BLESDK-2.2.x-CC2650RC Developers Guide

The CC2650RC remote control contains a multi-standard 2.4GHz CC2650 wireless MCU. Out of the box, the CC2650RC is loaded with a dual-image Bluetooth Low Engery demo as described in the CC2650RC User's Guide. The following sections are intended to get you started with BLE using BLE-STACK V2.2x hid_adv_remote example together with the TI Audio Profile for Voice over Bluetooth low energy on the CC2650RC.

Included with the CC2650RC development kit is an RF enabled remote control using a CC2650 and a DevPack emulator used for flashing and debugging. To get started quickly, the BLE *target* used in this guide is a CC2540EMK-USB as it's quickly flashed and debugged easily using a CC Debugger. For more detailed information on BLE technology and the TI BLE protocol stack, please consult the Texas Instruments Bluetooth® low energy Software Developer's Guide, which is included in the BLE-STACK V2.2.

Contents

[hide]

Software Overview[edit]

The following sections give a brief overview of the software included in the hid_adv_remote project, which can be found in C:\ti\simplelink\ble_sdk_2_02_01_18\examples\cc2650rc. The different sections are divided by the source file structure in the hid_adv_remote project workspace.

Application[edit]

Porject Files	Description

hid_adv_remote.c	Top level application. Initialization of hardware, connection settings, key handling and voice streaming control.
kcb.c	Key press handling.
util.c	This file contains utility functions commonly used by BLE applications for CC26xx with TI-RTOS.

The main application task can be found hid_adv_remote.c. Besides setting up advertisement and scan response data and connection related parameters in HIDAdvRemote_init, this task processes key press events and handles the API for voice streaming.

Profiles And Services[edit]

Porject Files	Description
audio_profile.c	TI Audio Profile Voice over BLE service.
hiddev.c	HID Profile. Support HID reads, writes, queuing reports, register services and device states.
hidkbdccservice.c	HID Service. Service for keyboard and consumer control reports.
battservice.c	Battery Service.
devinfoservice.c	Device Information Service.
scanparamsservice.c	Scan Parameter Service.

HID Profile[edit]

All HOGP (HID OVER GATT PROFILE) related functionality is abstracted from the application task and is handled in a separate task implemented in hiddev.c. This task handles:

- Registering of all services mentioned in GATT Services section.
- Device idle timer to terminate connection after set idle period. The default idle period DEFAULT_HID_IDLE_TIMEOUT is set in hid_adv_remote.c to 60 seconds.
- Building and sending HID reports.
- Callbacks for read and write of characteristics in the HID service (hidkbdccservice.c).
- Advertising modes.

- Device states.

The Battery Service, Device Information Service and Scan Parameter Service are all adopted services and the specification for each service can be found at developer.bluetooth.org.

TI Audio Profile[edit]

The audio data is transmitted using a proprietary service with UUID 0xF000B00004514000B000000000000000. This service is composed of the following 2 characteristics:

UUID	Description	Properties
AUDIOPROFILE_START - 0xF000B00104514000B000000000000000	The start characteristic is used to transmit a start command before the streaming starts and a stop command as the last packet of a stream. The commands are defined in hid_adv_remote.c as: • Start Command : 0x04 • Stop Command : 0x00	GATT_PROP_READ, GATT_PROP_NOTIFY
AUDIOPROFILE_AUDIO - 0xF000B00204514000B000000000000000	AUDIOPROFILE_AUDIO is used as the audio stream characteristic, all audio frames will be transmitted using this characteristic.	GATT_PROP_READ, GATT_PROP_NOTIFY

The primary vehicle for streaming audio data is BLE notifications. This scheme was chosen in order to optimize throughput by selecting a message type that has little overhead. Before the voice stream begins, the peer device must enable notifications by writing 01:00 to the CCCD of the AUDIOPROFILE_START and AUDIOPROFILE_AUDIO. If notifications are not enabled, the remote will not stream voice data.

The basic flow of a voice transmission is:

1. CC2650RC sends a start command (0x04) notification (if enabled) on the AUDIOPROFILE_START characteristic.
2. CC2650RC starts streaming voice data. See Voice Over BLE section for more details
3. CC2650 sends a stop command (0x00) notification on the AUDIOPROFILE_START characteristic.

