Push every boundary.™

CSR µEnergy®

Mouse

Application Note

Issue 7

Document History

Revision	Date	History
1	06 AUG 12	Original publication of this document
2	20 DEC 12	Updated to reference CSR1010 and CSR1011 devices, and added information on the new timer based approach in sending periodic mouse data.
3	20 DEC 12	Improved the CSR1011 development board picture quality
4	08 FEB 13	Updated for SDK 2.1
5	20 FEB 13	Updated current consumption values
6	10 MAY 13	Updated for SDK 2.2; connection parameter update as recommended by CSA3.
7	04 JAN 14	Updated to new CSR branding

Contacts

General information	www.csr.com
Information on this product	sales@csr.com
Customer support for this product	www.csrsupport.com
More detail on compliance and standards	product.compliance@csr.com
Help with this document	comments@csr.com

Trademarks, Patents and Licences

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

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

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 CSR plc or its affiliates.

CSR 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, CSR cannot accept responsibility for any errors.

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

Safety-critical Applications

CSR'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 CSR's products (or supply CSR's products for use) in such devices or systems.

Performance and Conformance

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

Contents

Tables, Figures and Equations

1. Introduction

This document describes the mouse application supplied with the CSR µEnergy® Software Development Kit (SDK) and provides guidance to developers on how to customise the on-chip application.

The application demonstrates the HID over GATT profile which is specified by the Bluetooth SIG.

1.1. Application Overview

1.1.1. Profiles Supported

The Mouse application supports the Bluetooth HID over GATT profile.

1.1.1.1. HID over GATT Profile

The HID over GATT profile defines protocols, procedures and features to be used by HID devices and HID hosts using the GATT profile. It defines the two roles described in Table 1.1.

Role	Description
HID Device	A HID device is a device that sends Input reports and associated HID data to a HID host.
HID Host	A HID host is a device that receives Input reports from HID devices.

Table 1.1: HID Profile Roles

For more information about the HID over GATT profile, see *HID over GATT Profile Specification Version 1.0*.

1.1.2. Application Topology

The Mouse application implements the HID over GATT profile in the HID Device role, see Table 1.2.

Role	HID over GATT Profile	GAP Service	GATT Service	Device Information Service	Battery Service	Scan Parameters Service
GATT Role	GATT Server	GATT Server	GATT Server	GATT Server	GATT Server	GATT Server
GAP Role	Peripheral	Peripheral	Peripheral	Peripheral	Peripheral	Peripheral

Table 1.2: Application Topology

Role	Responsibility
GATT Server	It accepts incoming commands and requests from the client and sends responses, indications and notifications to the client.
GAP Peripheral	It accepts connection requests from the remote device and acts as a slave in the connection.

Table 1.3: Responsibilities

For more information about GATT server and GAP peripheral, see *Bluetooth Core Specification Version 4.1*.

1.1.3. Services

This application exposes the following services:

- HID (Version 1.0)
- Device Information (Version 1.1)
- Battery (Version 1.0)
- Scan Parameters (Version 1.0)
- GAP
- GATT

HID over GATT profile mandates three services: Human Interface Device (HID), Device Information, and Battery. GAP and GATT services are mandated by the *Bluetooth Core Specification Version 4.1*. Scan Parameter service is optional as shown in Figure 1.1.

For more information on HID, Device Information, Battery and Scan parameters services, see *HID Service Specification Version 1.0*, *Device Information Service Specification Version 1.1*, *Battery Service Specification Version 1.0* and *Scan Parameters Service Specification Version 1.0* respectively. For more information on GATT and GAP services, see *Bluetooth Core Specification Version 4.1*.

See 8.Appendix B for information on characteristics supported by each service.

Note:

The Mouse application does not support any characteristic for GATT service. See *Bluetooth Core Specification Version 4.1* for more information.

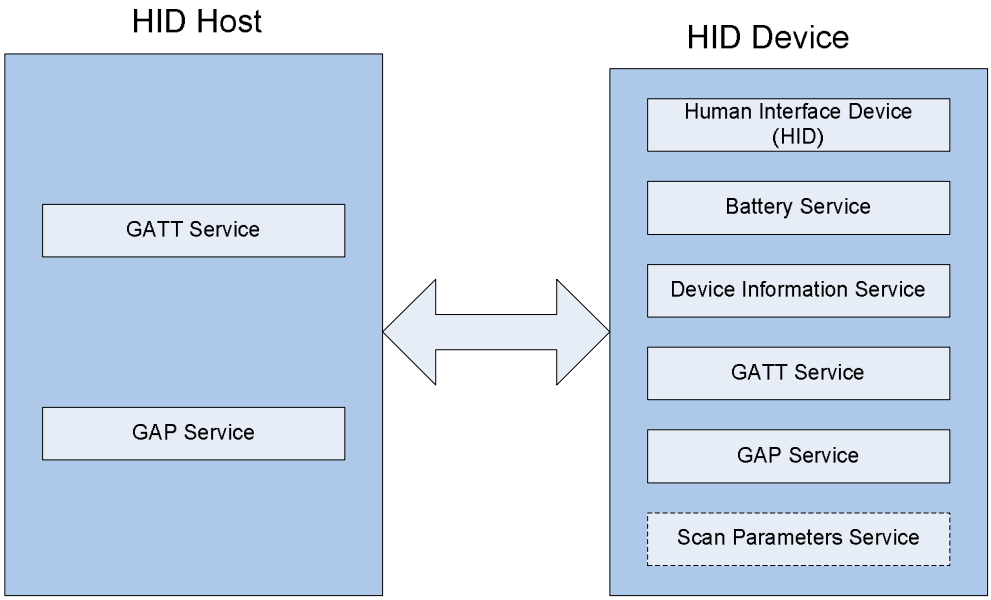

Figure 1.1: Primary Services

2. Using the Application

This section describes how the Mouse application can be used with the CSR µEnergy Profile Demonstrator application.

2.1. Demonstration Kit

The application can be demonstrated using the following components:

Component	Hardware	Application
HID Device	CSR Laser Mouse Example Design	Mouse application
HID Device	CSR10x1 Development Board	Mouse Application
HID Host	CSR µEnergy BT4.0 USB dongle	CSR µEnergy Profile Demonstrator Application

Table 2.1: Demonstration Components

The CSR µEnergy Profile Demonstrator application, CSR device driver and installation guide are included in the SDK.

2.1.1. HID Device

The SDK is used to build and download the mouse application to designs which include CSR10xx devices.

The application code uses the lower range of PIOs and can be demonstrated on the CSR1011 development board and laser mouse example design without modification. If a CSR1001 development board is used, the PIOs require re-mapping as they are not exposed by the 32-pin connector. See the *CSR µEnergy xIDE User Guide* for further information.

Figure 2.1 shows a CSR1001 development board with the switch set to receive power from the mini-USB connector when debugging and downloading the firmware image.

During normal use, set the switch to the **Batt** position to use the coin cell battery. Figure 2.2 shows the same for a CSR1011 development board with the switch set to receive power from the battery.

2.1.1.1. To power on the device:

- Ensure the power source is provided i.e. either the mini-USB cable is attached to the USB to SPI adapter (connected to the PC), or the coin cell battery is fitted.
- Ensure that the corresponding power source is selected using the power slider switch.

2.1.1.2. To power off the device, either:

- Remove the power source (currently selected by the power slider switch) i.e. by disconnecting the mini-USB cable, or removing the battery. (or)
- Ensure the power slider switch is set to a power source that is not provided. Figure 2.1 shows the switch in the USB position.

Note:

When the mini-USB cable has been disconnected, wait at least 1 minute before switching the board to receive power from the coin cell. This allows any residual charge received from the SPI connector to be dissipated.

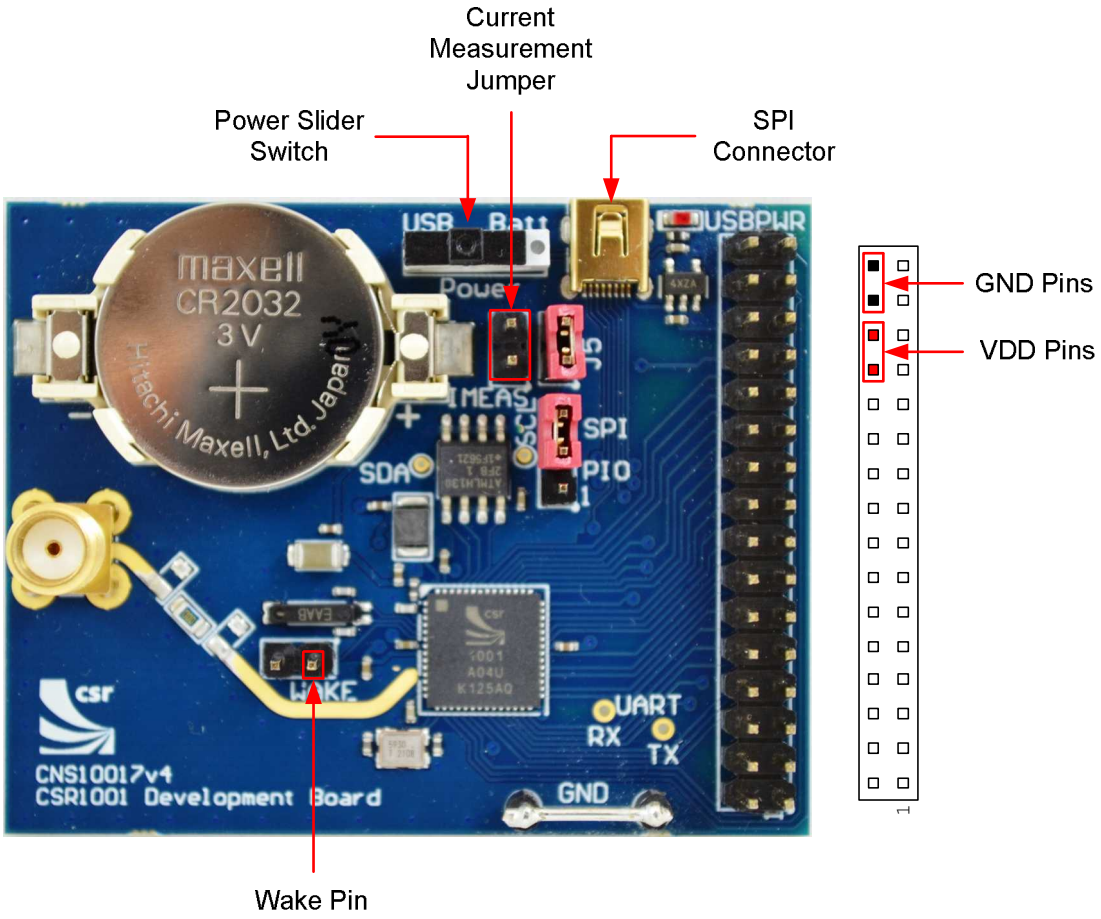

Figure 2.1: CSR1001 Development Board

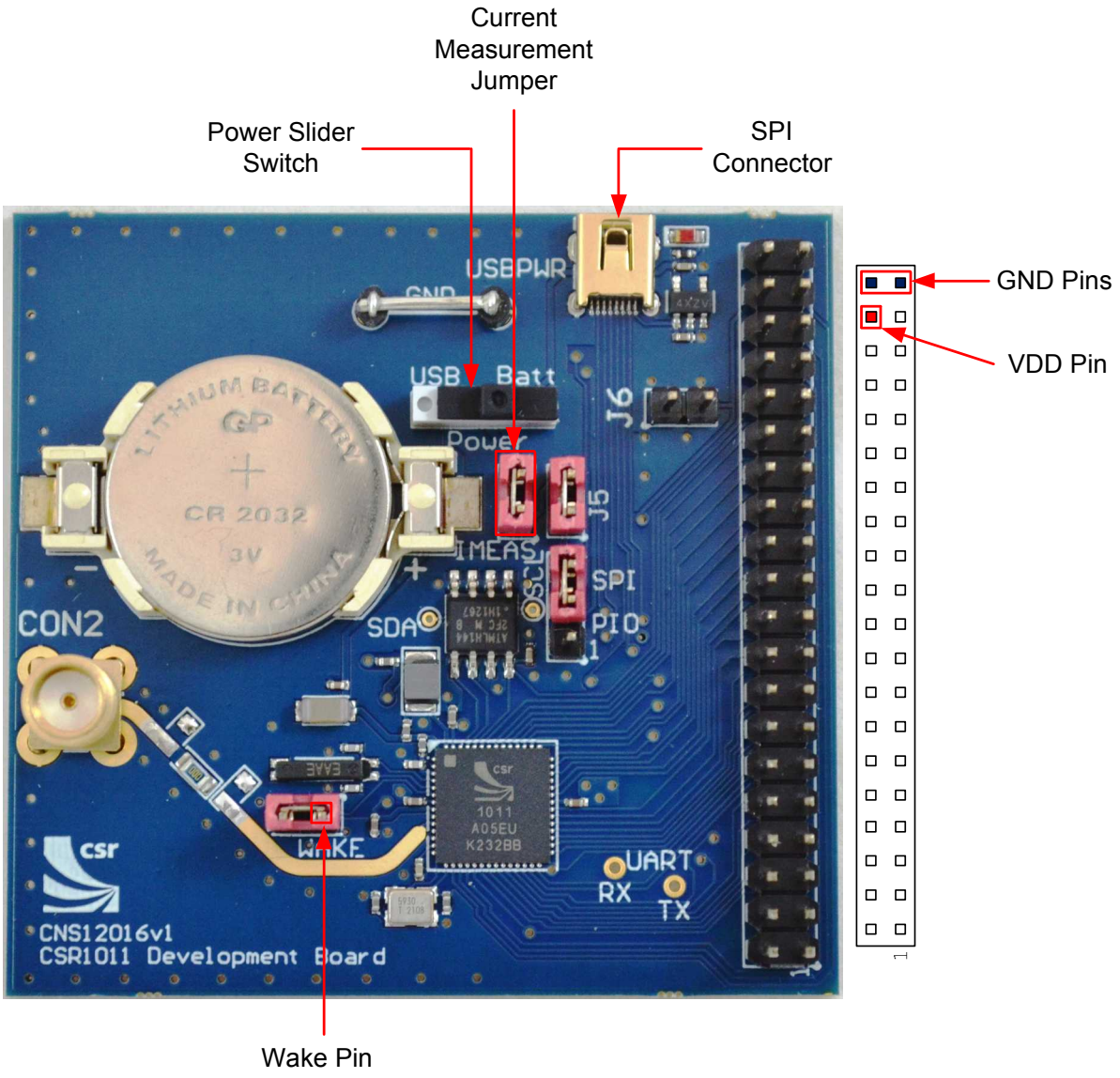

Figure 2.2: CSR1011 Development Board

2.1.1.3. User Interface

The Mouse application has been developed for hardware where the WAKE pin is mapped to a PIO which is active low.

However, the WAKE pin on the development board is, by default, connected to ground using a jumper. This jumper must be removed and the WAKE pin must be connected to a VDD pin on the development board, see Figure 2.1 and Figure 2.2 which show the VDD pins on the CSR10x1 development boards.

