

Capturing Bluetooth Host Controller Interface (HCI) Logs

Hari Nagalla

ABSTRACT

Identifying and debugging Bluetooth[®] issues can be challenging, especially in two-processor architecture with functionality spread across the host processor and the Bluetooth controller. Bluetooth specification allows the separation of the Bluetooth controller and hosts by partitioning the radio and physical/link handling to the controllers, and to the higher level profile, logical link, and connection/pairings to the host resident Bluetooth stack. This delineation allows flexible dual processor Bluetooth system architectures. The interface between the host and the controller is specified as host controller interface (HCI) in Bluetooth specification, and allows interoperability between various vendor host Bluetooth stacks and Bluetooth controllers.

TI's CC256x and WL18xx class of dual-mode Bluetooth controllers is in this two-processor category. TI provides a royalty-free Bluetooth host stack (Bluetopia) for MCU and Linux host environments. As these controllers conform to the HCI specification, they can also be integrated with other third party Bluetooth host stacks, such as Bluez, Bluedroid, Bluekitchen, and so forth.

This application note discusses various options to log or snoop HCI packets between the host and the controller to identify issues and failures. It considers both MCU and Linux/application processor host environments, discusses the available options for HCI logging, and proposes a new approach for resource-constrained MCU environments with Cortex®-M cores.

Contents

1	Introduction	3
2	Host Controller Interface	4
3	BTSnoop Logs in a Linux/Android Environment	6
4	HCI Logging in the MCU Host Environment	10
5	BTSnoop File Format	17
6	References	19

List of Figures

1	Logging Options in a Bluetooth System	3
2	4-Wire HCI/UART Interface (H4)	4
3	3-Wire HCI/UART Interface (H5)	4
4	HCI Protocol	4
5	Bluetooth® Stack Layers	5
6	HCI Logging in BluetopiaPM Environment	7
7	hcidump Tool Usage in a Linux Environment	8
8	btmon Tool Usage in a Linux Environment	9
9	Enable Blueooth HCI Snoop Log	9
10	Frontline Technologies HCI Sniffer (Using Serial Sniffer)	11
11	MSP432 Launchpad With CC2564xQFN-EM	11
12	HCI TX/RX Sniff With CC2564xQFN-EM Module	12
13	PC-Based HCIsniffer Tool	12
14	HCIsniffer Command Line Parameters	13
15	BTSnoop File (Created With HCIsniffer Tool) Viewed With Frontline GUI	13



16	Cortex-M With CoreSight Technology Components	14
17	SWO/ITM Stream Capture and Processing	15
18	SWOsniffer Program Run	17
19	Frontline HCI Sniffer	18
20	Wireshark	18
21	Ellisys	19

List of Tables

1	Acronyms	3
2	HCI Protocol	5

Trademarks

Cortex is a registered trademark of ARM Limited.

Bluetooth is a registered trademark of Bluetooth SIG.

Android is a trademark of Google, LLC.

Linux is a registered trademark of Linus Torvalds.

All other trademarks are the property of their respective owners.



1 Introduction

Debugging Bluetooth protocol-related issues is challenging without proper event and data logging between the host, controller, and peer devices. Bluetooth air sniffers (such as Ellisys or Frontline) provide a comprehensive overview of radio, protocol, and profile-level message and data exchange between the peer Bluetooth devices. These sniffer tools work by "spying" on the communication between the master and slave or central and peripheral devices, and they can identify both radio and profile-level issues. However, these tools can be very expensive, and a piece for each Bluetooth developer is not always feasible.

However, HCI message and event logging and BT firmware (at the controller) logs can also help identify many of the Bluetooth issues, and can supplement the air sniffer logs.

In this application note, the focus is to illustrate HCI logging at the host Bluetooth stack, UART driver level, or externally tapping the HCI/UART Tx/Rx lines to facilitate application and stack debugging.



Figure 1. Logging Options in a Bluetooth System

1.1 Terminology

Table 1. Acronyms

Acronym	Description
CCS	Code Composer Studio
SWD	Serial Wire Debug
SWO	Serial Wire Output
JTAG	Joint Test Action Group/ boundary scan. Technique used for testing and debugging PCBs, SoCs etc
HCI	Host Controller Interface
ITM	Instrumentation Trace Macrocell
DWT	Data Watch Point Trace
UART	Universal Async Receiver, Transmitter
H4	4-wire HCI protocol
H5	3-wire HCI protocol
SWV	Serial WireViewer

Introduction



2 Host Controller Interface

The HCI interface defines a physical and protocol connection between a Bluetooth Host (such as the MCU, application processor, and so forth) and a controller (such as the TICC256x/WL18xx). The Bluetooth specification defines UART, SDIO, and USB-based physical transports for the host interface. The TICC256x and WL18xx Bluetooth controllers only support the UART HCI interface. The UART HCI interface is further divided into two categories – a) 4-wire UART with RTS/CTS hardware flow control, or H4 protocol; and b) 3-wire UART, with no hardware flow control, or H5.

The UART, H4 protocol assumes the UART lines are free from errors and any line errors mandate a controller reset. On the other UART, H5 protocol is tolerant to bit errors and uses SLIP protocol to transmit packets and to deal with packet and bit losses by re-transmission. This adds extra complexity and reduces the effective HCI bandwidth. H4 is the predominantly-used protocol in the industry, and is the recommended protocol for the CC256x and WL18xx HCI interfaces.



Copyright © 2016, Texas Instruments Incorporated

Figure 2. 4-Wire HCI/UART Interface (H4)



Copyright © 2016, Texas Instruments Incorporated

Figure 3. 3-Wire HCI/UART Interface (H5)

2.1 HCI Protocol

4

HCI protocol defines how commands, events, and asynchronous and synchronous data packets are exchanged. Asynchronous packets (ACL) are used for data transfer, while synchronous packets (SCO) are used for voice with handset and hands-free profiles.







