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 referred to as RCP (Radio Co-Processor).

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.

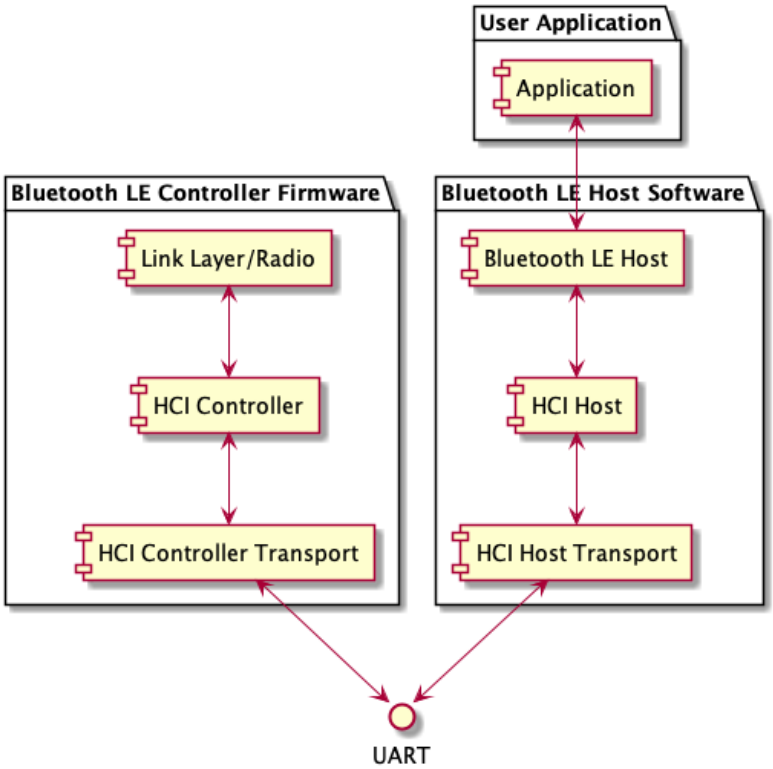


Figure 1.1. Bluetooth High-Level Components

The lowest layer between the host and controller communication is the HCI Transport. Currently, the RCP (Radio Co-Processor) sample applications support the HCI Three-Wire UART, or UART (Universal Asynchronous Receiver-Transmitter) as the HCI transport layer using either:

- Bluetooth SIG's UART (H4) transport protocol or
- Silicon Labs' proprietary CPC (Co-Processor Communication) protocol.

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.

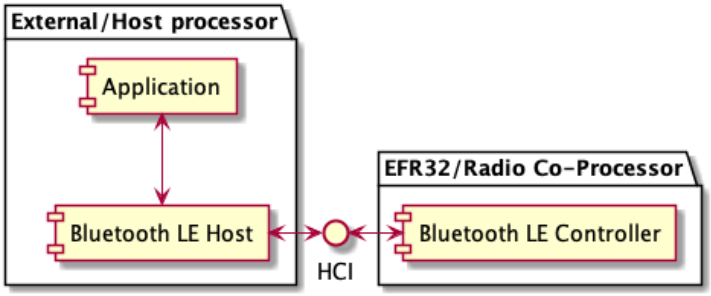


Figure 1.2. RCP Mode

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.

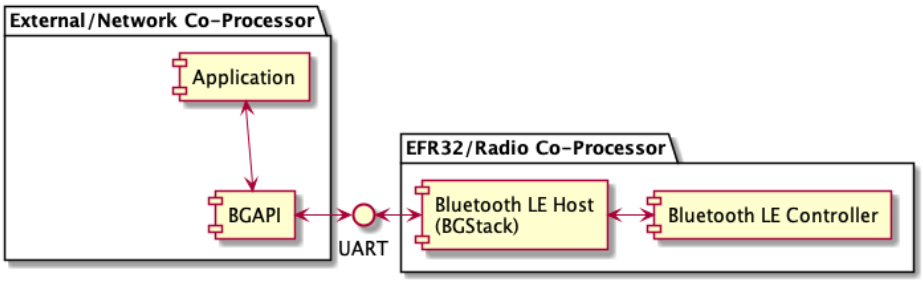


Figure 1.3. NCP Mode

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.

