CC2564 HFP / HID / SPP Integration Demonstration

Rio Chan

ABSTRACT

This application note demonstrates the HID/HFP/SPP integrated steps. These steps assist with IOT / BT voice recognition applications.

Contents

List of Figures

List of Tables

Trademarks

Bluetooth is a registered trademark of Bluetooth SIG.
All other trademarks are the property of their respective owners.

1 Introduction

The purpose of this document is to help the user to understand how to adapt the TI Audio codec using the HFP voice flow scenarios.

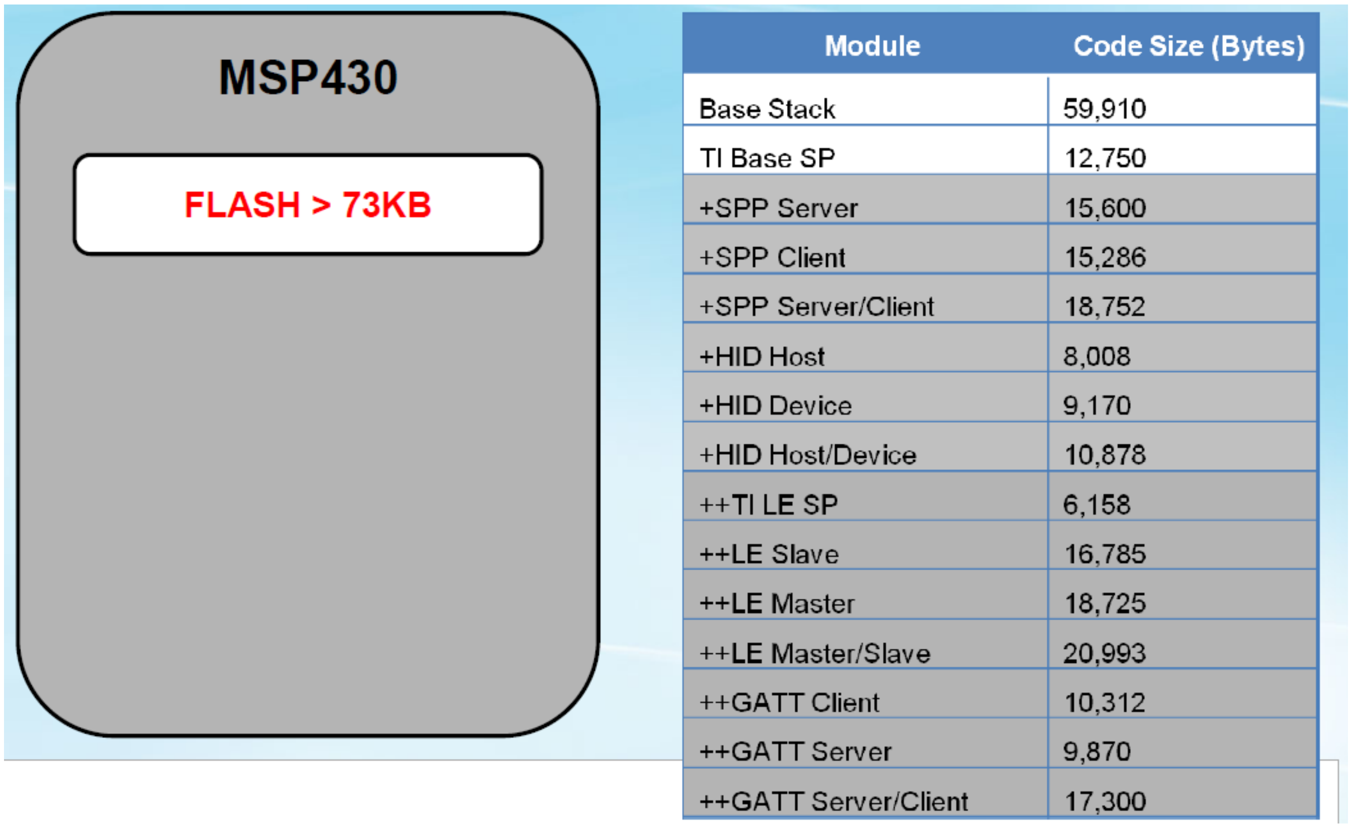
Before starting, evaluate the code size and choose the right flash size MCU.

The BT_Stack + TI Base SP (Service pack – CC2564 FW) + SPP + HID = 59 +12 + (15 +15) + (8 + 9) = 118K.

Checking the HFP-built MAP file, the HFP occupies = 20 + 14 = 34K.

Total requried flash size is then 118 + 34 = 152K.

Picking up the MSP430 with 256K flash is a suitable platform. This document will use the MSP430F5335 an an example.


MSP430
FLASH > 73KB

Module	Code Size (Bytes)
Base Stack	59,910
TI Base SP	12,750
+SPP Server	15,600
+SPP Client	15,286
+SPP Server/Client	18,752
+HID Host	8,008
+HID Device	9,170
+HID Host/Device	10,878
++TI LE SP	6,158
++LE Slave	16,785
++LE Master	18,725
++LE Master/Slave	20,993
++GATT Client	10,312
++GATT Server	9,870
++GATT Server/Client	17,300

Figure 1. Stack and Profile Code Size

2 IAR / CC2564B SDK Installation

TI recommends pre-installing these items:

- IAR for TI MCU: https://www.iar.com/iar-embedded-workbench/partners/texas-instruments/. The recommended version is 5.61 for TI's MSP430.
- CC2564B SDK: http://www.ti.com/tool/tibluetoothstack-sdk, http://processors.wiki.ti.com/index.php/CC256x_TI's_Bluetooth_Stack
- MSP430 Download Tool: https://www.elprotronic.com/products?show&id=35
- TI AIC3204 codec: http://www.ti.com/product/TLV320AIC3204

3 System Block Diagram

The system block diagram is shown in Figure 2.

The CC2564B acts as a transceiver. The MSP430 runs the BT stack and profiles.

The MSP430 connects the CC2564B with the UART for the HCI command RX/TX. The MSP430 connects with the AIC 3204 (Audio Codec), for controlling the I2C commands. The CC2564B connects the AIC 3204 (Audio Codec) with the PCM. MIC is the Microphone for Voice Input. SPK / headset is the speaker for Voice Output.

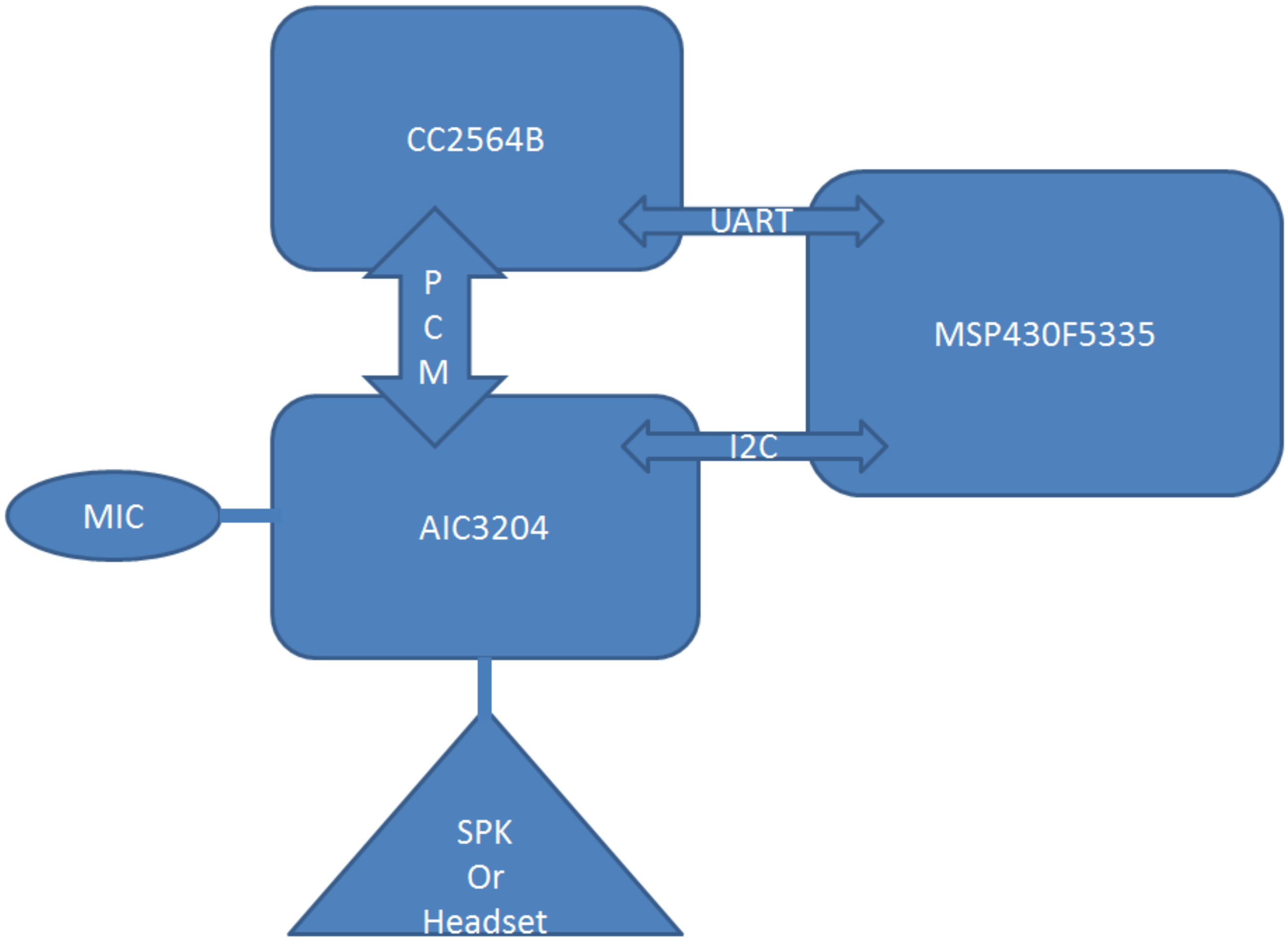


Figure 2. System Architecture

4 Knowing the Profiles

According to the wiki, the CC2564SDK on the MSP430 platform has these profile versions:

- SPP version 1.1
- HID version 1.0
- HSP version 1.2 / HFP version 1.6

For more information on each profile, refer to https://en.wikipedia.org/wiki/List_of_Bluetooth_profiles#Human_Interface_Device_Profile_(HID).