Table 2. HCI Protocol

HCI Packet Type	HCI Packet Indicator
HCI Command Packet	0x01
HCI ACL Data Packet	0x02
HCI Synchronous Data Packet	0x03
HCI Event Packet	0x04

Bluetooth stack is loosely based around the OSI model. In two-processor Bluetooth system architecture, the HCI layer is the hardware interface along with the HCI protocol between the host and controller as described earlier. Figure 5 shows the various layers of the host and controller stacks. L2CAP is placed above the HCI, and works across ACL connections. It facilitates the use of Bluetooth links by higher layers.



Figure 5. Bluetooth® Stack Layers



BTSnoop Logs in a Linux/Android Environment

3 BTSnoop Logs in a Linux/Android Environment

Application processors with Linux[®] or Android[™] operating systems, provide HCI logging at the stack level before the HCI packets are transferred to the UART driver. The following sections provide details for enabling BT snoop logs in Bluetopia, BlueZ, and Bluedroid (Android) stacks.

3.1 Logging With Bluetopia Stack

TI's Bluetopia stack provides logging features to capture HCI message and event transfers across the host and controller. The logs can be stored in BTsnoop format.

The following DEVM API can be used to enable logging from applications.

```
int BTPSAPI DEVM EnableBluetoothDebug
(
    Boolean_t
Enable,
       unsigned int
                      DebugType,
       unsigned long DebugFlags,
       unsigned int
                      DebugParameterLength,
       unsigned char * DebugParameter
    )
Parameters
Enable
       Boolean value (TRUE/FALSE) to enable or disable output debugging.
DebugType Indicates the type of debuging. May be one of the following:
DEVM_BLUETOOTH_DEBUG_TYPE_ASCII_LOG_FILE DEVM_BLUETOOTH_DEBUG_TYPE_TERMINAL
DEVM_BLUETOOTH_DEBUG_TYPE_FTS_LOG_FILE
           Optional flag for debuging. Currently the only optional value is the following:
DebuqFlaqs
DEVM_BLUETOOTH_DEBUG_FLAGS_APPEND_FILE
DebugParameterLength Length (in bytes) of the optional DebugParameter.
DebugParameter Optional name for FTS or ASCII debug log file. Note that if ASCII or FTS Log
file is specified than a valid filename must be specified (the preceding parameter and this
parameter).
Returns
This function returns zero if successful, or a negative return error code if there was an error.
```

Typically, applications in a BluetopiaPM environment may use a background process to handle devicelevel control and maintenance. When using TI BlutopiaPM stack demos, the user can open a new terminal and enable logging. The DEVM demo (LinuxDEVM.c) has a console command to enable and disable HCI logging at runtime.

DEVM>EnableBluetoothDebug

```
Usage: EnableBluetoothDebug [Enable (0/1)] [Type (1 - ASCII File, 2 - Terminal, 3 - FTS File)] [Debug Flags] [Debug Parameter String (no spaces)].
Function Error.
DEVM>EnableBluetoothDebug 1 3 0 DebugLog.log
```



Texas

RUMENTS

Terminal		🤶 🖬 📼 🕸 9:13 AM 雄
	🕼 🖨 🐵 hari@hari-Latitude-E6430: /media/hari/4F4544800EC2EBC21/AM335xBluetopiaLinuxProd	Ari@hari-Latitude-E6430: /media/hari/4F4544800EC2EBC21/AM335xBluetopiaLinuxProduct
Q	Usage: EnableBluetoothDebug [Enable (0/1)] [Type (1 - ASCII File, 2 - Terminal, 3 - FTS File)] [Debug Flags] [Debug Parameter String (no spaces)].	SPPM>StartDeviceDiscovery 100
	Function Error.	Attempting to Start Discovery (100 Seconds).
	DEVM>42 1 3 0 DebugLog.log	DEVM_startDeviceDiscovery() Success: 0. SPPM>
	DEVM_EnableBluetoothDebug(TRUE) Success. DEVM>	Local Device Properties Changed. ^S SPPM>
	Local Device Properties Changed. DEVM>	Device Discovery Started. SPPM>
	Device Discovery Started.	Remote Device Found.
	Remote Device Found.	DD_ADDR: 54759ACA001A COD: 0x0C0102
	BD ADDR: 34F39ACA601A	Device Name: LTA0868345B
	COD: 0x0C0102	Device Flags: 0x80000601
	Device Name: LTA0868345B	RSSI: -62
	Device Flags: 0x80000601	Friendly Name:
		App. Into: : 00000000 Daired state : FALSE
A .	EIR_UAIA_NIUMI RSST: -62	Connect State: FALSE
	Trans. Power: 7	Encryot State: FALSE
	Friendly Name:	Sniff State : FALSE
<u>a</u> ,	App. Info: : 00000000	Serv. Known : FALSE
	Paired State : FALSE	SPPM>17
100	Connect State: FALSE	
	Encrypt State: FALSE	DEVM_StopDeviceDiscovery() Success: 0.
	Sniff State : FALSE	SPPM>
	SETV. KNOWN : FALSE	conversions
4 🥢 🛛	Local Device Properties (bapad	SPMP
A	DEVits	SPPM-25
	Device Discovery Stopped.	
	DEVM>	Usage: QueryRemoteDeviceServices [BD ADDR] [Force Update] [Bytes to Query (specifie
	Local Device Properties Changed.	d if Force is 0)].
	DEVM>	Function Error.
₽ <u></u> (Device Discovery Started. DEVM>	SPPM>StartDeviceDiscovery 34F39ACA601A
	Remote Device Found.	Attempting to Start Discovery (34 Seconds).
	BD_ADDR: 34F39ACA601A	DEVM_StartDeviceDiscovery() Success: 0.
		SPMP
	Device Name: LIAOSOS345D	Codat Device Properties changed.
32	NAME KNOWN	Device Discovery Started.
1	EIR DATA KNOWN	SPPM>
	RSSI: -56	Remote Device Found.
20	Trans. Power: 7	BD_ADDR: 34F39ACA601A
	Friendly Name:	COD: 0x0C0102
	App. Info: : 00000000	Device Name: LTA0868345B
	Paired State : FALSE	Device Flags: 0x80000601
(Angel 1)		

