Application Report
SWPA234–January 2019

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

List of Figures

Copyright © 2019, Texas Instruments Incorporated

List of Tables

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.

Copyright © 2019, Texas Instruments Incorporated

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.

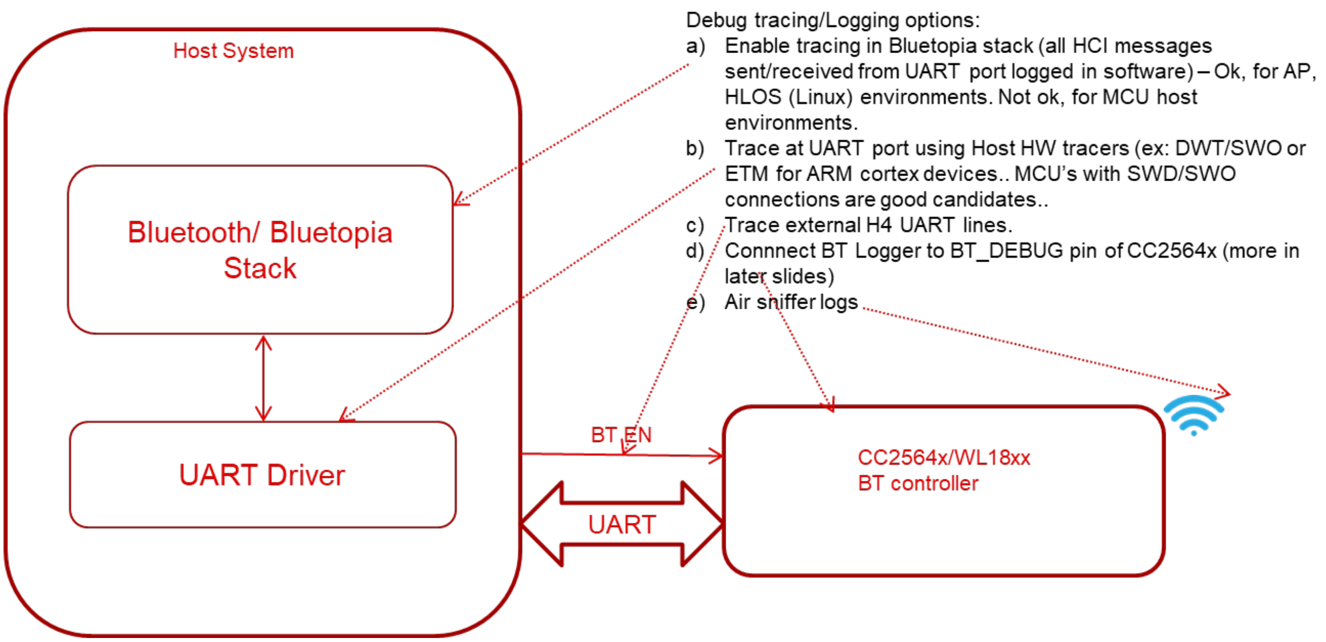


Figure 1. Logging Options in a Bluetooth System

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

Copyright © 2019, Texas Instruments Incorporated

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.

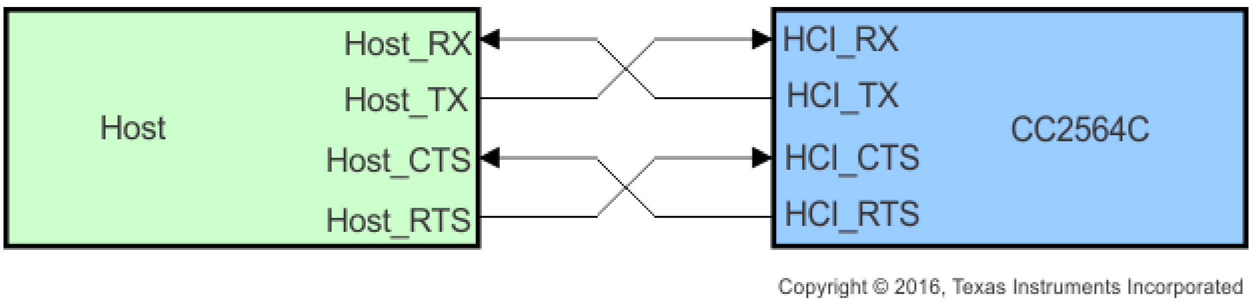


Figure 2. 4-Wire HCI/UART Interface (H4)

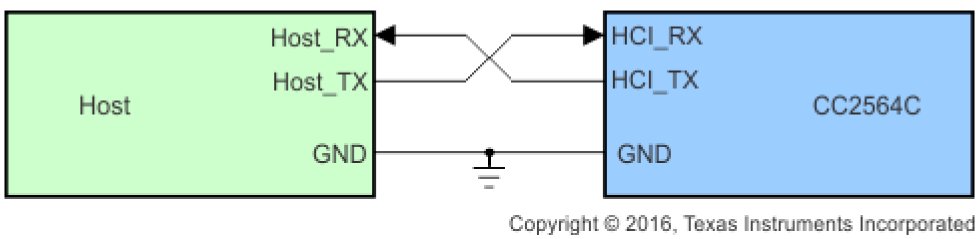


Figure 3. 3-Wire HCI/UART Interface (H5)

2.1 HCI Protocol

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.

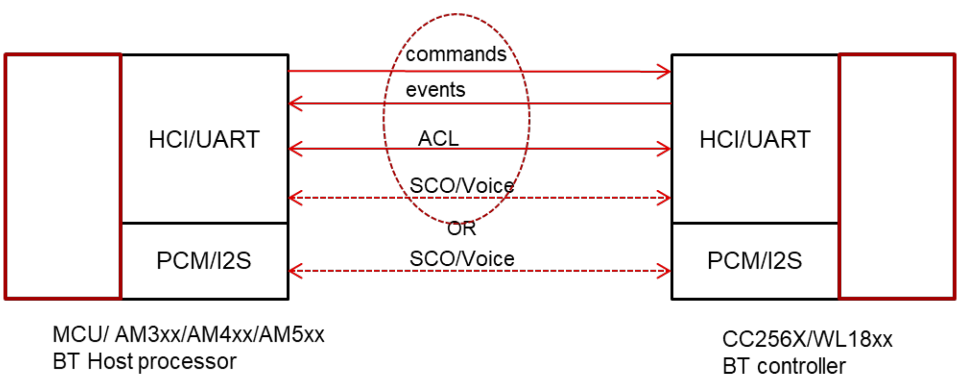


Figure 4. HCI Protocol

Copyright © 2019, Texas Instruments Incorporated

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.