Figure 2.3 shows the detailed hardware configuration for the Mouse application and Table 2.4 shows the hardware mapped to the pins exposed on the CSR1011 development board.

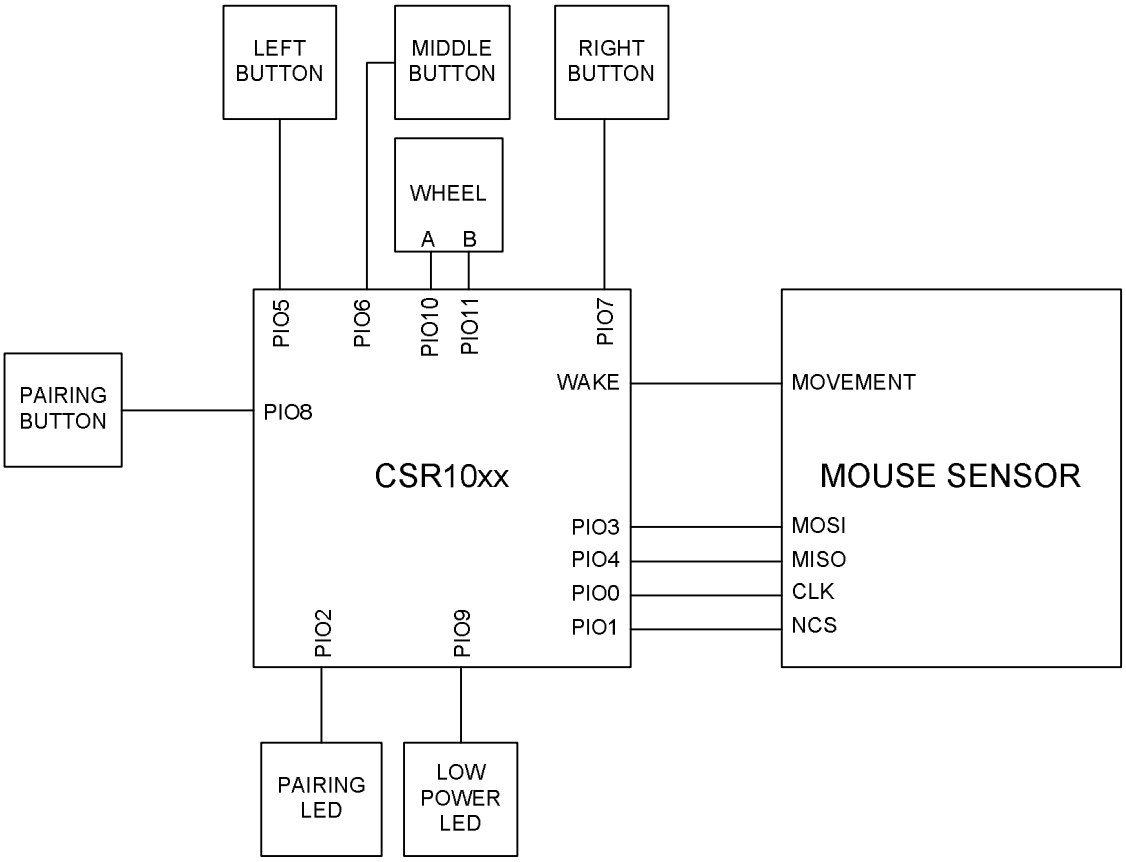

Figure 2.3: Hardware Configuration for the Mouse Application

CSR1011 PIO Pinout	Mouse Assignment	CSR1011 Dev Board Connector Pinout		CSR1011 Dev Board Connector Pinout	Mouse Assignment	CSR1011 PIO Pinout
GND		40		39		GND
VDD_PAD		38		37		AIO2
I2C_SDA		36		35		AIO1
I2C_SCL		34		33		AIO0
PIO2	PAIR LED	32		31		PIO31
PIO30		30		29		PIO29
PIO28		28		27		PIO27
PIO26		26		25		PIO25
PIO24		24		23		PIO23
PIO22		22		21		PIO21
PIO8/SPI_MISO	PAIRING	20		19	RIGHT	PIO7/SPI_MOSI
PIO20		18		17		PIO19
PIO6/SPI_CSB	MIDDLE	16		15		PIO18
PIO17		14		13	LEFT	PIO5/SPI_CLK
PIO16		12		11		PIO15
PIO14		10		9		PIO13
PIO1/UART_RX	NCS	8		7	CLK	PIO0/UART_TX
PIO12		6		5	WHEELB	PIO11
PIO10	WHEEL A	4		3	LED PWR	PIO9
PIO4/SF_CSB	MISO	2		1	MOSI	PIO3/SF_DOUT
			1			

Table 2.2: Mapping of the Pins Exposed on CSR1011 Development Board to the Buttons and LEDs

Simulated Mouse Movement

Mouse motion can be simulated on the development board by connecting the WAKE pin to GND.

Simulated Mouse Button Press

The left, middle and right buttons of the mouse are mapped to PIO5, PIO6 and PIO7 respectively in `mouse_hw.c`. These PIOs are exposed on the CSR1011 development board but as these are not exposed on the CSR1001 development board, the PIO assignment in `mouse_hw.c` can be modified such that the left, middle and right buttons of a mouse are mapped to one of the exposed PIOs, e.g. PIO12, PIO13 and PIO14 on the development board.

The PIOs configured as mouse buttons can be connected to one of the GND pins on the development board to simulate a button press.

Figure 2.4 shows the state transitions on user action, i.e. mouse motion and button presses.

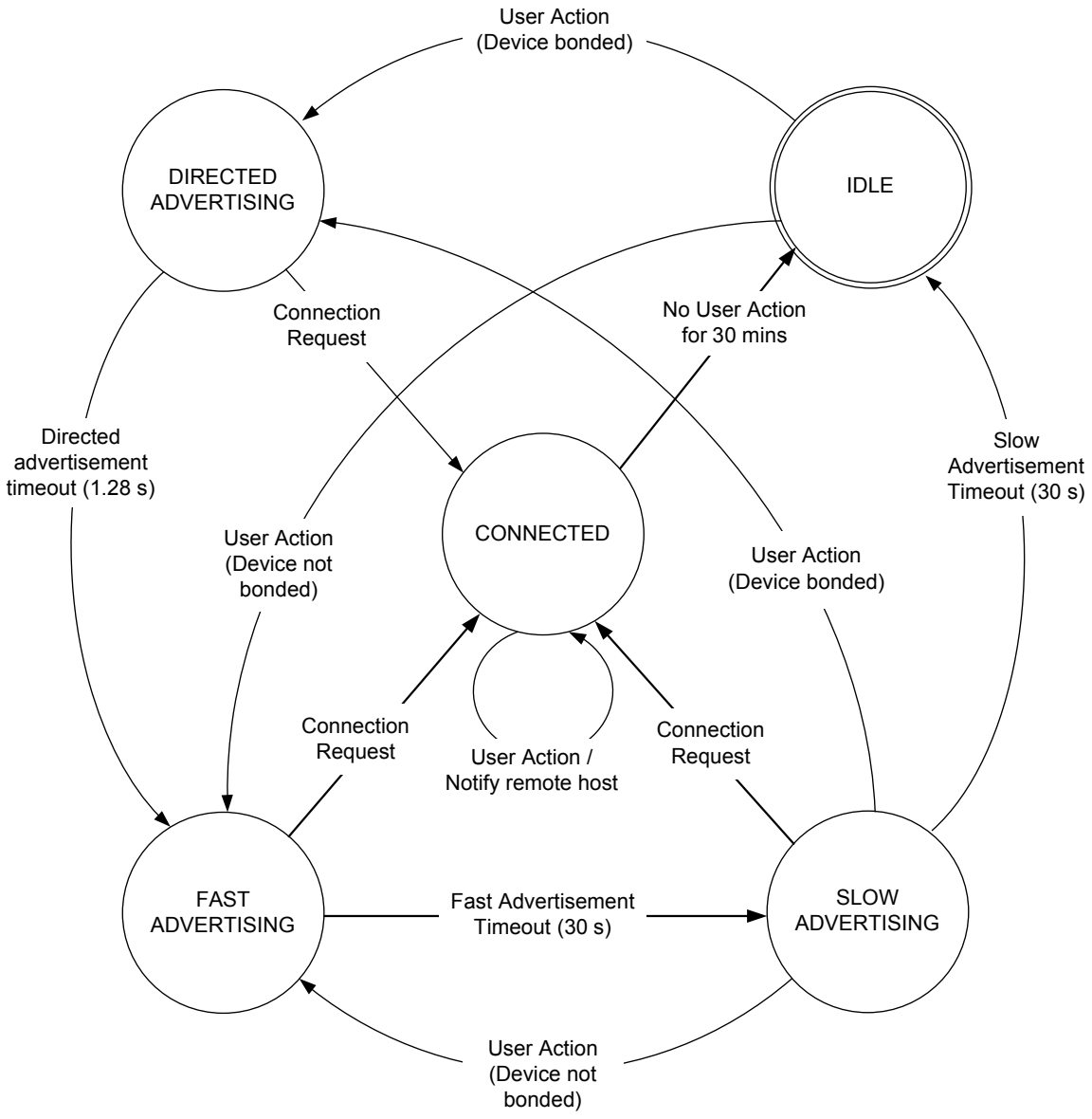

Figure 2.4: State Changes by User Action on the Mouse

Pairing Button Press Behaviour

The mouse application configures PIO8 as the pairing removal button. The pairing button functionality can be tested on a CSR1011 development board by shorting the PIO8 (pin 20) with ground for more than 1 second. Since PIO8 is not exposed on the CSR1001 development board, this functionality cannot be tested directly. See section 7.7 for details on how to modify the Mouse application to test this functionality on a CSR1001 development board.

The **Pairing Button Press** event is registered when the **Pairing** button is pressed and held down for a period of 1 second. On registering this event,

- If in CONNECTED/ACTIVE state, the Mouse application removes the bonding information and triggers a disconnection with the remote HID Host. When the disconnection is complete, the application triggers fast advertisements.

- If in the IDLE state, the Mouse application removes the bonding information and triggers fast advertisements.
- If in the ADVERTISING state, the Mouse application removes the bonding information and starts fast advertisements.

See Figure 2.5 for the pairing removal behaviour.

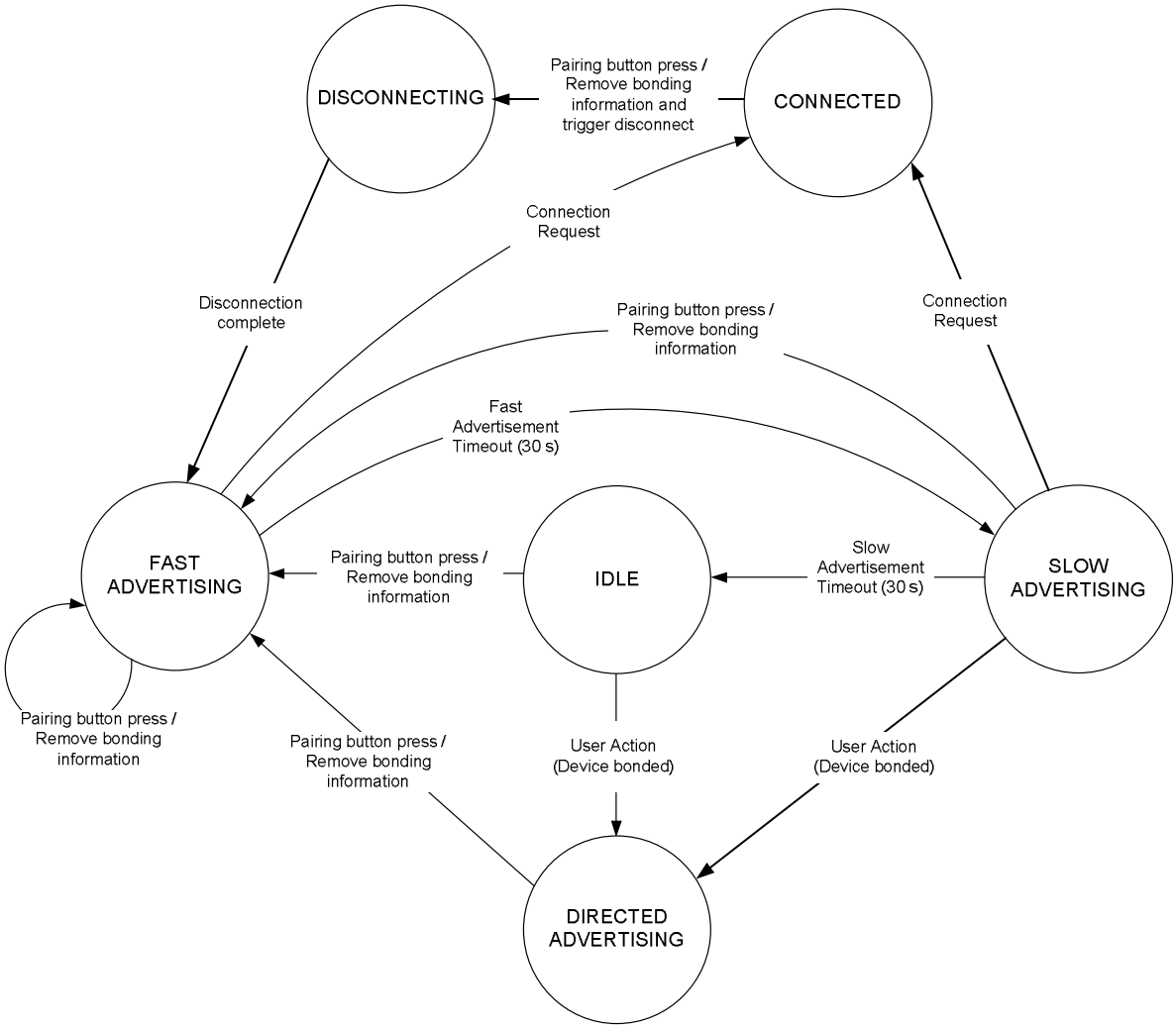

Figure 2.5: Pairing Button Press Behaviour

2.1.2. HID Host

2.1.2.1. CSR µEnergy BT4.0 USB Dongle

The CSR µEnergy BT4.0 USB dongle shown in Figure 2.6 must be used with the CSR µEnergy Profile Demonstrator application to complete the Bluetooth Smart link between two devices. To use the USB dongle, the default USB Bluetooth Windows device driver must be replaced with the CSR BlueCore device driver as described in the *Installing the CSR Driver for the Profile Demonstrator Application* user guide.

The CSR Laser Mouse example design schematic is available to download from www.csrsupport.com/uEnergy.

Figure 2.6: CSR µEnergy BT4.0 USB Dongle

2.1.2.2. CSR µEnergy Profile Demonstrator Application

The CSR µEnergy Profile Demonstration application is compatible with a PC running Windows XP, Windows 7 (32-bit and 64-bit) and Windows 8 (32-bit and 64-bit). Launch the application once the CSR8510 USB dongle is attached to the PC and the driver has been loaded.

