AN1328: Enabling a Radio Co-Processor using the *Bluetooth*® LE HCI Function

This document gives a short overview of the standard Host Controller Interface (HCI) and how to use it with Silicon Labs' Bluetooth LE controller. First it briefly describes the HCI layer, the supported features, and explains the difference between a Network Co-Processor (NCP) and a Radio Co-Processor (RCP) project. It then lists the available vendor-specific commands and shows how to get started with the RCP Example project included in the Bluetooth LE SDK.

This document assumes that you have installed Simplicity Studio 5 and the Bluetooth LE SDK, and that you are familiar with creating, configuring, building, and flashing projects. If not, see *QSG169: Bluetooth® SDK v3.x Quick-Start Guide*.

KEY POINTS

- Generic description of the HCI protocol
- Vendor-specific features
- Example project description

1 Introduction

The HCI is described in detail in Bluetooth Core Specification, Vol 4: Host Controller Interface. In the Silicon Labs software and documentation, the HCI is also known as RCP (Radio Communication Protocol).

The HCI is a standardized way for Bluetooth host and controller to communicate with each other. Since the interface is standard, the host and controller can be from different vendors.

The following figure illustrates the layered communication mechanism between the Bluetooth Controller and Host protocol stacks, and the user application.

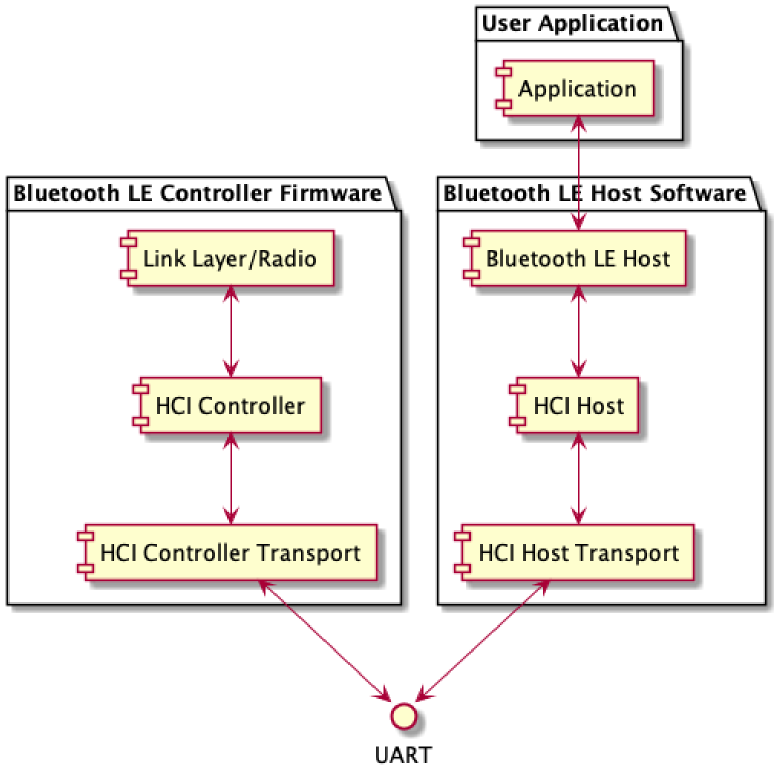


Figure 1.1. Bluetooth High-Level Components

The lowest layer between the host and controller communication is the HCI Transport. Currently Silicon Labs Bluetooth Controller supports UART (Universal Asynchronous Receiver-Transmitter) as the HCI transport layer.

The next layer is the HCI, which provides set of commands and events, and ACL data packets. The Host side sends commands to the controller. The commands are used to start advertising or scanning, establish connection to another Bluetooth device, read status information from the controller, and so on.

The events are sent from the Controller side to the Host. The events are used as a response to commands or to indicate various events in the controller such as scanning reports, connection establishment or closing, and various failures.

The ACL (Asynchronous ConnectionLess) data packets are used to deliver user application data between the host and controller in both directions. The data packets are exchanged between Bluetooth devices.