Figure 6. HCI Logging in BluetopiaPM Environment

3.2 Logging With Linux BlueZ Stack

BlueZ supports two tools for capturing HCI traffic. 'hcidump' is an older tool and is being deprecated by the 'btmon' tool. Both tools support the BTsnoop file format, and the captured files can be opened in Wireshark or Frontline HCI viewer.

```
#hcidump -B -w <BTsnoop file name>
```

Or

#hcidump -Xt -w <BTsnoop file name>

TEXAS INSTRUMENTS

BTSnoop Logs in a Linux/Android Environment

www.ti.com

😣 🗇 💿 hari@hari-Latitude-E6430: /media/hari/4F4544800EC2EBC21/AM335xBluetopiaLinuxProductio
root@am335x-evm:~# root@am335x-evm:~# hcidump -Xt -w hci-snoop.log HCI sniffer - Bluetooth packet analyzer ver 5.46 btsnoop version: 1 datalink type: 1002 device: hci0 snap_len: 1500 filter: 0x0 ^C
root@am335x-evm:~# btmon -h btmon - Bluetooth monitor Usage:
options:
<pre>-r,read <file> Read traces in btsnoop format -w,write <file> Save traces in btsnoop format -a,analyze <file> Analyze traces in btsnoop format -s,server <socket> Start monitor server socket -p,priority <level> Show only priority or lower -i,index <num> Show only specified controller -d,tty <tty> Read data from TTY -B,tty-speed <rate> Set TTY speed (default 115200) -t,time Show time instead of time offset -T,date Show time and date information -S,sco Dump SCO traffic -A,a2dp Dump A2DP stream traffic -E,ellisys [ip] Send Ellisys HCI Injection -h,help Show help options</rate></tty></num></level></socket></file></file></file></pre>
root@am335x-evm:~# btmon -w hci-fte-snoop.log
[496.280194] NET: Registered protocol family 38 = Note: Linux version 4.14.40-g4796173fc5 (armv7l) 0.827034
= Note: Bidetooth Subsystem Version 2.22 0.827046 = New Index: 54:4A:16:13:1C:21 (Primary,UART,hci0)
= Open Index: 54:4A:16:13:1C:21 [hci0] 0.827054
= Index Info: 54:4A:16:13:1C:21 (Texas Instruments Inc.) [hci0] 0.827057 @ MGMT Open: bluetootbd (privileged) version 1 14
<pre>@ MGMT Open: btmetotild (privileged) version 1.14 @ MGMT Open: btmon (privileged) version 1.14</pre>
{0x0002} 0.827147 @ RAW Open: hcitool (privileged) version 2.22

Figure 7. hcidump Tool Usage in a Linux Environment

btmon -w <BTsnoop file name>

TEXAS INSTRUMENTS

www.ti.com

BTSnoop Logs in a Linux/Android Environment

₽ 192.168.2.22 - PuTTY	_		\times
reorgam335x-evm:~# btmon -w hci-fte-snoop.log			^
= Note: Linux Version 1.11.10 g17561751c5 (armv71)		0.48361	14
= Note: Bluetooth subsystem version 2.22		0.48362	28
= New Index: 54:4A:16:13:1C:21 (Primary,UART, hci0)	[hci0]	0.48363	35
= Open Index: 54:4A:16:13:1C:21	[hci0]	0.48363	39
= Index Info: 54:4A:16:13:1C:21 (Texas Instruments Inc.)	[hci0]	0.48364	12
@ MGMT Open: bluetoothd (privileged) version 1.14	{0x0001}	0.48364	18
<pre>@ MGMT Open: btmon (privileged) version 1.14</pre>	{0x0002}	0.48375	56
@ RAW Open: hcitool (privileged) version 2.22	{0x0003}	73.22304	18
@ RAW Close: hcitool	{0x0003}	73.22450	01
@ RAW Open: hcitool (privileged) version 2.22	{0x0003}	73.22505	59
@ RAW Close: hcitool	{0x0003}	73.22541	19
@ RAW Open: hcitool (privileged) version 2.22	{0x0003}	73.22601	74
< HCI Command: Inquiry (0x01 0x0001) plen 5	#1 [hci0]	73.22644	46
Access code: 0x9e8b33 (General Inquiry)			
Length: 10.24s (0x08)			
Num responses: 0			
> HCI Event: Command Status (0x0f) plen 4	#2 [hci0]	73.23363	34
Inquiry (0x01 0x0001) ncmd 1			
Status: Success (0x00)			
> HCI Event: Extended Inquiry Result (0x2f) plen 255	#3 [hci0]	74.27662	21
Num responses: 1			
Address: 4C:EB:42:6E:E5:4B (Intel Corporate)			
Page scan repetition mode: Rl (0x01)			
Page period mode: P2 (0x02)			
Class: 0x0a010c			
Major class: Computer (desktop, notebook, PDA, organizers)			
Minor class: Laptop			
Networking (LAN, Ad hoc)			
Capturing (Scanner, Microphone)			
Clock offset: 0x5e5a			
RSSI: -42 dBm (0xd6)			
Name (complete): RAJANI-PC			
TX power: 4 dBm			
16-bit Service UUIDs (complete): 4 entries			
A/V Remote Control Target (0x110c)			
Audio Source (0x110a)			\sim

Figure 8. btmon Tool Usage in a Linux Environment

3.3 Logging With Bluedroid/Android



Figure 9. Enable Blueooth HCI Snoop Log

Capturing Bluetooth Host Controller Interface (HCI) Logs



HCI Logging in the MCU Host Environment

www.ti.com

There are two options for retrieving the HCI log from the Android device.