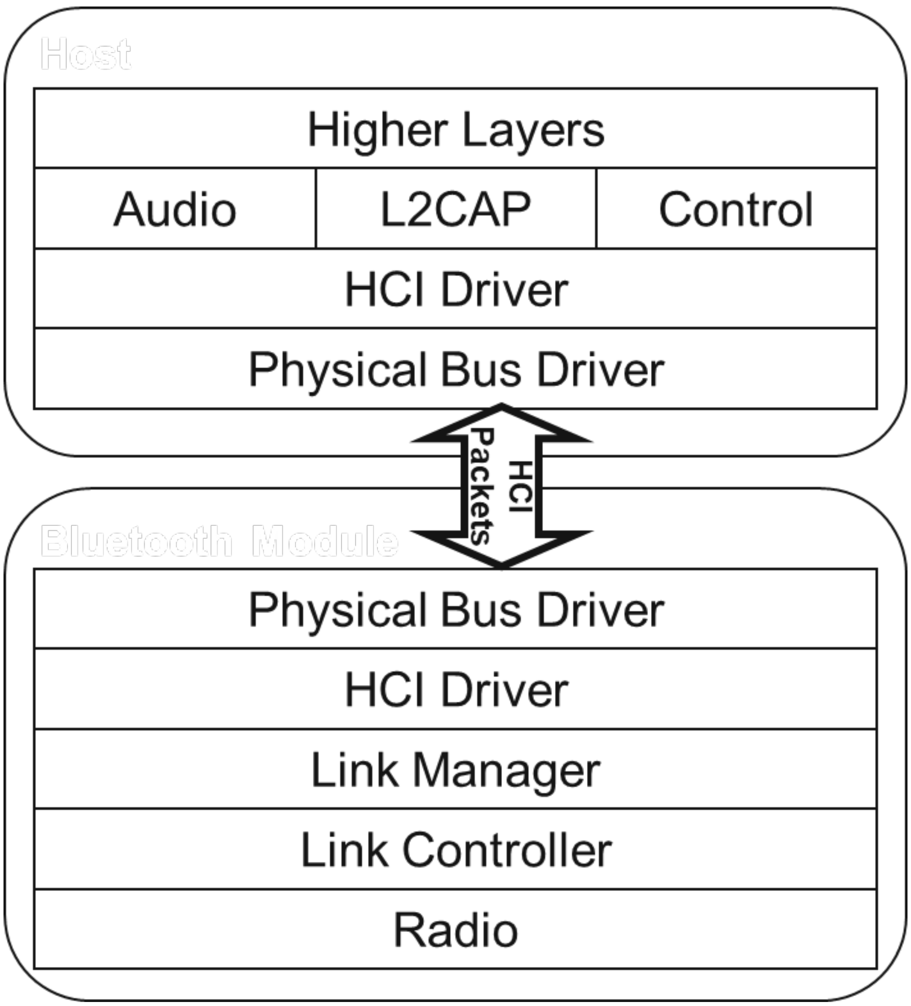


Figure 5. Bluetooth® Stack Layers

Copyright © 2019, Texas Instruments Incorporated

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
DebugFlags    Optional flag for debuging. Currently the only optional value is the following:
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 device-level 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
```


Copyright © 2019, Texas Instruments Incorporated

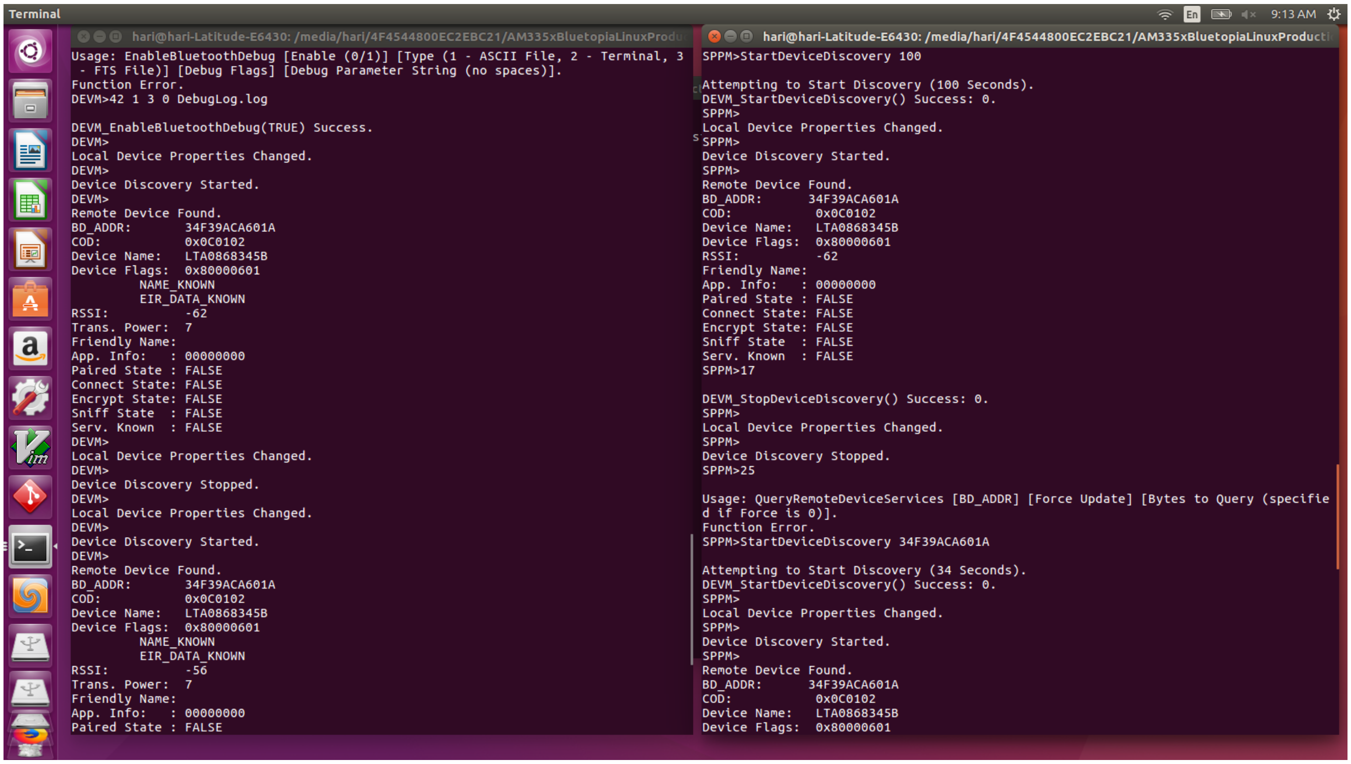


Figure 6. HCI Logging in BluetopiaPM Environment

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>
```