The Silicon Labs Bluetooth LE Controller does not support SCO (Synchronous Connection Oriented) and ISOC (Isochronous Channels) modes, and the related HCI messages are not supported.

The Bluetooth specification defines three types of controllers: Low Energy (LE), BR/EDR (classic) and AMP (alternate MAC and PHY). Silicon Labs supports only the LE controller.

The Silicon Labs Bluetooth LE Controller runs on EFR32 Radio Co-Processors, and external Bluetooth Host stacks can communicate with the controller over the HCI. This is also called RCP mode, and is illustrated in the following figure.

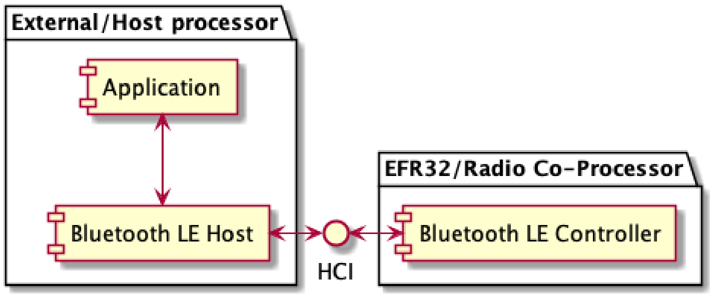


Figure 1.2. RCP Mode

Silicon Labs' previously provided solutions, NCP (Network Co-Processor) mode and SoC mode, are illustrated in Figure 1.3 and Figure 1.4, respectively. In the NCP mode the Silicon Labs Host and Controller protocol stacks run on the EFR32 Radio Co-Processor. The application runs on a separate processor and communicates with Silicon Labs Bluetooth stack over the proprietary NCP protocol (BGAPI), which is exposed to the application via UART.

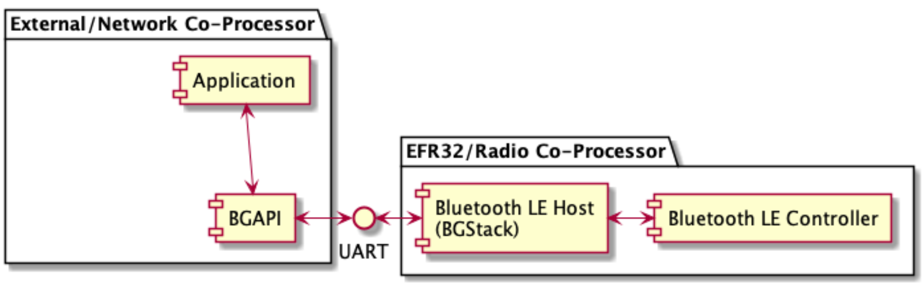


Figure 1.3. NCP Mode

In SoC mode the application runs on the same processor as the Silicon Labs Bluetooth LE protocol stacks. The application communicates with the Bluetooth LE protocol stack via BGAPI.

