sequence to a set of 16-bit integers, e.g. using the BYTE_JOIN_16 macro. This macro is defined using a GCC variadic macro.

11.1.26 GATT_DECL_CHAR_CLIENT_CFG

Definition

```
#define GATT_DECL_CHAR_CLIENT_CFG(_attr_flags, _flags)
GATT_DECL_ATTR(att_type_ch_c_config, _attr_flags, 2), BYTE_SWAP_16(_flags)
```

Description

Client Characteristic Configuration (fixed length): 2 bytes configuration flags (bitfield).

11.1.27 GATT_DECL_CHAR_SERVER_CFG

Definition

```
#define GATT_DECL_CHAR_SERVER_CFG(_attr_flags, _flags)
 GATT_DECL_ATTR(att_type_ch_s_config, _attr_flags, 2), BYTE_SWAP_16(_flags)
```

Description

Server Characteristic Configuration (fixed length): 2 bytes configuration flags (bitfield).

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

12 Security Manager

12.1 Functions

12.1.1 SMAddStoredKey

Syntax

```
bool SMAddStoredKey ( const TYPED_BD_ADDR_T * bd_addr, const
SM_KEYSET_T * keyset )
```

Description

Stores a key in the Security Manager persistent storage.

This may be called by the application in response to an SM_KEYS_IND or at any other time the application wishes to store a key in the Security Manager persistent storage

Where this call causes another key to be removed from the persistent store, it will be reported back to the application as per SMRemoveStoredKey()

Parameters

Parameter	Description
`bd_addr`	Peer address used as an index for this data
`keyset`	Pointer to security keys

Returns

TRUE if keys not already stored, otherwise FALSE

12.1.2 SMDistributeMasterLtk

Syntax

```
void SMDistributeMasterLtk ( bool distribute_ltk)
```

Description

Indicate whether the Security Manager should request distribution of the master's Long Term Key during bonding.

During bonding, the peer devices negotiate which keys to distribute to each other. This function allows the application to decide whether the LTK, EDIV and Rand should be distributed by the master of the connection. It can be used when the local device is the master or when it is the slave.

The default is for the Security Manager to not request distribution of the master key, as typically this key is only required if the master and slave devices are likely to swap roles but wish to retain the existing bond.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
`distribute_ltk`	Boolean flag indicating whether or not to distribute the master's LTK, EDIV, and Rand.

Returns

Nothing.

12.1.3 SMDivApproval

Syntax

```
void SMDivApproval ( uint16 cid, sm_div_verdict verdict )
```

Description

Approve or reject encrypting the link with the LTK referred to by diversifier in the SM_DIV_APPROVAL_IND.

Note:

Should only be called in response to SM_DIV_APPROVAL_IND, and needs to be called immediately after receiving that event.

Parameters

Parameter	Description
`cid`	Connection identifier for established BLE-U connection
`verdict`	Set to SM_DIV_APPROVED if the diversifier refers to an LTK which are valid for encrypting the link. Set to SM_DIV_REVOKED, if the diversifier referencing to a revoked LTK.

Returns

None

12.1.4 SMEncryptRawAes

Syntax

```
void SMEncryptRawAes ( uint16 * key, uint16 * data )
```

Description

AES encrypt a block of data.

Performs an in-place encryption of supplied plain-text data. The mode of operation is Electronic Code Book, and only one block is encrypted at a time. Data and key should be stored as word-wise little-endian. Example:

The key 0x000102030405060708090a0b0c0d0e0f (MSB -> LSB) is stored

uint16 key[] = {0x0e0f, 0x0c0d, 0x0a0b, 0x0809, 0x0607, 0x0405, 0x0203, 0x0001};

Parameters

Parameter	Description
`key`	Pointer to 8 word (128 bit) encryption key.
`[in] data`	Pointer to 8 word (128 bit) data block to be encrypted.
`[out] data`	Pointer to 8 word (128 bit) encrypted output.

Returns

Nothing.

12.1.5 SMFeaturesReq

Syntax

```
 void SMFeaturesReq ( bool enable_key_storage, bool
enable_keys_request, bool enable_div_approval, bool enable_pairing_auth, bool
enable_long_term_key )
```

Description

Runtime configuration of Security Manager features. On-chip applications will never need to call this function, as for those applications SM features are derived at link time. However, some classes of application that communicate with an off-chip host processor may find this function useful.

Note:

If both enable_key_storage and enable_keys_request are TRUE, then the internal persistent memory will be checked for valid keys. If no keys exist then the application will be asked to supply the keys if it knows about them.

This function cannot be called before GattInit() has been called.

Parameters

Parameter	Description
`enable_key_storage`	If TRUE then the SM will uses its internal persistent memory block for key storage.
`enable_keys_request`	If TRUE then the SM will request security keys from the application if it.
`enable_div_approval`	If TRUE then the SM will request DIV approval from the application before re-establishing a secure connection.
`enable_pairing_auth`	If TRUE then the SM will request pairing authorisation from the application before allowing the peer to start pairing.
`enable_long_term_key`	If TRUE then the SM will request an LTK from the application, otherwise it will recreate internally.

Returns

Nothing.

12.1.6 SMInit

Syntax

```
void SMInit ( uint16 div)
```

Description

Initialising the Security Manager. The application has to supply SMinit() with the latest distributed diversifier, in order to maintain unique keys after a power cycle. This function shall be called after GattInit().

Note:

If the application does not intend to revoke keys and does not consider key uniqueness important, this call can be left out.

Parameters

Parameter	Description
div	Latest distributed diversifier.

Returns

Nothing.

12.1.7 SMKeyRequestResponse

Syntax

```
void SMKeyRequestResponse ( const TYPED_BD_ADDR_T * bd_addr,
const SM_KEYSET_T * keyset )
```

Description

Supplies a key previously stored by the application.
Should only be called in response to SM_KEY_REQUEST_IND

Parameters

Parameter	Description
bd_addr	From SM_KEY_REQUEST_IND
keyset	Pointer to security keys or NULL if none available

Returns

None.

12.1.8 SMLongTermKeyRsp

Syntax

```
 void SMLongTermKeyRsp ( uint16 cid, uint16 * long_term_key, bool
mitm_protection, uint8 key_size )
```

Description

Called by the application in response to an SM_LONG_TERM_KEY_IND to provide the firmware with an externally-generated Long Term Key for the current connection (if it has one) or to indicate that it does not have an LTK available for this connection.

The application can use the EDIV and RAND parameters from the master for anything it likes. However they are not required, and can be set to zero at the master side, and ignored at the slave.

If the application has an LTK, it should provide a pointer to key. The 128-bit key 0x000102030405060708090a0b0c0d0e0f (MSB -> LSB) is stored:

```
uint16 key[] = {0x0e0f, 0x0c0d, 0x0a0b, 0x0809, 0x0607, 0x0405, 0x0203, 0x0001};
```

The application can indicate it does not have a key by setting long_term_key to NULL, or by setting key_size to 0. In this case the Security Manager will recreate the LTK internally using the EDIV and RAND. This may result in a subsequent SM_DIV_APPROVAL_IND message for the application to respond to.

The application should indicate the desired key size. For a 128-bit key, the key size should be set to 16. For shorter keys (for example to meet government export restrictions), the key size can be reduced to any length down to 7 bytes. In this case the firmware will zero the trailing bytes of the key. For example, if the application supplies key 0x123456789abcdef0123456789abcdef0 and states it wants a key size of 8, the firmware will shorten the key to 0x123456789abcdef00000000000000000.

Note:

This function should only be called in response to SM_LONG_TERM_KEY_IND, and shall be called immediately after receiving that event. Unlike internally-generated LTKs, the firmware does not compare the current security mode (unauthenticated or authenticated) against the MITM protection flag for the supplied key.

Parameters

Parameter	Description
`cid`	Connection identifier for established BLE-U connection
`long_term_key`	Pointer to an LTK for this connection, or NULL if the application does not have an LTK for the link. The LTK is an 8-word array.
`mitm_protection`	TRUE if the LTK includes Man-in-the-Middle protection. If the LTK was randomly-generated then this should be TRUE.
`key_size`	Encryption key size (7 to 16 bytes) of the LTK

Returns

Nothing.

12.1.9 SMPairingAuthRsp

Syntax

```
 void SMPairingAuthRsp ( void * data, bool authorised )
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

Authorise or reject a pairing request from the peer device.

If the application does not call (or reference) this function, then all pairing requests will be handled automatically by the firmware. * The 'data' parameter should be copied from the SM_PAIRING_AUTH_IND event sent to the application.

Note:

Should only be called in response to SM_PAIRIING_AUTH_IND, and needs to be called immediately after receiving that event.

Parameters

Parameter	Description
`data`	The data parameter from the original IND event
`authorised`	Boolean flag: TRUE indicates the pairing may proceed. FALSE rejects the pairing request.

Returns

None.

12.1.10 SMPasskeyInput

Syntax

```
 void SMPasskeyInput ( TYPED_BD_ADDR_T * bd_addr, const uint32 *
passkey )
```

Description

After receiving an SM_PASSKEY_INPUT_IND this function call indicates to the Security Manager that the user input a passkey.

Parameters

Parameter	Description
`bd_addr`	From SM_PASSKEY_INPUT_IND
`passkey`	Pointer to passkey value input

Returns

None.

12.1.11 SMPasskeyInputNeg

Syntax

```
 void SMPasskeyInputNeg ( TYPED_BD_ADDR_T * bd_addr)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

After receiving an SM_PASSKEY_INPUT_IND or SM_PASSKEY_DISPLAY_IND this function call indicates to the Security Manager that the user cancelled passkey pairing.

Parameters

Parameter	Description
`bd_addr`	From SM_PASSKEY_INPUT_IND or SM_PASSKEY_DISPLAY_IND

Returns

None.

12.1.12 SMPrivacyGetOwnIrk

Syntax

```
void SMPrivacyGetOwnIrk ( uint16 * irk)
```

Description

Get device's own IRK.

Parameters

Parameter	Description
`irk`	The IRK is 128-bits and consists of 8 consecutive uint16 values, which are stored word wise little endian. This memory should be allocated by the caller. For example: `uint16 irk[8];`

Returns

None.

12.1.13 SMPrivacyMatchAddress

Syntax

```
int SMPrivacyMatchAddress ( const TYPED_BD_ADDR_T * addr, const
uint16 * irk, uint16 num_irk, uint16 size_irk )
```

CSR µEnergy Firmware Library Documentation

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Attempt to resolve an address against a list of IRKs.
Example of key:
The irk 0x000102030405060708090a0b0c0d0e0f (MSB -> LSB) is stored:
`uint16 irk[] = {0x0e0f, 0x0c0d, 0x0a0b, 0x0809, 0x0607, 0x0405, 0x0203, 0x0001};`
Example simple usage:

```
uint16 bonded_irk[8];

bool IsBondedPeer(const TYPED_BD_ADDR_T* addr)
{
    return (SMPrivacyMatchAddress(addr, bonded_irk, 1, 8) == 0);
}
```

Example advanced usage:

```
struct
{
    MyData data;
    SM_KEYSET_T keys;
} peer_data[MAX_PEERS];

const TYPED_BD_ADDR_T* PeerResolve(const TYPED_BD_ADDR_T* addr)
{
    int known_peer = SMPrivacyMatchAddress(addr, peer_data[0].keys.irk,
          MAX_PEERS, sizeof(*peer_data));
    if (known_peer < 0)
        return addr;
    return &peer_data[known_peer].keys.id_addr;
}
```

Parameters

Parameter	Description
`addr`	address to resolve
`irk`	pointer to first entry in list of IRKs. Each IRK is 128-bits and so consists of 8 consecutive uint16 values, which are stored word wise little endian.
`num_irk`	number of IRKs supplied - suggested default 1
`size_irk`	size in words of each IRK record - suggested default 8, though if the list is actually an array of a larger structure it should be the sizeof array element

Returns

zero-based index of the first IRK record that is consistent with addr or negative if none are

Note:

That this function returns immediately with a negative return code if the type of addr is not private resolvable.

CSR µEnergy Firmware Library Documentation

12.1.14 SMPrivacyRegenerateAddress

Syntax

```
bool SMPrivacyRegenerateAddress ( uint24 random)
```

Description

Generate and set a new resolvable private address.
This function may be called at any time to change the current random address to a new resolvable private address. It can be called from master and/or slave configuration.

Parameters

Parameter	Description
`random`	random part of address or if zero an internal random number generator is used

Returns

FALSE on failure or if random >= 0x3FFFFF

12.1.15 SMReadStoredKey

Syntax

```
bool SMReadStoredKey ( const TYPED_BD_ADDR_T * bd_addr,
SM_KEYSET_T * keyset )
```

Description

Retrieves a key from the Security Manager persistent storage.
This may be called with keyset pointer NULL if a simple true/false return code is required.

Parameters

Parameter	Description
`bd_addr`	index address used in SMAddStoredKey()
`keyset`	Pointer to security key buffer or NULL

Returns

TRUE if key available, otherwise FALSE

12.1.16 SMRemoveStoredKey

Syntax

```
bool SMRemoveStoredKey ( const TYPED_BD_ADDR_T * bd_addr)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Remove a key from the Security Manager persistent storage.

Parameters

Parameter	Description
bd_addr	index address used in SMAddStoredKey()

Returns

TRUE if address found in key store, otherwise FALSE

12.1.17 SMRequestSecurityLevel

Syntax

```
bool SMRequestSecurityLevel ( TYPED_BD_ADDR_T * bd_addr)
```

Description

Starts security procedures on the link. For a master device this may involve initiating Encryption or Pairing. For a slave device this will send a Security Request to the master.

Parameters

Parameter	Description
bd_addr	peer address

Returns

FALSE if GAP security mode is NONE or the device is not connected to bd_addr.

12.1.18 SMSetIOCapabilities

Syntax

```
void SMSetIOCapabilities ( sm_io_capabilities io_capabilities)
```

Description

This function set the I/O capabilities of the device.
When initialising the security manager as well as when no security request are activate is possible to set the I/O capabilities of the device.

Parameters

Parameter	Description
`io_capabilities`	I/O capabilities of the device: SM_IO_CAP_NO_INPUT_NO_OUTPUT, SM_IO_CAP_DISPLAY_ONLY, SM_IO_CAP_DISPLAY_YES_NO, SM_IO_CAP_KEYBOARD_ONLY, SM_IO_CAP_KEYBOARD_DISPLAY

Returns

Nothing.

12.1.19 SMSetMaxEncKeySize

Syntax

```
void SMSetMaxEncKeySize ( sm_enc_key_size key_size)
```

Description

This function set the maximum accepted encryption key size.

When initialising the security manager as well as when no security request are activate is possible to set the maximum encryption key size. The key size will be negotiated with the peer and the smaller value of the maximum encryption key length will be used as key size. If this value is small than one of the set minimum encryption key size the paring will fail. Security Manager section 2.3.4

The maximum key size currently support in Low Energy Bluetooth is 16 octet corresponding to a 128 bit key.

Note:

Changing the key size while already encrypted will not change the keys before a key refresh request are send.

Parameters

Parameter	Description
`key_size`	The maximum key size in octets or zero to set default

Returns

Nothing.

12.1.20 SMSetMinEncKeySize

Syntax

```
void SMSetMinEncKeySize ( sm_enc_key_size key_size)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

This function set the minimum accepted encryption key size.

When initialising the security manager as well as when no security request are activate is possible to set the mimimum encryption key size. The key size will be negotiated with the peer and the smaller value of the maximum encryption key size will be used as key size. If this value is smaller than the set minimum encryption key size the paring will fail. Security Manager section 2.3.4

The minimum key size currently support in Low Energy Bluetooth is 7 octet corresponding to a 56 bit key.

Note:

Changing the key size while already encrypted will not change the keys before a key refresh request are send.

Parameters

Parameter	Description
`key_size`	The minimum key size in octets to zero to set default

Returns

Nothing.

12.1.21 SMSetOOBDataPresent

Syntax

```
void SMSetOOBDataPresent (sm_oob_data_present oob_data_present)
```

Description

Currently unsupported.

12.2 Enumerations

12.2.1 enum sm_div_verdict

Syntax

```
enum sm_div_verdict
```

Description

Response codes for the diversifier verdict.

Enumerations

Enumeration	Description
`SM_DIV_APPROVED`	Diversifier and the corresponding LTK valid
`SM_DIV_REVOKED`	Diversifier and the corresponding LTK has been revoked

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

12.2.2 enum sm_io_capabilities

Syntax

```
enum sm_io_capabilities
```

Description

Defines of the I/O capabilities of a device.

Enumerations

Enumeration	Description
SM_IO_CAP_DISPLAY_ONLY	Devices which only have a display
SM_IO_CAP_DISPLAY_YES_NO	Devices which have a display plus a yes and no button
SM_IO_CAP_KEYBOARD_ONLY	Devices which only have a (numeric) keyboard
SM_IO_CAP_NO_INPUT_NO_OUTPUT	Devices which have neither input nor output
SM_IO_CAP_KEYBOARD_DISPLAY	Devices which have both display and (numeric) keyboard

12.2.3 enum sm_key_type

Syntax

```
enum sm_key_type
```

Description

Defines types of security information present in an SM_KEYSET_T by the (1<<sm_key_type) bit being set in SM_KEYSET_T::keys_present.

Enumerations

Enumeration	Description
SM_KEY_TYPE_NONE	Not currently supported
SM_KEY_TYPE_ENC_CENTRAL	Peer LTK + EDIV + RAND
SM_KEY_TYPE_DIV	Local DIV sent to peer
SM_KEY_TYPE_SIGN	Reserved
SM_KEY_TYPE_ID	Peer IRK
SM_BD_ADDR	Peer public/static BD_ADDR

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

12.2.4 enum sm_oob_data_present

Syntax

```
enum sm_oob_data_present { SM_OOB_DATA_NOT_PRESENT = 0, SM_OOB_DATA_PRESENT }
```

Description

Currently unimplemented.

12.3 Typedefs

12.3.1 sm_enc_key_size

Syntax

```
typedef uint8 sm_enc_key_size
```

Description

Defines the size, in octets, of the encryption key.

12.4 Defines

12.4.1 SMPasskeyDisplayed

Definition

```
#define SMPasskeyDisplayed ( bd_addr ) SMPasskeyInput(bd_addr, NULL)
```

Description

After receiving SM_PASSKEY_DISPLAY_IND this function call confirms that the passkey has been displayed.

Note:

If the application wishes to display a different passkey from the one suggested in SM_PASSKEY_DISPLAY_IND it should call SMPasskeyInput() instead to tell the Security Manager the new passkey.

Parameter

Variable	Description
bd_addr	From SM_PASSKEY_DISPLAY_IND

Returns

None.

12.4.2 SM_MAX_ENC_KEY_SIZE

Syntax

```
#define SM_MAX_ENC_KEY_SIZE
((sm_enc_key_size)0x10)
```

12.4.3 SM_MIN_ENC_KEY_SIZE

Syntax

```
#define SM_MIN_ENC_KEY_SIZE
((sm_enc_key_size)0x07)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

13 Generic Access Profile (GAP)

13.1 Functions

13.1.1 GapGetConnChanMask

Syntax

```
ls_err GapGetConnChanMask ( uint16 cid, uint8 * channel_map )
```

Description

Reads the connection channel mask, for the specified connection.

See also Bluetooth Specification v4.0 Volume 2 Part E section 7.8.20, but note that this function is passed a GATT connection ID, not an HCI connection handle. The Channel map is a 5 octet array, where the least significant bit of the first octet corresponds to the first channel. The fourth most significant bit of the fifth octet corresponds to the last data channel.

Parameters

Parameter	Description
`cid`	GATT connection id
`channel_map`	Pointer to uint8 array which will be filled with the connection channel map.

Returns

ls_err_none on success

13.1.2 GapGetRandomAddress

Syntax

```
ls_err GapGetRandomAddress ( BD_ADDR_T * bd_addr)
```

Description

Return the current random address for this device.

If the application has not previously set a random address then the address returned by this function is the default static address, which is set on power-up. A random address is set/changed when the application calls GapSetRandomAddress(), GapSetStaticAddress(), or SMPrivacyRegenerateAddress().

Parameters

Parameter	Description
`bd_addr`	The (untyped) Bluetooth address to use as the device's random address.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

ls_err_none on success, or ls_err_arg if the address is unacceptable for some reason.

13.1.3 GapSetAdvAddress

Syntax

```
ls_err GapSetAdvAddress ( TYPED_BD_ADDR_T const * direct_addr)
```

Description

Set the advertising direct address (i.e. the target peer)
See also Bluetooth Specification v4 Volume 2 Part E section 7.8.5

Parameters

Parameter	Description
`direct_addr`	The Bluetooth address to use in directed advertising.

Returns

ls_err_none on success, or ls_err_arg if the address is unacceptable for some reason.

13.1.4 GapSetAdvChanMask

Syntax

```
ls_err GapSetAdvChanMask ( uint8 const mask)
```

Description

Set the advertising channel mask.
See also Bluetooth Specification v4 Volume 2 Part E section 7.8.5

Parameters

Parameter	Description
`mask`	a bit-mask of the advertising channels to use. Valid values are from 0 to 7.

Returns

ls_err_none on success, or ls_err_arg if the address is unacceptable for some reason.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

13.1.5 GapSetAdvInterval

Syntax

```
ls_err GapSetAdvInterval ( uint32 const adv_min_us, uint32
const adv_max_us )
```

Description

Set the advertising interval min & max times in microseconds.

This function is called by the application to store the advertising interval min and max times in microseconds. Valid ranges for the parameters are 20ms (20000) to 10240ms (10240000). The maximum interval must be greater than the minimum interval.

The firmware is free to use any advertising interval between the minimum and maximum values. The current implementation will always use the maximum value, to save power by transmitting as little as possible. However, future releases of the firmware library may use a different value if it improves advertising performance when one or more connections are also active.

Parameters

Parameter	Description
`adv_min_us`	Minimum advertising interval requested by application.
`adv_max_us`	Maximum advertising interval requested by application.

Returns

ls_err_none on success, or ls_err_arg if the intervals are unacceptable for some reason.

13.1.6 GapSetConnChanMask

Syntax

```
ls_err GapSetConnChanMask ( const uint8 * channel_map)
```

Description

Set or update the connection channel mask, used when the device is master of the connection.

See also Bluetooth Specification v4.0 Volume 2 Part E section 7.8.19 The Channel map is a 5 octet array, where the least significant bit of the first octet corresponds to the first channel. The fourth most significant bit of the fifth octet corresponds to the last data channel. The three most significant bits of the fifth octet are cleared to restrict the mask to data channels.

Parameters

Parameter	Description
`channel_map`	Pointer to uint8 array containing the bit-mask of channels to use when operating as master. Must have at least 2 bits set.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

ls_err_none on success, or ls_err_arg if the mask is unacceptable for some reason.

13.1.7 GapSetMode

Syntax

```
ls_err GapSetMode ( gap_role const role, gap_mode_discover
const discover, gap_mode_connect const connect, gap_mode_bond const
bond, gap_mode_security const security )
```

Description

Set the GAP modes.

The discover parameter is used to select both the Discoverability mode (when operating in the Peripheral role) and the Discovery Procedure (when operating in the Central role). An Observer does not use the discover parameter and will ignore the setting. A Broadcaster is not allowed to be discoverable, therefore with this role discover must always be set to gap_mode_discover_no.

The connect parameter is only used when the role is Peripheral. In other roles it must be set to gap_mode_connect_no.

The bonding and security flags are used for connections, therefore apply to Central and Peripheral roles. In other roles they are ignored.

Calling this function will reset the scan type to ls_scan_type_active and the advertising channel mask to 0x07 (i.e. all channels available).

Parameters

Parameter	Description
`role`	The GAP operational mode to use.
`discover`	The GAP discovery mode to use.
`connect`	The GAP connection mode to use.
`bond`	The GAP bonding mode to use.
`security`	The GAP security mode to use.

Returns

ls_err_none for success, or an error code on failure.

13.1.8 GapSetRandomAddress

Syntax

```
ls_err GapSetRandomAddress ( const BD_ADDR_T * ra)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Set the random address for this device.
The random address may subsequently be used in scan/advertise.

Parameters

Parameter	Description
ra	The Bluetooth address to use as the device's random address.

Returns

ls_err_none on success, or ls_err_arg if the address is unacceptable for some reason.

13.1.9 GapSetScanInterval

Syntax

```
 ls_err GapSetScanInterval ( uint32 const interval_us, uint32
const window_us )
```

Description

Set the scan interval and scan window times in microseconds.
Range: 2.5 msec (2500) to 10240 msec (10240000)
See also Bluetooth Specification v4 Volume 2 Part E section 7.8.10

Parameters

Parameter	Description
interval_us	The scan interval in microseconds.
window_us	The scan window in microseconds.

Returns

ls_err_none on success, or ls_err_arg if the parameters are unacceptable for some reason.

13.1.10 GapSetScanType

Syntax

```
void GapSetScanType ( ls_scan_type const st)
```

Description

Set the scan type. The firmware defaults to passive scanning.
See also Bluetooth Specification v4 Volume 2 Part E section 7.8.10

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
st	The type of scan to perform when scanning.

Returns

Nothing.

13.1.11 GapSetStaticAddress

Syntax

```
void GapSetStaticAddress ( void )
```

Description

Set the static address for this device.
The static address is set at manufacture and may be used in scan or advertise if a Random type address is selected. This function should be called after GapSetRandomAddress() or SMPrivacyRegenerateAddress() to return to using the static address

Parameters

Parameter	Description
None	-

Returns

Nothing.

13.2 Enumerations

13.2.1 enum gap_feature_set

Syntax

```
enum gap_feature_set
```

Description

GAP feature set.
See Bluetooth Specification v4 Volume 6 Part B section 4.6, and GAP Volume 3 section 9.
LE supported features is limited to encryption.

13.2.2 enum gap_mode_bond

Syntax

```
enum gap_mode_bond
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

GAP Bonding modes. See Vol 3 Part C Section 9.4 of the Bluetooth Specification Version 4.

Enumerations

Enumeration	Description
`gap_mode_bond_no`	GAP 9.4.2.1. A device in the non-bondable mode does not allow a bond to be created with a peer device. A device in the Peripheral or Central role shall support the non-bondable mode.
`gap_mode_bond_yes`	GAP 9.4.3.1. A device in the bondable mode allows a bond to be created with a peer device also in the bondable mode. A device in the Peripheral or Central role shall support the bondable mode.

13.2.3 enum gap_mode_connect

Syntax

```
enum gap_mode_connect
```

Description

GAP Connection modes. See Vol 3 Part C Section 9.3 of the Bluetooth Specification Version 4.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumerations

Enumeration	Description
`gap_mode_connect_no`	GAP 9.3.2. A device in the non-connectable mode shall not allow a connection to be established. While a device is in the Peripheral role it shall support the non-connectable mode.A Peripheral device in the non-connectable mode may send non-connectable undirected advertising events or discoverable undirected advertising events.
`gap_mode_connect_directed`	GAP 9.3.3. A device in the Peripheral role may support the directed connectable mode. **Note:** To use directed connectable mode with low duty cycle this option must be chosen.For enabling the advertisements, application call the following API's: (1) GapSetMode to set the GAP role as peripheral, type of advertisements (undirected, directed etc) and few other settings. (2) GattConnectReq() to start the advertisements. This API takes an input parameter i.e connection flag(masks) containing the type of advertisement. In case of low duty cycle directed advertisements flag is OR-ed with L2CAP_CONNECTION_SLAVE_DIRECTED_LDC.
`gap_mode_connect_undirected`	GAP 9.3.4. A device in the Peripheral role may support the undirected connectable mode.

13.2.4 enum gap_mode_discover

Syntax

```
enum gap_mode_discover
```

Description

GAP Discovery modes. See Vol 3 Part C Section 9.2 of the Bluetooth Specification Version 4.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumerations

Enumeration	Description
gap_mode_discover_no	GAP 9.2.2.2. A device in non-discoverable mode and shall support neither limited nor general discoverable mode.
gap_mode_discover_limited	GAP 9.2.3.2. While a device is the Peripheral role the device may support the limited discoverable mode.But while a device is in the Broadcaster, Observer or Central role the device shall not support the limited discoverable mode.
gap_mode_discover_general	GAP 9.2.4.2. While a device is in the Peripheral role the device may support the general discoverable mode.But while a device is in the Broadcaster, Observer or Central role the device shall not support the general discoverable mode.

13.2.5 enum gap_mode_security

Syntax

```
enum gap_mode_security
```

Description

GAP Security modes. See Vol 3 Part C Section 10.2 of the Bluetooth Specification Version 4. Only LE Security Mode 1 modes (section 10.2.1) are specified here.

Enumerations

Enumeration	Description
gap_mode_security_none	GAP 10.2. No security.
gap_mode_security_unauthenticate	GAP 10.3. Unauthenticated pairing involves performing the pairing procedure with authentication set to "No MITM protection" (i.e. no prior link-key theft and impersonation guard).
gap_mode_security_authenticate	GAP 10.3. Authenticated pairing involves performing the pairing procedure defined in Bluetooth Core Spec. Vol.3, Part H Section 2.1, with the authentication set to "MITM protection" (i.e. new exchange of link keys).

13.2.6 enum gap_role

Syntax

```
enum gap_role
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

GAP LE operational modes. See Vol 3 Part C Section 2.2.2 of the Bluetooth Specification Version 4.

Enumerations

Enumeration	Description
`gap_role_broadcaster`	GAP 2.2.2.1. A device operating in the Broadcaster role is a device that sends advertising events as described in Bluetooth Core Spec. Vol.6, Part B, Section 4.4.2.
`gap_role_observer`	GAP 2.2.2.2. A device operating in the Observer role is a device that receives advertising events as described in Bluetooth Core Spec. Vol.6, Part B Section 4.4.3.
`gap_role_peripheral`	GAP 2.2.2.3. Any device that accepts the establishment of an LE physical link is referred to as being in the Peripheral role, and will be in the Slave role in the Link Layer Connection State as described in Bluetooth Core Spec. Vol.6, Part B Section 4.5.
`gap_role_central`	GAP 2.2.2.4. A device that supports the Central role initiates the establishment of a physical connection and will be in the Master role in the Link Layer Connection State as described in Bluecore Core spec. Vol.6, Part B Section 4.5.

13.3 Defines

13.3.1 Discoverability Timeouts

GAP spec. sections 4.1 and 16.

13.3.2 GAP_TGAP_100

Definition

```
#define GAP_TGAP_100 10240000U
```

Description

10.24s - time to perform device discovery

13.3.3 GAP_TGAP_101

Definition

```
#define GAP_TGAP_101 10625U
```

Description

10.625ms - time to be discoverable

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

13.3.4 GAP_TGAP_102

Definition

```
#define GAP_TGAP_102 2560000U
```

Description

2.56s - time between being discoverable

13.3.5 GAP_TGAP_103

Definition

```
#define GAP_TGAP_103 30720000U
```

Description

30.72s - min. time to be discoverable

13.3.6 GAP_TGAP_104

Definition

```
#define GAP_TGAP_104 60000000U
```

Description

60.00s - max. time to be discoverable

13.3.7 GAP_TGAP_lim_disc_adv_max

Definition

```
#define GAP_TGAP_lim_disc_adv_max 30720000U
```

Description

30.72s - limited discovery max. advertise time

13.3.8 GAP_TGAP_lim_disc_adv_intvl_min

Definition

```
#define GAP_TGAP_lim_disc_adv_intvl_min 250000U
```

Description

250ms - limited discovery min. advertising interval

13.3.9 GAP_TGAP_lim_disc_adv_intvl_max

Definition

```
#define GAP_TGAP_lim_disc_adv_intvl_max 500000U
```

Description

500ms - limited discovery max. advertising interval

13.3.10 GAP_TGAP_gen_disc_adv_intvl_min

Definition

```
#define GAP_TGAP_gen_disc_adv_intvl_min 1280000U
```

Description

1.28s - general discovery min. advertising interval

13.3.11 GAP_TGAP_gen_disc_adv_intvl_max

Definition

```
#define GAP_TGAP_gen_disc_adv_intvl_max 2560000U
```

Description

2.56s - general discovery min. advertising interval

13.3.12 Scan Timeouts

See Bluetooth Specification v4 Volume 6 Part B section 4.4.3 Note - scan interval and scan window share the same start time, so the two being equal means continuous scanning.

13.3.13 GAP_TGAP_lim_disc_scan_min

Definition

```
#define GAP_TGAP_lim_disc_scan_min 10240000U
```

Description

10.24s - limited discovery min. time to perform scan

13.3.14 GAP_TGAP_lim_disc_scan_intvl

Definition

```
#define GAP_TGAP_lim_disc_scan_intvl 11250U
```

Description

11.25ms - limited discovery scan interval

13.3.15 GAP_TGAP_lim_disc_scan_window

Definition

```
#define GAP_TGAP_lim_disc_scan_window 11250U
```

Description

11.25ms - limited discovery scan duration

13.3.16 GAP_TGAP_gen_disc_scan_min

Definition

```
#define GAP_TGAP_gen_disc_scan_min 10240000U
```

Description

10.24s - general discovery min. time to perform scan

13.3.17 GAP_TGAP_gen_disc_scan_intvl

Definition

```
#define GAP_TGAP_gen_disc_scan_intvl 11250U
```

Description

11.25ms - general discovery scan interval

13.3.18 GAP_TGAP_gen_disc_scan_window

Definition

```
#define GAP_TGAP_gen_disc_scan_window 11250U
```

Description

11.25ms - general discovery scan duration

14 Link Supervisor

- LS Advscan Interface
- LS to APP Interface

15 LS Advscan Interface

15.1 Functions

15.1.1 GapLsFindAdType

Syntax

```
 uint16 GapLsFindAdType (HCI_EV_DATA_ULP_ADVERTISING_REPORT_T
*const evt_data, ad_type type, uint16 * ad_field, uint16 max_length )
```

Description

Search an advertising report for the specified GAP AD type.

This helper function can be used to parse an advertising or scan response data field for GAP-defined AD types (see also ad_type). If the requested type is found, the associated data (excluding the AD type header byte) will be copied into the buffer supplied by the caller (up to a maximum length). The total length copied is then returned.

Parameters

Parameter	Description
`evt_data`	Pointer to an advertising report event
`type`	The AD type to search for
`ad_field`	Pointer to a buffer to copy AD data into (as packed data)
`max_length`	The maximum number of bytes to copy

Returns

Number of bytes copied (0 if type was not found or had empty data).

15.1.2 LsSetTgapConnParamTimeout

Syntax

```
 ls_err LsSetTgapConnParamTimeout ( uint16 timeout)
```

Description

Set TGAP(conn_param_timeout) to a new value.

This function enables a slave to initiate master-slave data bursts, or an operating system to quickly negotiate optimal connection parameters. It sets TGAP(conn_param_timeout) to a user-selected value.

The slave shall not send an L2CAP Connection Parameter Update Request within TGAP(conn_param_timeout) of an L2CAP Connection Parameter Update Response being received (refer to Bluetooth Specification v4.0 Volume 3 Part C Section 9.3.9.2).

Note:

Using this function can reduce battery life. It is the application's responsibility to restore the recommended value as soon as possible after using this function.

Parameters

Parameter	Description
`timeout`	The timeout value in units of 10ms. Minimum is zero. Maximum is the recommended value of 30 seconds (refer to Bluetooth Specification v4.0 Volume 3 Part C Section 16).

Returns

ls_err_none on success, or an error code if the action fails.

15.1.3 LsStartStopAdvertise

Syntax

```
 ls_err LsStartStopAdvertise ( bool const go, whitelist_mode
const wl_mode, ls_addr_type const addr_type )
```

Description

Start or stop advertising, with or without a whitelist.

Configuration is stored in GAP; the application may have called LsStoreAdvScanData() before calling this function.

Parameters

Parameter	Description
`go`	TRUE (re)starts advertising, FALSE stops advertising.
`wl_mode`	Whether or not to use the whitelist.
`addr_type`	Whether to advertise using the device's public Bluetooth address, or a private "random" address.

Returns

ls_err_none on success, or an error code if the action fails.