4.1 SPP Profile

Refer to this BT SIG Spec for more detail:
http://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=8700.

The SPP profile works in the following way. Set up virtual serial ports (or equivalent) on two devices (such as PCs) and connect these with Bluetooth® to emulate a serial cable between the two devices. Any legacy application may be run on either device, using the virtual serial port as if there were a physical serial cable connecting the two devices (with RS232 control signaling).

4.2 HID Profile

Refer to this BT SIG Spec for more detail:
https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=246761.

The HID profile works in the following way. The Bluetooth HID Protocol (HIDP) operates over the Bluetooth L2CAP layer and is an adaptation of the USB HID Protocol. The USB HID specification [5] defines the Control and Interrupt channels, which are mapped to the USB Control and Interrupt pipes. This profile defines analogous logical channels, which are mapped to L2CAP channels. See the BT sig spec: HID_SPEC_V11.pdf, section 5.2 for details on how the L2CAP channels are opened and configured.

4.3 HFP Pofile

Refer to this BT SIG Spec for more detail:
https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=292287.

The HFP profile works in the following way. This profile lets two sides talk using voice over BT. (IE: HFP). There are two roles defined for this profile:

- Audio gateway (AG) – This device is the gateway of the audio, both for input and output. Typical devices acting as audio gateways are cellular phones.
- Hands-free unit (HF) – This device acts as the audio gateway’s remote audio input and output mechanism. It also provides some means for remote control.

5 How to Merge SPP / HID / HFP Profiles

Start with the SDK code arch. The upper part of Figure 3 shows the exampel arch of the three profiles. To merge those profiles examples, first identify the common parts. After the merge, the arch of the merged demo should resemble the lower part of Figure 3. All the CC2564 SDK profiles have these two common basis parts:

- CC2564 common API:

```
//CC2564 Common API
static int SetDisc(void);
static int SetConnect(void);
static int SetPairable(void);
static int DeleteLinkKey(BD_ADDR_t BD_ADDR);
static int InitializeHIDDevice(void);
static int OpenStack(HCI_DriverInformation_t *HCI_DriverInformation, BTPS_Initialization_t
*BTPS_Initialization);
static int CloseStack(void);
static unsigned long StringToUnsignedInteger(char *StringInteger);
static char *StringParser(char *String);
static int CommandParser(UserCommand_t *TempCommand, char *UserInput);
static int CommandInterpreter(UserCommand_t *TempCommand);
static int AddCommand(char *CommandName, CommandFunction_t CommandFunction);
static CommandFunction_t FindCommand(char *Command);
sstatic void ClearCommands(void);
static void BD_ADDRToStr(BD_ADDR_t Board_Address, char *BoardStr);
static void DisplayPrompt(void);
static void DisplayFunctionError(char *Function, int Status);
static void UserInterface(void);
static Boolean_t CommandLineInterpreter(char *Command);
```

These are dedicated for use by the BT connection / stack / command interpreter.

- CC2564 common user functions:

```
//CC2564 common user functions
static int Inquiry(ParameterList_t *TempParam);
static int DisplayInquiryList(ParameterList_t *TempParam);
static int SetDiscoverabilityMode(ParameterList_t *TempParam);
static int SetConnectabilityMode(ParameterList_t *TempParam);
static int SetPairabilityMode(ParameterList_t *TempParam);
static int ChangeSimplePairingParameters(ParameterList_t *TempParam);
static int Pair(ParameterList_t *TempParam);
static int EndPairing(ParameterList_t *TempParam);
static int PINCodeResponse(ParameterList_t *TempParam);
static int PassKeyResponse(ParameterList_t *TempParam);
static int UserConfirmationResponse(ParameterList_t *TempParam);
static int GetLocalAddress(ParameterList_t *TempParam);
static int SetLocalName(ParameterList_t *TempParam);
static int GetLocalName(ParameterList_t *TempParam);
static int SetClassOfDevice(ParameterList_t *TempParam);
static int GetClassOfDevice(ParameterList_t *TempParam);
static int GetRemoteName(ParameterList_t *TempParam);
static int DisplayHelp(ParameterList_t *TempParam);
```

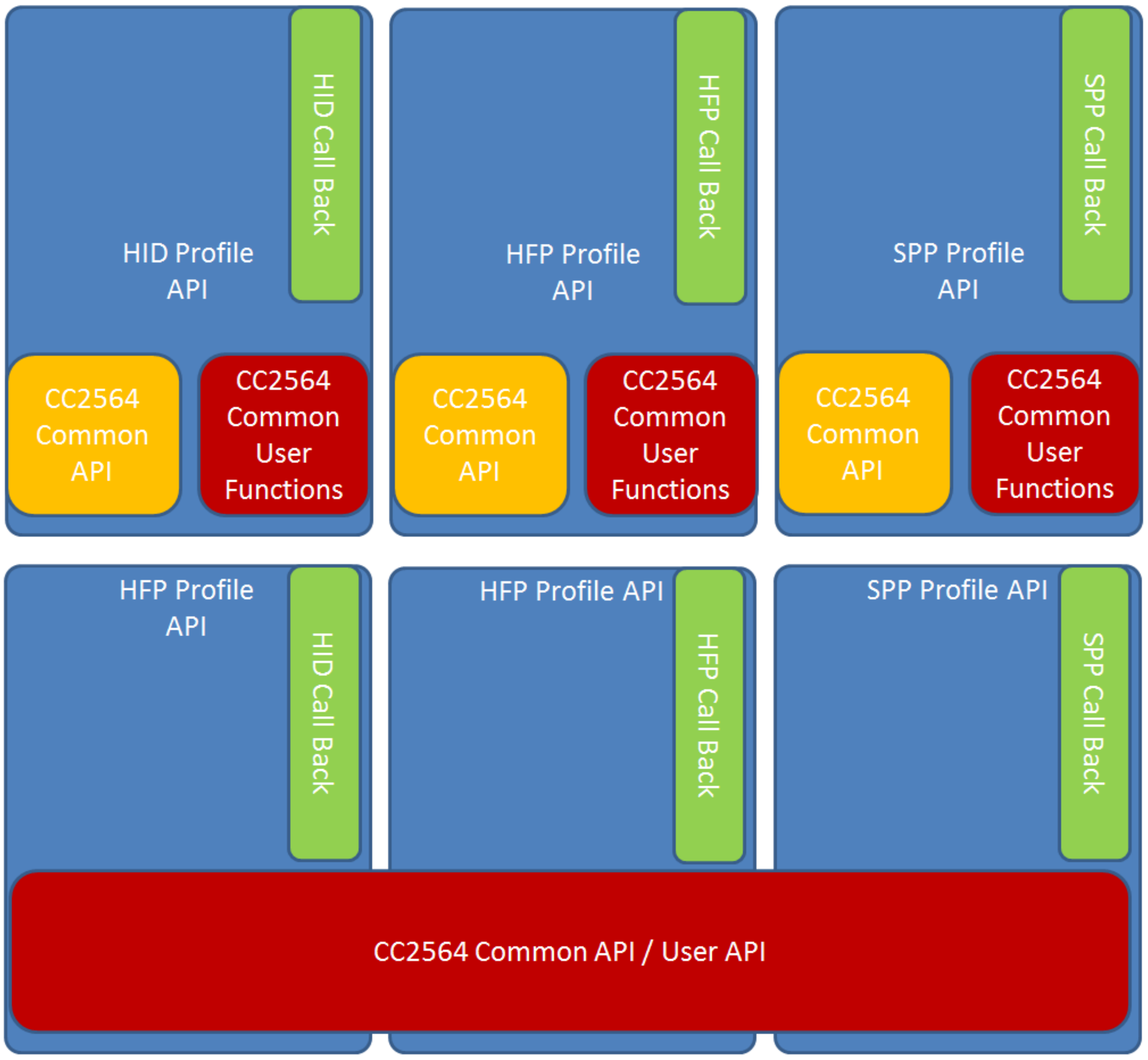


Figure 3. SDK API Architecture

At this point, the user must know how many APIs are dedicated for HID/HFP/SPP. The APIs are listed below.

```
//HFP profile common functions
static void FormatEIRData(unsigned int BluetoothStackID);
static int  UpdateEIRData(char *DeviceName);
static int HFRESetupAudioConnection(void);
static int HFREReleaseAudioConnection(void);
static int OpenHFServer(ParameterList_t *TempParam);
static int CloseHFServer(ParameterList_t *TempParam);
static int ClosePort(ParameterList_t *TempParam);
static int ManageAudioConnection(ParameterList_t *TempParam);
static int AnswerIncomingCall(ParameterList_t *TempParam);
static int HangUpCall(ParameterList_t *TempParam);

//HID profile common functions.
static void DisplayHIDDeviceMenu(void);
static void PopulateHIDDeviceCommandTable(void);
static int HIDRegisterDeviceServer(void);
static int HIDConnectRemoteHost(ParameterList_t *TempParam);
static int HIDCloseConnection(ParameterList_t *TempParam);
static int HIDControlRequest(ParameterList_t *TempParam);
static int HIDGetReportResponse(ParameterList_t *TempParam);
static int HIDSetReportResponse(ParameterList_t *TempParam);
static int HIDGetProtocolResponse(ParameterList_t *TempParam);
static int HIDSetProtocolResponse(ParameterList_t *TempParam);
static int HIDGetIdleResponse(ParameterList_t *TempParam);
static int HIDSetIdleResponse(ParameterList_t *TempParam);
static int HIDDataWrite(ParameterList_t *TempParam);
static int HIDDeviceMode(ParameterList_t *TempParam);//Rio

//SPP profile common functions.
static int OpenSPPServer(ParameterList_t *TempParam);
static int CloseSPPServer(ParameterList_t *TempParam);
```