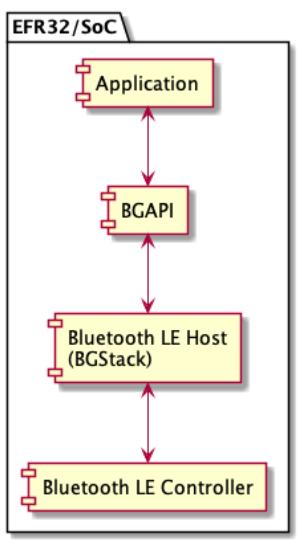
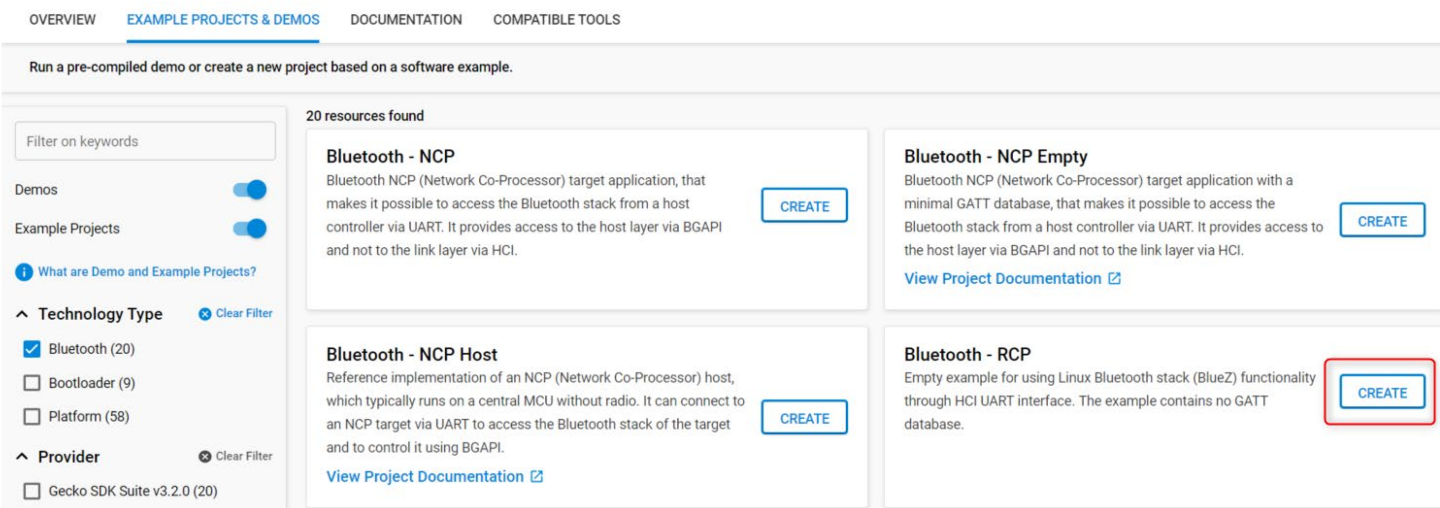