15.1.4 LsStartStopScan

Syntax

```
 ls_err LsStartStopScan ( bool go, whitelist_mode wl_mode,
ls_addr_type addr_type )
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Start or stop scanning, with or without a whitelist.

If the device is configured in the GAP Observer role, then the scan will return all advertising events received from other devices. In this role the whitelist can optionally be used to filter out events from device the application is not interested in.

If the device is configured in the GAP Central role, then the device will use the discovery mode (set by the discovery mode parameter to GapSetMode() at the same time as enabling the GAP Central role) to determine what type of scan should be performed. If the discovery mode is Limited Discovery or General Discovery then the advertising events will be filtered according to the rules of those procedures. Any other discovery mode is considered invalid for a GAP Central device, but will simply result in an un-filtered scan (useful if a GAP Central device wants to perform an Observer-style scan with or without whitelisting).

Parameters

Parameter	Description
`go`	TRUE (re)starts scanning, FALSE stops scanning.
`wl_mode`	Whether or not to use the whitelist (not used with the Limited Discovery or General Discovery procedures of the Central role).
`addr_type`	Whether to scan using the device's public Bluetooth address, or a private "random" address.

Returns

ls_err_none on success, or an error code if the action fails.

15.1.5 LsStoreAdvDataNoAdFlags

Syntax

```
ls_err LsStoreAdvDataNoAdFlags (uint16 const len, uint8 *const data)
```

Description

Set Advertising with no AD FLAGS. (ONLY USE FOR NON-CONNECTABLE ADVERTISING)

This function can be used by the application to add non connectable advertising data without the flags AD Structure. Each call to the function will add a single AD Structure (refer to Bluetooth specification Vol.3 Part C Section 11). Repeated calls will append new structures, to build up the data content.

The application should not include the "length" parameter within the supplied octet array - the GAP layer will add the length field in the appropriate position. The first octet of the array should be the AD Type field (see ad_type)

The data will be stored within GAP as AD structures, to a maximum of 31 octets (including the length octets); if the application exceeds this capacity an error is returned and the AD Structure is not stored.

An advertising data packet should not contain more than one instance for each Service UUID data size.

If the application wishes to clear any existing data it should call the function with the length parameter set to zero. This will clear all stored data.

Calling this function with the length set to a non zero value will stop the application from doing connectable adverts, until this function or LsStoreAdvScanData is called with the length parameter set to zero.

Advertising data is only used when the device is in the Broadcaster or Peripheral role. However, advertising data can be set while in any role, e.g. an application using the Observer role can update the advertising data.

Parameters

Parameter	Description
`len`	The number of bytes of data supplied.
`data`	A byte array of the AD Structure to add. The first byte will be the AD Type field (see ad_type). It is the caller's responsibility to ensure that the remainder of the data is correctly formatted and consistent with the specified AD Type.

Returns

ls_err_none for success, or an error code if the store fails.

15.1.6 LsStoreAdvScanData

Syntax

```
 ls_err LsStoreAdvScanData ( uint16 const len, uint8 *const
data, ad_src const src )
```

Description

Set Advertising or Scan Response data.

This function is called by the application to add either advertising or scan response data. Each call to the function will add a single AD Structure (refer to Bluetooth specification Vol.3 Part C Section 11). Repeated calls will append new structures, to build up the data content.

The application should not include the "length" parameter within the supplied octet array - the GAP layer will add the length field in the appropriate position. The first octet of the array should be the AD Type field (see ad_type)

The data will be stored within GAP as AD structures, to a maximum of 31 octets (including the length octets); if the application exceeds this capacity an error is returned and the AD Structure is not stored.

Note:

> The GAP layer will automatically add the AD Flags structure to the start of the Advertising data and manage the flags values. The application itself is not allowed to add this AD Type.

An extended inquiry response or advertising data packet should not contain more than one instance for each Service UUID data size.

If the application wishes to clear any existing data it should call the function with the length parameter set to zero. This will clear all stored data including the AD Flags structure. (It is therefore not possible to define a string that only contains the AD Flags structure).

Advertising data is only used when the device is in the Broadcaster or Peripheral role. However, advertising data can be set while in any role, e.g. an application using the Observer role can update the advertising data.

CSR µEnergy Firmware Library Documentation

Parameters

Parameter	Description
`len`	The number of bytes of data supplied.
`data`	A byte array of the AD Structure to add. The first byte will be the AD Type field (see ad_type). It is the caller's responsibility to ensure that the remainder of the data is correctly formatted and consistent with the specified AD Type.
`src`	A flag indicating whether the data supplied is advertising data (ad_src_advertise) or scan response data (ad_src_scan_rsp).

Returns

ls_err_none for success, or an error code if the store fails.

15.2 Enumerations

15.2.1 enum ad_type

Syntax

```
enum ad_type
```

Description

AD Type values.
Transmitted advertising data contains a sequence of AD Structures. Each contains a AD Type field which defines the remainder of the structure. See Bluetooth Specification Vol. 3 (GAP) Part C Section 11 and the latest Core Specification Supplement, CSSn Part A, for more details.

Enumerations

Enumeration	Description
`AD_TYPE_FLAGS`	Advertising Data Flags as defined in CSS v4 Part A Section 1.3.
`AD_TYPE_SERVICE_UUID_16BIT`	Incomplete list of 16-bit Service Class UUIDs.
`AD_TYPE_SERVICE_UUID_16BIT_LIST`	Complete list of 16-bit Service Class UUIDs.
`AD_TYPE_SERVICE_UUID_32BIT`	Incomplete list of 32-bit Service Class UUIDs.
`AD_TYPE_SERVICE_UUID_32BIT_LIST`	Complete list of 32-bit Service Class UUIDs.
`AD_TYPE_SERVICE_UUID_128BIT`	Incomplete list of 128-bit Service Class UUIDs.
`AD_TYPE_SERVICE_UUID_128BIT_LIST`	Complete list of 128-bit Service Class UUIDs.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Enumeration	Description
AD_TYPE_LOCAL_NAME_SHORT	Shortened local device name. This supplied data must match the first n characters of the complete name. See CSS v4 Part A Section 1.2.
AD_TYPE_LOCAL_NAME_COMPLETE	Complete local device name. See CSS v4 Part A Section 1.2.
AD_TYPE_TX_POWER	Transmitted power level. May be used with received RSSI to estimate the RF path loss, see CSS v4 Part A Section 1.5.
AD_TYPE_OOB_DEVICE_CLASS	Class of Device. See Bluetooth Assigned Numbers, https://www.bluetooth.org/assigned-numbers.
AD_TYPE_OOB_SSP_HASH_C_192	Out of Band simple pairing hash, (P-192 elliptic curve). To be sent OOB only, not over the air.
AD_TYPE_OOB_SSP_RANDOM_R_192	Out of Band simple pairing randomizer, (P-192 elliptic curve). To be sent OOB only, not over the air.
AD_TYPE_SM_TK	Security Manager TK value. To be sent OOB only, not over the air. See CSS v4 Part A Section 1.8.
AD_TYPE_SM_FLAGS	Security Manager Out of Band Flags. To be sent OOB only, not over the air. See CSS v4 Part A Section 1.7.
AD_TYPE_SLAVE_CONN_INTERVAL	Slave connection interval range. See CSS v4 Part A Section 1.9.
AD_TYPE_SERVICE_SOLICIT_UUID_16BIT	List of 16-bit Service Solicitation UUIDs.
AD_TYPE_SERVICE_SOLICIT_UUID_128BIT	List of 128-bit Service Solicitation UUIDs.
AD_TYPE_SERVICE_DATA_UUID_16BIT	Service Data - 16 bit UUID. See CSS v4 Part A Section 1.11.
AD_TYPE_PUBLIC_TARGET_ADDRESS	Public target address. See CSS v4 Part A Section 1.13.
AD_TYPE_RANDOM_TARGET_ADDRESS	Random target address. See CSS v4 Part A Section 1.14.
AD_TYPE_APPEARANCE	The Appearance data type defines the external appearance of the device. See CSS v4 Part A Section 1.12.
AD_TYPE_ADVERTISING_INTERVAL	Advertising interval. See CSS v4 Part A Section 1.15.
AD_TYPE_LE_BT_ADDRESS	Local device's Bluetooth address and type. To be sent OOB only, not over the air. See CSS v4 Part A Section 1.16.
AD_TYPE_LE_ROLE	Role of the local device. To be sent OOB only, not over the air. See CSS v4 Part A Section 1.17.

CSR

Enumeration	Description
AD_TYPE_OOB_SSP_HASH_C_256	Out of Band simple pairing hash, (P-256 elliptic curve). To be sent OOB only, not over the air.
AD_TYPE_OOB_SSP_RANDOM_R_256	Out of Band simple pairing randomizer, (P-256 elliptic curve). To be sent OOB only, not over the air.
AD_TYPE_SERVICE_SOLICIT_UUID_32BIT	List of 32-bit Service Solicitation UUIDs.
AD_TYPE_SERVICE_DATA_UUID_32BIT	Service Data - 32 bit UUID. See CSS v4 Part A Section 1.11.
AD_TYPE_SERVICE_DATA_UUID_128BIT	Service Data - 128 bit UUID. See CSS v4 Part A Section 1.11.
AD_TYPE_MANUF	Manufacturer specific data. See CSS v4 Part A Section 1.4.

15.2.2 enum ls_advert_type

Syntax

```
enum ls_advert_type {

 ls_advert_connectable_undirected = 0,
 ls_advert_connectable_directed = 1,
 ls_advert_discoverable = 2,
 ls_advert_non_connectable = 3,

 ls_advert_scan_response = 4 }
```

Description

Type corresponding to advert types received in an Advertising Report.

15.2.3 enum ad_src

Syntax

```
enum ad_src { ad_src_advertise, ad_src_scan_rsp }
```

15.3 Defines

15.3.1 Advertising Data Flags

Definitions for the Advertising Data Flags data type (AD_TYPE_FLAGS)

15.3.2 AD_FLAG_LE_LIMITED_DISCOVERABLE

Definition

```
#define AD_FLAG_LE_LIMITED_DISCOVERABLE 0x01
```

15.3.3 AD_FLAG_LE_GENERAL_DISCOVERABLE

Definition

```
#define AD_FLAG_LE_GENERAL_DISCOVERABLE 0x02
```

15.3.4 AD_FLAG_BR_EDR_NOT_SUPPORTED

Definition

```
#define AD_FLAG_BR_EDR_NOT_SUPPORTED 0x04
```

15.3.5 AD_FLAG_SIMUL_LE_BREDR_CONTROLLER

Definition

```
#define AD_FLAG_SIMUL_LE_BREDR_CONTROLLER 0x08
```

15.3.6 AD_FLAG_SIMUL_LE_BREDR_HOST

Definition

```
#define AD_FLAG_SIMUL_LE_BREDR_HOST 0x10
```

15.3.7 Security Manager Flags

Definitions for the Security Manager OOB Flags data type (AD_TYPE_SM_FLAGS).

15.3.8 AD_SM_FLAG_OOB_PRESENT

Definition

```
#define AD_SM_FLAG_OOB_PRESENT 0x01
```

15.3.9 AD_SM_FLAG_LE_SUPPORTED_HOST

Definition

```
#define AD_SM_FLAG_LE_SUPPORTED_HOST 0x02
```

15.3.10 AD_SM_FLAG_SIMUL_LE_BREDR_HOST

Definition

```
#define AD_SM_FLAG_SIMUL_LE_BREDR_HOST 0x04
```

15.3.11 AD_SM_FLAG_RANDOM_ADDRESS

Definition

```
#define AD_SM_FLAG_RANDOM_ADDRESS 0x08
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

16 LS to APP Interface

16.1 Functions

16.1.1 LsAddWhiteListDevice

Syntax

```
ls_err LsAddWhiteListDevice ( TYPED_BD_ADDR_T *const addrt)
```

Description

Add a device to the whitelist.

Parameters

Parameter	Description
addrt	The typed Bluetooth address of the device to add to the whitelist.

Returns

le_err_none on success, or an appropriate error code on failure.

16.1.2 LsConnectionParamUpdateReq

Syntax

```
ls_err LsConnectionParamUpdateReq ( TYPED_BD_ADDR_T * bdAddr,
ble_con_params * new_params )
```

Description

Request an update to the connection parameters.
The Connection Parameter Update procedure in initiated as described in Bluetooth Specification v4.0 Volume 6 Part B Section 5.1.1
When the procedure finishes, an LS_CONNECTION_PARAM_UPDATE_CFM event is raised

Parameters

Parameter	Description
bdAddr	Typed Bluetooth address of the connected peer, identifying the link to update
new_params	New connection parameters (minimum & maximum interval, slave latency, & supervision timeout)

Returns

ls_err_none for success, or an error code if the procedure cannot be initiated.

16.1.3 LsConnectionUpdateSignalingRsp

Syntax

```
void LsConnectionUpdateSignalingRsp ( uint16 con_handle, uint16
sig_identifier, bool accepted )
```

Description

Application response to an LS_CONNECTION_UPDATE_SIGNALLING_IND event.

Parameters

Parameter	Description
`con_handle`	The connection handle this response applies to. Should be the same as the con_handle received in the LS_CONNECTION_UPDATE_SIGNALLING_IND event.
`sig_identifier`	An identifier for the specific connection update signal that this response applies to. Should be the same as the sig_identifier received in the LS_CONNECTION_UPDATE_SIGNALLING_IND event.
`accepted`	TRUE if the updated parameters are acceptable to the application, FALSE otherwise.

Returns

Nothing.

16.1.4 LsDeleteWhiteListDevice

Syntax

```
ls_err LsDeleteWhiteListDevice ( TYPED_BD_ADDR_T *const addrt)
```

Description

Delete a device from the whitelist.

Parameters

Parameter	Description
`addrt`	The typed Bluetooth address of the device to add to the whitelist.

Returns

le_err_none on success, or an appropriate error code on failure.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

16.1.5 LsDisableSlaveLatency

Syntax

```
ls_err LsDisableSlaveLatency ( bool disable)
```

Description

Ignore slave latency value in a connection.

In some cases, a device operating as BLE slave may wish to not respect the master's wish for slave latency. This function allows the slave to select whether it obeys the master's requests to use latency.

Calling this function affects all subsequent connections made as a slave device. If called during a connection, it will take effect at the end of the next latency period (or when it would have been had latency been enabled).

If slave latency is enabled during a connection, then the latency will be set to the value last requested by the master (either at connection time or in a subsequent connection parameter update).

Note:

Disabling slave latency must only be done after careful consideration, and for as short a period as possible, as it will have a detrimental effect on power consumption.

Parameters

Parameter	Description
`disable`	Set TRUE to ignore slave latency on the link, and FALSE to allow it.

Returns

ls_err_none on success or an appropriate error code on failure.

16.1.6 LsHoldTxUntilRx

Syntax

```
ls_err LsHoldTxUntilRx ( uint16 cid, bool mode )
```

Description

Enable or disable delayed data packet transmission at the radio.

Enabling this mode will delay transmitting data packets until a data packet has been received from the peer. This mode is not for general use as it can stall the BLE connection, causing protocol stack timeouts and link loss.

Parameters

Parameter	Description
`cid`	GATT Connection Identifier for the link
`mode`	Boolean flag to enable delayed data TX

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

ls_err_none on success

16.1.7 LsRadioEventNotification

Syntax

```
ls_err LsRadioEventNotification ( uint16 cid, radio_event evt )
```

Description

Enable or disable notification to the application of radio events for a given GATT connection.

In some specific sensor-polling applications it can be useful to trigger the application to run once per radio event. This feature can be used to minimise the current consumption of the device, as sensor polling and radio activity can be performed within a single "wakeup" of the chip, and remove the need for the application to run a separate timer at the same rate as the link's Connection Interval.

For each radio event that occurs, the firmware will send an LS_RADIO_EVENT_IND event to the application event handler, AppProcessLmEvent(), with message payload type LS_RADIO_EVENT_IND_T.

When a connection is established, the application by default will not be notified of any radio activity, as for most purposes it is not useful (e.g. normal CFM messages for GATT messages are sufficient to keep the application running).

Note that when operating as a BLE slave, events are only generated upon successful RX transactions. No events will be generated if the master was not heard or if the slave is not listening due to latency.

Parameters

Parameter	Description
`cid`	GATT Connection Identifier for the link
`activity`	Level of activity to report for the link

Returns

ls_err_none on success, or an appropriate error code on failure.

16.1.8 LsReadRemoteUsedFeatures

Syntax

```
ls_err LsReadRemoteUsedFeatures ( uint16 cid)
```

Description

Trigger a Remote Used Features exchange with the connected peer.

The remote version information will be returned via an LM_EV_REMOTE_USED_FEATURES message sent to the application event handler, AppProcessLmEvent(), with message payload type LM_EV_REMOTE_USED_FEATURES_T. This event looks very similar to the corresponding HCI event (refer to Bluetooth Specification v4.0 Volume 2 Part E Section 7.7.65.4 for full details of this event). However, the HCI connection handle parameter is instead mapped onto the GATT Connection Identifier supplied in the 'cid' parameter.

Note:

This function can only be used when the device is connected as a BLE master. An error will be returned if it is called while the device is a slave.

Parameters

Parameter	Description
`cid`	GATT Connection Identifier for the link

Returns

ls_err_none on success, or an appropriate error code on failure.

16.1.9 LsReadRemoteVersionInformation

Syntax

```
ls_err LsReadRemoteVersionInformation ( uint16 cid)
```

Description

Trigger a remote version information exchange with the connected peer.

The remote version information will be returned via an LM_EV_REMOTE_VERSION_INFO message sent to the application event handler, AppProcessLmEvent(), with message payload type LM_EV_REMOTE_VERSION_INFO_T. This event looks very similar to the corresponding HCI event (refer to Bluetooth Specification v4.0 Volume 2 Part E Section 7.7.12 for full details of this event). However, the HCI connection handle parameter is instead mapped onto the GATT Connection Identifier supplied in the 'cid' parameter.

Parameters

Parameter	Description
`cid`	GATT Connection Identifier for the link

Returns

ls_err_none on success, or an appropriate error code on failure.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

16.1.10 LsReadRssi

Syntax

```
ls_err LsReadRssi ( uint16 cid, int8 * rssi_val )
```

Description

Return the last Received Signal Strength Indication for a connection.

The RSSI value is an absolute receiver signal strength value in dBm, to 6dBm accuracy. If the RSSI cannot be read, the (maximum) value 127 will be returned.

Parameters

Parameter	Description
cid	GATT Connection Identifier for the link
rssi_val	Pointer to a variable into which the RSSI value shall be stored

Returns

ls_err_none on success, or an appropriate error code on failure.

16.1.11 LsReadTransmitPowerLevel

Syntax

```
ls_err LsReadTransmitPowerLevel ( int8 * tx_power_lvl)
```

Description

Retrieve the current transmit power level setting.

LE does not automatically alter transmission power levels unlike BR/EDR, so the value retrieved will be whatever has been configured. The initial power level is set by the CS key tx_power_level ("Transmit power level") but this may be updated during system operation using the LsSetTransmitPowerLevel() function.

Warning:The returned power level is an approximate transmit power level in dBm estimated from the current power level setting; the actual transmitted power will depend on the physical characteristics of the board components and layout.

Parameters

Parameter	Description
tx_power_lvl	Pointer to a variable into which the transmit power level shall be stored.

Returns

ls_err_none on success.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

16.1.12 LsReadWhiteListMaxSize

Syntax

```
ls_err LsReadWhiteListMaxSize ( uint8 * sz)
```

Description

Read the capacity of the whitelist.

Parameters

Parameter	Description
sz	Pointer to a variable into which the maximum size will be written.

Returns

ls_err_none on success

16.1.13 LsResetWhiteList

Syntax

```
ls_err LsResetWhiteList ( void )
```

Description

Reset and clear the whitelist.

Parameters

Parameter	Description
None	-

Returns

ls_err_none on success, or an appropriate error code on failure.

16.1.14 LsRxTimingReport

Syntax

```
ls_err LsRxTimingReport ( uint16 cid, bool mode )
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

Enable or disable reporting to the application various packet timing parameters when data packets are received at the radio.

Warning:This feature is only supoprted when operating as a BLE Master. Enabling this feature as a slave may result in undefined behaviour.

Parameters

Parameter	Description
`cid`	GATT Connection Identifier for the link
`bool`	enable (T) or disable (F) the feature

Returns

ls_err_none on success

16.1.15 LsSetNewConnectionParamReq

Syntax

```
 ls_err LsSetNewConnectionParamReq ( ble_con_params *
conn_params, uint16 con_min_ce_len, uint16 con_max_ce_len, uint16
con_scan_interval, uint16 con_scan_window )
```

Description

Set connection parameters for new connections.

Devices operating as a BLE master will use these parameters for all subsequent connections. Changing these parameters will not affect existing connections - use LsConnectionParamUpdateReq() to do that.

This function is not used on slave-role devices.

Parameters

Parameter	Description
`conn_params`	Connection parameters (minimum & maximum interval, slave latency, & supervision timeout)
`con_min_ce_len`	Expected minimum Connection Event length (can be 0)
`con_max_ce_len`	Expected maximum Connection Event length (can be 0)
`con_scan_interval`	Scan interval during connection establishment
`con_scan_window`	Scan window during connection establishment

Returns

Status. Invalid parameters will be rejected

16.1.16 LsSetTransmitPowerLevel

Syntax

```
ls_err LsSetTransmitPowerLevel ( uint8 tx_power_lvl)
```

Description

Update the current transmit power level setting.

This function allows the Application code to change the transmit power dynamically. The level argument is not a level in dBm, it's an index value into the transmit power table.

This function can be called by the Application at any time. If the radio is active with transmissions then the power will alter immediately, if the radio is inactive the new power level will be observed when the radio next starts transmitting.

Warning:There are a number of configurable levels. The limits are defined by LS_MIN_TRANSMIT_POWER_LEVEL and LS_MAX_TRANSMIT_POWER_LEVEL Settings outside this range are not legal and the results are undefined.

Parameters

Parameter	Description
`tx_power_lvl`	Power level setting index, default from the CS key tx_power_level

Returns

ls_err_none on success.

16.2 Enumerations

16.2.1 enum whitelist_mode

Syntax

```
enum whitelist_mode
```

Description

Whitelist usage.

Enumerations

Enumeration	Description
`whitelist_disabled`	Whitelist is not used
`whitelist_enabled`	Whitelist is to be used

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

16.2.2 enum ls_addr_type

Syntax

```
enum ls_addr_type { ls_addr_type_public = 0,
 ls_addr_type_random = 1,
 ls_addr_type_null = -1 }
```

Description

The type of a typed Bluetooth address.

16.2.3 enum ls_scan_type

Syntax

```
enum ls_scan_type { ls_scan_type_passive = 0,
 ls_scan_type_active = 1,
 ls_scan_type_null = -1}
```

Description

The type of Scan to perform.

16.3 Defines

Default GAP Connection Establishment Parameters

These are the default values configured when the device powers up. The values are typically changed by a GATT Client prior to connecting or (after having discovered a GATT Server's Preferred Connection Parameters) reconnecting.

These values are defined in the Bluetooth Specification (Vol.3 Part C Section 16). However, these values are only "recommended" and it is likely that profiles will define their own preferred parameters.

- #define LS_CON_DEFAULT_MIN_INT 20
- #define LS_CON_DEFAULT_MAX_INT 20
- #define LS_CON_DEFAULT_SLAVE_LATENCY 0
- #define LS_CON_DEFAULT_SUPER_TIMEOUT 200
- #define LS_CON_DEFAULT_SCAN_INTERVAL 4100
- #define LS_CON_DEFAULT_SCAN_WINDOW 4096
- #define LS_CON_DEFAULT_MIN_CE_LENGTH 0
- #define LS_CON_DEFAULT_MAX_CE_LENGTH 0

Connection Establishment Parameters Valid Ranges

Valid ranges for connection parameters.

It is an error for an application to attempt to set connection establishment parameters that are not within the allowed range. Furthermore it is an error for an application to set a minimum connection interval that is greater than the maximum connection interval, to set a scan window that is longer than the scan interval, or to set a minimum CE interval that is greater than the maximum CE interval.

- #define LS_CON_INTERVAL_MIN 6
- #define LS_CON_INTERVAL_MAX 3200
- #define LS_CON_SLAVE_LATENCY_MAX 499
- #define LS_CON_TIMEOUT_MIN 10
- #define LS_CON_TIMEOUT_MAX 3200
- #define LS_CON_SCAN_MIN 0x0004
- #define LS_CON_SCAN_MAX 0x4000

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

General Definitions

- #define LS_MIN_TRANSMIT_POWER_LEVEL 0
- #define LS_MAX_TRANSMIT_POWER_LEVEL 7

16.3.1 LS_CON_DEFAULT_MIN_INT

Definition

```
#define LS_CON_DEFAULT_MIN_INT 20
```

Description

Default Min Connection Interval (25ms)

16.3.2 LS_CON_DEFAULT_MAX_INT

Definition

```
#define LS_CON_DEFAULT_MAX_INT 20
```

Description

Default Max Connection Interval (25ms)

16.3.3 LS_CON_DEFAULT_SLAVE_LATENCY

Definition

```
#define LS_CON_DEFAULT_SLAVE_LATENCY 0
```

Description

Default Slave Latency

16.3.4 LS_CON_DEFAULT_SUPER_TIMEOUT

Definition

```
#define LS_CON_DEFAULT_SUPER_TIMEOUT 200
```

Description

Default Link Supervision Timeout (2s)

16.3.5 LS_CON_DEFAULT_SCAN_INTERVAL

Definition

```
#define LS_CON_DEFAULT_SCAN_INTERVAL 4100
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Default Scan Interval

16.3.6 LS_CON_DEFAULT_SCAN_WINDOW

Definition

```
#define LS_CON_DEFAULT_SCAN_WINDOW 4096
```

Description

Default Scan Window

16.3.7 LS_CON_DEFAULT_MIN_CE_LENGTH

Definition

```
#define LS_CON_DEFAULT_MIN_CE_LENGTH 0
```

Description

Default Minimum Connection Event Length

16.3.8 LS_CON_DEFAULT_MAX_CE_LENGTH

Definition

```
#define LS_CON_DEFAULT_MAX_CE_LENGTH 0
```

Description

Default Maximum Connection Event Length

16.3.9 LS_CON_INTERVAL_MIN

Definition

```
#define LS_CON_INTERVAL_MIN 6
```

Description

Minimum allowed connection interval (7.5ms)

16.3.10 LS_CON_INTERVAL_MAX

Definition

```
#define LS_CON_INTERVAL_MAX 3200
```

Description

Maximum allowed connection interval (4s)

16.3.11 LS_CON_SLAVE_LATENCY_MAX

Definition

```
#define LS_CON_SLAVE_LATENCY_MAX 499
```

Description

Maximum allowed slave latency

16.3.12 LS_CON_TIMEOUT_MIN

Definition

```
#define LS_CON_TIMEOUT_MIN 10
```

Description

Minimum allowed supervision timeout (100ms)

16.3.13 LS_CON_TIMEOUT_MAX

Definition

```
#define LS_CON_TIMEOUT_MAX 3200
```

Description

Maximum allowed supervision timeout (32s)

16.3.14 LS_CON_SCAN_MIN

Definition

```
#define LS_CON_SCAN_MIN 0x0004
```

Description

Minimum scan window/interval (2.5ms)

16.3.15 LS_CON_SCAN_MAX

Definition

```
#define LS_CON_SCAN_MAX 0x4000
```

Description

Minimum scan window/interval (10.24s)

16.3.16 LS_MIN_TRANSMIT_POWER_LEVEL

Definition

```
#define LS_MIN_TRANSMIT_POWER_LEVEL 0
```

Description

Minimum allowed TX power level (range 0-7)

16.3.17 LS_MAX_TRANSMIT_POWER_LEVEL

Definition

```
#define LS_MAX_TRANSMIT_POWER_LEVEL 7
```

Description

Maximum allowed TX power level (range 0-7)

17 Programmable Input/Output

- Analogue IO
- Digital PIO
- Edge Capture Mode
- Pulse Width Modulation
- PIO Controller
- Quadrature Decoders

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

18 Analogue IO

18.1 Functions

18.1.1 AioDrive

Syntax

```
void AioDrive ( aio_select aio, uint16 level )
```

Description

The AIO is configured as an analogue output and driven to the specified voltage, in mV.

Parameters

Parameter	Description
`aio`	The AIO to set.
`level`	The analogue level to drive the AIO output at, in mV, from very near VDD voltage to 0.

Returns

Nothing.

18.1.2 AioOff

Syntax

```
void AioOff ( aio_select aio)
```

Description

Turns the specified AIO "off" as an output, i.e. stops it driving out.

Parameters

Parameter	Description
`aio`	The AIO to set.

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

18.1.3 AioRead

Syntax

```
uint16 AioRead ( aio_select aio)
```

Description

Read the level of an AIO, in mV. Values read will range from 0 to VDD.

Parameters

Parameter	Description
aio	The AIO to read.

Returns

The voltage in mV.

18.1.4 AioSetDig

Syntax

```
void AioSetDig ( aio_select aio, bool state )
```

Description

Set the specified AIO digital output level. The specified AIO is configured as a digital output.

Parameters

Parameter	Description
aio	The AIO to set.
state	TRUE to set the AIO high, FALSE to set it low.

Returns

Nothing.

18.2 Enumerations

18.2.1 enum aio_select

Syntax

```
enum aio_select
```

Description

AIO parameter number

Enumerations

Enumeration	Description

CSR

19 Digital PIO

19.1 Functions

19.1.1 PioGet

Syntax

```
bool PioGet ( uint16 pio)
```

Description

Allows the user application to read the high/low state of a PIO.

Parameters

Parameter	Description
pio	The index (0-31) of the PIO to be read.

Returns

TRUE if the PIO is currently being driven high either internally (for a PIO that is set as an output) or externally (for a PIO that is set as an input).

19.1.2 PioGetDir

Syntax

```
bool PioGetDir ( uint16 pio)
```

Description

Allows the user application to read the direction (input or output) of a particular PIO.

Parameters

Parameter	Description
pio	The PIO to read the direction of.

Returns

TRUE if the specified PIO is currently an output.

19.1.3 PioGetDirs

Syntax

```
uint32 PioGetDirs ( void )
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Reads the directions (input or output) of the PIOs.

Parameters

Parameter	Description
None	-

Returns

A bit-mask indicating which PIOs are currently set as outputs. A bit set in the bit-mask indicates that the corresponding PIO is an output.

19.1.4 PioGets

Syntax

```
uint32 PioGets ( void )
```

Description

Allows the user application to read the high/low state of the PIOs.

Parameters

Parameter	Description
None	-

Returns

A bit-mask indicating which PIOs are currently being driven high. A bit set in the bit-mask indicates that the corresponding PIO is being driven high either internally (for a PIO that is set as an output) or externally (for a PIO that is set as an input).

19.1.5 PioSet

Syntax

```
void PioSet ( uint16 pio, bool set )
```

Description

Allows the user application to set the state of a particular PIO.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
pio	The PIO to set.
set	TRUE to set the PIO high, FALSE to set it low.

Returns

Nothing.

19.1.6 PioSetAnaMonClk

Syntax

```
void PioSetAnaMonClk ( pio_ana_mon_clk pio_clk_source)
```

Description

Set the internal clock source for any PIO configured with mode pio_mode_ana_mon_clk_pio.

Parameters

Parameter	Description
pio_clk_source	The clock source to route to the PIOs.

Returns

Nothing.

19.1.7 PioSetDir

Syntax

```
void PioSetDir ( uint16 pio, bool output )
```

Description

Allows the user application to set the direction (input or output) of a particular PIO.

Parameters

Parameter	Description
pio	The index (0-31) of the PIO to be updated.
output	If set to TRUE, makes the specified PIO an output.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

Nothing.

19.1.8 PioSetDirs

Syntax

```
void PioSetDirs ( uint32 mask, uint32 outputs )
```

Description

Allows the user application to set the direction (input or output) of a number of PIOs.

Parameters

Parameter	Description
`mask`	A bit-mask of PIOs to set the direction of. If bit <n> of mask is set, PIO <n> will have its direction configured according to bit <n> of outputs. If bit <n> of mask is clear, PIO <n> will be left alone.
`outputs`	Subject to mask, PIO <n> is set as an output if bit <n> of outputs is high, and is set as an input if bit <n> of outputs is low.

Returns

Nothing.

19.1.9 PioSetEventMask

Syntax

```
void PioSetEventMask ( uint32 mask, pio_event_mode mode )
```

Description

Allows the user application to enable the generation of sys_event_pio_changed system events when any of the specified PIOs change input state.

Parameters

Parameter	Description
`mask`	A bit-mask of PIOs to affect.
`mode`	The PIO event mode to set for the specified PIOs.

Returns

Nothing.

CSR

19.1.10 PioSetI2CPullMode

Syntax

```
void PioSetI2CPullMode ( pio_i2c_pull_mode mode)
```

Description

Allows the user application to set the pull mode of PIOs configured for I2C operation.

Parameters

Parameter	Description
mode	The pio_i2c_pull_mode to apply.

Returns

Nothing.

19.1.11 PioSetMode

Syntax

```
void PioSetMode ( uint16 pio, pio_mode mode )
```

Description

Allows the user application to set the operating mode of a particular PIO.

Parameters

Parameter	Description
pio	The PIO to set the mode of.
mode	The pio_mode to set the selected PIO to.

Returns

Nothing.

19.1.12 PioSetModes

Syntax

```
void PioSetModes ( uint32 mask, pio_mode mode )
```

Description

Allows the user application to set the operating mode of a number of PIOs.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Parameters

Parameter	Description
mask	A bit-mask of PIOs to affect. If bit <n> of mask is set, PIO <n> will have its mode changed.
mode	The pio_mode to set the selected PIOs to.

Returns

Nothing.

19.1.13 PioSetPullModes

Syntax

```
void PioSetPullModes ( uint32 mask, pio_pull_mode mode )
```

Description

Allows the user application to set the pull mode of a number of PIOs.

Parameters

Parameter	Description
mask	A bit-mask of PIOs to set. If bit <n> of mask is set, PIO <n> will have its pull mode set.
mode	The pio_pull_mode to set the selected PIOs to.

Returns

Nothing.

19.1.14 PioSets

Syntax

```
void PioSets ( uint32 mask, uint32 data )
```

Description

Allows the user application to set the state of a number of PIOs.

CSR

Parameters

Parameter	Description
`mask`	A bit-mask of PIOs to set. If bit <n> of mask is set, PIO <n> will be set from bit <n> of data. If bit <n> of mask is clear, PIO <n> will be left alone.
`data`	Subject to mask, PIO <n> is high or low corresponding to bit <n> of data.

Returns

Nothing.

19.2 Enumerations

19.2.1 enum pio_ana_mon_clk

Syntax

```
enum pio_ana_mon_clk
```

Description

The clock source to monitor on a PIO.

This type defines the available internal clock sources. One of these sources can be selected for monitoring on a PIO by calling PioSetAnaMonClk(). The PIO used for monitoring the clock can be selected by calling PioSetMode() with mode pio_mode_ana_mon_clk_pio.

Enumerations

Enumeration	Description
`pio_ana_mon_clk_32k`	Select 32kHz clock
`pio_ana_mon_clk_16m`	Select 16MHz clock

19.2.2 enum pio_event_mode

Syntax

```
enum pio_event_mode
```

Description

Event modes controlling when sys_event_pio_changed events are generated.

CSR

Enumerations

Enumeration	Description
pio_event_mode_disable	Generate no events for these PIOs
pio_event_mode_rising	Generate events on a rising edge on these PIOs
pio_event_mode_falling	Generate events on a falling edge on these PIOs
pio_event_mode_both	Generate events on a rising or falling edge on these PIOs

19.2.3 enum pio_i2c_pull_mode

Syntax

```
enum pio_i2c_pull_mode
```

Description

I2C Pad pull modes.
This enumeration behaves as a bitfield for enabling the possible pull modes for dedicated I2C pads. The defined enumerated values represent the valid combinations of the individual bits.
Bit 0 - enable bit. Bit 1 - pull direction (0 = down, 1 = up). Bit 2 - pull strength (0 = weak, 1 = strong). Bit 3 - sticky pull (0 = off, 1 = on).

Enumerations

Enumeration	Description
pio_i2c_pull_mode_no_pulls	No pulling enabled
pio_i2c_pull_mode_weak_pull_down	Use weak pull-down
pio_i2c_pull_mode_weak_pull_up	Use weak pull-up
pio_i2c_pull_mode_strong_pull_down	Use strong pull-down
pio_i2c_pull_mode_strong_pull_up	Use strong pull-up
pio_i2c_pull_mode_weak_sticky	Use weak pull-down with sticky (non-floating) inputs
pio_i2c_pull_mode_strong_sticky	Use strong pull-down with sticky (non-floating) inputs

19.2.4 enum pio_mode

Syntax

```
enum pio_mode
```

Description

The mode of operation of an individual PIO.

The CSR1000 has a highly configurable set of PIO pads. Each pad can be configured either for direct control by the application (pio_mode_user) or assigned to specific hardware blocks within the chip, for example the PWM outputs.

Enumerations

Enumeration	Description
`pio_mode_user`	The PIO is under direct application control via PioSet() and PioGet()
`pio_mode_edge_capture`	Counts the number of edges subject to PioEnableEdgeCapture()
`pio_mode_pwm0`	Control the PIO via PWM0
`pio_mode_pwm1`	Control the PIO via PWM1
`pio_mode_pwm2`	Control the PIO via PWM2
`pio_mode_pwm3`	Control the PIO via PWM3
`pio_mode_quadrature0`	Use PIO for quadrature decoder. Even PIOs = Phase A, Odd PIOs = Phase B. Unavailable on CSR100x devices
`pio_mode_quadrature1`	Unavailable on CSR100x devices
`pio_mode_quadrature2`	Unavailable on CSR100x/CR101x devices
`pio_mode_quadrature3`	Unavailable on CSR100x/CR101x devices
`pio_mode_uart`	Use PIO for UART receive (odd-numbered PIOs) or transmit (even-numbered PIOs). PIOs 0 and 1 are configured for UART during firmware initialisation.
`pio_mode_radio_rx_en`	Use PIO for radio reception debug signals. The CS key debug_radio_rx controls which PIO is set to this mode during firmware initialisation.
`pio_mode_radio_tx_en`	Use PIO for radio transmission debug signals. The CS key debug_radio_tx controls which PIO is set to this mode during firmware initialisation.
`pio_mode_nvm_power_en`	Use PIO to control power to non-volatile memory devices. PIO 2 is configured for this by default.
`pio_mode_pio_controller`	Control the PIO via the 8051 PIO Controller unit
`pio_mode_pio_control_txd`	Control the PIO from the 8051 PIO Controller UART TX

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Enumeration	Description
pio_mode_pio_control_rxd	Control the PIO from the 8051 PIO Controller UART RX
pio_mode_ser_flash_dout	Assign SPI Flash MOSI to this PIO
pio_mode_ser_flash_csb	Assign SPI Flash Chip Select to this PIO
pio_mode_i2c_data	Assign I2C Serial Data / SPI Flash MISO to this PIO
pio_mode_i2c_clock	Assign I2C Serial Clock / SPI Flash Clock to this PIO
pio_mode_ana_mon_clk_pio	Monitor one of the internal clocks on a PIO. PioSetAnaMonClk() can be used to select which clock will be output to the PIO

19.2.5 enum pio_pull_mode

Syntax

```
enum pio_pull_mode
```

Description

PIO pad pull modes.
This enumeration behaves as a bitfield for enabling the possible pull modes for general PIO pads. The defined enumerated values represent the valid combinations of the individual bits.
Bit 0 - enable bit. Bit 1 - pull direction (0 = down, 1 = up). Bit 2 - pull strength (0 = weak, 1 = strong). Bit 3 - sticky pull (0 = off, 1 = on).

Enumerations

Enumeration	Description
pio_mode_no_pulls	No pulling enabled
pio_mode_weak_pull_down	Use weak pull-down
pio_mode_weak_pull_up	Use weak pull-up
pio_mode_strong_pull_down	Use strong pull-down
pio_mode_strong_pull_up	Use strong pull-up
pio_mode_weak_sticky	Use weak pull-down with sticky (non-floating) inputs
pio_mode_strong_sticky	Use strong pull-down with sticky (non-floating) inputs

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

20 Edge Capture Mode

20.1 Functions

20.1.1 PioEnableEdgeCapture

Syntax

```
void PioEnableEdgeCapture ( bool enable, bool rising )
```

Description

Allows the user application to enable or disable edge capture mode for all PIOs.

Parameters

Parameter	Description
`enable`	TRUE to enable, FALSE to disable
`rising`	TRUE to capture rising edges, FALSE for falling edges

Returns

Nothing.

20.1.2 PioReadEdgeCapture

Syntax

```
uint32 PioReadEdgeCapture ( void )
```

Description

Allows the user application to take a reading from the edge capture.

Parameters

Parameter	Description
`None`	-

Returns

A 24-bit reading of the number of edges detected.

21 Pulse Width Modulation

21.1 Functions

21.1.1 PioConfigPWM

Syntax

```
 bool PioConfigPWM ( uint16 pwm_id, pio_pwm_mode mode, uint8
dull_on_time, uint8 dull_off_time, uint8 dull_hold_time, uint8 bright_on_time,
uint8 bright_off_time, uint8 bright_hold_time, uint8 ramp_rate )
```

Description

Allows the user application to configure the PIO's Pulse Width Modulation driver.
The parameter descriptions are written in terms of LEDs and brightness since that will be the most common use case. However there is no reason why PIOs controlled by a PWM unit can't drive other devices than LEDs.

Parameters

Parameter	Description
`pwm_id`	the index (0-3) of the PWM unit to be configured.
`mode`	the operating mode (pio_pwm_mode) of the PIO pins used.
`dull_off_time`	the amount of time, in units of ~30us, for which the LED should be off during the dullest part of the flash sequence.
`dull_on_time`	the amount of time, in units of ~30us, for which the LED should be on during the dullest part of the flash sequence.
`dull_hold_time`	the amount of time, in units of ~16ms, for which the LED should be held in the dullest part of the flash sequence.
`bright_off_time`	the amount of time, in units of ~30us, for which the LED should be off during the brightest part of the flash sequence.
`bright_on_time`	the amount of time, in units of ~30us, for which the LED should be on during the brightest part of the flash sequence.
`bright_hold_time`	the amount of time, in units of ~16ms, for which the LED should be held in the brightest part of the flash sequence.
`ramp_rate`	the ramp rate for ramping between brightness levels, in units of ~30us per step with 0 being instantaneous (no ramp).

Returns

TRUE if request was successful.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

21.1.2 PioEnablePWM

Syntax

