AIROC™ HCI UART Control Protocol

ModusToolbox™

About this document

Scope and purpose

This document provides information on the HCI UART Control Protocol, which is an implementation example of how a host MCU can communicate with an AIROC™ device via HCI UART.

Intended audience

This document is intended for application developers using a ModusToolbox™ Bluetooth® software development kit (BTSDK) to create and test designs based on AIROC™ Bluetooth® devices.

Table of contents

1 About this document

Several paragraphs in the document refer the reader to variables and data structures that are not described in this document. For information on the variables and data structures mentioned in this document, see the AIROC™ API reference guide for the Bluetooth® device you are using, available from the "AIROC™ Bluetooth® SDK Documentation" link in the ModusToolbox™ Quick Panel documentation section.

1.1 Acronyms and abbreviations

In most cases, acronyms and abbreviations are defined on first use. For a comprehensive list of acronyms and other terms used in the documents, go to the **Glossary**.

1.2 IoT resources and technical support

The wealth of data available **here** will help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. You can access a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. You can acquire technical documentation and software from the **Support Community website**.

1.3 Hardware and software prerequisites

To fully use the content provided in this document, readers will need the following items:

- One AIROC™ Bluetooth® System-on-Chip (SoC) device (CYW20706, CYW20719, CYW20721, CYW20735, CYW20819, CYW20820, CYW20835, or CYW43012 chip families, referred to as CYWxxxxx device in this document) and a second Bluetooth® device, which can also be based on a similar CYWxxxxx device.
- Version 1.1 or greater of ModusToolbox™, which includes several applications that use the HCI control protocol defined in this document.
- A PC running Windows 7 or higher, Mac OS X 10.10 or higher, Ubuntu Linux 16 or higher, or Fedora Linux 23 or higher.

To prepare a CYWxxxxx-based Bluetooth® device, build an application from ModusToolbox™ that uses the HCI control protocol defined in this document. For help doing such a build, see the AIROC™ kit guide for the CYWxxxxx device you are using, for example the CYW920819EVB-02 evaluation kit user guide **[1]**.

Note: *Throughout the document, references to the 'watch' application are applicable to any application running on the CYWxxxxx that supports the HCI control protocol defined in this document. Any sample application that calls the `wiced_transport_init()` API with a valid callback in the `wiced_transport_data_handler_t` member of the parameter struct supports the HCI Control Protocol defined in this document.*

2 Introduction

The ModusToolbox™ BSDK includes sample applications that can be executed on AIROC™ CYWxxxxx Bluetooth® devices.

A real Bluetooth® product could have an onboard MCU that uses CYWxxxxx device to provide Bluetooth® functionality. For such a product, MCU software would likely be used to control the device through a UART or SPI interface via a protocol that allows the MCU to send and receive commands, events, and data. This document describes a sample protocol for communication between an MCU and a CYWxxxxx device.

The CYWxxxxx devices support two operating modes: the Bluetooth® Host Controller Interface (HCI) mode and the application mode. In the Bluetooth® HCI mode, the embedded stack in the device is not exercised and the device behaves as a standard Bluetooth® HCI controller. A standard Bluetooth® HCI controller supports the Bluetooth® HCI interface as defined in the Bluetooth® core specification [2]. In the Application mode, the embedded stack in the CYWxxxxx device is used and the device does not behave as a standard Bluetooth® controller.

Figure 1 shows the Bluetooth® HCI mode and application mode logical interfaces. In the Bluetooth® HCI mode, the MCU communicates to the CYWxxxxx device using the standard Bluetooth® HCI protocol. In the application mode, the MCU uses the AIROC™ HCI protocol defined in this document.

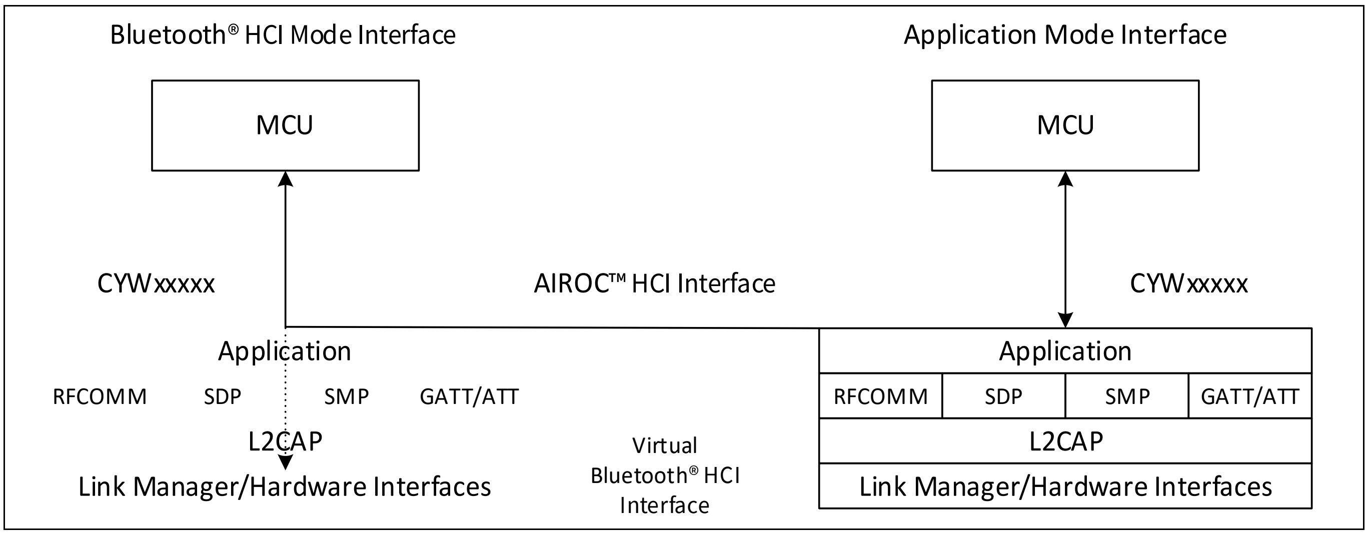
Bluetooth® HCI Mode Interface
Application Mode Interface
MCU
MCU
CYWxxxxx
AIROC™ HCI Interface
CYWxxxxx
Application
RFCOMM
SDP
SMP
GATT/ATT
L2CAP
Link Manager/Hardware Interfaces
Virtual Bluetooth® HCI Interface
Application
RFCOMM
SDP
SMP
GATT/ATT
L2CAP
Link Manager/Hardware Interfaces

Figure 1 CYWxxxxx MCU interfaces in the Bluetooth® HCI and application modes

This document provides a sample protocol, referred to as the AIROC™ HCI Control Protocol, which can be used in the application mode to support communication between the MCU (host) and an application running on the CYWxxxxx device (controller).

When the CYWxxxxx device powers on, boot logic determines whether a serial flash is connected and, if so, it contains a valid application image. If there is a valid application, the CYWxxxxx device loads and executes the application. If there is no serial flash, then the CYWxxxxx device boots into and stays in the Bluetooth® HCI mode where it waits for the MCU (host) commands. While in Bluetooth® HCI mode, the standard Bluetooth® HCI protocol is used to download an application to the CYWxxxxx device and change the device mode to an application mode. Note that the application may be downloaded and then executed from RAM, or may be downloaded to the serial flash and then executed on the subsequent device reboot.

The AIROC™ HCI Control Protocol is defined in **AIROC™ HCI Control Protocol definition**.

3 Downloading an application and configuration data

3.1 Introduction

This section describes the process of downloading an application to a CYWxxxxx device. The first scenario describes the the process to download an embedded application and its associated configuration data to RAM in a CYWxxxxx device. The second scenario describes the similar serial flash download process, which writes application and configuration data to serial flash before restarting the CYWxxxxx device. These processes can be used by an external MCU to perform the download in place of the ModusToolbox™ build system.

Downloading to RAM (the first scenario) is not supported by CYWxxxxx devices that have On-Chip-Flash (OCF). These devices include CYW20719, CYW20721, CYW20819, and CYW20820. OCF devices only support downloading to serial flash (the second scenario). See Downloading an application to RAM and Downloading an application to serial flash for details on each scenario.

Note: The code present in the ROM is in most cases is sufficient to perform the download. In some cases, the MCU needs to load the minidriver which is used during the remainder of the download process. The minidriver is a set of code and data that replaces the download code in the ROM of the CYWxxxxx device. The minidriver download, provides a way to adapt the download process to handle scenarios that the ROM code does not. For example, a design using the CYWxxxxx may require downloading to a serial flash that requires a different protocol than the ROM can supply. Downloading a minidriver that supports this protocol prior to downloading the application would solve this situation. Minidrivers are not required and are not supplied for some platforms. Minidrivers are optional and specific to each platform and when they are supplied can be found in the 'platforms' subdirectories of the wiced_btsdk project (created when creating any AIROC™ board application with ModusToolbox™). For example, the CYW20819 minidriver can be found here:

<USER_HOME>\mtw\mtb_shared\wiced_btsdk\dev-kit\20819A1\<git-branch>\platforms\minidriver-20819A1-uart-patchram.hex

Note that the vendor-specific HCI commands described in this section have address and length fields in little-endian byte order.

3.2 Preparing for HCI commands

When the device is initially powered on the boot code will attempt to identify the hardware interface to be used for HCI communication. For UART, the device behavior depends on the state of CTS when RST_N is de-asserted. If CTS is low at this time, the device enters the autobaud state (download mode). If CTS is high after the reset, the device will check NVRAM and apply any stored configuration, typically ending in a mode ready to accept all the HCI commands at a default baud rate. If no configuration is available, the device will also enter autobaud mode.

The autobaud mode will attempt to detect the UART baud rate by checking the RX line for the bit pattern of an HCI_RESET command. When detected, the HCI_RESET response is given at the same baud rate. In this mode, most HCI commands will have no response. The HCI_DOWNLOAD_MINIDRIVER command, described in point **4** in **HCI commands and events during a RAM download**, will also have no response when the device is in autobaud mode. To download to the device in this mode, ignore the “no response” to HCI_DOWNLOAD_MINIDRIVER and proceed with the download procedures as described in **Downloading an application to RAM** through **HCI commands and events during a serial flash download**.

3.3 Download file formats

Download images are kept in **.hcd* or **.hex* files. The **.hcd* is more typically used for RAM downloads and the **.hex* format is typically used for flash downloads. Each file format must be parsed and converted to HCI commands to successfully transfer the image to the device. During download operations to reference boards controlled by ModusToolbox™, the ChipLoad application performs these operations.

The **.hcd* format consists of binary records that can be parsed and interpreted directly as HCI commands:

1. The first two binary bytes are the command identifier, for example, the `HCI_WRITE` command, described in point **5** in **HCI commands and events during a RAM download**, 12 is represented by binary bytes 0x4c, 0xfc.
2. The following byte is the command payload length. For example, a binary 0x6 indicates that six more bytes to follow will complete the command.
3. The command payload follows, in binary bytes.

To convert the file successfully to HCI commands, only the transport indication needs to be added. For example, when using UART transport, the hex byte 0x1 should precede any HCI command to indicate that it is a command rather than an event. The detailed specifications for HCI transport are publicly available.

The **.hex* format follows the Intel I32HEX conventions that are widely documented and can be found on Wikipedia, for example. The format consists of records delimited by ASCII carriage return and line feed (0xd, 0xa), but each record also has a start indicator, as described below:

1. Start code ":", ASCII 0x3a.
2. Byte count in record payload as two hexadecimal digits, for example 'FF' is a count of 255.
3. Address as four hexadecimal digits, for example '1000' would represent 0x1000 or 4096.
4. Record type as two hexadecimal digits. The record types used for download images are:
   a) '00' for data record, where address field represents low 16-bits of image destination
   b) '01' for end of file (last record), the payload is zero bytes in length and the address field is not used and set as '0000'
   c) '04' for extended address (high 16 bits of subsequent data record addresses)
   d) '05' for a 32-bit address. The record address field is left at '0000', the length is '04' bytes, and the eight hexadecimal data digits are interpreted as a 32-bit address. For example, '00220001' would be the address 0x220001. This record is often used to indicate a `LAUNCH_RAM` destination described below.
5. Record payload data represented as hexadecimal ASCII digits, two digits per byte and extending for the number of bytes indicated by the record's byte count.
6. Checksum of the entire preceding record data represented as two hexadecimal ASCII digits.

When the **.hex* file is parsed, the data record payloads will resolve into one or more blocks of continuous data. When more than one block of data is present, there will be a discontinuity in the record addresses. The address gap may be described by a '04' record, used to reset the upper 16-bits of address, followed by '00' type data records forming the next block of data. Contiguous data blocks should be collected and segmented to form HCI WRITE_RAM download command payloads as described in point **5** in **HCI commands and events during a RAM download**.