Figure 1.4. SoC Mode

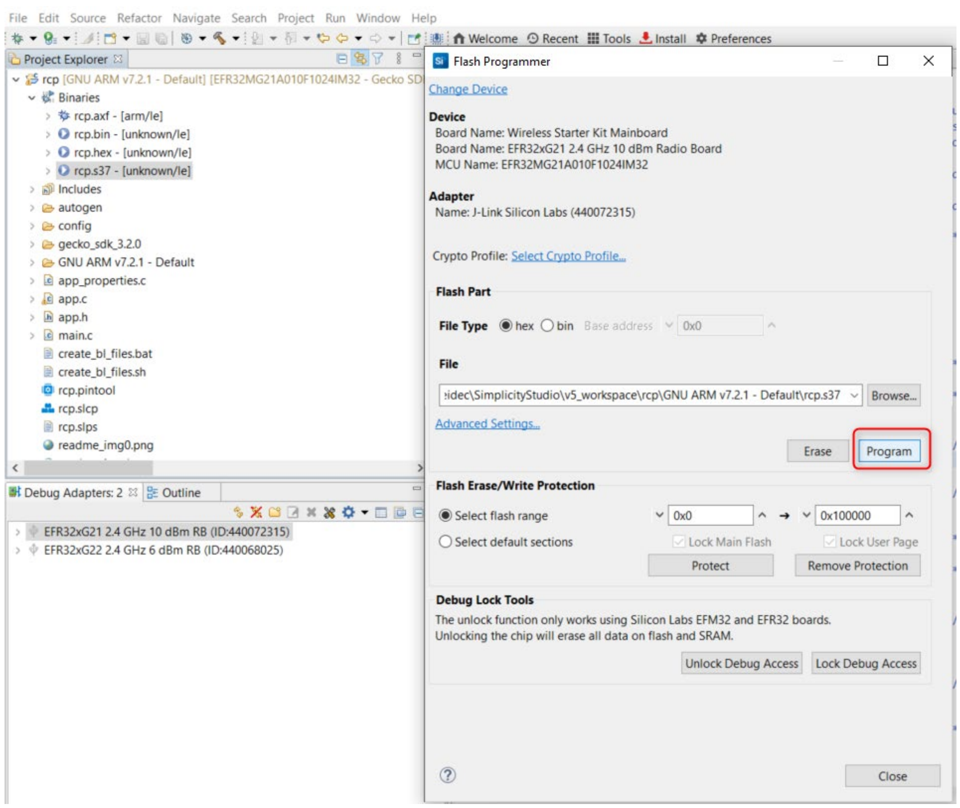
2 Enabling HCI Functionality

2.1 Using an Example Application

The Gecko SDK suite includes the **Bluetooth – RCP** example application, which can be used to create an HCI-capable Bluetooth Controller application that can be run in RCP mode. An external Bluetooth Host protocol stack, such as Linux Bluez, can use the Bluetooth Controller via HCI. To use the example, select the target device in the Debug Adapters view, find the **Bluetooth – RCP** example on the EXAMPLE PROJECTS & DEMOS tab, and create the project.



After building the project, flash it to the target device as with any other project:

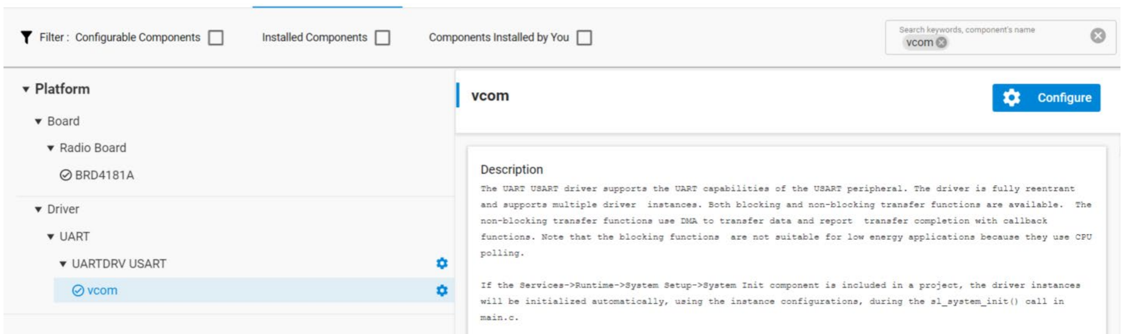


The application configuration can be reviewed and changed in the Project Configurator under the SOFTWARE COMPONENTS tab. To find a specific component, begin typing its name in the Search field.

The application uses UART as the HCI Transport layer. By default, the UART has been configured as follows.

- Hardware flow control: disabled
- Speed: 115200 kbps
- Data bits: 8
- Parity bit: N
- Stop bits: 1

The default configuration can be changed with the **vcom** component.



This will generate the values to the header file *sl_uartdrv_usart_vcom_config.h*. Especially for high-speed communication over UART, enabling hardware flow control is recommended. Instructions for enabling hardware flow control in the WSTKs (Wireless Starter Kits) are given in Section 4, Enabling Hardware Flow Control In the WSTK.