```
void PioEnablePWM ( uint16 pwm_id, bool enable )
```

Description

Allows the user application to enable or disable the PIO's Pulse Width Modulation drivers.

Parameters

Parameter	Description
`pwm_id`	the index (0-3) of the PWM unit to be enabled or disabled.
`enable`	TRUE to enable the unit, FALSE to disable it.

Returns

Nothing.

22 PIO Controller

22.1 Functions

22.1.1 PioCtrlrClock

Syntax

```
void PioCtrlrClock ( bool fastest_available)
```

Description

Select the clock source for the PIO Controller.

The PIO Controller normally runs off the 32kHz clock. This function can be used to instead request the use of the fastest available clock, in which case the PIO Controller will run off the 16MHz clock when that clock is running (which is any time the chip is not in Deep Sleep).

Note:

if you need a guaranteed 16MHz clock source, then as well as calling this function you must also call SleepModeChange() and set the sleep mode to either sleep_mode_never or sleep_mode_shallow to ensure that the 16MHz clock is not turned off when the radio is idle.

Parameters

Parameter	Description
`fastest_available`	If TRUE then the firmware will use the fastest available clock source for the PIO Controller. If FALSE, then the PIO Controller will always use the 32kHz clock.

Returns

Nothing.

22.1.2 PioCtrlrInit

Syntax

```
void PioCtrlrInit ( uint16 * code)
```

Description

Initialise the 8051 subsystem.

Loads the program code into the 8051 subsystem memory

Parameters

Parameter	Description
`code`	A word array of the code to be loaded into the 8051's memory, starting with the length of the code in bytes.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

Nothing.

22.1.3 PioCtrlrInterrupt

Syntax

```
void PioCtrlrInterrupt ( void )
```

Description

Generate an interrupt to the 8051 PIO Controller subsystem.
The PIO Controller subsystem must have been started with PioCtrlStart() prior to calling this function.

Parameters

Parameter	Description
None	-

Returns

Nothing.

22.1.4 PioCtrlrStart

Syntax

```
void PioCtrlrStart ( void )
```

Description

Start the 8051 PIO Controller subsystem.
Enables the PIO Controller subsystem to start execution of the 8051 application. The application must have been loaded by calling PioCtrlrInit() first.

Parameters

Parameter	Description
None	-

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

22.1.5 PioCtrlrStop

Syntax

```
void PioCtrlrStop ( void )
```

Description

Stop the 8051 PIO Controller subsystem.
Disables the PIO Controller subsystem to halt execution of the 8051 application.

Parameters

Parameter	Description
None	-

Returns

Nothing.

22.2 Defines

22.2.1 PIO_CONTROLLER_RAM_START

Definition

```
#define      PIO_CONTROLLER_RAM_START    ((uint16*)0xE800)
```

Description

Start address in XAP memory map of the PIO Controller's internal 128 byte data memory. Two bytes are mapped to one word in the XAP memory map.

22.2.2 PIO_CONTROLLER_RAM_SIZE_BYTES

Definition

```
#define      PIO_CONTROLLER_RAM_SIZE_BYTES    0x0080
```

Description

Size of PIO Controller's internal data memory, in *bytes*.

22.2.3 PIO_CONTROLLER_RAM_SIZE_WORDS

Definition

```
#define      PIO_CONTROLLER_RAM_SIZE_WORDS    0x0040
```

Description

Size of PIO Controller's internal data memory, in *words*.

22.2.4 PIO_CONTROLLER_DATA_WORD

Definition

```
#define     PIO_CONTROLLER_DATA_WORD   (PIO_CONTROLLER_RAM_START + 0x0020)
```

Description

Offset into the PIO Controller's internal data memory of the data word pointer passed to the application in a sys_event_pio_ctrlr event.

23 Quadrature Decoders

Quadrature Decoder PIO Functions

Functions for controlling the behaviour of PIOs which have a mode of pio_mode_quadrature.

Note:

> No quadrature decoders are available on the CSR100x devices and therefore these functions may be unavailable/ disabled in libraries for these devices.

Two quadrature decoders are available on CSR101x devices, with ids of 0 and 1.

Note:

> Setting an odd numbered PIO pin with a mode of pio_mode_quadratureN to use it as the Phase B input to a hardware quadrature decoder, causes the next lower pin to actually be used as the decoder's input. For example, after the following sequence:

```
PioSetMode(24, pio_mode_quadrature0);
     PioSetMode(23, pio_mode_quadrature0);
```

Pins 24 and 22 (i.e 23 - 1) will be the actual input pins for quadrature decoder 0. However pin 23 is not available for other purposes.

23.1 Functions

23.1.1 PioEnableQuadratureDecoder

Syntax

```
void PioEnableQuadratureDecoder ( uint16 quad_id, bool enable )
```

Description

Allows the user application to enable or disable a quadrature decoder.

Parameters

Parameter	Description
quad_id	The index (0-1) of the quadrature decoder to be enabled/disabled.
enable	TRUE to enable the selected decoders, FALSE to disable.

Returns

Nothing.

23.1.2 PioEnableQuadratureDecoders

Syntax

```
void PioEnableQuadratureDecoders ( uint16 id_mask, uint16
enables )
```

Description

Allows the user application to enable or disable the quadrature decoders.

Parameters

Parameter	Description
`id_mask`	Bit mask indicating which of the available decoders to enable or disable. Bit 0 corresponds to quadrature decoder 0, bit 1 to quadrature decoder 1, etc. Only decoders with their corresponding bit set will be affected; the remaining decoders will be left in their present enabled/disabled state.
`enables`	Subject to id_mask, quadrature decoder <n> is enabled (1) or disabled (0) corresponding to bit <n> of data.

Returns

Nothing.

23.1.3 PioReadQuadratureDecoder

Syntax

```
uint16 PioReadQuadratureDecoder ( uint16 quad_id)
```

Description

Allows the user application to take a reading from the given decoder.

Parameters

Parameter	Description
`quad_id`	The ID (0/1) of the quadrature decoder to read from.

Returns

A 16-bit counter reading.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

24 Serial Interfaces

- I2C Serial Interface
- SPI Serial Interface
- UART

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

25 I2C Serial Interface

25.1 Functions

25.1.1 I2cConfigClock

Syntax

```
void I2cConfigClock ( uint8 scl_high, uint8 scl_low )
```

Description

Set the high and low periods of the I2C clock.

The periods are given in 16th of a microsecond. When using a standard 100kHz or 400kHz period, the constants I2C_SCL_100KBPS_HIGH_PERIOD, I2C_SCL_100KBPS_LOW_PERIOD, I2C_SCL_400KBPS_HIGH_PERIOD, and I2C_SCL_400KBPS_LOW_PERIOD can be used to supply the correct values.

Parameters

Parameter	Description
scl_high	High period of I2C clock
scl_low	Low period of I2C clock

Returns

Nothing.

25.1.2 I2cEepromRead

Syntax

```
sys_status I2cEepromRead ( uint16 device, uint16 address, bool
wait, uint16 length, uint16 * data )
```

Description

Read bytes from a standard I2C EEPROM.

The data read from the device is stored as packed data in the buffer pointed to by data. Despite storing packed data, the length must be the number of bytes to read. Therefore, if the length is odd the last word of the buffer will have an undefined value in the Most-Significant Byte (MSB).

The data is read from the EEPROM within one I2C transaction. Therefore the EEPROM must support Sequential Reads. It is also assumed that the EEPROM does not have any page size restrictions on the Sequential Read (i.e. if requested it can support reading the entire memory contents at once). If sequential reads are not supported, the application can use this function to read one byte at a time.

Non-blocking reads are supported. If the wait parameter is set to FALSE the function will return as soon as the I2C hardware has been configured to start receiving. The application can then use I2cReady() to check if the read operation has completed, although it must also always call I2cEepromReadComplete() to properly terminate the procedure within the driver.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
device	I2C device address
address	Address in EEPROM memory to start reading the data from
wait	Wait for read to complete if TRUE
length	The number of bytes to read
data	Pointer to storage for the data that is read

Returns

Status of operation

25.1.3 I2cEepromReadComplete

Syntax

```
sys_status I2cEepromReadComplete ( void )
```

Description

Finish writing bytes to the I2C bus.
This function must be called if the application previously started an I2C byte write using I2cEepromRead() but didn't wait for it to finish. An error will be returned if no read has been started. If the read operation has already completed then the function will return immediately, otherwise it will wait until the operation has completed and then return.

Parameters

Parameter	Description
None	-

Returns

Status of operation

25.1.4 I2cEepromSetWriteCycleTime

Syntax

```
void I2cEepromSetWriteCycleTime ( uint16 cycle_time)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Set the EEPROM write cycle time (in microseconds).
The write cycle time is the amount of time required by the I2C EEPROM after a STOP condition to complete the write activity. The default time is 5ms.

Parameters

Parameter	Description
cycle_time	-

Returns

Nothing.

25.1.5 I2cEepromSetWritePageSize

Syntax

```
void I2cEepromSetWritePageSize ( uint16 page_size)
```

Description

Set the EEPROM page size for write operations.
The default page size is 128 bytes. An application only needs to call this function if it wishes to use an alternate page size. A page size of zero is not allowed / ignored.

Parameters

Parameter	Description
page_size	-

Returns

Nothing.

25.1.6 I2cEepromWrite

Syntax

```
sys_status I2cEepromWrite ( uint16 device, uint16 address, bool
wait, uint16 length, const uint16 * data )
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

Write bytes to a standard I2C EEPROM.

This function performs a complete I2C transaction, from START condition, sending the device address, writing the address in memory, writing the data bytes and generating a STOP condition.

The data to be written to the device should be passed to the function as packed data in the buffer pointed to by data. Despite being packed, the length must be the number of bytes to write. If the length is odd the last byte will be taken from the Least-Significant Byte (LSB) of the last word of the buffer.

The EEPROM Page Size parameter (see I2cEepromSetWritePageSize) defines the maximum number of bytes that will be written to an EEPROM within one I2C transaction (START condition to STOP condition). After each transaction the driver will wait for Write Cycle Time (see I2cEepromSetWriteCycleTime) to elapse before it starts the next transaction or completes.

If the start address for the write does not lie on a page boundary or the data to be written crosses page boundaries, the driver will also ensure that write is broken down into multiple writes, with one write per page. For example, with a Page Size of 128, a write starting at address 100 for 40 bytes would result in two I2C transactions: the first writing 28 bytes from 100 - 127, and the second writing 12 bytes from 128 to 139.

Warning:The EEPROM Write procedure does not currently support non-blocking operation. However to ensure forward compatibility applications are recommended to always set the wait parameter to TRUE. This will mean that if a future release supports non-blocking writes that existing applications will continue to work as expected.

Parameters

Parameter	Description
`device`	I2C device address
`address`	Address in EEPROM memory to start writing the data to
`wait`	[Not currently used - application must set to TRUE]
`length`	The number of bytes to write
`data`	Pointer to the data to be written

Returns

Status of operation

25.1.7 I2cEnable

Syntax

```
void I2cEnable ( bool enable)
```

Description

Enable or disable the I2C controller.

Parameters

Parameter	Description
`enable`	TRUE to enable the controller or FALSE to disable it

CSR µEnergy Firmware Library Documentation

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

Nothing.

25.1.8 I2cInit

Syntax

```
void I2cInit ( uint8 sda_pio, uint8 scl_pio, uint8 power_pio,
pio_pull_mode pull )
```

Description

Initialise the I2C library.

Configure the PIOs required for I2C bus communication, with the CSR1000 operating as I2C Bus Master. The I2C bus can be assigned to any of the 32 general purpose PIO pins (PIO[31:0] or to the reserved I2C bus pins by specifying a PIO of I2C_RESERVED_PIO for the sda_pio and scl_pio parameters. If the application selects PIO[31:16] (only available on a CSR1001 chip) while running on a CSR1000 chip no error will be returned, but of course the bus will not be externally available.

The 'power_pio' parameter is used to assign an optional PIO to manage the power rail to the I2C device. If the I2C device is permanently powered, or if power is managed directly by the application, then this PIO value should be set to I2C_POWER_PIO_UNDEFINED to disable it.

The 'pull' parameter sets the default pulling mode for the I2C pins. The application can change the pulling mode at any time after calling I2cInit by calling PioSetPullModes() for general PIOs or PioSetI2CPullMode() for the reserved I2C pins. If the I2C bus is initialised to use general PIOs then the reserved I2C pins will have their pulling mode set to pio_i2c_pull_mode_no_pulls.

After calling this function all subsequent I2C operations will use the selected PIOs for communicating with the I2C peripheral.

Calling this function will reset the I2C clock configuration parameters so that devices are clocked at the standard rate of 100kHz.

Parameters

Parameter	Description
`sda_pio`	PIO (0-31, 0xFF) to use for I2C Serial Data.
`scl_pio`	PIO (0-31, 0xFF) to use for I2C Serial Clock.
`power_pio`	PIO (0-31, 0xFF) to use for I2C power control
`pull`	The default PIO pull mode to use for the I2C bus.

Returns

Nothing.

25.1.9 I2cRawCommand

Syntax

```
sys_status I2cRawCommand ( i2c_command cmd, bool wait, uint16
timeout )
```

Description

Send a raw I2C command to I2C controller.

This function is used to generate single I2C events on the bus. This allows the application to communicate with I2C devices other than standard EEPROMs.

The device driver can optionally either wait for the command to complete, or can return immediately, with the command pending. If the command is left pending then the application should call I2cRawComplete() later on to complete and return the status of the command.

There are also a number of helper macros defined to make it simpler to generate the various I2C commands, using default timeout periods. These macros may be more obvious to use than directly calling I2cRawCommand.

Warning:If the application requests a i2c_cmd_wait_ack command but no ACK is received within the timeout period then the I2C transaction will have completed. The application must either then terminate the raw command sequence with I2cRawTerminate() or start a new sequence with I2cRawStart().

Parameters

Parameter	Description
`cmd`	The command to send
`wait`	TRUE if the function should wait for the transaction to complete
`timeout`	Timeout period (in microseconds) to wait for the transaction to complete

Returns

Status of operation

25.1.10 I2cRawRead

Syntax

```
sys_status I2cRawRead ( uint8 * data, uint16 length )
```

Description

Read data from the I2C bus, ACKing received bytes and NACKing the last byte.

This command assumes the I2C slave device has been put into a state where it is ready to transmit bytes. The function will block until the read operation has completed, and all bytes have been read.

Parameters

Parameter	Description
`data`	Pointer to storage for the bytes that are read.
`length`	The number of bytes to read.

Returns

Status of operation

25.1.11 I2cRawReadByte

Syntax

```
sys_status I2cRawReadByte ( uint8 * data)
```

Description

Read a byte of data from the I2C bus.

This command assumes the I2C slave device has been put into a state where it is ready to transmit a byte. The function will block until the read operation has completed.

Warning:This function does NOT generate the I2C ACK/NACK condition after reading the byte. If the application needs to read one or more bytes and generate standard ACK/NACK conditions after each byte then I2cRawRead() may be more suitable.

Parameters

Parameter	Description
`data`	Pointer to storage for the byte that is read.

Returns

Status of operation

25.1.12 I2cRawTerminate

Syntax

```
sys_status I2cRawTerminate ( void )
```

Description

Ends the sequence of I2C raw commands.

If an application uses both raw commands and the atomic read/write functions to access an I2C device it must properly terminate the raw command sequence by calling this function before it can use the byte read/write functions again. This is to ensure that the device is in a known state for the read/write functions.

This function only updates internal driver state - it does not generate any further transactions on the I2C bus. Therefore the application must ensure that it properly completes a raw command sequence with an I2C STOP condition before calling this function.

Parameters

Parameter	Description
`None`	-

Returns

Status of operation

25.1.13 I2cRawWrite

Syntax

```
sys_status I2cRawWrite ( const uint8 * data, uint16 length )
```

Description

Write data to the I2C bus, waiting for ACK after each byte.
This command assumes the I2C slave device has been put into a state where it is ready to receive bytes. The function will block until the write operation has completed, and all bytes have been transmitted.

Parameters

Parameter	Description
data	Pointer to the bytes that are to be written.
length	The number of bytes to write.

Returns

Status of operation

25.1.14 I2cRawWriteByte

Syntax

```
sys_status I2cRawWriteByte ( uint8 data)
```

Description

Write a byte of data to the I2C bus.
This command assumes the I2C slave device has been put into a state where it is ready to receive a byte. The function will block until the write operation has completed.

Parameters

Parameter	Description
data	The byte that is to be written.

Returns

Status of operation

25.1.15 I2cReady

Syntax

```
bool I2cReady ( void )
```

Description

Test to see if the current I2C transaction has completed.

Parameters

Parameter	Description
None	-

Returns

TRUE if I2C transaction has completed or FALSE if it is ongoing.

25.1.16 I2cReset

Syntax

```
sys_status I2cReset ( void )
```

Description

Reset the I2C controller.
Resets the I2C controller without waiting for any current read or write commands to finish. This is advised only as part of the initialisation procedure, or to recover from an incorrect state in the I2C controller.

Parameters

Parameter	Description
None	-

Returns

Status of operation

25.2 Enumerations

25.2.1 enum i2c_command

Syntax

```
enum i2c_command
```

Description

Raw I2C commands.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumerations

Enumeration	Description
`i2c_cmd_send_start`	Send START condition
`i2c_cmd_send_restart`	Send RESTART condition
`i2c_cmd_send_stop`	Send STOP condition
`i2c_cmd_wait_ack`	Wait for an ACK
`i2c_cmd_send_ack`	Send an ACK
`i2c_cmd_send_nack`	Send a NACK
`i2c_cmd_tx_data`	Internal use only
`i2c_cmd_rx_data`	Internal use only

25.3 Defines

25.3.1 I2C_EEPROM_POLLED_WRITE_CYCLE

Definition

```
#define I2C_EEPROM_POLLED_WRITE_CYCLE 0
```

Description

This constant can be used when setting the EEPROM Write Cycle Time with I2cEepromSetWriteCycleTime() to indicate that the I2C driver should poll the EEPROM for write completion instead of waiting a fixed period.

The EEPROM Write Cycle Time is the period of time to wait at the end of an EEPROM write transaction to allow the write to complete within the device. Some EEPROMs support a mode where they will not ACKnowledge any further activity during this internal write cycle, which allows the EEPROM driver to poll them and return immediately after the write has completed rather than waiting a fixed length of time (which, depending on the device, may be quite pessimistic in normal operating conditions).

Warning:To avoid a software lock-up due to an unresponsive EEPROM, the driver will abort polling and return if the EEPROM has not generated an ACK within 64ms. If this happens status i2c_status_fail_write_poll_timeout will be returned.

25.3.2 I2C_POWER_PIO_UNDEFINED

Definition

```
#define I2C_POWER_PIO_UNDEFINED 0xFF
```

Description

PIO value indicating that the I2C device driver should not manage the power for the I2C device(s).

This constant can be used when calling I2cInit().

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

25.3.3 I2C_RESERVED_PIO

Definition

```
#define I2C_RESERVED_PIO 0xFF
```

Description

PIO selection to use dedicated I2C bus pins instead of general purpose PIOs.
This constant can be used when calling I2cInit() if the application wants to use the reserved I2C pins for the I2C clock or data signals.

25.3.4 I2C_SCL_100KBPS_HIGH_PERIOD

Definition

```
#define I2C_SCL_100KBPS_HIGH_PERIOD 78
```

Description

SCL high period for 100kHz clock

25.3.5 I2C_SCL_100KBPS_LOW_PERIOD

Definition

```
#define I2C_SCL_100KBPS_LOW_PERIOD 78
```

Description

SCL low period for 100kHz clock

25.3.6 I2C_SCL_400KBPS_HIGH_PERIOD

Definition

```
#define I2C_SCL_400KBPS_HIGH_PERIOD 15
```

Description

SCL high period for 400kHz clock

25.3.7 I2C_SCL_400KBPS_LOW_PERIOD

Definition

```
#define I2C_SCL_400KBPS_LOW_PERIOD 21
```

Description

SCL low period for 400kHz clock

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

25.3.8 I2cRawRestart

Definition

```
#define I2cRawRestart ( wait ) I2cRawCommand(i2c_cmd_send_restart,
wait, I2C_WAIT_CMD_TIMEOUT)
```

Description

Send a RESTART condition

25.3.9 I2cRawSendAck

Definition

```
#define I2cRawSendAck ( wait ) I2cRawCommand(i2c_cmd_send_ack, wait,
I2C_WAIT_CMD_TIMEOUT)
```

Description

Send an ACK condition

25.3.10 I2cRawSendNack

Definition

```
#define I2cRawSendNack ( wait ) I2cRawCommand(i2c_cmd_send_nack,
wait, I2C_WAIT_CMD_TIMEOUT)
```

Description

Send a NACK condition

25.3.11 I2cRawStart

Definition

```
#define I2cRawStart ( wait ) I2cRawCommand(i2c_cmd_send_start, wait,
I2C_WAIT_CMD_TIMEOUT)
```

Description

Send a START condition

25.3.12 I2cRawStop

Definition

```
#define I2cRawStop ( wait ) I2cRawCommand(i2c_cmd_send_stop, wait,
I2C_WAIT_CMD_TIMEOUT)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Send a STOP condition

25.3.13 I2cRawWaitAck

Definition

```
 #define I2cRawWaitAck ( wait ) I2cRawCommand(i2c_cmd_wait_ack, wait,
I2C_WAIT_ACK_TIMEOUT)
```

Description

Wait for an ACK

26 SPI Serial Interface

26.1 Functions

26.1.1 SpiConfigReadRegisterDelay

Syntax

```
void SpiConfigReadRegisterDelay ( uint16 delay)
```

Description

Set the Read Register Delay configuration parameter.

The Read Register Delay is the minimum period in microseconds that the SPI driver will wait between the rising edge of SCLK for the last bit of the register address byte to the falling edge of SCLK for the first bit of data read. The driver has internal delays that account for about 3.5us, so if a SPI slave needs more time than that to prepare a response, the app should set this delay to a non-zero value.

This value is only applied between writing the register address and reading the value(s) (including burst register reads). Burst Reads do no insert any additional delays between reading of individual register values.

The delay parameter is reset to 0 each time SpiInit() is called.

Parameters

Parameter	Description
`delay`	Extra delay (in microseconds).

Returns

Nothing.

26.1.2 SpiConfigWriteIntervalDelay

Syntax

```
void SpiConfigWriteIntervalDelay ( uint16 delay)
```

Description

Set the Write Interval Delay configuration parameter.

The Write Interval Delay is the minimum period in microseconds between subsequent byte transfers under SpiWrite. Some devices may require a long period to allow the data to be processed before the next byte.

With no delays specified the SPI driver code has an approximate delay of 5us due to internal processing, although this is not guaranteed.

Parameters

Parameter	Description
`delay`	Extra delay (in microseconds).

CSR µEnergy Firmware Library Documentation

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

Nothing.

26.1.3 SpiConfigWriteTerminationDelay

Syntax

```
void SpiConfigWriteTerminationDelay ( uint16 delay)
```

Description

Set the Write Termination Delay configuration parameter.

The Write Termination Delay is the minimum period in microseconds that the chip select line will be held active after a write completes. Some devices may require a long period to allow the data to be stored before CS goes inactive.

With no delays specified the SPI driver code has an approximate delay of 8us due to internal processing, although this is not guaranteed.

Parameters

Parameter	Description
`delay`	Extra delay (in microseconds).

Returns

Nothing.

26.1.4 SpiFlashEraseBlock

Syntax

```
sys_status SpiFlashEraseBlock ( spi_erase_size size, uint16
address, bool wait )
```

Description

Erase a single block of the SPI Flash memory device.

The block is determined by the address passed in. Typically this will be the first address of the block, although devices may allow any address within the block to be used. The size of the block to be erased is determined by the size parameter. The driver currently supports erasing 4KB or 32KB blocks.

The caller can specify whether or not the driver waits for the erase operation to finish before returning. With typical block erase times of up to a second, this allows the application to continue doing other processing while the erase operation completes. If the driver does not wait, then the caller MUST ensure that the erase has finished by later calling SpiFlashEraseComplete() before performing any other operations with the SPi Flash driver.

Parameters

Parameter	Description
`size`	Size of block to erase
`address`	Address corresponding to block to be erased
`wait`	TRUE if the driver should wait for the erase to finish

Returns

Status of operation

26.1.5 SpiFlashEraseWaitComplete

Syntax

```
sys_status SpiFlashEraseWaitComplete ( void )
```

Description

Waits for an erase operation to complete.

Parameters

Parameter	Description
`None`	-

Returns

Status of operation

26.1.6 SpiFlashInit

Syntax

```
void SpiFlashInit ( uint16 ncs_pio, uint16 power_pio )
```

Description

Initialise the SPI Flash library.

Configure the NVM hardware for communication with a SPI Flash device. The SPI Flash must be on the same SPI bus as the boot flash (it is therefore not possible to use this interface to communicate with a SPI Flash device if the CSR1000 was booted from an I2C EEPROM).

The hardware has a fixed 8MHz SPI clock for all transactions.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
`ncs_pio`	PIO (0-15) to use for SPI Flash Chip Select (active low).
`power_pio`	PIO (0-15) to use for SPI Flash power.

Returns

Nothing.

26.1.7 SpiFlashRead

Syntax

```
 sys_status SpiFlashRead ( uint16 address, uint16 length, uint16
* data )
```

Description

Read data from the selected SPI Flash device.

The data is read in units of bytes. The application should supply a pointer to a packed array (that is, a uint16*). This is to optimise the amount of RAM required to read data from a SPI Flash device. If an odd number of bytes are requested, the unused byte of the final packed word will be set to 0x00.

Due to fixed internal timeouts, it is recommended that no more than 4KB (4096 bytes) is read in a single transaction.

Parameters

Parameter	Description
`address`	Address in SPI Flash memory to start reading the data from.
`length`	The number of bytes to read.
`data`	Pointer to storage for the data that is read (packed array)

Returns

Status of operation

26.1.8 SpiFlashWrite

Syntax

```
 sys_status SpiFlashWrite ( uint16 address, uint16 length, const
uint16 * data )
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Write data to the selected SPI Flash device.

The data is written in units of bytes. The application should supply a pointer to a packed array (that is, a uint16*). This is to optimise the amount of RAM required to write data to a SPI Flash device.

The caller must ensure that no more than one full page (often 256 bytes, although dependent on selected device parameters) is written in a single transaction.

Parameters

Parameter	Description
`address`	Address in SPI Flash memory to start writing the data to
`length`	The number of bytes to write
`data`	Pointer to the data to be written (packed array)

Returns

Status of operation

26.1.9 SpiInit

Syntax

```
 bool SpiInit ( uint16 mosi_pio, uint16 miso_pio, uint16 clk_pio,
uint16 ncs_pio )
```

Description

Initialise the SPI library.

Configure the PIOs required for SPI bus communication, with the CSR1xxx operating as SPI Master, using SPI Mode 3. All read and write operations assume 8-bit values. The Most Significant Bit (MSB) will be clocked out (for writes) or clocked in (for reads) first. The protocol is assumed to be half-duplex, i.e. when writing bytes, read data is ignored, and vice-versa.

SPI bus transactions are implemented using a software driver to control the selected PIO lines. Therefore the fundamental SPI clock rate is restricted by the speed of the processor. In practise we have found the clock rate to be approximately 470kHz. The clock rate is fixed.

After calling this function all subsequent Spi*() operations will use the selected PIOs for communicating with the SPI peripheral. If an application wishes to communicate with two independent SPI devices on the same bus it must call SpiInit() each time it wishes to switch to the other SPI device. Typically an implementation would have SPI Clock and SPI Data signals assigned to common PIOs while the SPI Chip Select signal for each SPI slave device would have a dedicated PIO. However this arrangement is not mandatory.

Calling this function will clear the delay configuration parameters (Read Register Delay and Write Termination Delay).

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
mosi_pio	PIO (0-31) to use for SPI Data Master Out/Slave In.
miso_pio	PIO (0-31) to use for SPI Data Master In/Slave Out.
clk_pio	PIO (0-31) to use for SPI Clock.
ncs_pio	PIO (0-31) to use for SPI Slave Chip Select (active low), or SPI_NCS_PIO_UNDEFINED to control chip select from the application.

Returns

TRUE if bus is initialised or FALSE if there was an error

26.1.10 SpiRead

Syntax

```
uint16 SpiRead ( uint8 * in_buffer, uint16 length )
```

Description

Read a sequence of bytes of data from a SPI peripheral device and copy the data into the supplied buffer.
The SPI slave device is selected, the array of bytes is clocked in, and then the device is de-selected.

Parameters

Parameter	Description
in_buffer	Array of bytes to store data read from the SPI device.
length	Number of bytes to read.

Returns

The number of bytes read (which will always be 'length').

26.1.11 SpiReadByte

Syntax

```
uint8 SpiReadByte ( void )
```

Description

Read one byte of data from a SPI peripheral device.
The SPI slave device is selected, the byte is clocked in, and then the device is de-selected.

Parameters

Parameter	Description
None	-

Returns

The byte value read from the SPI device.

26.1.12 SpiReadRegister

Syntax

```
uint8 SpiReadRegister ( uint8 reg_address)
```

Description

Read from a register on a SPI peripheral device.

The SPI slave device is selected, the register address is clocked out, then the register value is clocked in, and then the device is de-selected.

There is a delay of about 3.5us between writing the register address and reading the values. Some of this delay is fixed due to the internal processing carried out by the SPI device driver. However, as some devices may require longer delays to prepare the burst response, the application can use the function SpiConfigReadRegisterDelay() to set an additional delay. During this delay the SPI slave device remains selected.

Parameters

Parameter	Description
reg_address	The address of the register to be written.

Returns

The register value read from the SPI device.

26.1.13 SpiReadRegisterBurst

Syntax

```
uint16 SpiReadRegisterBurst ( uint8 reg_address, uint8 *
in_buffer, uint16 length, bool toggle_clk )
```

CSR µEnergy Firmware Library Documentation

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

Burst read multiple registers on a SPI peripheral device.

The SPI slave device is selected, the register address is clocked out, multiple register values are clocked in, and then the device is de-selected.

This procedure assumes that the SPI slave device will prepare the Burst Read response, such that from the first register address it will return a device-specific series of register values within one SPI transaction, with no further address writes required between values.

There is a delay of about 3.5us between writing the register address and reading the values. Some of this delay is fixed due to the internal processing carried out by the SPI device driver. However, as some devices may require longer delays to prepare the burst response, the application can use the function SpiConfigReadRegisterDelay() to set an additional delay. During this delay the SPI slave device remains selected.

The parameter 'toggle_clk' should normally be left set to FALSE. It is provided for compatibility with certain SPI devices. It adds an extra clock cycle between writing the register address and reading the first register value.

Parameters

Parameter	Description
`reg_address`	The first register address.
`in_buffer`	Array of bytes to store register values read from the SPI device.
`length`	Number of bytes to read.
`toggle_clk`	Configuration parameter to toggle clock after write

Returns

The number of bytes read (which will always be 'length').

26.1.14 SpiWrite

Syntax

```
uint16 SpiWrite ( const uint8 * out_buffer, uint16 length )
```

Description

Write a sequence of bytes of data to a SPI peripheral device.

The SPI slave device is selected, the array of bytes is clocked out, and then the device is de-selected.

Parameters

Parameter	Description
`out_buffer`	Array of bytes to write to the SPI device.
`length`	Number of bytes in the array to write.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

The number of bytes written (which will always be 'length').

26.1.15 SpiWriteByte

Syntax

```
void SpiWriteByte ( uint8 byte)
```

Description

Write one byte of data to a SPI peripheral device.
The SPI slave device is selected, the byte is clocked out, and then the device is de-selected.

Parameters

Parameter	Description
`byte`	The byte to write.

Returns

Nothing.

26.1.16 SpiWriteRegister

Syntax

```
void SpiWriteRegister ( uint8 reg_address, uint8 reg_value )
```

Description

Write to a register on a SPI peripheral device.
The SPI slave device is selected, the register address is clocked out, the register value is clocked out, and then the device is de-selected.

Parameters

Parameter	Description
`reg_address`	The address of the register to be written.
`reg_value`	The value to write to the register.

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

26.2 Enumerations

26.2.1 enum spi_erase_size

Syntax

```
enum spi_erase_size
```

Description

SPI Flash block erase sizes.

Enumerations

Enumeration	Description
`spi_erase_4KB`	Erase a 4KB block
`spi_erase_32KB`	Erase a 32KB block

26.3 Defines

26.3.1 SPI_FLASH_DEFAULT_PIO

Definition

```
#define SPI_FLASH_DEFAULT_PIO 0xFF
```

Description

Default PIO should be used for SPI Flash power or chip select signal.
This constant can be used when calling SpiFlashInit() if the application wants to use the default PIO for the power or chip select signals. For example, a design with two SPI Flash devices may use the same PIO for the boot device and the secondary device (to save a PIO) and use discrete chip select signals for each device.

26.3.2 SPI_FLASH_POWER_PIO_UNDEFINED

Definition

```
#define SPI_FLASH_POWER_PIO_UNDEFINED 0xFF
```

Description

PIO value indicating that the SPI Flash device driver should not manage the power for the SPI Flash device.
This constant can be used when calling SpiFlashInit().

26.3.3 SPI_NCS_PIO_UNDEFINED

Definition

```
#define SPI_NCS_PIO_UNDEFINED 0xff
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

PIO selection to be used if the application wants to control the SPI chip select line(s) instead of the firmware. This constant can be used when calling SpiInit().

27 UART

27.1 Functions

27.1.1 UartConfig

Syntax

```
void UartConfig ( uint16 baud_rate_enum, uint16 config )
```

Description

Configure the UART baud rate and port configuration.

After calling this function the UART will be left disabled. The caller must therefore call UartEnable before transmitting or receiving any data.

If the baud rate is set to UART_RATE_DEFAULT then the baud rate and port configuration will be read from the corresponding Configuration Store keys.

Parameters

Parameter	Description
`baud_rate_enum`	Described by UART baudrate.
`config`	A 16-bit bitfield described by UART configuration.

Returns

Nothing.

27.1.2 UartEnable

Syntax

```
void UartEnable ( bool enable)
```

Description

Enable/disable UART interface hardware.

Enabling the UART interface defaults to waking the CSR1000 when it receives RX data (see SleepWakeOnUartRX).

Parameters

Parameter	Description
`enable`	TRUE to enable, FALSE to disable.

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

27.1.3 UartInit

Syntax

```
 void UartInit ( uart_data_in_fn data_in_clbk, uart_data_out_fn
data_out_clbk, uint16 * rx_buffer, uart_buf_size_bytes rx_size_bytes, uint16 *
tx_buffer, uart_buf_size_bytes tx_size_bytes, uart_data_mode new_data_mode
)
```

Description

Initialise the UART interface.

Sets the function pointers to be called on data in/out events. Defines the TX & RX buffers (start address and size for each buffer). RX & TX buffers are normally declared using the UART_DECLARE_BUFFER macro. If the application wants to use the UART in 'packed' mode (see UART configuration) it must still provide the length of the TX & RX buffers to this function in bytes. The rx_size and tx_size parameters are an enum uart_buf_size_bytes to restrict the possible values that these parameters can accept.

This function also configures the UART with the default settings stored in the CS keys. If the application wishes to change the port configuration it should call UART configuration after calling this function.

The application may provide NULL pointers for the data in and/or data out events, if it it does not care about receiving and/or transmitting data. For example, a simple debug interface that only uses the UART to send debug messages would not need an RX callback at all, and would likely not need a TX callback either.

Warning:The UART interface only supports buffers of size 32, 64, 128, or 256 bytes. If the application tries to create a buffer that is a different size the UART interface will generate a fault.

Parameters

Parameter	Description
`data_in_clbk`	Pointer to a function of type uart_data_in_fn.
`data_out_clbk`	Pointer to a function of type uart_data_out_fn.
`rx_buffer`	Pointer to the RX (read) buffer
`rx_size_bytes`	Size of the RX buffer
`tx_buffer`	Pointer to the TX (write) buffer
`tx_size_bytes`	Size of the TX buffer
`data_mode`	Desired data packing mode for UART (unpacked or packed)

Returns

Nothing.

27.1.4 UartRead

Syntax

```
 bool UartRead ( uint16 length, uint32 timeout )
```

Description

Read the specified amount of data from the UART.

Requests that the UART driver returns the specified amount of received UART data to the application once said amount of data is available. The actual data is made available to the caller via the data_in_clbk function that was registered with UartInit.

The length parameter depends on the current UART data mode ('unpacked' or 'packed'). For packed data, the length is the number of words to read, therefore the UART driver will wait until an even number of bytes have been received over the wire (for example if the application has set 'packed' mode and requests length=4, the driver will wait until 4 words have been received (8 bytes) before calling the data_in_clbk callback function). When the UART data mode is 'unpacked' the length parameter is the number of bytes to receive.

If the application has not provided an RX callback function in UartInit() then requesting a read of an amount of data will result in that data being read from the UART buffer and discarded. However, as the application does not receive any event when this happens it should not be used to "empty" the UART buffer, as there is no way for the application to determine exactly how much data has been discarded.

Warning:The data pointer passed to the application in the data_in_clbk must not be used directly for writing data straight back to the UART. If the application needs to implement a loopback mechanism it must first copy the received data to a local buffer and then use that as the source for the data passed to UartWrite.

Parameters

Parameter	Description
`length`	Amount of data to get (in bytes or words depending on UART data mode).
`timeout`	Not currently used. Must always be set to 0 by the application

Returns

Nothing.

27.1.5 UartTxIsBusy

Syntax

```
bool UartTxIsBusy ( void )
```

Description

Check the UART for TX activity.

Returns TRUE if the UART is transmitting a packet.

Warning:This function is not intended for customer use.

Parameters

Parameter	Description
`None`	-

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

27.1.6 UartWrite

Syntax

```
bool UartWrite ( const void * data, uint16 length )
```

Description

Write a number of data bytes/words to the UART.

Given a pointer to an array, write this data into the UART's transmit buffer and initiate the UART transmit. The length parameter is the number of array "elements" to be written - depending on the UART mode this is either the number of bytes (uart_data_unpacked) or number of words (uart_data_packed).

The size of each element of array is defined by the current data mode of the UART (see UartInit and uart_data_mode). If the data mode is uart_data_unpacked then the data pointer is assumed to point to an array of (unpacked) uint8 data, and the LSB of each element will be copied into the UART transmit buffer. If the data mode is uart_data_packed then the data pointer is assumed to point to (packed) uint16 data and the LSB and MSB of each word will be copied into the UART transmit buffer, LSB first.

This function will return without writing any data if the internal transmit buffer does not have enough space to store all of the data. To ensure that the data is written on the first attempt (blocking write behaviour), the UartWriteBlocking function must be used instead. (The application must not use a while() loop to "poll" the UartWrite() function as doing so will not allow the UART to clean up after existing bytes have been transmitted).

INTERRUPTS: This function is non-reentrant.

Warning:The data pointer passed to the application in the data_in_clbk must not be used directly for writing data straight back to the UART. If the application needs to implement a loopback mechanism it must first copy the received data to a local buffer and then use that as the source for the data passed to UartWrite.