Copyright © 2019, Texas Instruments Incorporated

```
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:
        btmon [options]
options:
        -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
root@am335x-evm:~# btmon -w hci-fte-snoop.log
Bluetooth monitor ver 5.46
[  496.280194] NET: Registered protocol family 38
= Note: Linux version 4.14.40-g4796173fc5 (armv7l)
                             0.827034
= Note: Bluetooth subsystem version 2.22
                             0.827046
= New Index: 54:4A:16:13:1C:21 (Primary,UART,hci0)
                      [hci0] 0.827050
= 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: bluetoothd (privileged) version 1.14
                    {0x0001} 0.827063
@ MGMT Open: btmon (privileged) version 1.14
                    {0x0002} 0.827147
@ RAW Open: hcitool (privileged) version 2.22
```

Figure 7. hcidump Tool Usage in a Linux Environment

```
# btmon -w <BTsnoop file name>
```


Copyright © 2019, Texas Instruments Incorporated

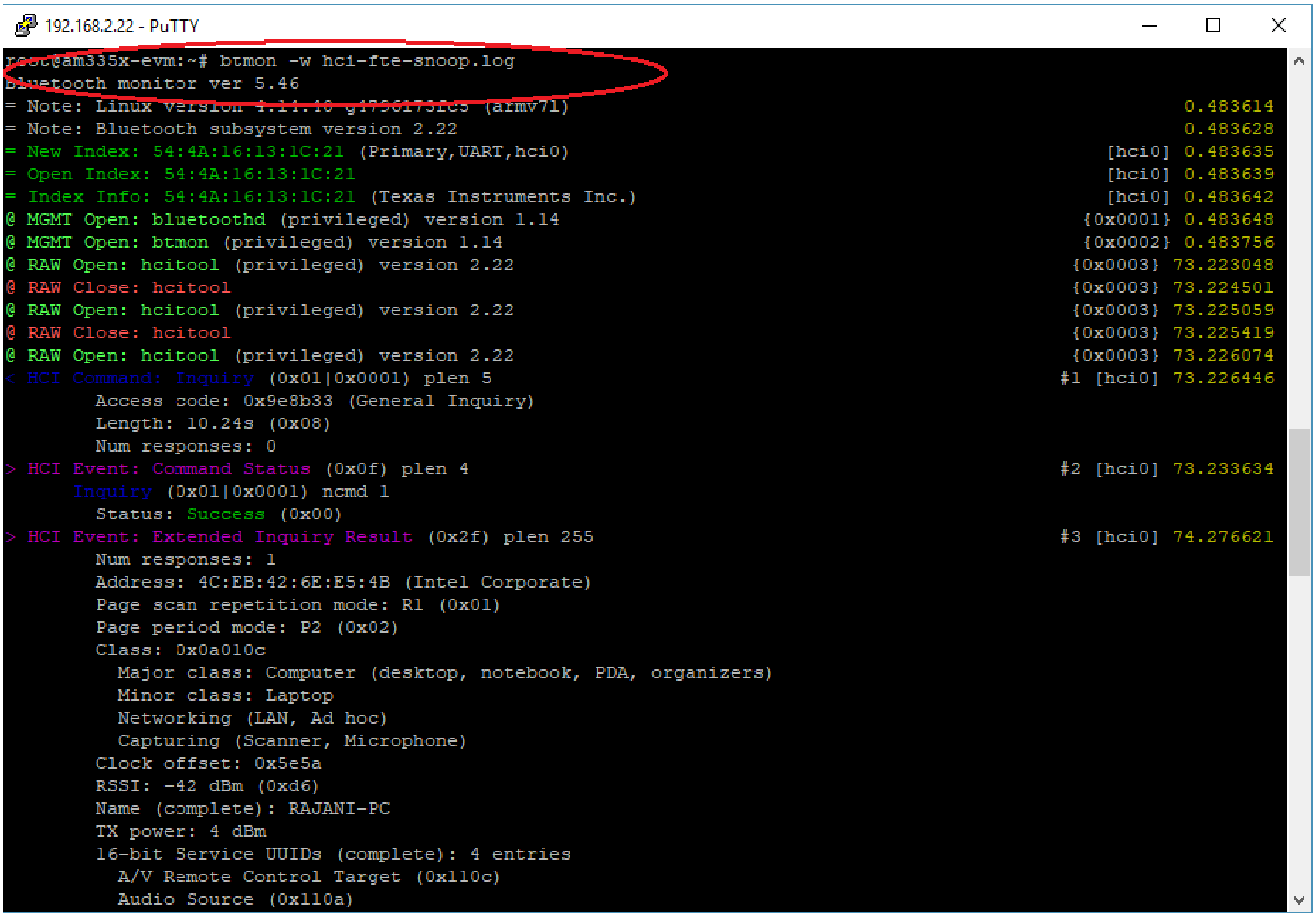


Figure 8. btmon Tool Usage in a Linux Environment

3.3 Logging With Bluedroid/Android

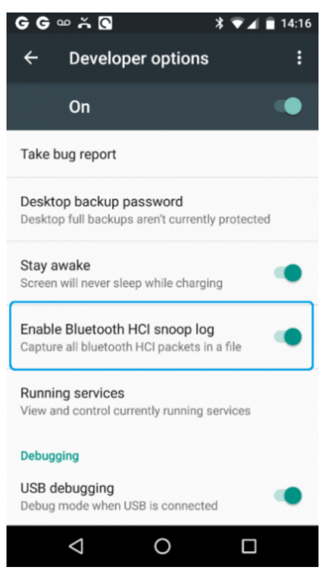


Figure 9. Enable Blueooth HCI Snoop Log

Copyright © 2019, Texas Instruments Incorporated

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
     XXXXXXXXXXX device
     ```
  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.