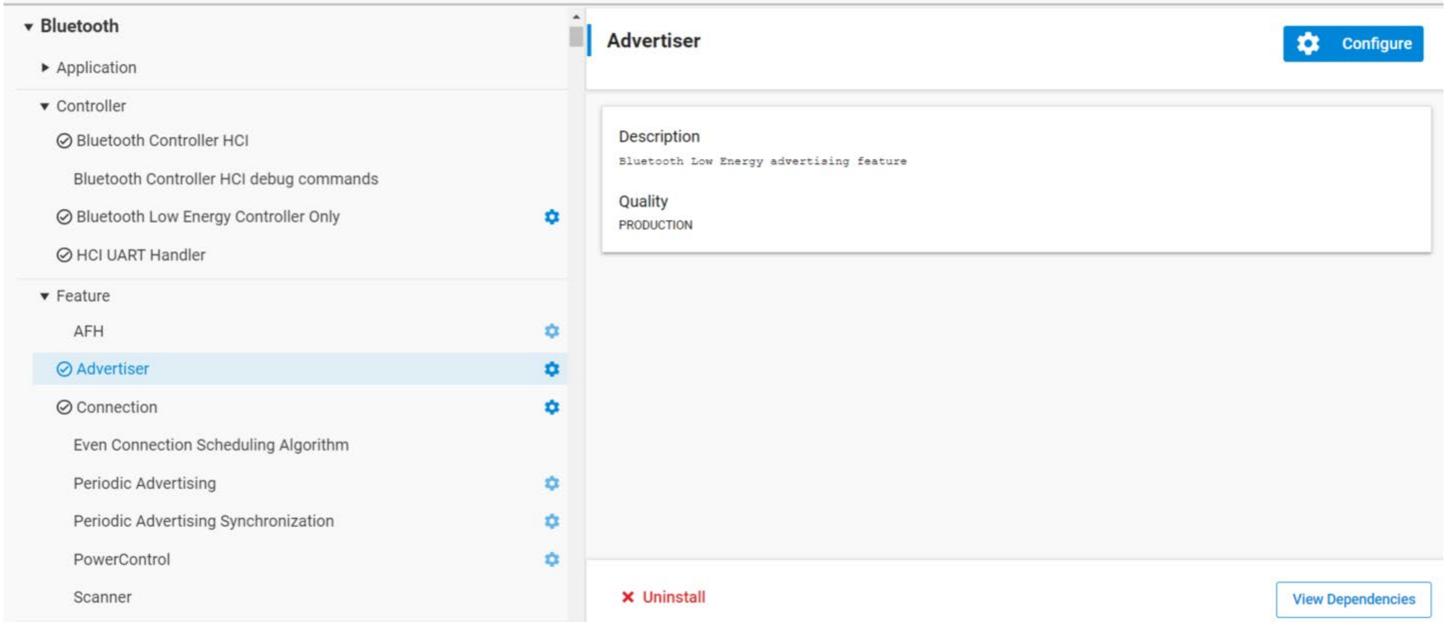
Note that by default nearly all HCI events sent from the controller have been filtered out. The events to be filtered or passed from the controller to host can be configured using the HCI commands *HCI_LE_Set_Event_Mask* and *HCI_Set_Event_Mask*.

2.2 Required Components for HCI Capable RCP Mode Application

For the basic HCI-enabled application in the RCP mode the following components are required, and are installed in the example applications by default.

- Bluetooth Low Energy Controller Only
- Bluetooth Controller HCI
- HCI UART Handler

Other components are required to make a reasonably functional application. For example, the **Connection** and **Advertiser** components are required (and installed by default) to make an application that can advertise and accept connections. If the application should be able to accept a large number of simultaneous connections, also including the **Even Connection Scheduling Algorithm** component is useful.



Another example is to add the ability to scan near-by devices and establish connections to them. This requires the **Connection** and **Scanning** components. For the application to use the power control functionality for connections, include the **Power Control** component.

3 Vendor-Specific HCI Commands and Events

The Silicon Labs HCI and Controller support some vendor-specific commands and events as described in the following two sections.

3.1 Vendor-Specific HCI Commands

The Silicon Labs HCI and Controller support the vendor-specific HCI commands listed in this section.

Table 3.1. HCI VS_SiliconLabs_Configure - command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Configure Configure various aspects in Silicon Labs Bluetooth Controller.	0x3f/0x07	key, length, data	status

Table 3.2. HCI VS_SiliconLabs_Configure - Parameters

Parameter	Size	Description
key	1	Configuration parameter key.
configuration data length	1	Length of the configuration data field.
data	0-255	Configuration data related to the configuration parameter (key).
status	1	Success (0x0), Invalid HCI Command Parameters (0x12), Unknown Advertising Identifier (0x42), Invalid LL Parameters (0x1E), Unsupported Feature or Parameter Value (0x11), Unspecified Error (0x1F)