Parameters

Parameter	Description
data	Pointer to the data buffer
length	Length of data

Returns

TRUE on success or FALSE if there was insufficient space in the buffer.

27.1.7 UartWriteBlocking

Syntax

```
void UartWriteBlocking ( const void * data, uint16 length )
```

CSR

Description

Write a number of data bytes/words to the UART, blocking until all bytes have been copied to the UART transmit buffer.

This function behaves very much like UartWrite, with the exception that it will not return until all the data has been copied to the UART transmit buffer. Therefore there is no return value (it will always succeed).

Note:

If the buffer is full when called, then the time taken to return will depend on how much data needs to be emptied before the all of the new data can be stored in the UART buffer, and the baud rate of the UART (which controls how quickly the hardware can transmit the contents of the transmit buffer over the wire).

INTERRUPTS: This function is non-reentrant.

Warning:This function will return when all supplied bytes have been buffered by the UART. This is not the same as having actually transmitted all bytes over the wire.

Parameters

Parameter	Description
`data`	Pointer to the data buffer
`length`	Length of data

Returns

Nothing

27.2 Enumerations

27.2.1 enum uart_buf_size_bytes

Syntax

```
enum uart_buf_size_bytes
```

Description

UART buffer size, in bytes.

These constants represent the possible size of UART buffers that be created. No other buffer sizes are supported by the UART interface.

Enumerations

Enumeration	Description
`UART_BUF_SIZE_BYTES_32`	32 byte buffer
`UART_BUF_SIZE_BYTES_64`	64 byte buffer
`UART_BUF_SIZE_BYTES_128`	128 byte buffer
`UART_BUF_SIZE_BYTES_256`	256 byte buffer

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

27.2.2 enum uart_data_mode

Syntax

```
enum uart_data_mode
```

Description

UART data mode.

The UART can be operated in packed or unpacked mode. By default the UART is unpacked, but the application can change this by calling UART configuration. The data mode affects the type of data passed from application to UART via the UartWrite or UartWriteBlocking functions, and the type of data passed back from the UART to application in the UartRead() callback function.

The data mode does not affect the TX & RX buffers passed to UartInit - these buffers are always packed uint16 buffers, with the UART driver unpacking & packing data as required.

Enumerations

Enumeration	Description
`uart_data_packed`	UART data is packed (arrays of uint16)
`uart_data_unpacked`	UART data is unpacked (arrays of uint8)

27.3 TypeDefs

27.3.1 typedef uint16(* uart_data_in_fn)(void *, uint16, uint16 *)

Syntax

```
typedef uint16(* uart_data_in_fn)(void *, uint16, uint16 *)
```

Description

Receive handler function.

A function that will be called whenever length number of bytes (which is set by uart_fetch_data or DebugInit) have been received over the UART.

The function takes three parameters: a pointer to a void buffer of the received data (which will be a uint8* if the UART data mode is 'unpacked' or a uint16* if the UART data mode is 'packed'), a uint16 containing the number of bytes ('unpacked') or words ('packed') received, and a pointer to a uint16 which should be filled in with the number of additional bytes ('unpacked') or words ('packed') the application wishes to receive (set to 0 if no further data is required at this time).

The function must return the number of bytes ('unpacked') or words ('packed') that the application has processed out of the available data (which may be less than was originally provided to the application). If the application does not process all of the data then the remaining data will remain in the buffer until a further amount of data requested by the application has been received.

Note:

> The application does not have to check for wrapping within the UART RX circular buffer. The memory management hardware ensures that the the data pointer supplied to the application presents the received data sequentially.

If the callback function is NULL, it indicates that the application is not interested in receiving data.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

27.3.2 typedef void(* uart_data_out_fn)(void)

Syntax

```
typedef void(* uart_data_out_fn)(void)
```

Description

Transmit handler function.

A function that will be called whenever a UART transmission has finished. The function takes no parameters and must return no value.

May be NULL, which indicates that these events are uninteresting to the application.

27.4 Defines

27.4.1 UART_DECLARE_BUFFER

Definition

```
#define UART_DECLARE_BUFFER ( _name, _size ) static uint16
_name[16<<(_size)] GCC_ATTRIBUTE(aligned (2));
```

Description

Declare RAM buffer for UART RX or UART TX operation.

This macro creates a buffer to be used by the UART interface for transmit or receive operations. The application is required to create a pair of buffers, one for RX and one for TX. The size of the buffers should be defined using one of the constants defined in uart_buf_size_bytes. The RX & TX buffers do not need to be the same size.

Warning:The GCC alignment attribute is required when declaring buffers to ensure that the buffers meets the alignment requirements of the hardware.

Parameter

Variable	Description
`_name`	The name of the buffer
`_size`	The size of the buffer, using one of the constants from uart_buf_size_bytes

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

28 Memory Management

- C Standard Library APIs
- Persistent Memory
- Configuration Store

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

29 C Standard Library APIs

29.1 Functions

29.1.1 CountSetBits32

Syntax

```
uint16 CountSetBits32 ( uint32 value)
```

Description

Counts number of set bits in a 32 bit word.

Parameters

Parameter	Description
`value`	Value to be tested.

Returns

Number of bits equal to 1

29.1.2 CountTransitions32

Syntax

```
uint16 CountTransitions32 ( uint32 value)
```

Description

Counts bit transitions in a 32 bit word.

Parameters

Parameter	Description
`value`	Value to be tested.

Returns

Number of transitions between 0 and 1

29.1.3 IsDigit

Syntax

```
int IsDigit ( int value)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Is the character presented a decimal digit?

Parameters

Parameter	Description
`value`	Value to be tested.

Returns

Non-zero if the value is an ASCII digit.

29.1.4 IsSpace

Syntax

```
int IsSpace ( int value)
```

Description

Is the character presented whitespace?

Parameters

Parameter	Description
`value`	Value to be tested.

Returns

Non-zero if the value is an ASCII whitespace character.

29.1.5 IsUpper

Syntax

```
int IsUpper ( int value)
```

Description

Is the character presented an uppercase roman letter?

Parameters

Parameter	Description
`value`	Value to be tested.

Returns

Non-zero if the value is an ASCII uppercase roman character.

29.1.6 MemChr

Syntax

```
void* MemChr ( const void * buff, int value, uint16 length )
```

Description

Find the first occurrence of 'value' in an array of given size.

Parameters

Parameter	Description
buff	Pointer to the array.
value	Value to be written.
length	Size of the array.

Returns

A pointer to the character as found in memory, or NULL if not found.

29.1.7 MemCmp

Syntax

```
int MemCmp ( const void * buff1, const void * buff2, uint16
length )
```

Description

Compare two arrays of memory.

Parameters

Parameter	Description
buff1	Pointer to the 1st array.
buff2	Pointer to the 2nd array.
length	Size of the array.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Returns

-1 if the first array is "less than" the second in the usual string comparison sense, +1 if the first is "greater than" the second, and 0 if they are equal.

29.1.8 MemCopy

Syntax

```
void* MemCopy ( void * dest, const void * source, uint16 length
)
```

Description

Copy memory, 16bit word(s) to 16 bit word(s)
This is implemented inline by the GCC code generator rather than as a function call. If for some reason the address of MemCopy must be taken then a wrapper around the builtin function must be written:

Parameters

Parameter	Description
`dest`	Pointer to the destination buffer.
`source`	Pointer to the source buffer.
`length`	Size of the destination buffer.

Returns

Original pointer to the destination buffer

29.1.9 MemCopyPack

Syntax

```
void MemCopyPack ( uint16 * dest, const uint8 * source, uint16
length )
```

Description

Copy memory, turning 2 * uint8 into uint16.
Like MemCopy but the source is a uint8 array and the destination is a uint16 array. The first word of the uint16 array is built from the first 2 words of the uint8 array (LSB first). If the uint8's contain set bits in there msb's these are masked away.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
dest	Pointer to the destination buffer.
source	Pointer to the source buffer.
length	Size of the source buffer.

Returns

Nothing

29.1.10 MemCopyUnPack

Syntax

```
void MemCopyUnPack ( uint8 * dest, const uint16 * source, uint16
length )
```

Description

Copy memory, turning uint16 into 2 * uint8.
Like MemCopy but the source is a uint16 array and the destination is a uint8 array. The first word of the uint16 array is split into the first 2 words of the uint8 array (LSB first).

Parameters

Parameter	Description
dest	Pointer to the destination buffer.
source	Pointer to the source buffer.
length	Size of the destination buffer.

Returns

Nothing

29.1.11 MemSet

Syntax

```
void* MemSet ( void * dest, uint16 value, uint16 length )
```

Description

Fill memory with a specified 16 bit word.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
dest	Pointer to the destination buffer.
value	Value to be written.
length	Size of the destination buffer.

Returns

Original pointer to the destination buffer

29.1.12 StrChr

Syntax

```
char* StrChr ( const char * string, int value )
```

Description

Locate a value in a string.

Parameters

Parameter	Description
string	Pointer to the character array.
value	Search character.

Returns

A pointer to the character located in the string, or NULL if not found.

29.1.13 StrLen

Syntax

```
uint16 StrLen ( const char * string)
```

Description

Get the length of a string.

Parameters

Parameter	Description
string	Pointer to the character array.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

The number of characters in the string.

29.1.14 StrNCopy

Syntax

```
char* StrNCopy ( char * dest, const char * source, uint16 length
)
```

Description

Copy a limited number of characters between strings.

Parameters

Parameter	Description
`dest`	Pointer to the destination buffer.
`source`	Pointer to the source buffer.
`length`	Size of the destination buffer.

Returns

Original pointer to the destination buffer

29.1.15 ToLower

Syntax

```
int ToLower ( int value)
```

Description

Convert a character to its lowercase equivalent.

Parameters

Parameter	Description
`value`	Value to be converted.

Returns

The lowercase version of the character presented if it was uppercase, otherwise the character presented.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

30 Persistent Memory

30.1 Functions

30.1.1 PersistentMemErase

Syntax

```
void PersistentMemErase ( void )
```

Description

Clear the contents of the application persistent memory region.

Clear the contents of the application persistent memory region and indicate that the application is no longer using it. The region will be cleared and invalidated.

Parameters

Parameter	Description
None	-

Returns

Nothing.

30.1.2 PersistentMemGetSize

Syntax

```
uint8 PersistentMemGetSize ( void )
```

Description

Return the size of the application persistent memory region.

Return the size of the application persistent memory region. This represents the maximum number of words that can be stored in the region, although the application can store less if needed.

Parameters

Parameter	Description
None	-

Returns

Size of memory region in words.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

30.1.3 PersistentMemIsValid

Syntax

```
bool PersistentMemIsValid ( void )
```

Description

Allows the user application to determine if the persistent memory appears valid.

As the persistence is controlled by the slow discharge of a capacitor the exact period the persistent data remains valid is not specified. Also, the exact nature of corruption on the RAM around the point the capacitor charge fully dissipates cannot be specified. "Validity" therefore cannot be guaranteed.

Parameters

Parameter	Description
None	-

Returns

TRUE if the application region appears to hold valid data.

30.1.4 PersistentMemRead

Syntax

```
uint8 PersistentMemRead ( uint16 * buffer, uint8 num_words )
```

Description

Read the contents of the application persistent memory region.

Copy the contents of the application persistent memory region into the buffer supplied by the application. The application must specify how big the buffer is; if the buffer is larger than the region then only the contents of the region will be returned. The actual number of words copied is returned by the function.

Note:

This function will always return the contents of the region, even if those contents do not appear to be valid. PersistentMemIsValid() should be used to check validity.

Parameters

Parameter	Description
buffer	A pointer to the the buffer supplied by the application
num_words	Size of the buffer

Returns

The number of words copied to the buffer.

30.1.5 PersistentMemWrite

Syntax

```
uint8 PersistentMemWrite ( uint16 * buffer, uint8 num_words )
```

Description

Write to the application persistent memory region.

Copy the contents of the buffer supplied by the application into the application persistent memory region. The application must specify how big the buffer is; if the buffer is larger than the region then only enough data to fill the region will be copied. The actual number of words copied is returned by the function. After the data has been copied to the region, the firmware will calculate validation information, which will be used by PersistentMemIsValid() to determine whether or not the region is valid.

Parameters

Parameter	Description
buffer	A pointer to the the buffer supplied by the application
num_words	Size of the buffer

Returns

The number of words copied from the buffer.

31 Configuration Store

31.1 Functions

31.1.1 CSReadBdaddr

Syntax

```
bool CSReadBdaddr ( BD_ADDR_T * bdaddr)
```

Description

Read the device's Bluetooth address.
Unpacks the device's Bluetooth address from the Configuration Store into the buffer passed, in a format suitable for the rest of the firmware.

Parameters

Parameter	Description
bdaddr	A pointer to the storage buffer supplied by the application

Returns

TRUE for success, FALSE if some error

31.1.2 CSReadTxPower

Syntax

```
uint16 CSReadTxPower ( void )
```

Description

Read the TX Power setting.
Return the value stored in the TX Power CS key. This value is an integer, which corresponds to fixed steps in the transmit power level used by the device. It DOES NOT directly define a power level in dBm, as this will be board/design-specific Therefore if an application needs to determine an absolute power level in dBm the customer will need to perform additional calibration of the final product to determine how each step maps to an output power.

Parameters

Parameter	Description
None	-

Returns

The transmit power level

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

31.1.3 CSReadUserKey

Syntax

```
uint16 CSReadUserKey ( uint16 key_index)
```

Description

Read a key from the user key set.
Indexes into the array of user keys and returns the appropriate value. Raises a fault if the index is out of range.

Parameters

Parameter	Description
`key_index`	Index into the array of user keys

Returns

Value stored in the Configuration Store User Keys

32 Non-Volatile Memory

32.1 Functions

32.1.1 LargeSpiFlashDisable

Syntax

```
void LargeSpiFlashDisable ( void )
```

Description

Disable the Large SPI Flash library. Assumes the NVM controller is already initialised. Enables the NVM controller and restore PIOs.

Parameters

Parameter	Description
None	-

Returns

Nothing.

32.1.2 LargeSpiFlashEnable

Syntax

```
void LargeSpiFlashEnable ( void )
```

Description

Enable the PIOs for the Large SPI Flash library. Assumes memory is already powered up and NVM controller is not busy. Disable the NVM controller and set PIOs.

Parameters

Parameter	Description
None	-

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

32.1.3 LargeSpiFlashEraseBlock

Syntax

```
sys_status LargeSpiFlashEraseBlock ( large_spi_flash_erase_size
size, uint32 address, bool wait )
```

Description

Erase a single block of the Large SPI Flash device.

Parameters

Parameter	Description
`size`	[in] Size of block to erase. Refer large_spi_flash_erase_size Note: Not all Flash devices accept all sizes.
`address`	[in] 32-bit address in block to erase (typically set to the start of the block, though some Flash devices accept any address in the block).
`wait`	[in] TRUE: Wait for the erase operation to complete FALSE: Return as soon as the erase command has been issued

Returns

spi_status_hardware_busy: Another operation in progress spi_status_fail: Unknown block size requested spi_status_fail_timeout: Erase operation timed out sys_status_success: Success

32.1.4 LargeSpiFlashEraseWaitComplete

Syntax

```
sys_status LargeSpiFlashEraseWaitComplete ( void )
```

Description

Wait for an erase operation to complete.

Parameters

Parameter	Description
`None`	-

Returns

spi_status_fail_timeout: Erase operation timed out sys_status_success: Success.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

32.1.5 LargeSpiFlashInit

Syntax

```
void LargeSpiFlashInit ( uint16 mosi, uint16 miso, uint16 clk,
uint16 ncs, uint16 pow )
```

Description

Configure the Large SPI Flash library.

Parameters

Parameter	Description
mosi	[in] PIO to use for SPI MOSI line, or SPI_FLASH_DEFAULT_PIO for the default
miso	[in] PIO to use for SPI MISO line
clk	[in] PIO to use for SPI clock line
ncs	[in] PIO to use for SPI slave select line, or SPI_FLASH_DEFAULT_PIO for the default
pow	[in] PIO to use for NVM power control, or SPI_FLASH_DEFAULT_PIO for the default

Returns

Nothing.

32.1.6 LargeSpiFlashRead

Syntax

```
sys_status LargeSpiFlashRead ( uint32 address, uint16 length,
uint16 * data )
```

Description

Read data from the Large SPI Flash device.

Parameters

Parameter	Description
address	[in] 32-bit address to start read from
length	[in] Number of octets to read
data	[in] Data buffer to hold data read

Returns

spi_status_hardware_busy: Another operation in progress spi_status_fail_timeout: Read operation timed out sys_status_success: Success

32.1.7 LargeSpiFlashWrite

Syntax

```
 sys_status LargeSpiFlashWrite ( uint32 address, uint16 length,
const uint16 * data )
```

Description

Write data to the Large SPI Flash device.

Parameters

Parameter	Description
address	[in] 24-bit address to write to
length	[in] Number of octets to write (maximum LSF_PAGE_SIZE)
data	[in] Data buffer to write to Large SPI Flash device

Returns

spi_status_hardware_busy: Another operation in progress spi_status_page_overflow: Too much data to write in one operation
spi_status_fail_timeout: Write operation timed out sys_status_success: Success

32.1.8 NvmConfigureI2cEeprom

Syntax

```
 sys_status NvmConfigureI2cEeprom ( void )
```

Description

Configure the NVM manager to use an I2C EEPROM for the NVM Store.
CSR1000 supports both I2C EEPROM and SPI Flash boot devices, therefore the application must initialise the NVM manager with the appropriate type of device. This function is used to set up access to an I2C EEPROM device.
This function is typically called once, either as part of the application initialisation (in AppInit()) or just before the first read, write, or erase access to the NVM device.
The CS key "I2C EEPROM Initialisation Time" defines the time required by the I2C EEPROM device after it has powered on until it is ready to operate. This figure can typically be found in the datasheet for the selected device.
Refer to the description of CS Key nvm_start_address and NvmSetI2cEepromDeviceAddress usage in case EEPROM is greater than 0.5 Mbits (64 kilobytes).

Parameters

Parameter	Description
`None`	-

Returns

Status of operation.

32.1.9 NvmConfigureSpiFlash

Syntax

```
sys_status NvmConfigureSpiFlash ( void )
```

Description

Configure the NVM manager to use a SPI Flash for the NVM Store.

CSR1000 supports both I2C EEPROM and SPI Flash boot devices, therefore the application must initialise the NVM manager with the appropriate type of device. This function is used to set up access to a SPI Flash device.

This function is typically called once, either as part of the application initialisation (in AppInit()) or just before the first read, write, or erase access to the NVM device.

The CS key "SPI Flash Initialisation Time" defines the time required by the SPI Flash device after it has powered on until it is ready to operate. This figure can typically be found in the datasheet for the selected device.

The CS key "SPI flash block size" should be set to the size in *bytes* of a single erasable block within the SPI Flash boot device. This figure can be found in the data-sheet for the selected device. If nvm_num_spi_blocks is 2 then it's mandatory to set this CS key equal to 'flash erase sector size'.

Note:

Scenarios:

(1) spi_flash_block_size is smaller than the actual erase block size and nvm_num_spi_blocks is 1 => Some of the storage in the flash block is unused(and thus wasted).

(2) spi_flash_block_size is smaller than the actual erase block size and nvm_num_spi_blocks is 2 => Erase operations will span multiple NVM blocks and data corruption will occur.

(3) spi_flash_block_size is larger than the actual erase block size, =>An NVM erase will only erase part of the block, and subsequent writes to the unerased area will fail. This scenario occurs whether nvm_num_spi_blocks is 1 or 2.

The CS key "NVM num SPI blocks" specifies the number of consecutive erasable SPI Flash memory blocks that are available for the NVM. If two blocks are available, then when the first block becomes full, the SPI Flash NVM manager will be able to automatically copy the information into the second (spare) block before the first block is erased. This will reduce the chance of data corruption if the power fails during a write operation. If only one block is available, then when it becomes full the application will need to erase the block (and optionally copy existing data back into it) before new information can be stored. This introduces a small window in which power loss could result in loss of data.

Parameters

Parameter	Description
`None`	-

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

Status of operation.

32.1.10 NvmDisable

Syntax

```
sys_status NvmDisable ( void )
```

Description

Disable the NVM manager and power off the underlying storage device.

This function can be used to turn off the power to the NVM storage device. The NVM driver will retain the configuration state after powering off the device. This allows the application to subsequently call NvmRead(), NvmWrite(), and NvmErase() without having to call NvmConfigureI2cEeprom() or NvmConfigureSpiFlash() again.

Parameters

Parameter	Description
None	-

Returns

Status of operation.

32.1.11 NvmErase

Syntax

```
sys_status NvmErase ( bool erase_all)
```

Description

Erase the contents of the NVM Store.

For SPI Flash devices, if the erase_all parameter is TRUE then all firmware control information is also erased. The erase_all parameter is not used on an I2C EEPROM.

If the NVM device is currently disabled (via a previous call to NvmDisable()) it will be automatically re-enabled before the Erase operation is performed.

Parameters

Parameter	Description
erase_all	SPI Flash only: erase all firmware control information as well

Returns

Status of operation.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

32.1.12 NvmRead

Syntax

```
sys_status NvmRead ( uint16 * buffer, uint16 length, uint16
offset )
```

Description

Read words from the NVM Store.

Read words starting at the word offset, and store them in the supplied buffer.

If the NVM device is currently disabled (via a previous call to NvmDisable()) it will be automatically re-enabled before the Read operation is performed.

Parameters

Parameter	Description
buffer	The buffer to read words into
length	The number of words to read
offset	The word offset within the NVM Store to read from

Returns

Status of operation.

32.1.13 NvmSetI2cEepromDeviceAddress

Syntax

```
void NvmSetI2cEepromDeviceAddress ( uint16 address)
```

Description

Set the EEPROM device address during I2C operation.

This sets EEPROM device address excluding the Read/Write bit. Firmware will left shift this value by 1 and set the R/W bit according to the requested operation. The default device address is 0x50.

Parameters

Parameter	Description
address	Device address

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

32.1.14 NvmSize

Syntax

```
sys_status NvmSize ( uint16 * size_of_storage)
```

Description

Return the size in words of the NVM Store.
This function can be called prior to initialising the NVM device via NvmConfigureI2cEeprom() or NvmConfigureSpiFlash(). If it is called while the NVM device is disabled called, it will *not* re-enable the NVM device.

Parameters

Parameter	Description
`size_of_storage`	Pointer to integer to size in

Returns

Status of operation.

32.1.15 NvmWrite

Syntax

```
sys_status NvmWrite ( const uint16 * buffer, uint16 length,
uint16 offset )
```

Description

Write words to the NVM Store.
Write words from the supplied buffer into the NVM Store, starting at the given word offset
If the NVM device is currently disabled (via a previous call to NvmDisable()) it will be automatically re-enabled before the Write operation is performed.

Parameters

Parameter	Description
`buffer`	The buffer to write
`length`	The number of words to write
`offset`	The word offset within the NVM Store to write to

Returns

Status of operation.

32.2 Enumerations

32.2.1 enum large_spi_flash_erase_size

Syntax

```
enum large_spi_flash_erase_size
```

Description

Range of block sizes that may be erased. Not all sizes are supported on all devices: AT25DF011: Supports all sizes MX25L4006E: large_spi_erase_4KB supported large_spi_erase_32KB erases 64KB blocks large_spi_erase_256B not supported.

Enumerations

Enumeration	Description
large_spi_erase_4KB	4KB block
large_spi_erase_32KB	32KB block (64KB on MX25L4006E)
large_spi_erase_256B	256B page (not supported on MX25L4006E)

32.3 Defines

32.3.1 NVM_DEFAULT_ERASED_WORD

Definition

```
#define NVM_DEFAULT_ERASED_WORD    0xFFFF
```

Description

The default value of an erased location within the NVM Store.
This value may be device-dependent.

32.3.2 NVM_MINIMUM_SIZE

Definition

```
#define      NVM_MINIMUM_SIZE    32
```

Description

The minimum size allowed for the NVM Store, in words.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

33 System

- Battery
- Build Identifier: Information about application build, firmware build and Bluetooth version supported.
- Panic: Incorrect function from firmware should normally trigger a fault. Based on CSKey settings this can trigger a watchdog.
- Power Management
- Random Numbers
- Reset
- Thermometer
- Time
- Timers
- System-wide Constants
- System-wide Status Codes

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

34 Battery

34.1 Functions

34.1.1 BatteryReadLowThreshold

Syntax

```
uint16 BatteryReadLowThreshold ( void )
```

Description

Allows the application to query the Low Battery Threshold CSKey.

Parameters

Parameter	Description
None	-

Returns

The value of the CS Key "Low Battery Threshold".

34.1.2 BatteryReadVoltage

Syntax

```
uint16 BatteryReadVoltage ( void )
```

Description

Allows the application to query the battery voltage.

Parameters

Parameter	Description
None	-

Returns

The current battery voltage in mV.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

35 Build Identifier

35.1 Functions

35.1.1 IdGetLlVerBtle

Syntax

```
uint16 IdGetLlVerBtle (void)
```

Description

Supported version of Bluetooth specification. Refer to Bluetooth assigned numbers for Link Layer version values.

Returns

Link Layer version.

35.1.2 IdGetRomBuild

Syntax

```
uint16 IdGetRomBuild(void)
```

Description

Get build identifier of ROM part of firmware.

Returns

Build Identifier of ROM.

35.1.3 IdSetAppBuild

Syntax

```
void IdSetAppBuild(uint16 id)
```

Description

Set build identifier of Application. (Used by UCI).

Parameters

Parameter	Description
`[in] id`	Unique id to identify an application build.

CSR

35.1.4 IdSetAppString

Syntax

```
void IdSetAppString(const char * id_str, uint16 str_len)
```

Description

Set location of application's build identifier string. (Used by UCI).

Parameters

Parameter	Description
`[in] id_str`	Unique string to identify an application build.
`[in] str_len`	Length of string passed in.

Returns

Parameter `id_str` should point to a string that will exist permanently in memory.

35.2 Defines

35.2.1 BUILD_IDENTIFIER_STRING

Definition

```
#define BUILD_IDENTIFIER_STRING   "bdkSDK2_6_1_268_16..."
```

Description

Firmware abbreviated ID string.
Not currently used by the FW but retained to keep build system happy.

35.2.2 BUILD_IDENTIFIER_STRING_FULL

Definition

```
#define BUILD_IDENTIFIER_STRING_FULL "bdk_SDK_2_6_1_268_________1606061318"
```

Description

Firmware full ID string.
This is stored in ROM constant data and can be retrieved by various means (e.g. direct access via SLT, UCI command, etc.). Non-release builds use a generic "Unknown_TIMESTAMP" string rather than a formal label like below.

36 Panic

36.1 Functions

36.1.1 Panic

Syntax

```
void Panic ( uint16 panic_code)
```

Description

Raise a system panic, normally resetting the chip.

In normal operation, this function will not return since it resets the chip. If the CS Key err_panic ("Cause faults to panic") is changed from its default setting to prevent FAULT_APPLICATION_PANIC from panicking the chip, this function will return. This is not normally advisable.

Parameters

Parameter	Description
panic_code	An application-defined value. This value is preserved across resets to assist debugging. Applications are strongly advised to refrain from using 0 as valid panic code, even though it is currently treated as valid code by firmware. See PANIC_NONE.

Returns

Nothing. Does not return under normal circumstances.

36.1.2 PanicClearAppPanic

Syntax

```
sys_status PanicClearAppPanic ( void )
```

Description

Clear application panic code; setting it to PANIC_NONE.

Parameters

Parameter	Description
None	-

Returns

Status of clear operation.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

36.1.3 PanicClearFwFault

Syntax

```
sys_status PanicClearFwFault ( void )
```

Description

Allows the user application to clear last FW fault ID.
This does not affect the fault statistics maintained by firmware.

Parameters

Parameter	Description
None	-

Returns

Status of clear operation.

36.1.4 PanicReadAppPanic

Syntax

```
sys_status PanicReadAppPanic ( uint16 * pcode)
```

Description

Retrieve last application panic code.
This does not take care of validity of persistent store, so it could return PANIC_NONE in case firmware determined that it needs to reset whole persistent memory at boot up.

Parameters

Parameter	Description
pcode	pointer to location where panic code will be placed.

Returns

Status of read operation.

36.1.5 PanicReadFwFault

Syntax

```
sys_status PanicReadFwFault ( uint16 * fid)
```

Description

Allows the user application to retrieve last FW fault ID.

FW fault ID is stored in persistent memory. This API does not take care of persistent memory validity. This could return FAULT_NONE in case FW determines that whole persistent memory needs to be reset at boot up. See fault_doxy.h for more information on fault ID.

Parameters

Parameter	Description
`fid`	pointer to location where fault ID will be placed.

Returns

Status of read operation.

36.2 Firmware Library Fault Codes

The firmware library has an internal fault-reporting system which can detect some exceptional error conditions. In most cases customers should never see these fault codes recorded, other than FAULT_APPLICATION_PANIC which will be recorded if the application calls the Panic() function.

FAULT_NONE (0x00)	Marks unused entries in the fault log. Also occasionally useful in test circumstances to indicate success.
FAULT_MYSTERY (0x01)	Indicates that some unspecified error has occurred. Except in test circumstances a more specific fault code should always be preferred.
FAULT_BAD_LC_STATE (0x02)	State machine controlling a BTLE link is in an invalid state.
FAULT_BUFFER_CORRUPTED (0x03)	The internal state of one of the firmware's circular buffers has been corrupted. Note that at present the circular buffer subsystem is not in use, so this fault should not appear.
FAULT_USER_CSKEY_OUT_OF_RANGE (0x04)	The CSR1000 device provides a set of eight 16-bit values that can be set when the device is programmed. Applications can read these values using the function CSReadUserKey(), supplying an index between 0 and 7 inclusive. This fault indicates that an application supplied an index of 8 or more to CSReadUserKey().
FAULT_INVALID_LC_INDEX (0x05)	A CSR1000 device has a fixed number of controllers managing BTLE links. This fault indicates that an attempt was made to access a link controller that does not exist.
FAULT_H4_RX_BAD_PDU (0x06)	An error was detected while receiving data from a host device.
FAULT_BAD_FAULT (0x07)	A fault was raised, but the fault code supplied was not in the valid range. This fault code is used in place of the invalid one.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

FAULT_ADC_TENBIT_TIMEOUT (0x08)	CSR1000 devices have built-in Analogue to Digital Converters for a number of purposes. This fault indicates that an ADC has become "stuck" in some manner.
FAULT_WD_TIMER_RESOURCE (0x09)	CSR1000 devices maintain a background process that performs various vital pieces of system maintenance. This fault indicates that the timer controlling this process could not be acquired because the firmware ran out of resources. It can occur if applications use too many timers for their own control.
FAULT_HAL_CDAC_TABLE_BUILD (0x0a)	This fault indicates that for some reason the radio could not be set up within expected tolerances.
FAULT_HCI_BUFFER_FULL (0x0b)	Failed to send a message over the HCI to the host application because of a resource shortage. Note that this error is not reported by default, to avoid entering a possible infinite loop.
FAULT_H4_UNKNOWN_EVENT (0x0c)	This fault indicates that the firmware was asked to send an event to the host which was not recognised as a valid BTLE event.
FAULT_UNEXP_MSG_RCVD_FROM_ATT (0x0d)	An unexpected message was received from the ATT module while carrying out a GATT procedure.
FAULT_SA_HNDL_ARRAY_VIOLATION (0x0e)	The internal firmware state driving GATT procedures was found to be in an invalid state.
FAULT_GATT_CON_DB_FULL_MASTER_ROLE (0x0f)	This fault is raised if a device successfully creates a connection in Master mode but runs out of resources to record it internally.
FAULT_GATT_CON_DB_FULL_SLAVE_ROLE (0x10)	This fault is raised if a device successfully creates a connection in Slave mode but runs out of resources to record it internally.
FAULT_APPLICATION_PANIC (0x11)	This fault is raised when the application calls the Panic() function. It is not reported by default, to avoid confusing the application further, but does panic the device.
FAULT_UPDATE_EXCEEDED_RUNTIME (0x12)	Background firmware tasks must run to tight timescales to avoid disrupting radio traffic and breaking the specification. This fault is raised when a task exceeded its allotted time, and will consequently have interfered with existing connections in an unpredictable manner.
FAULT_INTERRUPT_UNBLOCK (0x13)	The firmware's internal interrupt management state has become inconsistent. Attempting to report this fault is probably futile, so by default it simply panics.
FAULT_L2CAP_HANDLER_NOT_REGISTERED (0x14)	The L2CAP code relies on a handler function being registered with it for each of the L2CAP services being used. This fault indicates that a call has been made to a particular service for which no handler has been registered. This may suggest that the L2CAP initialiser function, l2cap_init, has not been called.

FAULT_HIBERNATE_TIME_TOO_SHORT (0x15)	When the application requests the CSR1000 device to move to the Hibernate state it has to provide the minimum time spent hibernating. This time should be at least 2^20 microseconds (1.048576s). If the supplied time is too short this fault will be raised.
FAULT_LS_INVALID_CONNECTION (0x16)	Upper layers need to allocate resources to BLE connections as they are established. These resources should always be available; this fault indicates that a major firmware error has caused them to be unavailable.
FAULT_SM_UNEXPECTED_CID (0x17)	The Security Manager has been asked to handle security through a Channel ID that is not the fixed CID reserved for it.
FAULT_ATT_UNEXP_MSG_RCVD_FROM_L2CAP (0x18)	This fault indicates that an unexpected message is received by ATT module from L2CAP.
FAULT_SLOW_CLOCK_FREQ_TRIM (0x19)	The firmware was unable to complete the trim procedure for the 32kHz slow clock frequency, when trimmed against the 16MHz clock.
FAULT_INVALID_UART_BUFFER_SIZE (0x1a)	The application requested an invalid buffer size for the UART RX or UART TX buffer. Supported buffer sizes are defined by the 'uart_buf_size_bytes' enumeration in uart.h
FAULT_FW_TIMER_RESOURCES_EXHAUSTED (0x1b)	The firmware library tried to allocate an internal timer but did not have any free timer resources.
FAULT_INVALID_UART_CONSUMPTION (0x1c)	The application claims to have consumed more data from the UART RX buffer than was available, causing a receive buffer underflow.
FAULT_INCORRECT_ROM_VERSION (0x1d)	The ROM version is not compatible with the SDK used to build the application.

CSR

37 Power Management

37.1 Functions

37.1.1 SleepModeChange

Syntax

```
void SleepModeChange ( sleep_mode new_mode)
```

Description

Tell the firmware to use a particular sleep mode for all subsequent periods when it is able to sleep.

Parameters

Parameter	Description
new_mode	The new sleep mode to use

Returns

Nothing.

37.1.2 SleepRequest

Syntax

```
void SleepRequest ( sleep_state new_sleep_state, bool
wake_active_high, time48 hibernate_duration )
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

Request a transition to one of the sleep states Hibernate or Dormant, as specified by new_sleep_state.

Hibernate and Dormant states can be requested irrespective of the currently selected sleep mode (i.e. they are allowed even if the sleep mode is set to sleep_mode_never).

The wakeup condition is specified by the wake_active_high parameter. Set this to TRUE if the chip should wake when the WAKE pin is at logic level 1, or set to FALSE if the chip should wake at logic level 0.

Note:

> The CSR100x/CSR101x does not have an internal pull-up or pull-down resistor on the WAKE pin therefore an external resistor should be used.

When requesting Hibernate, a non-zero duration must be provided. This will be added onto the current system time to calculate the next wakeup time. The device will automatically wake up after the time has passed, or sooner if there is an external wake event on the dedicated WAKE pin. hibernate_duration is expressed in microseconds. However, the minimum allowed duration is 2^20 microseconds (1.048576s). In practise most real-world applications will actually require a much longer duration in order to realise any additional power-savings over Deep Sleep mode. The firmware will raise a fault and panic if the duration is too small.

The requested hibernate_duration is a suggested minimum. The system will sleep for at least that time, unless there is an external WAKE event, but it will always be slightly longer until the chip has completely woken up, due to system startup time. The exact time from an application requesting Hibernate until it gets called again at wakeup cannot be guaranteed.

The hibernate_duration is ignored if the application requests Dormant. In this state the device will *only* wake up if there is an external event on the WAKE pin.

WARNING! This function will *NOT* return. The device will go to a low power state and will perform a full RAM reset when it wakes up again. Any information that you need to save must be stored in the persistent memory before requesting the low power state.

Parameters

Parameter	Description
`new_sleep_state`	The requested sleep state (Hibernate or Dormant)
`wake_active_high`	WAKE pin polarity (acitve high or active low)
`hibernate_duration`	Time (in microseconds) to hibernate for

Returns

NEVER RETURNS!

37.1.3 SleepWakeOnUartRX

Syntax

```
void SleepWakeOnUartRX ( bool enable)
```

Description

Wake from Deep Sleep if the UART sees incoming data. On enabling the UART, the default is to wake on incoming data. This function may only be called when the UART is enabled either via DebugInit or UartEnable.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
`enable`	TRUE to wake from Deep Sleep on UART RX, FALSE to remain asleep

Returns

Nothing.

37.1.4 SleepWakePinEnable

Syntax

```
void SleepWakePinEnable ( wakepin_mode mode)
```

Description

Set the operating mode of the WAKE pin.
Allows the user application to configure the operating mode of the chip's WAKE pin.

Parameters

Parameter	Description
`mode`	The new WAKE pin mode to use

Returns

Nothing.

37.1.5 SleepWakePinStatus

Syntax

```
bool SleepWakePinStatus ( void )
```

Description

Return the current state of the WAKE pin.

Parameters

Parameter	Description
`None`	-

Returns

TRUE if WAKE pin is high else FALSE.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

37.2 Enumerations

37.2.1 enum sleep_mode

Syntax

```
enum sleep_mode
```

Description

The power state used by the chip when the radio is idle.

These are the values that can be written to the sleep_mode Configuration Store key, and also requested by the application using the SleepModeChange() API call.

Three sleep modes are supported: Deep Sleep; Shallow Sleep; Always Awake

Deep Sleep is the most useful, and ensures that the chip is using the 32kHz clock when the application, firmware, and radio are idle. The firmware will automatically handle the transition between the Deep Sleep and Awake states subject to system events (e.g. radio activity, expiry of timers, external interrupt sources, etc.)

Shallow Sleep is useful in certain application that want to use the main processor or the 8051 PIO Controller to perform high-speed or low-latency I/O. In this mode, the chip remains running off the 16MHz clock at all times therefore there is little latency when the chip needs to wake up. However this mode will increase the overall current consumption quite considerably. Whilst in Shallow Sleep the internal Power Supply will run in an optimised state, in order to reduce its current consumption, at any time the radio is not in use. As with Deep Sleep, in this state the firmware will manage the power supply settings subject to system events.

Always Awake is of limited use. The chip remains running off the 16MHz clock and the internal power supply runs in its normal "awake" state where it is capable of supplying full power to the radio, even if the radio is not in use.

There are two methods of controlling the sleep mode used by the firmware. The first is to set the sleep_mode CS key. The value set in this key will be used by the firmware from power-up until (and if) the application requests an alternate mode by calling the SleepModeChange() API. If the application does not need to change the selected sleep mode at runtime then it is sufficient to just set the CS key.