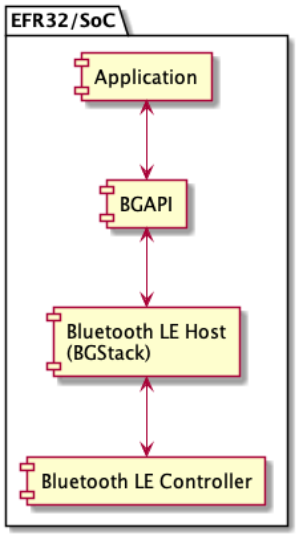


Figure 1.4. SoC Mode

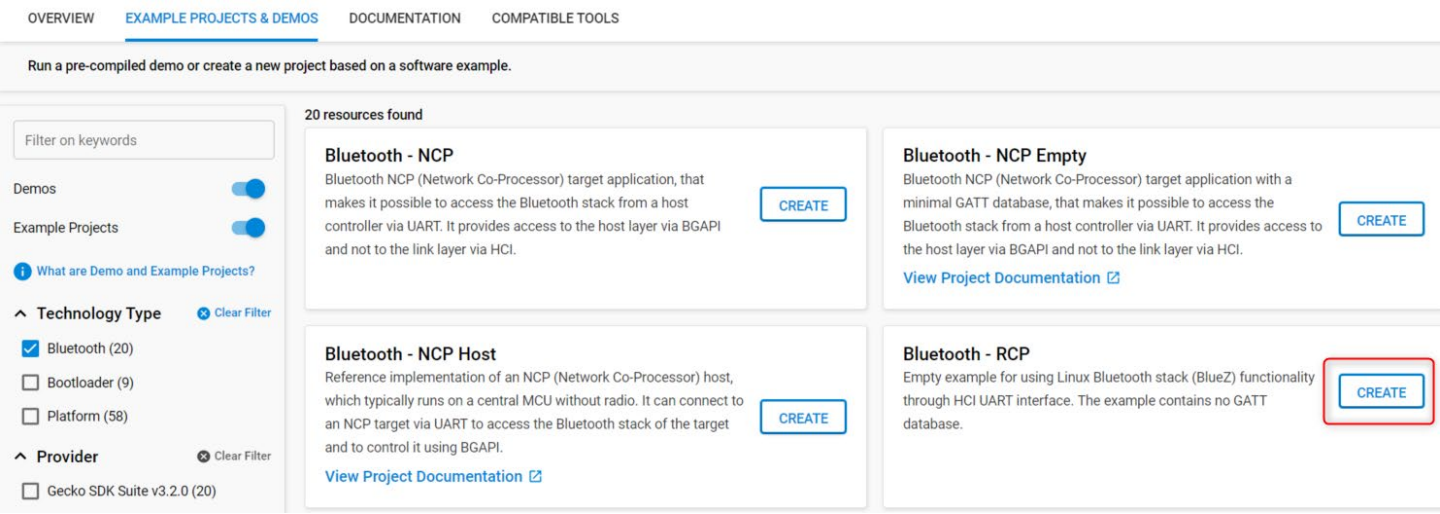
2 Enabling HCI Functionality

2.1 Using an Example Application

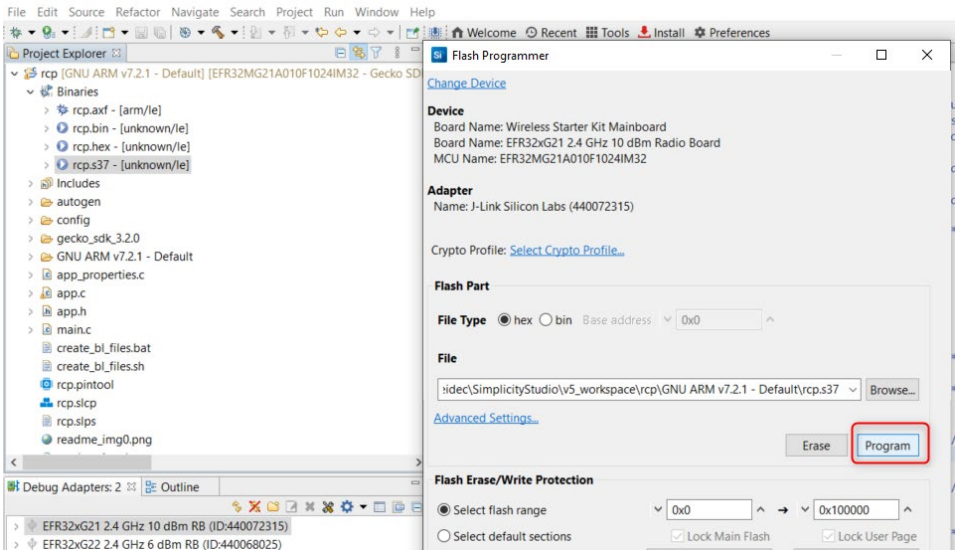
The Gecko SDK suite includes the **Bluetooth – RCP** and **Bluetooth – RCP CPC** example applications, which can be used to create an HCI-capable Bluetooth Controller application that can be run in RCP mode. An external Bluetooth Host protocol stack, such as Linux Bluez, can use the Bluetooth Controller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

Bluetooth – RCP creates an RCP project that uses Bluetooth SIG's UART (H4) transport protocol as the transport protocol.

Bluetooth – RCP CPC creates a project that uses Silicon Labs' proprietary CPC (Co-Processor Communication) protocol as the transport protocol. The latter one is useful in DMP (Dynamic Multi-protocol) use cases, when the host device communicates with multiple protocol stacks running on the Radio Co-Processor, but it may also be useful if you need a robust and secure transport protocol. For more information on CPC, refer to https://github.com/SiliconLabs/cpc-daemon/blob/main/readme.md. To learn more about the DMP use case, see *AN1333: Running Zigbee, OpenThread, and Bluetooth Concurrently on a Linux Host with a Multiprotocol RCP*.



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



Finally, attach the Bluetooth controller to your Bluetooth host stack using hciattach, and reset the device before issuing commands.

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.