3.4 Downloading an application to RAM

The ModusToolbox™ BTSDK build system automatically downloads the application to either RAM or serial flash depending on which method the board supports, either via command line with the 'make program' command, or through the IDE with the 'Program' launch in the ModusToolbox™ Quick Panel. This section describes the operation of the RAM download process that may be used by an external MCU to perform the same operation using the .hcd file produced by the build process. For example:

<USER_HOME>\mtw\Audio_Watch\build\<board>\Debug\Watch_download.hcd

3.4.1 HCI commands and events during a RAM download

After a download is initiated, host and controller messages are exchanged in the following sequence:

1. The PC (MCU) host issues the following standard Bluetooth® `HCI_RESET` command:
   `01 03 0C 00`
   The following response is expected from the CYWxxxxx device within 100 ms:
   `04 0E 04 01 03 0C 00`
2. To speed up application downloading, the MCU host commands the CYWxxxxx device to communicate at a new, higher rate by issuing the following vendor-specific `UPDATE_BAUDRATE` command:
   `01 18 FC 06 00 00 xx xx xx`
   In the above command, the `xx xx xx xx` bytes specify the 32-bit little-endian value of the new rate in bits per second. For example, 115200 is represented as `00 C2 01 00`.
   The following response to the `UPDATE_BAUDRATE` command is expected within 100 ms:
   `04 0E 04 01 18 FC 00`
3. The host switches to the new baud rate after receiving the response at the old baud rate.
4. If successful, the host issues the following `DOWNLOAD_MINIDRIVER` vendor-specific command:
   `01 2E FC 00`
   The following response is expected from the CYWxxxxx device within 100 ms:
   `04 0E 04 01 2E FC 00`
   If there is not response to the DOWNLOAD_MINIDRIVER command, the device may be in autobaud mode (see **Preparing for HCI commands**). While it is required to send the DOWNLOAD_MINIDRIVER command, it is optional to download a minidriver itself. The ROM download code behavior is sufficient to perform the download for most cases. For these cases, the download process continues directly to **step 5** to download the application image.
   If needed, the minidriver is loaded using `WRITE_RAM` vendor-specific commands, as described in **step 5**. The hex file format indicates the RAM address for each data chunk in the file. Data chunks from the file can be grouped up to the payload size of the `WRITE_RAM` command. To start the minidriver, use a `LAUNCH_RAM` command, as described in **step 6**, to begin minidriver execution at the first address of the minidriver image. For example, if the minidriver download starts at 0x220000, then the `LAUNCH_RAM` command should use 0x220000 as the launch address. After launching the minidriver, continue the application download process with **step 5**.
5. After downloading the minidriver (optional), the host writes application code and configuration data to the CYWxxxxx device by sending `WRITE_RAM` vendor-specific commands. Since the writes are destined for the CYWxxxxx device's RAM, the destination addresses in the `WRITE_RAM` commands are absolute RAM locations.
   The following `WRITE_RAM` command is an example:
   `01 4C FC nn xx xx xx xx yy yy yy …`
   In the above `WRITE_RAM` command:
   – `nn` is 4 + N, which represents 4 address bytes plus N payload bytes.

  - `xx xx xx xx` is the 4-byte, absolute RAM address.
  - `yy yy yy …` are the N payload bytes to be loaded into the addressed RAM location.

6. The following response to each `WRITE_RAM` command is expected within 200 ms:
   ```
   04 0E 04 01 4C FC 00
   ```
7. After the host has written all application and configuration data to RAM, it sends a `LAUNCH_RAM` command with the address stored in the last record of the hardware configuration data (HCD) file.

   An example `LAUNCH_RAM` command is shown here:
   ```
   01 4E FC 04 xx xx xx xx
   ```
   In the above `LAUNCH_RAM` command, `xx xx xx xx` is the 4-byte absolute RAM address of the last HCD record. Typically, the last address is 0xFFFFFFFF.

   The following response to the `LAUNCH_RAM` command is expected within 200 ms:
   ```
   04 0E 04 01 4E FC 00
   ```

Note: *Following a successful LAUNCH_RAM command, the device is in the application mode and the application is running.*

In the application mode, the UART configuration depends on the application. If the application sets the baud rate to 3 Mbps at start-up then the MCU must also configure the UART for 3 Mbps operation to successfully communicate with the CYWxxxxx device. The application sets the baud rate using the following command: `uart_SetBaudrate(0, 0, 3000000)`. The default application baud rate is configured in the call to `wiced_transport_init()`.

3.5 Downloading an application to serial flash

The ModusToolbox™ BTSDK build system automatically downloads the application to either RAM or serial flash depending on which method the board supports, either via command line with the 'make program' command, or through the IDE with the 'Program' launch in the ModusToolbox™ Quick Panel. This section describes the operation of the serial flash download process that may be used by an external MCU to perform the same operation using the .hex file produced by the build process. For example:

<USER_HOME>\mtw\Audio_Watch\build\<board>\Debug\Watch_download.hex

Note that the hex file format consists of records that include data and address information. For serial flash download hex files, the addresses used map to offsets in the flash device. For example, CYW20706 and CYW20735 hex records map the address 0xFF000000 to the base, or 0, offset in the attached serial flash device. Similarly, the CYW20719 and CYW20819 use 0x00500000 as the base address for the on-chip flash.

Note that the vendor-specific HCI commands `READ_RAM`, `WRITE_RAM`, and `LAUNCH_RAM` are not limited to actual RAM address ranges. The same commands are used to write to non-volatile storage like serial flash by using mapped addresses that correspond to offsets within these devices.

3.5.1 HCI commands and events during a serial flash download

This section describes the protocol for the download process described above for the situations when an MCU needs to load the image to the serial flash attached to the CYWxxxxx device.

During the download process, the host and controller exchange messages in the following sequence:

1. The PC (MCU) host issues the following standard Bluetooth® `HCI_RESET` command:
   ```
   01 03 0C 00
   ```
   The following response is expected from the CYWxxxxx device within 100 ms:
   ```
   04 0E 04 01 03 0C 00
   ```

2. To speed up application downloading, the MCU host commands the CYWxxxxx device to communicate at a new, higher rate by issuing the following vendor-specific `UPDATE_BAUDRATE` command:
   `01 18 FC 06 00 00 xx xx xx xx`
   In the above command, the `xx xx xx xx` bytes specify the 32-bit value of the new rate in bits per second. For example, 115200 is represented as `00 C2 01 00`.
   The following response to the `UPDATE_BAUDRATE` command is expected within 100 ms:
   `04 0E 04 01 18 FC 00`
3. The host switches to the new baud rate after receiving the response at the old baud rate.
4. If successful, the host issues the following `DOWNLOAD_MINIDRIVER` command:
   `01 2E FC 00`
   The following response is expected from the CYWxxxxx device within 100 ms:
   `04 0E 04 01 2E FC 00`
   If there is not response to the `DOWNLOAD_MINIDRIVER` command, the device may be in autobaud mode (see **Preparing for HCI commands**). While it is required to send the `DOWNLOAD_MINIDRIVER` command, it is optional to download a minidriver itself. The ROM download code behavior is sufficient to perform the download for some cases. For these cases, the download process continues directly to **step 5** to download the application image.
   If needed, the minidriver is loaded using `WRITE_RAM` vendor-specific commands, as described in **step 5**. The hex file format indicates the RAM address for each data chunk in the file. Data chunks from the file can be grouped up to the payload size of the `WRITE_RAM` command. To start the minidriver, use a LAUNCH_RAM command, as described in **step 8**, to begin minidriver execution at the first address of the minidriver image. For example, if the minidriver download starts at 0x220000, then the LAUNCH_RAM command should use 0x220000 as the launch address. After launching the minidriver, continue the application download process with **step 5**.
5. After downloading the minidriver, the `CHIP_ERASE` command is sent. If it is desirable to preserve some data in flash and only erase sectors that will be written, this step may be skipped.

Note: *A `SECTOR_ERASE` command is also available for scenarios where only certain targeted sectors need to be erased, though that is not part of the download procedure. Contact **Infineon support** for information if that advanced functionality is needed.*

   The following `CHIP_ERASE` command is an example:
   `01 CE FF 04 xx xx xx xx`
   In the above `CHIP_ERASE` command, `xx xx xx xx` is the 4-byte address indicating the range of the non-volatile memory to be erased. A special value of `EF EE BE FC` (0xFCBEEEEF) is used to signal “use the lowest valid non-volatile memory range”. Otherwise, the device to be erased is determined from the value when compared to valid ranges, where 0x500000 would be the start of on-chip flash when supported and 0xFF000000 would be the start of off-chip flash.
6. After downloading the mini-driver(optional), perform `CHIP_ERASE`, the host writes application code and configuration data to the CYWxxxxx device by sending `WRITE_RAM` commands.
   The following `WRITE_RAM` command is an example:
   `01 4C FC nn xx xx xx xx yy yy yy …`
   In the above `WRITE_RAM` command:
   a) `nn` is 4 + N, which represents 4 address bytes plus N payload bytes.
   b) `xx xx xx xx` is the 4-byte, mapped address for serial flash offset.
   c) `yy yy yy …` are the N payload bytes to be loaded into the mapped address. The following response to each `WRITE_RAM` command is expected within 200 ms:

```
04 0E 04 01 4C FC 00
```

7. After the host has written application and configuration data to flash, it can be validated with a CRC check command. Also, any block can be read back using the `READ_RAM` command.
   a) CRC method: returns the CRC-32 calculated by reading the data range indicated in the command. The following CRC validation command is an example:
   ```
   01 CC FC 08 xx xx xx xx yy yy yy yy
   ```
   In the above command, the `xx xx xx xx` bytes specify the 32-bit value of the mapped address for the serial flash offset and the `yy yy yy yy` bytes specify the 32-bit value of the number of bytes to be read from the serial flash for the CRC calculation.

   The following response is expected after the `READ_RAM` command:
   ```
   04 0E 08 01 CC FC 00 xx xx xx xx
   ```
   In the above response, the xx bytes are the 32-bit CRC-32 value calculated by reading the data bytes from flash starting at the virtual address given in the CRC command and continuing for the number of bytes provided in the CRC command.

   b) The following READ_RAM command is an example:
   ```
   01 4D FC 05 xx xx xx xx yy
   ```
   In the above command, the `xx xx xx xx` bytes specify the value of the serial flash offset's mapped address and the yy byte specifies the length of data to be read.

   The following response is expected within 100 milliseconds of the `READ_RAM` command:
   ```
   04 0E xx 01 4D FC 00 yy yy yy …
   ```
   In the above response, the `xx` byte represents N+4, where N is the number of data bytes read from the flash. The `yy` bytes are the actual data read back from the mapped offset.
8. After the host has written and validated all application and configuration data to RAM, it sends a `LAUNCH_RAM` command with the special destination address specifying reboot for the device, typically 0xFFFFFFFF.

   An example LAUNCH_RAM command is shown here:
   ```
   01 4E FC 04 xx xx xx xx
   ```
   In the above command, the `xx xx xx xx` bytes represent the destination address for the CPU branch.

   The following response to the `LAUNCH_RAM` command is expected within 200 ms:
   ```
   04 0E 04 01 4E FC 00
   ```

4 AIROC™ HCI Control Protocol definition

The CYWxxxxx uses the following 5-byte packet header for command/event exchanges with the host MCU.

Packet type	Command/ event code	Group code	Packet length	
HCI_WICED_PKT(0x19)	HCI_CONTROL_ COMMAND_...	HCI_CONTROL_ GROUP_...	Low byte	High byte

The protocol follows the standard Bluetooth® HCI rules for parameter byte ordering. For example, the attribute handle 0x210 is sent in two bytes, 0x10 followed by 0x02.

All commands and events are split into groups. **Table 1** shows the groups defined by the AIROC™ HCI Control Protocol.

Table 1 AIROC™ HCI Control Protocol command and event groups

Group name	Group value	Description
HCI_CONTROL_GROUP_DEVICE	0x00	General control of CYWxxxxx management and Bluetooth® functionality
HCI_CONTROL_GROUP_LE	0x01	LE device-related commands and events
HCI_CONTROL_GROUP_GATT	0x02	GATT commands and events
HCI_CONTROL_GROUP_HF	0x03	Hands-Free Profile commands, events, and data
HCI_CONTROL_GROUP_SPP	0x04	Serial Port Profile commands, events, and data
HCI_CONTROL_GROUP_AUDIO	0x05	Audio/video (AV) commands, events, and data
HCI_CONTROL_GROUP_HIDD	0x06	HID device (HIDD) commands and events
HCI_CONTROL_GROUP_AVRC_TARGET	0x07	AV Remote Control (AVRC) Target commands and events
HCI_CONTROL_GROUP_TEST	0x08	Test commands
HCI_CONTROL_GROUP_TIME	0x0A	Current Time Client Application events
HCI_CONTROL_GROUP_ANCS	0x0B	Apple Notification Center Service (ANCS) commands and events
HCI_CONTROL_GROUP_ALERT	0x0C	Immediate Alert Service (IAS) events
HCI_CONTROL_GROUP_LN	0x0D	Location and navigation commands and events.
HCI_CONTROL_GROUP_IAP2	0x0E	iPod Accessory Protocol implementation (iAP2) commands and events
HCI_CONTROL_GROUP_AG	0x0F	Hands-Free Audio Gateway (AG) commands and events
HCI_CONTROL_GROUP_AIO_SERVER	0x10	Automation IO (AIO) server commands and events
HCI_CONTROL_GROUP_AIO_CLIENT	0x10	AIO Client commands and events
HCI_CONTROL_GROUP_AVRC_CONTROLLER	0x11	AV remote control (AVRC) controller commands and events
HCI_CONTROL_GROUP_AMS	0x12	Apple Media Service (AMS) commands and events

Group name	Group value	Description
HCI_CONTROL_GROUP_DFU	0x2A	Device Firmware Upgrade over HCI
HCI_CONTROL_GROUP_MISC	0xFF	Miscellaneous commands and events

See **AIROC™ HCI Control Protocol commands** for information on the AIROC™ HCI Control Protocol commands.