Enumerations

Enumeration	Description
sleep_mode_never	Always Awake
sleep_mode_deep	Deep Sleep
sleep_mode_shallow	Shallow Sleep

37.2.2 enum sleep_state

Syntax

```
enum sleep_state
```

Description

The state the chip woke up from.

If the application requests Hibernate or Dormant, and the power is subsequently removed, the sleep state recorded on the next wake event will be dependent on how long power was lost for (i.e. whether or not the data in the persistent memory was still valid).

CSR

Enumerations

Enumeration	Description
`sleep_state_cold_powerup`	The device powered up after a long time without power
`sleep_state_warm_powerup`	The device powered up after a short time without power (less than ~1 minute, based on how long data remains valid in the persistent memory)
`sleep_state_dormant`	The device powered up after being placed into the Dormant state
`sleep_state_hibernate`	The device powered up after being placed into the Hibernate state
`sleep_state_warm_reset`	The device powered up after an application-triggered warm reset

37.2.3 enum wakepin_mode

Syntax

```
enum wakepin_mode
```

Description

The WAKE pin mode controls which edges the CSR100x/CSR101x will wake on (if required).

Enumerations

Enumeration	Description
`wakepin_mode_disable`	Disable the WAKE pin
`wakepin_mode_low_level`	Pulling the WAKE pin low keeps the CSR100x/CSR101x awake
`wakepin_mode_high_level`	Pulling the WAKE pin high keeps the CSR100x/CSR101x awake

38 Random Numbers

38.1 Functions

38.1.1 Random16

Syntax

```
uint16 Random16 ( void )
```

Description

Obtain a 16-bit random number.

Parameters

Parameter	Description
`None`	-

Returns

A 16-bit pseudo-random number

38.1.2 RandomGenPrbs

Syntax

```
void RandomGenPrbs ( uint8 feed_back_max, uint8 feed_back_min,
uint8 * data, uint16 size )
```

Description

Generate prbs data using two feed back bits.

Parameters

Parameter	Description
`feed_back_max`	See ITU-T 0.150 for details.
`feed_back_min`	
`data`	Pointer to the buffer where the numbers will be placed.
`size`	How many numbers to generate. Limits 0 to 2**15-1.

Returns

Void

CSR µEnergy Firmware Library Documentation

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

38.2 Defines

38.2.1 Random32

Definition

```
#define Random32 ( ) ((((uint32)Random16()) << 16) | Random16())
```

Description

Obtain a 32-bit random number.

38.2.2 RandomGenPrbs15

Definition

```
#define RandomGenPrbs15 ( data, length ) RandomGenPrbs(14, 13,
data, length)
```

Description

Generate PRBS15 data.

Parameter

Variable	Description
data	Pointer to the buffer where the numbers will be placed.
size	How many numbers to generate. Limits 0 to 2**15-1.

38.2.3 RandomGenPrbs9

Definition

```
#define RandomGenPrbs9 ( data, length ) RandomGenPrbs(8, 4, data,
length)
```

Description

Generate PRBS9 data.

Parameter

Variable	Description
data	Pointer to the buffer where the numbers will be placed.
size	How many numbers to generate. Limits 0 to 2**15-1.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

39 Reset

39.1 Functions

39.1.1 WarmReset

Syntax

```
void WarmReset ( void )
```

Description

Trigger a warm reset of the CSR1000, reloading code from the boot device and resetting the hardware.

Parameters

Parameter	Description
None	-

Returns

[Never returns].

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

40 Thermometer

40.1 Functions

40.1.1 ThermometerReadTemperature

Syntax

```
int16 ThermometerReadTemperature ( void )
```

Description

Allows the application to query the thermometer.

Note that this function does not trigger a temperature measurement, it returns a cached value of the temperature that the firmware updates every watchdog period, by default every 15 seconds.

Parameters

Parameter	Description
None	-

Returns

The cached temperature in degrees centigrade.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

41 Time

41.1 Functions

41.1.1 TimeCmp48LT

Syntax

```
bool TimeCmp48LT ( time48 t1, time48 t2 )
```

Description

Compare two 48-bit time values.
Check if the first time is behind the second time, using the usual "half the wrap period" definitions of 'behind' and 'ahead'.

Parameters

Parameter	Description
`t1`	A 3 word array containing the 1st time value.
`t2`	A 3 word array containing the 2nd time value.

Returns

TRUE if the first time is behind the second, otherwise FALSE.

41.1.2 TimeDelayUSec

Syntax

```
void TimeDelayUSec ( uint16 delay)
```

Description

Delay for a given number of microseconds.
Delay for at least delay microseconds, and less than delay + 1 microseconds (this arises because the code allows for the timer ticking just before it first measures it, so errs on the side of a longer delay rather than an unexpectedly short one).

Warning:"delay" should be strictly less than 65535 microseconds. For safety, especially where the possibility exists of running at a slower clock speed, a few microseconds of margin are advised.

Parameters

Parameter	Description
`delay`	The delay value in microseconds.

Returns

Nothing

41.1.3 TimeGet16

Syntax

```
uint16 TimeGet16 ( void )
```

Description

Read the current system time, 16 bits worth.

Returns the current value of the low 16 bits of the system's 48 bit 1MHz clock. The resolution of the returned uint16 is one microsecond, so this value wraps after approximately 65ms.

Note:

Due to the fast speed of the clock, using just the lowest 16 bits is typically only useful when making quick differential measurements, e.g to check execution time of task that is guaranteed to finish within 65ms.

The function can be called from the machine's background or from interrupt routines.

Parameters

Parameter	Description
None	-

Returns

The current system time.

Note:

The time manipulation macros are not appropriate here. But make sure you know what you are doing with wraps and int16/uint16 stuff.

41.1.4 TimeGet32

Syntax

```
uint32 TimeGet32 ( void )
```

Description

Read the current system time, 32 bits worth.

Returns the current value of the low 32 bits of the system's 48 bit 1MHz clock. The resolution of the returned uint32 is one microsecond, so this value wraps after approximately 71 minutes.

This clock is the basis of all timed events in the chip's hardware, notably other functions declared in this file.

The function can be called from the machine's background or from interrupt routines.

Parameters

Parameter	Description
None	-

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

The current system time.

41.1.5 TimeGet48

Syntax

```
uint32 TimeGet48 ( uint16 * time_msw)
```

Description

Read the current full 48-bit system time.

Returns all 48 bits of the current value of the system's 1MHz clock. The resolution of the returned 32 bit value is one microsecond, with the most significant word returned through the pointer passed as a parameter. The clock therefore wraps after approximately 8.9 years. That should be enough for most purposes.

Except as noted above, this function can be treated as per TimeGet32().

Parameters

Parameter	Description
`time_msw`	A pointer to store the most significant word of the current system time

Returns

Lower 32 bits to the current system time.

41.1.6 TimeGet48WithOffset

Syntax

```
void TimeGet48WithOffset ( uint16 * time48, uint16 offset )
```

Description

Read the current 48-bit time plus a 16-bit offset.

As per TimeGet48(), but treats the time as an array of three uint16s rather than a uint16 MSW and uint32 LSW, and adds a uint16 offset to the current time.

Parameters

Parameter	Description
`time48`	A 3 word array to store the return value.
`offset`	Value to added to the current system time.

Returns

Nothing

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

41.1.7 TimeIncrement48

Syntax

```
void TimeIncrement48 ( time48 t, uint16 increment )
```

Description

Add a 16-bit offset to a 48-bit time in-place.
The result is written back into the original parameter.

Parameters

Parameter	Description
`t`	A 3 word array containing the source/destination variable.
`increment`	The amount "t" is to increased.

Returns

Nothing

41.1.8 TimeSub48

Syntax

```
void TimeSub48 ( time48 t_result, time48 t1, time48 t2 )
```

Description

Subtract two 48-bit time values.
Sets t_result to t1-t2 (using 48-bit unsigned arithmetic).

Parameters

Parameter	Description
`t_result`	A 3 word array containing the result
`t1`	A 3 word array containing the 1st time value.
`t2`	A 3 word array containing the 2nd time value.

Returns

Nothing

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

41.2 Defines

41.2.1 TimeAdd

Definition

```
#define TimeAdd ( t1, t2 ) ((t1) + (t2))
```

Description

Add two 16- or 32-bit time values.

Parameter

Variable	Description
t1	1st time value.
t2	2nd time value.

41.2.2 TimeCmpEQ

Definition

```
#define TimeCmpEQ ( t1, t2 ) ((t1) == (t2))
```

Description

Determine if two 16- or 32-bit time values are equal.

Parameter

Variable	Description
t1	1st time value.
t2	2nd time value.

41.2.3 TimeCmpGE

Definition

```
#define TimeCmpGE ( t1, t2 ) (TimeSub((t1), (t2)) >= 0)
```

Description

Determine if one 16- or 32-bit time value is greater than or equal to another.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Parameter

Variable	Description
t1	1st time value.
t2	2nd time value.

41.2.4 TimeCmpGT

Definition

```
#define TimeCmpGT ( t1, t2 ) (TimeSub((t1), (t2)) > 0)
```

Description

Determine if one 16- or 32-bit time value is greater than another.

Parameter

Variable	Description
t1	1st time value.
t2	2nd time value.

41.2.5 TimeCmpLE

Definition

```
#define TimeCmpLE ( t1, t2 ) (TimeSub((t1), (t2)) <= 0)
```

Description

Determine if one 16- or 32-bit time value is less than or equal to another.

Parameter

Variable	Description
t1	1st time value.
t2	2nd time value.

41.2.6 TimeCmpLT

Definition

```
#define TimeCmpLT ( t1, t2 ) (TimeSub((t1), (t2)) < 0)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Determine if one 16- or 32-bit time value is less than another.

Parameter

Variable	Description
t1	1st time value.
t2	2nd time value.

41.2.7 TimeCopy48

Definition

```
#define TimeCopy48 ( td, ts ) ((td)[0] = (ts)[0], (td)[1] =
(ts)[1], (td)[2] = (ts)[2])
```

Description

Copy one 48-bit time value to another.
Copies the time in ts to td.

Parameter

Variable	Description
td	A 3 word array containing the destination variable.
ts	A 3 word array containing the source variable.

41.2.8 TimeSub

Definition

```
#define TimeSub ( t1, t2 ) ((int32) (t1) - (int32) (t2))
```

Description

Subtract two 16- or 32-bit time values.

Parameter

Variable	Description
t1	1st time value.
t2	2nd time value.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

41.2.9 TimeWaitWithAbsoluteTimeout16

Definition

```
#define TimeWaitWithAbsoluteTimeout16 ( cond, endtime, result )
```

Description

```
Value: do { \
 while (!(cond) && ((int16)(TimeGet16() - endtime) < 0)) \
 ; \
 result = (cond); \
 } while (0)
```

Busy-wait for a condition with a timeout. Busy wait for a condition to become true, with a timeout. This is a common pattern which people tend to get wrong: in a naive implementation, if we spend time processing an interrupt between checking the condition and checking the timeout, we can falsely decide that the condition is not met in time. This implementation checks the condition again after exiting the timing loop (so works on the assumption that the condition is stable once it has become TRUE).

Parameter

Variable	Description
`cond`	The expression you're waiting for.
`endtime`	The time at which to timeout.
`result`	Set to TRUE if the condition was met in time, FALSE otherwise.

41.2.10 TimeWaitWithAbsoluteTimeout32

Definition

```
#define TimeWaitWithAbsoluteTimeout32 ( cond, endtime, result )
```

Description

```
Value: do { \
 while (!(cond) && TimeCmpLT(TimeGet32(), endtime)) \
 ; \
 result = (cond); \
 } while (0)
```

Busy-wait for a condition on a timeout.

Busy wait for a condition to become true, with a timeout. This is a common pattern which people tend to get wrong: in a naive implementation, if we spend time processing an interrupt between checking the condition and checking the timeout, we can falsely decide that the condition is not met in time. This implementation checks the condition again after exiting the timing loop (so works on the assumption that the condition is stable once it has become TRUE).

CSR

Parameter

Variable	Description
cond	The expression you're waiting for.
endtime	The time at which to timeout.
result	Set to TRUE if the condition was met in time, FALSE otherwise.

41.2.11 TimeWaitWithTimeout16

Definition

```
#define TimeWaitWithTimeout16 ( cond, delay, result )
```

Description

```
Value: do { \
 uint16 wwt_start_time = TimeGet16(); \
 while (!(cond) && ((TimeGet16() - wwt_start_time) \
 <= (uint16)(delay))) \
 ; \
 result = (cond); \
 } while (0)
```

Busy-wait for a condition with a timeout.

Busy wait for a condition to become true, with a timeout. This is a common pattern which people tend to get wrong: in a naive implementation, if we spend time processing an interrupt between checking the condition and checking the timeout, we can falsely decide that the condition is not met in time. This implementation checks the condition again after exiting the timing loop (so works on the assumption that the condition is stable once it has become TRUE).

Note:

The maximum time you can wait for is strictly less than 65535 microseconds.

Parameter

Variable	Description
cond	The expression you're waiting for.
delay	The timeout in microseconds.
result	Set to TRUE if the condition was met in time, FALSE otherwise.

41.2.12 TimeWaitWithTimeout32

Definition

```
#define TimeWaitWithTimeout32 ( cond, delay, result )
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

```
Value: do { \
 TIME wwt_end_time = TimeAdd(TimeGet32(), delay); \
 while (!(cond) && TimeCmpLT(TimeGet32(), wwt_end_time)) \
 ; \
 result = (cond); \
 } while (0)
```

Busy-wait for a condition on a timeout.

Busy wait for a condition to become true, with a timeout. This is a common pattern which people tend to get wrong: in a naive implementation, if we spend time processing an interrupt between checking the condition and checking the timeout, we can falsely decide that the condition is not met in time. This implementation checks the condition again after exiting the timing loop (so works on the assumption that the condition is stable once it has become TRUE).

Parameter

Variable	Description
`cond`	The expression you're waiting for.
`delay`	The timeout in microseconds.
`result`	Set to TRUE if the condition was met in time, FALSE otherwise.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

42 Timers

42.1 Functions

42.1.1 AppBackgroundTick

Syntax

```
void AppBackgroundTick ( bool enable)
```

Description

Enable or disable the Application Background Tick event.

The Background Tick event is generated via the firmware internal watchdog timer (although at a lower priority). Therefore one event will be sent to the application for each watchdog tick. The tick rate is defined by the Configuration Store "Watchdog period" key.

One advantage of using the Background Tick rather than running a separate application timer is that the number of times the chip has to wake from Deep Sleep can be minimised (as the firmware watchdog processing and application Background Tick processing will occur during the same wakeup period). If the application does not require to perform "background" processing as often as the watchdog period, it could use a simple counter to skip some Background Tick events.

The Background Tick is presented to the application as a regular event (SYS_BACKGROUND_TICK_IND) to the AppProcessLmEvent() event handler.

By default when the system powers up the Background Tick is disabled. The application can enable or disable the tick whenever it needs it.

Parameters

Parameter	Description
`enable`	TRUE to enable the tick event; FALSE to disable the tick event

Returns

Nothing

42.1.2 TimerCreate

Syntax

```
timer_id TimerCreate ( uint32 const time, bool const relative,
timer_callback_arg handler )
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Insert an application timer into the timer queue.

The timeout period is measured in microseconds. time.h defines a number of constants for MILLISECOND, SECOND and MINUTE, e.g. allowing 10*SECOND to be used when starting a timer. Note that although the timeout value is a 32-bit number the maximum timeout period is actually (2^31)-1 microseconds (not (2^32)-1 microseconds) to enable safe 'roll over' handling. (2^31)-1 microseconds corresponds to approximately 35 minutes 47 seconds.

When the timer expires, the firmware will call the application timer handler function. Prior to calling the handler, the firmware will have "cleaned up" the timer structure, therefore the application does not need to manually delete the timer. If the application needs to restart the timer (for the same or a different duration) it can simply call TimerCreate() again.

Parameters

Parameter	Description
`time`	The number of microseconds the timer should run for.
`relative`	True => time is offset from "now".
`handler`	Pointer to the expiry callback function.

Returns

Timer reference, or TIMER_INVALID if the timer could not be started.

42.1.3 TimerDelete

Syntax

```
void TimerDelete ( timer_id const tid)
```

Description

Delete a timer from the queue.

Parameters

Parameter	Description
`tid`	Timer reference.

Returns

Nothing

42.1.4 TimerInit

Syntax

```
void TimerInit ( uint16 max_timers, void * timer_array )
```

Description

Initialise the application timers.

This function is used by the application to set up an array of timers. The Firmware library uses this array to manage timers on behalf of the application.

Warning:It is the responsibility of the application to ensure that the size of the array pointed to by timer_array is big enough to hold all the timer data required by the firmware. This size is SIZEOF_APP_TIMER * max_timers.

Parameters

Parameter	Description
`max_timers`	Maximum number of application timers available.
`timer_array`	Fixed-length array to hold timer information.

Returns

Nothing.

42.2 TypeDefs

42.2.1 typedef void(* timer_callback_arg)(timer_id const)

Syntax

```
typedef void( * timer_callback_arg)(timer_id const )
```

Description

Timer expiry call back function type.

42.2.2 typedef uint16 timer_id

Syntax

```
typedef uint16 timer_id
```

Description

Opaque type used to reference a timer.

42.3 Defines

42.3.1 SIZEOF_APP_TIMER

Definition

```
#define SIZEOF_APP_TIMER 6
```

CSR

Description

Size of structure for a single timer. See TimerInit for details on how to use this value.

CSR

43 System-wide Status Codes

43.1 Enumeration: enum sys_status

Syntax

```
enum sys_status
```

Description

HCI and extended system-wide status codes.

Please refer to the BlueTooth specifications V4.0, volume 2, part D for details of the HCI error codes in the range 0x00 - 0x3F.

The extended error codes (those above 0x0100) are documented here.

Enumerations

Enumeration	Description
sys_status_success	Generic "success" status code
nvm_status_empty	NVM is initialised but NVM Store is currently empty/ unused
nvm_status_needs_erase	NVM Store only has one block and is full (SPI Flash only)
nvm_status_invalid_configuration	NVM configuration in CS keys is invalid / not supported
nvm_status_not_initialised	NVM Store is not initialised
nvm_status_invalid_offset	Offset parameter is invalid or length runs past end of NVM Store
nvm_status_invalid_buffer	Buffer address parameter is invalid / NULL
i2c_status_waiting	Transaction is continuing
i2c_status_firmware_busy	Firmware is busy with another request
i2c_status_hardware_busy	Hardware is busy with another request
i2c_status_controller_disabled	The controller is currently disabled (see I2cEnable)
i2c_status_fail	General failure
i2c_status_fail_nacked	NACK received from I2C slave
i2c_status_fail_bus_busy	I2C bus was busy (external I2C bus master?)
i2c_status_fail_arb_lost	Bus arbitration lost during transaction
i2c_status_fail_timeout	Transaction failed to complete within timeout period
i2c_status_fail_inactive	No active transaction to complete

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumeration	Description
i2c_status_fail_unknown	Unknown error
i2c_status_fail_write_poll_timeout	Timeout while polling the EEPROM for completion of a write cycle
spi_status_waiting	Transaction is continuing
spi_status_hardware_busy	Hardware is busy with another request
spi_status_erase_pending	Erase operation has started but not completed
spi_status_fail	General failure
spi_status_fail_timeout	Transaction failed to complete within timeout period
spi_status_page_overflow	A write was requested with more data than fits in one page
l2cap_status_invalid_conn_state	Invalid connection state
l2cap_status_conn_disallowed	Connection disallowed
l2cap_status_conn_not_ongoing	Connection not ongoing for L2CAP client (ATT/SMP)
l2cap_status_buffer_full	Buffer full; cannot process data
ls_status_limited_advertising_timeout	Limited advertising time out
gatt_status_invalid_handle	The attribute handle given was not valid
gatt_status_read_not_permitted	The attribute cannot be read
gatt_status_write_not_permitted	The attribute cannot be written
gatt_status_invalid_pdu	The attribute PDU was invalid
gatt_status_insufficient_authentication	The attribute requires an authentication before it can be read or written
gatt_status_request_not_supported	Target device doesn't support request
gatt_status_invalid_offset	Offset specified was past the end of the long attribute
gatt_status_insufficient_authorization	The attribute requires authorization before it can be read or written
gatt_status_prepare_queue_full	Too many prepare writes have been queued
gatt_status_attr_not_found	No attribute found within the given attribute handle range.
gatt_status_not_long	This attribute cannot be read or written using the Read Blob Request or Write Blob Requests.
gatt_status_insufficient_encr_key_size	The Encryption Key Size used for encrypting this link is insufficient.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Enumeration	Description
gatt_status_invalid_length	The attribute value length is invalid for the operation.
gatt_status_unlikely_error	The attribute request that was requested has encountered an error that was very unlikely, and therefore could not be completed as requested.
gatt_status_insufficient_encryption	The attribute requires encryption before it can be read or written
gatt_status_unsupported_group_type	The attribute type is not a supported grouping attribute as defined by a higher layer specification.
gatt_status_insufficient_resources	Insufficient Resources to complete the request.
gatt_status_device_not_found	Error to indicate that request to LS can not be completed because the device entity is not found
gatt_status_sign_failed	Attribute signing failed.
gatt_status_busy	Operation can't be done now.
gatt_status_timeout	Current operation timed out.
gatt_status_invalid_mtu	Invalid MTU
gatt_status_invalid_uuid	Invalid UUID type
gatt_status_success_more	Operation was successful, and more responses will follow
gatt_status_success_sent	Indication sent, awaiting confirmation from the client
gatt_status_invalid_cid	Invalid connection identifier
gatt_status_invalid_db	Attribute database is invalid
gatt_status_db_full	Attribute server database is full
gatt_status_invalid_permissions	Attribute permissions are not valid
gatt_status_invalid_operation	Operation requested by HL is not valid
gatt_status_invalid_param_value	Invalid parameter value passed
gatt_status_data_validation_failed	Data validation failed during Reliable Writes procedure
gatt_status_irq_proceed	The application has authorised the Read or Write access which should now proceed through the database
gatt_status_app_mask	Start of Application error codes that may be defined by a higher layer specification. This value should be ORed with application status codes in the range 0x80 - 0xFF to ensure they are sent over the air to the peer device.
gatt_status_app_first_code	First valid GATT Application Error code

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumeration	Description
gatt_status_app_last_code	Last valid GATT Application Error code
sm_status_reserved	Reserved value
sm_status_passkey_entry_failed	Passkey input cancelled
sm_status_oob_not_available	Peer has no OOB data
sm_status_authentication_requirements	Unauthenticated pairing is not acceptable
sm_status_confirm_value_failed	Passkey input wrong
sm_status_pairing_not_supported	Peer is currently unable to perform pairing
sm_status_encryption_key_size	The negotiated encryption strength is not acceptable
sm_status_command_not_supported	Peer does not support this operation
sm_status_unspecified_reason	Something else went wrong
sm_status_repeated_attempts	Peer is experiencing excessive failed pairings; please wait
sm_status_invalid_parameters	Incorrect or Invalid arguments have been supplied
sm_status_last_standardised	Subsequent SM status definitions are internal and not standardised
sm_status_timeout	Peer did not respond
sys_status_invalid	Generic "invalid" status code

43.2 System-wide Status Codes

43.2.1 STATUS_GROUP_GATT

Definition

```
#define STATUS_GROUP_GATT   0x0A00
```

Description

GATT Status Codes

43.2.2 STATUS_GROUP_I2C

Definition

```
#define STATUS_GROUP_I2C   0x0200
```

Description

I2C Status Codes (incl. I2C EEPROM)

43.2.3 STATUS_GROUP_L2CAP

Definition

```
#define STATUS_GROUP_L2CAP   0x0800
```

Description

L2CAP Status Codes

43.2.4 STATUS_GROUP_LS

Definition

```
#define STATUS_GROUP_LS   0x0900
```

Description

Link Supervisor Status Codes

43.2.5 STATUS_GROUP_NVM

Definition

```
#define STATUS_GROUP_NVM   0x0100
```

Description

NVM Store Status Codes

43.2.6 STATUS_GROUP_SKM

Definition

```
#define STATUS_GROUP_SKM   0x0400
```

Description

Security Manager Status Codes

43.2.7 STATUS_GROUP_SM

Definition

```
#define STATUS_GROUP_SM   0x0B00
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Security Manager Status Codes

43.2.8 STATUS_GROUP_SPI

Definition

```
#define STATUS_GROUP_SPI   0x0300
```

Description

SPI Status Codes (incl. SPI Flash)

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

44 Debug

44.1 Functions

44.1.1 DebugInit

Syntax

```
void DebugInit ( uint16 rx_threshold, uart_data_in_fn
rx_event_handler, uart_data_out_fn tx_event_handler )
```

Description

Set up the IO system.

Parameters

Parameter	Description
rx_threshold	Number of bytes of input required to trigger a call to rx_event_handler. 1 is a safe value to pass if the application does not intend to receive data.
tx_event_handler	Pointer to a function of type uart_data_out_fn that will be called whenever a UART transmission has finished.
rx_event_handler	Pointer to a function of type uart_data_in_fn that will be called whenever the threshold number of bytes have been received over the UART.

Returns

Nothing.

44.1.2 DebugWriteChar

Syntax

```
void DebugWriteChar ( char const val)
```

Description

Write an ASCII character (unsigned 8-bit value) to the UART.

Parameters

Parameter	Description
val	The character to send.

Returns

Nothing.

44.1.3 DebugWriteString

Syntax

```
void DebugWriteString ( const char * string)
```

Description

Write a C string (i.e. unpacked data) over the UART.

Parameters

Parameter	Description
`string`	The NUL-terminated string to send.

Returns

Nothing.

44.1.4 DebugWriteTime48

Syntax

```
void DebugWriteTime48 ( uint16 const * val)
```

Description

Convert a 48-bit time value into an ASCII string of twelve hexadecimal digits and send it to the UART.

Parameters

Parameter	Description
`val`	The time48 value to convert and send.

Returns

Nothing.

44.1.5 DebugWriteUint16

Syntax

```
void DebugWriteUint16 ( uint16 const val)
```

Description

Convert a 16-bit value into an ASCII string of four hexadecimal digits and send it to the UART.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Parameters

Parameter	Description
`val`	The value to convert and send.

Returns

Nothing.

44.1.6 DebugWriteUint32

Syntax

```
void DebugWriteUint32 ( uint32 const val)
```

Description

Convert a 32-bit value into an ASCII string of eight hexadecimal digits and send it to the UART.

Parameters

Parameter	Description
`val`	The value to convert and send.

Returns

Nothing.

44.1.7 DebugWriteUint8

Syntax

```
void DebugWriteUint8 ( uint8 const val)
```

Description

Convert an 8-bit value into an ASCII string of two hexadecimal digits and send it to the UART.

Parameters

Parameter	Description
`val`	The value to convert and send.

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

44.2 Defines

44.2.1 DebugWriteData

Definition

```
#define DebugWriteData ( data, nbytes ) UartWriteBlocking(data,
nbytes)
```

Description

Write unpacked data over the UART.

Parameter

Variable	Description
data	The uint8 data to send.
nbytes	The number of bytes of data to send.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

45 Production Test

45.1 Functions

45.1.1 DirectTestEnd

Syntax

```
sys_status DirectTestEnd ( uint16 * num_packets)
```

Description

End the current radio test and return the number of packets processed.
This function can only be called when a test is running.

Parameters

Parameter	Description
num_packets	Pointer to storage for number of packets processed

Returns

Status of operation

45.1.2 DirectTestExtended

Syntax

```
void DirectTestExtended ( bool enable)
```

Description

Enable or disable CSR1000 Extended Direct Test Mode.
Extended Direct Test Mode allows the test system to use proprietary commands for trimming the 16MHz crystal or continuously transmitting a Carrier Wave on a specific RF channel. As these functions may interfere with third-party test equipment the application needs to enable this mode before it can be used.
This function should be called after calling DirectTestInit(). Extended Direct Test Mode is disabled if DirectTestInit() is called again.

Parameters

Parameter	Description
enable	TRUE to enable Extended Direct Test Mode, or FALSE to disable.

Returns

Nothing.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

45.1.3 DirectTestInit

Syntax

```
void DirectTestInit ( sleep_state last_sleep_state)
```

Description

Initialise the Bluetooth low energy 2-wire UART Direct Test Mode.

This function configures the UART for Direct Test Mode. While enabled the firmware will automatically handle all test commands received over the UART and start or stop the requested test mode.

The normal way to terminate Direct Test Mode is to perform a device reset (by sending the LE_Reset from the tester, or if the application calls WarmReset()). If the application manually reconfigures the UART without first resetting (e.g. directly calling UartInit(), or by calling DebugInit()), the test mode handler will be disabled, but there is a risk that a radio test will remain active.

The UART data rate and configuration is taken from the CS.

Warning:After calling this function it is recommended that the application does not directly call DirectTestReceive(), DirectTestTransmit() or DirectTestEnd(), as this could interfere with any tests initiated by an external tester via the UART.

Parameters

Parameter	Description
`last_sleep_state`	-

Returns

Nothing.

45.1.4 DirectTestReceive

Syntax

```
sys_status DirectTestReceive ( uint8 rx_channel)
```

Description

Start radio test mode, receiving test packets generated by a Bluetooth tester.

The caller should supply the RF channel number to receive packets on. The valid range for the channel is 0x00 to 0x27, which corresponds to a frequency range of 2402MHz - 2480MHz, calculated as F = (2*rx_channel) + 2402.

This function should not be called if a test is already running.

Warning:THIS FUNCTION IS FOR RF TESTING DURING DEVELOPMENT AND PRODUCTION TEST. It must not be used during normal application operation. After calling this and the other BLE Test functions, the radio may be in an unknown state therefore a device reset is recommended.

Parameters

Parameter	Description
`rx_channel`	Channel number to use for the test

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Returns

Status of operation

45.1.5 DirectTestReceiveResults

Syntax

```
uint16 DirectTestReceiveResults ( int16 * p_average_rssi)
```

Description

Return the number of packets received since the start of the test. If the passed in pointer is not NULL use that to store the averaged RSSI.

Parameters

Parameter	Description
p_average_rssi	Pointer to variable to hold averaged RSSI value or NULL, if the RSSI is not required

Returns

Number of packets

45.1.6 DirectTestTransmit

Syntax

```
sys_status DirectTestTransmit ( uint8 tx_channel, uint8
payload_length, ble_test_pkt_type payload_type, uint16 num_packets )
```

Description

Start radio test mode, transmitting test packets to a Bluetooth tester.

The caller should supply the RF channel number to transmit packets on. The valid range for the channel is 0x00 to 0x27, which corresponds to a frequency range of 2402MHz - 2480MHz, calculated as F = (2*rx_channel) + 2402.

The payload length ranges from 0x00 to 0x25 bytes. All standard Bluetooth test packet types are supported.

The caller can also provide the number of packets to transmit. If the test should run indefinitely, set the count to 0, and the test will only stop transmitting if DirectTestEnd() is called.

This function should not be called if a test is already running.

Warning:THIS FUNCTION IS FOR RF TESTING DURING DEVELOPMENT AND PRODUCTION TEST. It must not be used during normal application operation. After calling this and the other BLE Test functions, the radio may be in an unknown state therefore a device reset is recommended.

Parameters

Parameter	Description
`tx_channel`	Channel number to use for the test
`payload_length`	Number of bytes to transmit in the data payload
`payload_type`	Data payload format
`num_packets`	Number of packets to transmit, or 0 for indefinite

Returns

Status of operation

45.1.7 TestDisableCarrierWave

Syntax

```
void TestDisableCarrierWave ( void )
```

Description

Extended test function to allow the application to stop transmitting an unmodulated carrier wave.
This function only guarantees to stop radio transmission. It will leave the radio hardware and other parts of the chip in an undefined state unsuitable for normal operation. After CW testing has been completed it is recommended that the chip is reset using WarmReset().

Warning:This function must only be called during production test.

Parameters

Parameter	Description
`None`	-

Returns

Nothing

45.1.8 TestEnableCarrierWave

Syntax

```
void TestEnableCarrierWave ( uint16 rf_channel)
```

Description

Extended test function to allow the application to request continuous transmission of an unmodulated carrier wave on a given RF channel.

Legitimate values for the rf_channel parameter are 0 to 39. If the application attempts to use a channel higher than 39 the request will be ignored. The carrier wave transmission can be disabled by calling TestDisableCarrierWave().

Warning:This function must only be called during production test.

Parameters

Parameter	Description
`rf_channel`	RF channel range 0-39

Returns

Nothing

45.1.9 TestGetXtalTrim

Syntax

```
uint16 TestGetXtalTrim ( void )
```

Description

Read the current crystal trim value from the hardware and return the value to the caller.

Legitimate values for the xtal_trim parameter are 0x00 to 0x3F (bits 0 to 5).

This function is only for use during production test. During normal operation the crystal trim value will not change.

Parameters

Parameter	Description
`None`	-

Returns

Crystal trim value in range 0x00 to 0x3F

45.1.10 TestSetXtalTrim

Syntax

```
void TestSetXtalTrim ( uint16 xtal_trim)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

Extended test function to allow the application to adjust the 16MHz crystal trim value.
Legitimate values for the xtal_trim parameter are 0x00 to 0x3F (bits 0 to 5). Any other bits set in the xtal_trim parameter will be ignored.
This function is typically used in conjunction with TestEnableCarrierWave() to allow the crystal frequency to be measured and trimmed, to get as close to an ideal value as possible. Once the best value has been found it should be written to the device Configuration Store (using appropriate tools on the host PC).

Warning:This function must only be called during production test. Adjusting this parameter during normal operation could result in the chip operating in a non-compliant manner.

Parameters

Parameter	Description
`xtal_trim`	Crystal trim value in range 0x00 to 0x3F

Returns

Nothing

45.2 Enumerations

45.2.1 enum ble_test_pkt_type

Syntax

```
enum ble_test_pkt_type
```

Description

Supported packet payloads for BLE Transmit test.

Enumerations

Enumeration	Description
`ble_test_pkt_prbs9`	Pseudo-Random Bit Sequence 9
`ble_test_pkt_11110000`	Pattern of alternating bits '11110000'
`ble_test_pkt_10101010`	Pattern of alternating bits '10101010'
`ble_test_pkt_prbs15`	Pseudo-Random Bit Sequence 15
`ble_test_pkt_all_1`	Pattern of all '1' bits
`ble_test_pkt_all_0`	Pattern of all '0' bits
`ble_test_pkt_00001111`	Pattern of alternating bits '00001111'
`ble_test_pkt_01010101`	Pattern of alternating bits '01010101'

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

45.3 Defines

45.3.1 DirectTestTransmitCount

Definition