Drivers[edit]

Passkey Entry[edit]

The hid_adv_remote application has the IO capabilities “Keyboard Only” which enables the possibility to enter a passkey during the paring procedure. When the passkey is shown on the peer device, simply enter the passkey using the buttons 0-9 on the CC2650RC.

- To remove the passkey entry feature, the IO capabilities can be set to “NoInputNoOutput” by updating the DEFAULT_IO_CAPABILITIES in hid_adv_remote.c from

GAPBOND_IO_CAP_KEYBOARD_ONLY to GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT as:

```
// Default GAP bonding I/O capabilities
#define DEFAULT_IO_CAPABILITIES
GAPBOND_IO_CAP_NO_INPUT_NO_OUTPUT
```

HID Over GATT[edit]

The software supports the HID Over GATT Profile specification which was approved by the BT SIG in December 2011. For more information about the HID over GATT specifications refer to:

- HID over GATT profile, Version 1.0 (27-Dec-2011) [1]
- HID Service, Version 1.0 (27-Dec-2011) [2]

HID Terminology[edit]

Name	Description
HID Host	The target machine that the user interacts with (e.g. Laptop, tablet, smartphone, etc…)
Report Host	A Report Host is a HID Host that is required to support a HID Parser and be able to handle HID reports with arbitrary formats.
Boot Host	A Boot Host is a HID Host that is not required to support a HID Parser as all Reports for the Boot Protocol Mode have predefined length and format.
HID Device	The device that is used by the user to interact with the Host (e.g. Keyboard, mouse, remote control, game controller, etc...)
HID Report	A data message sent between the host and device. Input Reports go from HID Device to HID Host, such as a key press. Output reports go from HID Host to HID Device, such as a PC changing the caps lock LED on a keyboard.
HID Report Descriptor	A data structure that the device sends to the host which describes the HID device's capabilities, including the types, sizes, and directions of the reports that are supported. In HID over GATT this is also referred to as the **Report Map**.

GATT Services[edit]

The following services are defined in the HID profile specification as either mandatory or optional. The hid_adv_remote project includes all these services:

Service	Requirement	Supported
HID Service	Mandatory	Yes
Battery Service	Mandatory	Yes
Device Information Service	Mandatory	Yes
Scan Parameter Service	Optional	Yes

Report Map Characteristic[edit]

The Report Map characteristic contains the HID Report Descriptor which is used to define formatting information for Input Report, Output Report, and Feature Report data transferred between a HID Device and Report Host.

Each HID Service can only include one instance of the Report Map characteristic. The length of the Report Map characteristic value is limited to 512 bytes.

Report Map as shown at the right side is included in the HID Service in the hid_adv_remote project:

HID Reports[edit]

The Report Reference characteristic descriptors contain the Report ID and Report Type for each Report Characteristic defined in the Report Map. This information is used on the Report Host to route USB HID class driver data into and out of GATT characteristic values.

Report IDs in hidkbdccservice.h:

```
 // HID Report IDs for the service
#define HID_RPT_ID_KEY_IN        1  // Keyboard input report ID
#define HID_RPT_ID_CC_IN         2  // Consumer Control input report ID
#define HID_RPT_ID_LED_OUT       0  // LED output report ID
```

Report Types in hiddev.h:

```
 /* HID Report type */
#define HID_REPORT_TYPE_INPUT        1  // Keyboard input report ID
#define HID_REPORT_TYPE_OUTPUT       2  // Consumer Control input report ID
#define HID_REPORT_TYPE_FEATURE      3  // LED output report ID
```

The Report ID is not transmitted as a part of the HID Report over-the-air but is prepended by the Report Host, using the information in Report Reference characteristic descriptors, to each received HID report before passing them to a USB HID Class driver.

Key Board Input Reports[edit]

The Report Map includes the full 101-key part of the Keyboard Usage Page. This gives a one-to-one mapping between the transmitted report and the Usage ID in the usage table. The Usage IDs can be found in hiddev.c.

Consumer Control Input Reports[edit]

Only a subset of the Consumer Control Usage Page is included in the Report Map. The HID Report transmitted is in this case not the Usage ID but the index of the report as include in the Report Map.