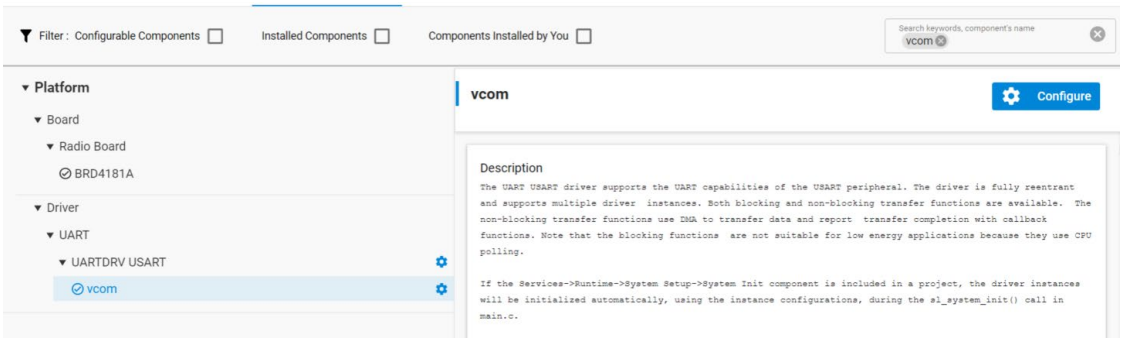
2.2 Transport layer configuration

2.2.1 UART

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

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

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



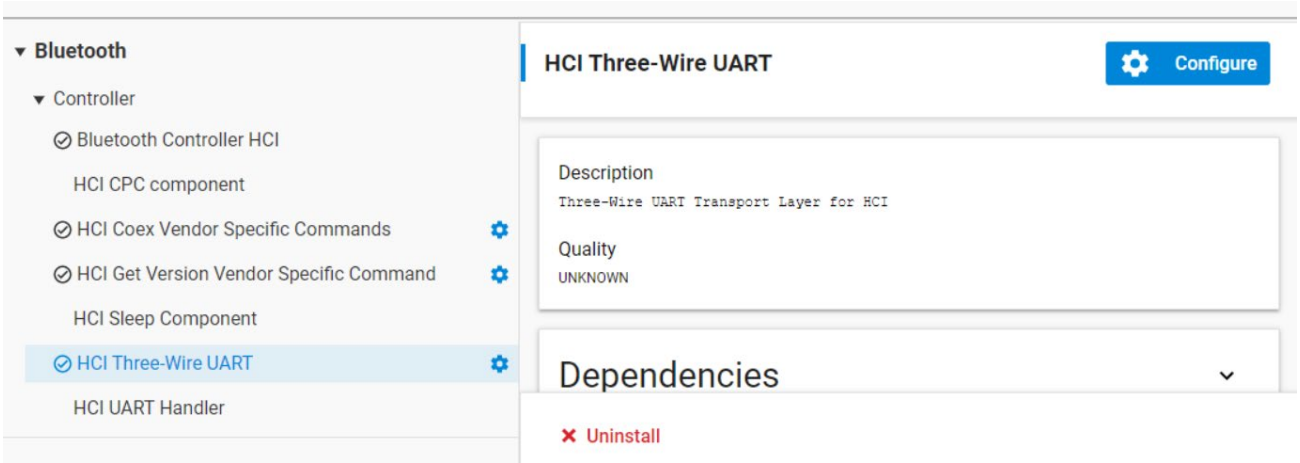
This will generate the values to the header file *sl_uartdrv_usart_vcom_config.h*. Especially for high-speed communication over UART, enabling hardware flow control is recommended. If it needs to be enabled, instructions for setting the hardware flow control in the WSTKs (Wireless Starter Kits) are given in Section 5, Configuring 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*.

2.2.2 Three-Wire UART

Note: To use the Three-Wire UART transport layer (H5) for transmitting HCI messages instead of the default UART transport layer (H4), add the HCI Three-Wire UART software component to the project. This will add framing, software flow control, and data integrity check to the transport layer, making the communication via UART much more reliable.

To enable the Three-Wire UART transport layer instead of the UART, add the *HCI Three-Wire UART* software component to the project:

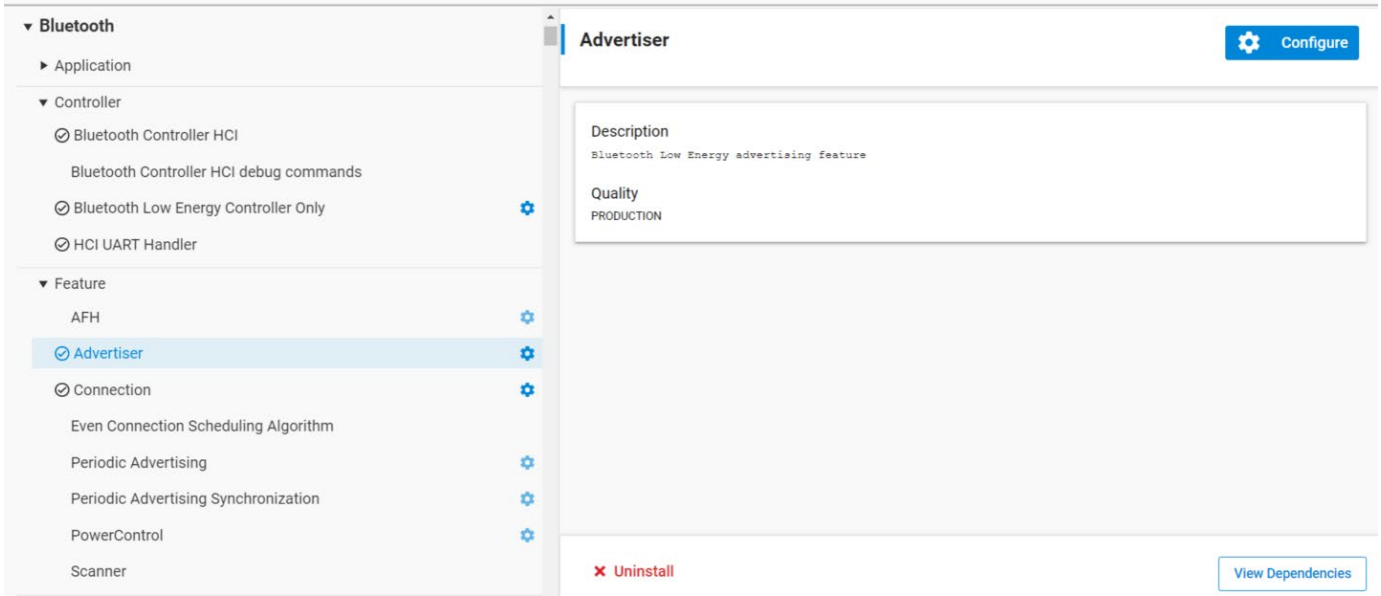


2.3 Required Components for HCI Capable RCP Mode Application

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 (if H4 UART transport protocol is used)
- HCI CPC component (if CPC transport protocol is used)

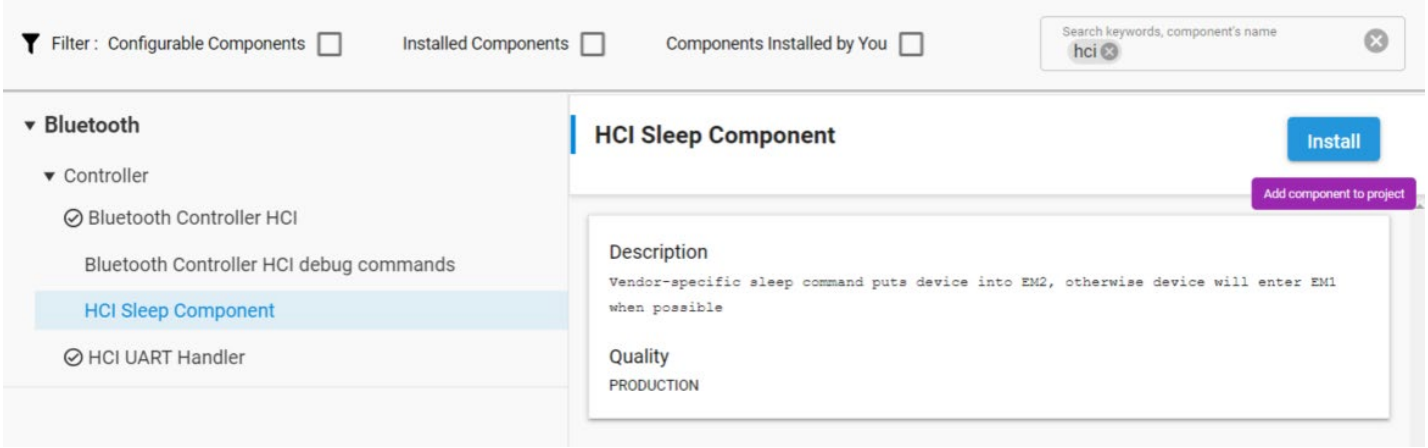
Other components are required to make a reasonably functional application. For example, the **Connection** and **Advertiser** components are required (and installed by default) to make an application that can advertise and accept connections. If the application should be able to accept a large number 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 Sleep Modes

In RCP mode two sleep modes can be activated, EM1 and EM2. To do so, install the **HCI Sleep Component**:



The device enters EM1 when it is idle and wakes from EM1 when data is received via USART.