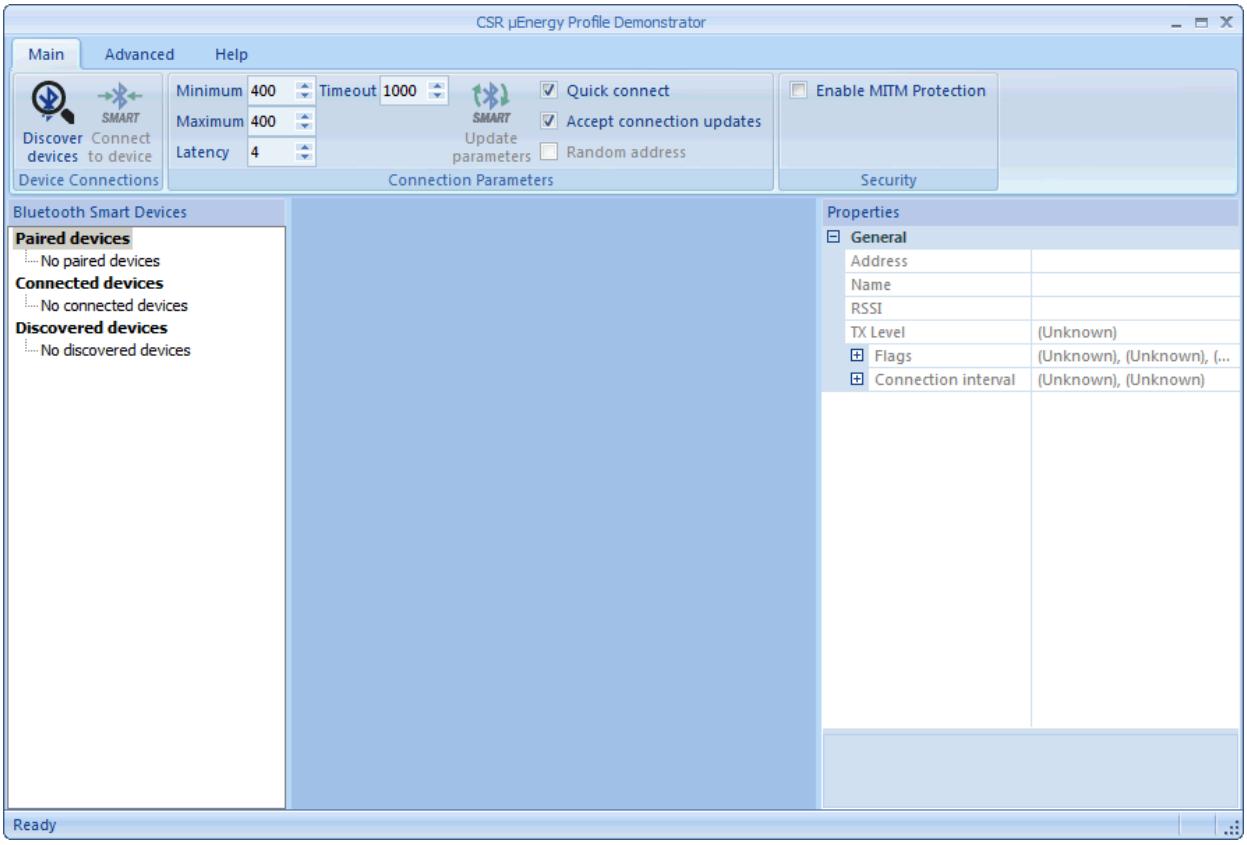

Figure 2.7: CSR µEnergy Profile Demonstrator

2.2. Demonstration Procedure

1. After the Mouse application has been downloaded to the development board, switch on the development board and it will trigger advertisements.
2. Click on the **Discover devices** button in the **CSR µEnergy Profile Demonstrator** window. The software searches for Bluetooth Smart devices and lists all the discovered devices on the left hand side of the application window see Figure 2.8.

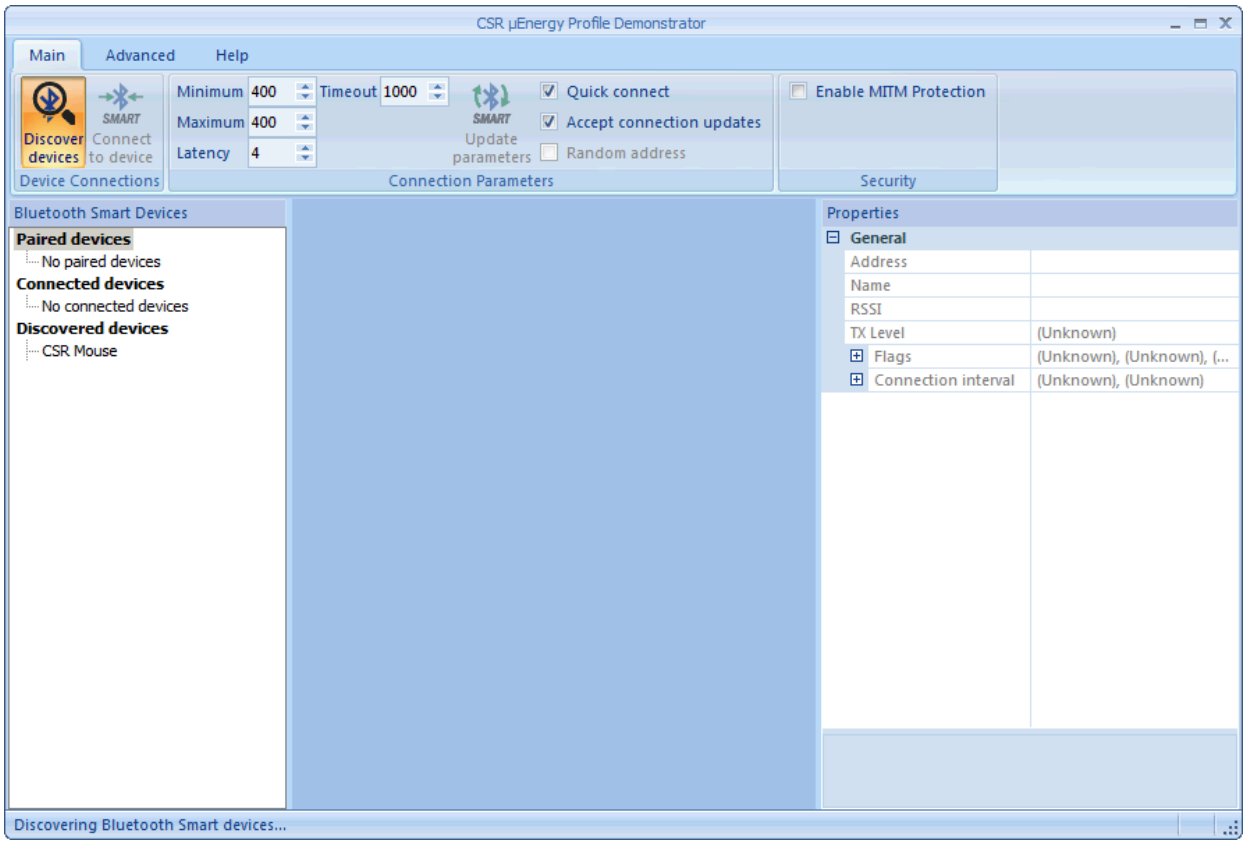

Figure 2.8: Mouse Device Discovered

3. When the device labelled **CSR Mouse** appears, select it and its device address appears on the right hand side of the screen.
4. Enter the preferred connection parameters from Table 7.3 in the **Connection Parameters** tab. To connect to this device, click on the **Connect to device** button.
5. The CSR µEnergy Profile Demonstrator application displays a tabbed pane corresponding to different services supported by the Mouse application, see Figure 2.9.

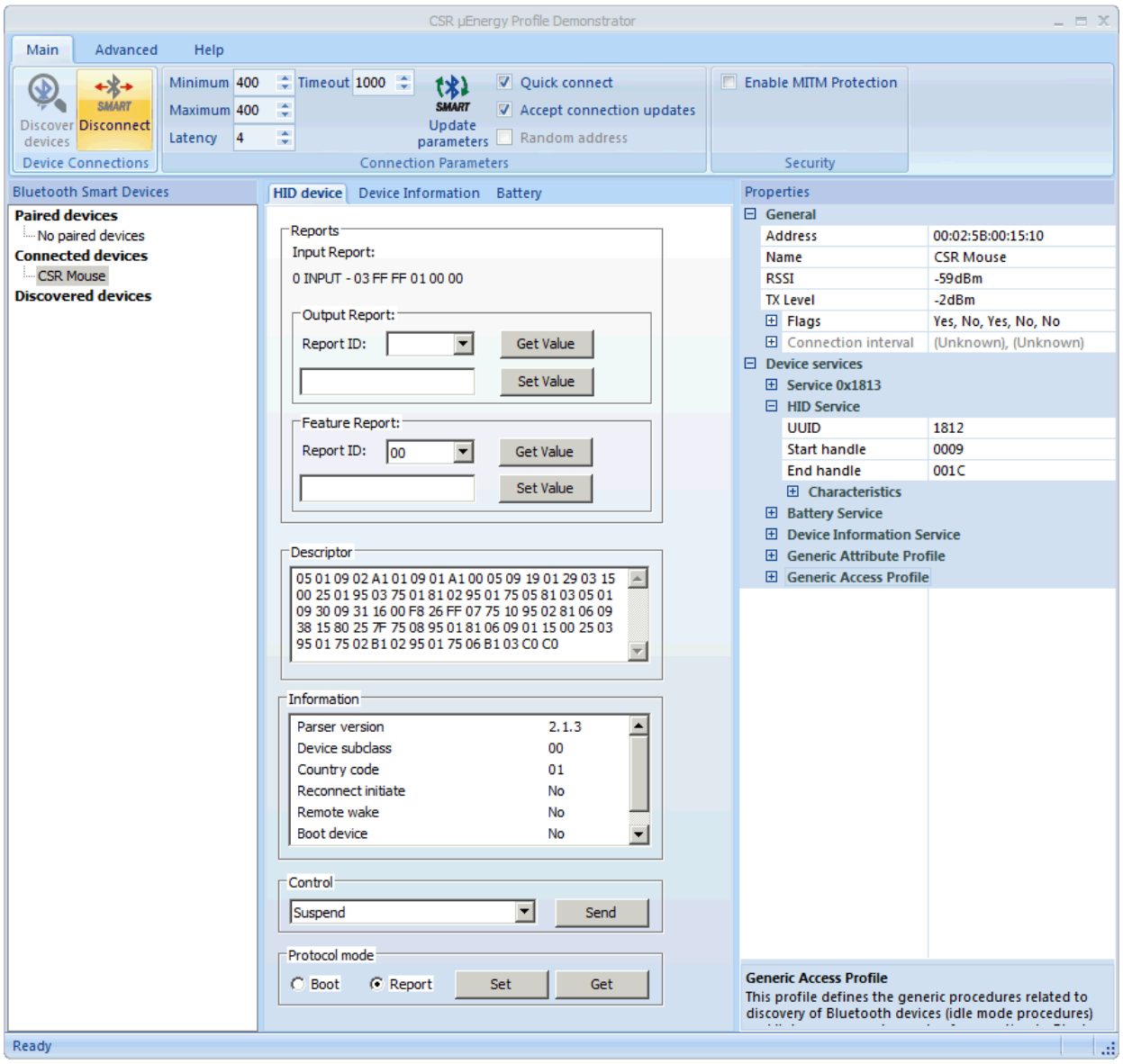

Figure 2.9: Device Connected

6. The **HID device** tab in the **CSR µEnergy Profile Demonstrator** window shows five sections:

6.1. **Reports**

This section shows the latest HID Input report received from the Mouse application.

6.2. **Descriptor**

This section shows the HID Descriptor used by the Mouse application.

6.3. **Information**

This section shows the HID Information supported by the Mouse application.

6.4. **Control**

This section is used to trigger Control Point operations on the connected Mouse application by selecting the required operation from the drop down box. See *HID Service Specification Version 1.0* for more information on HID Control Point operations.

6.5. **Protocol Mode**

This section is used to change the protocol mode of the connected Mouse application by selecting the appropriate radio button. See *HID Service Specification Version 1.0* for more information on HID Protocol modes.

7. The **Battery** tab shown in Figure 2.10 displays the current state of the battery.

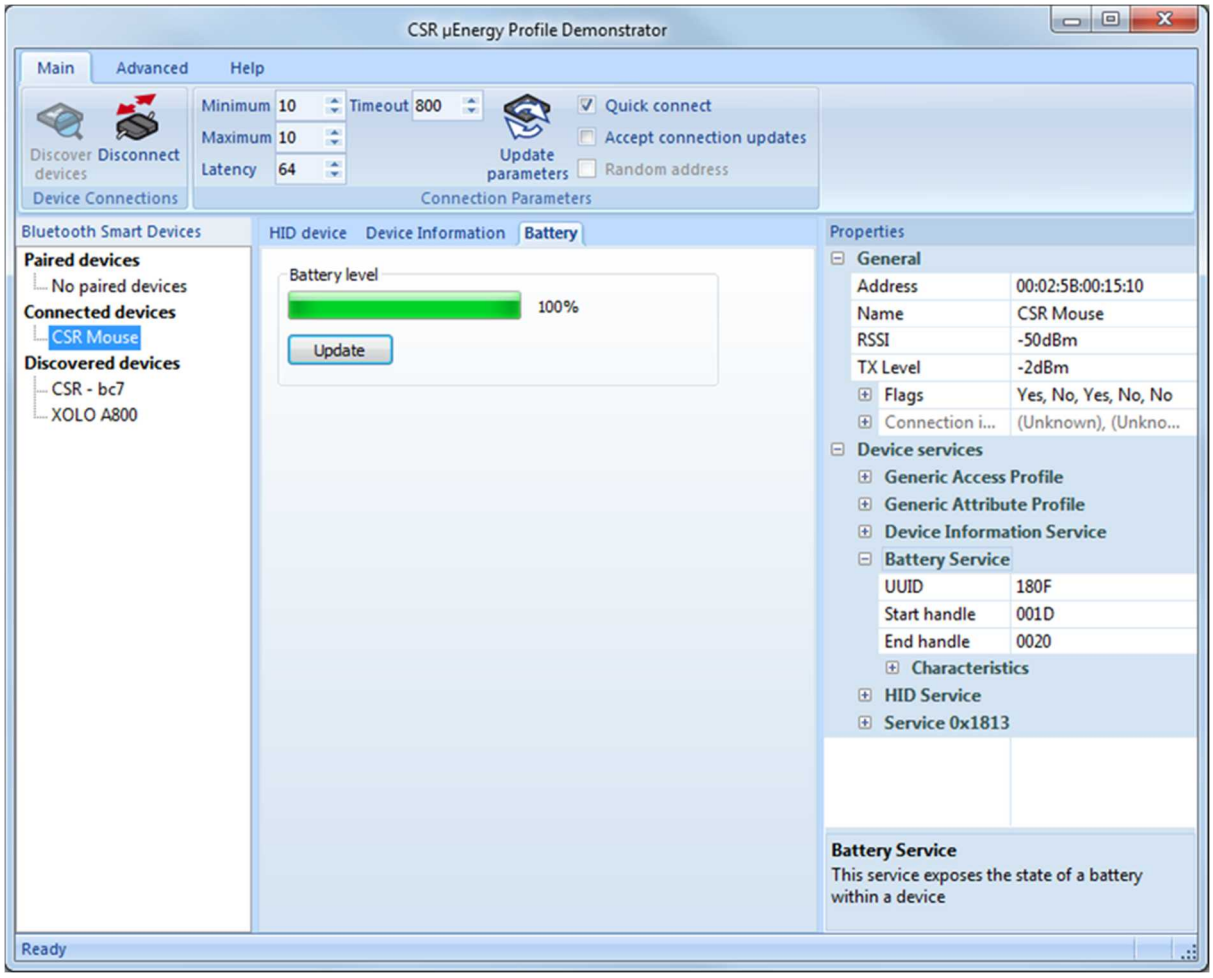

Figure 2.10: Battery Level

Note:

The battery level may fluctuate depending on the connection state.