The Consumer Control reports that can be sent can be found in hid_adv_remote.h as:

```
 // HID Consumer Control keycodes
// (based on the HID Report Map characteristic value)
#define HID_CC_RPT_POWER                    1
#define HID_CC_RPT_PLAY_PAUSE               2
#define HID_CC_RPT_STOP                     3
#define HID_CC_RPT_SCAN_NEXT_TRK            4
#define HID_CC_RPT_SCAN_PREV_TRK            5
#define HID_CC_RPT_FAST_FWD                 6
#define HID_CC_RPT_REWIND                   7
#define HID_CC_RPT_RECORD                   8
#define HID_CC_RPT_VOLUME_UP                9
#define HID_CC_RPT_VOLUME_DOWN              10
#define HID_CC_RPT_MUTE                     11
```

LED Output Reports[edit]

The LED Output reports are set in the Report Map to be a 5 bit bit-vector where each bit represents a report value. When an output is received on the CC2650RC it will be passed to the application callback in hid_adv_remote.c:

```
 static uint8_t HIDAdvRemote_handleReportCB(uint8_t id, uint8_t type,
                                            uint16_t uuid, uint8_t oper,
                                            uint16_t *pLen, uint8_t *pData)
{
  // Intentionally left blank
  return SUCCESS;
}
```

No specific handling of received Output Reports is implanted hid_adv_remote project so the correct handling must be added to use the information in the LED Output Report.

Voice Over BLE[edit]

The CC2650 samples audio data from the digital PDM microphone on the CC2650RC using the I2S interface. The PDM driver performs decimation of the PDM input to a PCM data stream with a 16kHz sample rate and 16 bit resolution. The PDM driver also encodes the PCM data using a software codec based on IMA ADPCM with a 4:1 compression rate.

The voice quality has been qualified by Nuance and is sufficient for voice recognition solutions.

The flow of voice data is as follows: PDM microphone --> Sampled by TI-RTOS PDM driver --> Compressed to 4:1 ADPCM using TI firmware codec --> Sent over BLE Audio Profile as described above

PDM Driver and IMA ADPCM Codec[edit]

More information on the PDM driver can be found at PDM Driver Overview

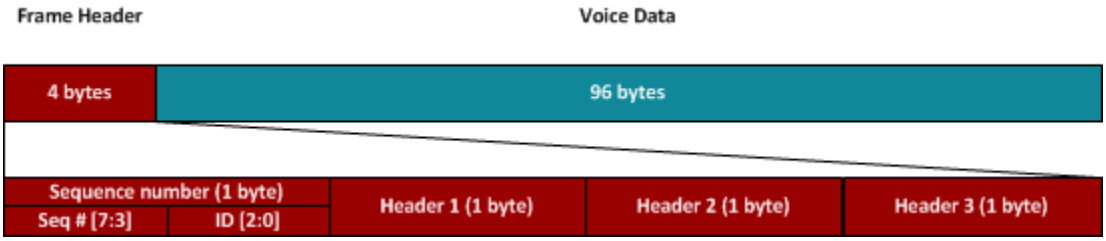
The PDM driver uses the I2S hardware module to sample the PDM microphone, the (PDM specific) I2S driver can be found PDM Driver Util (I2S) Overview

If `applyCompression` is enabled in the PDM Driver parameters (remote enables this by default), then the PDM driver will use the IMA ADPCM codec implementation bundled with the TI-RTOS drivers. This sample implementation can be found within the `\tirtos_cc13xx_cc26xx_2_20_01_08\tidrivers_cc13xx_cc26xx_2_20_01_10\packages\ti\drivers\pdm\Codec1.*` files.

Audio Streaming Format[edit]