The final part is the Callback API for each profile.

```
//Callbacks.
   /* Callback Function Prototypes.                                     */
static void BTPSAPI GAP_Event_Callback(unsigned int BluetoothStackID, GAP_Event_Data_t
*GAPEventData, unsigned long CallbackParameter);

static void BTPSAPI HID_Event_Callback(unsigned int BluetoothStackID, HID_Event_Data_t
*HIDEventData, unsigned long CallbackParameter);

static void BTPSAPI HFRE_Event_Callback(unsigned int BluetoothStackID, HFRE_Event_Data_t
*HFRE_Event_Data, unsigned long CallbackParameter);

static void BTPSAPI SPP_Event_Callback(unsigned int BluetoothStackID, SPP_Event_Data_t
*SPP_Event_Data, unsigned long CallbackParameter);
```

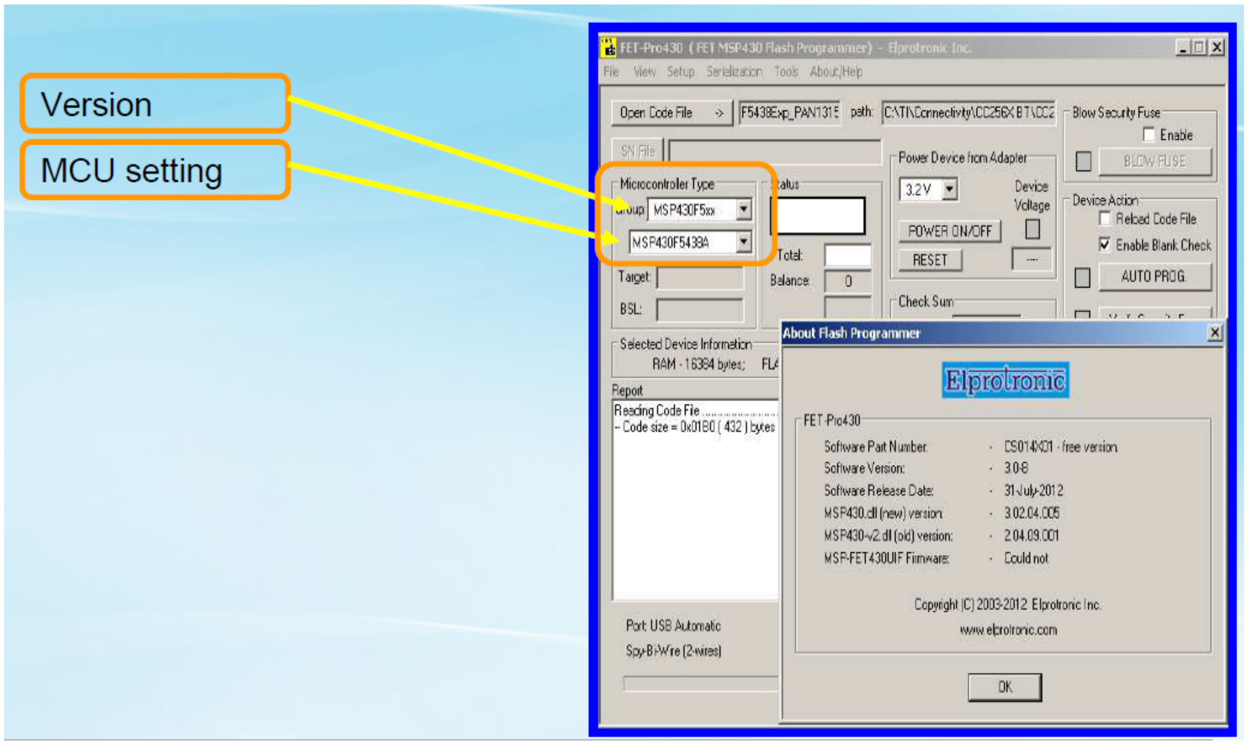
The steps are listed below:

1. Use the HFP example code as the basis, because it is the most complicated.
2. Use the HID / SPP example code, and merge those two profiles onto the HFP example basis.
3. Remove the redundant functions, as there are common implemented functions.
4. Build the code first, then solve the compile error.

6 How to Flash the Image

Follow these steps to flash the image:

1. Select the MCU family and correct version, as shown in Figure 4.



FET-Pro430 (FET MSP430 Flash Programmer) - Elprotronic Inc.

File View Setup Serialization Tools About/Help

Open Code File -> F5438Exp_PAN1315 path: C:\TI\Connectivity\CC256X BT\CC2

Microcontroller Type
Group: MSP430F5xx
MSP430F5438A

Power Device from Adapter: 3.2 V, Device Voltage

Blow Security Fuse: [ ] Enable

Device Action: [ ] Reload Code File, [✓] Enable Blank Check

Selected Device Information: RAM - 16384 bytes;

Report: Reading Code File .... -- Code size = 0x01B0 (432) bytes

Port: USB Automatic
Spy-Bi-Wire (2-wires)

About Flash Programmer

Elprotronic

FET-Pro430

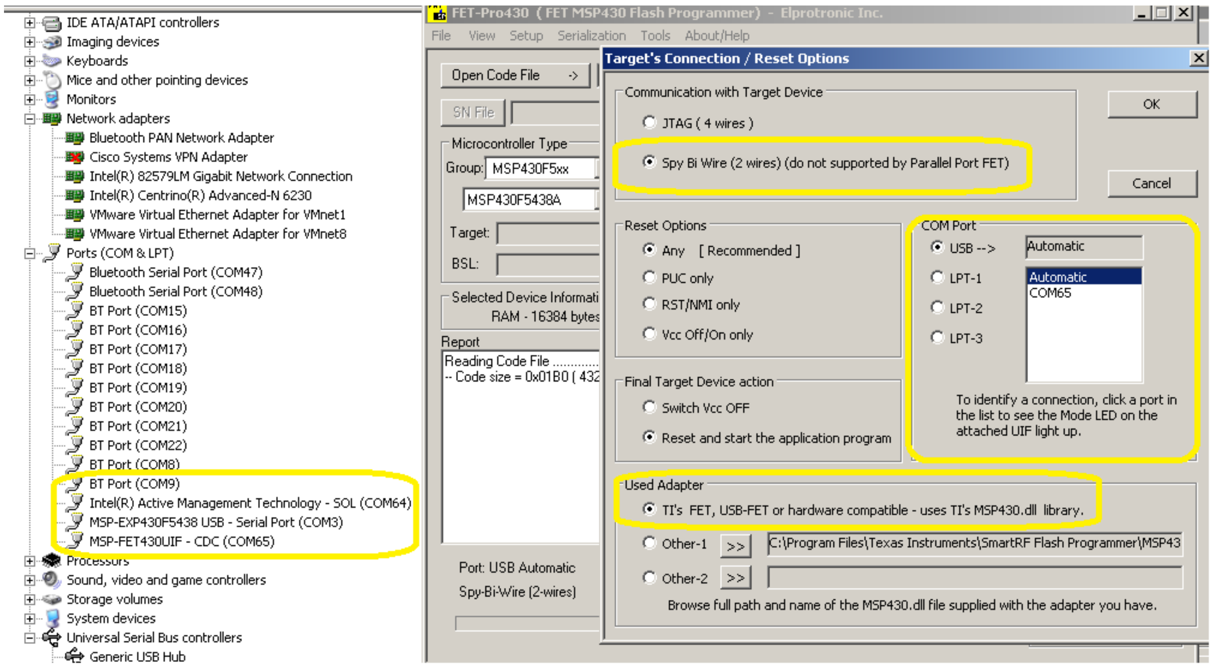
Software Part Number:	CS014X01 - free version
Software Version:	3.0-8
Software Release Date:	31-July-2012
MSP430.dll (new) version:	3.02.04.005
MSP430-v2.dll (old) version:	2.04.09.001
MSP-FET430UIF Firmware:	Could not

Copyright (C) 2003-2012 Elprotronic Inc.
www.elprotronic.com

OK

Figure 4. FETPro430 Setting MCU

2. Select the “Spy by Wires” / “Any” / “Used Adapter” / “Connect type”, as shown in Figure 5.



IDE ATA/ATAPI controllers
Imaging devices
Keyboards
Mice and other pointing devices
Monitors
Network adapters
- Bluetooth PAN Network Adapter
- Cisco Systems VPN Adapter
- Intel(R) 82579LM Gigabit Network Connection
- Intel(R) Centrino(R) Advanced-N 6230
- VMware Virtual Ethernet Adapter for VMnet1
- VMware Virtual Ethernet Adapter for VMnet8

Ports (COM & LPT)
- Bluetooth Serial Port (COM47)
- Bluetooth Serial Port (COM48)
- BT Port (COM15)
- BT Port (COM16)
- BT Port (COM17)
- BT Port (COM18)
- BT Port (COM19)
- BT Port (COM20)
- BT Port (COM21)
- BT Port (COM22)
- BT Port (COM8)
- BT Port (COM9)
- Intel(R) Active Management Technology - SOL (COM64)
- MSP-EXP430F5438 USB - Serial Port (COM3)
- MSP-FET430UIF - CDC (COM65)

Processors
Sound, video and game controllers
Storage volumes
System devices
Universal Serial Bus controllers
- Generic USB Hub

FET-Pro430 (FET MSP430 Flash Programmer) - Elprotronic Inc.