8. Figure 2.11 shows the **Device Information** tab which displays the device information characteristics of the application.

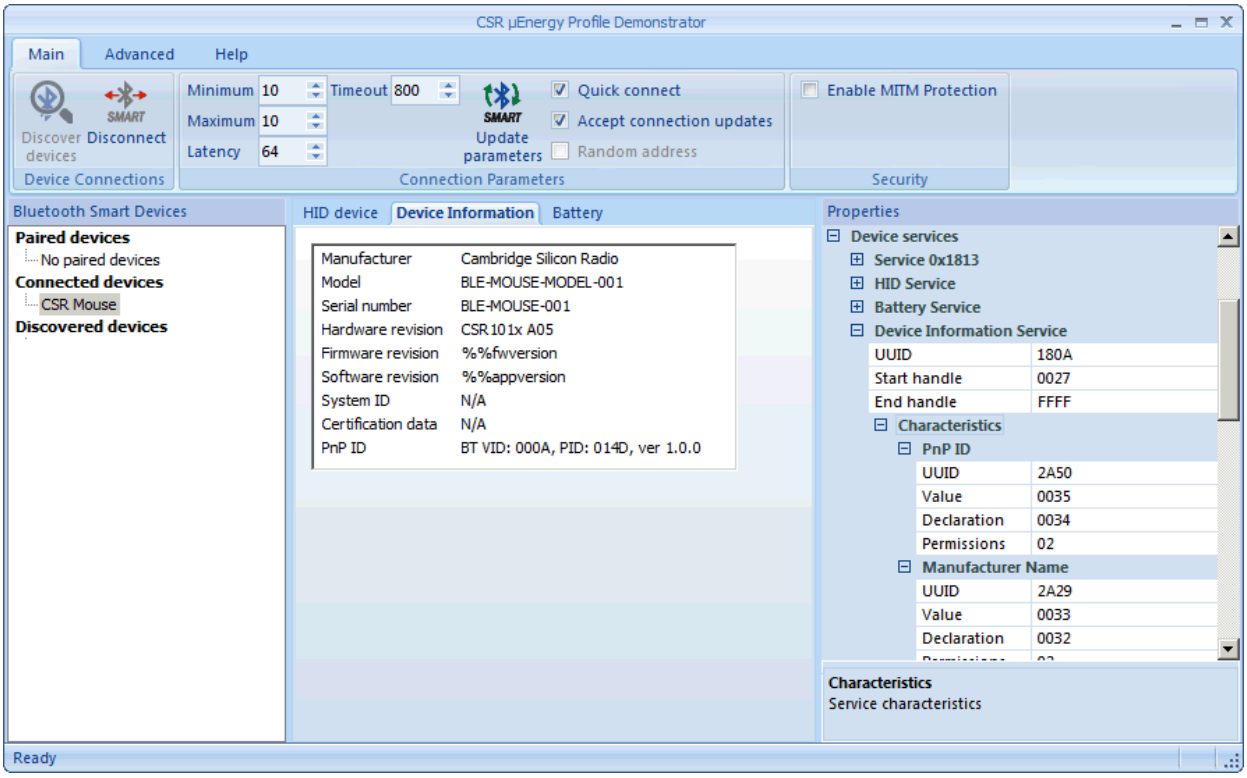

Figure 2.11: Device Information Service Information

9. Click the **Disconnect** button to disconnect the Bluetooth Smart link between the device and the USB dongle.

3. Application Structure

3.1. Source Files

Table 3.1 lists the source files and their purpose.

File name	Purpose
mouse.c	Implements all the entry functions e.g. AppInit(), AppProcessSystemEvent() and AppProcessLmEvent(). Events received from the hardware and firmware are first handled here. This file contains handling functions for all the LM and system events.
mouse_gatt.c	Implements routines for triggering advertisement procedures.
mouse_hw.c	Implements routines for hardware initialisation, key press, pairing button press handling and LEDs when output reports are written.
hid_service.c	Implements routines required for HID service e.g. Input report notifications corresponding to mouse motion data to the remote device, and handling read/write access indications on the HID service specific ATT attributes.
scan_param_service.c	Implements routines required for storing the scan interval and scan window used by the HID host.
battery_service.c	Implements routines required for the Battery service e.g. reading Battery Level, notifying it to the remote device and handling access indications on the Battery service specific ATT attributes.
gap_service.c	Implements routines for the GAP service e.g. handling read/write access indication on the GAP service characteristics, reading/writing device name on NVM etc.
nvm_access.c	Implements the NVM read/write routines.
sensor.c	Implements all the routines for interacting with the sensor used in the mouse. The routines are for initialising the sensor, in addition to accessing and modifying the settings of the registers of the sensor.

Table 3.1: Source Files

3.2. Header Files

Table 3.2 lists the header files and their purpose.

File name	Purpose
mouse.h	Contains application data structure definition and prototypes of externally referred functions defined in mouse.c file.
mouse_gatt.h	Contains timeout values for fast/slow advertising and prototypes of externally referred functions defined in mouse_gatt.c file.
mouse_hw.h	Contains prototypes of externally referred hardware routines of mouse_hw.c file.
app_gatt.h	Contains macro definitions, user defined data type definitions and function prototypes which are being used across the application.
appearance.h	Contains the appearance value macro of the mouse application.
battery_service.h	Contains prototypes of externally referred functions defined in battery_service.c file.
battery_uuids.h	Contains macro definitions for UUIDs of the Battery service and related characteristics.

File name	Purpose
dev_info_uuids.h	Contains macros for UUID values of the Device Information service.
gap_conn_params.h	Contains macro definitions for fast/slow advertising, preferred connection parameters, maximum number of connection parameter update requests etc.
gap_service.h	Contains prototypes of the externally referred functions defined in gap_service.c file.
gap_uuids.h	Contains macro definitions for UUIDs of the GAP service and related characteristics.
gatt_serv_uuid.h	Contains macros for UUID values of the GATT service.
hid_service.h	Contains macro values for the values that can be taken by specific characteristics of HID service and prototypes of externally referred functions defined in hid_service.c file.
hid_uuids.h	Contains macro definitions for UUIDs of the HID service and related characteristics.
nvm_access.h	Contains prototypes of externally referred NVM read/write functions defined in nvm_access.c file.
scan_param_service.h	Contains macro values for the values that can be taken by specific characteristics of scan parameters service and prototypes of externally referred functions defined in scan_param_service.c file.
scan_parameters_uuids.h	Contains macro definitions for UUIDs of the scan parameters service and related characteristics.
sensor.h	Contains macro values for the configurable parameters of the sensor and prototypes of externally referred functions defined in sensor.c file
user_config.h	Contains macro values which would normally be changed when implementing the mouse for a different vendor.

Table 3.2: Header Files

3.3. Database Files

The SDK uses database files to generate attribute database for the application. For more information on how to write database files, see the *GATT Database Generator User Guide*.

Table 3.3 lists the database files and their purpose.

File Name	Purpose
app_gatt_db.db	Master database file which includes all service specific database files. This file is imported by the GATT Database Generator.
battery_service_db.db	Contains information related to Battery service characteristics, their descriptors and values. See Table B.1 for more information on Battery service characteristics.
dev_info_service_db.db	Contains information related to Device Information service characteristics, their descriptors and values. See Table B.2 for Device Information service characteristics.
gap_service_db.db	Contains information related to GAP service characteristics, their descriptors and values. See Table B.3 for GAP characteristics.
gatt_service_db.db	Contains information related to GATT service characteristics, their descriptors and values.
hid_service_db.db	Contains information related to HID service characteristics, their descriptors and values. See Table B.4 for HID service characteristics.
scan_parameters_db.db	Contains information related to Scan Parameters service characteristics, their descriptors and values. See Table B.5 for Scan Parameters service characteristics.

Table 3.3: Database Files

3.4. Assembler Code

3.4.1. Quad_decoder.asm

The PIO controller implements a quadrature decoder for the mouse wheel. When the mouse wheel is scrolled down (clockwise rotation) by one tick, the state of A and B pins changes sequentially, see Table 3.4

A	B
0	0
0	1
1	1
1	0

Table 3.4: A-B State Change for Clockwise Rotation

Figure 3.1 shows the waveform representation of the phase variation of the A and B pins when the scroll wheel is rotated clockwise.

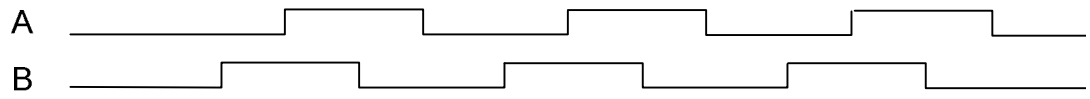

Figure 3.1: State Change of A and B Pins for Clockwise Rotation

When the mouse wheel is scrolled up (counter-clockwise rotation) by one tick, the state of the A and B pins changes sequentially see Table 3.5.

A	B
1	0
1	1
0	1
0	0

Table 3.5: A-B State Change for Counter-clockwise Rotation

Figure 3.2 shows the waveform representation of the phase variation of the A and B pins when the scroll wheel is rotated in the counter-clockwise direction.

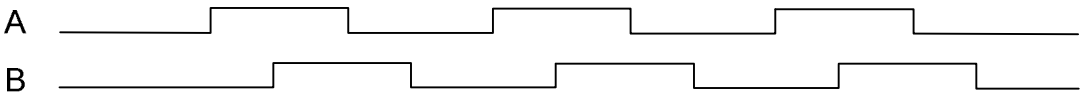

Figure 3.2: State change of A and B pins for Counter-clockwise Rotation

The states of the A and B PIOs are first de-bounced with an interval of 3.2 milliseconds to filter out any noise. These state changes are then used to detect whether the wheel is moving in a clockwise or an anti-clockwise direction, see Figure 3.3 in which table states are shown in bold.

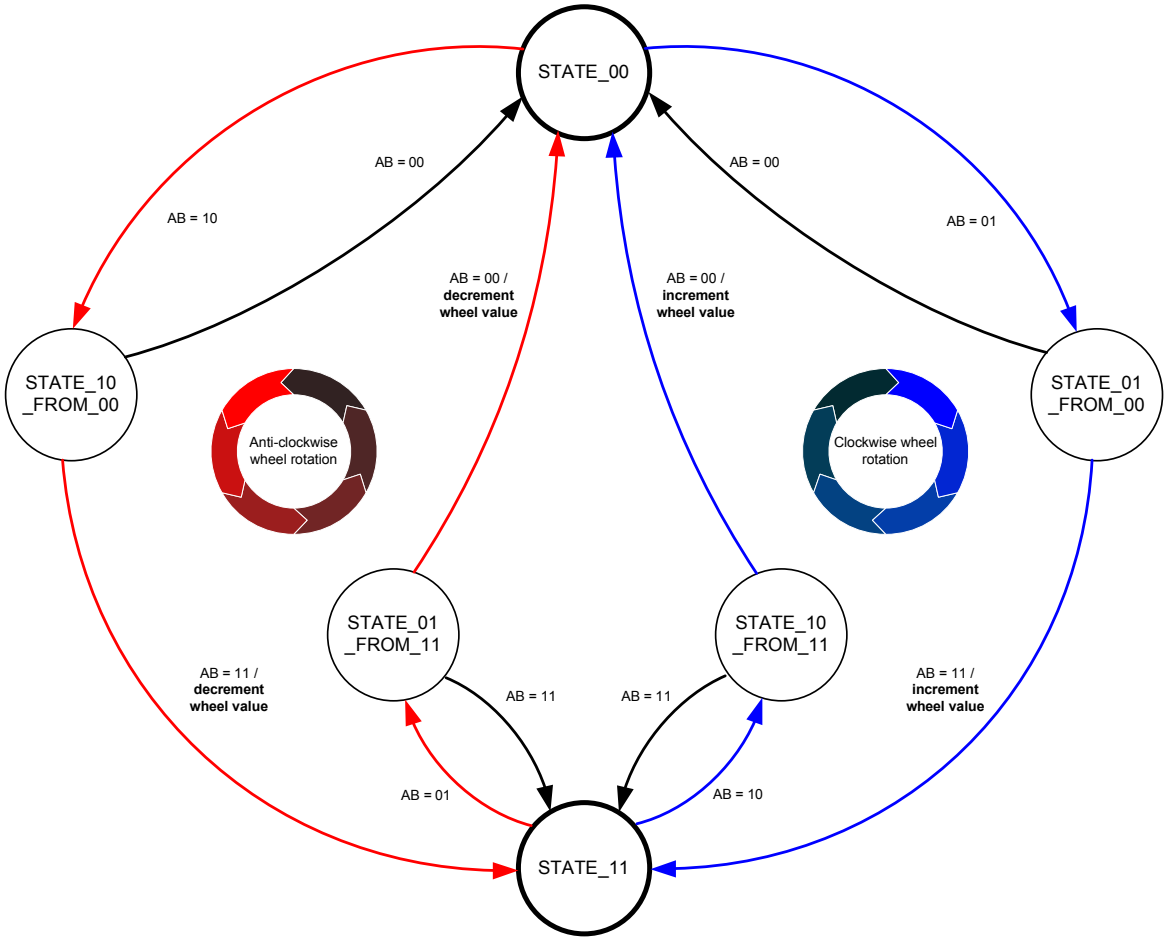

Figure 3.3: Quadrature Decoder Algorithm State Diagram

Three contiguous memory locations in the shared memory region between the main processor and the PIO controller (quadrature decoder) are used for the handshake mechanism between both devices, see Figure 3.4.

WHEEL VALUE	ENABLE INTERRUPT	RESET COUNTER

Figure 3.4: Memory Map for Handshake Mechanism

Description of Memory Locations:

- **WHEEL VALUE**: Current `WHEEL VALUE` with base value of `0x7f`.
- **ENABLE INTERRUPT**: When set to `1` by the application, the PIO controller sends an interrupt to the main processor when the `WHEEL VALUE` changes.
- **RESET COUNTER**: When set to `1` by the application, the quadrature decoder resets the `WHEEL VALUE` to `0x7f`.