See **AIROC™ HCI Control Protocol events** for information on the AIROC™ HCI Control Protocol events.

5 AIROC™ HCI Control Protocol commands

5.1 Device commands: HCI_CONTROL_GROUP_DEVICE

The device commands allow the host to manage the behavior of the CYWxxxxx.

5.1.1 Reset

The Reset command causes the CYWxxxxx to restart. After initialization completes, the CYWxxxxx sends a Device Started event (see **Device Started**).

Table 2 Reset Command

Item	Description
Operating code	0x01
Parameters	–

5.1.2 Trace Enable

The Trace Enable command instructs the CYWxxxxx to start or stop forwarding the AIROC™ logs and virtual HCI traces.

The CYWxxxxx provides the following two trace types:

- An output of the WICED_BT_TRACE statements.
- A binary dump of the virtual HCI commands, events, and data packets between the embedded host stack and the CYWxxxxx controller.

The WICED_BT_TRACE output is forwarded in the HCI_CONTROL_EVENT_WICED_TRACE when a corresponding trace is enabled.

The virtual HCI traces are sent over UART using HCI_CONTROL_EVENT_HCI_DATA.

Table 3 Trace Enable Command

Item	Description	
Operating code	0x02	
Parameters	Bluetooth® HCI trace enable (1 byte)	If true, HCI traces are routed through the AIROC™ HCI interface to the host.
	AIROC™Trace route (1 byte)	0: Traces are not generated. 1: Traces are forwarded to the HCI UART, encoded in BTSpy format. 2: Traces are forwarded to the HCI UART as raw text. 3: Traces are forwarded to the debug UART (deprecated). 4: Traces are forwarded to the peripheral UART.

5.1.3 Set Local Bluetooth® Device Address

The Set Local Bluetooth® Device Address command configures the CYWxxxxx to use a new Bluetooth® device address. The MCU host typically sends this command during a start-up operation. The address is passed as a parameter in little-endian format.

Table 4 Set Local Bluetooth® Device Address command

Item	Description
Operating code	0x03
Parameters	A 6-byte Bluetooth® device address

5.1.4 Push NVRAM Data

If a CYWxxxxx does not have an embedded NVRAM, it relies on the MCU to save application-specific NVRAM data, which the CYWxxxxx can provide in NVRAM Data events (see **NVRAM Data**). At start-up, the MCU host should push all saved NVRAM information to the CYWxxxxx before the CYWxxxxx establishes any Bluetooth® connections.

Table 5 Push NVRAM Data command

Item	Description	
Operating code	0x05	
Parameters	nvram_id (2 bytes)	ID of an NVRAM information chunk
	nvram_data (variable bytes)	Data corresponding to nvram_id

5.1.5 Delete NVRAM Data

An application running on the MCU host may request the CYWxxxxx to delete NVRAM information for a specific nvram_id.

Table 6 Delete NVRAM Data command

Item	Description	
Operating code	0x06	
Parameters	nvram_id	2-byte ID of an NVRAM information chunk

5.1.6 Inquiry

The Inquiry command lets an application cancel or start a Bluetooth® inquiry procedure.

If a device is found during an inquiry, the CYWxxxxx sends an Inquiry Result event (see **Inquiry result**).

When an inquiry procedure completes, the CYWxxxxx sends an Inquiry Complete event (see **Inquiry complete**).

Table 7 Inquiry command

Item	Description	
Operating code	0x07	
Parameters	Enable (1 byte)	0: Cancel the inquiry procedure.
		1: Start an inquiry procedure

5.1.7 Set Visibility

The Set Visibility command allows the host to turn discoverability and connectability ON and OFF. After a CYWxxxxx restart, it is not discoverable (non-discoverable) and not connectable (non-connectable).

Note: *Attempts to make the CYWxxxxx discoverable and non-connectable will be rejected because, according to the Bluetooth® specifications, a discoverable device should also be connectable.*

After the CYWxxxxx receives this command, it reports command success or failure in the Command Status event (see **Command Status**).

Table 8 Set Visibility command

Item	Description	
Operating code	0x08	
Parameters	Enable (1 byte)	0: Cancel the inquiry procedure.
		1: Start an inquiry procedure
	Connectability (1 byte)	0: Not connectable
		1: Connectable

5.1.8 Set Pairing Mode

The MCU can set the CYWxxxxx to be pairable or not pairable using this command. A BR/EDR connection will be rejected if a device is not pairable and there is no link key to secure the connection. Similarly, while a device is not pairable, access to LE characteristics requiring security will fail. While pairable, a pairing attempt from a peer device will be accepted.

After the CYWxxxxx receives this command, it reports command success or failure in the Command Status event (see **Command Status**).

Table 9 Set Pairing Mode command

Item	Description	
Operating code	0x09	
Parameters	Pairing mode (1 byte)	0: Not pairable
		1: Pairable

5.1.9 Unbond

The MCU can use this command to instruct the CYWxxxxx to remove bonding information (that is, security keys) for the device whose Bluetooth® device address is passed as a parameter.

After the CYWxxxxx receives this command, it reports command success or failure in the Command Status event (see **Command Status**).

Table 10 Unbond command

Item	Description	
Operating code	0x0A	
Parameters	Address (6 bytes)	Bluetooth® device address

5.1.10 User Confirmation

The MCU should send this command after it receives a User Confirmation Request event (see **User Confirmation Request**) from the CYWxxxxx to accept or reject pairing. It is assumed that an MCU will display the numeric comparison code provided in the User Confirmation Request event and a user will provide the yes/no input that will be passed to the CYWxxxxx as the User Confirmation command.

Table 11 User Confirmation Command

Item	Description	
Operating code	0x0B	
Parameters	Address (6 bytes)	Bluetooth® device address
	Accept/Reject (1 byte)	0: Reject pairing, or the numeric comparison code does not match.
		1: Accept pairing

5.1.11 Enable Coexistence

This command allows the MCU to enable the coexistence functionality in designs that include Bluetooth®/Bluetooth® LE and Wi-Fi applications.

Table 12 Enable Coexistence command

Item	Description
Operating code	0x0C
Parameters	–

5.1.12 Disable Coexistence

This command allows the MCU to disable the coexistence functionality in designs that include Bluetooth®/Bluetooth® LE and Wi-Fi applications.

Table 13 Disable Coexistence command

Item	Description
Operating code	0x0D
Parameters	–

5.1.13 Set Battery Level

This miscellaneous command allows the MCU to set the battery level in the GATT database of the CYWxxxxx. A connected peer device can read the battery level using a standard GATT read operation.

Table 14 Set Battery Level command

Item	Description
Operating code	0x0E
Parameters	Battery level (1 byte) remaining battery capacity as a percentage (1 to 100).

5.1.14 Read Local Bluetooth® Device Address

The MCU can send this command to read the local Bluetooth® Device Address of the CYWxxxxx. When the CYWxxxxx receives this command, it responds with the Read Local BDA event message containing the Bluetooth® device address.

Table 15 Read Local Bluetooth® Device Address command

Item	Description
Operating code	0x0F
Parameters	–

5.1.15 Start Bond

The MCU can send this command to initiate bonding with an unbonded device.

Table 16 Start Bond command

Item	Description	
Operating code	0x10	
Parameters	Address (6 bytes)	
	Transport (1 byte)	1 = BR/EDR, 2 = LE
	Address type (1 bytes)	0 = Public, 1 = Random (LE only)

5.1.16 Read Buffer Pool Usage Statistics

The MCU can send this command to read the buffer pool usage statistics to understand the buffer pool usage by the application running on the CYWxxxxx, and to identify if there is a possibility of buffers running out for a given application use case. The Buffer Pool Usage Statistics event will be sent from the CYWxxxxx to the MCU which includes the Buffer Pool Usage Statistics.

Table 17 Read Buffer Pool Usage Statistics command

Item	Description
Operating code	0x11
Parameters	–

5.1.17 Set Local Name

This command configures the CYWxxxxx to use a new user-friendly name. The MCU host typically sends this command during a start-up. The name is a UTF-8 encoded user-friendly descriptive name for the device.

After the CYWxxxxx receives this command, it reports command success or failure in the Command Status event (see **Command Status**)

Table 18 Set Local Name command

Item	Description
Operating code	0x12
Parameters	A UTF-8 encoded user-friendly descriptive name for the device. If the name contained in the parameter is shorter than 248 octets, the end of the name is

Item	Description
	indicated by a null octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values. Default name configured by the firmware is device_name field of wiced_bt_cfg_settings_t (which is in the *wiced_bt_cfg.c* file of Sample applications)

5.2 LE commands: HCI_CONTROL_GROUP_LE

The LE commands let the MCU perform various LE Generic Access Profile (GAP) procedures using the CYWxxxxx.

5.2.1 LE Scan

The LE scan command instructs the CYWxxxxx to start or stop device discovery. The scan mode, window, interval, and duration are configured locally in the application running on the CYWxxxxx (see the *wiced_bt_cfg.c* file in ModusToolbox™). When the device starts scanning, it executes a high- duty-cycle scan where it listens for advertisements during programmed windows occurring at programmed intervals for a programmed duration. Unless canceled by the application, the device then automatically switches to a low-duty-cycle scan. The device stops scanning after the low-duty-cycle scan duration.

Figure 2 shows an advertisement scanning cycle.

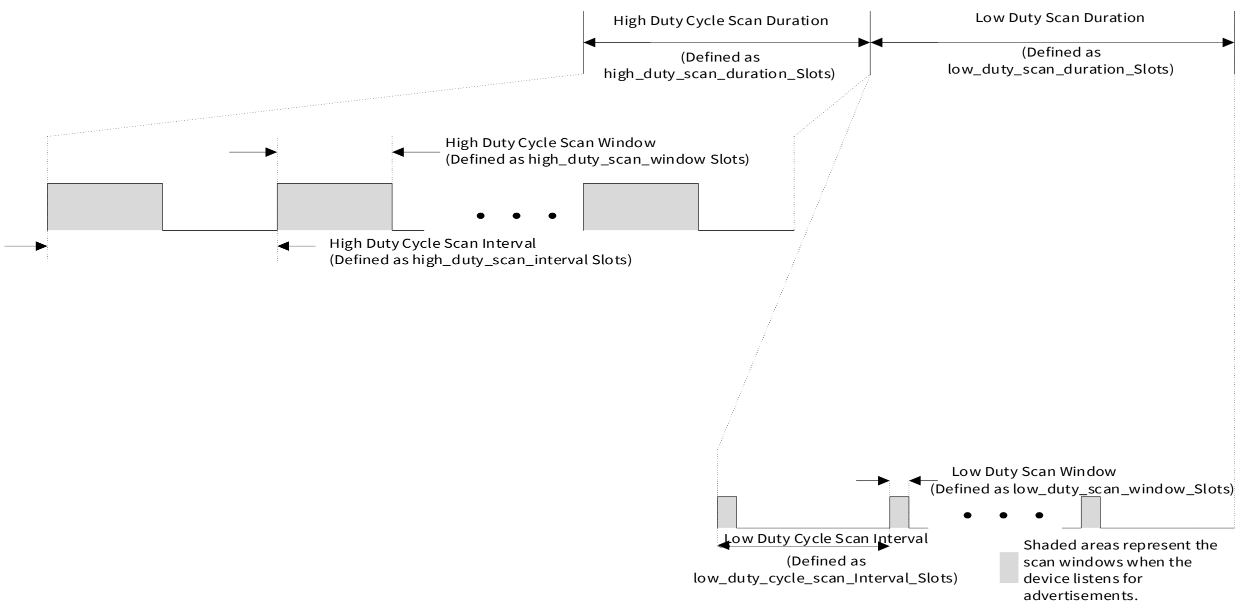
High Duty Cycle Scan Duration
(Defined as high_duty_scan_duration_Slots)
Low Duty Scan Duration
(Defined as low_duty_scan_duration_Slots)
High Duty Cycle Scan Window
(Defined as high_duty_scan_window Slots)
High Duty Cycle Scan Interval
(Defined as high_duty_scan_interval Slots)
Low Duty Scan Window
(Defined as low_duty_scan_window_Slots)
Low Duty Cycle Scan Interval
(Defined as low_duty_cycle_scan_Interval_Slots)
Shaded areas represent the scan windows when the device listens for advertisements.

Figure 2 Advertisement scanning

When the CYWxxxxx receives and processes this command, it reports the scan state change in the Scan Status event (see **LE Scan Status**). Scan Status events are also sent when the CYWxxxxx switches from a high-duty-cycle scan to a low-duty-cycle scan and from a low-duty-cycle scan to not scanning.

Table 19 LE Scan command

Item	Description	
Operating code	0x01	
Parameters	Enable (1 byte)	0: Stop device-discovery scanning. 1: Start device-discovery scanning.
	Filter duplicates (1 byte)	0: Do not filter duplicate advertisements. 1: Filter duplicate advertisements.

5.2.2 LE Advertise

The LE Advertise command instructs the CYWxxxxx to stop or start sending advertisements. Typically, advertisements are sent so that a central device peer can discover and optionally connect to a peripheral device peer. When a CYWxxxxx receives this command, it sends advertisements based on parameters configured in the *wiced_bt_cfg_ble_advert_settings_t* structure of the *wiced_bt_cfg_settings_t* structure (which is in the *wiced_bt_cfg.c* file of ModusToolbox™).