Table 3.3. HCI VS_SiliconLabs_Configure - Parameter Key

Configuration parameter key [key value]	Parameters [size]	Description
CONFIG_KEY_HALT [1]	halt [1]	Halt (1) or resume (0) the radio
CONFIG_KEY_PRIORITY_RANGE [2]	rail_mapping_offset [2], rail_mapping_range[2]	Sets the RAIL priority_mapping offset field of the link layer priority configuration structure to the first byte of the value field. Used with multiprotocol. See UG305: Dynamic Multiprotocol User's Guide
CONFIG_KEY_SCAN_CHANNELS [3]	channel_map [1]	Set primary channels to be scanned. Only the three least significant bits are meaningful. 0x1 = Channel 37, 0x2 = Channel 38, 0x4 = Channel 39

Configuration parameter key [key value]	Parameters [size]	Description
CONFIG_KEY_SET_FLAGS [4]	flags [4]	Sets the link layer configuration flags. The value is a little endian 32-bit integer. Currently supported flag values: 0x00000001 - Disable Feature Exchange when slave 0x00000002 - Disable Feature Exchange when master 0x00000004 - Enable Completed Packets Event 0x00000008 - Enable Advertisement Channel Info 0x00000010 - Enable DCDC when configuring power 0x00000040 - Enable Raw IQ Sampling mode 0x00000080 – Disable 1M PHY 0x00000100 – Disable 2M PHY 0x00000200 – Disable Coded PHY 0x00000800 – Enable Even Connection Scheduling 0x00001000 – Enable Connection Power Control
CONFIG_KEY_CLR_FLAGS [4]	flags [4]	Clear the link layer configuration flags. The supported values are the same as with CONFIG_KEY_SET_FLAGS.
CONFIG_KEY_SET_AFH_INTERVAL [7]	scanning_interval [1]	Set the AFH scanning interval. The unit is 0.1 secs.
CONFIG_KEY_SET_PRIORITY_TABLE [9]	scan_min [1], scan_max [1], adv_min [1], adv_max [1], conn_min [1], conn_max [1], init_min [1], init_max [1], rail_mapping_offset [1], rail_mapping_range [1], reserved [1], adv_step [1], scan_step [1]	Configure link layer task priorities.

Configuration parameter key [key value]	Parameters [size]	Description
(1) CONFIG_KEY_SET_RX_PACKET_FILTERING [10]	filter_count [1], filter_offset [1], filter_length [1], filter_bitmask [1], filter_list [variable]	Enable and configure, or disable, RX packet filtering. **Filter_count:** number of template filters in the list. At most four filters can be configured. Setting the value 0 disables the feature, and all other parameters are ignored. **Filter_offset:** offset of the field in the received link layer packet where the filters and bitmask are applied. The offset 0 is the first octet after the *access address* field. **filterLength:** The length of the filters and bitmask in octets. All filters and bitmask must be equal in length. **Filter_bitmask:** Bitmask of F*ilter_length* octets used for filtering. The LSB must be the first byte. The bitmask must be given in the following format as a byte string: xx:xx:xx:xx:xx:xx ^- LSB MSB -^ The same bitmask is applied to all filter templates. **Filter_list:** Up to four filters, *Filter_length* octets each, used for filtering. The filters must be given the LSB first order. The filtering list must be given in the following format as a byte string: xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx ... ^- LSB MSB -^ ^- LSB MSB -^ ^- field1 -^ ^- field2 -^ ...
(1) CONFIG_KEY_SET_SIMULTANEOUS_SCANNING [11]	enable [1]	Enable (1) or disable (0) simultaneous 1M and Coded PHY scanning feature.
CONFIG_KEY_POWER_CONTROL_GOLDEN_RANGE [16]	min_1m [1], max_1m [1], min_2m [1], max_2m [1], min_coded_s8 [1], max_coded_s8 [1], min_coded_s2 [1], max_coded_s2 [1]	Configure the golden RSSI range for each PHY. For each PHY *min* must be less than *max.*

(1) Supported only by EFR32XG22/24 devices.