When the quadrature decoder increments/decrements the `WHEEL VALUE`, the main processor is awoken if interrupts are configured, i.e. `ENABLE INTERRUPT` is set. The main processor sends the wheel data read from the `WHEEL VALUE` to the connected HID host. After reading the scroll wheel data, the main processor sets the `RESET COUNTER` to `1` to indicate to the quadrature decoder that the wheel value has been read. The quadrature decoder checks the

status of RESET COUNTER at the start of each scan cycle and resets the scroll wheel data if RESET COUNTER is set to 1.

When the application receives the first interrupt from the quadrature decoder, the main processor disables any further interrupts by clearing ENABLE INTERRUPT to 0 and on receiving the LS_RADIO_EVENT_IND indication, checks for new wheel data. When there is no data to be sent, the main processor re-enables interrupts from the quadrature decoder.

4. Code Overview

Sections 4.1 and 4.2 describe functions of the Mouse application.

4.1. Application Entry Points

4.1.1. AppInit()

This function is invoked when the application is powered on or when the IC resets. It performs the following steps:

1. Initialises the application timers, application data structures and hardware
2. Configures GATT entity for server role and installs support for optional GATT procedures like write long characteristic value
3. Configures the NVM manager to use I^2C EEPROM
4. Initialises all the services
5. Reads the persistent store
6. Registers the ATT database with the firmware

4.1.2. AppProcessLmEvent()

This function is invoked whenever a LM-specific event is received by the system. The following events are handled in this function:

4.1.2.1. Database Access

- `GATT_ADD_DB_CFM`: This confirmation event marks the completion of database registration with the firmware. On receiving this event, the Mouse application starts advertising.
- `GATT_ACCESS_IND`: This indication event is received when the remote HID host tries to access an ATT characteristic managed by the application.

4.1.2.2. LS Events

- `LS_CONNECTION_PARAM_UPDATE_CFM`: This confirmation event is received in response to the connection parameter update request by the application. The connection parameter update request from the application triggers the L2CAP connection parameter update signalling procedure. See Volume 3, Part A, Section 4.20 of *Bluetooth Core Specification Version 4.1*.
- `LS_CONNECTION_PARAM_UPDATE_IND`: This indication event is received when the remote central device updates the connection parameters. On receiving this event, the application validates the new connection parameters against the preferred connection parameters and triggers a connection parameter update request if the new connection parameters do not comply with the preferred connection parameters. The new connection parameters are also used to calculate the value of the sensor polling period which is used in `APP_ACTIVE` state to periodically send the mouse reports. The mouse sensor registers are updated based on the new connection parameter values.
- `LS_RADIO_EVENT_IND`: The Mouse application registers for this indication in ACTIVE state. The application uses the `radio_event_first_tx` option to receive the `LS_RADIO_EVENT_IND` event from firmware when the first data packet is transmitted in a connection event. This event is used to align the sensor polling timer with the connection event as described in section 4.2.5.

4.1.2.3. SMP Events

- SM_KEYS_IND: This indication event is received on completion of the bonding procedure. It contains keys and security information used on a connection that has completed short term key generation. The application stores the received diversifier (DIV) and Identity Resolving Key (IRK) (if the collector device supports resolvable random addresses) to the NVM. See Volume 3, Part H, section 2.1 of the *Bluetooth Core Specification Version 4.1*.
- SM_SIMPLE_PAIRING_COMPLETE_IND: This indication event indicates that the pairing has completed successfully or otherwise. See Volume 3, Part H, Section 2.3 of *Bluetooth Core Specification Version 4.1*. In the case of a successful completion of the pairing procedure, the mouse application is bonded with a HID host and the bonding information is stored in the NVM. The bonded device address will be added to the white list, if it is not a resolvable random address.
- SM_DIV_APPROVE_IND: This indication event is received when the remote connected device re-encrypts the link or triggers encryption at the time of reconnection. The firmware sends the diversifier in this event and waits for the application to approve or disapprove the encryption. The application shall disapprove the encryption if the bond has been removed by the user. The application compares this diversifier with the one received in the SM_KEYS_IND event at the time of the first encryption. If similar, the application approves the encryption, otherwise it disapproves it.
- SM_PAIRING_AUTH_IND: This indication is received when the remote connected device initiates pairing. The application can either accept or reject the pairing request from the peer device. The application shall reject the pairing request if it is already bonded to a HID host to prevent any new device disguising itself as one previously bonded to the Mouse application.

4.1.2.4. Connection Events

- GATT_CONNECT_CFM: This confirmation event indicates that the connection procedure has completed. If it has not successfully completed, the application restarts advertising. If the application is bonded to a device using a resolvable random address and the connection is established, the application tries to resolve the connected device address using the IRK stored in the NVM. If the application fails to resolve the address, it disconnects the link and restarts advertising.
- GATT_CANCEL_CONNECT_CFM: This confirmation event confirms the cancellation of the connection procedure. When the application stops advertisements to change advertising parameters or to save power, this signal confirms the successful stopping of advertisements by the mouse application.
- LM_EV_DISCONNECT_COMPLETE: This event is received on a link disconnection. Disconnection could be due to link loss, locally triggered or triggered by the remote connected device.
- LM_EV_CONNECTION_COMPLETE: This event is received when the connection with a HID host is considered to be complete and includes the new connection parameters. These parameters are also used to calculate the value of the sensor polling period which is used in APP_ACTIVE state to periodically send the mouse reports. The mouse sensor registers are updated based on the new connection parameter values.
- LM_EV_ENCRYPTION_CHANGE: This event indicates a change in the link encryption.
- LM_EV_CONNECTION_UPDATE: This event indicates that the connection parameters have been updated and is generated when the connection parameter update procedure is initiated by either the master or the slave.

4.1.3. AppProcessSystemEvent()

The `AppProcessSystemEvent()` handles the system events such as a PIO change. It currently handles the following system events:

- `sys_event_pio_changed`: This event indicates a change in the PIO status. On a **Pairing Button Press**, the status of the mapped PIO changes and the application receives a PIO changed event. On receiving this event, the application takes the appropriate action as described in Figure 2.5. This event is also received on a left, right and middle button press. The corresponding button presses are de-bounced after which the global variable storing the button status is updated to reflect the latest status.
- `sys_event_wakeup`: This event is received when the mouse sensor has new data in its movement registers. The WAKE pin of the chip is mapped to the active low motion pin of the sensor used in the mouse. When the sensor has new data in its movement registers, the sensor sets drives this pin low. This causes the WAKE pin of the chip to go low which triggers this event.
- `sys_event_battery_low`: This event is received whenever the battery voltage crosses the low battery voltage threshold. If connected and notifications are configured, the mouse application notifies the battery level to the connected HID host. The low-power LED starts flashing to indicate that the battery needs to be replaced.
- `sys_event_pio_ctrlr`: This event is received from the PIO controller (which includes quadrature decoder functionality) when the scroll wheel data has been updated and the events have been configured by the application. See section 3.4.1 for more information.

4.2. Internal State Machine

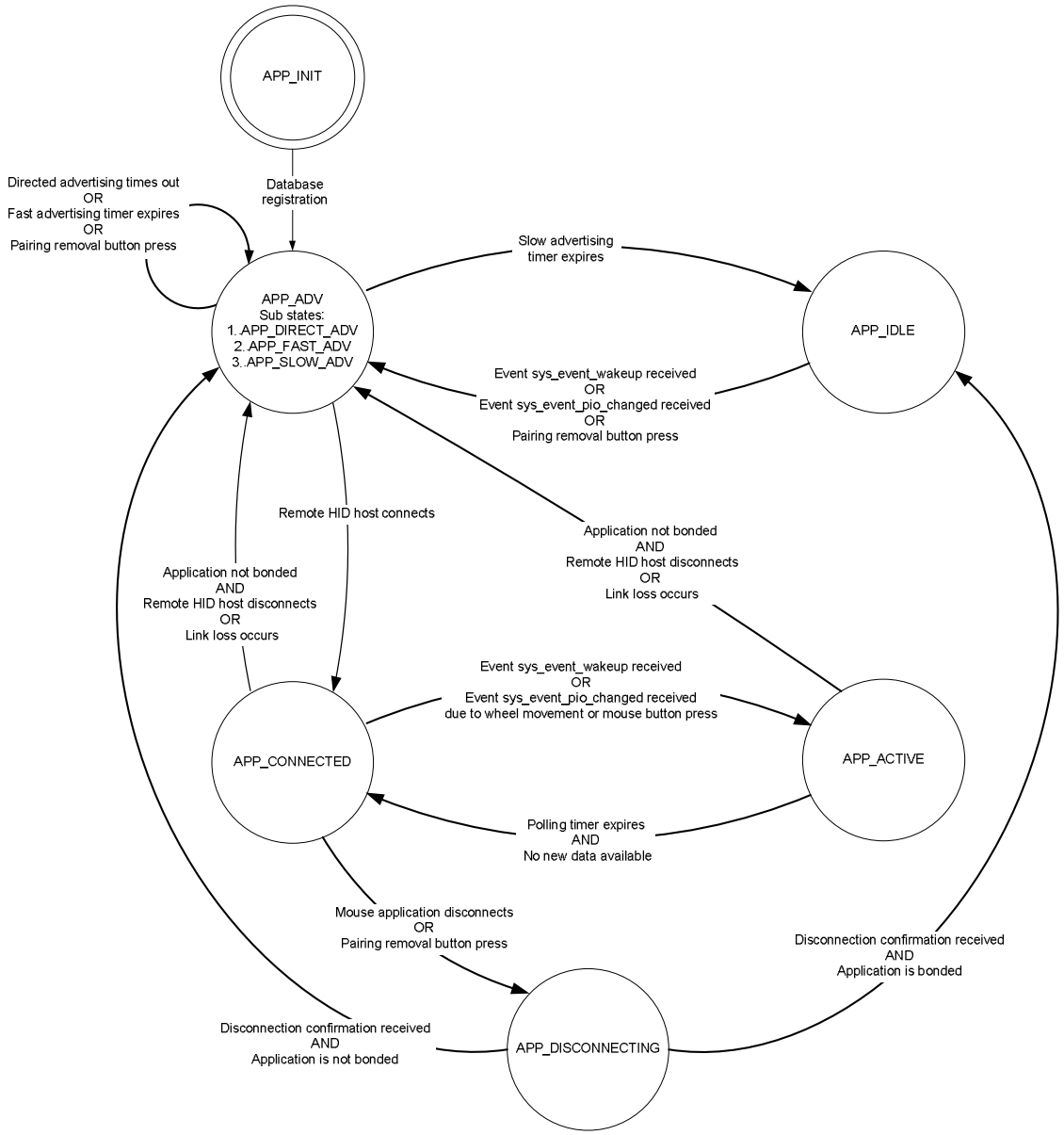

Figure 4.1: Internal State Machine Diagram

The Mouse application uses the internal states as described in sections 4.2.1 to 4.2.6.

4.2.1. APP_INIT

When the Mouse application is powered on or the chip resets, the mouse application initialises in this state and registers the database with the firmware. When the application receives a successful confirmation for the database registration it enters either the APP_FAST_ADVERTISING state or the APP_DIRECT_ADVERT state.

4.2.2. APP_IDLE

In this state, the Mouse application is not connected to any HID host.

When the user moves the mouse or presses one of the left, middle and right buttons, the application triggers advertisements and enters the APP_FAST_ADVERTISING or APP_DIRECT_ADVERT state.

- On a **Pairing button press**, the application removes the bonding information, clears the white list and starts advertising. See the *Bluetooth Core Specification Version 4.1* for more information on white lists.

4.2.3. APP_ADVERTISING

The Mouse application triggers advertisements in the states.

- Sub state APP_DIRECT_ADVERT: The application starts in this sub state if:
  - Privacy is enabled and a valid re-connection address has been written to the Reconnection address characteristic of the GAP service (see Appendix A) by the bonded HID host. The application triggers directed advertisements using the reconnection address as written by the bonded HID host.
  - Privacy is disabled and the mouse is bonded to a HID host that uses public or static address.
  - If the earlier bonded host re-connects, the application enters the APP_CONNECTED state. If the bonded host fails to connect before the directed advertising times out, the application enters the APP_FAST_ADVERTISING state.
- Sub state APP_FAST_ADVERTISING: This sub-state uses fast advertising parameters. The application starts in this sub state if:
  - Privacy is disabled and
    - the mouse is not bonded to any HID host

      or
    - the mouse is bonded to a HID host which is using a random address
  - Privacy is enabled and no reconnection address has been written by the remote HID host.
  - If a remote HID host connects to the application, the mouse stops advertisements and enters the APP_CONNECTED state. If the fast advertising timer expires before connection is made, the application enters the APP_SLOW_ADVERTISING sub state. See section 7.2 for more information on advertisement timers.
- Sub state APP_SLOW_ADVERTISING: The application uses slow advertising parameters in this sub state. If a remote device connects to it, the application stops advertisements and enters the APP_CONNECTED state. If the slow advertising timer expires before a connection is made, the application enters the APP_IDLE state. If the user moves the mouse or presses any button when the application is in the APP_SLOW_ADVERTISING state, it enters the APP_DIRECT_ADVERT or the APP_FAST_ADVERTISING state.
  - For detailed state transitions between the sub-states APP_DIRECT_ADVERT, APP_FAST_ADVERTISING and APP_SLOW_ADVERTISING, see Figure 2.4.

4.2.4. APP_CONNECTED

In this state, the mouse is connected to a remote HID host.

- When the user moves the mouse or presses any buttons, the mouse enters the `APP_ACTIVE` state and sends the mouse data to the connected host.
- On a **Pairing button press**, the application disconnects the link, removes any bonding information, clears the white list and starts advertising. See *Bluetooth Core Specification Version 4.1* for more information on white lists.
- The application disconnects the link if kept idle for some time and enters the `APP_DISCONNECTING` state. See section 7.4 for more information on the idle timer.
- In the case of link loss, the application enters the `APP_FAST_ADVERTISING` or `APP_DIRECT_ADVERT` state.
- In the case of remote triggered disconnection and if the application is not bonded to any HID host, the application enters the `APP_FAST_ADVERTISING` state; otherwise it enters the `APP_IDLE` state.

4.2.5. APP_ACTIVE

The Mouse application enters this state when it new sensor data is to be transmitted to the connected HID host. One of the following user actions can cause the application to move to the `APP_ACTIVE` state:

- Mouse movement: On receiving the `sys_event_wakeup` event on mouse movement (see section 4.1.3), the application disables any further events generated by mouse movement and enters the `APP_ACTIVE` state. In this state, the application reads the mouse sensor registers and sends delta-x and delta-y values indicating mouse movement to the connected HID host.
- Mouse button press: On receiving the `sys_event_pio_changed` event on left, right or middle button press, the application filters out noise and bouncing on the PIOs corresponding to the mouse buttons. Once de-bounced, the status of the buttons is updated which is maintained as a global variable. This updated button status is sent to the connected HID host by entering the `APP_ACTIVE` state.
- Mouse wheel scroll: On receiving the `sys_event_pio_ctrlr` event on the scrolling of the mouse wheel, the application disables any further events generated by the quadrature decoder and enters the `APP_ACTIVE` state. The application reads the scroll wheel data from the memory location shared between the main processor (where the application runs) and the PIO controller and sends it to the connected HID host.