Microcontroller Type — Group: MSP430F5xx, MSP430F5438A

Selected Device Information: RAM - 16384 bytes

Report: Reading Code File .... -- Code size = 0x01B0 (432

Port: USB Automatic
Spy-Bi-Wire (2-wires)

Target's Connection / Reset Options

Communication with Target Device
- ( ) JTAG (4 wires)
- (●) Spy Bi Wire (2 wires) (do not supported by Parallel Port FET)

Reset Options
- (●) Any [Recommended]
- ( ) PUC only
- ( ) RST/NMI only
- ( ) Vcc Off/On only

COM Port
- (●) USB --> Automatic
- ( ) LPT-1
- ( ) LPT-2
- ( ) LPT-3

Automatic
COM65

To identify a connection, click a port in the list to see the Mode LED on the attached UIF light up.

Final Target Device action
- ( ) Switch Vcc OFF
- (●) Reset and start the application program

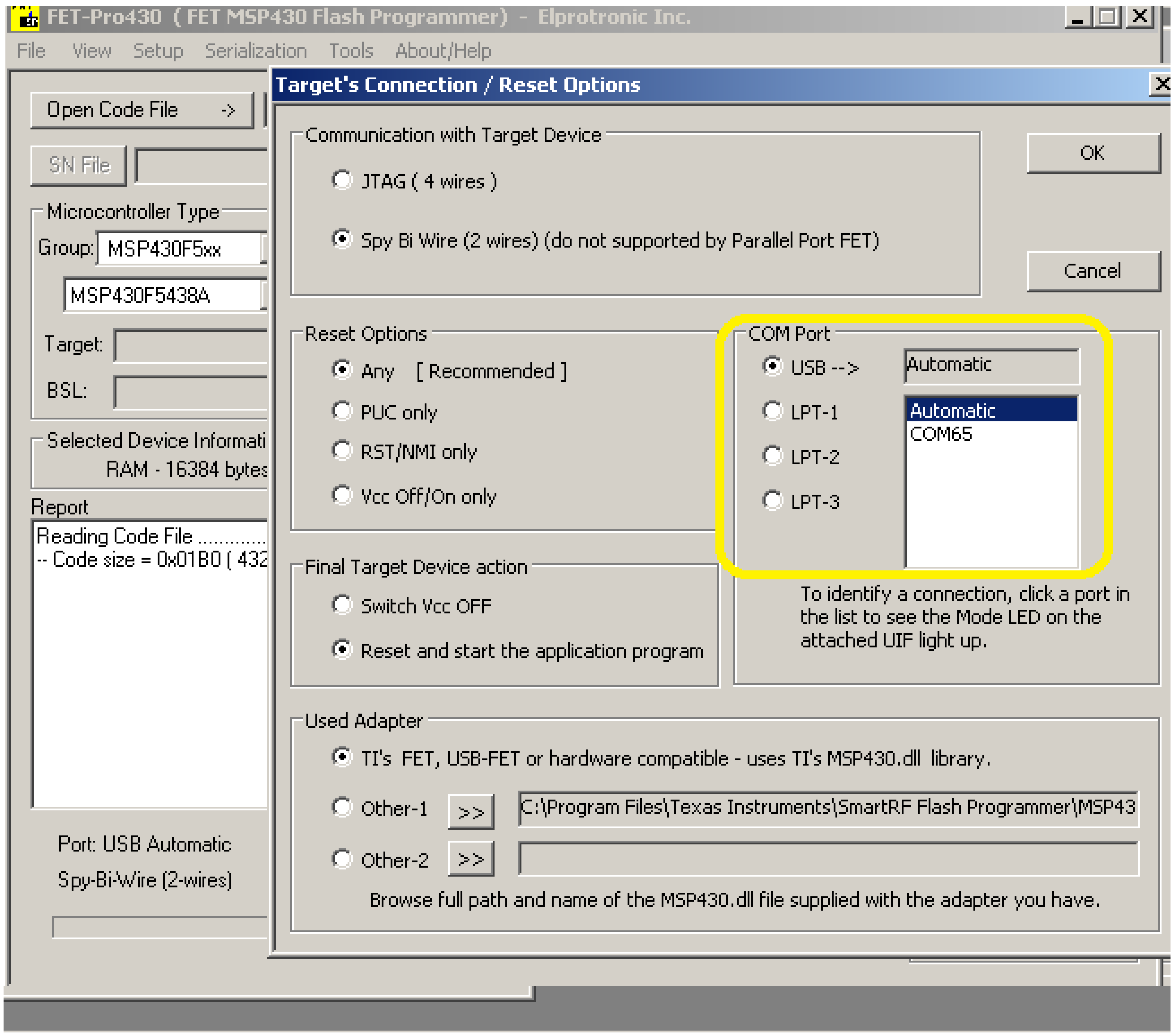
Used Adapter
- (●) TI's FET, USB-FET or hardware compatible - uses TI's MSP430.dll library.
- ( ) Other-1 >> C:\Program Files\Texas Instruments\SmartRF Flash Programmer\MSP43
- ( ) Other-2 >>

Browse full path and name of the MSP430.dll file supplied with the adapter you have.

OK Cancel

Figure 5. Connect Type

3. The NB has two USB cables to link to the EVM. One is used for the FETPro430 kit, and the other is used for the MSP430. Thus, there are two interfaces shown in the Device Manager. Select the Automatic option for downloading the image in the FETPro430.


FET-Pro430 (FET MSP430 Flash Programmer) - Elprotronic Inc.
File View Setup Serialization Tools About/Help
Open Code File ->
SN File
OK
Cancel
>>
>>

Target's Connection / Reset Options

Communication with Target Device
( ) JTAG (4 wires)
(•) Spy Bi Wire (2 wires) (do not supported by Parallel Port FET)

Reset Options
(•) Any [Recommended]
( ) PUC only
( ) RST/NMI only
( ) Vcc Off/On only

COM Port
(•) USB --> Automatic
( ) LPT-1
( ) LPT-2
( ) LPT-3
Automatic
COM65

Final Target Device action
( ) Switch Vcc OFF
(•) Reset and start the application program

To identify a connection, click a port in the list to see the Mode LED on the attached UIF light up.

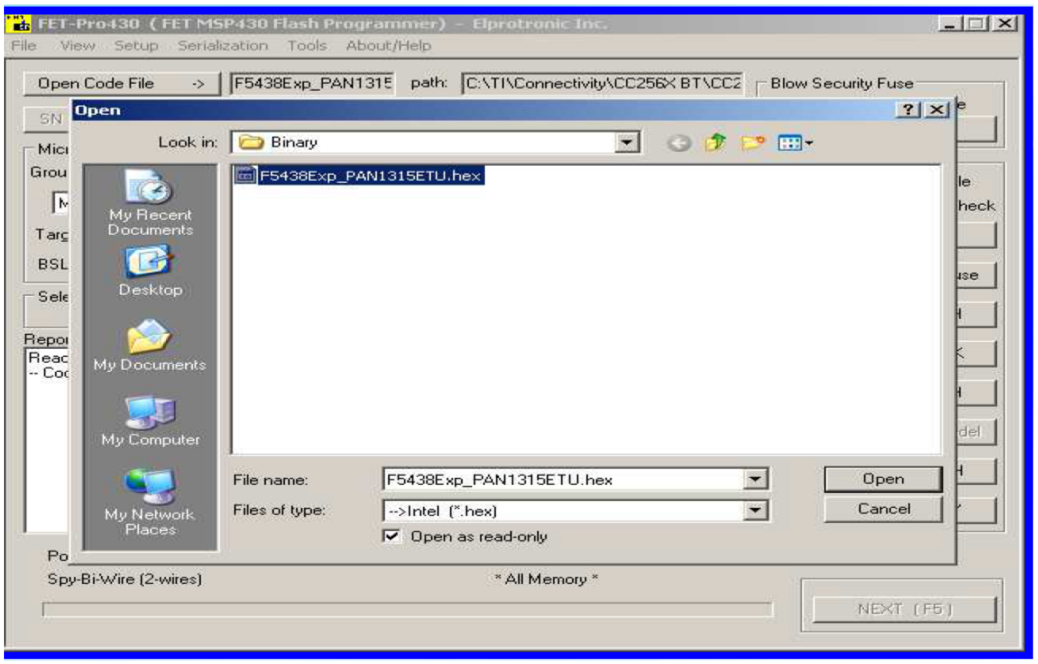
Used Adapter
(•) TI's FET, USB-FET or hardware compatible - uses TI's MSP430.dll library.
( ) Other-1 C:\Program Files\Texas Instruments\SmartRF Flash Programmer\MSP43
( ) Other-2
Browse full path and name of the MSP430.dll file supplied with the adapter you have.