The device enters EM2 on receiving the `HCI_VS_Silabs_Sleep` vendor-specific command, with the sleep bit set to 1. It handles all outstanding events before entering EM2.

An interrupt triggered by received GPIO data wakes the device from EM2. The first byte wakes the device but is discarded. This means that a normal HCI command cannot be used to wake a device from sleep. The recommended way is to send a “0”. If the first byte (Packet Type) is “0”, it will be discarded in other energy modes as well.

All Bluetooth activities operate normally during sleep except HCI interface reception, which is disabled during EM2.

If a host processor must be put to sleep and events prevented from being sent over the HCI interface, then it is recommended to only have Bluetooth activities operating that do not generate any events. For example, using non-connectable advertising or periodic advertising would not generate any events over the HCI. Note that advertising must not have a timeout and must not report scan requests.

Events can also be disabled by using HCI event mask commands. Depending on the masked events, these are handled by the link layer itself. See the Bluetooth Core specification for details.

4 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. Additional vendor-specific commands can also be implemented, as described in section 4.3 Custom Commands.

4.1 Vendor-Specific HCI Commands

The Silicon Labs HCI and Controller support the following vendor-specific HCI commands.

Table 4.1. HCI VS_SiliconLabs_Forcefully_Kill_Connection – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Forcefully_Kill_Connection	0x3f/0x04	handle	status

Table 4.2. HCI VS_SiliconLabs_Forcefully_Kill_Connection – Command Parameters

Parameter	Size	Description
handle	2	Connection handle
status	1	Success (0x0), Unknown Connection Identifier (0x02)

Table 4.3. HCI_VS_SiliconLabs_Set_Connection_Config_Bits – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Set_Connection_Config_Bits Configure various parameters of the Bluetooth connection.	0x3f/0x05	handle config_bits	status

Table 4.4. HCI_VS_SiliconLabs_Set_Connection_Config_Bits – Command Parameters

Parameter	Size	Description
handle	2	Connection handle
config_bits	4	Configure the connection. Supported values: 0x01 – Disable peripheral latency. Peripheral will ignore peripheral latency and treat it as being 0. 0x02 – Disable 1M PHY 0x04 – Disable 2M PHY 0x08 – Disable Coded PHY These affect PHY-update procedure to allow selecting certain PHYs if remote end indicates support for multiple PHYs.
status	1	Success (0x0), Unknown Connection Identifier (0x02)

Table 4.5. HCI_VS_SiliconLabs_Clear_Connection_Config_Bits – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Clear_Connection_Config_Bits	0x3f/0x06	handle config_bits	status

Table 4.6. HCI_VS_SiliconLabs_Clear_Connection_Config_Bits – Command Parameters

Parameter	Size	Description
handle	2	Connection handle
config_bits	4	Clears the connection configuration. The supported values are the same as the *config_bits* in HCI_VS_SiliconLabs_Set_Connection_Config_Bits command.
status	1	Success (0x0), Unknown Connection Identifier (0x02)

Table 4.7. 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 4.8. HCI_VS_SiliconLabs_Configure – Command 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 4.9. 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 0x00000400 – Enable Host Session Key Generation 0x00000800 – Enable Even Connection Scheduling 0x00001000 – Enable PAwR Connection Scheduling 0x00002000 – Enable Connection Power Control 0x00004000 – Disable Connection Duplicate Address Check 0x00008000 – Disable Auto Data Length Update
CONFIG_KEY_CLR_FLAGS [5]	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_PERIODIC_ADV_STATUS_EVENT [8]	handle [1], enable [1]	**handle**: Advertising handle **enable**: Enable (1) or disable (0) status event on the sync advertiser
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_SET_CHANNELMAP_FLAGS [12]	flags [4]	Configure channel map flags. Supported flag values: 0x01 – Enable adaptivity Note: AFH must be enabled to enable adaptivity.
CONFIG_KEY_POWER_CONTROL_GOLDEN_RANGE [16]	golden_rssi_min_1m [1], golden_rssi_max_1m [1], golden_rssi_min_2m [1], golden_rssi_max_2m [1], golden_rssi_min_coded_s8 [1], golden_rssi_max_coded_s8 [1], golden_rssi_min_coded_s2 [1] golden_rssi_max_coded_s2 [1]	Configure the golden range values of the power control feature

Configuration parameter key [key value]	Parameters [size]	Description
CONFIG_KEY_ACTIVE_SCANNER_BACKOFF_UPPER_LIMIT [17]	backoff_upper_limit [2]	Sets a new maximum for the scanner backoff upper limit. This value is used to mitigate collisions between different scanners in a busy environment. Lower value implies a more aggressive scanner. Set to 0 for default value (256). Minimum value: 16 Maximum value: 256
CONFIG_KEY_AFH_RSSI_THRESHOLD [18]	threshold [1]	Configure the cutoff RSSI used to block channels. Default is -70 dBm.
CONFIG_KEY_AFH_CHANNEL_COOLDOWN [19]	cooldown [2]	Configure the cooldown value when a channel is blocked. Default is 8 seconds.
CONFIG_KEY_SET_REPORT_ALL_SCAN_RSPS [20]	enable [1]	Enable (1) or disable (0) all received SCAN_RSP reporting

(1) Supported only by EFR32XG22/24 devices.