Table 3.4. HCI_VS_SiliconLabs_Get_Counters - Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Get_Counters Read radio counters.	0x3f/0x12	reset	status

Table 3.5. HCI_VS_SiliconLabs_Get_Counters - Command Parameters

Parameter	Size	Description
reset	1	Reset counters after reading them 1 – yes, 0 – no.
status	1	Success (0x0)
tx_packets	2	Number of transmitted radio packets.
rx_packets	2	Number of received radio packets.
crc_errors	2	Number of received packets detected with a CRC error.
failures	2	Number of radio failures, indicating errors in radio resource scheduling.

Table 3.6. HCI_VS_SiliconLabs_Set_Min_Max_TX_Power - Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Set_Min_Max_TX_Power Set minimum and maximum TX power levels.	0x3f/0x14	min_tx_power, max_tx_power	status

Table 3.7. HCI_VS_SiliconLabs_Set_Min_Max_TX_Power - Command Parameters

Parameter	Size	Description
min_tx_power	2	Minimum TX power to be used. The unit is in deci-dBm and the value must be within the range min_supported_tx_power--max_supported_tx_power. See *HCI_VS_SiliconLabs_Read_Current_TX_Power_Configuration*
max_tx_power	2	Maximum TX power to be used. The unit is in deci-dBm and the value must be within the range min_supported_tx_power--max_supported_tx_power. See *HCI_VS_SiliconLabs_Read_Current_TX_Power_Configuration*
status	1	Success (0x0), Unspecified Error (0x1F)

Table 3.8. HCI_VS_SiliconLabs_Set_Cte_Transmit_Enable - Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Set_Cte_Transmit_Enable Enable CTE transmission.	0x3f/0x15	advertising_handle, cte_enable, cte_length, cte_type, cte_count, switching_pattern_len, antenna_ids	status

Table 3.9. HCI_VS_SiliconLabs_Set_Cte_Transmit_Enable - Command Parameters

Parameter	Size	Description
advertising_handle	1	Handle of the advertiser used for CTE transmission.
cte_enable	1	Enable (1) or disable (0) CTE transmission. If transmission is disabled, the remaining parameters can be omitted.
cte_length	1	Length of the CTE. Valid range 0x2 – 0x14.
cte_type	1	Type of the CTE (0x0 or 0x1).
cte_count	1	CTE count. Valid range 0x1 – 0x10.
switching_pattern_length	1	Length of the switching pattern.
antenna_ids	variable	Antenna identifiers for CTE transmission (number of IDs must equal switching_pattern_length).
status	1	Success (0x0), Memory Capacity Exceeded (0x7), Unknown Advertising Identifier (0x42), Invalid HCI Command Parameters (0x12), Unsupported Feature or Parameter Value (0x11)

Table 3.10. HCI_VS_SiliconLabs_Set_Iq_Sampling_Enable - Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Set_Iq_Sampling_Enable Enable SiliconLabs proprietary IQ sampling. For further information refer to the following documents: UG103.18: Bluetooth® Direction-Finding Fundamentals QSG175: Silicon Labs Direction-Finding Solution Quick-Start Guide AN1296: Application Development with Silicon Labs' RTL Library	0x3f/0x16	sampling_enable, slot_durations, max_sampled_ctes, switching_pattern_len, antenna_ids	status

Table 3.11. HCI_VS_SiliconLabs_Set_Iq_Sampling_Enable - Command Parameters

Parameter	Size	Description
sampling_enable	1	Enable (1) or disable (0) IQ sampling. If sampling is disabled, the remaining parameters can be omitted.
slot_durations	1	CTE slot durations.
max_sampled_ctes	1	Currently always 0.
switching_pattern_length	1	Length of the switching pattern.
antenna_ids	variable	Antenna identifiers for IQ sampling (number of IDs must equal switching_pattern_length).
status	1	Success (0x0), Memory Capacity Exceeded (0x7), Invalid HCI Command Parameters (0x12), Unsupported Feature or Parameter Value (0x11)