Initially, advertisements are sent out using a programmed high-duty-cycle advertisement profile. After the high- duty-cycle duration (for example, high_duty_duration) expires, advertisements are sent out in accordance with a programmed low-duty-cycle advertisement profile, which also has a duration (for example, low_duty_duration). After the low_duty_duration, the CYWxxxxx stops sending advertisements.

Figure 3 shows the high-duty-cycle and low-duty-cycle advertisement-sending profiles.

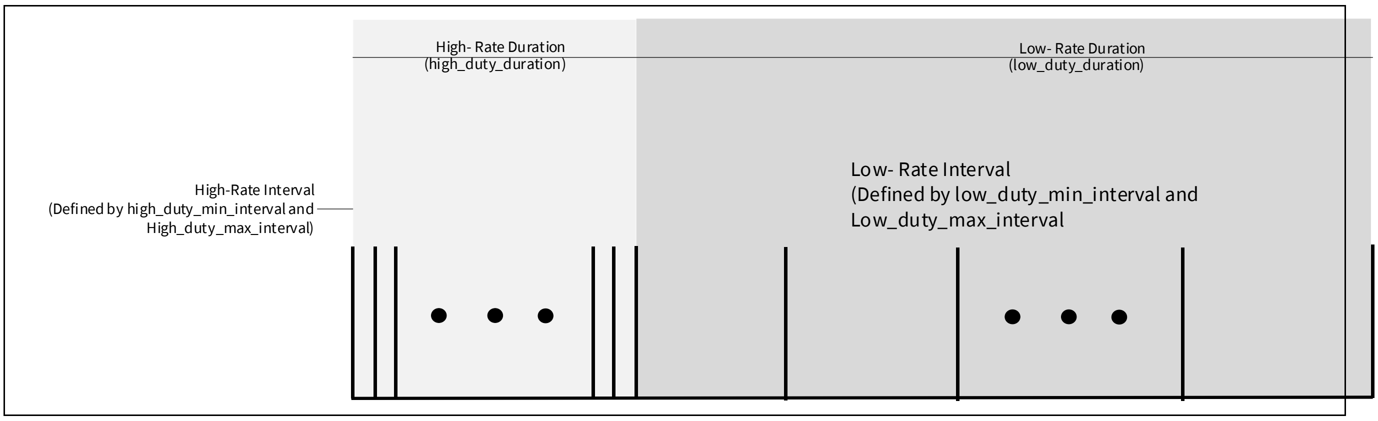

Figure 3 Advertisement-sending profile

When the CYWxxxxx receives and processes this command, it reports advertisement state changes in the Advertisement State event (see **LE Advertisement State**). Advertisement State events are also sent when the CYWxxxxx controller switches from the high-duty-cycle advertisements to low-duty-cycle advertisements and from low-duty-cycle advertisements to no advertisements.

Table 20 LE Advertise command

Item	Description	
Operating code	0x02	
Parameters	Enable (1 byte)	0: Disable the ability to be discovered (don't send advertisements). 1: Enable the ability to be discovered (send advertisements).

5.2.3 LE Connect

The LE Connect command instructs the CYWxxxxx to establish a connection to a specified peer device.

When the CYWxxxxx receives and processes this command, it reports the status to the Command Status event (see **Command Status**).

When a connection is established, the CYWxxxxx sends the LE Connected event (see **LE Connected**).

Table 21 LE Connect command

Item	Description
Operating code	0x03
Parameters	Address type (1 byte)
	Device address (6 bytes)

5.2.4 LE Cancel Connect

The LE Cancel Connect command instructs the CYWxxxxx to stop a connection-establishment attempt.

When the CYWxxxxx receives and processes this command, it reports the status via the Command Status event (see **Command Status**).

Table 22 LE Cancel Connect command

Item	Description
Operating code	0x04
Parameters	–

5.2.5 LE Disconnect

The LE Disconnect command terminates a previously established Bluetooth® LE connection. The connection handle is a two-byte value reported in the LE Connected event (see **LE Connected**), sent by the CYWxxxxx upon a successful connection.

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist, it reports not connected in the Command Status event (see **Command Status**).
- If the connections exist:
  - It reports success in the Command Status event.
  - It starts the disconnection process.
  - It reports the LE Disconnected event when the disconnection process finishes.

Table 23 LE Disconnect command

Item	Description
Operating code	0x05
Parameters	Connection handle (2 bytes)

5.2.6 LE Repair

This command instructs the CYWxxxxx to delete link keys associated with a previously paired device and re-initiate a pairing sequence with the same device.

The NVRAM ID parameter should match the value reported to the MCU after the successful pairing in the NVRAM Data event (see **NVRAM Data**).

Table 24 LE Repair Command

Item	Description	
Operating code	0x06	
Parameters	Device address (6 bytes)	Address of the device from the original pairing.
	NVRAM ID (2 bytes)	ID associated with the address of the device from the original pairing.

5.2.7 LE Get Identity Address

When an initial connection with a peer is established, the MCU will receive a private random address (if a private random address is used) of the device in the LE Connection Up event message. The LE Get Identity Address command can be used by the MCU to retrieve the identity address, which is a public or a static random address of the peer device.

If the MCU attempts to retrieve the resolved identity address of the peer, then this command can be used. The resolved identity address of the peer will be returned in the LE Identity Address event message (see **LE Identity Address**).

Table 25 LE Get Identity Address command

Item	Description	
Operating code	0x07	
Parameters	Device address (6 bytes)	Device address (6 bytes)

5.2.8 LE Set Channel Classification

The MCU can send this command to the CYWxxxxx and set the channel classification for data channels. This channel classification is only applicable to connections where the CYWxxxxx is the central. This command contains 37 1-bit fields which correlate to the value for the link layer channel index 0 - 36.

Table 26 LE Set Channel Classification command

Item	Description	
Operating code	0x08	
Parameters	Bluetooth® LE Channel map (5 bytes)	This parameter contains 37 1-bit fields for the link layer channel indexes 0 -36. Channel n is bad = 0 Channel n is unknown = 1 At least one channel should be marked as unknown.

5.2.9 LE Set Connection Parameters

The MCU can send this command to the CYWxxxxx and change the connection parameters (interval min/max, latency and timeout) of the LE link.

Table 27 LE Set Connection Parameters command

Item	Description	
Operating code	0x08	
Parameters	Connection handle (2 bytes)	
	Connection interval minimum (2 bytes)	Time = N * 1.25 ms
	Connection interval maximum (2 bytes)	Time = N * 1.25 ms
	Peripheral latency (2 bytes)	In number of connection event
	Timeout (2 bytes)	Time = N * 10 ms

5.2.10 LE Set Raw Advertisement Data

The MCU can send this command to the CYWxxxxx to set the data used in advertising packets that have a data field. The data is represented in TLV format. The TLVs must fit in 31 bytes. If the last TLV exceeds the 31-byte boundary, the entire TLV will be ignored.

After the CYWxxxxx receives this command, it reports command success or failure in the Command Status event (see **Command Status**).

Table 28 LE Set Raw Advertisement Data command

Item	Description
Operating code	0x0a
Parameters	Number of TLVs (1 byte)
	Array of TLVs. Each TLV is of format:

Item	Description
	Advertisement data type (1 byte) See wiced_bt_ble_advert_type_e which is in the *wiced_bt_ble.h* Length (2 bytes) In little endian format Value (variable length – number of bytes indicated by length field)

5.3 GATT commands: HCI_CONTROL_GROUP_GATT

The GATT commands let the MCU perform various Generic Attribute Profile (GATT) procedures using the CYWxxxxx.

5.3.1 GATT Discover Services

The GATT Discover Services command enables service discovery over an established Bluetooth® LE connection. The connection handle is a two-byte value reported in the LE Connected event (see **LE Connected**).

The *hci_control* application uses the Discover all primary services GATT procedure. The start and end handles are passed to the GATT Read by group type request.

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist or the device is busy performing another action, such as discovery, reading, writing, and so, it reports the relevant status in the Command Status event (see **Command Status**).
- If the connection exists and the device is not busy performing another action, then it reports Success in the Command Status event and starts the discovery process.

The CYWxxxxx sends a GATT Service Discovered event (see **GATT Service Discovered**) for each discovered service. When a peer reports that there are no more services, the GATT Discovery Complete event (see **GATT Discovery Complete**) is issued. The MCU should not send any new discovery, read, or write commands until after receiving the GATT Discovery Complete event.

Table 29 GATT Discover Services command

Item	Description
Operating code	0x01
Parameters	Connection handle (2 bytes)
	Start handle (2 bytes)
	End handle (2 bytes)

5.3.2 GATT Discover Characteristics

The GATT Discover Characteristics command enables characteristic discovery over an established Bluetooth® LE connection. The connection handle is a two-byte value reported in the LE Connected event (see **LE Connected**).

The *hci_control* application uses the Discover all characteristics of a service GATT procedure. The start and end handles are passed to the GATT read by type request.

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist or the device is busy performing another action, such as discovery, reading, writing, and so on, it reports the relevant status in the Command Status event (see **Command Status**).
- If the connection exists and the device is not busy performing another action, then it reports Success in the Command Status event and starts the discovery process.

The CYWxxxxx reports a GATT Characteristic Discovered event (see **GATT Service Discovered**) for each discovered characteristic. When a peer reports that there are no more characteristics, the GATT Discovery Complete event (see **GATT Discovery Complete**) is issued. The MCU should not send any new discovery, read, or write commands until after receiving the GATT Discovery Complete event.

Table 30 GATT Discover Characteristics command

Item	Description
Operating code	0x02
Parameters	Connection handle (2 bytes)
	Start handle (2 bytes)
	End handle (2 bytes)

5.3.3 GATT Discover Descriptors

The GATT Discover Descriptors command enables characteristic-descriptors discovery over an established Bluetooth® LE connection. The connection handle is a two-byte value reported in the LE Connected event (see **LE Connected**).

The hci_control application uses the Discover all characteristic descriptors GATT procedure. The start and end handles are passed to the find info request.

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist or the device is busy performing another action, such as discovery, reading, writing, etc., then it reports the relevant status in the Command Status event (see **Command Status**).
- If the connection exists and the device is not busy performing another action, then it reports Success in the Command Status event and starts the discovery process.

The CYWxxxxx reports a GATT Descriptor Discovered event (see **GATT Service Discovered**) for each discovered characteristic descriptor. When a peer reports that there are no more descriptors, the CYWxxxxx sends a GATT Discovery Complete event (see **GATT Discovery Complete**). The MCU should not send any new discovery, read, or write commands until after receiving the GATT Discovery Complete event.

Table 31 GATT Discover Descriptors command

Item	Description
Operating code	0x03
Parameters	Connection handle (2 bytes)
	Start handle (2 bytes)
	End handle (2 bytes)

5.3.4 GATT Command Read Request

The GATT Command Read Request command enables MCU to read a characteristic value or a descriptor value over an established Bluetooth® LE connection. The connection handle is a two-byte value reported in the LE Connected event (see **LE Connected**).

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist or the device is busy performing another action, such as discovery, reading, writing, and so on, then it reports the relevant status in the Command Status event (see **Command Status**).
- If the connection exists and the device is not busy performing another action, then it reports success in the Command Status event and starts the read process.

When a GATT Command Read Request is received over the UART, the *hci_control* application sends the read request for the attribute handle received in the command.

Figure 4 shows an example message sequence that takes place when one device (represented as the combination of MCU1 and BT1) requests a static attribute such as the Bluetooth® device name from a second device (represented as the combination of MCU2 and BT2). In this scenario, BT2 has the attribute value and returns it.

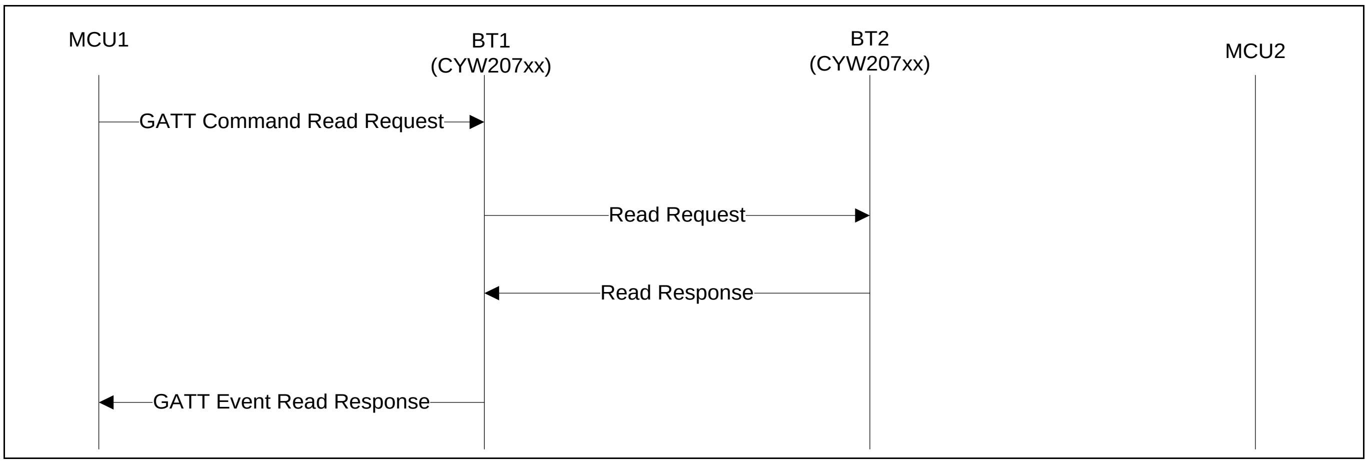

Figure 4 **Reading a static attribute from a peer**

Figure 5 shows an example where BT2 must get an attribute value from MCU2.