Table 4.10. HCI_VS_SiliconLabs_Get_Timing - Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Get_Timing	0x3f/0x08	timers clear_timers	status

Table 4.11. HCI_VS_SiliconLabs_Get_Timing – Command Parameters

Parameter	Size	Description
timers	1	Select timer. 0 – Start timestamp 1 – Maximum timer value 2 – Number of measurements performed 3 – Total accumulated time
clear_timers	1	0 – Do not clear timers 1 – Clear timers
status	1	Success (0x0) Unsupported Feature or Parameter Value (0x11)

Table 4.12. HCI_VS_SiliconLabs_Config_Flags – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Config_Flags Configure the link layer configuration flags.	0x3f/0x11	set clr read	flags status

Table 4.13. HCI_VS_SiliconLabs_Config_Flags – Command Parameters

Parameter	Size	Description
set	4	Enable the link layer flags. Supported values are listed in Table 4.9, CONFIG_KEY_SET_FLAGS.
clr	4	Clear the link layer flags.
read	4	Return link layer configuration flags with bitmask applied as determined by this field.
flags	4	Return the flags.
status	1	Success (0x0) Unsupported Feature or Parameter Value (0x11)

Table 4.14. 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 4.15. 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 4.16. HCI_VS_Silabs_Sleep – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_Silabs_Sleep Enter EM2 sleep mode	0x3f/0x13	sleep	status sleep

Table 4.17. HCI_VS_Silabs_Sleep – Command Parameters

Parameter	Size	Description
sleep	1	Set to 1 to enter sleep mode. Returns 0 if sleep request unsuccessful.
Status	1	Success (0x0), Unsupported Feature or Parameter Value (0x11)

Table 4.18. HCI_VS_Silabs_Set_Min_Max_TX_Power – Command

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

Table 4.19. HCI_VS_Silabs_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 4.20. HCI_VS_Silabs_Set_Cte_Transmit_Enable – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_Silabs_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 4.21. HCI_VS_Silabs_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 4.22. HCI_VS_Silabs_Set_Iq_Sampling_Enable – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_Silabs_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 4.23. HCI_VS_Silabs_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 4.24. HCI_VS_Silabs_Read_Current_TX_Power_Configuration – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_Silabs_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 4.25. HCI_VS_Silabs_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 4.26. HCI_VS_Silabs_Enter_Bootloader_Mode – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_Silabs_Enter_Bootloader_Mode Set Controller to bootloader mode and reset it, for example for firmware update purposes.	0x3f/0x18	-	-

Note: the controller does not reply with a Command Complete event.

Table 4.27. HCI_VS_SiliconLabs_Set_Advertising_Config_Bits – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Set_Advertising_Config_Bits	0x3f/0x19	advertising_handle config_bits	status

Table 4.28. HCI_VS_SiliconLabs_Set_Advertising_Config_Bits – Command Parameters

Parameter	Size	Description
advertising_handle	1	Advertising handle
config_bits	4	0x00000001 – Force public address usage in advertising packets.
Status	1	Success (0x0), Unknown Advertising Identifier (0x42)

Table 4.29. HCI_VS_SiliconLabs_Clear_Advertising_Config_Bits – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_ VS_SiliconLabs_Clear_Advertising_Config_Bits	0x3f/0x1a	advertising_handle config_bits	-

Table 4.30. HCI_VS_SiliconLabs_Clear_Advertising_Config_Bits – Command Parameters

Parameter	Size	Description
advertising_handle	1	Advertising handle
config_bits	4	0x00000001 – Remove requirement of public address usage in advertising packets.
Status	1	Success (0x0), Unknown Advertising Identifier (0x42)

Table 4.31. HCI_VS_SiliconLabs_Set_Max_Low_Tx_Power – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Set_Max_Low_Tx_Power Set the maximum power in low-power mode for each PHY.	0x3f/0x1b	max_1m_low_power max_2m_low_power max_125k_low_power max_500k_low_power	-

Table 4.32. HCI_VS_SiliconLabs_Set_Max_Low_Tx_Power – Command Parameters

Parameter	Size	Description
max_1m_low_power	2	The maximum power in low-power mode (DTS mode) for 1M PHY set to a level allowed by the region.
max_2m_low_power	2	The maximum power in low-power mode (DTS mode) for 2M PHY set to a level allowed by the region.
max_125k_low_power	2	The maximum power in low-power mode (DTS mode) for 125k PHY set to a level allowed by the region.
max_500k_low_power	2	The maximum power in low-power mode (DTS mode) for 500k PHY set to a level allowed by the region.

Table 4.33. HCI_VS_SiliconLabs_Allocate_Connections – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Allocate_Connections Allocates memory for connection descriptors.	0x3f/0x20	num_connections	-

Table 4.34. HCI_VS_SiliconLabs_Allocate_Connections – Command Parameter

Parameter	Size	Description
num_connections	1	The number of connections for which memory will be allocated.

Table 4.35. HCI_VS_SiliconLabs_Allocate_Advertisers – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Allocate_Advertisers Allocates memory for advertiser descriptors.	0x3f/0x21	num_advertisers	-

Table 4.36. HCI_VS_SiliconLabs_Allocate_Advertisers – Command Parameter

Parameter	Size	Description
num_advertisers	1	The number of advertisers for which memory will be allocated.

Table 4.37. HCI_VS_SiliconLabs_Allocate_Addresses – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Allocate_Addresses Allocates memory for stored addresses.	0x3f/0x22	num_addresses	-

Table 4.38. HCI_VS_SiliconLabs_Allocate_Addresses – Command Parameter

Parameter	Size	Description
num_addresses	1	The number of addresses for which memory will be allocated.

Table 4.39. HCI_VS_SiliconLabs_Allocate_PeriodicAdv – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Allocate_PeriodicAdv Allocates memory for periodic advertiser descriptors.	0x3f/0x23	num_periodicadv	-

Table 4.40. HCI_VS_SiliconLabs_Allocate_PeriodicAdv – Command Parameter

Parameter	Size	Description
num_periodicadv	1	The number of periodic advertisers for which memory will be allocated.

Table 4.41. HCI_VS_SiliconLabs_Allocate_PeriodicScan – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Allocate_PeriodicScan Allocates memory for periodic advertisement synchronization descriptors.	0x3f/0x24	num_periodicscan	-

Table 4.42. HCI_VS_SiliconLabs_Allocate_PeriodicScan – Command Parameter

Parameter	Size	Description
num_periodicscan	1	The number of periodic advertisement synchronizations for which memory will be allocated.