- Attach the Android device to the computer. The /sdcard/btsnoop_hci.log file is in the root of one of the mountable drives. Copy the file to the C:/Users/Public/Public Documents/Frontline Test Equipement/My Capture File/ directory.
- Use the Android Debug Bridge (ADB) with the following steps. The debug bridge is included with the Android Software Developer Kit.
 - 1. On the Android device Development screen, select Android debugging or USB debugging.
 - 2. Connect the computer and Android device with a USB cable.
 - 3. Open a terminal on the computer and run the following command. adb devices
 - 4. The Android device should appear in this list confirming that ADB is working. List of devices attached
 - 5. In the terminal, enter the following command to copy the HCI Log to the computer. adb pull /sdcard/btsnoop_hci.log

4 HCI Logging in the MCU Host Environment

Because of a lack of a file system and limited CPU processing power on the MCU, it is untenable to do stack-level HCI logging in an MCU environment. This leaves two options for logging:

- External HCI/UART line sniffing with either Ellisys, Frontline, or custom probes
- Using data tracers with DWT/ITM on Cortex-M based MCUs

4.1 Tapping UART Lines

When the HCI/UART TX and RX lines between the host and controller are accessible for tapping, use the HCI sniffer to capture the data.

4.1.1 FTE and Ellisys HCI Sniffer

Commercial HCI sniffers use serial sniffers (UART) to externally capture the HCI traffic and process it at runtime, or offline to present the message and data exchanges in a user-friendly GUI.





Figure 10. Frontline Technologies HCI Sniffer (Using Serial Sniffer)

4.1.2 Custom HCI Sniffer

Custom probes can sniff HCI/UART TX and RX lines, and post process into a BTSnoop file format. The reformatted BTSnoop data/log can be viewed with Wireshark, FTE, or the Ellisys HCI viewer GUI to analyze.

This is one example of a custom procedure to sniff HCI Tx/Rx lines on the MSP432 with CC2564C.

1. Identify the HCI Tx/Rx pins for sniffing.



Figure 11. MSP432 Launchpad With CC2564xQFN-EM



2. Use a suitable level shifter along with the UART to USB adapter to read the ports on a PC or workstation.



Figure 12. HCI TX/RX Sniff With CC2564xQFN-EM Module

3. Use the HCIsniffer tool on the PC. This tool is developed in Java, and assumes a Java runtime environment on the PC/workstation. It takes port numbers and HCI baud rate as parameters. The tool outputs a BTSnoop format file, which can be opened with Wireshark or FTE HCI viewer GUIs.



Figure 13. PC-Based HCIsniffer Tool

Figure 14 and Figure 15 show the execution of the tool and the resulting BTSnoop log, viewed with the Frontline HCI viewer.

Command Prompt		- 🗆 ×	
			^
C:\Users\hari\testojava -jar hcisniffer.jar COM15	COM16 2000000 hci-debug.log		
Com thread started on COM15			
Com thread started on COM16	External HCI/UART Ports: COM15, COM16		
Parser thread started !!	Baudrate: 2Mbps		
Parser thread started !!			
HCI : EVent code: UXUF			
HCI : CMD opcode: 0x0401			
HCI : Event code: 0x02			
HCI : Event code: 0x02			
HCI : Event code: 0x01			
HCI : EVENT CODE. 0X0E			
HCI : Event code: 0x0CSS			
HCI : CMD opcodo: 0x040E			
HCI : Event code: $0x0405$			
HCI : Event code: 0x02			
HCI : CMD opcode: 0x055			
HCI : Event code: 0x0E			
HCI : Event code: 0x1B			
HCT: ACL Datal ength=12			
HCI : Event code: 0x13			
HCI: ACLDataLength=16			
HCI : Event code: 0x0E			
HCI : CMD opcode: 0x0C37			
HCI : Event code: 0x13			
HCI: ACLDataLength=18			
HCI: ACLDataLength=14			
HCI: ACLDataLength=8			
HCI : Event code: 0x13			
			v



∐ Message File Edit	Sequence Chart (MSC) View Help	-	٥	×
P , P	R R 🟦 🛍			
All Layers	Ctrl Summary No	on-Mag Summary HCI L2CAP RFCOMM		
Frame#	Time	RFCOMM_M L2CAP_M HCI_H HCI_C L2CAP_S RFCOMM_S		^
43	20:37:57.455055	(Tran. ID=1, PSM=RFCOMM, SCID=0x0080)		
44	20:37:57.469024	HCL_Event [Max Slots Change] (Handle: 0x0001)		
45	20:37:57.470987	HCI_Event (Number Of Completed Packets) (Handle: 0x0001, Num Of Completed Pkts: 1, Available Host's ACL credits: 0)		
46	20:37:57.484842	ACL Data (Handle: 0x0001, PBF: First L2CAP Packet) Connection pending (Tran. ID=1, DCID=0x0040, SCID=0x0040, Result=Connection pending)		
47	20:37:57.484900	L2CAP_Configure Request (Tran. ID-2, DCID=0x0040) ACL Data		
48	20:37:57.487158	ACL Data [Handle: 0x0001, PBF: First L2CAP Packet] RFCOMM connection created (Tran. ID=1, DCID=0x0040, SCID=6x0080, Result=Connection successful)		
48	20:37:57.487158	L2CAP connection(s) that are alive: RFCOMM		
49	20:37:57.492493	HCI_Write_Link_Supervision_Timeout (Handle: 0x0001)		v