Copyright © 2019, Texas Instruments Incorporated

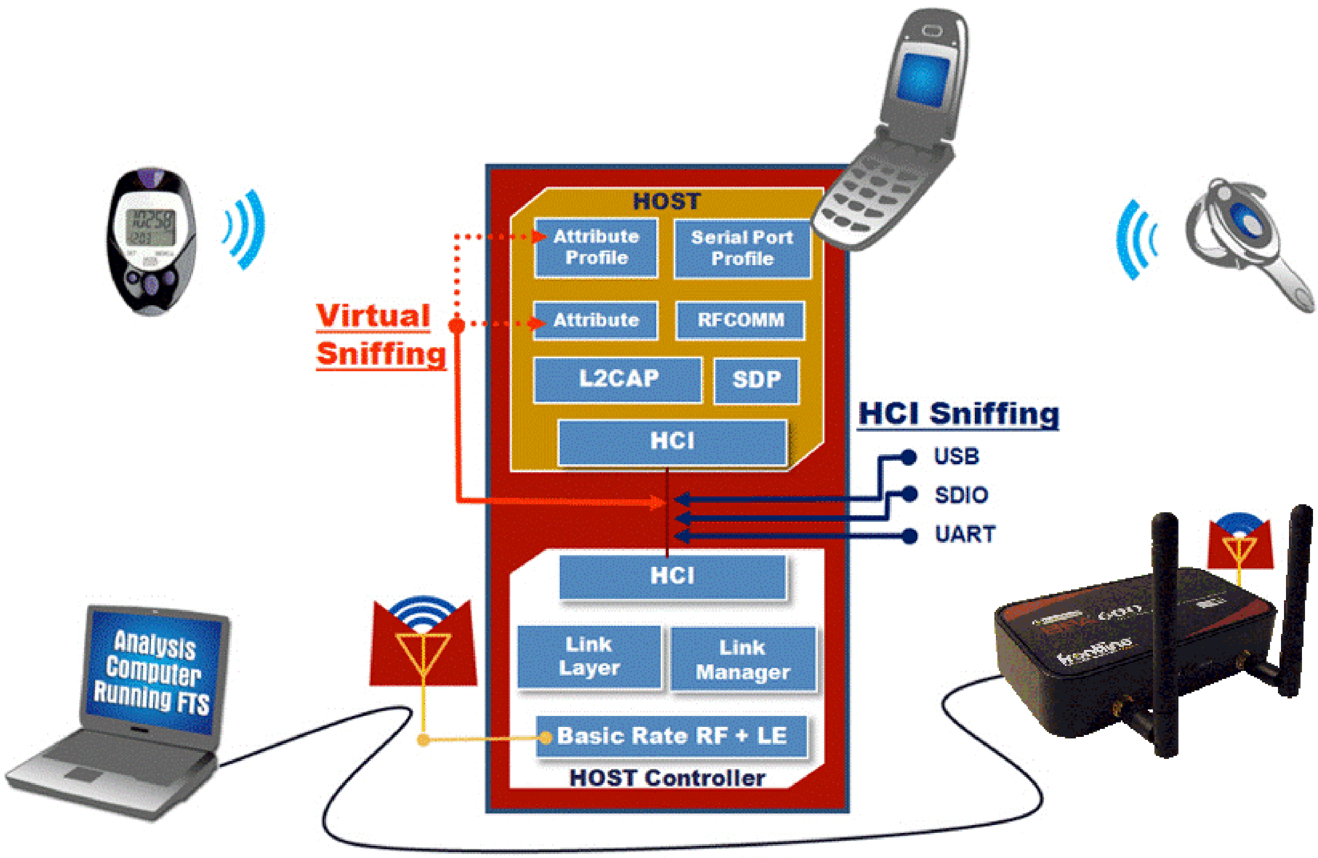


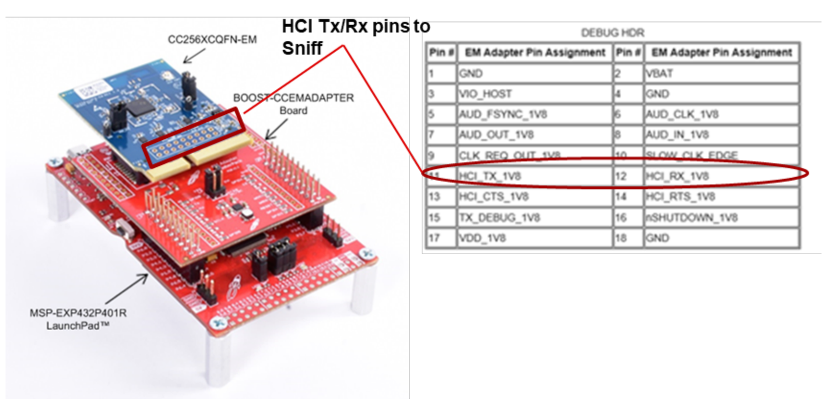
Figure 10. Frontline Technologies HCI Sniffer (Using Serial Sniffer)

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.



Pin #	EM Adapter Pin Assignment	Pin #	EM Adapter Pin Assignment
1	GND	2	VBAT
3	VIO_HOST	4	GND
5	AUD_FSYNC_1V8	6	AUD_CLK_1V8
7	AUD_OUT_1V8	8	AUD_IN_1V8
9	CLK_REQ_OUT_1V8	10	SLOW_CLK_EDGE
11	HCI_TX_1V8	12	HCI_RX_1V8
13	HCI_CTS_1V8	14	HCI_RTS_1V8
15	TX_DEBUG_1V8	16	nSHUTDOWN_1V8
17	VDD_1V8	18	GND

Figure 11. MSP432 Launchpad With CC2564xQFN-EM

Copyright © 2019, Texas Instruments Incorporated

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

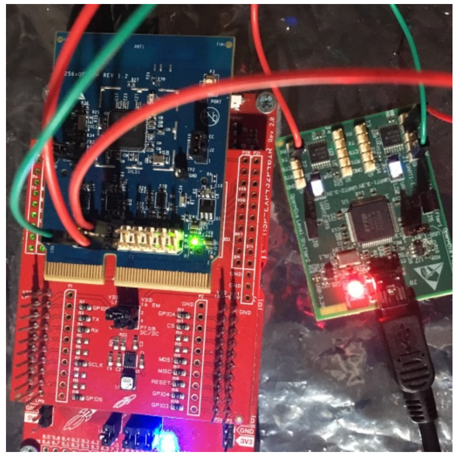


Figure 12. HCI TX/RX Sniff With CC2564xQFN-EM Module

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.