Microcontroller Type
Group: MSP430F5xx
MSP430F5438A
Target:
BSL:
Selected Device Informati
RAM - 16384 bytes
Report
Reading Code File ....
-- Code size = 0x01B0 (432
Port: USB Automatic
Spy-Bi-Wire (2-wires)

Figure 6. Select Automatic Option

4. Go to the FETPro430 Setup, then to Connection/Device Reset.
5. Choose the SPY biware, then choose TI's MSP430.dll library.
6. Plug the FET device and check if Windows can detect it or not. If yes, the FET driver appears in the Windows device manager.

7. Select the .Hex file to download.



F5438Exp_PAN1315 path: C:\TI\Connectivity\CC256X BT\CC2

Look in: Binary

F5438Exp_PAN1315ETU.hex

File name: F5438Exp_PAN1315ETU.hex

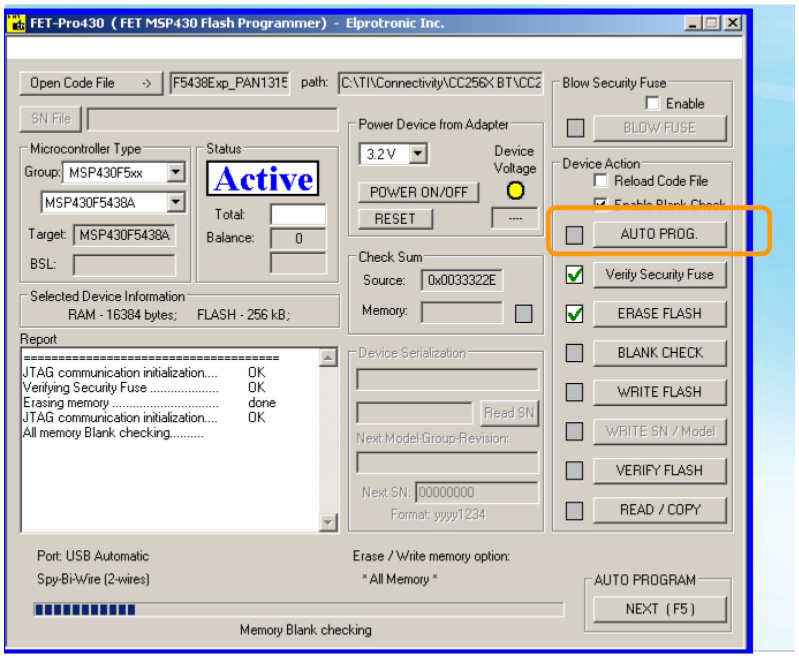
Files of type: -->Intel (*.hex)

☑ Open as read-only

Spy-Bi-Wire (2-wires) * All Memory *

Figure 7. FETPro430 File Selection Type

8. Select the correct "Power From Device" voltage, according to the MCU specification.
9. Click "Auto Prog".



F5438Exp_PAN1315 path: C:\TI\Connectivity\CC256X BT\CC2

Microcontroller Type
Group: MSP430F5xx
MSP430F5438A
Target: MSP430F5438A
BSL:

Status: Active
Total:
Balance: 0

Power Device from Adapter: 3.2 V

Blow Security Fuse: ☐ Enable

Device Action: ☐ Reload Code File, ☑ Enable Blank Check

☑ Verify Security Fuse
☑ ERASE FLASH

Check Sum
Source: 0x0033322E
Memory:

Selected Device Information
RAM - 16384 bytes; FLASH - 256 kB;

Device Serialization
Next Model-Group-Revision:
Next SN: 00000000
Format: yyyy1234

Report
```
=======================================
JTAG communication initialization....    OK
Verifying Security Fuse ..................    OK
Erasing memory ...............................    done
JTAG communication initialization....    OK
All memory Blank checking..........
```

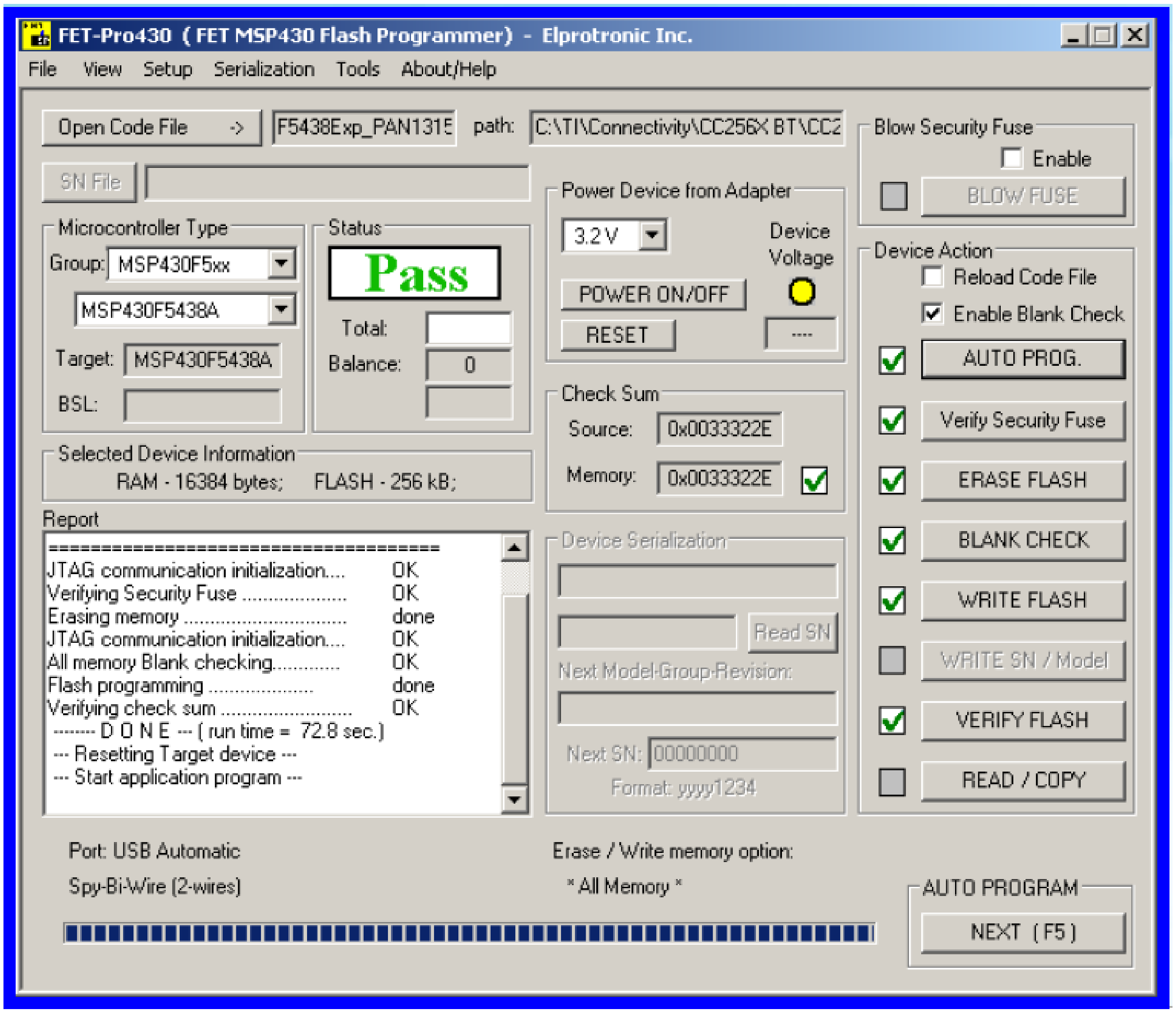
Port: USB Automatic
Spy-Bi-Wire (2-wires)

Erase / Write memory option:
* All Memory *

Memory Blank checking

Figure 8. Auto Programming

10. Wait until it finishes.



Code file: F5438Exp_PAN1315 path: C:\TI\Connectivity\CC256X BT\CC2

Microcontroller Type — Group: MSP430F5xx; MSP430F5438A; Target: MSP430F5438A; BSL:

Status: Pass; Total: ; Balance: 0

Device Voltage: 3.2 V

Check Sum — Source: 0x0033322E; Memory: 0x0033322E

Next SN: 00000000

Selected Device Information: RAM - 16384 bytes; FLASH - 256 kB;

Report

```
========================================
JTAG communication initialization....      OK
Verifying Security Fuse ..................  OK
Erasing memory ...........................  done
JTAG communication initialization....      OK
All memory Blank checking............      OK
Flash programming ...................      done
Verifying check sum ......................  OK
 ------- D O N E --- ( run time =  72.8 sec.)
 --- Resetting Target device ---
 --- Start application program ---
```

Port: USB Automatic
Spy-Bi-Wire (2-wires)