Once the first packet of data has been sent to the peer device, a polling timer is started by the mouse application. This timer ensures that when the mouse is being continuously moved by the user, the mouse reports shall be sent at the rate of 1 report per connection interval. The sensor associated with the mouse is also calibrated so that it updates its registers with the data resulting from the movement, at a faster rate than the polling rate. In the ACTIVE state, the data from the sensor is read continuously on the expiry of the polling timer and sent to the peer device. Each time the data is sent to the peer device, the polling timer is restarted.

Since the polling period is less than the connection interval, over a prolonged period, the timer expiry point drifts away from the starting point of the next connection interval and may cause two timer expiry points to fall into one connection interval. To avoid this, the timer starting point is aligned with the start of the connection interval using the `radio_event_first_tx`. These events are enabled while entering the ACTIVE state. Upon receiving this event, the timer is deleted and re-created, thus aligning the timer starting point with the start of the connection interval. This timer expires close to the start of the next connection interval and, upon expiry, the data from the sensor is read and sent to the firmware before being transmitted to the connected HID host. The point at which the data is sent to the firmware should be at least 1.8 ms before the timer starting point of the next connection interval. If this is not achieved, then the data sent to the firmware is delayed by a connection interval before being sent to the connected HID host.

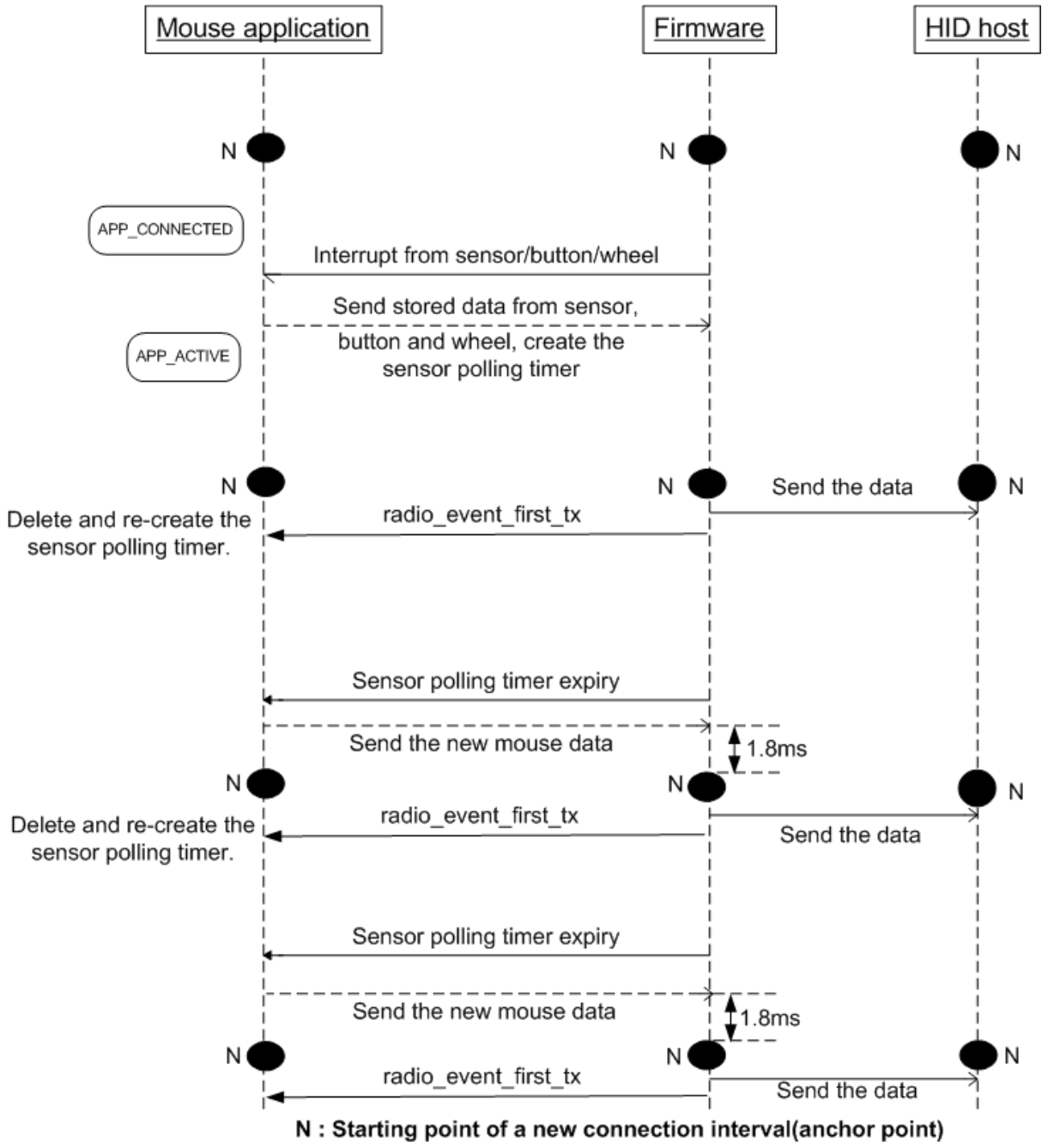

Figure 4.2: Periodic Transmission of Mouse Data using a Timer

Figure 4.2 illustrates this timer-based approach used to send the mouse data periodically.

On the expiry of the polling timer, if there is no new data either in the mouse sensor movement registers or from the button and wheel sensor, the application returns to the `APP_CONNECTED` state, re-enabling all the interrupts.

Since the data from the sensor is periodically read and passed to the firmware to be transmitted to the peer device, in a poor RF environment it is possible that the firmware is unable to transmit the data to the peer device. In this case, the firmware responds with `gatt_status_busy` in the result field of the `GATT_CHAR_VAL_NOT_CFM` event, the polling timer is deleted and the application waits for the `radio_event_first_tx` event. Upon receiving

`radio_event_first_tx`, the application attempts to re-send the data to the connected HID host, as illustrated in Figure 4.3.

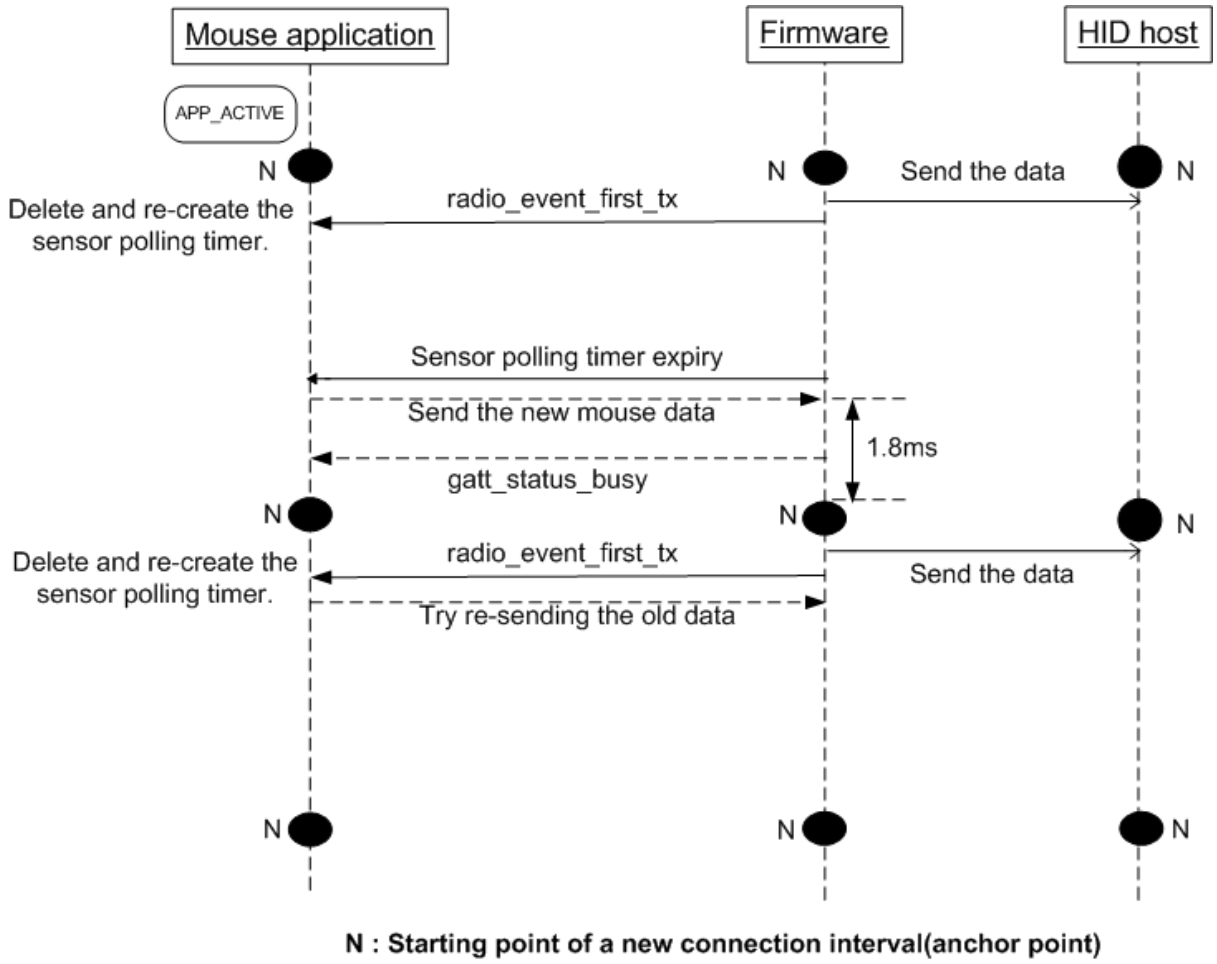

Figure 4.3: Re-transmission of Data when the Firmware is Busy

In the `APP_ACTIVE` state, the data is transmitted by the mouse in the form of HID reports. These reports are formed as per the report descriptor described in Appendix C when the protocol mode is set to 'report' protocol.

When the protocol mode is set to 'boot' protocol, the report is formed as per the standard mouse report descriptor in boot mode as described in the *USB HID Specification*. In this mode, the report structure is as shown in Figure 4.4

Byte 1	Byte 2	Byte 3
Button Status	Delta-X	Delta-Y

Figure 4.4: Report Structure of Mouse in Boot Mode

The delta-**X** and delta-**Y** values are read from the sensor as 12-bit wide values. These are converted to 8-bit wide values when the reports are formed in boot mode. The resultant delta-X and delta-Y values are each checked for overflow while forming the reports in boot mode. The triple byte reports are transmitted to the connected HID host.

4.2.6. APP_DISCONNECTING

The Mouse application enters this state after initiating the disconnect procedure.

- When it receives the event from the firmware that indicates that the disconnection is complete, it checks if it is bonded to any HID host.
  - If the application is bonded, it enters the `APP_IDLE` state and waits for user activity.
  - If the application is not bonded, it enters the `APP_FAST_ADVERTISING` state.
- On a **Pairing button press**, the application removes the bonding information and clears the white list.

5. Interrupt Handling

The Mouse application receives interrupts from various user actions and sources, see Table 5.1.

User action	Interrupt source	Interrupt event
Mouse movement	WAKE pin	`sys_event_wakeup`
Mouse button press/release	Button PIOs	`sys_event_pio_changed`
Wheel scroll	PIO controller (quadrature decoder)	`sys_event_pio_ctrlr`
Pairing removal button press	Pairing removal PIO	`sys_event_pio_changed`

Table 5.1: Interrupt Mapping Table

The state changes when the pairing removal button is pressed by the user for more than a second and is described in Figure 2.5.

The handling of other interrupts and the relevant state changes are described in Figure 5.1.

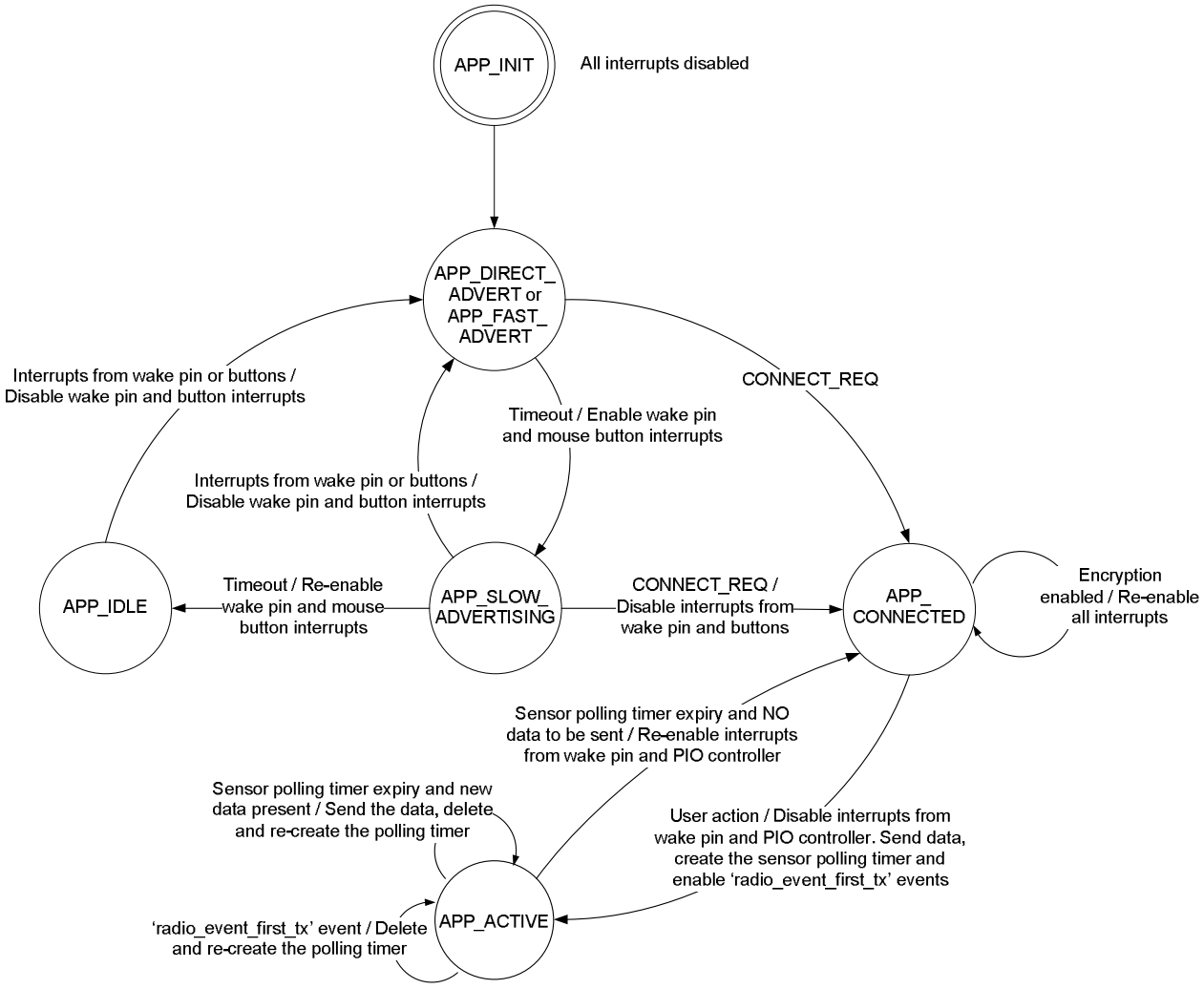

Figure 5.1: Interrupt Handling and State Changes

6. NVM Memory Map

The applications can store data in NVM to prevent data loss in the event of a power off or a chip reset. Table 6.1 shows the memory map the Mouse application uses for NVM.