```
#define DirectTestTransmitCount ( ) DirectTestReceiveResults(NULL)
```

Description

Return the number of packets transmitted since the start of the test.

For now, the RX and TX counts are shared, so this function redirects to DirectTestReceiveResults(). However, this may change in future, so applications should always call DirectTestTransmitCount() to get the transmit count.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

46 Data Structures

Description

The CSR uEnergy firmware library contains the following data structures.

Data Structure	Description
att_attr_full128_t	Full attribute type with 128-bit UUID (att_type_full128)
att_attr_full_t	Full attribute type with 16-bit UUID (att_type_full)
battery_low_data	The data associated with a sys_event_battery_low event
BD_ADDR_T	Standard Bluetooth Address type
ble_con_params	Structure for common Bluetooth low energy Connection Parameters, used for new connections and connection updates
EV_MNFR_EXTN_PAYLOAD_T	Allow manufacturer's extension events to go over HCI
GATT_ACCESS_IND_T	This event requires handling by a call to GattAccessRsp() if the ATT_ACCESS_PERMISSION or ATT_ACCESS_WRITE_COMPLETE flags are set
GATT_ADD_DB_CFM_T	This event is raised after a call to GattAddDatabaseReq()
GATT_ATT_EXECUTE_WRITE_CFM_T	This event is raised after a call to GattAttExecuteWriteReq()
GATT_ATT_PREPARE_WRITE_CFM_T	This event is raised after a call to GattAttPrepareWriteReq()
GATT_CANCEL_CONNECT_CFM_T	This event is raised after a call to GattCancelConnectReq()
GATT_CHAR_DECL_INFO_IND_T	Zero or more of these events may be raised after a call to GattDiscoverServiceChar() and before the GATT_DISC_SERVICE_CHAR_CFM event
GATT_CHAR_DESC_INFO_IND_T	Zero or more of these events may be raised after a call to GattDiscoverAllCharDescriptors() and before the GATT_DISC_ALL_CHAR_DESC_CFM event
GATT_CHAR_VAL_IND_CFM_T	This event is raised after a call to GattCharValueIndication() or GattCharValueNotification()
GATT_CHAR_VAL_IND_T	Zero or more of these events may be raised after a call to GattReadCharUsingUuid() and before the GATT_READ_CHAR_USING_UUID_CFM event, or, after receiving a Characteristic Value Indication or Characteristic Value Notification from the server
GATT_CONNECT_CFM_T	This event is raised after a call to GattConnectReq()

Data Structure	Description
GATT_CONNECT_IND_T	Unsupported
GATT_DISC_ALL_CHAR_DESC_CFM_T	This event is raised after a call to GattDiscoverAllCharDescriptors() to indicate that all service charactistics have been reported by GATT_CHAR_DESC_INFO_IND events
GATT_DISC_ALL_PRIM_SERV_CFM_T	This event is raised after a call to GattDiscoverAllPrimaryServices() to indicate that all discovered services have been reported by GATT_SERV_INFO_IND events
GATT_DISC_PRIM_SERV_BY_UUID_CFM_T	This event is raised after a call to GattDiscoverPrimaryServiceByUuid() to indicate that all discovered services have been reported by GATT_DISC_PRIM_SERV_BY_UUID_IND events
GATT_DISC_PRIM_SERV_BY_UUID_IND_T	Zero or more of these events may be raised after a call to GattDiscoverPrimaryServiceByUuid() and before the GATT_DISC_PRIM_SERV_BY_UUID_CFM event
GATT_DISC_SERVICE_CHAR_CFM_T	This event is raised after a call to GattDiscoverServiceChar() to indicate that all service charactistics have been reported by GATT_CHAR_DECL_INFO_IND events
GATT_DISCONNECT_CFM_T	This event is raised after a call to GattDisconnectReq(). There may be a race with the peer disconnecting so it is recommended that you handle this event identically to GATT_DISCONNECT_IND
GATT_DISCONNECT_IND_T	This event is raised after the peer disconnects. There may be a race with a call to GattDisconnectReq() so it is recommended that you handle this event identically to GATT_DISCONNECT_IND
GATT_EXCHANGE_MTU_CFM_T	This event is raised after a call to GattExchangeMtuReq() when the MTU Exchange Procedure is finished
GATT_EXCHANGE_MTU_IND_T	This event is raised after the peer initiates a MTU Exchange Procedure. It is handled by a call to GattExchangeMtuRsp()
GATT_FIND_INCLUDED_SERV_CFM_T	This event is raised after a call to GattFindIncludedServices() to indicate that all included services have been reported by GATT_SERV_INFO_IND events
GATT_LONG_CHAR_VAL_IND_T	Zero or more of these events may be raised after a call to GattReadLongCharValue() and before the GATT_READ_LONG_CHAR_VAL_CFM event
GATT_READ_CHAR_USING_UUID_CFM_T	This event is raised after a call to GattReadCharUsingUuid() to indicate that all characteristic values have been reported by GATT_UUID_CHAR_VAL_IND events

Data Structure	Description
GATT_READ_CHAR_VAL_CFM_T	This event is raised after a call to GattReadCharValue()
GATT_READ_LONG_CHAR_VAL_CFM_T	This event is raised after a call to GattReadLongCharValue() to indicate that all value parts have been reported by GATT_LONG_CHAR_VAL_IND events
GATT_READ_MULTI_CHAR_VAL_CFM_T	This event is raised after a call to GattReadMultipleCharValues()
GATT_SERV_INFO_IND_T	Zero or more of these events may be raised after a call to GattDiscoverAllPrimaryServices() and before the GATT_DISC_ALL_PRIM_SERV_CFM event, or, after a call to GattFindIncludedServices() and before the GATT_FIND_INCLUDED_SERV_CFM event
GATT_WRITE_CHAR_VAL_CFM_T	This event is raised after a call to GattWriteCharValueReq()
GATT_WRITE_LONG_CHAR_VAL_CFM_T	This event is raised after a call to GattWriteLongCharValueReq()
HANDLE_COMPLETE_T	Variable argument field
HCI_EV_DATA_BUFFER_OVERFLOW_T	Data Buffer Overflow Event
HCI_EV_DATA_COMMAND_COMPLETE_T	Command Complete Event
HCI_EV_DATA_COMMAND_STATUS_T	Command Status Event
HCI_EV_DATA_CONN_COMPLETE_T	Connection Complete Event
HCI_EV_DATA_DATA_BUFFER_OVERFLOW_T	
HCI_EV_DATA_DISCONNECT_COMPLETE_T	Disconnection Complete Event
HCI_EV_DATA_ENCRYPTION_CHANGE_T	Encryption Change Event
HCI_EV_DATA_ENCRYPTION_KEY_REFRESH_COMPLETE_T	Encryption Key Refresh Complete Event
HCI_EV_DATA_HARDWARE_ERROR_T	
HCI_EV_DATA_MNFR_EXTENSION_T	Manufacturer-specific Event
HCI_EV_DATA_READ_REMOTE_VER_INFO_COMPLETE_T	Read Remote Version Information Complete Event
HCI_EV_DATA_ULP_ADVERTISING_REPORT_T	ULP Advertising
HCI_EV_DATA_ULP_CONNECTION_COMPLETE_T	ULP Connection Creation
HCI_EV_DATA_ULP_CONNECTION_UPDATE_COMPLETE_T	ULP Connection Update

Data Structure	Description
HCI_EV_DATA_ULP_LONG_TERM_KEY_REQUESTED_T	ULP Encryption
HCI_EV_DATA_ULP_READ_REMOTE_USED_FEATURES_COMPLETE_T	ULP Read Remote Used Features
HCI_EV_DEBUG_T	Debug Event
HCI_EV_HARDWARE_ERROR_T	Hardware Error Event
HCI_EV_ULP_ADVERTISING_REPORT_T	ULP Advertising
HCI_EV_ULP_CONNECTION_COMPLETE_T	ULP Connection Creation
HCI_EV_ULP_CONNECTION_UPDATE_COMPLETE_T	ULP Connection Update
HCI_EV_ULP_LONG_TERM_KEY_REQUESTED_T	ULP Encryption
HCI_EV_ULP_READ_REMOTE_USED_FEATURES_COMPLETE_T	ULP Read Remote Used Features
HCI_EVENT_COMMON_T	Event Packet Common Fields
HCI_ULP_EVENT_COMMON_T	Common header for ULP events
LM_EV_ACL_DATA_T	Unsupported
LM_EV_ADVERTISING_REPORT_T	This event is raised when an advertisement or scan response is received
LM_EV_BUFFER_OVERFLOW_T	Unsupported
LM_EV_COMMAND_COMPLETE_T	Unsupported
LM_EV_COMMAND_STATUS_T	Unsupported
LM_EV_CONN_COMPLETE_T	Unsupported
LM_EV_CONNECTION_COMPLETE_T	This event is raised when a new connection has has been established. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details
LM_EV_CONNECTION_UPDATE_T	This event is raised when connection update has been completed by the controller. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details
LM_EV_DISCONNECT_COMPLETE_T	This event is raised when a connection is terminated. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details
LM_EV_ENCRYPTION_CHANGE_T	This event is raised when change of encryption mode has been completed. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details

Data Structure	Description
LM_EV_ENCRYPTION_KEY_REFRESH_T	This event is raised when encryption key has been refreshed due to encryption being started or resumed. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details
LM_EV_HARDWARE_ERROR_T	This event is raised when there is a hardware failure. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details
LM_EV_LONG_TERM_KEY_REQUESTED_T	The LE Long Term Key Request event indicates that the master device is attempting to encrypt or re-encrypt the link and is requesting the Long Term Key from the Host. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details
LM_EV_MANUFACTURER_EXTENSION_T	Unsupported
LM_EV_NUMBER_COMPLETED_PACKETS_T	Unsupported
LM_EV_REMOTE_USED_FEATURES_T	This event is raised after the completion of the process of Link Manager obtaining the supported features. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details
LM_EV_REMOTE_VERSION_INFO_T	This event is raised to indicate the completion of the process of obtaining the version information. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details
LM_EVENT_COMMON_T	All LM_EVENT_T events have this common event header
LM_EVENT_T	This union is the common datatype for all events received by AppProcessLmEvent(). Events are identified by the value of the event member, usually processed via a switch() statement. The datatype documentation for each event describes the correlating value of the event member
LS_CONNECTION_PARAM_UPDATE_CFM_T	This event is raised after a call to LsSetNewConnectionParamReq() when the Connection Parameter Update procedure has finished
LS_CONNECTION_PARAM_UPDATE_IND_T	This event is raised when the Connection Parameter Update procedure initiated by a peer has finished
LS_CONNECTION_UPDATE_SIGNALLING_IND_T	This event is raised on a master after a slave initiates a Connection Parameter Update procedure. It is handled by a call to LsConnectionUpdateSignalingRsp() accepting or rejecting the proposed connection parameters
LS_DATA_RX_TIMING_IND_T	This event requires handling by a c
LS_RADIO_EVENT_IND_T	This event is raised if the application has requested notification of specific radio events for a GATT connection
pio_changed_data	The data associated with a sys_event_pio_changed event

Data Structure	Description
pio_ctrlr_data	The data associated with a sys_event_pio_ctrlr event
skm_encryption_key	Security Key Manager Encryption Key information
SM_CSRK_COUNTER_CHANGE_IND_T	Unsupported
SM_DIV_APPROVE_IND_T	This event requires handling by a call to SMDivApproval(), but the event will only be raised if the Application links SMDivApproval() otherwise an implicit 'approved' response is assumed
SM_KEY_REQUEST_IND_T	This event requires handling by a call to SMKeyRequestResponse() but will only be raised if the Application links SMKeyRequestResponse() otherwise an implicit 'none' response is assumed
SM_KEYS_IND_T	This event is raised after Bonding
SM_KEYSET_T	Security Information block
SM_LONG_TERM_KEY_IND_T	This event requires handling by a call to SMLongTermKeyRsp(), but the event will only be raised if the Application links SMLongTermKeyRsp() otherwise Long Term Key generation will be managed by the firmware Security Manager as normal
SM_LOST_BOND_IND_T	This is an information only event to application where local device is link master. The event indicates that device has lost previous bond on link with connection identifier cid. This event is generated when encryption request is rejected by peer device with error code "PIN OR key Missing". Applications are expected to handle lost bond indications on a link where local device is a slave with help of SM_KEYS_IND and keys available

CSR

Data Structure	Description
`SM_PAIRING_AUTH_IND_T`	This event requires handling by a call to SMPairingAuthRsp(), but the event will only be raised if the Application links SMPairingAuthRsp() otherwise an implicit 'authorised' response is assumed
`SM_PASSKEY_DISPLAY_IND_T`	This event is raised during Passkey Pairing and requires handling by a call to SMPasskeyDisplayed() or SMPasskeyInputNeg(). If the Application displays an alternative passkey, SMPasskeyInput() is used instead of SMPasskeyDisplayed() to inform the SM stack
`SM_PASSKEY_INPUT_IND_T`	This event is raised during Passkey Pairing and requires handling by a call to SMPasskeyInput() or SMPasskeyInputNeg()
`SM_SIMPLE_PAIRING_COMPLETE_IND_T`	This event is raised after Link Encryption or Pairing to indicate an error or the new link security level, which may be equal to or higher than the minimum security level set by GapSetMode()
`TYPED_BD_ADDR_T`	Typed Bluetooth Address type for LE
`wakeup_data`	The data associated with a sys_event_wakeup event

46.1 att_attr_full128_t Struct Reference

Structure Definition

Data	Fields
`uint32`	uuid [4]
`uint16`	perm
`uint16`	data [1]

46.1.1 uint16 att_attr_full128_t::data[1]

Syntax

```
uint16 att_attr_full128_t::data[1]
```

Description

Attribute value

46.1.2 uint16 att_attr_full128_t::perm

Syntax

```
uint16 att_attr_full128_t::perm
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Attribute permissions

46.1.3 uint32 att_attr_full128_t::uuid[4]

Syntax

```
uint32 att_attr_full128_t::uuid[4]
```

Description

UUID128

46.2 att_attr_full_t Struct Reference

Structure Definition

Data	Fields
`uint16`	uuid
`uint16`	perm
`uint16`	data [1]

46.2.1 uint16 att_attr_full_t::data[1]

Syntax

```
uint16 att_attr_full_t::data[1]
```

Description

Attribute value

46.2.2 uint16 att_attr_full_t::perm

Syntax

```
uint16 att_attr_full_t::perm
```

Description

Attribute permissions

46.2.3 uint16 att_attr_full_t::uuid

Syntax

```
uint16 att_attr_full_t::uuid
```

Description

UUID16

46.3 battery_low_data Struct Reference

Structure Definition

Data	Fields
`bool`	is_below_threshold
`uint16`	current_voltage
`uint16`	threshold_voltage

46.3.1 uint16 battery_low_data::current_voltage

Syntax

```
uint16 battery_low_data::current_voltage
```

Description

The current voltage as read from the battery, in millivolts (mV)

46.3.2 bool battery_low_data::is_below_threshold

Syntax

```
bool battery_low_data::is_below_threshold
```

Description

TRUE if the voltage has dropped below the threshold, FALSE is it has risen above the threshold.

46.3.3 uint16 battery_low_data::threshold_voltage

Syntax

```
uint16 battery_low_data::threshold_voltage
```

Description

The threshold voltage (mV) against which the battery voltage is compared. This value is taken from the Configuration Store.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.4 BD_ADDR_T Struct Reference

Structure Definition

Data	Fields
`uint24`	lap
`uint8`	uap
`uint16`	nap

46.4.1 uint24 BD_ADDR_T::lap

Syntax

```
uint24 BD_ADDR_T::lap
```

Description

Lower Address Part 00..23.

46.4.2 uint8 BD_ADDR_T::uap

Syntax

```
uint8 BD_ADDR_T::uap
```

Description

Upper Address Part 24..31.

46.4.3 uint16 BD_ADDR_T::nap

Syntax

```
uint16 BD_ADDR_T::nap
```

Description

Non-significant 32..47.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.5 ble_con_params Struct Reference

Structure Definition

Data	Fields
`uint16`	con_min_interval
`uint16`	con_max_interval
`uint16`	con_slave_latency
`uint16`	con_super_timeout

46.5.1 uint16 ble_con_params::con_max_interval

Syntax

```
uint16 ble_con_params::con_max_interval
```

Description

Maximum Connection Interval

46.5.2 uint16 ble_con_params::con_min_interval

Syntax

```
uint16 ble_con_params::con_min_interval
```

Description

Connection Interval Minimum

46.5.3 uint16 ble_con_params::con_slave_latency

Syntax

```
uint16 ble_con_params::con_slave_latency
```

Description

Slave Latency

46.5.4 uint16 ble_con_params::con_super_timeout

Syntax

```
uint16 ble_con_params::con_super_timeout
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

Supervision Timeout

46.6 EV_MNFR_EXTN_PAYLOAD_T Union Reference

46.7 GATT_ACCESS_IND_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`uint16`	cid
`uint16`	handle
`uint16`	flags
`uint16`	offset
`uint16`	size_value
`uint8 *`	value

46.7.1 uint16 GATT_ACCESS_IND_T::cid

Syntax

```
uint16 GATT_ACCESS_IND_T::cid
```

Description

Connection Identifier

46.7.2 LM_EVENT_COMMON_T GATT_ACCESS_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_ACCESS_IND_T::event
```

Description

== GATT_ACCESS_IND

46.7.3 uint16 GATT_ACCESS_IND_T::flags

Syntax

```
uint16 GATT_ACCESS_IND_T::flags
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Flags - combination of ATT_ACCESS_READ, ATT_ACCESS_WRITE, ATT_ACCESS_PERMISSION, ATT_ACCESS_WRITE_COMPLETE

46.7.4 uint16 GATT_ACCESS_IND_T::handle

Syntax

```
uint16 GATT_ACCESS_IND_T::handle
```

Description

The handle of the attribute

46.7.5 uint16 GATT_ACCESS_IND_T::offset

Syntax

```
uint16 GATT_ACCESS_IND_T::offset
```

Description

Offset of the first octet to be accessed

46.7.6 uint16 GATT_ACCESS_IND_T::size_value

Syntax

```
uint16 GATT_ACCESS_IND_T::size_value
```

Description

Length to be accessed

46.7.7 uint8* GATT_ACCESS_IND_T::value

Syntax

```
uint8* GATT_ACCESS_IND_T::value
```

Description

NULL or pointer to content proposed to be written

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.8 GATT_ADD_DB_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
sys_status	result

46.8.1 LM_EVENT_COMMON_T GATT_ADD_DB_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_ADD_DB_CFM_T::event
```

Description

== GATT_ADD_DB_CFM

46.8.2 sys_status GATT_ADD_DB_CFM_T::result

Syntax

```
sys_status GATT_ADD_DB_CFM_T::result
```

Description

sys_status_success or error

46.9 GATT_ATT_EXECUTE_WRITE_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result

46.9.1 uint16 GATT_ATT_EXECUTE_WRITE_CFM_T::cid

Syntax

```
uint16 GATT_ATT_EXECUTE_WRITE_CFM_T::cid
```

Description

Connection Identifier

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.9.2 LM_EVENT_COMMON_T GATT_ATT_EXECUTE_WRITE_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_ATT_EXECUTE_WRITE_CFM_T::event
```

Description

== ATT_ATT_EXECUTE_WRITE_CFM

46.9.3 sys_status GATT_ATT_EXECUTE_WRITE_CFM_T::result

Syntax

```
sys_status GATT_ATT_EXECUTE_WRITE_CFM_T::result
```

Description

sys_status_success or error

46.10 GATT_ATT_PREPARE_WRITE_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result
uint16	handle
uint16	offset
uint16	size_value
uint8 *	value

46.10.1 uint16 GATT_ATT_PREPARE_WRITE_CFM_T::cid

Syntax

```
uint16 GATT_ATT_PREPARE_WRITE_CFM_T::cid
```

Description

Connection Identifier

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.10.2 LM_EVENT_COMMON_T GATT_ATT_PREPARE_WRITE_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_ATT_PREPARE_WRITE_CFM_T::event
```

Description

== ATT_ATT_PREPARE_WRITE_CFM

46.10.3 uint16 GATT_ATT_PREPARE_WRITE_CFM_T::handle

Syntax

```
uint16 GATT_ATT_PREPARE_WRITE_CFM_T::handle
```

Description

The handle of the attribute

46.10.4 uint16 GATT_ATT_PREPARE_WRITE_CFM_T::offset

Syntax

```
uint16 GATT_ATT_PREPARE_WRITE_CFM_T::offset
```

Description

Offset of the first octet to be written

46.10.5 sys_status GATT_ATT_PREPARE_WRITE_CFM_T::result

Syntax

```
sys_status GATT_ATT_PREPARE_WRITE_CFM_T::result
```

Description

sys_status_success or error

46.10.6 uint16 GATT_ATT_PREPARE_WRITE_CFM_T::size_value

Syntax

```
uint16 GATT_ATT_PREPARE_WRITE_CFM_T::size_value
```

Description

Number of octets to be written

46.10.7 uint8* GATT_ATT_PREPARE_WRITE_CFM_T::value

Syntax

```
uint8* GATT_ATT_PREPARE_WRITE_CFM_T::value
```

Description

NULL or pointer to data to be written

46.11 GATT_CANCEL_CONNECT_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
sys_status	result

46.11.1 LM_EVENT_COMMON_T GATT_CANCEL_CONNECT_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_CANCEL_CONNECT_CFM_T::event
```

Description

== GATT_CANCEL_CONNECT_CFM

46.11.2 sys_status GATT_CANCEL_CONNECT_CFM_T::result

Syntax

```
sys_status GATT_CANCEL_CONNECT_CFM_T::result
```

Description

sys_status_success or error

46.12 GATT_CHAR_DECL_INFO_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
uint8	prop
uint16	val_handle
GATT_UUID_T	uuid_type
uint16	uuid [8]

46.12.1 uint16 GATT_CHAR_DECL_INFO_IND_T::cid

Syntax

```
uint16 GATT_CHAR_DECL_INFO_IND_T::cid
```

Description

Connection Identifier

46.12.2 LM_EVENT_COMMON_T GATT_CHAR_DECL_INFO_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_CHAR_DECL_INFO_IND_T::event
```

Description

== GATT_CHAR_DECL_INFO_IND

46.12.3 uint8 GATT_CHAR_DECL_INFO_IND_T::prop

Syntax

```
uint8 GATT_CHAR_DECL_INFO_IND_T::prop
```

Description

Characteristic properties

46.12.4 uint16 GATT_CHAR_DECL_INFO_IND_T::uuid[8]

Syntax

```
uint16 GATT_CHAR_DECL_INFO_IND_T::uuid[8]
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Characteristic UUID. Where the UUID type is 16-bit, only the first entry in the array is valid

46.12.5 GATT_UUID_T GATT_CHAR_DECL_INFO_IND_T::uuid_type

Syntax

```
GATT_UUID_T GATT_CHAR_DECL_INFO_IND_T::uuid_type
```

Description

UUID type - 16-bit or 128-bit

46.12.6 uint16 GATT_CHAR_DECL_INFO_IND_T::val_handle

Syntax

```
uint16 GATT_CHAR_DECL_INFO_IND_T::val_handle
```

Description

Characteristic value handle

46.13 GATT_CHAR_DESC_INFO_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
uint16	desc_handle
GATT_UUID_T	uuid_type
uint16	uuid [8]

46.13.1 uint16 GATT_CHAR_DESC_INFO_IND_T::cid

Syntax

```
uint16 GATT_CHAR_DESC_INFO_IND_T::cid
```

Description

Connection Identifier

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.13.2 uint16 GATT_CHAR_DESC_INFO_IND_T::desc_handle

Syntax

```
uint16 GATT_CHAR_DESC_INFO_IND_T::desc_handle
```

Description

Characteristic descriptor handle

46.13.3 LM_EVENT_COMMON_T GATT_CHAR_DESC_INFO_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_CHAR_DESC_INFO_IND_T::event
```

Description

== GATT_CHAR_DESC_INFO_IND

46.13.4 uint16 GATT_CHAR_DESC_INFO_IND_T::uuid[8]

Syntax

```
uint16 GATT_CHAR_DESC_INFO_IND_T::uuid[8]
```

Description

Characteristic descriptor UUID. Where the UUID type is 16-bit, only the first entry in the array is valid

46.13.5 GATT_UUID_T GATT_CHAR_DESC_INFO_IND_T::uuid_type

Syntax

```
GATT_UUID_T GATT_CHAR_DESC_INFO_IND_T::uuid_type
```

Description

UUID type - 16-bit or 128-bit

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.14 GATT_CHAR_VAL_IND_CFM_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`uint16`	cid
`sys_status`	result
`uint16`	handle

46.14.1 uint16 GATT_CHAR_VAL_IND_CFM_T::cid

Syntax

```
uint16 GATT_CHAR_VAL_IND_CFM_T::cid
```

Description

Connection Identifier

46.14.2 LM_EVENT_COMMON_T GATT_CHAR_VAL_IND_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_CHAR_VAL_IND_CFM_T::event
```

Description

== GATT_CHAR_VAL_IND_CFM or == GATT_CHAR_VAL_NOT_CFM

46.14.3 uint16 GATT_CHAR_VAL_IND_CFM_T::handle

Syntax

```
uint16 GATT_CHAR_VAL_IND_CFM_T::handle
```

Description

attribute handle from corresponding request

46.14.4 sys_status GATT_CHAR_VAL_IND_CFM_T::result

Syntax

```
sys_status GATT_CHAR_VAL_IND_CFM_T::result
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

sys_status_success or error

46.15 GATT_CHAR_VAL_IND_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`uint16`	cid
`uint16`	handle
`uint16`	size_value
`uint8 *`	value

46.15.1 uint16 GATT_CHAR_VAL_IND_T::cid

Syntax

```
uint16 GATT_CHAR_VAL_IND_T::cid
```

Description

Connection Identifier

46.15.2 LM_EVENT_COMMON_T GATT_CHAR_VAL_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_CHAR_VAL_IND_T::event
```

Description

== GATT_UUID_CHAR_VAL_IND or == GATT_NOT_CHAR_VAL_IND or == GATT_IND_CHAR_VAL_IND

46.15.3 uint16 GATT_CHAR_VAL_IND_T::handle

Syntax

```
uint16 GATT_CHAR_VAL_IND_T::handle
```

Description

Characteristic value handle

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.15.4 uint16 GATT_CHAR_VAL_IND_T::size_value

Syntax

```
uint16 GATT_CHAR_VAL_IND_T::size_value
```

Description

Characteristic value size in octets

46.15.5 uint8* GATT_CHAR_VAL_IND_T::value

Syntax

```
uint8* GATT_CHAR_VAL_IND_T::value
```

Description

Characteristic value as received

46.16 GATT_CONNECT_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
TYPED_BD_ADDR_T	bd_addr
uint16	cid
sys_status	result

46.16.1 TYPED_BD_ADDR_T GATT_CONNECT_CFM_T::bd_addr

Syntax

```
TYPED_BD_ADDR_T GATT_CONNECT_CFM_T::bd_addr
```

Description

address of peer

46.16.2 uint16 GATT_CONNECT_CFM_T::cid

Syntax

```
uint16 GATT_CONNECT_CFM_T::cid
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Connection Identifier if result == sys_status_success

46.16.3 LM_EVENT_COMMON_T GATT_CONNECT_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_CONNECT_CFM_T::event
```

Description

== GATT_CONNECT_CFM

46.16.4 sys_status GATT_CONNECT_CFM_T::result

Syntax

```
sys_status GATT_CONNECT_CFM_T::result
```

Description

sys_status_success or error

46.17 GATT_CONNECT_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
TYPED_BD_ADDR_T	bd_addr
uint16	cid

46.17.1 TYPED_BD_ADDR_T GATT_CONNECT_IND_T::bd_addr

Syntax

```
TYPED_BD_ADDR_T GATT_CONNECT_IND_T::bd_addr
```

Description

address of peer

46.17.2 uint16 GATT_CONNECT_IND_T::cid

Syntax

```
uint16 GATT_CONNECT_IND_T::cid
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Connection Identifier

46.17.3 LM_EVENT_COMMON_T GATT_CONNECT_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_CONNECT_IND_T::event
```

Description

== GATT_CONNECT_IND

46.18 GATT_DISCONNECT_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result

46.18.1 uint16 GATT_DISCONNECT_CFM_T::cid

Syntax

```
uint16 GATT_DISCONNECT_CFM_T::cid
```

Description

Connection Identifier

46.18.2 LM_EVENT_COMMON_T GATT_DISCONNECT_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_DISCONNECT_CFM_T::event
```

Description

== GATT_DISCONNECT_CFM

46.18.3 sys_status GATT_DISCONNECT_CFM_T::result

Syntax

```
sys_status GATT_DISCONNECT_CFM_T::result
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

sys_status_success or error

46.19 GATT_DISCONNECT_IND_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`uint16`	cid
`sys_status`	reason

46.19.1 uint16 GATT_DISCONNECT_IND_T::cid

Syntax

```
uint16 GATT_DISCONNECT_IND_T::cid
```

Description

Connection Identifier

46.19.2 LM_EVENT_COMMON_T GATT_DISCONNECT_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_DISCONNECT_IND_T::event
```

Description

== GATT_DISCONNECT_IND

46.19.3 sys_status GATT_DISCONNECT_IND_T::reason

Syntax

```
sys_status GATT_DISCONNECT_IND_T::reason
```

Description

ls_err_oetc_user or error

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.20 GATT_DISC_ALL_CHAR_DESC_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result

46.20.1 uint16 GATT_DISC_ALL_CHAR_DESC_CFM_T::cid

Syntax

```
uint16 GATT_DISC_ALL_CHAR_DESC_CFM_T::cid
```

Description

Connection Identifier

46.20.2 LM_EVENT_COMMON_T GATT_DISC_ALL_CHAR_DESC_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_DISC_ALL_CHAR_DESC_CFM_T::event
```

Description

== GATT_DISC_ALL_CHAR_DESC_CFM

46.20.3 sys_status GATT_DISC_ALL_CHAR_DESC_CFM_T::result

Syntax

```
sys_status GATT_DISC_ALL_CHAR_DESC_CFM_T::result
```

Description

sys_status_success or error

46.21 GATT_DISC_ALL_PRIM_SERV_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.21.1 uint16 GATT_DISC_ALL_PRIM_SERV_CFM_T::cid

Syntax

```
uint16 GATT_DISC_ALL_PRIM_SERV_CFM_T::cid
```

Description

Connection Identifier

46.21.2 LM_EVENT_COMMON_T GATT_DISC_ALL_PRIM_SERV_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_DISC_ALL_PRIM_SERV_CFM_T::event
```

Description

== GATT_DISC_ALL_PRIM_SERV_CFM

46.21.3 sys_status GATT_DISC_ALL_PRIM_SERV_CFM_T::result

Syntax

```
sys_status GATT_DISC_ALL_PRIM_SERV_CFM_T::result
```

Description

sys_status_success or error

46.22 GATT_DISC_PRIM_SERV_BY_UUID_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result

46.22.1 uint16 GATT_DISC_PRIM_SERV_BY_UUID_CFM_T::cid

Syntax

```
uint16 GATT_DISC_PRIM_SERV_BY_UUID_CFM_T::cid
```

Description

Connection Identifier

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

46.22.2 LM_EVENT_COMMON_T GATT_DISC_PRIM_SERV_BY_UUID_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_DISC_PRIM_SERV_BY_UUID_CFM_T::event
```

Description

== GATT_DISC_PRIM_SERV_BY_UUID_CFM

46.22.3 sys_status GATT_DISC_PRIM_SERV_BY_UUID_CFM_T::result

Syntax

```
sys_status GATT_DISC_PRIM_SERV_BY_UUID_CFM_T::result
```

Description

sys_status_success or error

46.23 GATT_DISC_PRIM_SERV_BY_UUID_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
uint16	strt_handle
uint16	end_handle

46.23.1 uint16 GATT_DISC_PRIM_SERV_BY_UUID_IND_T::cid

Syntax

```
uint16 GATT_DISC_PRIM_SERV_BY_UUID_IND_T::cid
```

Description

Connection Identifier

46.23.2 uint16 GATT_DISC_PRIM_SERV_BY_UUID_IND_T::end_handle

Syntax

```
uint16 GATT_DISC_PRIM_SERV_BY_UUID_IND_T::end_handle
```

Description

End handle for the service

46.23.3 LM_EVENT_COMMON_T GATT_DISC_PRIM_SERV_BY_UUID_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_DISC_PRIM_SERV_BY_UUID_IND_T::event
```

Description

== GATT_DISC_PRIM_SERV_BY_UUID_IND

46.23.4 uint16 GATT_DISC_PRIM_SERV_BY_UUID_IND_T::strt_handle

Syntax

```
uint16 GATT_DISC_PRIM_SERV_BY_UUID_IND_T::strt_handle
```

Description

Start handle for the service

46.24 GATT_DISC_SERVICE_CHAR_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result

46.24.1 uint16 GATT_DISC_SERVICE_CHAR_CFM_T::cid

Syntax

```
uint16 GATT_DISC_SERVICE_CHAR_CFM_T::cid
```

Description

Connection Identifier

46.24.2 LM_EVENT_COMMON_T GATT_DISC_SERVICE_CHAR_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_DISC_SERVICE_CHAR_CFM_T::event
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

== GATT_DISC_SERVICE_CHAR_CFM

46.24.3 sys_status GATT_DISC_SERVICE_CHAR_CFM_T::result

Syntax

```
sys_status GATT_DISC_SERVICE_CHAR_CFM_T::result
```

Description

sys_status_success or error

46.25 GATT_EXCHANGE_MTU_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result
uint16	mtu

46.25.1 uint16 GATT_EXCHANGE_MTU_CFM_T::cid

Syntax

```
uint16 GATT_EXCHANGE_MTU_CFM_T::cid
```

Description

Connection Identifier

46.25.2 LM_EVENT_COMMON_T GATT_EXCHANGE_MTU_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_EXCHANGE_MTU_CFM_T::event
```

Description

== GATT_EXCHANGE_MTU_CFM

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.25.3 uint16 GATT_EXCHANGE_MTU_CFM_T::mtu

Syntax

```
uint16 GATT_EXCHANGE_MTU_CFM_T::mtu
```

Description

Negotiated MTU for the connection

46.25.4 sys_status GATT_EXCHANGE_MTU_CFM_T::result

Syntax

```
sys_status GATT_EXCHANGE_MTU_CFM_T::result
```

Description

sys_status_success or error

46.26 GATT_EXCHANGE_MTU_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
uint16	client_mtu

46.26.1 uint16 GATT_EXCHANGE_MTU_IND_T::cid

Syntax

```
uint16 GATT_EXCHANGE_MTU_IND_T::cid
```

Description

Connection Identifier

46.26.2 uint16 GATT_EXCHANGE_MTU_IND_T::client_mtu

Syntax

```
uint16 GATT_EXCHANGE_MTU_IND_T::client_mtu
```

Description

Max MTU support by peer

CSR

46.26.3 LM_EVENT_COMMON_T GATT_EXCHANGE_MTU_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_EXCHANGE_MTU_IND_T::event
```

Description

== GATT_EXCHANGE_MTU_IND

46.27 GATT_FIND_INCLUDED_SERV_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result

46.27.1 uint16 GATT_FIND_INCLUDED_SERV_CFM_T::cid

Syntax

```
uint16 GATT_FIND_INCLUDED_SERV_CFM_T::cid
```

Description

Connection Identifier

46.27.2 LM_EVENT_COMMON_T GATT_FIND_INCLUDED_SERV_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_FIND_INCLUDED_SERV_CFM_T::event
```

Description

== GATT_FIND_INCLUDED_SERV_CFM

46.27.3 sys_status GATT_FIND_INCLUDED_SERV_CFM_T::result

Syntax

```
sys_status GATT_FIND_INCLUDED_SERV_CFM_T::result
```

Description

sys_status_success or error

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.28 GATT_LONG_CHAR_VAL_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
uint16	offset
uint16	size_value
uint8 *	value

46.28.1 uint16 GATT_LONG_CHAR_VAL_IND_T::cid

Syntax

```
uint16 GATT_LONG_CHAR_VAL_IND_T::cid
```

Description

Connection Identifier

46.28.2 LM_EVENT_COMMON_T GATT_LONG_CHAR_VAL_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_LONG_CHAR_VAL_IND_T::event
```

Description

== GATT_LONG_CHAR_VAL_IND

46.28.3 uint16 GATT_LONG_CHAR_VAL_IND_T::offset

Syntax

```
uint16 GATT_LONG_CHAR_VAL_IND_T::offset
```

Description

Offset of this part in the characteristic value

46.28.4 uint16 GATT_LONG_CHAR_VAL_IND_T::size_value

Syntax

```
uint16 GATT_LONG_CHAR_VAL_IND_T::size_value
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Characteristic part value size in octets

46.28.5 uint8* GATT_LONG_CHAR_VAL_IND_T::value

Syntax

```
uint8* GATT_LONG_CHAR_VAL_IND_T::value
```

Description

Characteristic part value as received

46.29 GATT_READ_CHAR_USING_UUID_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result

46.29.1 uint16 GATT_READ_CHAR_USING_UUID_CFM_T::cid

Syntax

```
uint16 GATT_READ_CHAR_USING_UUID_CFM_T::cid
```

Description

Connection Identifier

46.29.2 LM_EVENT_COMMON_T GATT_READ_CHAR_USING_UUID_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_READ_CHAR_USING_UUID_CFM_T::event
```

Description

== GATT_READ_CHAR_USING_UUID_CFM

46.29.3 sys_status GATT_READ_CHAR_USING_UUID_CFM_T::result

Syntax

```
sys_status GATT_READ_CHAR_USING_UUID_CFM_T::result
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

sys_status_success or error

46.30 GATT_READ_CHAR_VAL_CFM_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`uint16`	cid
`sys_status`	result
`uint16`	size_value
`uint8 *`	value

46.30.1 uint16 GATT_READ_CHAR_VAL_CFM_T::cid

Syntax

```
uint16 GATT_READ_CHAR_VAL_CFM_T::cid
```

Description

Connection Identifier

46.30.2 LM_EVENT_COMMON_T GATT_READ_CHAR_VAL_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_READ_CHAR_VAL_CFM_T::event
```

Description

== GATT_READ_CHAR_VAL_CFM

46.30.3 sys_status GATT_READ_CHAR_VAL_CFM_T::result

Syntax

```
sys_status GATT_READ_CHAR_VAL_CFM_T::result
```

Description

sys_status_success or error

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.30.4 uint16 GATT_READ_CHAR_VAL_CFM_T::size_value

Syntax

```
uint16 GATT_READ_CHAR_VAL_CFM_T::size_value
```

Description

Characteristic value size in octets

46.30.5 uint8* GATT_READ_CHAR_VAL_CFM_T::value

Syntax

```
uint8* GATT_READ_CHAR_VAL_CFM_T::value
```

Description

Characteristic value as received

46.31 GATT_READ_LONG_CHAR_VAL_CFM_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`uint16`	cid
`sys_status`	result

46.31.1 uint16 GATT_READ_LONG_CHAR_VAL_CFM_T::cid

Syntax

```
uint16 GATT_READ_LONG_CHAR_VAL_CFM_T::cid
```

Description

Connection Identifier

46.31.2 LM_EVENT_COMMON_T GATT_READ_LONG_CHAR_VAL_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_READ_LONG_CHAR_VAL_CFM_T::event
```

Description

== GATT_READ_LONG_CHAR_VAL_CFM

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

46.31.3 sys_status GATT_READ_LONG_CHAR_VAL_CFM_T::result

Syntax

```
sys_status GATT_READ_LONG_CHAR_VAL_CFM_T::result
```

Description

sys_status_success or error

46.32 GATT_READ_MULTI_CHAR_VAL_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
sys_status	result
uint16	size_value
uint8 *	value

46.32.1 uint16 GATT_READ_MULTI_CHAR_VAL_CFM_T::cid

Syntax

```
uint16 GATT_READ_MULTI_CHAR_VAL_CFM_T::cid
```

Description

Connection Identifier

46.32.2 LM_EVENT_COMMON_T GATT_READ_MULTI_CHAR_VAL_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_READ_MULTI_CHAR_VAL_CFM_T::event
```

Description

== GATT_READ_MULTI_CHAR_VAL_CFM

46.32.3 sys_status GATT_READ_MULTI_CHAR_VAL_CFM_T::result

Syntax

```
sys_status GATT_READ_MULTI_CHAR_VAL_CFM_T::result
```

Description

sys_status_success or error

46.32.4 uint16 GATT_READ_MULTI_CHAR_VAL_CFM_T::size_value

Syntax

```
uint16 GATT_READ_MULTI_CHAR_VAL_CFM_T::size_value
```

Description

Characteristic multi value size in octets

46.32.5 uint8* GATT_READ_MULTI_CHAR_VAL_CFM_T::value

Syntax

```
uint8* GATT_READ_MULTI_CHAR_VAL_CFM_T::value
```

Description

Characteristic value for multiple handles

46.33 GATT_SERV_INFO_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
uint16	strt_handle
uint16	end_handle
GATT_UUID_T	uuid_type
uint16	uuid [8]

46.33.1 uint16 GATT_SERV_INFO_IND_T::cid

Syntax

```
uint16 GATT_SERV_INFO_IND_T::cid
```

Description

Connection Identifier

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.33.2 uint16 GATT_SERV_INFO_IND_T::end_handle

Syntax

```
uint16 GATT_SERV_INFO_IND_T::end_handle
```

Description

End handle for the service

46.33.3 LM_EVENT_COMMON_T GATT_SERV_INFO_IND_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_SERV_INFO_IND_T::event
```

Description

== GATT_SERV_INFO_IND

46.33.4 uint16 GATT_SERV_INFO_IND_T::strt_handle

Syntax

```
uint16 GATT_SERV_INFO_IND_T::strt_handle
```

Description

Start handle for the service

46.33.5 uint16 GATT_SERV_INFO_IND_T::uuid[8]

Syntax

```
uint16 GATT_SERV_INFO_IND_T::uuid[8]
```

Description

Service UUID. Where the UUID type is 16-bit, only the first entry in the array is valid

46.33.6 GATT_UUID_T GATT_SERV_INFO_IND_T::uuid_type

Syntax

```
GATT_UUID_T GATT_SERV_INFO_IND_T::uuid_type
```

Description

UUID type - 16-bit or 128-bit

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

46.34 GATT_WRITE_CHAR_VAL_CFM_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`uint16`	cid
`sys_status`	result

46.34.1 uint16 GATT_WRITE_CHAR_VAL_CFM_T::cid

Syntax

```
uint16 GATT_WRITE_CHAR_VAL_CFM_T::cid
```

Description

Connection Identifier

46.34.2 LM_EVENT_COMMON_T GATT_WRITE_CHAR_VAL_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_WRITE_CHAR_VAL_CFM_T::event
```

Description

== GATT_WRITE_CHAR_VAL_CFM

46.34.3 sys_status GATT_WRITE_CHAR_VAL_CFM_T::result

Syntax

```
sys_status GATT_WRITE_CHAR_VAL_CFM_T::result
```

Description

sys_status_success or error

46.35 GATT_WRITE_LONG_CHAR_VAL_CFM_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`uint16`	cid
`sys_status`	result

46.35.1 uint16 GATT_WRITE_LONG_CHAR_VAL_CFM_T::cid

Syntax

```
uint16 GATT_WRITE_LONG_CHAR_VAL_CFM_T::cid
```

Description

Connection Identifier

46.35.2 LM_EVENT_COMMON_T GATT_WRITE_LONG_CHAR_VAL_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T GATT_WRITE_LONG_CHAR_VAL_CFM_T::event
```

Description

== GATT_WRITE_LONG_CHAR_VAL_CFM

46.35.3 sys_status GATT_WRITE_LONG_CHAR_VAL_CFM_T::result

Syntax