Figure 15. BTSnoop File (Created With HCIsniffer Tool) Viewed With Frontline GUI



NOTE: H4 protocol assumes error-free UART lines, and there are no synchronization patterns in the HCI byte stream. This poses challenges to HCI serial sniffers to properly sync on the HCI byte stream. TI recommends starting the serial HCI sniffer from a known anchor point, such as start hci sniffer, when the host and controller are in sleep or in low activity state. Or, start the hcisniffer at the beginning, with the board unpowered. Though the hcisniffer has detectors to detect when it is out of sync with the HCI stream and attempts to resync with the stream, it is advised to maintain one HCI baudrate (115200) from the time the board is started. This facilitates easy synchronization and is suited when debugging startup, pairing, and connection-related issues.

4.2 Cortex-M Data Tracers

Cortex-M3/4's CoreSight technology allows data tracing. The user can exploit this feature to trace data read and writes to the UART RX and TX registers on the MCU/SoC. The Cortex-M ITM/DWT tracing incurs no CPU penalties, and is an effective way to capture data at runtime from the core. Typically, Cortex-M based MCUs provide SWD/SWO output, as the pins needed to bring out debug and tracing capabilities are much less compared to JTAG and TPIU. Figure 16 shows the main components of the CoreSight technology components.



Figure 16. Cortex-M With CoreSight Technology Components

The MSP432 uses Cortex-M4F with ITM and DWT CoreSight technology components. It does not have ETM, so instruction trace is not possible. However, using DWT the user can perform PC sampling and data trace. The MSP432 LP comes with an XDS-110 emulator and is preconfigured for SWD and SWO output, with SDO routed to the auxiliary port. The emulation package with XDS-110 must be above "7.0.89.0".





Figure 17. SWO/ITM Stream Capture and Processing

The following code snippets illustrate the configuration of ITM/DWT for data tracing on the MSP432. Similar configurations would be required on any MCU host with Cortex-M3/M4 CoreSight technology.

```
/** Initialize ITM module **/
void ITM_initModule(const ITM_config itm_config)
{
    // Disable module
    SCS_DEMCR &= (~SCS_DEMCR_TRCEN);
    ITM_TCR = 0x0000000;
    // Enable trace
    SCS_DEMCR | = SCS_DEMCR_TRCEN;
    // Unlock and Setup TPIU for SWO UART mode
    TPIU_LAR = CS_LAR_UNLOCK;
    TPIU_SPPR = TPIU_SPPR_SWO_UART;
    TPIU_CSPSR = TPIU_CSPSR_PIN_1;
    \ensuremath{{//}} Unlock and enable all ITM stimulus ports with default settings
    ITM_LAR
             = CS_LAR_UNLOCK;
    ITM_TER
              = ITM_TER_ENABLE_ALL;
    ITM_TPR
             = ITM_TPR_ENABLE_USER_ALL;
    // Setup Baud rate
    if (itm_config.systemClock)
    {
        uint32_t prescalar = itm_config.systemClock / itm_config.baudRate;
        // Offset with current prescalar value
        uint32_t diff1 = itm_config.systemClock - (prescalar * itm_config.baudRate);
        // Offset with prescalar+1 value
        uint32_t diff2 = ((prescalar+1) * itm_config.baudRate) - itm_config.systemClock;
        if (diff2 < diff1) prescalar++;</pre>
        // Program prescalar value as (prescalar factor - 1)
        TPIU_ACPR = (prescalar - 1);
    }
    // Disable formatter
    TPIU_FFCR = 0;
    // Unlock DWT
    DWT_LAR
             = CS_LAR_UNLOCK;
}
/** Enable ITM */
void ITM_enableModule()
{
    // Enable ITM module
    ITM_TCR | = ITM_TCR_ENABLE_ITM;
}
```



```
HCI Logging in the MCU Host Environment
```

```
/** Enable Data trace **/
bool ITM_enableDataTrace(const uint32_t *variable, uint8_t index)
{
    uint_least8_t numDwtComp = (DWT_CTRL & DWT_CTRL_MASK_NUM_COMP) >>
DWT CTRL SHIFT NUM COMP;
   uint_least8_t dwtIndex = 0;
    bool dwtAvailable = false;
    DWT_COMP(index) = (uint32_t)variable;
    DWT_MASK(index) = 0x0;
    DWT_FUNC(index) = ( DWT_FUNC_ENABLE_ADDR_OFFSET | DWT_FUNC_ENABLE_COMP_RW);
    dwtAvailable = true;
    ITM_TCR | = ITM_TCR_ENABLE_DWT_TX;
    return dwtAvailable;
}
/* Enable ITM timing */
void ITM_enableTiming(ITM_tsPrescale tsPrescale)
{
    // Set timestamp prescalar enable timestamp generation
    ITM_TCR |= ((tsPrescale << ITM_TCR_TS_PRESCALE_SHIFT) & ITM_TCR_TS_PRESCALE_MASK);</pre>
    ITM_TCR
            = (ITM_TCR_ENABLE_TS);
}
      /* Enable Sync packets */
void ITM_enableSyncPackets(ITM_syncPacketRate syncPacketRate)
   {
    // Clear sync packet rate
    DWT_CTRL &= ~(DWT_CTRL_MASK_SYNCTAP);
    // Set sync packet rate
    DWT_CTRL |= ((syncPacketRate << DWT_CTRL_SHIFT_SYNCTAP) & DWT_CTRL_MASK_SYNCTAP);
    // Enable sync packet generation
    DWT_CTRL | = DWT_CTRL_ENABLE_CYC_CNT;
    ITM_TCR |= (ITM_TCR_ENABLE_SYNC);
  }
/* Insert the below code snippet to Initialize ITM and enable data trace ^{\prime /}
ITM_config itm_config = { 48000000, ITM_12000000 };
    /* Setup ITM */
    {
                     *hci_rx_ptr = (uint32_t*) 0x4000180C; // MSP432 UART Rx register
        uint32 t
        uint32_t
                     *hci_tx_ptr = (uint32_t*) 0x4000180E; // MSP432 UART Tx register
        ITM_initModule(itm_config);
        ITM_enableDataTrace(hci_rx_ptr,0);
        ITM_enableDataTrace(hci_tx_ptr,1);
        ITM_enableModule();
      }
```