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
Sanity Word	uint16	1	0
Bonded Flag	Boolean	1	1
Bonded Device Address	Structure	5	2
Diversifier	uint16	1	7
IRK	uint16 array	8	8

Table 6.1: NVM Memory Map for Application

Table 6.2 shows NVM offsets for the GAP service.

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
GAP Device Name Length	uint16	1	16
GAP Device Name	uint8 array	20	17
GAP Peripheral Privacy Flag[1]	Boolean	1	37[2]
GAP Reconnection address[1]	Structure	4	38[2]
Note: [1] The NVM value of these characteristics will only be added if privacy is enabled. [2] The NVM Offset values used when privacy is enabled.			

Table 6.2: NVM Memory Map for GAP Service

Table 6.3 shows NVM offsets for the HID Service.

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
Boot mouse input report Characteristic Client Configuration Descriptor	uint16	1	37/42[1]
Mouse input report Characteristic Client Configuration Descriptor	uint16	1	38/43[1]
Note: [1] The NVM Offset values used when privacy is enabled.			

Table 6.3: NVM Memory Map for HID Service

Table 6.4 shows NVM offsets for the Battery service.

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
Battery Level characteristic Client Configuration Descriptor	uint16	1	39/44[1]
Note: [1] The NVM Offset values used when privacy is enabled.			

Table 6.4: NVM Memory Map for Battery Service

Table 6.5 shows NVM offsets for the Scan Parameters service.

Entity Name	Type	Size of Entity (Words)	NVM Offset Words)
Scan refresh characteristic Client Configuration Descriptor	uint16	1	40/45[1]
Note: [1] The NVM Offset values used when privacy is enabled.			

Table 6.5: NVM Memory Map for Scan Parameters Service

The Mouse application does not pack the data before writing it to the NVM. This means that writing a `uint8` value takes one word of NVM memory.

7. Customising the Mouse Application

Developers can easily customise the application by modifying the parameter values described in sections 7.1 to 7.8.

7.1. Advertising Parameters

The mouse application uses the parameters in Table 7.1 for fast and slow advertisements. The macros for these values are defined in the `gap_conn_params.h` file. These values have been chosen by considering the overall current consumption of the device. See *Bluetooth Core Specification Version 4.1* for advertising parameter range.

Parameter Name	Slow Advertisements	Fast Advertisements
Minimum Advertising Interval	1280 ms	60 ms
Maximum Advertising Interval	1280 ms	60 ms

Table 7.1: Advertising Parameters

7.2. Advertisement Timers

The Mouse application enters the appropriate state on expiry of the advertisement timers. See section 4.2 for more information. The macros for these timer values are defined in the `gap_conn_params.h` file.

Timer Name	Timer Values
Fast Advertisement Timer Value	30 s
Slow Advertisement Timer Value	30 s

Table 7.2: Advertisement Timers

7.3. Connection Parameters

The Mouse application uses the connection parameters listed in Table 7.3 by default, and should be used to configure the Profile Demonstrator before a connection is attempted, see section 2.2. The macros for these values are defined in the `gap_conn_params.h` file. These values have been chosen by considering the overall current consumption of the device. See *Bluetooth Core Specification Version 4.1* for the connection parameter range.

Parameter Name	Parameter Value	Profile Demonstrator Configuration
Minimum Connection Interval	12.5 ms	10 ms
Maximum Connection Interval	12.5 ms	10 ms
Slave Latency	100 intervals	100
Supervision Timeout	8000 ms	800

Table 7.3: Connection Parameters (Default)

When connected to Apple products, the connection parameters listed in Table 7.4 should be used.

Parameter Name	Parameter Value	Profile Demonstrator Configuration
Minimum Connection Interval	20 ms	16
Maximum Connection Interval	40 ms	32

Parameter Name	Parameter Value	Profile Demonstrator Configuration
Slave Latency	4	4
Supervision Timeout	6000 ms	600

Table 7.4: Preferred Connection Parameters for Apple Products

7.4. Idle Connected Timeout

The Mouse application disconnects the link if there is no user action for some time. The macro for this idle connected timeout is defined in the `gap_conn_params.h` file.

Parameter Name	Parameter Value
Idle Connected Timeout	30 minutes

Table 7.5: Idle Connected Timeout

7.5. Connection Parameter Update

The Mouse application can request the remote HID host to update the connection parameters according to its power requirements. The application requests a connection parameter update as per the recommendations in *Bluetooth Core Specification Version 4.1* [Vol. 3], Part C Section 9.3.9, or 30 seconds after the remote Server changes the connection parameters. Upon connection establishment with the Server, the following procedure is used to send a connection parameter update request:

1. Upon connection, the 5s $T_{GAP(conn\_pause\_peripheral)}$ timer is started.
2. Upon the expiry of $T_{GAP(conn\_pause\_peripheral)}$, the 1s $T_{GAP(conn\_pause\_central)}$ timer is started.
3. During this 1s $T_{GAP(conn\_pause\_central)}$ period, if the application receives a `GATT_ACCESS_IND LM` event, the timer will be deleted and re-created. The receipt of this event means that the service discovery procedure is in progress and the Mouse application should not request a connection parameter update.
4. Upon the expiry of $T_{GAP(conn\_pause\_peripheral)}$, a connection parameter update request will be sent from the Mouse application.

The remote HID host may or may not accept the requested parameters. If the remote server rejects the new requested parameters, the application again requests an update after 30 seconds. The macro for this time value is defined in file app_gatt`.h` and can be modified as required but should be greater than 30 seconds as per the recommendation of the core specification. See *Bluetooth Core Specification Version 4.1* [Vol. 3], Part C Section 9.3.9. After two failed attempts, the CSC Sensor application tries to update with the Apple preferred connection parameters another two times. Ensure the **Accept connected parameters** option on the CSR µEnergy Profile Demonstrator application is checked to accept the requested parameters, see Figure 7.1. See Figure 7.2 for the detailed procedure.

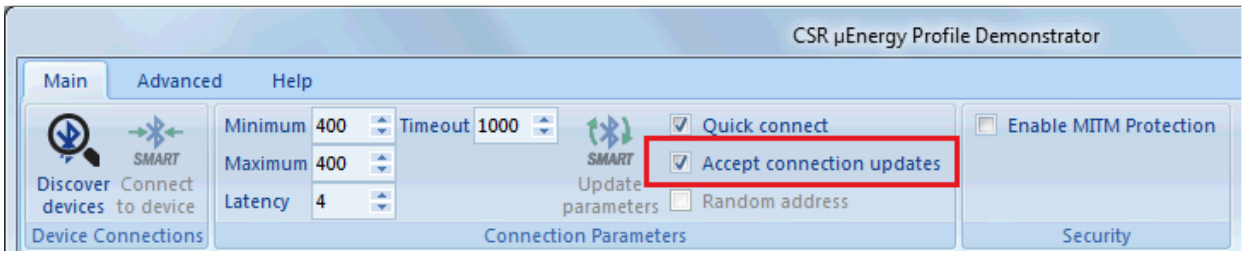

Figure 7.1: Accept Connection Parameters

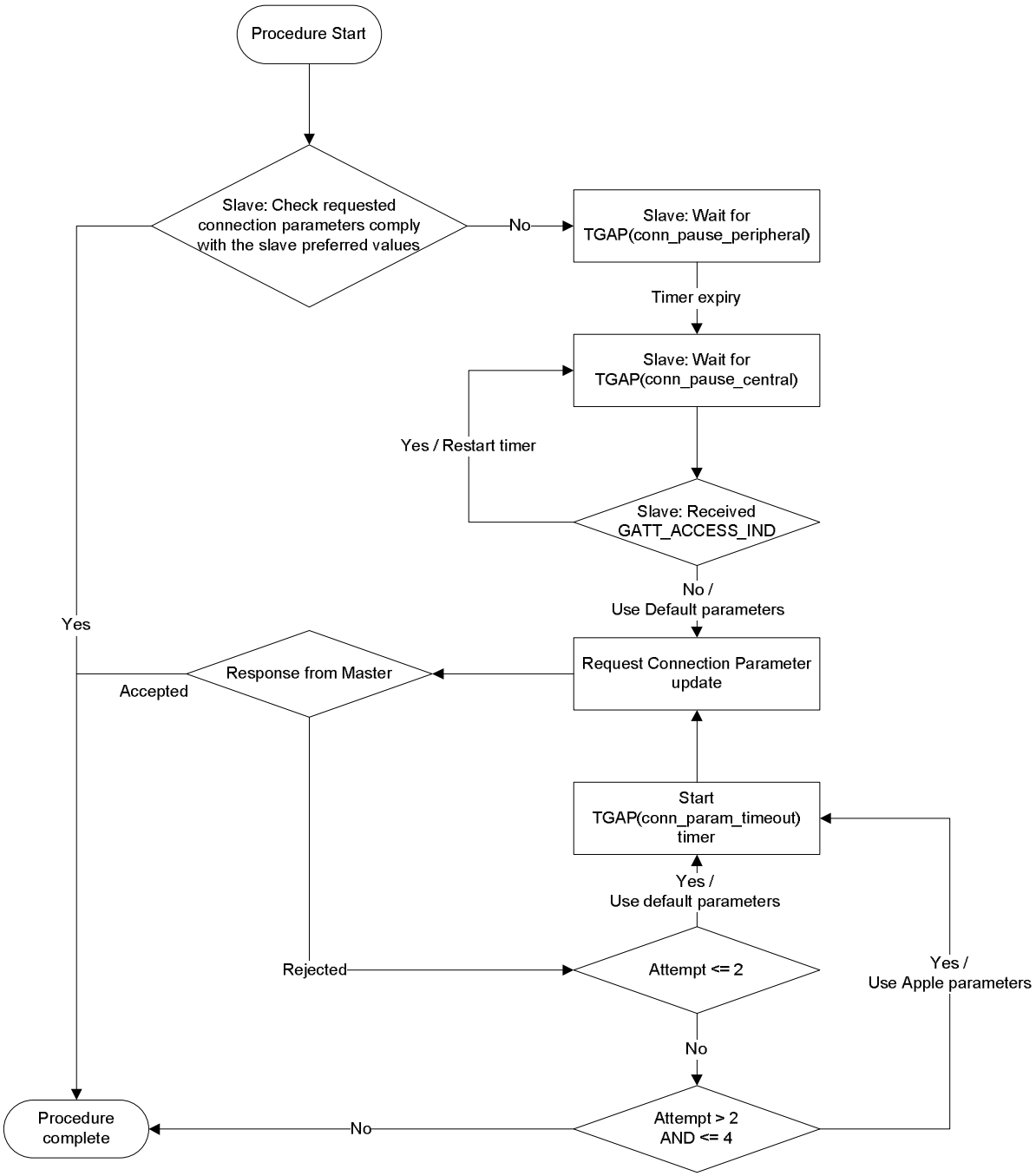

Figure 7.2: Connection Parameter Update Procedure

7.6. Device Name

The user can change the device name for the application. By default, it is set to `CSR Mouse` in the `gap_service.c` file. The maximum length of the device name is 20 octets.

7.7. Pairing Button Press

The Mouse application configures PIO8 as the button that removes pairing information.

Since PIO8 is not exposed on the CSR1001 development board, this functionality cannot be tested directly. To test the **Pairing button press** functionality on the CSR1001 development board, the pairing button can be configured to any of the exposed PIOs, for example PIO15, by changing the value of macro `BUTTON_PAIRING_PIO` to 15 in the `mouse_hw.c` file. The modified Mouse application can be built and downloaded to the CSR1001 development board.

The pairing removal functionality can be tested by shorting the PIO configured to be the pairing button with GND for more than 1 second.

7.8. Privacy

The macro `__GAP_PRIVACY_SUPPORT__` is used to enable or disable privacy support. By default, the support of privacy is disabled. To enable support of privacy, uncomment the definition of the macro `__GAP_PRIVACY_SUPPORT__` in the `user_config.h` file.

Note:

The support for Privacy Mode and Reconnection Address characteristics has been deprecated from *Bluetooth Core Specification version 4.1*.

If privacy support is enabled:

- The Mouse application uses resolvable random address for undirected advertising
- GAP service characteristics 'Peripheral privacy flag' and 'Re-connection address' are supported
- Directed advertisements use the re-connection address if it has been written by the remote device.

8. Current Consumption

The current consumed by the application can be measured by removing the Current Measuring Jumper, see Figure 2.1, which is labelled **IMEAS** and installing an ammeter in its place. The ammeter should be set to DC, measuring current from µA to mA.

The setup used while measuring current consumption is described in section 2.1.1. In addition, the CSR µEnergy Profile Demonstrator application must be configured to accept connection parameter update requests.

Table 8.1 shows the typical current consumption values measured when testing under noisy RF conditions with Channel Map Updates disabled, and with typical connection parameter values using the CSR1001 development board. See the Release Notes for the actual current consumption values measured for the application.

Test Scenario	Description	Average Current Consumption	Remarks
Fast Advertisements	1. Switch on the HID device 2. Wait for 5 s 3. Take the measurement	440 µA[1]	▪ Advertisement Interval: 60 ms ▪ Advertisement data length: 22 octets ▪ Measurement Time Duration: 20 s
Slow Advertisements	1. Switch on the HID device 2. Wait for 40 s 3. Take the measurement	27 µA[1]	▪ Advertising Interval: 1.28 s ▪ Advertisement Data length: 22 octets ▪ Measurement Time Duration: 20 s
Connected Idle	1. Connect to the HID host 2. Wait for 60 s 3. Take the measurement	17 µA	▪ Connection Parameters ▪ Minimum connection interval: 12.5 ms ▪ Maximum connection interval: 12.5 ms ▪ Slave latency: 64 ▪ Measurement Time Duration: 60 s
Disconnected Idle	1. Switch on the device 2. Take the current measurement after 2 minutes	5 µA	▪ Connection Parameters ▪ Minimum connection interval: 12.5 ms ▪ Maximum connection interval: 12.5 ms ▪ Slave latency: 64 ▪ Measurement Time Duration: 60 s

Note:

[1] The current consumption in the advertising states may vary depending on the number of the active scanners in the area (since the Mouse application sets up the scan response).

- Average current consumption is measured at 3 V
- Ammeter used: Agilent 34411A
- Channel Map Update has been disabled on the USB dongle by setting the AFH options PS Key to 0x0037 (Default Value: 0x0017) using the PSTool application (included in CSR BlueSuite tools only and available on www.CSRSupport.com)
- It is recommended to pull PIO4 to GND when measuring current on the CSR10x1 development board.