```
sys_status GATT_WRITE_LONG_CHAR_VAL_CFM_T::result
```

Description

sys_status_success or error

46.36 LM_EVENT_COMMON_T Struct Reference

Structure Definition

Data	Fields
lm_event_code	event_code

46.36.1 lm_event_code LM_EVENT_COMMON_T::event_code

Syntax

```
lm_event_code LM_EVENT_COMMON_T::event_code
```

Description

Identifies the type of event and hence the corresponding member of LM_EVENT_T union

46.37 LM_EV_ADVERTISING_REPORT_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`HCI_EV_DATA_ULP_ADVERTISING_REPORT_T`	data
`int8`	rssi

46.37.1 HCI_EV_DATA_ULP_ADVERTISING_REPORT_T LM_EV_ADVERTISING_REPORT_T::data

Syntax

```
HCI_EV_DATA_ULP_ADVERTISING_REPORT_T LM_EV_ADVERTISING_REPORT_T::data
```

Description

46.37.2 LM_EVENT_COMMON_T LM_EV_ADVERTISING_REPORT_T::event

Syntax

```
LM_EVENT_COMMON_T LM_EV_ADVERTISING_REPORT_T::event
```

Description

== LM_EV_ADVERTISING_REPORT

46.37.3 int8 LM_EV_ADVERTISING_REPORT_T::rssi

Syntax

```
int8 LM_EV_ADVERTISING_REPORT_T::rssi
```

Description

RSSI in signed 8-bit format. Note that the sign bit is bit 7 and bits 15..8 should mimic bit 7 following sign extension applied in dbase_get_rssi().

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

46.38 LS_CONNECTION_PARAM_UPDATE_CFM_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
sys_status	status
TYPED_BD_ADDR_T	address

46.38.1 TYPED_BD_ADDR_T LS_CONNECTION_PARAM_UPDATE_CFM_T::address

Syntax

```
TYPED_BD_ADDR_T LS_CONNECTION_PARAM_UPDATE_CFM_T::address
```

Description

address of peer

46.38.2 LM_EVENT_COMMON_T LS_CONNECTION_PARAM_UPDATE_CFM_T::event

Syntax

```
LM_EVENT_COMMON_T LS_CONNECTION_PARAM_UPDATE_CFM_T::event
```

Description

== LS_CONNECTION_PARAM_UPDATE_CFM

46.38.3 sys_status LS_CONNECTION_PARAM_UPDATE_CFM_T::status

Syntax

```
sys_status LS_CONNECTION_PARAM_UPDATE_CFM_T::status
```

Description

sys_status_success or error

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.39 LS_CONNECTION_PARAM_UPDATE_IND_T Struct Reference

Structure Definition

Data	Fields
`LM_EVENT_COMMON_T`	event
`sys_status`	status
`TYPED_BD_ADDR_T`	address
`uint16`	conn_interval
`uint16`	conn_latency
`uint16`	supervision_timeout

46.39.1 TYPED_BD_ADDR_T LS_CONNECTION_PARAM_UPDATE_IND_T::address

Syntax

```
TYPED_BD_ADDR_T LS_CONNECTION_PARAM_UPDATE_IND_T::address
```

Description

address of peer

46.39.2 uint16 LS_CONNECTION_PARAM_UPDATE_IND_T::conn_interval

Syntax

```
uint16 LS_CONNECTION_PARAM_UPDATE_IND_T::conn_interval
```

Description

Negotiated connection interval

46.39.3 uint16 LS_CONNECTION_PARAM_UPDATE_IND_T::conn_latency

Syntax

```
uint16 LS_CONNECTION_PARAM_UPDATE_IND_T::conn_latency
```

Description

Negotiated connection latency

46.39.4 LM_EVENT_COMMON_T LS_CONNECTION_PARAM_UPDATE_IND_T::event

Syntax

```
LM_EVENT_COMMON_T LS_CONNECTION_PARAM_UPDATE_IND_T::event
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

== LS_CONNECTION_PARAM_UPDATE_IND

46.39.5 sys_status LS_CONNECTION_PARAM_UPDATE_IND_T::status

Syntax

```
sys_status LS_CONNECTION_PARAM_UPDATE_IND_T::status
```

Description

sys_status_success or error

46.39.6 uint16 LS_CONNECTION_PARAM_UPDATE_IND_T::supervision_timeout

Syntax

```
uint16 LS_CONNECTION_PARAM_UPDATE_IND_T::supervision_timeout
```

Description

Negotiated supervision timeout

46.40 LS_CONNECTION_UPDATE_SIGNALLING_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	con_handle
uint16	sig_identifier
uint16	conn_interval_min
uint16	conn_interval_max
uint16	slave_latency
uint16	supervision_timeout

46.40.1 uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::con_handle

Syntax

```
uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::con_handle
```

Description

To be passed into LsConnectionUpdateSignallingRsp()

46.40.2 uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::conn_interval_min

Syntax

```
uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::conn_interval_min
```

Description

Proposed minimum connection interval

46.40.3 uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::conn_interval_min

Syntax

```
uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::conn_interval_min
```

Description

Proposed minimum connection interval

46.40.4 LM_EVENT_COMMON_T LS_CONNECTION_UPDATE_SIGNALLING_IND_T::event

Syntax

```
LM_EVENT_COMMON_T LS_CONNECTION_UPDATE_SIGNALLING_IND_T::event
```

Description

== LS_CONNECTION_UPDATE_SIGNALLING_IND

46.40.5 uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::sig_identifier

Syntax

```
uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::sig_identifier
```

Description

To be passed into LsConnectionUpdateSignallingRsp()

46.40.6 uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::slave_latency

Syntax

```
uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::slave_latency
```

CSR

Description

Proposed slave latency

46.40.7 uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::supervision_timeout

Syntax

```
uint16 LS_CONNECTION_UPDATE_SIGNALLING_IND_T::supervision_timeout
```

Description

Proposed supervision timeout

46.41 LS_DATA_RX_TIMING_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
uint16	tx_duration
time48	tx_event_offset
time48	tx_transmit_offset

46.41.1 uint16 LS_DATA_RX_TIMING_IND_T::cid

Syntax

```
uint16 LS_DATA_RX_TIMING_IND_T::cid
```

Description

Connection Identifier

46.41.2 LM_EVENT_COMMON_T LS_DATA_RX_TIMING_IND_T::event

Syntax

```
LM_EVENT_COMMON_T LS_DATA_RX_TIMING_IND_T::event
```

Description

== LS_DATA_RX_TIMING_IND

46.41.3 uint16 LS_DATA_RX_TIMING_IND_T::tx_duration

Syntax

```
uint16 LS_DATA_RX_TIMING_IND_T::tx_duration
```

Description

Duration of most recent TX preceding RX packet

46.41.4 time48 LS_DATA_RX_TIMING_IND_T::tx_event_offset

Syntax

```
time48 LS_DATA_RX_TIMING_IND_T::tx_event_offset
```

Description

Offset of most recent TX within Connection Event

46.41.5 time48 LS_DATA_RX_TIMING_IND_T::tx_transmit_offset

Syntax

```
time48 LS_DATA_RX_TIMING_IND_T::tx_transmit_offset
```

Description

Offset of most recent TX from first TX opportunity

46.42 LS_RADIO_EVENT_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
radio_event	radio

46.42.1 uint16 LS_RADIO_EVENT_IND_T::cid

Syntax

```
uint16 LS_RADIO_EVENT_IND_T::cid
```

Description

Connection Identifier

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.42.2 LM_EVENT_COMMON_T LS_RADIO_EVENT_IND_T::event

Syntax

```
LM_EVENT_COMMON_T LS_RADIO_EVENT_IND_T::event
```

Description

== LS_RADIO_EVENT_IND

46.42.3 radio_event LS_RADIO_EVENT_IND_T::radio

Syntax

```
radio_event LS_RADIO_EVENT_IND_T::radio
```

Description

The radio event that occurred

46.43 pio_changed_data Struct Reference

Structure Definition

Data	Fields
uint32	pio_state
uint32	pio_cause

46.43.1 uint32 pio_changed_data::pio_cause

Syntax

```
uint32 pio_changed_data::pio_cause
```

Description

The PIO event(s) that caused the event to be sent. One or more of these bits may be set depending on how rapidly PIOs are changing. A bit is '1' if the corresponding PIO changed state.

46.43.2 uint32 pio_changed_data::pio_state

Syntax

```
uint32 pio_changed_data::pio_state
```

Description

The state of the PIOs at the time the event was processed

46.44 pio_ctrlr_data Struct Reference

Structure Definition

Data	Fields
`uint16 *`	pio_ctrlr_data_word

46.44.1 uint16* pio_ctrlr_data::pio_ctrlr_data_word

Syntax

```
uint16* pio_ctrlr_data::pio_ctrlr_data_word
```

Description

Pointer to the PIO Controller data word

46.45 skm_encryption_key Struct Reference

Structure Definition

Data	Fields
`TYPED_BD_ADDR_T`	bd_addr
`uint16`	flags
`uint16`	ltk [8]
`uint16`	ediv
`uint16`	rand [4]

46.45.1 TYPED_BD_ADDR_T skm_encryption_key::bd_addr

Syntax

```
TYPED_BD_ADDR_T skm_encryption_key::bd_addr
```

Description

Bluetooth Device Address

46.45.2 uint16 skm_encryption_key::ediv

Syntax

```
uint16 skm_encryption_key::ediv
```

Description

Encrypted Diversifier

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.45.3 uint16 skm_encryption_key::flags

Syntax

```
uint16 skm_encryption_key::flags
```

Description

Internal flags

46.45.4 uint16 skm_encryption_key::ltk[8]

Syntax

```
uint16 skm_encryption_key::ltk[8]
```

Description

Long-Term Key

46.45.5 uint16 skm_encryption_key::rand[4]

Syntax

```
uint16 skm_encryption_key::rand[4]
```

Description

Random Number

46.46 SM_DIV_APPROVE_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
uint16	div

46.46.1 uint16 SM_DIV_APPROVE_IND_T::cid

Syntax

```
uint16 SM_DIV_APPROVE_IND_T::cid
```

Description

Connection Identifier

CSR

46.46.2 LM_EVENT_COMMON_T SM_DIV_APPROVE_IND_T::event

Syntax

```
LM_EVENT_COMMON_T SM_DIV_APPROVE_IND_T::event
```

Description

== SM_DIV_APPROVE_IND

46.47 SM_KEYSET_T Struct Reference

Structure Definition

Data	Fields
uint16	keys_present
uint16	encryption_key_size
uint16	div
uint16	ediv
uint16	rand [4]
uint16	ltk [8]
uint16	irk [8]
uint16	csrk [8]
uint16	sign_counter
TYPED_BD_ADDR_T	id_addr

46.47.1 uint16 SM_KEYSET_T::csrk[8]

Syntax

```
uint16 SM_KEYSET_T::csrk[8]
```

Description

reserved

46.47.2 uint16 SM_KEYSET_T::div

Syntax

```
uint16 SM_KEYSET_T::div
```

Description

Local Encryption DIV

46.47.3 uint16 SM_KEYSET_T::ediv

Syntax

```
uint16 SM_KEYSET_T::ediv
```

Description

Peer Encryption EDIV

46.47.4 uint16 SM_KEYSET_T::encryption_key_size

Syntax

```
uint16 SM_KEYSET_T::encryption_key_size
```

Description

Negotiated Encryption Key Size

46.47.5 TYPED_BD_ADDR_T SM_KEYSET_T::id_addr

Syntax

```
TYPED_BD_ADDR_T SM_KEYSET_T::id_addr
```

Description

Peer Public/Static Address ID

46.47.6 uint16 SM_KEYSET_T::irk[8]

Syntax

```
uint16 SM_KEYSET_T::irk[8]
```

Description

Peer Private Address Resolution IRK (Identity Resolving Key)
The IRK is stored as word wise little endian. I.e. irk[0] = Least Significant Word, irk[7] = Most Significant Word.
Example: The irk 0x000102030405060708090a0b0c0d0e0f (MSB -> LSB) is stored

```
uint16 irk[] = {0x0e0f, 0x0c0d, 0x0a0b, 0x0809, 0x0607, 0x0405, 0x0203, 0x0001};
```

46.47.7 uint16 SM_KEYSET_T::keys_present

Syntax

```
uint16 SM_KEYSET_T::keys_present
```

Description

Bits (1<<sm_key_type) set indicate valid fields

46.47.8 uint16 SM_KEYSET_T::ltk[8]

Syntax

```
uint16 SM_KEYSET_T::ltk[8]
```

Description

Peer Encryption Long Term Key (LTK)
The LTK is stored as word wise little endian. I.e. ltk[0] = Least Significant Word, ltk[7] = Most Significant Word.
Example: The ltk 0x000102030405060708090a0b0c0d0e0f (MSB -> LSB) is stored

```
uint16 ltk[] = {0x0e0f, 0x0c0d, 0x0a0b, 0x0809, 0x0607, 0x0405, 0x0203, 0x0001};
```

46.47.9 uint16 SM_KEYSET_T::rand[4]

Syntax

```
uint16 SM_KEYSET_T::rand[4]
```

Description

Peer Encryption RAND

46.47.10 uint16 SM_KEYSET_T::sign_counter

Syntax

```
uint16 SM_KEYSET_T::sign_counter
```

Description

reserved

46.48 SM_KEYS_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
TYPED_BD_ADDR_T	remote_addr
const SM_KEYSET_T *	keys

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

46.48.1 LM_EVENT_COMMON_T SM_KEYS_IND_T::event

Syntax

```
LM_EVENT_COMMON_T SM_KEYS_IND_T::event
```

Description

== SM_KEYS_IND

46.48.2 const SM_KEYSET_T* SM_KEYS_IND_T::keys

Syntax

```
const SM_KEYSET_T* SM_KEYS_IND_T::keys
```

Description

Pointer to security information block

46.48.3 TYPED_BD_ADDR_T SM_KEYS_IND_T::remote_addr

Syntax

```
TYPED_BD_ADDR_T SM_KEYS_IND_T::remote_addr
```

Description

Current, possibly private, address of peer

46.49 SM_KEY_REQUEST_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
TYPED_BD_ADDR_T	remote_addr

46.49.1 LM_EVENT_COMMON_T SM_KEY_REQUEST_IND_T::event

Syntax

```
LM_EVENT_COMMON_T SM_KEY_REQUEST_IND_T::event
```

Description

== SM_KEY_REQUEST_IND

46.49.2 TYPED_BD_ADDR_T SM_KEY_REQUEST_IND_T::remote_addr

Syntax

```
TYPED_BD_ADDR_T SM_KEY_REQUEST_IND_T::remote_addr
```

Description

Current, possibly private, address of peer

46.50 SM_LONG_TERM_KEY_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid
uint16	ediv
uint16	rand [4]

46.50.1 uint16 SM_LONG_TERM_KEY_IND_T::cid

Syntax

```
uint16 SM_LONG_TERM_KEY_IND_T::cid
```

Description

Connection Identifier

46.50.2 LM_EVENT_COMMON_T SM_LONG_TERM_KEY_IND_T::event

Syntax

```
LM_EVENT_COMMON_T SM_LONG_TERM_KEY_IND_T::event
```

Description

== SM_DIV_APPROVE_IND

46.50.3 uint16 SM_LONG_TERM_KEY_IND_T::rand[4]

Syntax

```
uint16 SM_LONG_TERM_KEY_IND_T::rand[4]
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Random Number from master

46.51 SM_LOST_BOND_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint16	cid

46.51.1 uint16 SM_LOST_BOND_IND_T::cid

Syntax

```
uint16 SM_LOST_BOND_IND_T::cid
```

Description

Connection Identifier

46.51.2 LM_EVENT_COMMON_T SM_LOST_BOND_IND_T::event

Syntax

```
LM_EVENT_COMMON_T SM_LOST_BOND_IND_T::event
```

Description

== SM_LOST_BOND_IND

46.52 SM_PAIRING_AUTH_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
uint8	type
void *	data

46.52.1 void* SM_PAIRING_AUTH_IND_T::data

Syntax

```
void* SM_PAIRING_AUTH_IND_T::data
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Internal handle for SM procedure

46.52.2 LM_EVENT_COMMON_T SM_PAIRING_AUTH_IND_T::event

Syntax

```
LM_EVENT_COMMON_T SM_PAIRING_AUTH_IND_T::event
```

Description

== SM_PAIRING_AUTH_IND

46.52.3 uint8 SM_PAIRING_AUTH_IND_T::type

Syntax

```
uint8 SM_PAIRING_AUTH_IND_T::type
```

Description

Type of pairing request

46.53 SM_PASSKEY_DISPLAY_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
TYPED_BD_ADDR_T	bd_addr
uint32	passkey

46.53.1 TYPED_BD_ADDR_T SM_PASSKEY_DISPLAY_IND_T::bd_addr

Syntax

```
TYPED_BD_ADDR_T SM_PASSKEY_DISPLAY_IND_T::bd_addr
```

Description

of the remote device

46.53.2 LM_EVENT_COMMON_T SM_PASSKEY_DISPLAY_IND_T::event

Syntax

```
LM_EVENT_COMMON_T SM_PASSKEY_DISPLAY_IND_T::event
```

Description

== SM_PASSKEY_DISPLAY_IND

46.53.3 uint32 SM_PASSKEY_DISPLAY_IND_T::passkey

Syntax

```
uint32 SM_PASSKEY_DISPLAY_IND_T::passkey
```

Description

Passkey to be displayed

46.54 SM_PASSKEY_INPUT_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
TYPED_BD_ADDR_T	bd_addr

46.54.1 TYPED_BD_ADDR_T SM_PASSKEY_INPUT_IND_T::bd_addr

Syntax

```
TYPED_BD_ADDR_T SM_PASSKEY_INPUT_IND_T::bd_addr
```

Description

of the remote device

46.54.2 LM_EVENT_COMMON_T SM_PASSKEY_INPUT_IND_T::event

Syntax

```
LM_EVENT_COMMON_T SM_PASSKEY_INPUT_IND_T::event
```

Description

== SM_PASSKEY_INPUT_IND

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

46.55 SM_SIMPLE_PAIRING_COMPLETE_IND_T Struct Reference

Structure Definition

Data	Fields
LM_EVENT_COMMON_T	event
TYPED_BD_ADDR_T	bd_addr
sys_status	status
uint16	flags
gap_mode_security	security_level

46.55.1 TYPED_BD_ADDR_T SM_SIMPLE_PAIRING_COMPLETE_IND_T::bd_addr

Syntax

```
TYPED_BD_ADDR_T SM_SIMPLE_PAIRING_COMPLETE_IND_T::bd_addr
```

Description

of the remote device

46.55.2 LM_EVENT_COMMON_T SM_SIMPLE_PAIRING_COMPLETE_IND_T::event

Syntax

```
LM_EVENT_COMMON_T SM_SIMPLE_PAIRING_COMPLETE_IND_T::event
```

Description

== SM_SIMPLE_PAIRING_COMPLETE_IND

46.55.3 uint16 SM_SIMPLE_PAIRING_COMPLETE_IND_T::flags

Syntax

```
uint16 SM_SIMPLE_PAIRING_COMPLETE_IND_T::flags
```

Description

reserved

46.55.4 gap_mode_security SM_SIMPLE_PAIRING_COMPLETE_IND_T::security_level

Syntax

```
gap_mode_security SM_SIMPLE_PAIRING_COMPLETE_IND_T::security_level
```

Description

The obtained security level

46.55.5 sys_status SM_SIMPLE_PAIRING_COMPLETE_IND_T::status

Syntax

```
sys_status SM_SIMPLE_PAIRING_COMPLETE_IND_T::status
```

Description

sys_status_success or error

46.56 wakeup_data Struct Reference

Structure Definition

Data	Fields
`bool`	wake_asserted
`bool`	wake_on_high

46.56.1 bool wakeup_data::wake_asserted

Syntax

```
bool wakeup_data::wake_asserted
```

Description

TRUE if the WAKE pin has logic 1 level at the time it was sampled after waking up. Note that the WAKE pin is not debounced, therefore this value may not be accurate if WAKE is connected to a mechanical switch. This is equivalent to the value returned by SleepWakePinStatus().

46.56.2 bool wakeup_data::wake_on_high

Syntax

```
bool wakeup_data::wake_on_high
```

Description

TRUE if the WAKE pin was configured to WAKE on a low-to-high edge, FALSE if waking on a high-to-low edge.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

47 Reference Files

The CSR uEnergy firmware library contains the following reference files.

File	Description
aio.h	Analogue I/O configuration and control functions
att_prim.h	Attribute Protocol application interface
battery.h	Read the battery voltage
ble_direct_test.h	Implements the BLE 2-wire Direct Test mode, using the UART
ble_hci_test.h	Defines common functions that provide control over RF test functions, that can be used directly by applications or indirectly via the the BLE 2-wire Direct Test mode
bluetooth.h	Bluetooth specific type definitions
bt_event_types.h	Type definitions for the handling of events that are related to RF activity
buf_utils.h	Functions for reading and writing a little-endian byte buffer
config_store.h	Interface to the Configuration Store (CS)
core_event_types.h	Core type definitions used with all other event definitions
crypt.h	-
debug.h	Simple host interface to the uart driver
doxygen_modules.h	Top level module structure for the doxygen documentation
fault.h	Application services for firmware fault
fault_doxy.h	Fault Code Doxygen documentation
gap_app_if.h	Generic Access Profile interface for Applications
gap_types.h	Generic Access Profile interface for Applications
gatt.h	Defines the GATT interface to the application
gatt_prim.h	Generic Attribute Profile application interface
gatt_uuid.h	Common Bluetooth UUIDs and macros to help applications create in-code GATT databases
hci_error_codes.h	Defines for HCI error codes
hci_event_types.h	Header file of type definitions for the HCI event handling
hci_types.h	Basic type definitions for the HCI
i2c.h	Public header file for functions relating to I2C bus transactions

File	Description
id.h	Build identifier functions
large_flash.h	This module provides access to large (> 512-kbit) SPI Flash devices
ls_app_if.h	Link Supervisor interface to Applications
ls_app_if_event.h	Link Supervisor interface to Applications
ls_err.h	CSR1000 Upper Stack Link Supervisor error codes
ls_types.h	Link Supervisor type definitions
macros.h	Commonly used macros
main.h	Functions relating to powering up the device
mem.h	Services of the memory library
nvm.h	Application services for the non-volatile storage area within the CSR1000 boot device
panic.h	Support for applications to panic due to unrecoverable errors
persistent.h	Application services for the persistent memory store
pio.h	PIO configuration and control functions
pio_ctrlr.h	Drivers for the 8051 PIO Controller
random.h	Generators for pseudo-random data sequences
reset.h	Chip/firmware reset functionality
security.h	Exposes the Security Manager interface to the application
sleep.h	Control the CSR100x/CSR101x sleep states
spi.h	Public header file for functions relating to SPI data transactions
spi_flash.h	Public header file for functions relating to SPI Flash memory access
status.h	CSR1000 System-wide status codes
status_deprecated.h	Legacy status code definitions that have since been replaced with the global status codes defined in status.h

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

File	Description
sys_events.h	System Event definitions and declarations
thermometer.h	Read the temperature sensor
time.h	Application interface to System Time
timer.h	The chip's timers
types.h	Commonly used typedefs
uart.h	Functions to interface with the chip's UART

47.1 att_prim.h

47.1.1 Enumertions

47.1.1.1 enum ATT_ATTR_SEC_T

Syntax

```
enum ATT_ATTR_SEC_T
```

Enumerations

Enumeration	Description
ATT_ATTR_SEC_NONE	No security requirements.
ATT_ATTR_SEC_ENCRYPTION	Encrypted link is required for access.
ATT_ATTR_SEC_AUTHENTICATION	Authenticated MITM protection is required for access.

47.1.1.2 enum att_type_tag

Syntax

```
enum att_type_tag
```

Description

Flat DB attribute types.

Enumerations

Enumeration	Description
att_type_pri_service	0 Primary Service
att_type_sec_service	1 Secondary service
att_type_include	2 Include

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumeration	Description
att_type_declaration	3 Characteristic Declaration
att_type_ch_extended	4 Characteristic Extended Properties
att_type_ch_descr	5 Characteristic User Description
att_type_ch_c_config	6 Client Characteristic Configuration
att_type_ch_s_config	7 Server Characteristic Configuration
att_type_ch_format	8 Characteristic Format
att_type_ch_agg	9 Characteristic Aggregate Format
att_type_reserved_a	a unused
att_type_handle_padding	b Meta-Attribute to pad handle count
att_type_value128	c Characteristic value
att_type_value	d Characteristic value
att_type_full	e Generic Attribute
att_type_full128	f Generic Attribute

47.1.2 Defines

47.1.2.1 ATT_ACCESS_PERMISSION

Definition

```
#define ATT_ACCESS_PERMISSION   0x8000
```

Description

An Access Response is required to grant access.

47.1.2.2 ATT_ACCESS_READ

Definition

```
#define ATT_ACCESS_READ   0x0001
```

Description

Read in progress.

47.1.2.3 ATT_ACCESS_WRITE

Definition

```
#define ATT_ACCESS_WRITE   0x0002
```

Description

Write in progress.

47.1.2.4 ATT_ACCESS_WRITE_COMPLETE

Definition

```
#define ATT_ACCESS_WRITE_COMPLETE   0x4000
```

Description

An Access Response is required to accept value(s) written.

47.1.2.5 ATT_EXECUTE_CANCEL

Definition

```
#define ATT_EXECUTE_CANCEL   0x0000
```

Description

Cancel all pending prepared writes

47.1.2.6 ATT_EXECUTE_WRITE

Definition

```
#define ATT_EXECUTE_WRITE   0x0001
```

Description

Immediately write all pending prepared values

47.1.2.7 ATT_HANDLE_INVALID

Definition

```
#define ATT_HANDLE_INVALID   0x0000
```

Description

Handle 0 is defined as invalid

47.1.2.8 ATT_HANDLE_MAX

Definition

```
#define ATT_HANDLE_MAX    0xFFFF
```

Description

Handle 0xFFFF is defined as the maximum

47.1.2.9 ATT_PERM_AUTHENTICATED

Definition

```
#define ATT_PERM_AUTHENTICATED    ATT_PERM_WRITE_SIGNED
```

Description

This constant is deprecated. Please use ATT_PERM_WRITE_SIGNED instead.

47.1.2.10 ATT_PERM_CONFIGURE_BROADCAST

Definition

```
#define ATT_PERM_CONFIGURE_BROADCAST    0x01
```

Description

If set, permits broadcasts of the Characteristic Value using Characteristic Configuration Descriptor.

47.1.2.11 ATT_PERM_EXTENDED

Definition

```
#define ATT_PERM_EXTENDED    0x80
```

Description

If set, additional characteristic properties are defined in the Characteristic Extended Properties Descriptor.

47.1.2.12 ATT_PERM_INDICATE

Definition

```
#define ATT_PERM_INDICATE    0x20
```

Description

If set, permits indications of a Characteristic Value with acknowledgement.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

47.1.2.13 ATT_PERM_NOTIFY

Definition

```
#define ATT_PERM_NOTIFY    0x10
```

Description

If set, permits notifications of a Characteristic Value without acknowledgement.

47.1.2.14 ATT_PERM_READ

Definition

```
#define ATT_PERM_READ    0x02
```

Description

If set, permits reads of the Characteristic Value.

47.1.2.15 ATT_PERM_RELIABLE_WRITE

Definition

```
#define ATT_PERM_RELIABLE_WRITE    0x0001
```

Description

If set, permits reliable writes of the Characteristic Value.

47.1.2.16 ATT_PERM_WRITE_AUX

Definition

```
#define ATT_PERM_WRITE_AUX    0x0002
```

Description

If set, permits writes to the characteristic descriptor.

47.1.2.17 ATT_PERM_WRITE_CMD

Definition

```
#define ATT_PERM_WRITE_CMD    0x04
```

Description

If set, permit writes of the Characteristic Value without response.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

47.1.2.18 ATT_PERM_WRITE_REQ

Definition

```
#define ATT_PERM_WRITE_REQ    0x08
```

Description

If set, permits writes of the Characteristic Value with response.

47.1.2.19 ATT_PERM_WRITE_SIGNED

Definition

```
#define ATT_PERM_WRITE_SIGNED    0x40
```

Description

If set, permits signed writes to the Characteristic Value.

47.1.2.20 ATT_UUID128

Definition

```
#define ATT_UUID128    0x0002
```

Description

UUID is a 128-bit UUID

47.1.2.21 ATT_UUID16

Definition

```
#define ATT_UUID16    0x0001
```

Description

UUID is a 16-bit Attribute UUID

47.1.2.22 ATT_UUID_NONE

Definition

```
#define ATT_UUID_NONE    0x0000
```

Description

No UUID present.

CSR

47.1.2.23 ATT_WRITE_COMMAND

Definition

```
#define ATT_WRITE_COMMAND   0x0040
```

Description

Send Write Command to the server.

47.1.2.24 ATT_WRITE_REQUEST

Definition

```
#define ATT_WRITE_REQUEST   0x0000
```

Description

Send Write Request to the server

47.1.2.25 ATT_WRITE_SIGNED

Definition

```
#define ATT_WRITE_SIGNED   0x0080
```

Description

Send Signed Write to the server. Only Write Command can be signed.

47.1.3 Typedefs

47.1.3.1 typedef uint16 att_uuid_type_t

Syntax

```
typedef uint16 att_uuid_type_t
```

Description

UUID Type

47.2 bluetooth.h

47.2.1 Enumerations

47.2.1.1 enum radio_event

Syntax

```
enum radio_event
```

Description

Define radio events that may be reported to the application.

Enumerations

Enumeration	Description
radio_event_none	Do not report any radio events
radio_event_tx_data	Report each transmitted packet
radio_event_connection_event	Report once per connection event, even if no data was sent
radio_event_first_tx	Report only first TX data packet in a connection event

47.3 bt_event_types.h

47.3.1 Enumerations

47.3.1.1 enum GATT_UUID_T

Syntax

```
enum GATT_UUID_T
```

Description

Type of UUID specified.

Enumerations

Enumeration	Description
GATT_UUID_NONE	No UUID.
GATT_UUID16	16-bit UUID
GATT_UUID128	128-bit UUID

47.3.1.2 enum L2CA_ADDR_TYPE_T

Syntax

```
enum L2CA_ADDR_TYPE_T
```

Description

L2CAP Bluetooth Address Type.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumerations

Enumeration	Description
L2CA_PUBLIC_ADDR_TYPE	Public address type used.
L2CA_RANDOM_ADDR_TYPE	Random address type used.

47.3.1.3 enum L2CA_CONNECTION_T

Syntax

```
enum L2CA_CONNECTION_T
```

Description

L2CAP Connection Type.

Enumerations

Enumeration	Description
L2CA_CONNECTION_LE_MASTER_DIRECTED	Connect as Master.
L2CA_CONNECTION_LE_MASTER_WHITELIST	Connect as Master using whitelist.
L2CA_CONNECTION_LE_SLAVE_DIRECTED	Start directed adverts.
L2CA_CONNECTION_LE_SLAVE_WHITELIST	Use whitelist for undirected adverts.
L2CA_CONNECTION_LE_SLAVE_UNDIRECTED	Start undirected adverts.
L2CA_CONNECTION_LE_SLAVE_DIRECTED_LDC	Start low duty cycle directed adverts.

47.3.2 Typedefs

47.3.2.1 typedef uint16 l2ca_conflags_t

Syntax

```
typedef uint16 l2ca_conflags_t
```

47.4 buf_utils.h

FUNCTIONS HERE

47.4.1 Title

47.4.1.1 BufReadUint16

Syntax

```
uint16 BufReadUint16 ( uint8 ** buf)
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

Read a little-endian 16-bit word from the buffer and advance the buffer pointer.

Parameters

Parameter	Description
buf	A pointer to the buffer pointer.

47.4.1.2 BufReadUint32

Syntax

```
uint32 BufReadUint32 ( uint8 ** buf)
```

Description

Read a little-endian 32-bit word from the buffer and advance the buffer pointer.

Parameters

Parameter	Description
buf	A pointer to the buffer pointer.

47.4.1.3 BufReadUint8

Syntax

```
uint8 BufReadUint8 ( uint8 ** buf)
```

Description

Read a byte from the buffer and advance the buffer pointer.

Parameters

Parameter	Description
buf	A pointer to the buffer pointer.

47.4.1.4 BufWriteUint16

Syntax

```
void BufWriteUint16 ( uint8 ** buf, uint16 val )
```

Description

Write a 16-bit word to the buffer in little-endian byte order and advance the buffer pointer.

CSR

Parameters

Parameter	Description
`buf`	A pointer to the buffer pointer.
`val`	The word to write.

47.4.1.5 BufWriteUint32

Syntax

```
void BufWriteUint32 ( uint8 ** buf, uint32 * p_val )
```

Description

Write a 32-bit word to the buffer in little-endian byte order and advance the buffer pointer.

Parameters

Parameter	Description
`buf`	A pointer to the buffer pointer.
`val`	A pointer to the word to write.

47.4.1.6 BufWriteUint8

Syntax

```
void BufWriteUint8 ( uint8 ** buf, uint8 val )
```

Description

Write a byte to the buffer and advance the buffer pointer.

Parameters

Parameter	Description
`buf`	A pointer to the buffer pointer.
`val`	The byte to write.

47.5 core_event_types.h

47.5.1 Enumerations

47.5.1.1 enum lm_event_code

Syntax

```
enum lm_event_code
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Description

Definitions of the event identifiers used by LM_EVENT_T to select the structure of each event. This enumeration lists all possible requests that the host application can make of or receive from the Command Interface. Event codes 0x0000 - 0x003D are the standard Bluetooth HCI event codes, and are not documented here. See Volume 2 Part E section 7.7 of the Bluetooth Specification v4.1 for details.

Enumerations

Enumeration	Description
LM_EV_DISCONNECT_COMPLETE	The Disconnection Complete event is used to indicate that a connection is terminated. See Volume 2 Part E section 7.7.5 of the Bluetooth Specification v4.1 for details. See also LM_EV_DISCONNECT_COMPLETE_T.
LM_EV_ENCRYPTION_CHANGE	The Encryption Change event is used to indicate that the change of the encryption mode has been completed. See Volume 2 Part E section 7.7.8 of the Bluetooth Specification v4.1 for details. See also LM_EV_ENCRYPTION_CHANGE_T.
LM_EV_REMOTE_VERSION_INFO	HCI Remote Version Info event. See Volume 2 Part E section 7.7.12 of the spec. See also LM_EV_REMOTE_USED_FEATURES_T.
LM_EV_COMMAND_COMPLETE	For internal use.
LM_EV_COMMAND_STATUS	For internal use.
LM_EV_CONNECTION_COMPLETE	BLE Connection Complete meta event, handled by GATT module. See Volume 2 Part E Section 7.7.65.1 of the spec. See also LM_EV_CONNECTION_COMPLETE_T.
LM_EV_ADVERTISING_REPORT	BLE Advertising Report meta event. See Volume 2 Part E section 7.7.65.2 of the spec. See also LM_EV_ADVERTISING_REPORT_T.
LM_EV_CONNECTION_UPDATE	BLE Connection Update meta event, handled by LS module. See Volume 2 Part E section 7.7.65.3 of the spec. See also LM_EV_CONNECTION_UPDATE_T.
LM_EV_REMOTE_USED_FEATURES	BLE Remote Used Features meta event. See Volume 2 Part E section 7.7.65.4 of the spec. See also LM_EV_REMOTE_USED_FEATURES_T.
LM_EV_LONG_TERM_KEY_REQUESTED	BLE Long Term Key Request meta event, handled internally by the Security Manager module. See Volume 2 Part E section 7.7.65.5 of the spec. See also LM_EV_LONG_TERM_KEY_REQUESTED_T.
LM_EV_ACL_DATA_START	For internal use.
LM_EV_ACL_DATA_CONT	For internal use.
LM_EV_MANUFACTURER_EXTENSION	Manufacturer-specific HCI event.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Enumeration	Description
GATT_ADD_DB_REQ	Unused: corresponds to GattAddDatabaseReq()
GATT_ADD_DB_SEGMENT_REQ	Unused.
GATT_CONNECT_REQ	Try to establish a connection with the specified remote device. Note:This command is appropriate only when this device has the Central role. It will result in this device being the link Master. Corresponds to GattConnectReq()
GATT_CANCEL_CONNECT_REQ	Cancel a connect attempt initiated using the GATT_CONNECT_REQ message. Corresponds to GattCancelConnectReq()
GATT_DISCONNECT_REQ	Unused: Disconnect from the remote device (if any). corresponds to GattDisconnectReq()
GATT_DISCONNECT_REASON_REQ	Unused: corresponds to GattDisconnectReasonReq()
GATT_EXCHANGE_MTU_REQ	Unused: corresponds to GattExchangeMtuReq()
GATT_EXCHANGE_MTU_RSP	Unused: corresponds to GattExchangeMtuRsp()
GATT_DISC_ALL_PRIM_SERV_REQ	Unused: corresponds to GattDiscoverAllPrimaryServices()
GATT_DISC_PRIM_SERV_BY_UUID_REQ	Unused: corresponds to GattDiscoverPrimaryServiceByUuid()
GATT_FIND_INCLUDED_SERV_REQ	Unused: corresponds to GattFindIncludedServices()
GATT_DISC_SERVICE_CHAR_REQ	Unused: corresponds to GattDiscoverServiceChar()
GATT_DISC_ALL_CHAR_DESC_REQ	Unused: corresponds to GattDiscoverAllCharDescriptors()
GATT_READ_CHAR_VAL_REQ	Unused: corresponds to GattReadCharValue()
GATT_READ_CHAR_USING_UUID_REQ	Unused: corresponds to GattReadCharUsingUuid()
GATT_READ_LONG_CHAR_VAL_REQ	Unused: corresponds to GattReadLongCharValue()
GATT_READ_MULTI_CHAR_VAL_REQ	Unused: corresponds to GattReadMultipleCharValues()
GATT_WRITE_CHAR_VAL_REQ	Unused: corresponds to GattWriteCharValueReq()
GATT_WRITE_LONG_CHAR_VAL_REQ	Unused: corresponds to GattWriteLongCharValueReq()
GATT_CHAR_VAL_NOTIFICATION_REQ	Unused: corresponds to GattCharValueNotification()
GATT_CHAR_VAL_INDICATION_REQ	Unused: corresponds to GattCharValueIndication()
GATT_ACCESS_RSP	Unused: corresponds to GattAccessRsp()
GATT_STOP_CURRENT_PROC_REQ	Unused: corresponds to GattStopCurrentProcCmd()

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumeration	Description
GATT_ATT_PREPARE_WRITE_REQ	Unused: corresponds to GattAttPrepareWriteReq()
GATT_ATT_EXECUTE_WRITE_REQ	Unused: corresponds to GattAttExecuteWriteReq()
GATT_TRAFFIC_GEN_REQ	Used to manage automatic traffic generation for test purposes. Supported in uci command interface from version 0.4.
GATT_ADD_DB_CFM	Confirmation that the attribute database has been installed. see GATT_ADD_DB_CFM_T.
GATT_ADD_DB_SEGMENT_CFM	Unused.
GATT_CONNECT_CFM	Indicates completion of a GattConnectReq() see GATT_CONNECT_CFM_T.
GATT_CONNECT_IND	Indicates remotely-initiated connection has completed see GATT_CONNECT_IND_T.
GATT_CANCEL_CONNECT_CFM	Indicates completion of a GattCancelConnectReq() see GATT_CANCEL_CONNECT_CFM_T.
GATT_DISCONNECT_CFM	Indicates completion of a GattDisconnectReq() see GATT_DISCONNECT_CFM_T.
GATT_DISCONNECT_IND	Indicates a remotely-initiated disconnection has occurred. see GATT_DISCONNECT_IND_T.
GATT_EXCHANGE_MTU_CFM	Indicates the Exchange MTU sub-procedure has completed. See Volume 3 Part G section 4.3.1 of the spec. see GATT_EXCHANGE_MTU_CFM_T.
GATT_EXCHANGE_MTU_IND	Indicates client has initiated the Exchange MTU sub-procedure. See Volume 3 Part G section 4.3.1 of the spec. The application must respond by calling GattExchangeMtuRsp(). see GATT_EXCHANGE_MTU_IND_T.
GATT_DISC_ALL_PRIM_SERV_CFM	Indicates the Discover All Primary Services sub-procedure (see Volume 3 Part G section 4.4.1 of the spec) has completed. Service data is returned in GATT_SERV_INFO_IND messages. see GATT_DISC_ALL_PRIM_SERV_CFM.
GATT_SERV_INFO_IND	Lists services discovered through the service discovery procedures. see GATT_SERV_INFO_IND_T.
GATT_DISC_PRIM_SERV_BY_UUID_CFM	Indicates the Discover Primary Service by Service UUID sub-procedure (see Volume 3 Part G section 4.4.2 of the spec) has completed. see GATT_DISC_PRIM_SERV_BY_UUID_CFM_T.