Erase / Write memory option:
* All Memory *

Figure 9. Finish Programming

7 Running the Demonstration

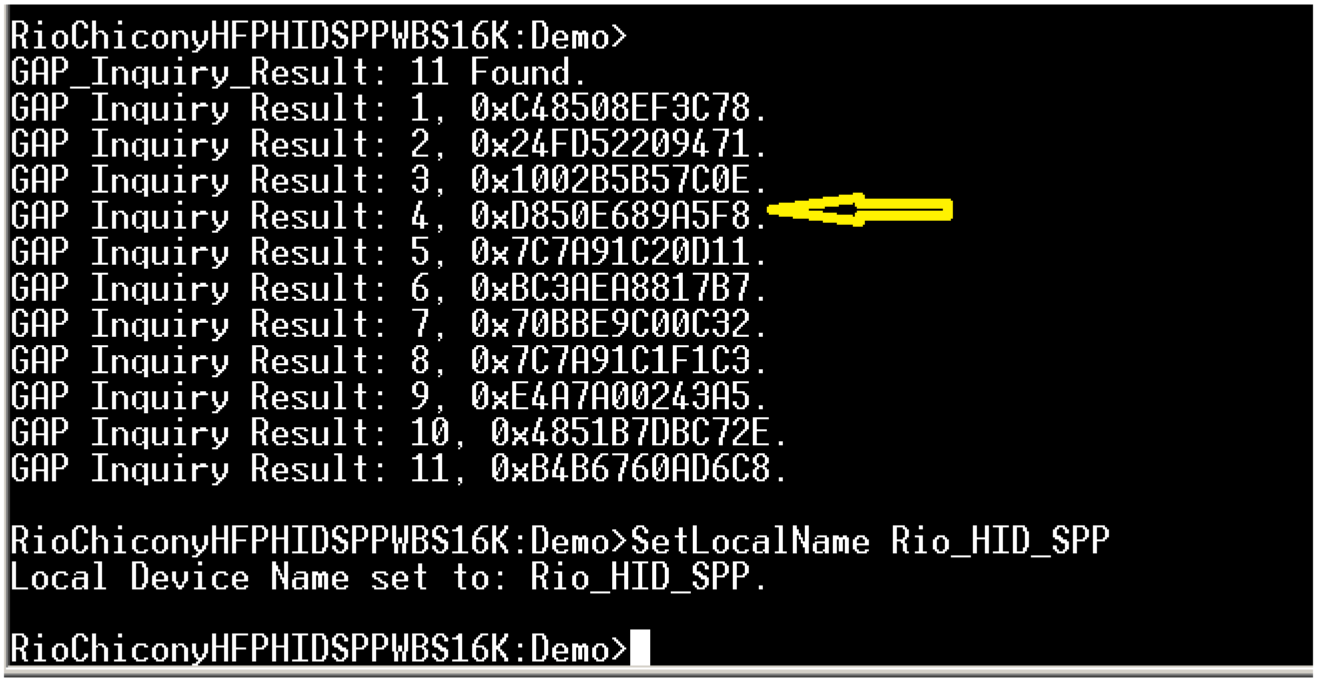
Vist Youtube to see the demo: https://www.youtube.com/watch?v=1RT72kdhJIl or search for “TI Rio Voice Command” on the Youtube search bar.

This demo uses several voice commands. The result is good if the pronunciation is correct.

7.1 How to Demo HID Over SPP

The HID is based on the SPP profile to send and receive the HID key event and data. Thus, the user can combine those two profile demonstrations into one scenario, as shown in the following steps. Type the following commands listed:

1. Set Server Mode.
2. Set Local Name Rio_HID_SPP.



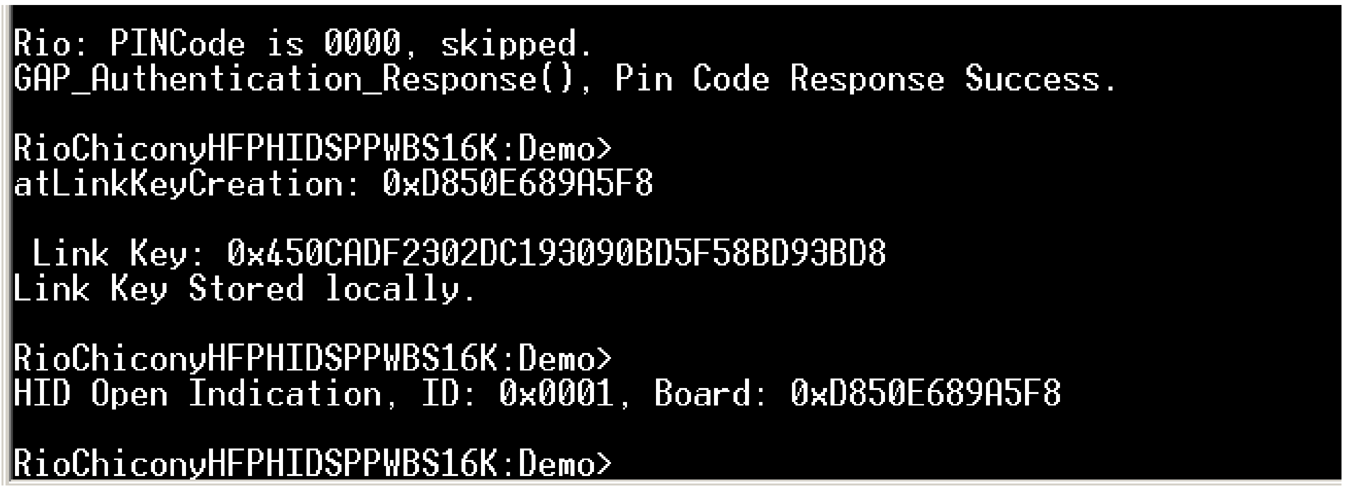
```
RioChiconyHFPHIDSPPWBS16K:Demo>
GAP_Inquiry_Result: 11 Found.
GAP Inquiry Result: 1, 0xC48508EF3C78.
GAP Inquiry Result: 2, 0x24FD52209471.
GAP Inquiry Result: 3, 0x1002B5B57C0E.
GAP Inquiry Result: 4, 0xD850E689A5F8.
GAP Inquiry Result: 5, 0x7C7A91C20D11.
GAP Inquiry Result: 6, 0xBC3AEA8817B7.
GAP Inquiry Result: 7, 0x70BBE9C00C32.
GAP Inquiry Result: 8, 0x7C7A91C1F1C3.
GAP Inquiry Result: 9, 0xE4A7A00243A5.
GAP Inquiry Result: 10, 0x4851B7DBC72E.
GAP Inquiry Result: 11, 0xB4B6760AD6C8.

RioChiconyHFPHIDSPPWBS16K:Demo>SetLocalName Rio_HID_SPP
Local Device Name set to: Rio_HID_SPP.

RioChiconyHFPHIDSPPWBS16K:Demo>
```

Figure 10. HID/SPP Demo Inquiry

3. Connect the Rio_HID_SPP from an Android phone.



```
Rio: PINCode is 0000, skipped.
GAP_Authentication_Response(), Pin Code Response Success.

RioChiconyHFPHIDSPPWBS16K:Demo>
atLinkKeyCreation: 0xD850E689A5F8

 Link Key: 0x450CADF2302DC193090BD5F58BD93BD8
Link Key Stored locally.

RioChiconyHFPHIDSPPWBS16K:Demo>
HID Open Indication, ID: 0x0001, Board: 0xD850E689A5F8

RioChiconyHFPHIDSPPWBS16K:Demo>
```

Figure 11. HID/SPP Demo Pairing (1 of 2)

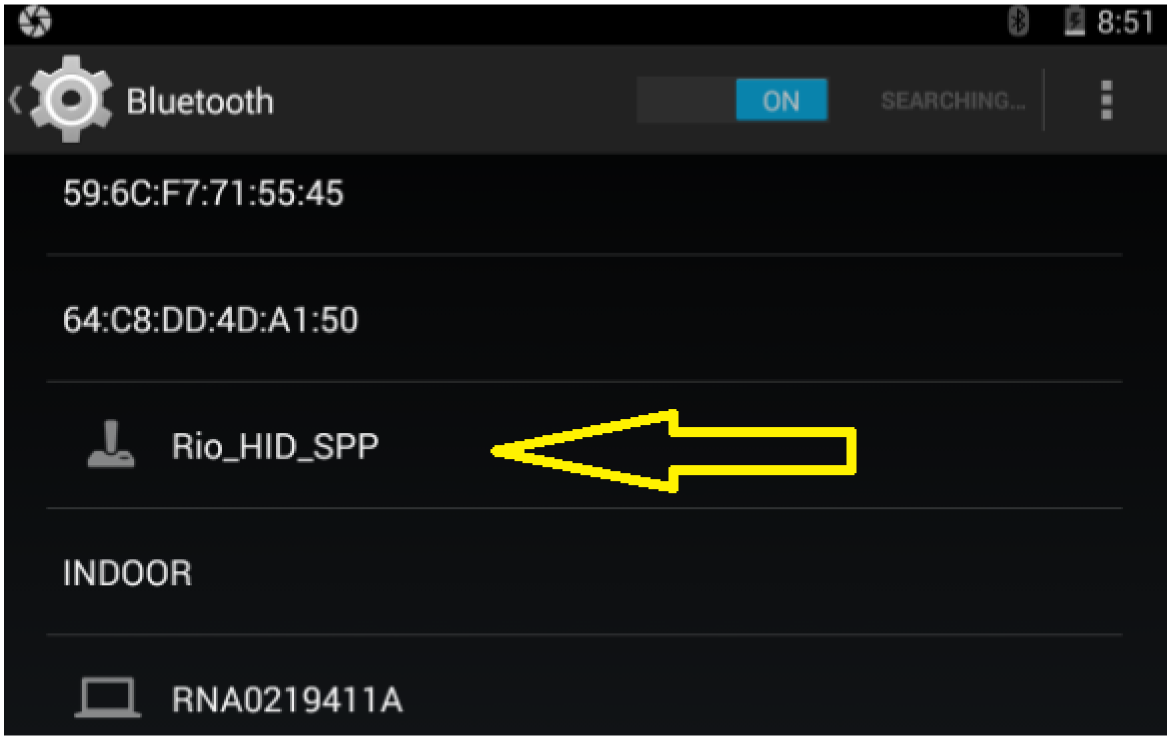


Figure 12. HID/SPP Demo Pairing (2 of 2)