Table 8.1: Current Consumption Values

Appendix A Definitions

Term	Meaning
Input report	Low latency transfer of information from HID device to HID host
Pairing button press	The pairing button is pressed and held down for a period of 1 second

Table A.1: Definitions

Appendix B GATT Database

B.1 Battery Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Battery Level	0x0023	Read, Notify	Application	Security mode 1 and Security level 2	Current battery level
Battery Level-Client Configuration Descriptor	0x0024	Read, Write	Application	Security mode 1 and Security level 2	Current client configuration for “Battery level” characteristic

Table B.1: Battery Service Database

For more information on Battery Service, see *Battery Service Specification Version 1.0*.

B.2 Device Information Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Serial Number String	0x002d	Read	Firmware	Security Mode 1 and Security Level 2	"BLE-MOUSE-001"
Model Number String	0x002f	Read	Firmware	Security Mode 1 and Security Level 2	"BLE-MOUSE-MODEL-001"
Hardware Revision String	0x0031	Read	Firmware	Security Mode 1 and Security Level 2	*<Chip Identifier>*
Firmware Revision String	0x0033	Read	Firmware	Security Mode 1 and Security Level 2	*<SDK version>*
Software Revision String	0x0035	Read	Firmware	Security Mode 1 and Security Level 2	*<Application version>*

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Manufacturer Name String	0x0037	Read	Firmware	Security Mode 1 and Security Level 2	"Cambridge Silicon Radio"
PnP ID	0x0039	Read	Firmware	Security Mode 1 and Security Level 2	Vendor Id source is BT Vendor Id is 0x000a Product Id is 0x014d Product Version is 1.0.0

Table B.2: Device Information Service Database

For more information on Device Information Service, see *Device Information Service Specification Version 1.1*.

For more information on Security permissions, see *Bluetooth Core Specification Version 4.1*.

B.3 GAP Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Device Name	0x0003	Read, Write	Application	Security Mode 1 and Security Level 2	Device name. Default name: "CSR Mouse"
Appearance	0x0005	Read	Firmware	Security Mode 1 and Security Level 1	Mouse - 0x03c2
Peripheral preferred connection parameters	0x0007	Read	Firmware	Security Mode 1 and Security Level 1	Min connection interval - 12.5 ms Max connection interval – 12.5 ms Slave latency - 64 Connection timeout - 12.5 s
Peripheral privacy flag[1]	0x0009	Read, Write	Application	Security Mode 1 and Security Level 2	0x0001 (TRUE)
Reconnection address[1]	0x000b	Write	Application	Security Mode 1 and Security Level 2	0x4000 00 000000
Note: [1] These characteristics will only be added if privacy is enabled.					

Table B.3: GAP Service Database

For more information on GAP service and security permissions, see *Bluetooth Core Specification Version 4.1*.

B.4 HID Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
HID Information	0x000f	Read	Firmware	Security Mode 1 and Security Level 2	Base USB HID Specification Version - 0x0213 Country Code - 0x40 Flags - 0x01 (Remote Wakeup Supported)
Report Map	0x0011	Read	Firmware	Security Mode 1 and Security Level 2	See Appendix C for value
Boot Mouse Input Report	0x0013	Read, Notify	Application	Security Mode 1 and Security Level 2	Current mouse report value in Boot mode
Boot Mouse Input Report - Client Characteristic Configuration descriptor	0x0014	Read, Write	Application	Security Mode 1 and Security Level 2	Current client configuration for the “Boot Mouse Input Report” characteristic
Mouse Input Report	0x0016	Read, Notify	Application	Security Mode 1 and Security Level 2	Current mouse report in Report mode
Mouse Input Report - Client Characteristic Configuration descriptor	0x0017	Read, Write	Application	Security Mode 1 and Security Level 2	Current client configuration for the “Mouse Input Report” characteristic
Mouse Input Report - Report Reference Characteristic descriptor	0x0018	Read	Firmware	Security Mode 1 and Security Level 2	Report Id - 0 Report Type - 1 (Input Report)
Mouse Feature Report	0x001a	Read, write	Application	Security Mode 1 and Security Level 2	03 (Mouse resolution of 1600 CPI)
Mouse Feature Report - Report Reference Characteristic descriptor	0x001b	Read	Application	Security Mode 1 and Security Level 2	Report Id - 0 Report Type - 3 (Feature Report)
Mouse Feature Report - User description	0x001c	Read	Firmware	Security Mode 1 and Security Level 2	Mouse resolution in CPI 0 = 400 1 = 800 2 = 1200 3 = 1600

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
HID Control Point	0x001e	Write without response	Application	Security Mode 1 and Security Level 2	*<Control point value>*
Protocol Mode	0x0020	Read, Write without response	Application	Security Mode 1 and Security Level 2	Current protocol mode value (0x00 - Boot Protocol Mode, 0x01 - Report Protocol Mode)

Table B.4: HID Service Database

See *Bluetooth Core Specification Version 4.1* for more information on security permissions. For more information on HID Service, see *HID Service Specification Version 1.0*.

B.5 Scan Parameter Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Scan Interval Window	0x0027	Write without response	Application	Security mode 1 and Security level 2	Scan interval window value written by the HID host
Scan Refresh	0x0029	Notify	Application	Security mode 1 and Security level 2	Server requires refresh - 0x00
Scan Refresh - Client Configuration Descriptor	0x002a	Read, Write	Application	Security mode 1 and Security level 2	Current client configuration for “Scan Refresh” characteristic

Table B.5: Scan Parameter Service Database

For more information on Scan Parameter Service, see *Scan Parameter Service Specification Version 1.0*.

Note:

Characteristics are managed by either the firmware or the application. The characteristics managed by the application have flags set to FLAG_IRQ in the corresponding database file. When the remote connected device accesses that characteristic, the application receives an GATT_ACCESS_IND LM event which is handled in the AppProcessLmEvent() function defined in the mouse.c file. See section 4.1.2.1 for more information on handling of the GATT_ACCESS_IND LM event. For more information on flags, see the *GATT Database Generator User Guide*.

Appendix C Report Map Value

The Mouse application makes use of the HID report descriptors (described in Figure C.1) as report map characteristic values.

```
0x05, 0x01          /* USAGE_PAGE(Generic Desktop) */
0x09, 0x02,         /* USAGE (Mouse) */
0xA1, 0x01,         /* COLLECTION (Application) */
0x09, 0x01,         /* USAGE (Pointer) */
0xA1, 0x00,         /* COLLECTION (Physical) */
0x05, 0x09,         /* USAGE_PAGE (Button) */
0x19, 0x01,         /* USAGE_MINIMUM (Button 1) */
0x29, 0x03,         /* USAGE_MAXIMUM (Button 3) */
0x15, 0x00,         /* LOGICAL_MINIMUM (0) */
0x25, 0x01,         /* LOGICAL_MAXIMUM (1) */
0x95, 0x03,         /* REPORT_COUNT (3) */
0x75, 0x01,         /* REPORT_SIZE (1) */
0x81, 0x02,         /* INPUT (Data,Var,Abs) */
0x95, 0x01,         /* REPORT_COUNT (1) */
0x75, 0x05,         /* REPORT_SIZE (5) */
0x81, 0x03,         /* INPUT (Cnst,Var,Abs) */
0x05, 0x01,         /* USAGE_PAGE (Generic Desktop) */
0x09, 0x30,         /* USAGE (X) */
0x09, 0x31,         /* USAGE (Y) */
0x16, 0x00, 0xF8,   /* LOGICAL_MINIMUM (-2048) */
0x26, 0xFF, 0x07,   /* LOGICAL_MAXIMUM (2047) */
0x75, 0x10,         /* REPORT_SIZE (16) */
0x95, 0x02,         /* REPORT_COUNT (2) */
0x81, 0x06,         /* INPUT (Data,Var,Rel) */
0x09, 0x38,         /*  USAGE (Wheel) */
0x15, 0x80,         /* LOGICAL_MINIMUM (-128) */
0x25, 0x7F,         /* LOGICAL_MAXIMUM (127) */
0x75, 0x08,         /* REPORT_SIZE (8) */
0x95, 0x01,         /* REPORT_COUNT (1) */
0x81, 0x06,         /* INPUT (Data,Var,Rel) */
0x09, 0x01,         /* USAGE (Pointer) */
0x15, 0x00,         /* LOGICAL_MINIMUM (0) */
0x25, 0x03,         /* LOGICAL_MAXIMUM (03) */
0x95, 0x01,         /* REPORT_COUNT (1) */
0x75, 0x02,         /* REPORT_SIZE (2) */
0xb1, 0x02,         /* FEATURE (Data, Var, Abs) */
0x95, 0x01,         /* REPORT_COUNT (1) */
0x75, 0x06,         /* REPORT_SIZE (6) */
0xb1, 0x03,         /* FEATURE (Const, array, const) - Padding (0) */
0xC0, 0xC0          /*  END_COLLECTION */
```

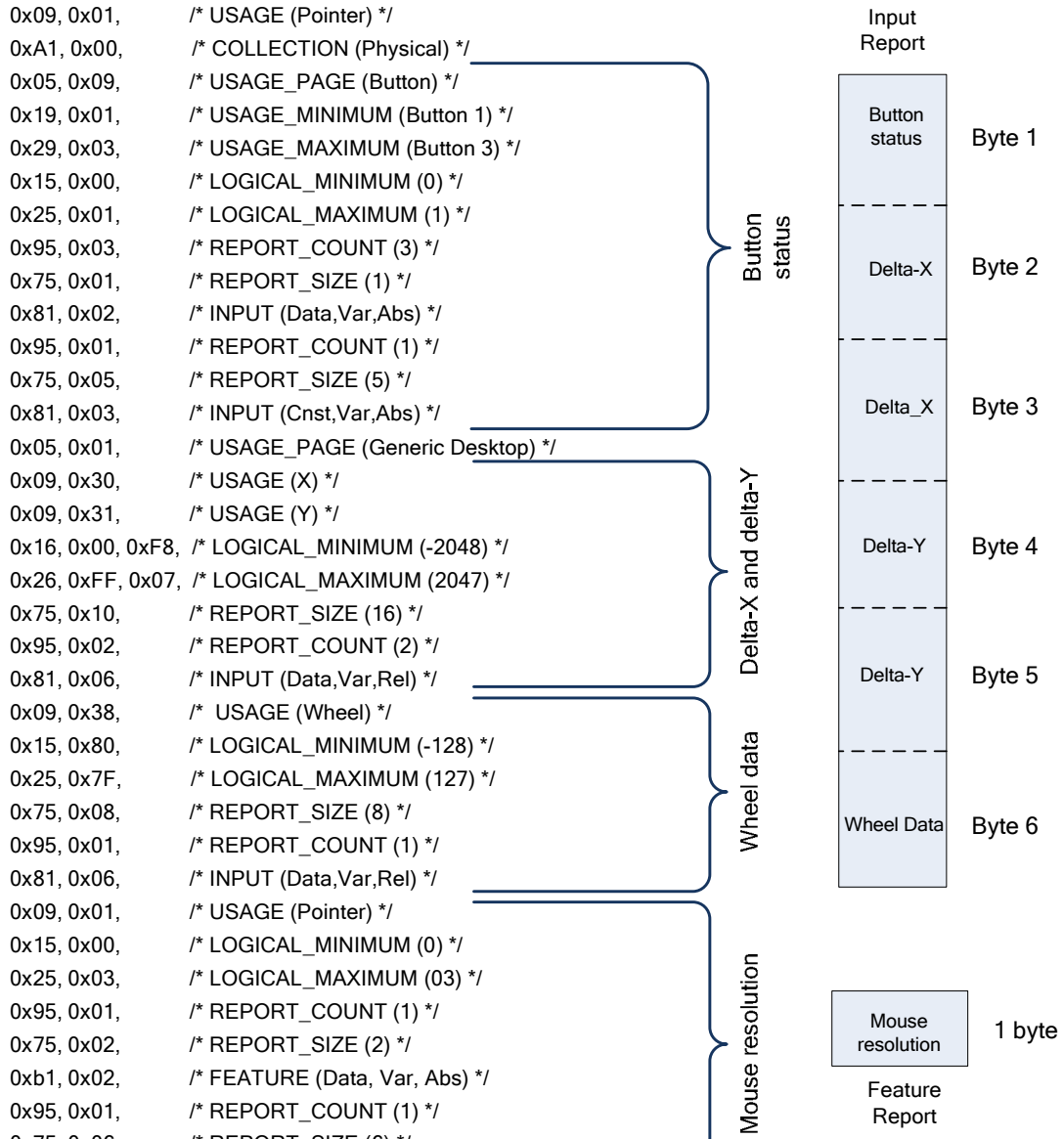

Figure C.1: Report Descriptor Used by the Mouse Application

Appendix D Advertising and Scan Response Data

Table D.1 details the fields added to the Advertising data by the Mouse application.

Advertising Data Field	Contents
Flags	The Mouse application sets the General Discoverable Mode bit when not bonded to any device. It sets the Limited Discoverable Mode bit when bonded to a device. See Section 11, Part C of Volume 3 in *Bluetooth Core Specification Version 4.*1 for more information.
Service UUIDs	The mouse application adds a 16-bit UUID for the following service: ▪ HID
Device Appearance	Mouse: `0x03c2`
Tx Power	Current Tx Power level
Device Name[1]	Device name (Default value : “CSR Mouse”)
Note: [1] The application adds the Device Name to the Scan Response data if the Device Name length is greater than the space left in the Advertising data field.	

Table D.1: Advertising and Scan Response Data Field

Appendix E Known Issues or Limitations

See the Mouse application and SDK Release Notes.

Document References

Document	Reference
Bluetooth Core Specification Version 4.1	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Core Specification Addendum 3	https://www.bluetooth.org/Technical/Specifications/adopted.htm
HID over GATT Profile Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
HID Service Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Battery Service Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Device Information Service Specification Version 1.1	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Scan Parameter Service Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Service Characteristics And Descriptions	http://developer.bluetooth.org/gatt/characteristics/Pages/default.aspx
GATT Database Generator	CS-219225-UG
CSR µEnergy xIDE User Guide	CS-212742-UG
Installing the CSR Driver for the Profile Demonstrator Application	CS-235358-UG
USB HID specification	www.usb.org/developers/devclass_docs/HID1_11.pdf/

Terms and Definitions

ATT	Attribute
Batt	Battery
BLE	Bluetooth Low Energy (now known as Bluetooth Smart)
Bluetooth®	Set of wireless technologies providing audio and data transfer over short-range radio connections
Bluetooth Smart	Formerly known as Bluetooth Low Energy
BT	Bluetooth
CFM	Confirmation
CSR®	Cambridge Silicon Radio
DIV	Diversifier
EEPROM	Electrically Erasable Programmable Read Only Memory
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GND	Ground
HID	Human Interface Device
i.e.	*Id est*, that is
I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
IND	Indication
IRK	Identity Resolving Key
L2CAP	Logical Link Control and Adaptation Protocol
LM	Link Manager
NVM	Non Volatile Memory
PC	Personal Computer
PIO	Programmable Input Output
PnP	Plug and Play
SDK	Software Development Kit
SMP	Security Manager Protocol
SPI	Serial Peripheral Interface
Tx	Transmit
USB	Universal Serial Bus

UUID	Universally Unique Identifier
xIDE	CSR's Integrated Development Environment
CSR10x1	CSR1001 or CSR1011 chip