NOTE: The selected SWO baud rate must be supported by the MCU. On the MSP432E4, speeds up to 15 MHz are supported. Thus, in the above example code snippets SWO is configured for 12 MHz, with the MSP432 running at 48 MHz. If too much data is generated, ITM may insert overflow packets (0x70), thus the parser must be able to handle them and resync on the next synchronization packet.

16 Capturing Bluetooth Host Controller Interface (HCI) Logs

Command Prompt	_	×
C:\Users\hari\tesava -jar swosniffer.jar COM10 12000000 swo-hci-sniff2.log		^
Com thread started on COMID		
SWO Parser thread started !!		
SWO: Sync Detected !! SWO baudrate:12Mbps		
Parser thread started !! SWO Port: Through XDS-110 on MSP432LP		
Parser thread started !!		
SWO: Sync Detected !!		
HCI : CMD opcode: 0x0401		
HCI : Event code: 0x0F		
SWO: Sync Detected !!		
HCI : Event code: 0x02		
HCI : Event code: 0x02		
SWO: Sync Detected !!		
HCI : Event code: 0x01		
SWO: Sync Detected !!		
SWO: Sync Detected !!		
HCI : CMD opcode: 0x0C55		
HCI : Event code: 0x0E		
HCI : CMD opcode: 0x0405		
HCI : Event code: 0x0F		
HCI : Event code: 0x12		
HCI : Event code: 0x03		
HCI : CMD opcode: 0x0C55		
HCI : Event code: 0x0E		
HCI : EVent code: 0XIB		
HCI: EVent code: 0X13		
HCI: ACLUATALENGTN=12		
HCI: ACLUATALENGTH=10		
HCI. ACLDatalength 15		
HCI: ACLUAIALENGIN=10		
HCL : ACDARAtempth=10		
her . Event Code. Oxoc		\checkmark

Figure 18. SWOsniffer Program Run

5 BTSnoop File Format

The BTSnoop file format is suitable for storing Bluetooth HCI traffic. It closely resembles the snoop format, as documented in RFC 1761. Wireshark, FTE, and Ellisys BT analyzers accept BT snoop formatted files to display the message, event, and data sequences between the BT host and controllers.

Refer to the following link from Frontline Technologies for details on BTsnoop file format: http://www.fte.com/webhelp/bpa600/Content/Technical_Information/BT_Snoop_File_Format.htm.

5.1 HCI Viewers

Wireshark is a free and open source protocol analyzer. Both Frontline and Ellisys also provide free HCI viewers that can be used with BTsnoop files.

5.1.1 Frontline HCI Sniffer

Download Frontline HCI sniffer here



BTSnoop File Format

ComProbe Protocol Analysis System - hci-debug						H Message Sequence Chart (MSC)								
File View Options Window Help						File Edit View Help								
2 / P I R 2 3 W 2 8						♀ ♀ ♀ 								
						All Layer	All Layers Ctrl Summary Non-Msg Summary HCI L2CAP RFCOMM							
Capture file: C:\Users\hari\test\hci-debug.log							-	-	DECO					
arrame Uispiay - nci-debug							rrame#	Time	RFCU					
ile colit view rormat ritter bookmarks Uptions Window Help						34	01:30:24.064172		(Handle:	0×0001, Num Of Compl	HCI_Event (Number Of Completed Packets) eted Pkts: 1, Available Host's ACL credits: -1)			
Frame 35: (Controller) Len=13 }- HCI UART: HCI Packet Type: ACL Data Packet g HCI:	Unfiltered											ACL Data Handle: 0x0001, PBF: First L2CAP Packet)		
Packet from Corroller Handle 0001 Hondoxtar Flog No Boostaat, port-top Packet Boundy Flog. First automaticals Total Length 8 ZCAP Roke Master Addes: 1 POU Length 4 Down 10: 00000 (FRCDMM) FRCDMM POUTOND (FRCDMM) FOLOM POUTOND (FRCDMM) POUTON	B Frame# 26 28 29 31 32	Role Slave Slave Master Slave	Addr. [1 0 1 1 1 1 1 1	DLCI 0x00 0x00 0x00 0x00 0x02 0x02	Channel 0 0 1	Frame T; SABM UIH UA SABM	^{Ty} 35	01:30:24.065829					L2CAP_E (Length=	
	33 35 36 38 39 40	Master Master Slave Slave Master Master	1 1 1 1	0x00 0x02 0x00 0x02 0x00 0x00	0 1 0 1 0 0 0 0 0	UIH UA UIH UIH UIH UIH	36	01:30:24.073054	(Channel=Signalir	[L 4UIH 9, Length=4, DV=¥a	L2CAP_Data ength=8, CID=0x0040) d→ Modem status Response	ACL Data (Handle: 0x0001, PBF: First L2CAP Packet, A	vailable H	
	\$ 02 01 20 08 00 04 00 80 00 0b 73 01 92					-	37	01:30:24.073006		(Handle:	0×0001, Num Of Compl	HCI_Event (Number Of Completed Packets) eted Pkts: 1, Available Host's ACL credits: -1)		
	.× ₽ ▲ N E						38	01:30:24.077422		(L UIH	L2CAP_Data ength=5, CID=0x0040)	ACL Data (Handle: 0x0001, PBF: First L2CAP Packet, A	vailable H	
>									٢					
al Frames: 52 Frames Filtered In: 11 Fra	me #s Selected: 35;	(1 total)											100%	
								No. of Concession, Name				5-43 D	100%	