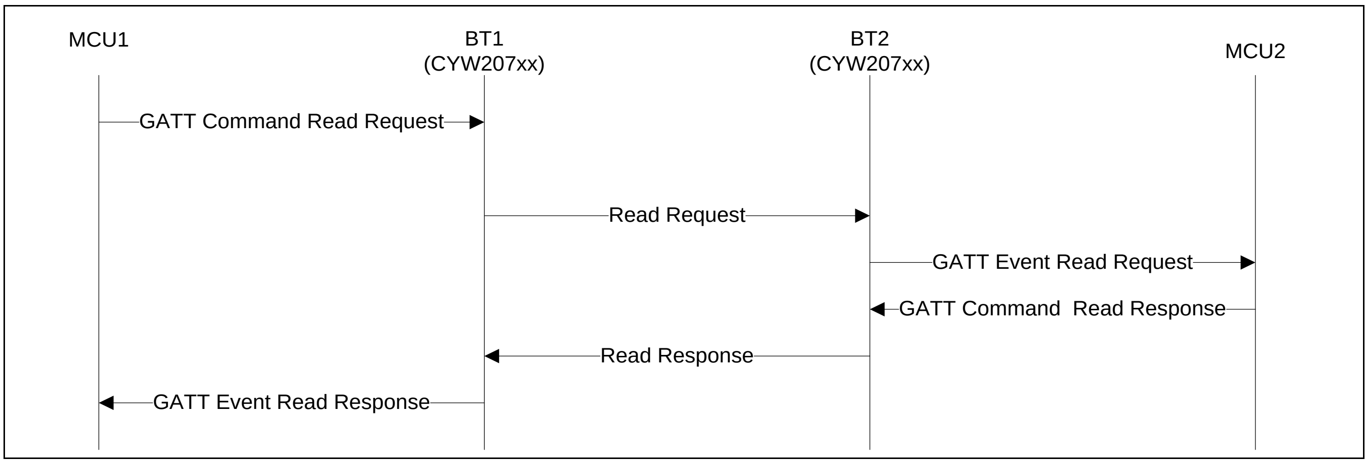

Figure 5 Reading a dynamic attribute from a peer

When a GATT Read Response or a GATT Error Response is received over the Bluetooth® link, the *hci_control* application sends the GATT Event Read Response (see **GATT Event Read Response**). The MCU should not send any new discovery, read, or write commands until after receiving the GATT Event Read Response.

Table 32 GATT Command Read Request

Item	Description
Operating code	0x04
Parameters	Connection handle (2 bytes)
	Attribute handle (2 bytes)

5.3.5 GATT Command Read Response

The GATT Command Read Response is sent by the MCU in response to a GATT Event Read Request (see **Figure 5** in GATT Event Write Request). The connection and attribute handles are the same 2- byte values that were sent in the GATT Event Read Request.

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist, then it reports the relevant status in the Command Status event (see **Command Status**).
- If the connection exists, then it reports success in the Command Status event and sends the response to the connected Bluetooth® device.

Table 33 GATT Command Read Response

Item	Description
Operating code	0x05
Parameters	Connection handle (2 bytes)
	Attribute handle (2 bytes)
	Read status (1 byte)
	Data (variable bytes)

5.3.6 GATT Command Write

The GATT Command Write command enables MCU to schedule transmissions over an established LE connection. The connection handle is a two-byte value reported in the LE Connected event (see **LE Connected**).

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist or the device is busy performing another action, such as discovery, reading, writing, etc., then it reports the relevant status in the Command Status event (see **Command Status**).
- If the connection exists and the device is not busy performing another action, then it reports Success in the Command Status event and starts the write process.

The CYWxxxxx has a limited number of transmit buffers. If the *hci_control* application is able to allocate a buffer and schedule it for transmission, then the write operation is considered complete and the *hci_control* application sends the GATT Event Write Response (see

GATT Event Write Response). If all transmit buffers are already allocated and, thus, unavailable, then the *hci_control* application saves the data received in the command and delays sending the GATT Event Write Response until a transmit buffer becomes available and the data gets scheduled for transmission. The MCU should not send any new discovery, read, or write commands until after receiving the GATT Event Write Response.

Table 34 GATT Command Write command

Item	Description
Operating code	0x06
Parameters	Connection handle (2 bytes)
	Attribute handle (2 bytes)
	Data (variable bytes)

Figure 6 shows an example GATT Command Write message sequence.

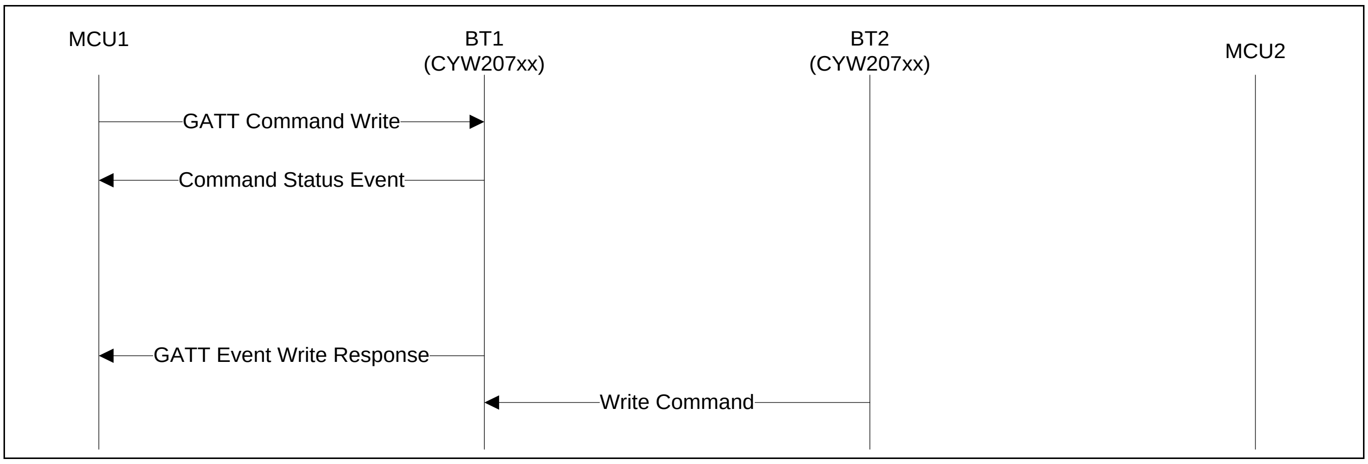

Figure 6 GATT Command Write Message sequence

5.3.7 GATT Command Write Request

The GATT Command Write Request enables MCU to write a characteristic value or a descriptor value over an established LE connection. The connection handle is a two-byte value reported in the LE Connected event (see **LE Connected**).

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist or the device is busy performing another action, such as discovery, reading, writing, etc., then it reports the relevant status in the GATT Event Write Response event (see **Command Status**).
- If the connection exists and the device is not busy performing another action, then it reports success in the GATT Event Write Response event and starts the write process.

When the command is received over the UART, the *hci_control* application sends a GATT Write Request for the attribute handle received in the command. When a GATT Write Response or a GATT Error Response is received from a connected peer device, the hci_control application sends the GATT Event Write Response (see

GATT Event Write Response) to a connected MCU. The MCU should not send any new discovery, read, or write commands until after receiving the GATT Write Completed event.

Table 35 GATT Command Write Request

Item	Description
Operating code	0x07
Parameters	Connection handle (2 bytes)
	Attribute handle (2 bytes)
	Data (variable bytes)

Figure 7 shows a GATT Command Write Request sequence where the peer device does not require involvement from its MCU.

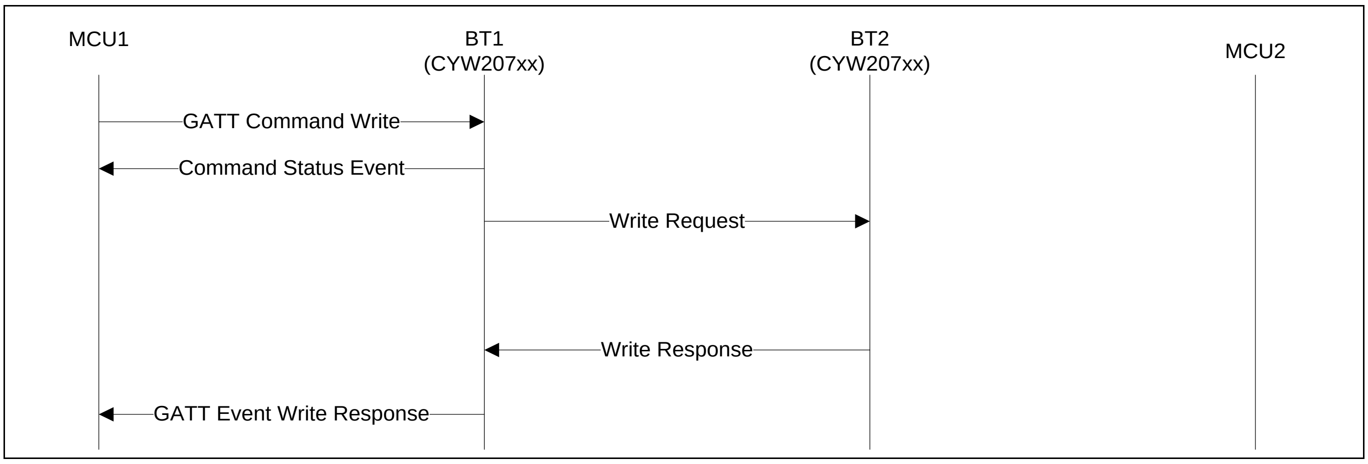

Figure 7 GATT Command Write Request – peer MCU not involved in the write

Figure 8 shows a GATT Command Write Request sequence where the peer device requires involvement from its MCU before executing the write.

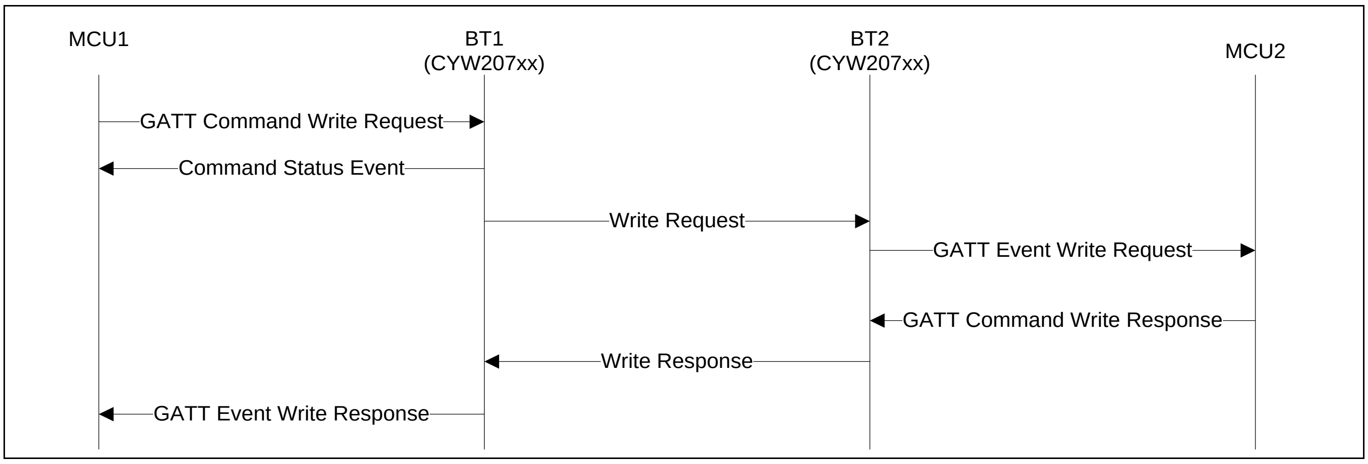

Figure 8 GATT Command Write Request – MCU is involved in a write

5.3.8 GATT Command Write Response

The GATT command Write Response command is used to confirm a received Write Request from a peer device. The connection handle and attribute handle should match the parameters received in GATT Event Write Request (see

GATT Event Write Response). See **Figure 8** for an example message sequence where this command is used.

When the command is received over the UART, the *hci_control* application sends a GATT Event Write Response for the attribute handle received in the command.

Table 36 GATT Command Write Response

Item	Description
Operating code	0x08
Parameters	Connection handle (2 bytes)
	Attribute handle (2 bytes)
	Status (1 byte) *Note:* *Application status codes are typically 0x80 and higher.*

5.3.9 GATT Command Notify

The GATT Command Notify lets an MCU schedule the sending of a notify packet over an established LE connection. The connection handle is a two-byte value reported in the LE Connected event (see **LE Connected**).