Table 4.43. HCI_VS_SiliconLabs_Deinit – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Deinit De-initializes the Bluetooth Controller and frees up allocated memory.	0x3f/0x25	-	-

Table 4.44. HCI_VS_SiliconLabs_MemBufResize – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_MemBufResize Allocates memory for periodic advertisement synchronization descriptors.	0x3f/0x26	buffer_memory	status buffers_allocated

Table 4.45. HCI_VS_SiliconLabs_MemBufResize – Command Parameters

Parameter	Size	Description
buffer_memory	4	Size of memory to allocate
status	1	Success (0x0), Unspecified Error (0x1F)
buffers_allocated	4	Number of memory buffers allocated

Table 4.46. HCI_VS_SiliconLabs_ExtScanPHYsAllowed – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_ExtScanPHYsAllowed Sets the used PHYs used for scanning for external advertisements, Aux Pointer will not be followed unless it has one of the allowed PHYs. If a given PHY is not supported by the device, bt_err_invalid_command_parameters will be returned	0x3f/0x27	phys	status

Table 4.47. HCI_VS_SiliconLabs_ExtScanPHYsAllowed – Command Parameters

Parameter	Size	Description
phys	1	Bitfield: ll_phy_1M = 1 ll_phy_2M = 2, ll_phy_Coded = 4
status	1	Success (0x0), Invalid HCI Command Parameters (0x12)

Table 4.48. HCI_VS_SiliconLabs_Set_Public_Address – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Set_Public_Address	0x3f/0x28	bdaddr[6]	status

Table 4.49. HCI_VS_SiliconLabs_Set_Public_Address – Command Parameters

Parameter	Size	Description
bdaddr[6]	1	Address to set the public address. If NULL, device unique address is used.
status	1	Success (0x0)

Table 4.50. HCI_VS_SiliconLabs_Periodic_Advertising_Update_Sync_Parameters – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Periodic_Advertising_Update_Sync_Parameters	0x3f/0x29	sync_handle skip sync_timeout	status

Table 4.51. HCI_VS_SiliconLabs_Periodic_Advertising_Update_Sync_Parameters – Command Parameters

Parameter	Size	Description
sync_handle	2	Periodic advertising sync handle
skip	2	New value for the skip parameter
sync_timeout	2	New timeout value
status	1	Success (0x0), Invalid HCI Command Parameters (0x12), Command Disallowed (0xC), Unknown Advertising Identifier (0x42)

Table 4.52. HCI_VS_SiliconLabs_Get_Conn_Params – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Get_Conn_Params Get the Bluetooth connection parameters.	0x3f/0x2a	handle	status interval access_address, crc_init. start_time, event_counter, channel, flags, hop, central_phy, peripheral_phy, channel_map[5], central_sca supervision_timeout

Table 4.53. HCI_VS_SiliconLabs_Get_Conn_Params – Command Parameters

Parameter	Size	Description
handle	2	Connection handle
status	1	Success (0x0), Command Disallowed (0xC)
interval	2	Connection interval
access_address	4	Access Address Field of packet
crc_init	4	Value used to initialize the CRC algorithm
start_time	4	Start time of the connection task
event_counter	2	Number of connection events
channel	1	Channel on which next event will occur
flags	1	Bit 1 set if Channel Selection Algorithm #2 in use
hop	1	Channel hop value
central_phy	1	PHY in use on Central device
peripheral_phy	1	PHY in use on Peripheral device
channel_map[5]	1	Bitmap of valid channels
Central_sca	1	Sleep clock accuracy
Supervision_timeout	2	Connection supervision timeout configured for the connection

Table 4.54. HCI_VS_SiliconLabs_Allocate_ResolvingList – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Allocate_ResolvingList Allocates memory for the resolving list.	0x3f/0x2c	entries	-

Table 4.55. HCI_VS_SiliconLabs_Set_CS_Antenna_Config – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Set_CS_Antenna_Config Set the distance offset of antenna(s) for channel sounding.	0x3f/0x2d	num_antenna_elements antenna_element_offset[]	status

Table 4.56. HCI_VS_SiliconLabs_Set_CS_Antenna_Config – Command Parameters

Parameter	Size	Description
num_antenna_elements	1	Number of antennas
antenna_element_offset[]	2	Antenna offset in cm. Array size is length of antenna elements.
status	1	Success (0x0), Unspecified Error (0x1F)

Table 4.57. HCI_VS_SiliconLabs_Allocate_PawrAdv – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Allocate_PawrAdv Allocates memory for PAwR advertisers.	0x3f/0x2e	num_pawradv	status

Table 4.58. HCI_VS_SiliconLabs_Allocate_PawrAdv – Command Parameter

Parameter	Size	Description
num_pawradv	1	Allocate memory to the specified number of PAwR advertisers
status	1	Success (0x0), No More Resource (0x1A)

Table 4.59. HCI_VS_SiliconLabs_Allocate_PawrSync – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Allocate_PawrSync Allocates memory for PAwR synchronizers.	0x3f/0x2f	num_pawrsync	status

Table 4.60. HCI_VS_SiliconLabs_Allocate_PawrSync – Command Parameter

Parameter	Size	Description
num_pawrsync	1	Allocate memory to the specified number of PAwR synchronizers.
Status	1	Success (0x0), Memory Capacity Exceeded (0x7)

Table 4.61. HCI_VS_Siliconlabs_Set_Connection_Tx_Power – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_Siliconlabs_Set_Connection_Tx_Power	0x3f/0x30	handle tx_power	tx_power

Table 4.62. HCI_VS_Siliconlabs_Set_Connection_Tx_Power – Command Parameters

Parameter	Size	Description
handle	2	Connection handle
Tx_power	2	TX power in deci-dBm
status	1	Success (0x0), Unknown Connection Identifier (0x02)

Table 4.63. HCI_VS_SiliconLabs_Read_Connection_Statistics – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Read_Connection_Statistics	0x3f/0x31	handle reset	-

Table 4.64. HCI_VS_SiliconLabs_Read_Connection_Statistics – Commands Parameters

Parameter	Size	Description
handle	2	Connection handle
reset	1	Reset the connection parameters

Table 4.65. HCI_VS_SiliconLabs_Sniff_Connection_Packets – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Sniff_Connection_Packets Sniff Bluetooth LE connection packets based on the provided connection settings.	0x3f/0x32	interval access_address, crc_init. start_time, event_counter, channel, options, hop, central_phy, peripheral_phy, channel_map[5], central_sca supervision_timeout	sniffer_handle