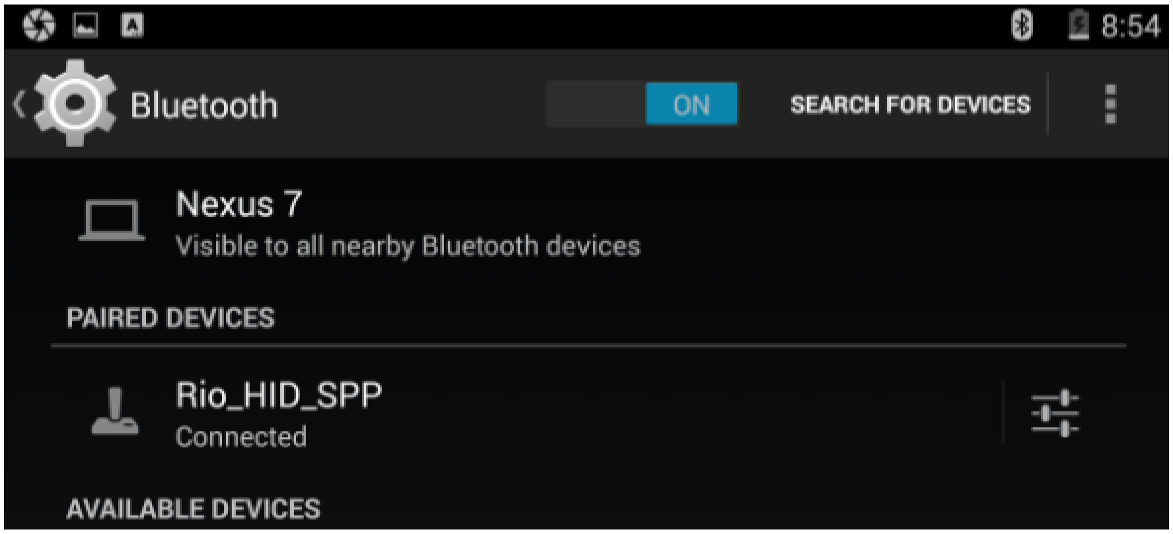
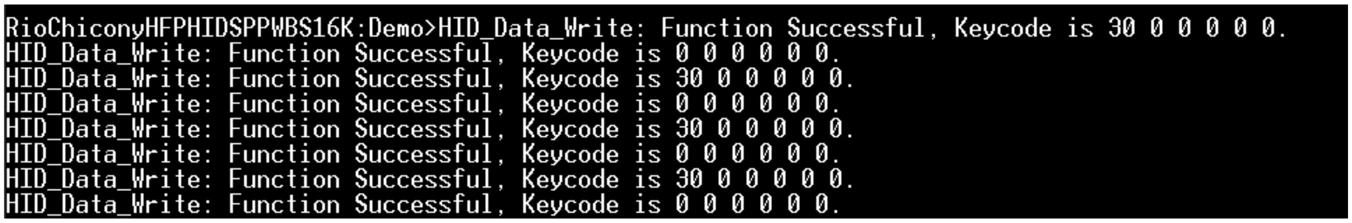


Figure 13. HID/SPP APP Connection

4. Launch any Text Editor tool.



```
RioChiconyHFPHIDSPPWBS16K:Demo>HID_Data_Write: Function Successful, Keycode is 30 0 0 0 0 0 0.
HID_Data_Write: Function Successful, Keycode is 0 0 0 0 0 0.
HID_Data_Write: Function Successful, Keycode is 30 0 0 0 0 0 0.
HID_Data_Write: Function Successful, Keycode is 0 0 0 0 0 0.
HID_Data_Write: Function Successful, Keycode is 30 0 0 0 0 0 0.
HID_Data_Write: Function Successful, Keycode is 0 0 0 0 0 0.
HID_Data_Write: Function Successful, Keycode is 30 0 0 0 0 0 0.
HID_Data_Write: Function Successful, Keycode is 0 0 0 0 0 0.
```

Figure 14. HID/SPP Demo Result

5. Click the Key from the HID demo. In Figure 14, keycode 30 is the '1' shown in Figure 15.

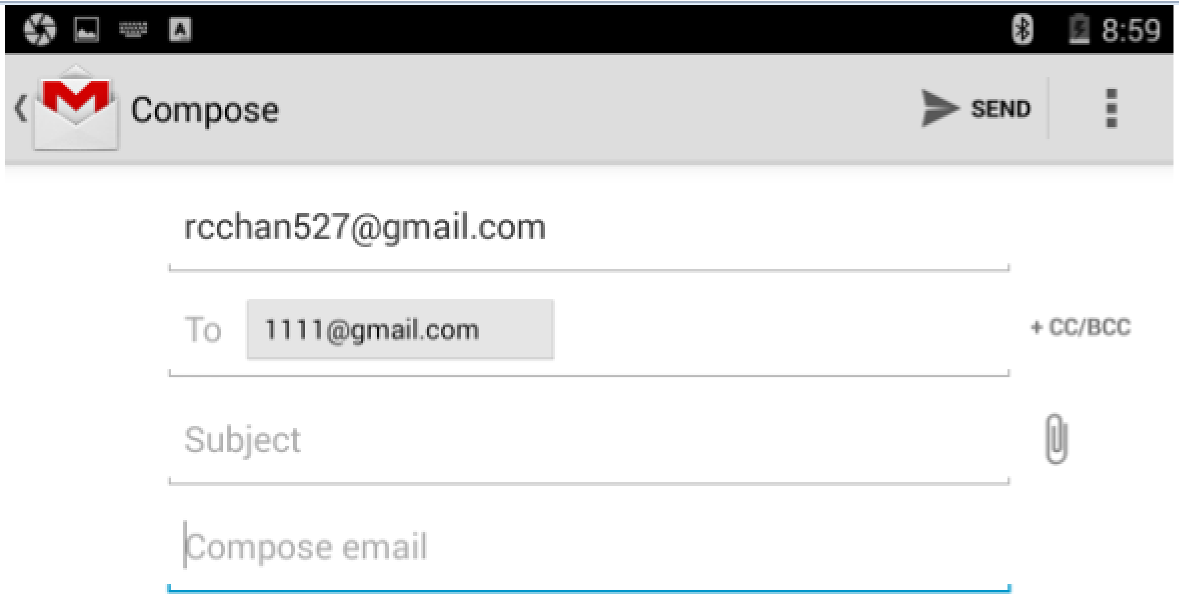


Figure 15. HID/SPP Demo Result

7.2 *How to Demo HFP*

The HFP has two demo cases, shown in Figure 16 and Figure 17.

Figure 16 is the normal HFP BT MO/MT call setup. Figure 17 is the HFP triggering the APPLE SIRI / Google Search to do voice transmission; this application can be applied as voice recognition.

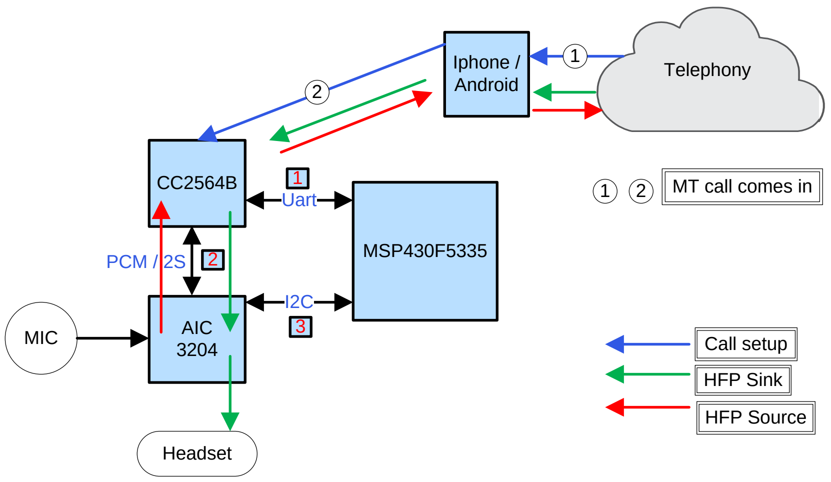


Figure 16. HFP Telephony Voice Flow

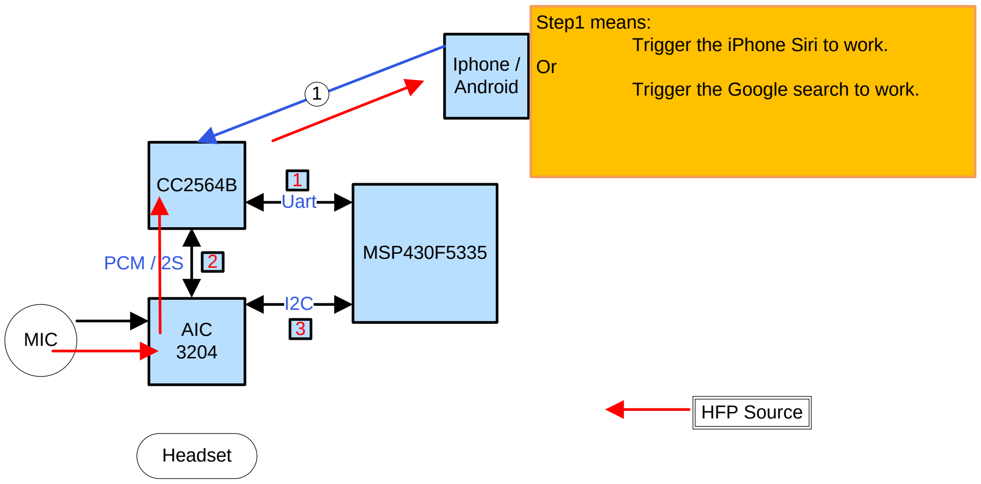


Figure 17. HFP Voice Recognition

Case 1 demo steps:

1. Download the demo code.
2. Turn on the iPhone's BT.
3. Open the console with the MSP430 UART-USB port (9600/N/81).
4. Type this command: OpenHFServer 1. (This asks the kit to send the broadcast to the iPhone). The iPhone then shows the "SS1-WBS-16KHz" (DemoKit name).
5. Connect with the kit from the iPhone.
6. Dial the iPhone's number from the telephony network.
7. On the console, type: Manage Audio 1. It is now ready for use.