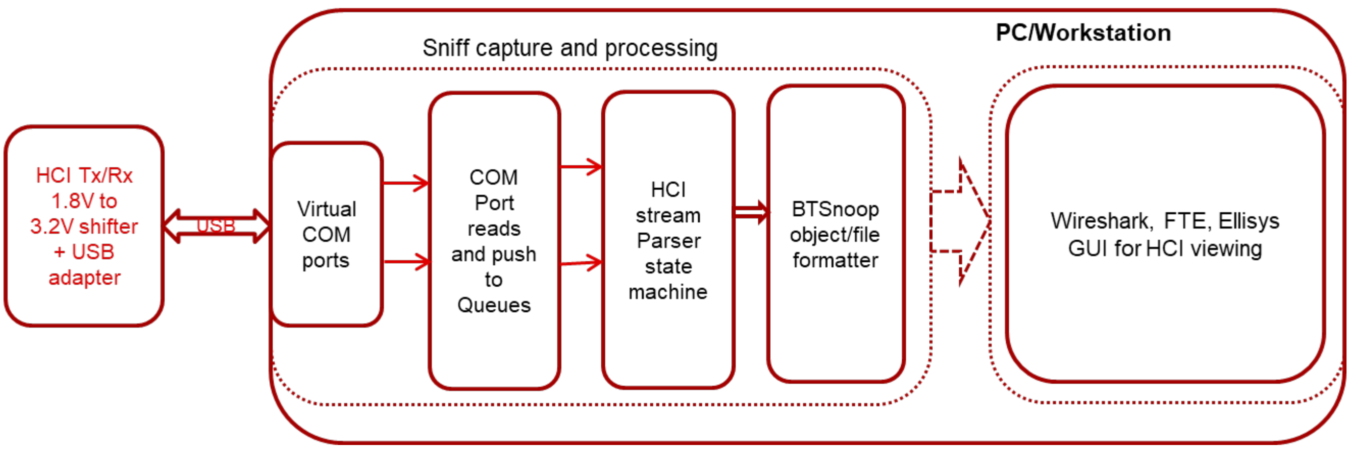


Figure 13. PC-Based HCIsniffer Tool

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

Copyright © 2019, Texas Instruments Incorporated

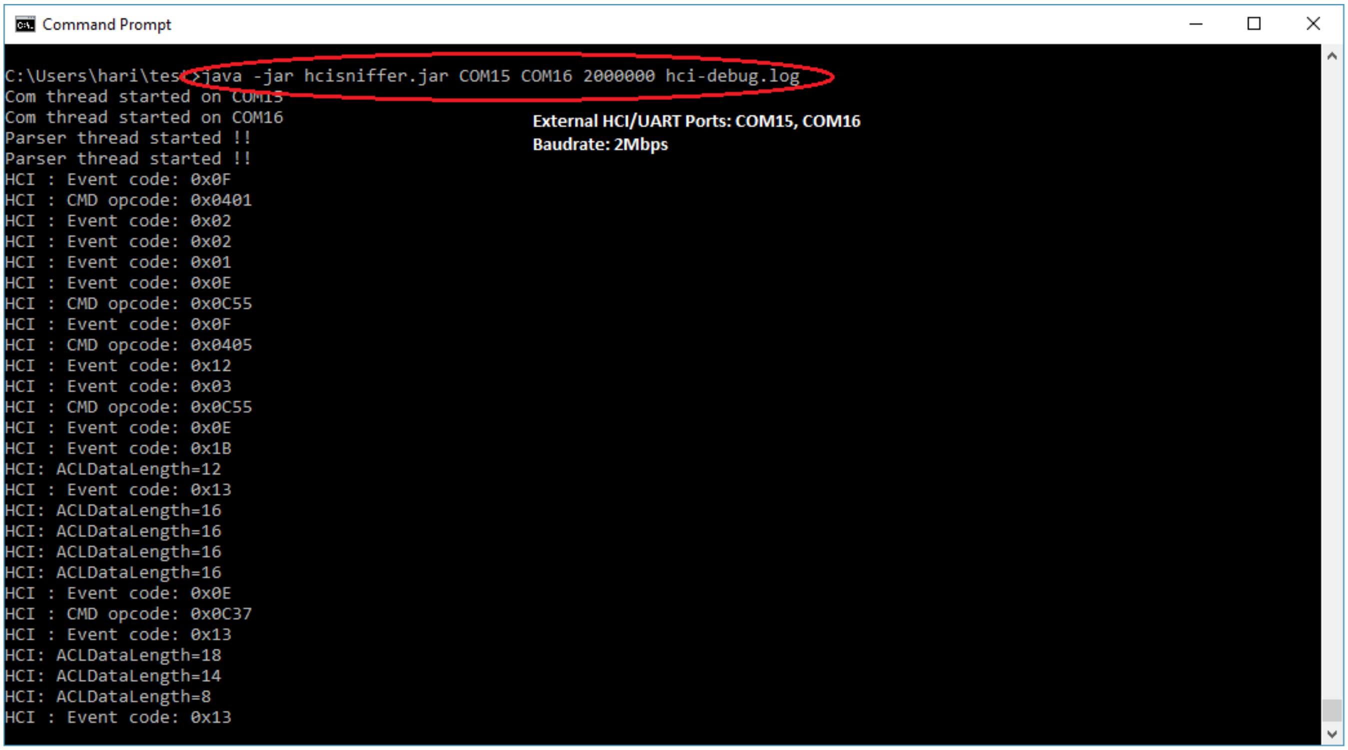


Figure 14. HCIsniffer Command Line Parameters

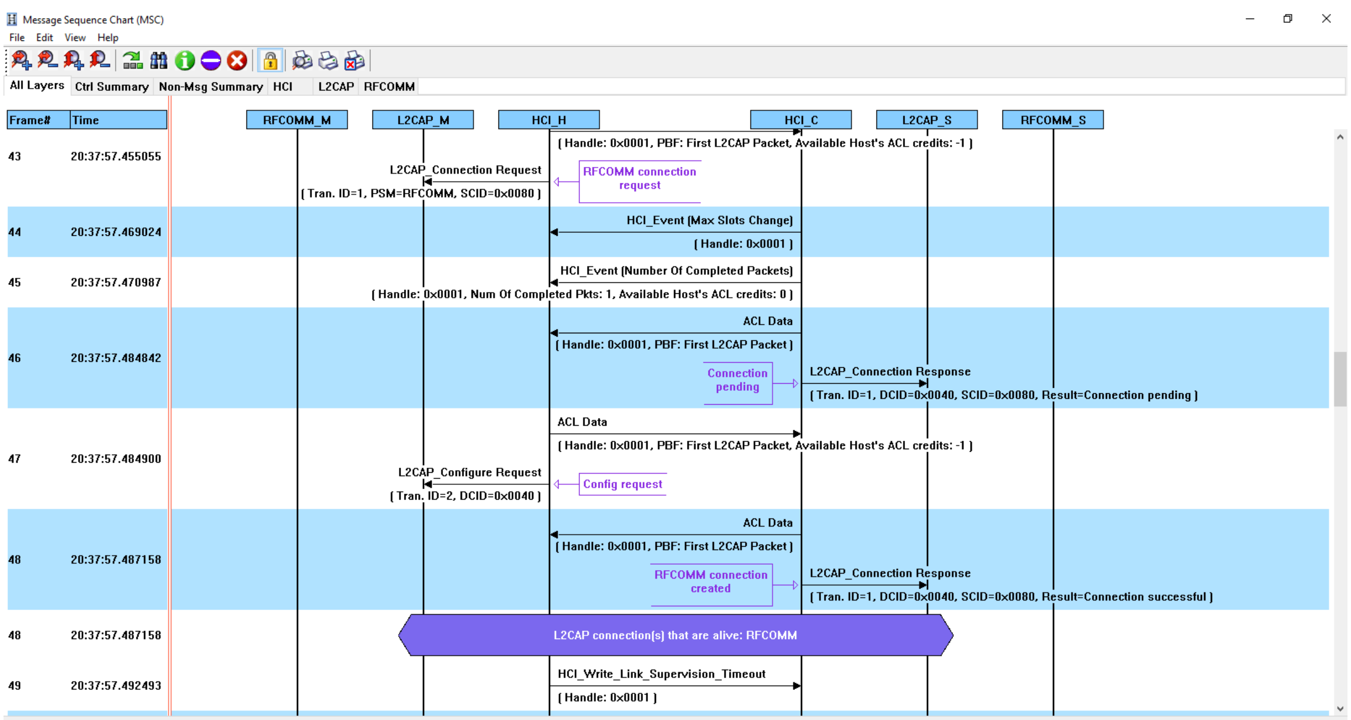


Figure 15. BTSnoop File (Created With HCIsniffer Tool) Viewed With Frontline GUI

Copyright © 2019, Texas Instruments Incorporated

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.

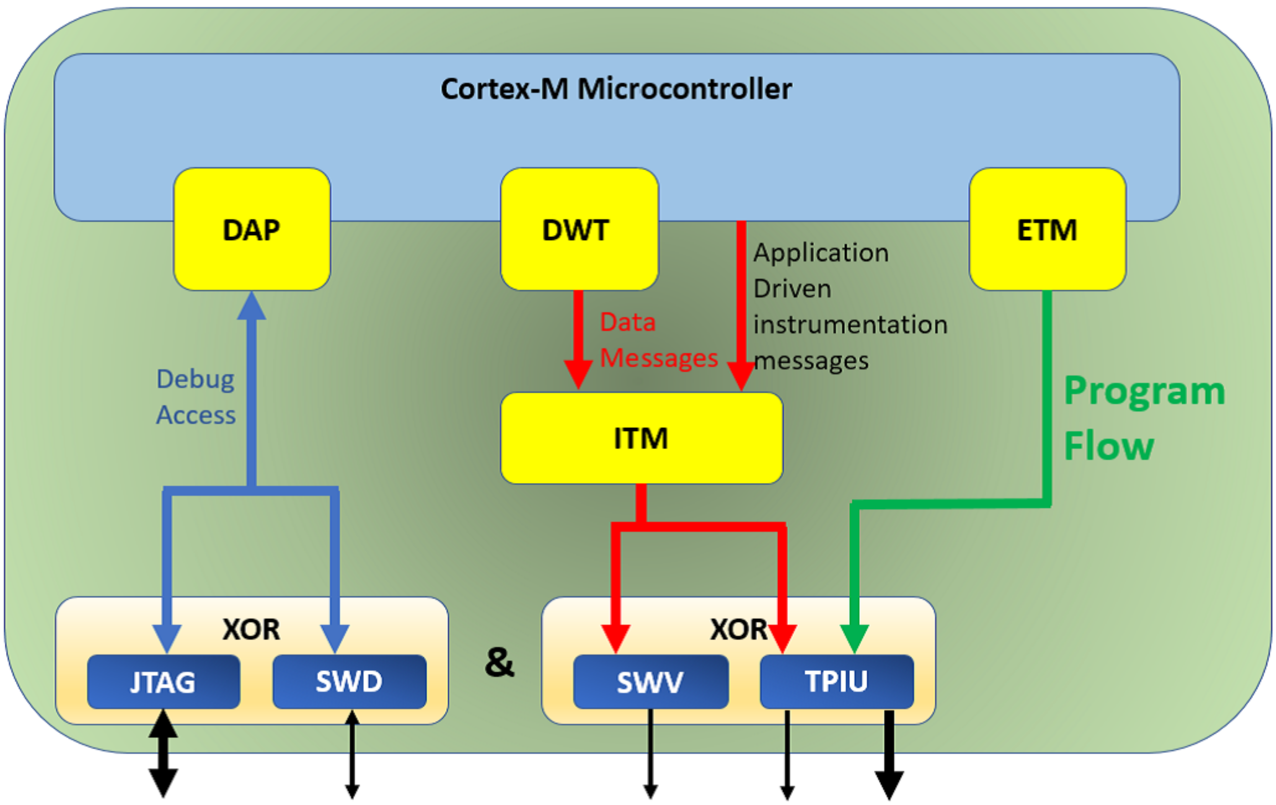


Figure 16. Cortex-M With CoreSight Technology Components

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".

Copyright © 2019, Texas Instruments Incorporated

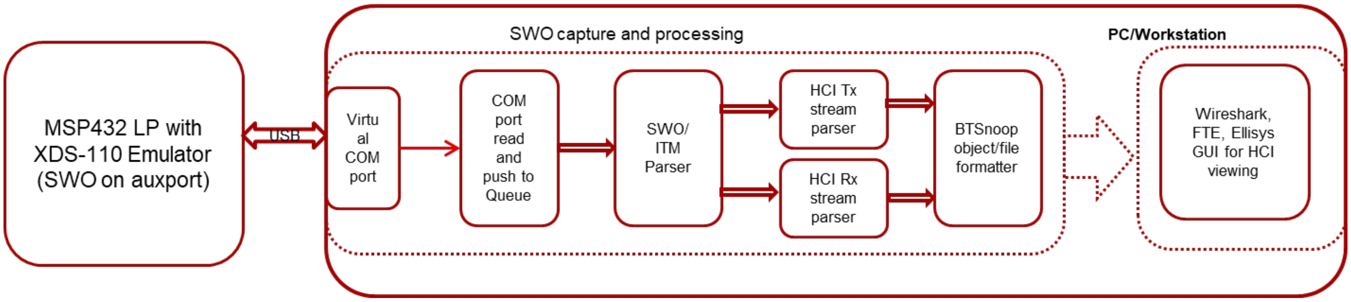


Figure 17. SWO/ITM Stream Capture and Processing

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   = 0x00000000;

    // 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;

    // 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++;
        // 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;
}
```


Copyright © 2019, Texas Instruments Incorporated

```
/** 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);
    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 */
ITM_config itm_config = { 48000000, ITM_12000000};
    /* Setup ITM */
    {
        uint32_t     *hci_rx_ptr = (uint32_t*) 0x4000180C; // MSP432 UART Rx register
        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.