Figure 19. Frontline HCI Sniffer

Frontline HCI viewer accepts BTSnoop format files, and provides multiple views to analyze the data.

5.1.2 Wireshark

Wireshark is an open source, network protocol analyzer. It accepts capture files, in BTSnoop format, captured with 'hcidump' and 'btmon' utilities in a Linux host environment. Download it here.

📕 hci-snoop.log					- • •
Eile Edit View Go Capture Anal	yze Statistics Telephony Wireless Tools	Help			
	• • • • • • • • • • • • • • • • • •				
Apply a display filter <ctrl-></ctrl->					
No. Time	Source	Destination	Protocol	Length Info	
1.0.000000	host	controller	HCT CMD	9 Sent Trauiry	
20,007393	controller	host	HCI EVT	7 Royd Command Status (Inquiry)	
30.604662	controller	host	HCI EVT	258 Royd Extended Inquiry Result	=
42.708933	controller	host	HCI EVT	258 Royd Extended Inquiry Result	
510.249339	controller	host	HCI EVT	4 Royd Inquiry Complete	
610.250794	host	controller	HCI CMD	14 Sent Remote Name Request	
710.251432	controller	host	HCI EVT	7 Royd Command Status (Remote Name Request)	
810.325370	controller	host	HCI EVT	17 Rovd Remote Host Supported Features Notification	
918,334308	controller	host	HCT_EVT	258 Royd Remote Name Request Complete	Ψ.
•		ш			
> Frame 3: 258 bytes on wire	(2064 bits), 258 bytes captured (2	864 bits)			*
4 Bluetooth					
[Source: controller]					
[Destination: host]					
# Bluetooth HCI H4					
[Direction: Rcvd (0x01)]					
HCI Packet Type: HCI Eve	nt (0x04)				=
4 Bluetooth HCI Event - Exten	ided Inquiry Result				
Event Code: Extended Ing	uiry Result (0x2f)				
Parameter Total Length:	255				
Number of responses: 1					
BD_ADDR: IntelCor_ca:60:	1a (34:f3:9a:ca:60:1a)				
Page Scan Repetition Mod	le: R1 (0x01)				
Reserved: 0x02					
Class of Device: 0x02010	c (Computer:Laptop - services: Net	working)			
.111 1011 0001 1101 = Cl	ock Offset: 0x7b1d				
#ST7.128					· · · · · · · · · · · · · · · · · · ·
0000 04 2f ff 01 1a 60 ca 9a	i f3 34 🔃 02 0c 01 02 1d -/`	· · · 4 <mark>·</mark> · · · · ·			*
0010 7b c7 0c 09 4c 54 41 30	9 38 36 38 33 34 35 42 02 {LT	A0 868345B-			
0020 0a 07 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00				=
8050 08 08 80 80 80 08 08 08 AA	08 08 08 08 08 08 08 08 08 00				
8070 08 08 80 80 00 08 08 80	08 08 08 08 08 08 08 08 00				
00 00 00 00 00 00 00 00 00 00 00	08 08 08 08 08 08 08 09 00				
					Ψ.
A hci-snoop.log				Packets: 13 · Displayed: 13 (100.0%)	Profile: Default



5.1.3 Ellisys

Ellisys protocol analyzer allows the import of BTSnoop format files. Use the Analyzer version 6928 or higher. Download it here.