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist or the device is busy performing another action, such as discovery, reading, writing, and so on, then it reports the relevant status in the GATT Command Write Request event (see **Command Status**).
- If the connection exists and the device is not busy performing another action, then it reports Success in the Command Status event and starts the notification process.

The *hci_control* application sends a notification with the attribute handle received in the command.

The CYWxxxxx has a fixed number of transmit buffers. If the *hci_control* application allocates a buffer and schedules it for transmission, then the GATT Command Notify operation is considered complete and the *hci_control* application sends the GATT Event Write Response (see

GATT Event Write Response). If no transmit buffers are available, then the *hci_control* application saves the notification data and delays sending the GATT Event Write Response until a transmit buffer becomes available and the data is scheduled for transmission. The MCU should not send new discovery, read, or write commands until after receiving the GATT Event Write Response.

Table 37 GATT Command Notify

Item	Description
Operating code	0x09
Parameters	Connection handle (2 bytes)
	Attribute handle (2 bytes)
	Data (variable bytes)

Figure 9 shows a GATT Command Notify message sequence where a peer server (BT1) is prompted by its MCU (MCU1) to send a characteristic value notification to the client (BT2).

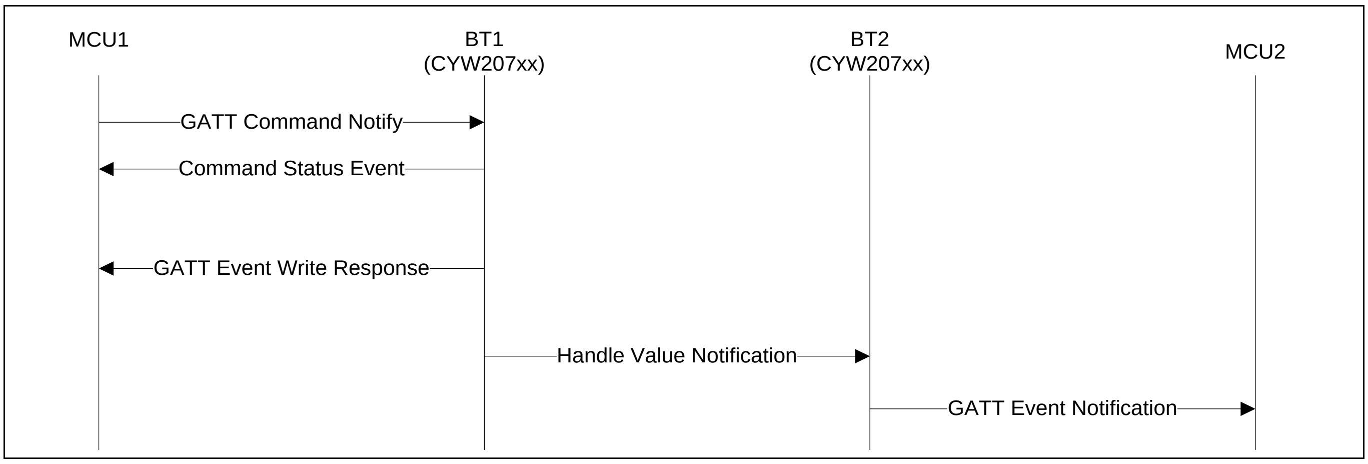

Figure 9 GATT Command Notify Message sequence

5.3.10 GATT Command Indicate

The GATT Command Indicate lets an MCU perform a value indication procedure over an established LE connection. The connection handle is a two-byte value reported in the LE Connected event (see **LE Connected**).

When the CYWxxxxx receives and processes this command, it takes one of the following actions:

- If the connection does not exist or the device is busy performing another action, such as discovery, reading, writing, etc., then it reports the relevant status in the Command Status event (see **Command Status**).
- If the connection exists and the device is not busy performing another action, then it reports success in the Command Status event and starts the indication process.

The hci_control application sends a Handle value indication with the attribute handle received in the command. When a handle value confirmation is received from the connected device, the *hci_control* application sends the GATT Event Write Response (see

GATT Event Write Response). The MCU should not send new discovery, read, or write commands until after receiving the GATT Event Write Response.

Table 38 GATT Command Indicate

Item	Description
Operating code	0x0A
Parameters	Connection handle (2 bytes)
	Attribute handle (2 bytes)
	Data

Figure 10 shows a GATT Command Indicate message sequence where a peer server (BT1) is prompted by its MCU (MCU1) to send a characteristic value indication to the client (BT2).

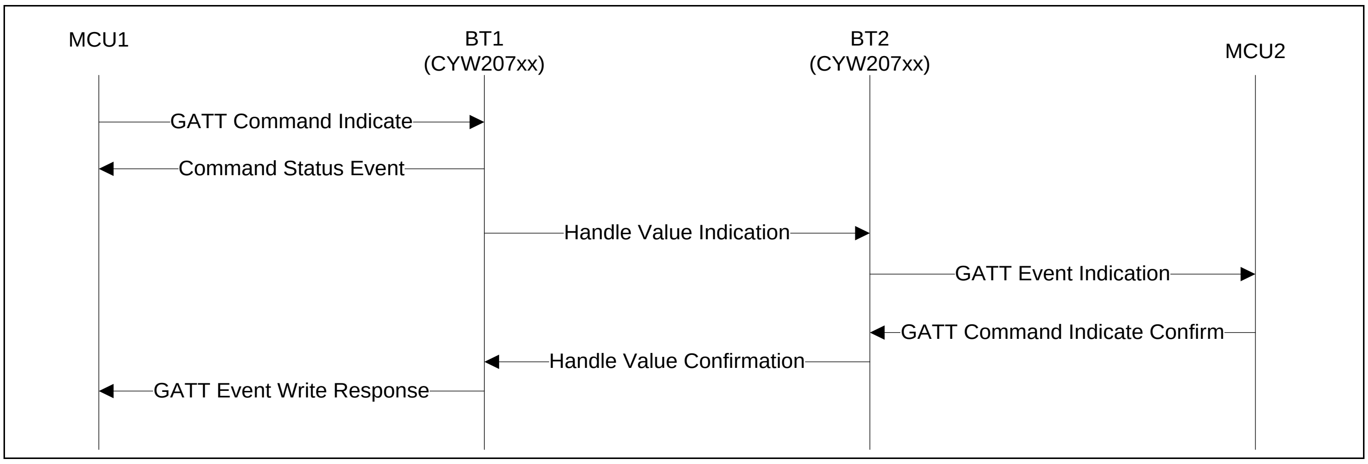

Figure 10 GATT Command Indicate message sequence

5.3.11 GATT Command Indicate Confirm

The GATT Command Indicate Confirm allows the MCU send a confirmation to an indication received from a peer device. The connection handle and attribute handle should match the parameters received in the GATT Event Indicate event (see **GATT Event Indication**).

When the command is received over the UART, the *hci_control* application sends a Handle value confirmation (see **Figure 10**) for the attribute handle received in the command.

Table 39 GATT Command Indicate Confirm

Item	Description
Operating code	0x0B
Parameters	Connection handle (2 bytes)
	Attribute handle (2 bytes)

5.3.12 GATT DB Add Primary Service

The GATT DB Add Primary Service command instructs the GATT DB app on CYWxxxxx to add a desired primary service into the GATT Database.

Table 40 GATT DB Add Primary Service

Item	Description
Operating code	0x0C
Parameters	Service handle (2 bytes)
	UUID (16/128 bits)

5.3.13 GATT DB Add Secondary Service

The GATT DB Add Secondary Service command instructs the GATT DB app on CYWxxxxx to add a desired secondary service into the GATT Database.

Table 41 GATT DB Add Secondary Service

Item	Description
Operating code	0x0D
Parameters	Service handle (2 bytes)
	UUID (16/128 bits)

5.3.14 GATT DB Add Included Service

The GATT DB Add Included Service command instructs the GATT DB app on CYWxxxxx to add a desired included service into the GATT Database.

Table 42 GATT DB Add Included Service

Item	Description
Operating code	0x0E
Parameters	Included service handle (2 bytes)
	Service handle (2 bytes)
	End group (2 bytes)

5.3.15 GATT DB Add Characteristic

The GATT DB Add Characteristic command instructs the GATT DB app on CYWxxxxx to add characteristic into the GATT Database.

Table 43 GATT DB Add Characteristic

Item	Description
Operating code	0x0F
Parameters	Service handle (2 bytes)
	Handle value (2 bytes)
	Property (2 bytes)
	Permission (2 bytes)

5.3.16 GATT DB Add Descriptor

The GATT DB Add Descriptor command instructs the GATT DB app on CYWxxxxx to add descriptor into the GATT Database.

Table 44 GATT DB Add Descriptor

Item	Description
Operating code	0x10
Parameters	Handle (2 bytes)
	Permission (1 byte)

5.4 Hands-Free commands— HCI_CONTROL_GROUP_HF

The Hands-Free (HF) commands instructs the MCU to perform various HF procedures using the CYWxxxxx.

5.4.1 HF Connect

The HF Connect command instructs the CYWxxxxx to establish a connection to a specified Audio Gateway (AG), which is typically a phone.

When a connection is established, the CYWxxxxx sends an HF Open event. The status field of that event tells whether the connection could be established or not.

Table 45 HF Connect command

Item	Description
Operating code	0x01
Parameters	AG Bluetooth® device address (6 bytes)

When the CYWxxxxx receives and processes this command, it:

- Allocates a handle for the connection.
- Starts paging the AG using the passed-in address.
- Establishes a Hands-Free profile-defined Service Level Connection (SLC) if the connection is created.
- Sends an HF Open event with the connection assigned handle and success/failure status.
- Sends an HF Connected event if the SLC gets established.

5.4.2 HF Disconnect

The HF Disconnect command instructs the CYWxxxxx to remove an existing connection to an AG.

Table 46 HF Disconnect command

Item	Description
Operating code	0x02
Parameters	Connection handle (2 bytes)

When the CYWxxxxx receives and processes this command, it disconnects the connection identified by the passed handle. When the connection is disconnected, it sends an HF Closed event.

5.4.3 HF Open Audio

The HF Open Audio command instructs the CYWxxxxx to create an audio connection on the AG identified by the connection handle.

Table 47 HF Open Audio command

Item	Description
Operating code	0x03
Parameters	Connection handle (2 bytes)

When the CYWxxxxx receives and processes this command, it attempts to open an audio connection on the AG identified by the passed connection handle. When an audio connection is established, it sends an HF Audio Open event.

5.4.4 HF Close Audio

The HF Close Audio command instructs the CYWxxxxx to close the audio connection on the AG identified by the connection handle.

Table 48 HF Close Audio command

Item	Description
Operating code	0x04
Parameters	Connection handle (2 bytes)

When the CYWxxxxx receives and processes this command, it attempts to close an audio connection on the AG identified by the passed handle connection. When the audio connection is closed, it sends an HF Audio Close event.

5.4.5 HF Accept/Reject Audio Connection

The HF Accept/Reject Audio Connection command instructs the CYWxxxxx to accept or reject the SCO connection request on the AG identified by SCO index.

Table 49 HF Accept/Reject Audio Connection command

Item	Description	
Operating code	0x05	
Parameters	SCO index (2 bytes)	
	Flag (1 byte)	0: Reject audio Connection request 1: Accept audio connection request

When the CYWxxxxx receives and processes this command, it attempts to accept/reject the SCO connection on the AG identified by the passed handle connection.

5.4.6 HF Turn off PCM Clock

The HF Turn Off PCM Clock command instructs the CYWxxxxx to turn off the PCM clock and reset the PCM settings. This command should be sent on receiving HF Audio Close. This command lets the MCU mute the codec output (or send other commands to codec chip) before CYWxxxxx stops the clock to avoid undesirable artifacts when audio stops.

Table 50 HF Turn Off PCM Clock command

Item	Description
Operating code	0x06
Parameters	-

5.4.7 HF AT commands

Each HF AT command instructs the CYWxxxxx to send a specific AT command to an AG.

Table 51 HF AT command

Item	Description
Operating code	See **Table 51**
Parameters	Connection handle (2 bytes)
	Command code (1 byte)
	Numeric value (2 bytes)
	Optional supporting character string (variable bytes)

Table 51 shows various available settings for the command code, numeric value, and optional string parameters of the HF AT command.

Table 52 HF AT command parameters

Command code		Numeric value	Optional string
Code	**Description**		
0x20	Speaker gain	0–15	–
0x21	Microphone gain	0–15	–
0x22	Answer incoming call	–	–
0x23	Get number from voice tag	1	–
0x24	Voice recognition	0: Disable 1: Enable	–
0x25	Last number redial	–	–
0x26	Call hold	0: Release all held calls 1: Release all active calls 2: Swap active and held calls 3: Hold active call	–
0x27	Hang up	–	–
0x28	Read indicator status	–	–

Command code		Numeric value	Optional string
Code	**Description**		
0x29	Retrieve subscriber number	–	–
0x2A	Dial	–	–
0x2B	Noise/echo control	0: Disable 1: Enable	–
0x2C	Transmit DTMF tone	–	–
0x2D	Response and hold	0: Hold incoming call 1: Accept held incoming call 2: Reject held incoming call	–
0x2E	Get operator information	-	–
0x2F	Extended result codes	1: Enable	–
0x30	Get current call list	–	–
0x31	Indicator control	–	–
0x32	Send HF indicator	–	–
0x33	Send proprietary AT command	–	–

When the CYWxxxxx receives and processes this command, it attempts to send the corresponding AT command to the AG identified by the connection handle. When a response is received from the AG, it is sent back via an HF Response event (see **HF Response**). Another command should not be sent until after the response event is received.