Table 4.66. HCI_VS_SiliconLabs_Sniff_Connection_Packets – Command Parameters

Parameter	Size	Description
interval	2	Connection interval
access_address	4	Access Address Field of packet
crc_init	4	Value used to initialize the CRC algorithm
start_time	4	Start time of the connection task
event_counter	2	Number of connection events
channel	1	Channel on which next event will occur
options	1	Bit 1 – Channel Selection Algorithm #2 in use Bit 2 – Ignore start_time parameter
hop	1	Channel hop value
central_phy	1	PHY in use on Central device
peripheral_phy	1	PHY in use on Peripheral device
channel_map[5]	1	Bitmap of valid channels
central_sca	1	Sleep clock accuracy
supervision_timeout	2	Connection supervision timeout
sniffer_handle	1	Index of sniffer

Table 4.67. HCI_VS_SiliconLabs_Get_Stack_Space – Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Get_Stack_Space	0x3f/0x33	-	transport_thr_stack_space linklayer_thr_stack_space status

Table 4.68. HCI_VS_SiliconLabs_Get_Stack_Space – Command Parameters

Parameter	Size	Description
transport_thr_stack_space	4	Stack space of transport thread in a RTOS
linklayer_thr_stack_space	4	Stack space of link layer thread in a RTOS
status	1	Success (0x0), Command Disallowed (0xC)

Table 4.69. HCI_VS_SiliconLabs_Stop_Sniff_Connection_Packets - Command

Command	Command Value (OGF/OCF)	Command Parameters	Return Parameters
HCI_VS_SiliconLabs_Stop_Sniff_Connection_Packets	0x3f/0x34	sniffer_handle	status

Table 4.70. HCI_VS_SiliconLabs_Stop_Sniff_Connection_Packets – Command Parameters

Parameter	Size	Description
sniffer_handle	1	Index of sniffer to stop
status	1	Success (0x0), Invalid HCI Command Parameters (0x12),

4.2 Vendor-Specific HCI Events

The Silicon Labs HCI and Controller support the following vendor-specific HCI events.

Note: The vendor-specific events are wrapped inside the LE_Meta_Event (0x3e). The subevent field contains the event number.

Table 4.71. HCI_Event_Le_Silabs_IQ_Report – Event

Event	Event Value	Event Parameters
HCI_Event_Le_Silabs_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	0x3e	subevent_code address_type, address, rx_phy, channel_index, rssi, rssi_antenna_id, cte_type, slot_durations, packet_status, packet_counter, sample_count, sample

Table 4.72. HCI_Event_Le_Silabs_IQ_Report – Event Parameters

Parameter	Size	Description
subevent_code	1	Subevent where the IQ report is received
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

Table 4.73. HCI_Event_Le_Silabs_Sk_Request - Event

Event	Event Value	Event Parameters
HCI_Event_Le_Silabs_Sk_Request Request for host to generate session key.	0x3e	Subevent_code Handle Random Diversifier SKD

Table 4.74. HCI_Event_Le_Silabs_Sk_Request – Event Parameters

Parameter	Size	Description
Subevent_code	1	Subevent where the session key is requested
Handle	2	Connection handle
Random	1	Random data
Diversifier	2	Encryption diversifier
SKD	1	Session key diversifier

Table 4.75. HCI_Event_VS_SiliconLabs_Connection_Statistics – Event

Event	Event Value	Event Parameters
HCI_Event_VS_SiliconLabs_Connection_Statistics Statistics report of a connection.	0x3e	subevent_code handle rssi_min rssi_max events_total events_success events_missed crc_errors

Table 4.76. HCI_Event_VS_SiliconLabs_Connection_Statistics – Event Parameters

Parameter	Size	Description
subevent_code	1	Subevent where the connection statistics are reported
handle	2	Connection handle
rssi_min	1	Minimum RSSI received in connection packets. 0x7f if RSSI is unknown.
rssi_max	1	Maximum RSSI receiver in connection packets. -0x7f if RSSI is unknown.
events_total	4	Total number of connection events
events_success	4	Number of successful connection events
events_missed	4	Number of missed connection events
crc_errors	4	Increased when packet with bad crc is received

Table 4.77. HCI_Event_Le_Silabs_Sniff_Connection – Event

Event	Event Value	Event Parameters
HCI_Event_Le_Silabs_Sniff_Connection Report of the sniffed connection.	0x3e	subevent_code central_rssi peripheral_rssi flags sniffer_handle

Table 4.78. HCI_Event_Le_Silabs_Sniff_Connection – Event Parameters

Parameter	Size	Description
subevent_code	1	Subevent where a connection is sniffed
central_rssi	1	Central packet RSSI
peripheral_rssi	1	Peripheral packet RSSI
flags	1	Sniffer flags
sniffer_handle	1	Sniffer instance handle

Table 4.79. HCI_Event_Le_Silabs_Sniff_Complete– Event

Event	Event Value	Event Parameters
HCI_Event_Le_Silabs_Sniff_Complete Received when connection sniffing is complete.	0x3e	subevent_code sniffer_handle reason

Table 4.80. HCI_Event_Le_Silabs_Sniff_Complete– Event Parameters

Parameter	Size	Description
subevent_code	1	Subevent where the sniffing is complete
sniffer_handle	1	Index of sniffer
reason	1	Reason to stop

4.3 Custom Commands

On RCP builds it is possible to hook custom message handlers to the message processing. The Application needs to allocate the `sl_btctrl_command_handler_t` structure in the heap and implement a callback function. The stack adds this to the list of message handlers, and calls it during message processing. When the handler receives a message, if it does not handle it must return false. If the message is processed, the handler must return true. If no handler handles the HCI command, then a Command Complete event with 0x01 (Unknown HCI Command) is returned to the host.

Silicon Labs uses opcodes starting from 0xfc00 forward. To prevent collisions with these opcodes, use opcodes starting from 0xff00 in your applications.

4.3.1 Example

This example implements a simple vendor-specific HCI command with opcode 0xff00 that receives a 32-bit number, then returns the same number increased by one.