🔀 Untitled* - Ellisys Bluetooth Analyzer									- 0	×
File View Layout Search Record Tools Help								🔲 Analysis 🖽 Ful	l screen 🛛 📑 Add	7
🗄 🏫 🙀 🙀 🌺 🕨 Record 🗸 🗉 Stop 🔯 Restart 👼 Save	8. Continue 🐝	💥 🖉 Naviga	ste 🕞 🖾 📄 Ma	rkers • 🖉 🖫	Filtering: Exclude Background •					/
HCI Overview (Injection) Message Log H Instant Spectrum	n					4 6 ×	Details			а х
Protocol: Multiple selection •	🔬 🎝 💼 💰	pi 🕕 🖂 🔛	I 📖 🕥 🎿 💷	19 items dis	splayed	Search - 😪	★ All fields Show in overview	Display -	Search	
Type filter	Type filter Y	 Type filt 	Type filter	7 • No 7 ·	• Type filter	▼ • Type ▼ •	Name	Value		~
Item	Status	Payload	Time	Time de	Communication	Application	* RFCOMM SABM Frame (Channel=	1)		
HCI Inquiry (LAP=9E:8B:33 (General Inquiry), Length=10.2 s, R	OK		0.921 912 300		Master: "Host" Unknown BD ADDR <->	SI HCI	🗉 📥 L2CAP Frame	Dst=0x0040 (RFCOMM)		
A HCI Read Simple Pairing Mode > No	OK		22.036 331 300	21.114	Master: "Host" Unknown BD ADDR <->	SI HCI	🗉 🛶 L2CAP SDU	Basic, 4 bytes		
HCI Create Connection (34:F3:9A:CA:60:1A, Allow Switch=Y	OK		22.037 457 300	0.001 1	Master: "Host" Unknown BD_ADDR <->	SI HCI				
A HCI Read Simple Pairing Mode > No	OK		22.938 546 300	0.9010	Master: "Host" Unknown BD ADDR <->	SI HCI	B B RECOMMERTAME			
E CAP Connection (Src=0x0080, PSM=RFCOMM > Dst=0x00	OK		22.941 053 300	0.002 5	Master: Unknown BD ADDR <-> Slave:	3 L2CAP	Information			
← HCI Max Slots Change (Connection=0x0001, Slots=5)	OK		22.942 038 300	0.000 9	Master: "Host" Unknown BD ADDR <->	SI HCI	Initiator	Master: Unknown BD_ADDR		
⊕ ↔ L2CAP Configure (Dst=0x0040, MTU=335 > Src=0x0080)	OK		23.020 487 300	0.078 4	Master: Unknown BD_ADDR <-> Slave:	3 L2CAP	Responder	Slave: 34:F3:9A:CA:60:1A		
E + 12CAP Configure (Dst=0x0080, MTLL=1'017 > Src=0x0040, M.	OK		23.020 880 300	0.000 3	Master: Linknown BD_ADDB_<-> Slave:	3 L2CAP	🗏 🔧 Header			
Generation=0x0001. To	OK		23.022 548 300	0.0016	Master: "Host" Linknown BD_ADDB <->	SL., HCT	🖂 🧼 DLCI	0x02		
	OK		23 028 122 300	0.005.5	Master: Linknown BD, ADDR, <-> Slave:	3 RECOMM	Direction Bit	Served by Responder		
PECOMM DLC Parameter Negotiation (Channel – 1 Initial Cre	OK		23.029.460.300	0.001 3	Master: Unknown BD_ADDR <-> Slave:	3 RECOMM	Server Channel	1		
REPERT COMM Connect (Channel = 1)	OK		23.042.148.300	0.012.6	Master: Unknown BD_ADDR <-> Slave:	3 RECOMM	Ontrol	SABM Frame		
B C BECOMM SABM Frame (channel = 1)	OK OK	No data	22.042 148 300	0.012 0	Master: Unknown BD_ADDR <-> Slave:	2 RECOMM	Frame Type (part 1)	1111		
	OK OK	No data	23.042 148 300	0.001.0	Master University DD_ADDR <> Clause	3 NI COMM	Poll/Final Bit	1		
RECOMM Madem Status (Channel 1)	OK	NO Gata	23.043 763 300	0.0016	Master: Unknown BD_ADDR <-> Slave:	3 RECOMM	Frame Type (part 2)	001		
Recommendations (charmen 1, bata value wo (No)	UK IS		23.042 804 300	0.000.7	Master: Unknown BD_ADDR <-> Slave:	S RECOMM	RECOMM LIA Frame (Channel=1)			_
RECOMM Modem Status (Channel = 1, Data Valid=Yes)	No Request Fro		23.043 522 300	0.000 7	Master: Unknown BU_ADDR <-> Slave:	3 RFCOMM				~
RFCOMM UIH Frame (Channel=1, Credits=1: 0 R: 0+3=3)	OK	No data	23.044 095 300	0.000 5	Master: Unknown BD_ADDR <-> Slave:	3 RFCOMM	Details 😴 Instant Piconet 🥶 Summ	nary 🔤 🏙 Instant Channels		
RFCOMM Modem Status (Channel=1, Data Valid=Yes)	No Response Fr		23.044 368 300	0.000 2	Master: Unknown BD_ADDR <-> Slave:	3 RFCOMM	Raw data			ąχ
+ HCI Number Of Completed Packets (Connection=0x0001, Pa	OK		23.086 217 300	0.0418	Master: "Host" Unknown BD_ADDR <->	SI HCI			10.1	_
+ HCI Number Of Completed Packets (Connection=0x0001, Pa	OK		23.086 737 300	0.000 5	Master: "Host" Unknown BD_ADDR <->	SI HCI	Data type: RECOMM SABM Frame (CF	hannel=1) 🔹	Search	
HCI Mode Change (Success, Connection=0x0001, Mode=Sni)	I OK		43.426 970 300	20.340	Master: "Host" Unknown BD_ADDR <->	SI HCI	0 1 2 3 4 5 0x0000: 04 00 40 00 0B 3B	0123456789A	3C	
Instant Timing						4 X				
🔪 🖓 🔍 🔲 🚡 🔹 origin: 22,119.06 ms 🔹 span: 1.94 m	s •	Bluetooth 👻 Wi	Fi HCI WCI WF	PAN Logic Mis	c 🔹 Display 🔹 Logic inputs	ि । अ				
HCT (Injected)		bd		<u> </u>		^				
				·		_				
Link OUT										
Link IN										
RFCOMM										
0 22,119.00 ms	0.70 0.80	^{0.90} 22,120.0	0 ms 0.10	0.30 0.40	0.50 0.60 0.70 0.80	0.90 22,121				
🚧 Instant Timing 🎝 Instant Audio 📥 Instant Throughput							101 Raw data 👌 Security			_
Ready									5.0.6928.24	421
Type here to search) 🗄 🤅) 🖬 🗄	· 4	o <mark>z 💻</mark>	🏭 🖊 🔮 <u>X</u>			я ^р ^ (5:46 PM 12/29/2018	1

Figure 21. Ellisys

6 References

- Cortex-M4 User guide (ARM DUI 0508A)
- Cortex-M4 Technical Reference Manual (ARM DDI 0439B)
- Bluetooth Core Specification V4.2
- BT-HCI Logging

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2019, Texas Instruments Incorporated