5.5 Serial Port Profile commands—HCI_CONTROL_GROUP_SPP

The Serial Port Profile (SPP) commands instructs the MCU to establish the SPP connection to a peer and send data.

5.5.1 SPP Connect

The MCU can send the SPP Connect command to the CYWxxxxx to establish SPP connection to a specified device. Upon receiving the command, the CYWxxxxx establishes an ACL data connection, performs a Service Discovery Protocol (SDP) search for the RFCOMM service, and establishes an RFCOMM connection to that service.

If the operation is successful, the CYWxxxxx will send the SPP connected event to the MCU. If the operation fails, the SPP Connection Failed or SPP Service Not Found event is sent.

Table 53 SPP Connect command

Item	Description
Operating code	0x01
Parameters	Bluetooth® device address of the peer device (6 bytes)

5.5.2 SPP Disconnect

The MCU can send an SPP Disconnect command to disconnect a previously established SPP connection.

Table 54 SPP Disconnect command

Item	Description	
Operating code	0x02	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the SPP Connected event.

5.5.3 SPP Data

The MCU issues the SPP Data command to send data over an established SPP connection.

Upon receiving an SPP Data command, the CYWxxxxx attempts to allocate a buffer and queue a data packet for transmission. After the packet is enqueued, the CYWxxxxx sends the TX Completed event. If the queue is full because data is received over the UART faster than it can be delivered to the peer, then the TX Completed event is delayed until the operation can be completed.

Table 55 SPP Data command

Item	Description	
Operating code	0x03	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the SPP connected event.
	Data (variable bytes)	-

5.6 Audio commands— HCI_CONTROL_GROUP_AUDIO

The audio commands let an MCU establish an AV source connection to a peer device over the AVDT protocol and then send data.

5.6.1 Audio Connect

The MCU can send an Audio Connect command to the CYWxxxxx to establish an AV source connection to a specified device. Upon receiving the command, the CYWxxxxx establishes an ACL data connection, performs Service Discovery Protocol (SDP) searches for the A2DP service, and establishes an AVDTP signaling connection and the data channel.

If the operation succeeds, the CYWxxxxx will send the Audio Connected event to the MCU. If the operation fails, the Audio Connection Failed event, or Audio Service Not Found event is sent.

Table 56 Audio connect command

Item	Description
Operating code	0x01

Item	Description	
Parameters	Address (6 bytes)	Bluetooth® device address of the peer device.
	Audio route (1 byte)	0: I^{2}S
		1: UART
		2: Sine (sends a sine wave)

5.6.2 Audio Disconnect

The MCU can send an Audio Disconnect command to disconnect a previously established an AV source connection.

Table 57 Audio Disconnect command

Item	Description	
Operating code	0x02	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio connected event.

5.6.3 Audio Start

The MCU can send an Audio Start command to the CYWxxxxx to start streaming audio from the MCU to the remote device. Upon receiving the command, if the CYWxxxxx determines that it's appropriate and necessary, it reconfigures the channel for a new sampling frequency or channel mode. If successful, it begins requesting raw audio data from the MCU.

The MCU can send an Audio Start command only after an audio connection to the peer device has been established; that is, after an Audio Connected event has been received (see **Audio Connected**).

If the MCU was previously streaming data and it issued the Audio Stop command (see **Audio Stop**), it should not send another Audio Start command until after it receives the Audio Stopped event (see **Audio Stopped**).

Sending the Audio Start command configures the CYWxxxxx for specific stream settings, including sample frequency and channel mode. Configured parameters will persist across stream suspend and resume.

If the peer device disconnects and then reconnects (see **Audio Connected** and **Audio Disconnected**), the CYWxxxxx will not start streaming until the MCU resends an Audio Start command.

If the operation is successful, then the CYWxxxxx will send an Audio Started event (see **Audio Started**) back to the MCU. If the operation fails, then an Audio Stopped event (see **Audio Stopped**) will be sent.

Table 58 Audio Start command

Item	Description	
Operating code	0x03	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Connected event.
	Sampling frequency	0: 16 kHz

Item	Description	
	(1 byte)	1: 32 kHz
		2: 44.1 kHz
		3: 48 kHz
	Channel mode (1 byte)	0: Mono
		1: Stereo

5.6.4 Audio Stop

The MCU can send an Audio Stop command to the CYWxxxxx to stop streaming audio from the MCU, through the platform, to the remote device. Upon receiving the command, the CYWxxxxx stops requesting audio data buffers from the MCU. When the CYWxxxxx finishes sending queued data, it will send the Audio Stopped event (see **Audio Stopped**) to the MCU and, upon timeout (if not restarted), it will place the AVDTP connection in a suspended state.

Table 59 Audio Stop command

Item	Description	
Operating code	0x04	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Connected event.

After sending the Audio Stop command, the MCU should not send an Audio Start command until after it receives an Audio Stop event.

5.6.5 Audio Data

The MCU can send an Audio Data command in response to an Audio Data Request event (see **Audio Data Request**). The Audio Data Request indicates the bytes per packet and number of packets that the MCU needs to send.

An Audio Data command from the MCU carries high-priority, real-time data. The type of raw PCM data (stereo/mono, sampling frequency) is set by the MCU in the Audio Start Command (see **Audio Start**).

Table 60 Audio Data command

Item	Description	
Operating code	0x06	
Parameters	PCM data packet length (2 bytes)	-
	PCM data (variable bytes)	Each 16-bit audio sample is 2 bytes.

5.6.6 Audio Read Statistics

The MCU can send an Audio Read Statistics command to CYWxxxxx to read audio data streaming statistics. Upon receiving the command, the CYWxxxxx read the audio statistics and it will send Audio Statistics Event in response.

Table 61 Audio Read Statistics command

Item	Description
Operating code	0x07
Parameters	-

5.7 HID Device commands: HCI_CONTROL_GROUP_HIDD

The HID Device (HIDD) commands let the MCU perform various HIDD-related procedures using the CYWxxxxx.

5.7.1 HID Accept Pairing

The HID Accept Pairing command instructs the CYWxxxxx to enter or exit a discoverable and connectable mode. When the CYWxxxxx is in a discoverable and connectable mode, peer devices can find the device and establish a bonding relationship with it.

Table 62 HID Accept Pairing command

Item	Description
Operating code	0x01
Parameters	Enable (1 byte)

When a peer device establishes a connection, the HID Opened event (see **HID Opened**) will be sent to the MCU. At that time, the MCU can start sending HID reports.

5.7.2 HID Send Report

When a connection is established, the MCU can send a HID report over the HID interrupt or control channel. The report should be a fully formatted packet, including the report ID and the data.

Table 63 HID Send Report command

Item	Description	
Operating code	0x02	
Parameters	Report channel (1 byte)	0: Control 1: Interrupt
	Report type (1 byte)	0: Other 1: Input 2: Output 3: Feature
	Report data (variable bytes)	

If the CYWxxxxx is not connected to a paired host when it receives a HID send report command, it will try to establish a HID connection. When this happens, the report will be lost.

5.7.3 HID Push Pairing Host Info

If the CYWxxxxx is not connected to an external serial flash, then the MCU is responsible for storing the paired host information. At start-up, the MCU should download the paired host information that it previously received in an NVRAM Data event (see **NVRAM Data**).

Table 64 HID Push Pairing Host Info command

Item	Description	
Operating code	0x03	
Parameters	Data (variable bytes)	Data received in the NVRAM Data event.

5.7.4 HID Connect

The HID Connect command instructs the CYWxxxxx to establish a connection to a previously paired HID host. Prior to issuing this command, information about the host, including the Bluetooth® device address and link key, should be downloaded to the CYWxxxxx using the HID Push Pairing Host Info command.

When a connection is established, the CYWxxxxx sends a HID Opened event (see **HID Opened**).

Table 65 HID Connect command

Item	Description	
Operating code	0x04	
Parameters	Address (6 bytes)	Bluetooth® device address of the HID host to which a connection is made.

5.8 AV Remote Control Target commands: HCI_CONTROL_GROUP_AVRC_TARGET

5.8.1 AVRC Target Connect

Note: *This command should only be used in the case of PTS testing. Target side connections are made in conjunction with an audio source connections.*

The MCU can send this to the CYWxxxxx to establish an AV remote control target connection to a specified device. Upon receiving this command, the CYWxxxxx establishes an ACL data connection if one does not exist, performs the Service Discovery Protocol (SDP), searches for the AVRC service, and establishes an AVCTP channel.

If the operation succeeds, the CYWxxxxx sends the AVRC Connected event (see **AVRC Controller Connected**) back to the MCU. If the operation fails, the CYWxxxxx sends the AVRC Disconnected event (see **AVRC Controller Disconnected**).

Table 66 AVRC Target Connect command

Item	Description	
Operating code	0x01	
Parameters	bd_addr (6 bytes)	Bluetooth® address of the peer device.

5.8.2 AVRC Target Disconnect

Note: This command should only be used in the case of PTS testing. Target side connections are made in conjunction with an audio source connections.

The MCU can send this command to disconnect a previously established AV remote control connection.

Table 67 AVRC Target Disconnect command

Item	Description
Operating code	0x02
Parameters	-

5.8.3 AVRC Target Track Information

The MCU can send this command to inform the CYWxxxxx of the updates to be performed in the track information for the currently playing track. The MCU shall send information about all changed attributes in a single command. The command can include all attributes or it can be limited to one or several attributes. For example, one could send title and track number if only those attributes have changed. If an attribute is not available for the new track, the MCU should include the attribute with length of zero.

Table 68 AVRC Target Track Information command

Item	Description	
Operating code	0x05	
Parameters	Attribute Id of the first attribute (1 byte)	1: Title 2: Artist 3: Album 4: Track number 5: Number of tracks 6: Genre 7: Playing time
	Attribute length of the first attribute (1 byte)	Attribute string length
	Attribute value of the first attribute (n bytes as defined above)	Attribute value expressed as a character string.
	Attribute Id of the second attribute (1 byte)	(see list above)
	Attribute length of the second attribute (1 byte)	Attribute string length
	Attribute value of the second attribute (n bytes as defined above)	Attribute value expressed as a character string.
	… (up to 7 entries)	…

5.8.4 AVRC Target Player Status

The MCU can send this command to the CYWxxxxx with the player play state and track position information. It is mandatory for the MCU to send this status update for every change in the playback status of the local player. If last reported status is playing, the CYWxxxxx will assume that the track position on the player is continuously updating 1000 ms every second. The MCU must send the track position only if changes are not due to the standard playback. For example, the command needs to be sent regularly if the player is performing fast forward or rewind operations, or if the position jumps due to the local update on the player.

Table 69 AVRC Target Player Status command

Item	**Description**	
Operating code	0x06	
Parameters	Play state	0x00: STOPPED 0x01: PLAYING 0x02: PAUSED 0x03: FWD_SEEK 0x04: REV_SEEK
	Track length (4 bytes)	Length of the current track in milliseconds
	Track position (4 bytes)	Position in the current track in ms within Track Length defined above.

5.8.5 AVRC Target Repeat Mode Changed

The MCU can send this command to the CYWxxxxx to inform the CYWxxxxx of a change in the mode of the local player repeat setting.

Table 70 AVRC Target Repeat Mode Changed command

Item	**Description**	
Operating code	0x07	
Parameters	Repeat mode	0x01: Off 0x02: Single track repeat 0x03: All track repeat 0x04: Group repeat

5.8.6 AVRC Target Shuffle Mode Changed

The MCU can send this command and inform CYWxxxxx of a change in the mode of the local player shuffle setting.

Table 71 AVRC Target Shuffle Mode Changed command

Item	Description	
Operating code	0x08	
Parameters	Shuffle mode	0x01: Off 0x02: All track scan 0x04: Group scan

5.8.7 AVRC Target Equalizer Status Changed

The MCU can send this command andinform the CYWxxxxx of a toggle in the ON or OFF status of the local player equalizer.

Table 72 AVRC Target Equalizer Status Changed command

Item	Description	
Operating code	0x09	
Parameters	Equalizer status	0x01: Off 0x02: On

5.8.8 AVRC Target Scan Mode Changed

The MCU can send this command to the CYWxxxxx to reflect the change of the status of the local player scan control setting.

Table 73 AVRC Target Scan Mode Changed command

Item	Description	
Operating code	0x0A	
Parameters	Scan Mmde	0x01: Off 0x02: All track scan 0x04: Group scan

5.8.9 AVRC Target Register for Notification

The MCU can send this command to the CYWxxxxx to register for notifications.

Table 74 AVRC Target Register for Notification command

Item	Description
Operating code	0x99
Parameters	-

5.9 AV Remote Control Controller commands: HCI_CONTROL_GROUP_AVRC_CONTROLLER

The AV Remote Control controller group of the commands are used by the MCU when implementing a remote-control application. For example, the MCU can send play, pause and other commands to the remote connected Bluetooth® player.

5.9.1 AVRC Controller Connect

The MCU can send this to the CYWxxxxx to establish an AV remote control connection to a specified device. Upon receiving this command, the CYWxxxxx establishes an ACL data connection if one does not exist yet, performs the Service Discovery Protocol (SDP), searches for the AVRC service, and establishes an AVCTP channel.