Case2 demo steps:

1. Download the demo code.
2. Turn on the iPhone's BT.
3. Open the console with the MSP430 UART-USB port (9600/N/81).
4. Type this command: OpenHFServer 1. (This asks the kit to send the broadcast to the iPhone). The iPhone then shows the "SS1-WBS-16KHz" (DemoKit name).
5. Connect with the kit from the iPhone.
6. Long-press the iPhone Siri to trigger the HFP.
7. Speak. The demo is ready for use.

8 Function APIs

These are the important functions that require a special focus.

HID related:

- [HIDConnectRemoteHost] – This function is used to connect the remote peer.
- [HIDRegisterDeviceServer] – This function registers the HID to the stack, and registers the SDP record.

HFP related:

- [OpenHFServer] – This function is responsible for opening a serial port or server on the local device. This function opens the serial port server on the specified RFCOMM channel.
- [ManageAudioConnection] – This function calls the other sub-functions according this flow: ManageAudioConnection → HFRESetupAudioConnection → HFRE_Setup_Audio_Connection. It then sets up the audio with the stack.

SPP related:

- [OpenSPPServer] – This function is responsible for opening a serial port or server on the local device, or the serial port or server on the specified RFCOMM channel.

Table 1. HID SDP Records

Item	Definition	Type	Value	Status[1]	Attribute ID
Protocol ID	L2CAP	UUID	0x0100 (L2CAP)[2]	M	
Parameter #0	PSM	Uint16	HID Control	M	
Protocol Descriptor #1		Data element sequence		M	
ProtocolID	HID	UUID	0x0011 (HID Protocol)[2]	M	
Service Name	Displayable text name	Data-element/ string	'XYZ Wireless Device'[3]	O	0x0100[4]
Service Description	Displayable text name	Text string	'HID Widget'[3]	O	0x0101[4]
Provider Name	Displayable text name	Text string	'XYZ Company'[3]	O	0x0102[4]
BluetoothProfile DescriptorList				M	0x0009
Profile 0		UUID	0x1124 (HID Profile)[2]	M	
Parameter for Profile 0	Version	uint16	0x0101[5]	M	
AttributeIDList		Data element/ sequence	Define primary (and alternate) language	M	0x0006
HIDParserVersion		uint16	0x0111[6]	M	0x0201
HIDDeviceSubclass		uint8	See Bluetooth Defined Numbers [9]	M	0x0202
HIDCountryCode		uint8	See Section 6.2.1 of the USB HID Specification [5]	M	0x0203
HIDVirtualCable		Bool (8)	TRUE/FALSE	M	0x0204
HIDReconnectInitiate	0	Bool (8)	TRUE/FALSE	M	0x0205
HIDDescriptorList		Sequence		M	0x0206
HIDLANGIDBaseList		Sequence		M	0x0207
HIDBatteryPower		Bool (8)	TRUE/FALSE	O	0x0209
HIDRemoteWake		Bool (8)	TRUE/FALSE	O	0x020A
HIDSupervisionTimeout		uint16		O	0x020C
HIDNormallyConnectable		Bool (8)	TRUE/FALSE	O	0x020D
HIDBootDevice		Boolean	TRUE/FALSE	M	0x020E
HIDSSRHostMaxLatency		uint16		O	0x020F
HIDSSRHostMinTimeout		uint16		O	0x0210

(1) M = Mandatory; O = Optional
(2) For natural language support for all "displayable" text string attributes, an offset must be added to the LanguageBaseAttributeIDList value for the selected language (see the SDP Specification in [6] for details).
(3) Indicating version 1.1.
(4) The Service Name, Service Description, and Provider Name attribute values suggested here are only examples; a service-provider may define more descriptive names and descriptions for their device.
(5) Indicating version 1.11.
(6) Notation: uint = Unsigned Integer, 8 = 8-bit, 16 = 16-bit, Bool = Boolean, Array = array of specified data type.

9 Appendix

9.1 All APIs Referred to

```
//CC2564 common user functions
static int Inquiry(ParameterList_t *TempParam);
static int DisplayInquiryList(ParameterList_t *TempParam);
static int SetDiscoverabilityMode(ParameterList_t *TempParam);
static int SetConnectabilityMode(ParameterList_t *TempParam);
static int SetPairabilityMode(ParameterList_t *TempParam);
static int ChangeSimplePairingParameters(ParameterList_t *TempParam);
static int Pair(ParameterList_t *TempParam);
static int EndPairing(ParameterList_t *TempParam);
static int PINCodeResponse(ParameterList_t *TempParam);
static int PassKeyResponse(ParameterList_t *TempParam);
static int UserConfirmationResponse(ParameterList_t *TempParam);
static int GetLocalAddress(ParameterList_t *TempParam);
static int SetLocalName(ParameterList_t *TempParam);
static int GetLocalName(ParameterList_t *TempParam);
static int SetClassOfDevice(ParameterList_t *TempParam);
static int GetClassOfDevice(ParameterList_t *TempParam);
static int GetRemoteName(ParameterList_t *TempParam);
static int DisplayHelp(ParameterList_t *TempParam);

//CC2564 Common API
static int SetDisc(void);
static int SetConnect(void);
static int SetPairable(void);
static int DeleteLinkKey(BD_ADDR_t BD_ADDR);
static int InitializeHIDDevice(void);
static int OpenStack(HCI_DriverInformation_t *HCI_DriverInformation, BTPS_Initialization_t
*BTPS_Initialization);
static int CloseStack(void);
static unsigned long StringToUnsignedInteger(char *StringInteger);
static char *StringParser(char *String);
static int CommandParser(UserCommand_t *TempCommand, char *UserInput);
static int CommandInterpreter(UserCommand_t *TempCommand);
static int AddCommand(char *CommandName, CommandFunction_t CommandFunction);
static CommandFunction_t FindCommand(char *Command);
static void ClearCommands(void);
static void BD_ADDRToStr(BD_ADDR_t Board_Address, char *BoardStr);
static void DisplayPrompt(void);
static void DisplayFunctionError(char *Function, int Status);
static void UserInterface(void);
static Boolean_t CommandLineInterpreter(char *Command);


//HFP profile common functions
static void FormatEIRData(unsigned int BluetoothStackID);
static int  UpdateEIRData(char *DeviceName);
static int HFRESetupAudioConnection(void);
static int HFREReleaseAudioConnection(void);
static int OpenHFServer(ParameterList_t *TempParam);
static int CloseHFServer(ParameterList_t *TempParam);
static int ClosePort(ParameterList_t *TempParam);
static int ManageAudioConnection(ParameterList_t *TempParam);
static int AnswerIncomingCall(ParameterList_t *TempParam);
static int HangUpCall(ParameterList_t *TempParam);

//HID profile common functions.
static void DisplayHIDDeviceMenu(void);
static void PopulateHIDDeviceCommandTable(void);
static int HIDRegisterDeviceServer(void);
static int HIDConnectRemoteHost(ParameterList_t *TempParam);
static int HIDCloseConnection(ParameterList_t *TempParam);
static int HIDControlRequest(ParameterList_t *TempParam);
static int HIDGetReportResponse(ParameterList_t *TempParam);
static int HIDSetReportResponse(ParameterList_t *TempParam);
static int HIDGetProtocolResponse(ParameterList_t *TempParam);
static int HIDSetProtocolResponse(ParameterList_t *TempParam);
static int HIDGetIdleResponse(ParameterList_t *TempParam);
static int HIDSetIdleResponse(ParameterList_t *TempParam);
static int HIDDataWrite(ParameterList_t *TempParam);
static int HIDDeviceMode(ParameterList_t *TempParam);//Rio

//SPP profile common functions.
static int OpenSPPServer(ParameterList_t *TempParam);
static int CloseSPPServer(ParameterList_t *TempParam);

//Callbacks.
   /* Callback Function Prototypes.                                     */
static void BTPSAPI GAP_Event_Callback(unsigned int BluetoothStackID, GAP_Event_Data_t
*GAPEventData, unsigned long CallbackParameter);
static void BTPSAPI HID_Event_Callback(unsigned int BluetoothStackID, HID_Event_Data_t
*HIDEventData, unsigned long CallbackParameter);
static void BTPSAPI HFRE_Event_Callback(unsigned int BluetoothStackID, HFRE_Event_Data_t
*HFRE_Event_Data, unsigned long CallbackParameter);
static void BTPSAPI SPP_Event_Callback(unsigned int BluetoothStackID, SPP_Event_Data_t
*SPP_Event_Data, unsigned long CallbackParameter);
```

9.2 References

- BT SIG SPP Profile: http://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=8700
- BT SIG HID Profile: https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=246761
- BT SIG HFP Profile: https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=292287