1. Include this header in the application file:

```
#include "sl_btctrl_hci_handler.h"
```

2. Allocate the structure in heap:

```
struct sl_btctrl_command_handler custom_handler;
```

3. Implement the callback function:

```
bool custom_message_handler(struct sl_btctrl_hci_message * msg)
{
  uint16_t opcode;
  if(sl_btctrl_hci_message_get_opcode(msg, &opcode) != SL_STATUS_OK){
      return false;
  }
  size_t length;
  if(sl_btctrl_hci_message_get_length(msg, &length) != SL_STATUS_OK){
      return false;
  }
```

```
  switch(opcode)
  {
    case 0xff00:{
      uint32_t params;
       if(sl_btctrl_hci_message_get_parameters(msg,     (uint8_t*)&params,     sizeof(params))     !=
SL_STATUS_OK){
         sl_btctrl_hci_message_set_response(msg, 0x12, NULL, 0);
         // 0x12 is "Invalid HCI Command Parameters"-error.
         // Set response only fails if NULL pointers or too long data is passed
      }else{
        params++;//Increase
        sl_btctrl_hci_message_set_response(msg, 0x0, (uint8_t*)&params, sizeof(params));
      }
    }
    return true;//Command is handled
}
return false;
```

4. Register handler in startup code:

```
sl_btctrl_hci_register_handler(&custom_handler, &custom_message_handler);
```

4.3.2 Custom Commands Provided as Software Components

Some Silicon Labs specific features are provided as software components. They implement custom command handling for different functionalities.

4.3.2.1 HCI Get Version

Add the software component **HCI Get Version Vendor Specific Command** to the project. The Opcode for this vendor specific command can be defined in the component. By default, it is 0xFF10.

Return parameters:

Parameter	Size	Description
major	2	The major version number
minor	2	The minor version number
patch	2	The patch version number
build	2	The build number of the version

The return parameter structure is defined in the file *sl_bt_hci_version_config.h*.

4.3.2.2 HCI Coex

Add the component **HCI Coex Vendor Specific Commands** to the project. The Opcodes for this vendor specific command can be defined in the component.

The provided commands have the same inputs and return parameters as the Coex commands defined in SoC projects: https://docs.si-labs.com/bluetooth/4.0/a00103

5 Configuring Hardware Flow Control In the WSTK

Hardware flow control can be enabled or disabled between the UART controller of the WSTK and Silicon Labs SoC. In the SoC, hardware flow control can be configured with the configuration parameter shown in Section 2.1, Using an Example Application. In the WSTK the hardware flow control can be configured as described in this section. The example below shows how to enable hardware flow control.

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

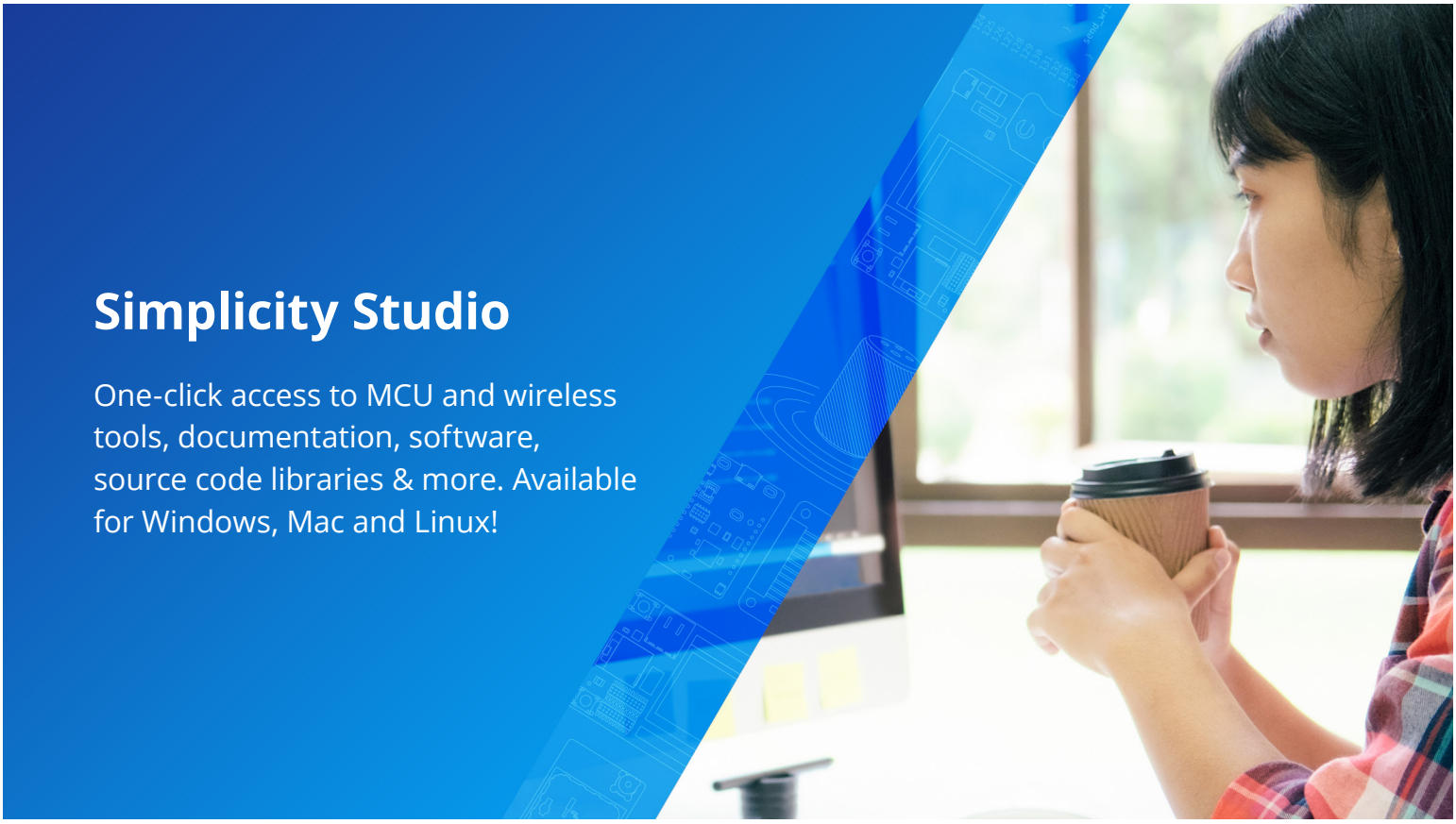
1. Open Simplicity Studio and, in the Debug Adapters view, right-click the target device.
2. Select **Connect**.
3. Right-click the device again and select **Launch Console**.
4. Select the admin tab.
5. Set flow control with the following command:

```
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 control can be disabled by setting *handshake* parameter to *none* in step 5 above.



IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

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

www.silabs.com