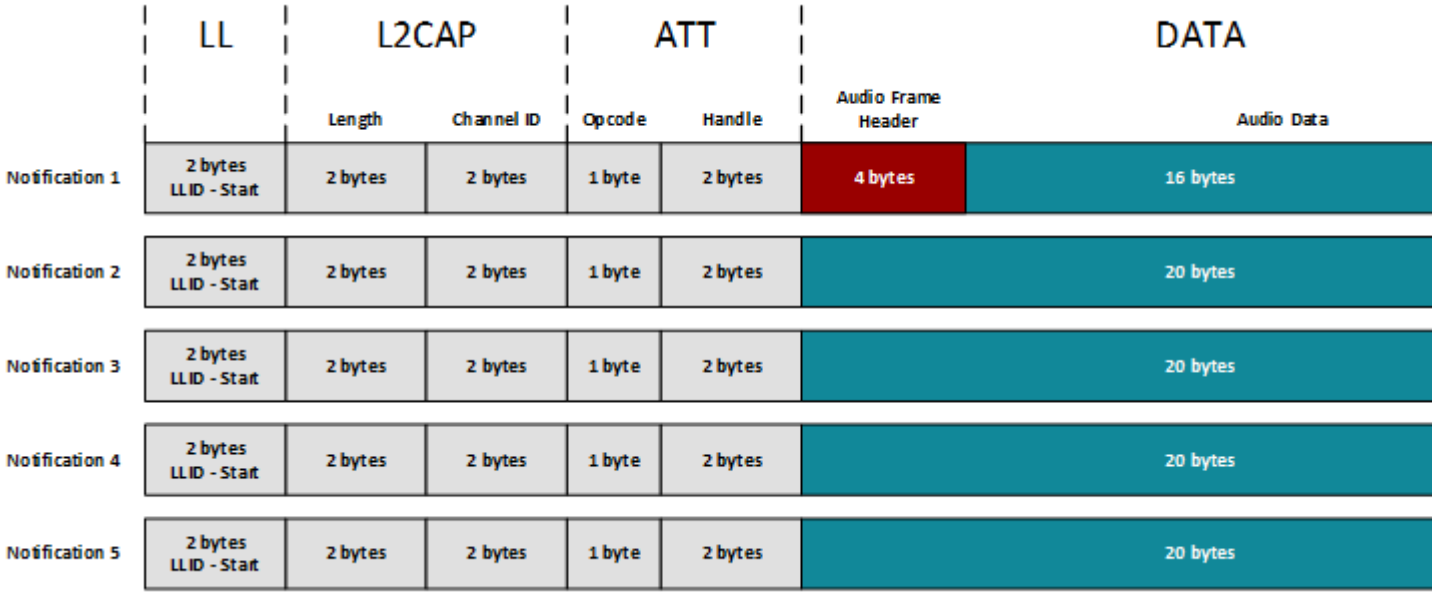
The encoded audio data is packaged in 100 byte audio frames (including 4 bytes header) each containing 12 ms of compressed audio:
((100 - 4) * 8)/(4 * 16k) = 12(ms)
Frame headers 1, 2, and 3 are generated by the TI-RTOS PDM driver's `PDMCC26XX_metaData` data structure's `si` and `pv` fields. See the PDM Metadata.

The numbered headers in the audio frame above are the metadata fields provided by the PDM driver. These fields are required to be sent over the air so that the receiving device can decode the audio stream.



Name	Description
Header 1	SI
Header 2	PV (low)

Header 3	PV (high)



When transmitted over the air, the audio frames are fragmented into 20 byte notifications, this means that each audio frame is sent as 5 notifications:

The built in flow control in the Bluetooth low energy Protocol is used to ensure delivery of full audio frames during streaming. Since the header of each audio frame contain the information to be able to decode that frame separately, this is the safest way to discard data in e.g. a noisy environment. The PDM driver will drop full audio frames when there are no available buffers, and the application will handle one frame at a time until it has been successfully queued up in the TX FIFO.

Throughput Requirement[edit]

The audio data is transmitted as ~83 audio frames per second(1/12ms ~= 83.3) corresponding to ~417 (83.3 * 5)audio notifications per second.
This gives the following throughput requirement:

- Audio data throughput (audio frame data only, 20 bytes) = 417 * 20 * 8 ≈ 66.7kbps (64kbps audio data + audio frame header data)
- Payload throughput (including L2CAP and ATT header, 27 bytes) = 417 * 27 * 8 ≈ 90kbps
- Packet throughput (complete packets transmitted, 41 bytes) = 417 * 41 * 8 ≈ 136.8kbps

The hid_adv_remote application will try to transmit as many available audio notifications as possible for every connection event. This means that the required throughput can be obtained with different settings for the connection interval as longs as enough packets can be transmitted in each connection event to successfully reach the ~417 notifications per second limit. Typically a connection interval of 10ms can be used where 3-5 notifications are transmitted every connection event.

Receiver Implementation[edit]

A sample receiver (host) implementation has been created to work with the voice streaming feature of the HID Advanced Remote projects and the sensortag_audio projects. This can be found on the TI-SimpleLink BLE Examples page:

ble_examples Github --> See simple_central_audio_receiver