Copyright © 2019, Texas Instruments Incorporated

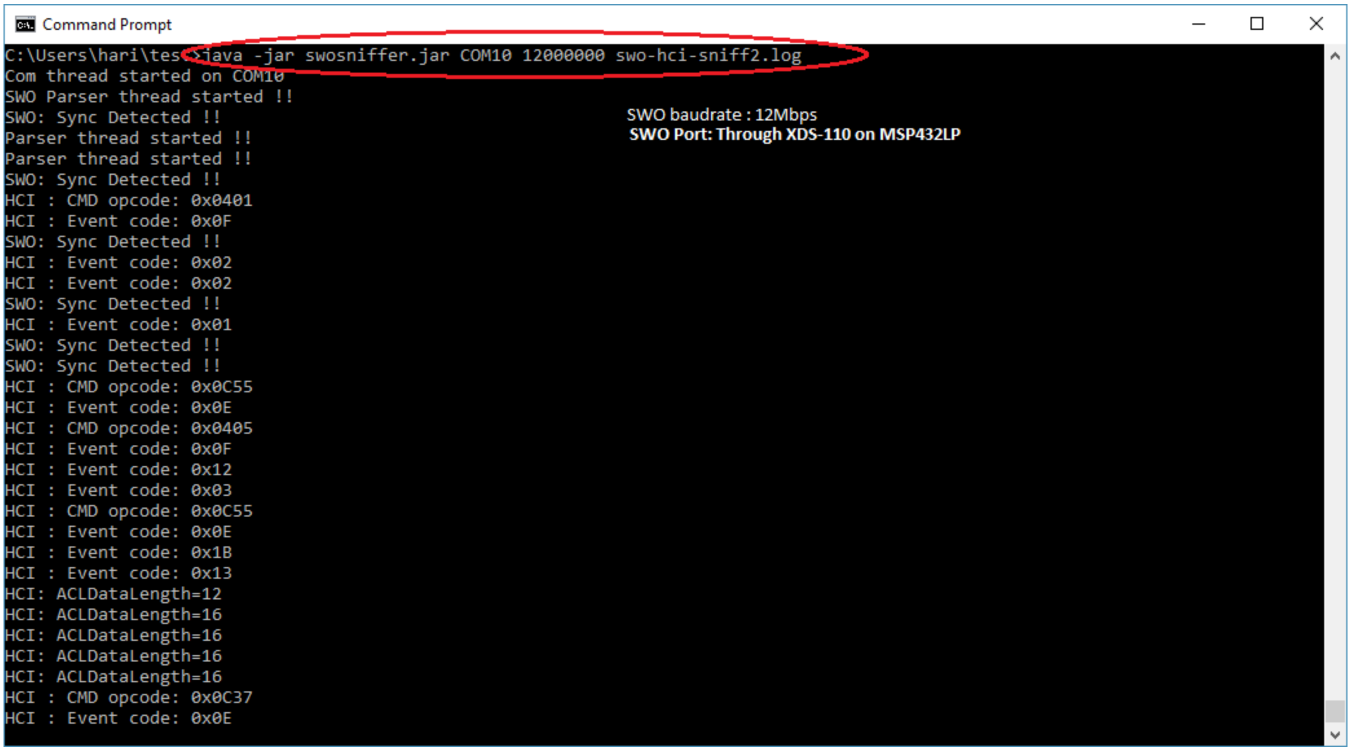


Figure 18. SWOsniffer Program Run

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.

Copyright © 2019, Texas Instruments Incorporated

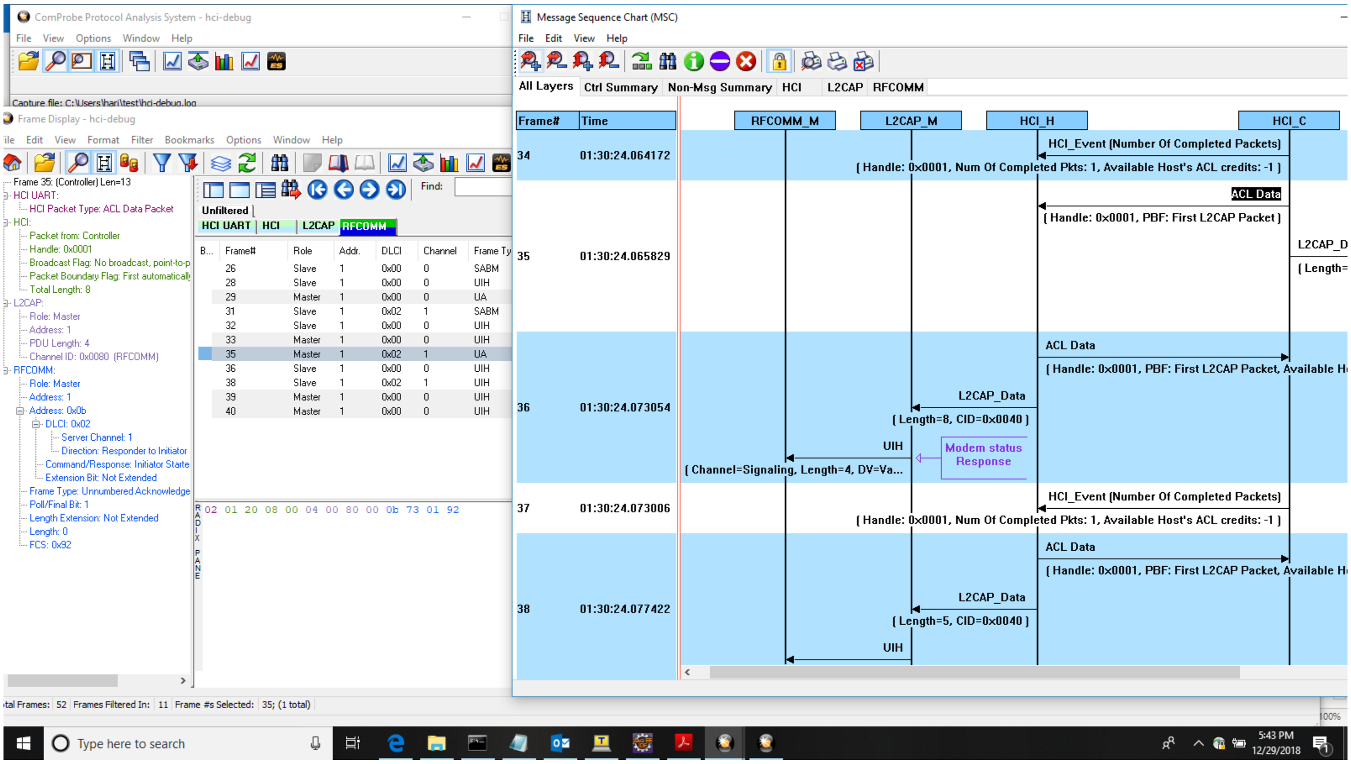


Figure 19. Frontline HCI Sniffer

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.