Table 3.12. HCI_VS_SiliconLabs_Read_Current_TX_Power_Configuration - Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Read_Current_TX_Power_Configuration Read the TX power range supported by the radio, and the current TX power configuration.	0x3f/0x17	-	Status, min_supported_tx_power, max_supported_tx_power, min_configured_tx_power, max_configured_tx_power, tx_rf_path_compensation

Table 3.13. HCI_VS_SiliconLabs_Read_Current_TX_Power_Configuration - Command Parameters

Parameter	Size	Description
status	1	Success (0x0)
min_supported_tx_power	2	Minimum TX power supported by the radio. The unit is deci-dBm.
max_support_tx_power	2	Maximum TX power supported by the radio. The unit is deci-dBm
min_configured_tx_power	2	Minimum TX power configured to be used. The unit is in deci-dBm and value must be within the range min_supported_tx_power--max_supported_tx_power.
max_configured_tx_power	2	Maximum TX power configured to be used. The unit is in deci-dBm and value must be within the range min_supported_tx_power--max_supported_tx_power.
tx_rf_path_compensation	2	Currently configured TX RF path compensation in deci-dBms.

Table 3.14. HCI_VS_SiliconLabs_Enter_Bootloader_Mode - Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Enter_Bootloader_Mode Set Controller to bootloader mode, for example for firmware update purposes. *Note: the controller does not reply with a Command Complete event.*	0x3f/0x18	-	-

3.2 Vendor-Specific HCI Events

The Silicon Labs HCI and Controller support the vendor-specific HCI events listed in this section.

Table 3.15. HCI_VS_SiliconLabs_IQ_Report - Event

Event	Event Value	Event Parameters
HCI_VS_SiliconLabs_IQ_Report Receive SiliconLabs proprietary IQ sampling reports. For further information refer to the following documents: UG103.18: Bluetooth® Direction Finding Fundamentals QSG175: Silicon Labs Direction Finding Solution Quick-Start Guide AN1296: Application Development with Silicon Labs' RTL Library	0xfe	address_type, address, rx_phy, channel_index, rssi, rssi_antenna_id, cte_type, slot_durations, packet_status, packet_counter, sample_count, sample

Table 3.16. HCI_VS_SiliconLabs_IQ_Report - Event Parameters

Parameter	Size	Description
address_type	1	Bluetooth address type
address	6	Bluetooth address
rx_phy	1	Used PHY
channel_index	1	Channel index for the report.
rssi	1	RSSI
rssi_antenna_id	1	ID of the antenna where the samples are collected.
cte_type	1	CTE type
slot_durations	1	Slot duration
packet_status	1	Status of received packets
packet_counter	2	Number of received packets
sample_count	1	Number of samples
sample	variable	IQ samples

4 Enabling Hardware Flow Control In the WSTK

Hardware flow control can be enabled between the UART controller of the WSTK and Silicon Labs SoC. In the SoC, hardware flow control can be enabled with the configuration parameter shown in Section 2.1, Using an Example Application. In the WSTK the hardware flow control can be enabled as described in this section.

Important: if the hardware flow control settings are not the same in the SoC and WSTK, the HCI will not work.

1. Open Simplicity Studio and, in the Debug Adapters view, right-click the target device.
2. Select **Connect**.
3. Right-click the device again and select **Launch Console**.
4. Select the admin tab.
5. Set flow control with the following command:

```
WSTK> serial vcom config handshake rtscts
RTS handshake enabled
CTS handshake enabled
Serial configuration saved
```

6. Check the configuration with the following command:

```
WSTK> serial vcom
----- Virtual COM port -----
Stored port speed  : 115200
Active port speed  : 115226
Stored handshake   : rtscts
Actual handshake   : rtscts
RTS Asserted - Ready to Receive.
```

The flow can be disabled by setting *handshake* parameter to *none* in step three above.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com