Enumeration	Description
GATT_DISC_PRIM_SERV_BY_UUID_IND	Contains service data requested by GattDiscoverPrimaryServiceByUuid(). See Volume 3 Part G section 4.4.2 of the spec. see GATT_DISC_PRIM_SERV_BY_UUID_IND_T.
GATT_FIND_INCLUDED_SERV_CFM	Indicates the Find Included Services sub-procedure (see Volume 3 Part G section 4.5.1 of the spec) has completed. Service data are returns in GATT_SERV_INFO_IND messages. see GATT_FIND_INCLUDED_SERV_CFM_T.
GATT_DISC_SERVICE_CHAR_CFM	Indicates the Discover All Characteristics of a Service sub-procedure (see Volume 3 Part G section 4.6.1 of the spec) or the Discover Characteristics by UUID sub-procedure (section 4.6.2) has completed. Service characteristics are returned in GATT_CHAR_DECL_INFO_IND messages. see GATT_DISC_SERVICE_CHAR_CFM_T.
GATT_CHAR_DECL_INFO_IND	Lists characteristics discovered through the characteristic discovery procedures. see GATT_CHAR_DECL_INFO_IND_T.
GATT_DISC_ALL_CHAR_DESC_CFM	Indicates the Discover All Characteristic Descriptors sub-procedure (see Volume 3 Part G section 4.7.1 of the spec) has completed. Characteristic descriptors are returned in GATT_CHAR_DESC_INFO_IND messages. see GATT_DISC_ALL_CHAR_DESC_CFM_T.
GATT_CHAR_DESC_INFO_IND	Lists characteristic descriptors discovered through the characteristic discovery procedures. see GATT_CHAR_DESC_INFO_IND_T.
GATT_READ_CHAR_VAL_CFM	Contains the characteristic value requested by GattReadCharValue(). See Volume 3 Part G section 4.8.1 for details. see GATT_READ_CHAR_VAL_CFM_T.
GATT_READ_CHAR_USING_UUID_CFM	Indicates the Read Using Characteristic UUID sub-procedure (see Volume 3 Part G section 4.8.2 of the spec) has completed. Characteristic values are returned in GATT_CHAR_VAL_IND messages. see GATT_READ_CHAR_USING_UUID_CFM_T.
GATT_UUID_CHAR_VAL_IND	Contains the characteristic value requested by GattReadCharUsingUUid(). see GATT_CHAR_VAL_IND_T.
GATT_READ_LONG_CHAR_VAL_CFM	Indicates the Read Long Characteristic Values sub-procedure (see Volume 3 Part G section 4.8.3 of the spec) has completed. Characteristic values are returned in GATT_LONG_CHAR_VAL_IND messages. see GATT_READ_LONG_CHAR_VAL_CFM_T.
GATT_LONG_CHAR_VAL_IND	Contains the characteristic value requested through the Read Long Characteristic Values sub-procedure. see GATT_LONG_CHAR_VAL_IND_T.

Enumeration	Description
GATT_READ_MULTI_CHAR_VAL_CFM	Contains a characteristic values requested through the Read Multiple Characteristic Values sub-procedure (see Volume 3 Part G section 4.8.4 of the spec). One message will be generated for each value requested. see GATT_READ_MULTI_CHAR_VAL_CFM_T.
GATT_WRITE_CHAR_VAL_CFM	Indicates that a Characteristic Value Write procedure other than the Write Long Characteristic Values sub-procedure has completed. See Volume 3 Part G section 4.9 of the spec for details. see GATT_WRITE_CHAR_VAL_CFM_T.
GATT_WRITE_LONG_CHAR_VAL_CFM	Indicates the Write Long Characteristic Values sub-procedure (see Volume 3 Part G section 4.9.4 of the spec) has completed. see GATT_WRITE_LONG_CHAR_VAL_CFM_T.
GATT_CHAR_VAL_IND_CFM	Indicates the Indications sub-procedure (see Volume 3 Part G section 4.11.1 of the spec) has completed. see GATT_CHAR_VAL_IND_CFM_T.
GATT_CHAR_VAL_NOT_CFM	Indicates the Notifications sub-procedure (see Volume 3 Part G section 4.10.1 of the spec) has completed. see GATT_CHAR_VAL_IND_CFM_T.
GATT_ACCESS_IND	Indicates that an attribute controlled directly by the application (ATT_ATTR_IRQ attribute flag is set) is being read from or written to. The application shall shall treat it as an atomic event and respond by calling GattAccessRsp()immediately without any context switch or calling other gatt functions. see GATT_ACCESS_IND_T.
GATT_IND_CHAR_VAL_IND	Indicates the peer has indicated a characteristic value (see Volume 3 Part G section 4.11.1 of the spec). see GATT_CHAR_VAL_IND_T.
GATT_NOT_CHAR_VAL_IND	Indicates the peer has notified a characteristic value (see Volume 3 Part G section 4.10.1 of the spec). see GATT_CHAR_VAL_IND_T.
GATT_ATT_PREPARE_WRITE_CFM	Indicates the peer has accepted the ATT prepare write request see GATT_ATT_PREPARE_WRITE_CFM_T.
GATT_ATT_EXECUTE_WRITE_CFM	Indicates the peer has accepted the ATT execute write request see GATT_ATT_EXECUTE_WRITE_CFM_T.
GATT_TRAFFIC_GEN_CFM	Response to automatic traffic generation requests. Used in uci command interface from version 0.4.
GATT_TRAFFIC_GEN_IND	Indication of the completion of a automatic traffic generation operation. Supported in uci command interface from version 0.4.
LS_LONG_TERM_KEY_REQUESTED_IND	Indicates long term key request from the peer.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumeration	Description
LS_CONNECTION_UPDATE_IND	Indicates connection update request from the peer.
LS_REMOTE_USED_FEATURES_IND	Indicates remote used request from the peer.
LS_SET_TRANSMIT_POWER_LEVEL_CFM	Confirmation that the Tx power level has been modified.
LS_HOLD_TX_UNTIL_RX_CFM	Enables the send after receive feature.
LS_RX_TIMING_REPORT_CFM	Enables data timing reports on each RX packet (master only)
LS_DATA_RX_TIMING_IND	Provides data timing report to application on receiving a data packet. see LS_DATA_RX_TIMING_IND_T.
GAP_GET_CONNECTION_CHANNEL_MAP_CFM	reads the connection channel map
GAP_SET_CONNECTION_CHANNEL_MAP_CFM	set the connection channel map
LS_GAP_SEED_STATIC_ADDR_CFM	the static address generated using seed
LS_GAP_SET_TGAP_CONN_PARAM_TIMEOUT_CFM	set the TGAP(conn_param_timeout) timer
SM_SECURITY_LEVEL_REQ	Unused: corresponds to SMRequestSecurityLevel()
SM_KEY_REQUEST_RSP	Unused: corresponds to SMKeyRequestResponse()
SM_ADD_STORED_KEY_REQ	Unused: corresponds to SMAddStoredKey()
SM_REMOVE_STORED_KEY_REQ	Unused: corresponds to SMRemoveStoredKey()
SM_CONFIGURATION_REQ	Unused: corresponds to SMSetIOCapabilities(), SMSetMaxEncKeySize(), SMSetMinEncKeySize()
SM_PASSKEY_DISPLAY_RSP	Unused: corresponds to SMPasskeyDisplayed()
SM_PASSKEY_INPUT_RSP	A response to a SM_PASSKEY_DISPLAY_IND message. Corresponds to SMPasskeyInput()
SM_PASSKEY_INPUT_NEG_RSP	Unused: corresponds to SMPasskeyInputNeg()
SM_PRIVACY_REGENERATE_ADDRESS_REQ	Unused: corresponds to SMPrivacyRegenerateAddress()
SM_DIV_APPROVAL_RSP	Unused: corresponds to SMDivApproval()
SM_PAIRING_AUTH_RSP	Unused: corresponds to SMPairingAuthRsp()
SM_FEATURES_REQ	Unused.
SM_ENCRYPT_RAW_AES_REQ	Request data be encrypted.
SM_LONG_TERM_KEY_RSP	Unused: corresponds to SMLongTermKeyRsp()
SM_PRIVACY_GET_OWN_IRK_REQ	Request device's own IRK.

Enumeration	Description
SM_DISTRIBUTE_MASTER_LTK_REQ	Indicate whether the Security Manager should request distribution of the master's Long Term Key during bonding. During bonding, the peer devices negotiate which keys to distribute to each other. This function allows the application to decide whether the LTK, EDIV and Rand should be distributed by the master of the connection. It can be used when the local device is the master or when it is the slave.The default is for the Security Manager to not request distribution of the master key, as typically this key is only required if the master and slave devices are likely to swap roles but wish to retain the existing bond.
SM_SIMPLE_PAIRING_COMPLETE_IND	Indicates the Pairing Feature Exchange has completed, successfully or otherwise. See Volume 3 Part H section 2.3 of the Bluetooth v4.1 specification for more information on pairing. see SM_SIMPLE_PAIRING_COMPLETE_IND_T.
SM_SECURITY_LEVEL_CFM	Unused.
SM_CSRK_COUNTER_CHANGE_IND	Currently unimplemented.
SM_KEYS_IND	Contains the keys and associated security information used on a connection that has completed Short Term Key Generation or Transport Specific Key Distribution. See Volume 3 Part H section 2.1 of the Bluetooth v4.1 specification. see SM_KEYS_IND_T.
SM_KEY_REQUEST_IND	Indicates that the Security Manager cannot find security keys for the host in its persistent store. Application responds with either a SM_KEYSET_T or NULL pointer in SMKeyRequestResponse() see SM_KEY_REQUEST_IND_T.
SM_UNSTORED_KEY_IND	Currently unimplemented.
SM_PASSKEY_DISPLAY_IND	Indicates that the Security Manager is in pairing mode, and need the application to display the pass key which the peer has to enter. Application shall respond with SMPasskeyDisplayed() when the key has been displayed. see SM_PASSKEY_DISPLAY_IND_T.
SM_PASSKEY_INPUT_IND	Indicates that the Security Manager is in pairing mode, and need the user to enter the pass key displayed by the peer. The Application shall respond with SMPasskeyInput() containing the entered pass key or SMPasskeyInputNeg() to abort the pairing process. see SM_PASSKEY_INPUT_IND_T.
SM_PASSKEY_COMPARE_IND	Reserved for future use.

Enumeration	Description
SM_DIV_APPROVE_IND	Indicates that the Security Manager has received a encryption request from the peer. The peer want to encrypt the link with the key corresponding with the supplied diversifier. The Application can either approve the use of the diversifier or revoke it. This is the only way for the application to inform the peer, that it has revoked the key and removed the bond. The Application shall treat this event atomic and respond with SMDivApproval() immediately without any context switch or other GATT calls. see SM_DIV_APPROVE_IND_T.
SM_PAIRING_AUTH_IND	Indicates that a pairing request has been received from the peer device, allowing the application to either authorise or reject the request. This can be used, for example, to prevent pairing unless the user has pressed a "Pairing" button on the local device. See SM_PAIRING_AUTH_IND_T.
SM_FEATURES_CFM	Unused.
SM_ENCRYPT_RAW_AES_CFM	Confirms and returns encrypted data.
SM_LONG_TERM_KEY_IND	Indicates that the Security Manager has received a encryption request from the peer. The application has indicated that it wants to manage some Long Term Keys independently of Security Manager pairing. If the application has an LTK for the current connection then it should call SMLongTermKeyRsp() and provide the key. If it does not have an LTK it should call SMLongTermKeyRsp() with appropriate status to pass handling of encryption back to the Security Manager (in which case SM will recreate the LTK using the EDIV and RAND, and optionally may then ask the application for DIV approval). The Application shall treat this event as atomic and respond with SMDivApproval() immediately without any context switch or other API calls.See also SM_LONG_TERM_KEY_IND_T
SM_PRIVACY_GET_OWN_IRK_CFM	Returns device's own IRK.
SM_LOST_BOND_IND	Indicates that the device has lost the bond.
SM_DISTRIBUTE_MASTER_LTK_CFM	Unused.
LS_READ_WHITELIST_SIZE_REQ	Unused: corresponds to LsReadWhiteListMaxSize()
LS_RESET_WHITELIST_REQ	Unused: corresponds to LsResetWhiteList()
LS_ADD_DEVICE_TO_WHITELIST_REQ	Unused: corresponds to LsAddWhiteListDevice()
LS_DELETE_WHITELIST_DEVICE_REQ	Unused: corresponds to LsDeleteWhiteListDevice()
LS_READ_REMOTE_VERSION_INFO_REQ	Unused: corresponds to LsReadRemoteVersionInformation()

Enumeration	Description
LS_READ_RSSI_REQ	Unused: corresponds to LsReadRssi()
LS_READ_TRANSMIT_POWER_LEVEL_REQ	Unused: corresponds to LsReadTransmitPowerLevel()
LS_READ_REMOTE_USED_FEATURES_REQ	Unused: corresponds to LsReadRemoteUsedFeatures()
LS_SET_NEW_CONNECTION_PARAM_REQ	Set the connection parameters for new connections. Devices operating as a BLE master (Central devices, typically) will use these parameters for all subsequent connections. This command does not change existing connections - use LS_CONNECTION_PARAM_UPDATE_REQ to do that. Note:This function is not used on slave (peripheral) devices Corresponds to LsSetNewConnectionParamReq()
LS_CONNECTION_PARAM_UPDATE_REQ	Request an update to the connection parameters for an existing connection. This command is valid for both master and slave devices. Corresponds to LsConnectionParamUpdateReq()
LS_CONNECTION_UPDATE_SIGNALLING_RSP	Unused: corresponds to LsConnectionUpdateSignalingRsp()
LS_STORE_ADV_SCAN_DATA_REQ	Unused: corresponds to LsStoreAdvScanData()
LS_ADVERTISE_REQ	Start /stop advertising. Remote devices cannot connect to Command Interface unless it is advertising. Note:This command is applicable only when the local device is a Peripheral device. Corresponds to LsStartStopAdvertise()
LS_SCAN_REQ	Unused: Start/stop scanning for other devices in the vicinity. This command is appropriate only when this device has the Central role. Note:Enabling scanning at a high rate can result in a lot of messages being sent to the host application, since Command Interface does not perform any filtering on the results. Corresponds to LsStartStopScan()
LS_GAP_SET_MODE_REQ	Unused: corresponds to GapSetMode()
LS_GAP_SET_RANDOM_ADDR_REQ	Unused: corresponds to GapSetRandomAddress()
LS_GAP_SET_ADV_ADDR_REQ	Unused: corresponds to GapSetAdvAddress()
LS_GAP_SET_SCAN_INTERVAL_REQ	Unused: corresponds to GapSetScanInterval()
LS_GAP_SET_ADV_INTERVAL_REQ	Unused: corresponds to GapSetAdvInterval()
LS_GAP_SET_SCAN_TYPE_REQ	Unused: corresponds to GapSetScanType()
LS_GAP_SET_ADV_CHAN_MASK_REQ	Unused: corresponds to GapSetAdvChanMask()
LS_GAP_GET_RANDOM_ADDR_REQ	Unused: corresponds to GapGetRandomAddress()

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumeration	Description
LS_RADIO_EVENT_NOTIFICATION_REQ	Unused: corresponds to LsRadioEventNotification()
LS_SET_TRANSMIT_POWER_LEVEL_REQ	Unused: corresponds to LsSetTransmitPowerLevel()
LS_HOLD_TX_UNTIL_RX_REQ	Unused: corresponds to LsHoldTxUntilRx()
LS_RX_TIMING_REPORT_REQ	Unused: corresponds to LsRxTimingReport()
GAP_GET_CONNECTION_CHANNEL_MAP_REQ	get the channel map
GAP_SET_CONNECTION_CHANNEL_MAP_REQ	set the channel map
LS_GAP_SET_TGAP_CONN_PARAM_TIMEOUT_REQ	set the TGAP(conn_param_timeout)
LS_READ_WHITELIST_SIZE_CFM	Unused.
LS_RESET_WHITELIST_CFM	Unused.
LS_ADD_DEVICE_TO_WHITELIST_CFM	Unused.
LS_DELETE_WHITELIST_DEVICE_CFM	Unused.
LS_READ_REMOTE_VERSION_INFO_CFM	Unused.
LS_READ_RSSI_CFM	Unused.
LS_READ_TRANSMIT_POWER_LEVEL_CFM	Unused.
LS_READ_REMOTE_USED_FEATURES_CFM	Unused.
LS_SET_NEW_CONNECTION_PARAM_CFM	Unused.
LS_CONNECTION_PARAM_UPDATE_CFM	Response to LsConnectionParamUpdateReq() see LS_CONNECTION_PARAM_UPDATE_CFM_T.
LS_CONNECTION_PARAM_UPDATE_IND	Indicates remotely-triggered Connection Update has completed see LS_CONNECTION_PARAM_UPDATE_IND_T.
LS_CONNECTION_UPDATE_SIGNALLING_IND	L2CAP signal requesting a Connection Update has been received. The application must accept or reject it by calling LsConnectionUpdateSignalingRsp() appropriately. see LS_CONNECTION_UPDATE_SIGNALLING_IND_T.
LS_STORE_ADV_SCAN_DATA_CFM	Unused.
LS_ADVERTISE_CFM	Unused.
LS_SCAN_CFM	Unused.
LS_GAP_SET_MODE_CFM	Unused.
LS_GAP_SET_RANDOM_ADDR_CFM	Unused.

CSR

Enumeration	Description
LS_GAP_SET_ADV_ADDR_CFM	Unused.
LS_GAP_SET_SCAN_INTERVAL_CFM	Unused.
LS_GAP_SET_ADV_INTERVAL_CFM	Unused.
LS_GAP_SET_SCAN_TYPE_CFM	Unused.
LS_GAP_SET_ADV_CHAN_MASK_CFM	Unused.
LS_GAP_GET_RANDOM_ADDR_CFM	Unused.
LS_RADIO_EVENT_NOTIFICATION_CFM	Unused.
LS_RADIO_EVENT_IND	Optional radio activity event (see LsRadioEventNotification() and LS_RADIO_EVENT_IND_T for further information).
LS_ADVERTISING_REPORT_IND	Indicates an advertising or scan report message has been received.
LS_DISCONNECT_COMPLETE_IND	Indicates disconnect with peer has completed.
LS_ENCRYPTION_CHANGE_IND	Indicates an encryption change has been initiated by the peer.
LS_ENCRYPTION_KEY_REFRESH_IND	Indicates key refresh has been initiated by the peer.
LS_NUMBER_COMPLETED_PACKETS_IND	Indicates number of completed sent to peer.
LS_REMOTE_VERSION_INFO_IND	Indicates the response to a remote version info request has been received.
LS_CONNECTION_COMPLETE_IND	Indicates connect phase has completed.
SYS_BACKGROUND_TICK_REQ	Request the initiation of the background tick indication to the application.
SYS_GET_LOCAL_ADDR_REQ	Request the local Bluetooth address.
SYS_GET_LOCAL_VERSION_INFO_REQ	Request the version number of the current application plus library build info.
SYS_SET_EVENT_MASK_REQ	Request the setting of the event mask.
SYS_GET_TX_POWER_REQ	Request the TX Power setting.
SYS_GET_USER_KEY_REQ	Request a given user key setting.
SYS_GET_TEMPERATURE_REQ	Request the chip temperature.
SYS_SET_PIO_REQ	Request the setting of a PIO.
SYS_GET_PIO_REQ	Request the current setting of a PIO.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

CSR

Enumeration	Description
SYS_SET_PIOS_REQ	Request the setting of a number of PIOs.
SYS_GET_PIOS_REQ	Request the current setting of a number of PIOs.
SYS_SET_PIO_DIR_REQ	Request the direction setting of a PIO.
SYS_GET_PIO_DIR_REQ	Request the current direction setting of a PIO.
SYS_SET_PIOS_DIR_REQ	Request the direction setting of a number of PIOs.
SYS_GET_PIOS_DIR_REQ	Request the current direction setting of a number of PIOs.
SYS_SET_PIOS_PULL_MODE_REQ	Request the setting of pull mode for a number of PIOs.
SYS_SET_PIO_MODE_REQ	Request the mode setting of a PIO.
SYS_SET_PIOS_MODE_REQ	Request the mode setting of a number of PIOs.
SYS_SET_PIO_ANA_MON_CLK_REQ	Request the clock selection of any suitably configured PIO.
SYS_SET_PIOS_EVENT_MODE_REQ	Request the setting of the event mode for a number of PIOs.
SYS_SET_PIO_I2C_PULL_MODE_REQ	Request the setting of the pull mode for the dedicated I2C PIO.
SYS_SET_PIO_PWM_REQ	Request the configuration of 1 of the 4 PWM PIO.
SYS_SET_PIO_ENABLE_PWM_REQ	Request the enable of 1 of the 4 PWM PIO.
SYS_SET_PIO_ENABLE_EDGE_CAPTURE_REQ	Request the setting of edge capture of all PIOs.
SYS_GET_PIO_EDGE_CAPTURE_REQ	Request the current setting of edge capture of all PIOs.
SYS_SET_PIO_QUADRATURE_DECODER_REQ	Request the enable of a given quadrature decoder PIO.
SYS_SET_PIO_QUADRATURE_DECODERS_REQ	Request the enable of a number of quadrature decoder PIOs.
SYS_GET_PIO_QUADRATURE_DECODER_REQ	Request the current count of a quadrature decoder PIO.
SYS_GET_PERSISTENT_MEM_VALID_REQ	Request the validity of the persistent memory.
SYS_GET_PERSISTENT_MEM_SIZE_REQ	Request the size of the persistent memory.
SYS_GET_PERSISTENT_MEM_REQ	Request the read of the persistent memory.
SYS_SET_PERSISTENT_MEM_REQ	Request the write of the persistent memory.
SYS_RESET_PERSISTENT_MEM_REQ	Request the erasure of the persistent memory.
SYS_WARM_RESET_REQ	Request cpu warm start.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumeration	Description
SYS_GET_BUILD_ID_REQ	Request the application software build id.
SYS_GET_ROM_BUILD_ID_REQ	Request the rom software build id.
SYS_GET_BATTERY_VOLTAGE_REQ	Request the current battery voltage.
SYS_PANIC_REQ	Request the chip entry a panic state.
SYS_AIO_DRIVE_REQ	Request the setting of an AIO.
SYS_AIO_READ_REQ	Request the current setting of an AIO.
SYS_AIO_OFF_REQ	Request the disable of an AIO.
SYS_AIO_DIG_REQ	Request an AIO be used for digital output.
SYS_READ_APP_PANIC_CODE_REQ	Request for last application panic code.
SYS_CLEAR_APP_PANIC_CODE_REQ	Request for clearing application panic code.
SYS_READ_FW_FAULT_ID_REQ	Request for last fw fault ID.
SYS_CLEAR_FW_FAULT_ID_REQ	Request for clearing fw fault.
SYS_GET_BUILD_NAME_REQ	Request string identifying build.
SYS_GET_UCI_VERSION_REQ	Request UCI version number.
SYS_GET_BATTERY_LOW_THRESHOLD_REQ	Request value of the Low battery Threshold CS Key.
SYS_BACKGROUND_TICK_IND	Optional background tick event (see AppBackgroundTick() for further information). This event has no parameters.
SYS_BACKGROUND_TICK_CFM	Confirms background tick request.
SYS_GET_LOCAL_ADDR_CFM	Return the local address information.
SYS_GET_LOCAL_VERSION_INFO_CFM	return the local version information.
SYS_SET_EVENT_MASK_CFM	Confirms event mask has been set.
SYS_GET_TX_POWER_CFM	Confirms the TX Power setting.
SYS_GET_USER_KEY_CFM	Confirms the given user key setting.
SYS_GET_TEMPERATURE_CFM	Confirms the chip temperature.
SYS_SET_PIO_CFM	Confirms the set PIO request.
SYS_GET_PIO_CFM	Confirms the get PIO request.
SYS_SET_PIOS_CFM	Confirms the set PIOs request.

Enumeration	Description
SYS_GET_PIOS_CFM	Confirms the get PIOs request.
SYS_SET_PIO_DIR_CFM	Confirms the set PIO direction request.
SYS_GET_PIO_DIR_CFM	Confirms the get PIO direction request.
SYS_SET_PIOS_DIR_CFM	Confirms the set PIOs direction request.
SYS_GET_PIOS_DIR_CFM	Confirms the get PIOs direction request.
SYS_SET_PIOS_PULL_MODE_CFM	Confirms the set PIOs pull mode request.
SYS_SET_PIO_MODE_CFM	Confirms the set PIO mode request.
SYS_SET_PIOS_MODE_CFM	Confirms the set PIOs mode request.
SYS_SET_PIO_ANA_MON_CLK_CFM	Confirms the set PIO ana mon clk request.
SYS_SET_PIOS_EVENT_MODE_CFM	Confirms the set PIOs event mode request.
SYS_SET_PIO_I2C_PULL_MODE_CFM	Confirms the set PIO I2C pull mode request.
SYS_SET_PIO_PWM_CFM	Confirms the set PIO pwm request.
SYS_SET_PIO_ENABLE_PWM_CFM	Confirms the set PIO enable pwm request.
SYS_SET_PIO_ENABLE_EDGE_CAPTURE_CFM	Confirms the set PIO enable edge capture request.
SYS_GET_PIO_EDGE_CAPTURE_CFM	Confirms the get PIO edge capture request.
SYS_SET_PIO_QUADRATURE_DECODER_CFM	Confirms the set PIO quadrature decoder request.
SYS_SET_PIO_QUADRATURE_DECODERS_CFM	Confirms the set PIO quadrature decoders request.
SYS_GET_PIO_QUADRATURE_DECODER_CFM	Confirms the get PIO quadrature decoder request.
SYS_GET_PERSISTENT_MEM_VALID_CFM	Confirms the get persistent memory valid request.
SYS_GET_PERSISTENT_MEM_SIZE_CFM	Confirms the get persistent memory size request.
SYS_GET_PERSISTENT_MEM_CFM	Confirms the get persistent memory request.
SYS_SET_PERSISTENT_MEM_CFM	Confirms the set persistent memory request.
SYS_RESET_PERSISTENT_MEM_CFM	Confirms the reset persistent memory request.
SYS_GET_BUILD_ID_CFM	Confirms the get build id request.
SYS_GET_ROM_BUILD_ID_CFM	Confirms the get rom build id request.
SYS_GET_BATTERY_VOLTAGE_CFM	Confirms the get battery voltage request.
SYS_AIO_DRIVE_CFM	Confirms the AIO drive request.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Enumeration	Description
SYS_AIO_READ_CFM	Confirms the AIO read request.
SYS_AIO_OFF_CFM	Confirms the AIO off request.
SYS_AIO_DIG_CFM	Confirms the AIO dig request.
SYS_READ_APP_PANIC_CODE_CFM	Confirms request to get application panic code.
SYS_CLEAR_APP_PANIC_CODE_CFM	Confirms clearing of application panic code.
SYS_READ_FW_FAULT_ID_CFM	Confirms request to retrtieve fw fault id.
SYS_CLEAR_FW_FAULT_ID_CFM	Confirms clearing of fw fault id.
SYS_GET_BUILD_NAME_CFM	Confirms string identifying build request.
SYS_GET_UCI_VERSION_CFM	Confirms UCI version number.
SYS_GET_BATTERY_LOW_THRESHOLD_CFM	Confirms get value of battery low threshold CS Key request.
SYS_TEST_CHANNEL_MAP_REQ	Requests that random channel map are enabled/ disabled.
SYS_TEST_CHANNEL_MAP_CFM	Confirms that random channel map request.

47.6 gatt_prim.h

47.6.1 Enumerations

47.6.1.1 enum gatt_proc_tag

Syntax

```
enum gatt_proc_tag
```

Description

GATT process type

47.6.2 Typedefs

47.6.2.1 typedef enum gatt_proc_tag GATT_PROC_T

Syntax

```
typedef enum gatt_proc_tag GATT_PROC_T
```

Description

GATT process type

47.7 ls_err.h

47.7.1 Enumerations

47.7.1.1 enum ls_err

Syntax

```
enum ls_err
```

Description

HCI and extended error codes.
Please refer to the BlueTooth specifications V4.0, volume 2, part D for details of the HCI error codes in the range 0x00 - 0x3F.
The extended error codes (those above 0x40) are documented here.

Enumerations

Enumeration	Description
ls_err_arg	0x40 One or more arguements are in error, or incompatible.
ls_err_mode	0x41 Invalid role selected in advertising.
ls_err_lc_buf_full	0x42 Failure due to buffer full condition in LE controller.
ls_err_con_invalid_state	0x43 Message received in Invalid LS Connection State.
ls_err_con_param_rej_remote_dev	0x44 Connection parameter udpate rejected by remote device. This error can only be received in Slave mode.
ls_err_con_param_rej_tgap_violation	0x45 Connection Parameter Update Rejected as Slave device is not allowed to transmit another Connection Parameter Update request till time TGAP(conn_param_timeout). Refer to section 9.3.9.2, Vol 3, Part C of the Core 4.0 BT spec. The application should retry the 'connection paramter update' procedure after time TGAP(conn_param_timeout).
ls_err_con_param_timeout	0x46 Connection parameter udpate procedure timeout - Master device didn't respond to Connection Parameter Update request from Slave device within GAP_TGAP_com_param_proc_timeout (30 secs) period.

47.8 macros.h

47.8.1 Defines

47.8.1.1 COMPILE_TIME_ASSERT

Definition

```
#define COMPILE_TIME_ASSERT (   expr,   msg  )
```

CSR

Description

```
struct compile_time_assert_ ## msg { \
    int compile_time_assert_ ## msg [1 - (!(expr))*2]; \
}
```

47.9 sys_events.h

47.9.1 Enumerations

47.9.1.1 enum sys_event_id

Syntax

```
enum sys_event_id
```

Description

System event codes.

Enumerations

Enumeration	Description
`sys_event_wakeup`	The system was woken by an edge on the WAKE pin. See wakeup_data for associated data.
`sys_event_battery_low`	The system battery voltage has moved above or below the monitoring threshold. See battery_low_data for associated data.
`sys_event_pio_changed`	One or more PIOs specified by PioSetEventMask() have changed input level. See pio_changed_data for associated data.
`sys_event_pio_ctrlr`	An event was received from the 8051 PIO Controller. See pio_ctrlr_data for associated data.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Appendix A Fault Codes

This section contains fault codes found in the `Fault.xml` file.

A.1 ID=0 NONE

Label

```
No Error
```

Description

Marks unused entries in the fault log. Also occasionally useful in test circumstances to indicate success.

A.2 ID=1 MYSTERY

Label

```
An unknown fault
```

Description

Indicates that some unspecified error has occurred. Except in test circumstances a more specific fault code should always be preferred.

A.3 ID=2 BAD_LC_STATE

Label

```
Invalid Link Controller State
```

Description

State machine controlling a BTLE link is in an invalid state.

A.4 ID=3 BUFFER_CORRUPTED

Label

```
A circular buffer is corrupt
```

Description

The internal state of one of the firmware's circular buffers has been corrupted.

Note:

At present the circular buffer subsystem is not in use so this fault should not appear.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

A.5 ID=4 USER_CSKEY_OUT_OF_RANGE

Label

```
User-readable CS key index out of range
```

Description

The CSR1000 device provides a set of eight 16-bit values that can be set when the device is programmed. Applications can read these values using the function `CSReadUserKey()`, supplying an index between 0 and 7 inclusive. This fault indicates that an application supplied an index of 8 or more to `CSReadUserKey()`.

A.6 ID=5 INVALID_LC_INDEX

Label

```
Attempt to use a non-existent Link Controller
```

Description

A CSR1000 device has a fixed number of controllers managing BTLE links. This fault indicates that an attempt was made to access a link controller that does not exist.

A.7 ID=6 H4_RX_BAD_PDU

Label

```
Host Transport reception failed
```

Description

An error was detected while receiving data from a host device.

A.8 ID=7 BAD_FAULT

Label

```
The firmware raised an invalid fault code
```

Description

A fault was raised, but the fault code supplied was not in the valid range. This fault code is used in place of the invalid one.

A.9 ID=8 ADC_TENBIT_TIMEOUT

Label

```
Conversion hardware has failed
```

Description

CSR1000 devices have built-in Analogue to Digital Converters for a number of purposes. This fault indicates that an ADC has become "stuck" in some manner.

A.10 ID=9 WD_TIMER_RESOURCE

Label

```
Background timer could not be claimed
```

Description

CSR1000 devices maintain a background process that performs various vital pieces of system maintenance. This fault indicates that the timer controlling this process could not be acquired because the firmware ran out of resources. It can occur if applications use too many timers for their own control.

A.11 ID=10 HAL_CDAC_TABLE_BUILD

Label

```
Radio failed to initialise correctly
```

Description

This fault indicates that for some reason the radio could not be set up within expected tolerances.

A.12 ID=11 HCI_BUFFER_FULL

Label

```
Host Communications software ran out of resources
```

Description

Failed to send a message over the HCI to the host application because of a resource shortage.

Note:

This error is not reported by default to avoid entering a possible infinite loop.

A.13 ID=12 H4_UNKNOWN_EVENT

Label

```
Host communications asked to send an unknown event
```

Description

This fault indicates that the firmware was asked to send an event to the host which was not recognised as a valid BTLE event.

A.14 ID=13 UNEXP_MSG_RCVD_FROM_ATT

Label

```
Unexpected ATT message during GATT procedure
```

Description

An unexpected message was received from the ATT module while carrying out a GATT procedure.

A.15 ID=14 SA_HNDL_ARRAY_VIOLATION

Label

```
GATT internal state error
```

Description

The internal firmware state driving GATT procedures was found to be in an invalid state.

A.16 ID=15 GATT_CON_DB_FULL_MASTER_ROLE

Label

```
Ran out of GATT connections as a master
```

Description

This fault is raised if a device successfully creates a connection in Master mode but runs out of resources to record it internally.

A.17 ID=16 GATT_CON_DB_FULL_SLAVE_ROLE

Label

```
Ran out of GATT connections as a slave
```

Description

This fault is raised if a device successfully creates a connection in Slave mode but runs out of resources to record it internally.

A.18 ID=17 APPLICATION_PANIC

Label

```
The application called Panic()
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

This fault is raised when the application calls the `Panic()` function. It is not reported by default, to avoid confusing the application further, but does panic the device.

A.19 ID=18 UPDATE_EXCEEDED_RUNTIME

Label

```
A firmware background task overran its allotted time
```

Description

Background firmware tasks must run to tight timescales to avoid disrupting radio traffic and breaking the specification. This fault is raised when a task exceeded its allotted time, and will consequently have interfered with existing connections in an unpredictable manner.

A.20 ID=19 INTERRUPT_UNBLOCK

Label

```
Internal process management error
```

Description

The firmware's internal interrupt management state has become inconsistent. Attempting to report this fault is probably futile, so by default it simply panics.

A.21 ID=20 L2CAP_HANDLER_NOT_REGISTERED

Label

```
No handler has been registered for a used L2CAP service
```

Description

The L2CAP code relies on a handler function being registered with it for each of the L2CAP services being used. This fault indicates that a call has been made to a particular service for which no handler has been registered. This may suggest that the L2CAP initialiser function, `l2cap_init`, has not been called.

A.22 ID=21 HIBERNATE_TIME_TOO_SHORT

Label

```
Application didn't request a large enough hibernation duration
```

Description

When the application requests the CSR1000 device to move to the Hibernate state it has to provide the minimum time spent hibernating. This time should be at least 2^20 microseconds (1.048576s). If the supplied time is too short this fault will be raised.

© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

A.23 ID=22 LS_INVALID_CONNECTION

Label

```
Firmware out of resources allocating a connection
```

Description

Upper layers need to allocate resources to BLE connections as they are established. These resources should always be available; this fault indicates that a major firmware error has caused them to be unavailable.

A.24 ID=23 SM_UNEXPECTED_CID

Label

```
Security Manager was given an implausible CID by L2CAP
```

Description

The Security Manager has been asked to handle security through a Channel ID that is not the fixed CID reserved for it.

A.25 ID=24 ATT_UNEXP_MSG_RCVD_FROM_L2CAP

Label

```
Unexpected ATT message received from L2CAP
```

Description

This fault indicates that an unexpected message is received by ATT module from L2CAP.

A.26 ID=25 SLOW_CLOCK_FREQ_TRIM

Label

```
Unable to trim 32kHz frequency
```

Description

The firmware was unable to complete the trim procedure for the 32kHz slow clock frequency, when trimmed against the 16MHz clock.

A.27 ID=26 INVALID_UART_BUFFER_SIZE

Label

```
Invalid UART buffer size
```


© Qualcomm Technologies International, Ltd. 2014-2016
Confidential and Proprietary – Qualcomm Technologies International, Ltd.

Description

The application requested an invalid buffer size for the UART RX or UART TX buffer. Supported buffer sizes are defined by the `uart_buf_size_bytes` enumeration in `uart.h`.

A.28 ID=27 FW_TIMER_RESOURCES_EXHAUSTED

Label

```
Firmware timer resources exhausted
```

Description

The firmware library tried to allocate an internal timer but did not have any free timer resources.

A.29 ID=28 INVALID_UART_CONSUMPTION

Label

```
Invalid UART consumption
```

Description

The application claims to have consumed more data from the UART RX buffer than was available, causing a receive buffer underflow.

A.30 ID=29 INCORRECT_ROM_VERSION

Label

```
Incorrect ROM version
```

Description

The ROM version is not compatible with the SDK used to build the application.