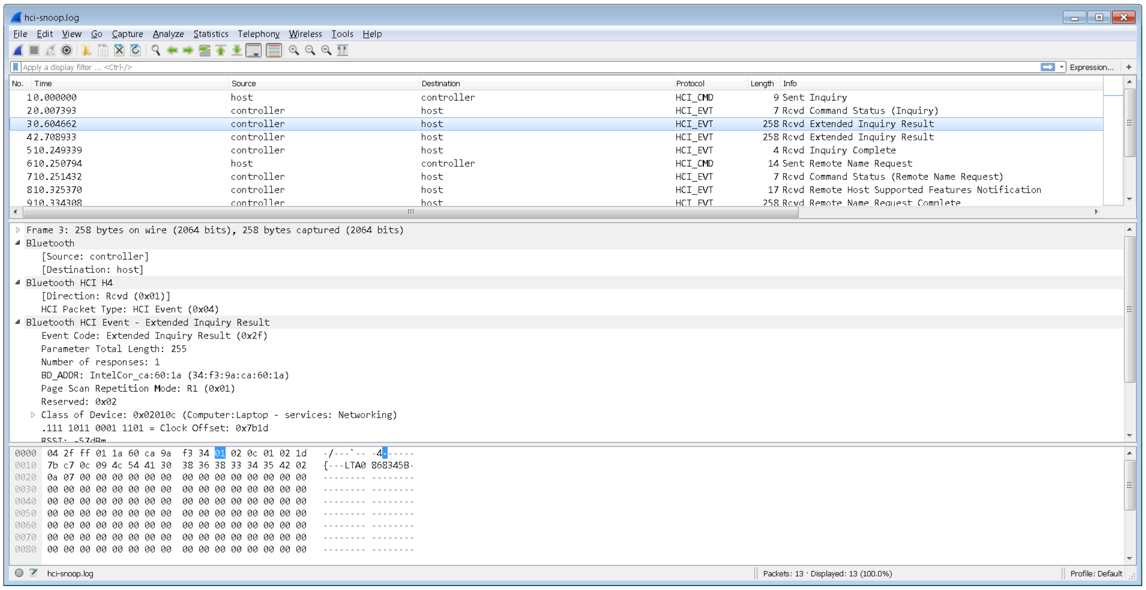


Figure 20. Wireshark

Copyright © 2019, Texas Instruments Incorporated

5.1.3 Ellisys

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

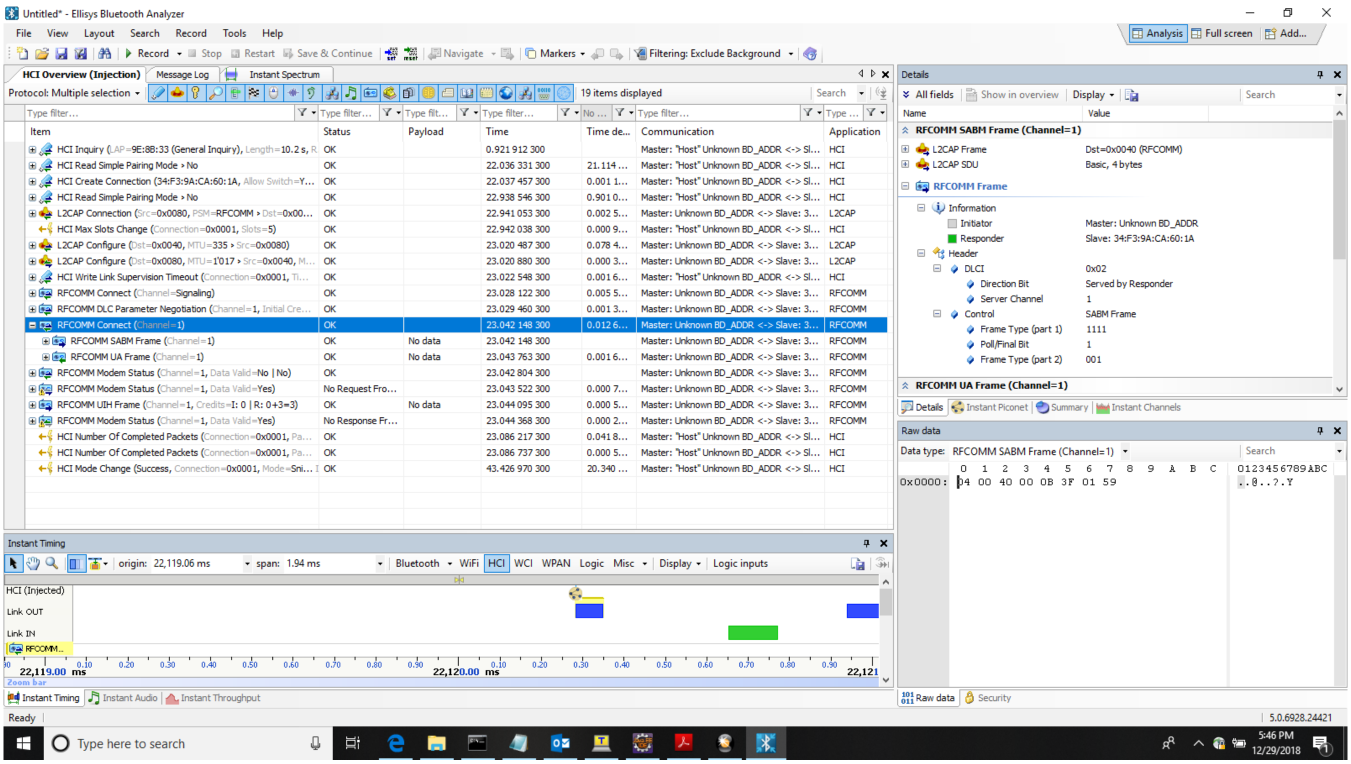


Figure 21. Ellisys

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

Copyright © 2019, Texas Instruments Incorporated

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