If the operation succeeds, the CYWxxxxx sends the AVRC Connected event to the MCU. If the operation fails, the CYWxxxxx sends an AVRC Disconnected event.

Note: *This command should only be used in the case of a standalone AVRC controller application. If remote controller functionality is combined with the speaker, the AVRC command will be established automatically when audio connection is established.*

Table 75 AVRC Controller Connect command

Item	Description	
Operating code	0x01	
Parameters	bd_addr (6 bytes)	Bluetooth® address of the peer device

5.9.2 AVRC Controller Disconnect

The MCU can send this command to disconnect a previously established AV remote control connection.

Table 76 AVRC Controller Disconnect command

Item	Description	
Operating code	0x02	
Parameters	bd_addr (6 bytes)	Bluetooth® address of the peer device

5.9.3 AVRC Controller Play

The MCU sends this command to start playing an audio on the connected Bluetooth® media player.

Table 77 AVRC Controller Play command

Item	Description	
Operating code	0x03	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.4 AVRC Controller Stop

The MCU sends this command to stop playing an audio on the connected Bluetooth® media player.

Table 78 AVRC Controller Stop command

Item	**Description**	
Operating code	0x04	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.5 AVRC Controller Pause

The MCU sends this command to pause playing audio on the connected Bluetooth® media player.

Table 79 AVRC Controller Pause command

Item	**Description**	
Operating code	0x05	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.6 AVRC Controller Begin Fast Forward

The MCU sends this command to begin fast forward operation on the connected Bluetooth® media player. Unlike most of the other AVRC commands, this command initiates the mode where the player plays audio at high speed. Use the AVRC End Fast Forward command to terminate this mode.

Table 80 AVRC Controller Begin Fast Forward command

Item	**Description**	
Operating code	0x06	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.7 AVRC Controller End Fast Forward

The MCU sends this command to terminate fast forward operation on the connected Bluetooth® media player.

Table 81 AVRC Controller End Fast Forward command

Item	Description	
Operating code	0x07	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.8 AVRC Controller Begin Rewind

The MCU sends this command to begin rewind operation on the connected Bluetooth® media player. Unlike most of the other AVRC commands, this command initiates the mode where the player plays audio in reverse at high speed. Use the AVRC End Rewind command to terminate this mode.

Table 82 AVRC Controller Begin Rewind command

Item	Description	
Operating code	0x08	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC connected event (see **AVRC Controller Connected**).

5.9.9 AVRC Controller End Rewind

The MCU sends this command to terminate rewind operation on the connected Bluetooth® media player.

Table 83 AVRC Controller End Rewind command

Item	Description	
Operating code	0x09	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.10 AVRC Controller Next Track

The MCU sends this command to instruct the player to move to the next track on the connected Bluetooth® media player.

Table 84 AVRC Controller Next Track command

Item	**Description**	
Operating code	0x0A	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.11 AVRC Controller Previous Track

The MCU sends this command to instruct the player to move to the previous track on the connected Bluetooth® media player.

Table 85 AVRC Controller Previous Track command

Item	**Description**	
Operating code	0x0B	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.12 AVRC Controller Volume Up

The MCU can send this command to the CYWxxxxx to request a volume increase on a connected AV player.

Table 86 AVRC Controller Volume Up command

Item	**Description**	
Operating code	0x0C	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.13 AVRC Controller Volume Down

The MCU can send this command to the CYWxxxxx to request a volume decrease on a connected AV player.

Table 87 AVRC Controller Volume Down command

Item	Description	
Operating code	0x0D	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

5.9.14 AVRC Controller Get Track Information

This is an optional command that the MCU can send to a CYWxxxxx to retrieve the current track information from the target player. The CYWxxxxx sends a request for the current track attributes to the peer. When the player responds the CYWxxxxx will send an event to the MCU for each of the track elements that it has retrieved. This can be invoked at any time or the MCU can choose to do so when informed by the CYWxxxxx of a track change (see **AVRC Controller Track Change**).

Table 88 AVRC Controller Get Track Information command

Item	Description	
Operating code	0x0E	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Connected event.
	Number of attributes (1 byte)	Number of attributes to return. 0 to return all attributes.
	Attributes (1 to 8 bytes)	Each byte represents an attribute to retrieve. Attribute values indicate the following options: 1: Title 2: Artist 3: Album 4: Track number 5: Number of tracks 6: Genre 7: Playing time

5.9.15 AVRC Controller Set Equalizer Status

The MCU can send this command to the CYWxxxxx to toggle the on/off state of the target player equalizer. The CYWxxxxx reports the initial state of the equalizer and subsequent state changes using the AVRC Setting Change event (see **AVRC Controller Setting Change**).

Table 89 AVRC Controller Set Equalizer Status command

Item	Description	
Operating code	0x0F	
Parameters	Connection handle (2 bytes)	The connection handle as reported in an Audio Connected event (see **Audio Connected**).

5.9.16 AVRC Controller Set Repeat Mode

The MCU can send this command to the CYWxxxxx to change the repeat mode of the target player. Each command submitted by the MCU will change the setting to the next available on the remote, cycling through all possible settings one at a time. The CYWxxxxx reports the initial repeat-mode state and subsequent state changes using the AVRC Setting Change event (see **AVRC Controller Setting Change**).

Table 90 AVRC Controller Set Repeat Mode Command

Item	Description	
Operating code	0x10	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Connected event (see **Audio Connected**).

5.9.17 AVRC Controller Set Shuffle Mode

The MCU can send this command to the CYWxxxxx to change the shuffle mode of the target player. Each command submitted by the MCU will change the setting on the remote to the next available setting, cycling through all possible settings one at a time. The CYWxxxxx reports the initial shuffle-mode state and subsequent state changes using the AVRC Setting Change event (see **AVRC Controller Setting Change**).

Table 91 AVRC Controller Set Shuffle Mode command

Item	Description	
Operating code	0x11	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Connected event (see **Audio Connected**).

5.9.18 AVRC Controller Set Scan Status

The MCU can send this command to the CYWxxxxx to change the scan status of the target player. Each command submitted by the MCU will change the setting on the remote to the next available setting, cycling through all possible settings one at a time. The CYWxxxxx reports the initial scan status and subsequent status changes in the AVRC Setting Change event (see **AVRC Controller Setting Change**).

Table 92 AVRC Controller Set Scan Status command

Item	Description	
Operating code	0x12	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Connected event (see **Audio Connected**).

5.9.19 AVRC Controller Set Volume

The MCU can send this command to the CYWxxxxx to pass a new volume setting to the connected AV sink device. An MCU should use this command only if the Absolute volume capable flag is true as indicated in the Audio Connected event (see **Audio Connected**).

Table 93 AVRC Controller Set Volume command

Item	Description	
Operating code	0x13	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Connected event (see **Audio Connected**).
	Volume level (1 byte)	The percentage (0 to 100) of the maximum volume level to be used by a connected peer device.

5.9.20 AVRC Controller Get Play Status

The MCU can send this command to the CYWxxxxx to get the status of the currently playing media at the TG.

Table 94 AVRC Controller Get Play Status command

Item	Description	
Operating code	0x014	
Parameters	Connection handle (2 bytes)	The connection handle reported in the AVRC Connected event.

5.9.21 AVRC Controller Power Passthrough command

The MCU can send this command to the CYWxxxxx to invoke AVRC Controller Power Passthrough command.

Table 95 AVRC Controller Power Passthrough command

Item	Description	
Operating code	0x015	
Parameters	-	-

5.9.22 AVRC Controller Mute Passthrough command

The MCU can send this command to the CYWxxxxx to invoke AVRC Controller Mute Passthrough command.

Table 96 AVRC Controller Mute Passthrough command

Item	Description	
Operating code	0x016	
Parameters	-	-

5.9.23 AVRC Controller Button Press

The MCU can use this command to simulate a button press on a stereo headphone.

Table 97 AVRC Controller Button Press command

Item	Description	
Operating code	0x017	
Parameters	-	-

5.9.24 AVRC Controller Long Button Press

The MCU can use this command to simulate a long button press on a stereo headphone.

Table 98 AVRC Controller Long Button Press command

Item	Description	
Operating code	0x018	
Parameters	-	-

5.9.25 AVRC Controller Unit Info

The MCU can send this command to the CYWxxxxx to send an AVRC Controller Unit Info command.

Table 99 AVRC Controller Send Unit Info command

Item	Description	
Operating code	0x019	
Parameters	Connection handle (2 bytes)	The connection handle reported in the AVRC Connected event.

5.9.26 AVRC Controller Send Sub Unit Info

The MCU can send this command to the CYWxxxxx to send an AVRC Sub Unit Info command.

Table 100 AVRC Controller Send Sub Unit Info command

Item	Description	
Operating code	0x01A	
Parameters	Connection Handle (2 bytes)	The connection handle reported in the AVRC Connected event.

5.10 Test commands— HCI_CONTROL_GROUP_TEST

The Test commands allow the host to execute various tests on the CYWxxxxx.

5.10.1 Encapsulated HCI command

Primarily for manufacturing test purposes, this test command allows the host to send encapsulated HCI commands to the CYWxxxxx to control the Bluetooth® controller for RF test purposes. For example, Bluetooth® LE RF testing usually requires the support of the LE Transmitter Test, LE Receiver Test, and LE Test End HCI commands (see Bluetooth® specification version 4.2 [Vol 2, Part E], Section 7.8.28, 7.8.29, and 7.8.30 [2] for details). All of which can be formatted into this Encapsulated HCI command.

The CYWxxxxx also provides support for vendor-specific commands (*Radio_Tx_Test* and *Radio_Rx_Test*) which enable a connectionless transmit and receive mode to send and receive respectively Bluetooth® packets at a specified frequency. See the WMBT tool included in with the *wiced_btsdk* project under *<USER_HOME>\mtw\mtb_shared\wiced_btsdk\tools\btsdk-utils\<git-branch>\wmbt*.

When the CYWxxxxx receives a test command, it is put into a Test Mode. While in the Test Mode all the events received from the controller are passed to the MCU as Encapsulated HCI events (see **Encapsulated HCI**) and not processed by the stack. Because of that the CYWxxxxx must be reset and reinitialized to continue normal application operation.

Table 101 Encapsulated HCI command

Item	Description	
Operating code	0x10	
Parameters	HCI command (variable bytes)	Fully formatted HCI command.

5.11 ANCS commands: HCI_CONTROL_GROUP_ANCS

The Apple Notification Control Service (ANCS) commands let the MCU perform various ANCS-related procedures using the CYWxxxxx. See the Apple ANCS specification [3] for more information.

5.11.1 ANCS Action

This command instructs the CYWxxxxx to pass a positive or negative action with respect to a specific notification sent by the iOS device. The command is sent after the CYWxxxxx reports the notification in the ANCS Notification event (see **ANCS Notification**).

Table 102 ANCS Action command

Item	Description	
Operating code	0x01	
Parameters	Notification ID (4 bytes)	The notification ID as reported in the ANCS Notification event.
	Action (1 byte)	0: Positive action. 1: Negative action.

5.11.2 ANCS Connect

This command instructs the CYWxxxxx to activate the ANCS service on the iOS device connected to the given LE Connection Handle. The MCU should not send this command until after it has received the ANCS Service Found event and has verified that the LE connection is Encrypted since the ANCS service requires authentication.

Table 103 ANCS Connect command

Item	Description	
Operating code	0x02	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.

5.11.3 ANCS Disconnect

This command instructs the CYWxxxxx to deactivate the ANCS service on the iOS device connected to the given LE Connection Handle by unsubscribing to notifications for the service.

Table 104 ANCS Disconnect command

Item	Description	
Operating code	0x03	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.

5.12 AMS commands: HCI_CONTROL_GROUP_AMS

The Apple Media Service (AMS) commands instructs the MCU perform various AMS-related procedures using the CYWxxxxx. Refer to the Apple developer AMS Specification [4] for more information:

5.12.1 AMS Connect

This command instructs the CYWxxxxx to activate the AMS service on the iOS device connected to the given LE Connection Handle. The MCU should not send this command until after it has received the AMS Service Found event and has verified that the LE Connection is encrypted since the AMS service requires authentication.

Table 105 AMS Connect command

Item	Description	
Operating code	0x01	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.

5.12.2 AMS Disconnect

This command instructs the CYWxxxxx to deactivate the AMS service on the iOS device connected to the given LE Connection Handle by unsubscribing to notifications for the service.

Table 106 AMS Disconnect command

Item	Description	
Operating code	0x02	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.

5.13 iAP2 commands: HCI_CONTROL_GROUP_IAP2

The Apple iPod Accessory Protocol (iAP2) commands allows MCU to establish and send data over an iAP2 External Accessory (EA) session implemented on a CYWxxxxx.

5.13.1 IAP2 Connect

The MCU can send this command to the CYWxxxxx to establish an EA session with a specified device. Upon receiving the command, the CYWxxxxx establishes an ACL data connection, performs an SDP search for the iAP2 service, and establishes an EA session to the iAP2 service.

After the EA session is successfully established, the CYWxxxxx will send an IAP2 Connected event (see **IAP2 Connected**) back to the MCU. If the operation fails, then either the IAP2 Connection Failed event (see **IAP2 Connection Failed**) or IAP2 Service Not Found event (see **IAP2 Service Not Found**) is sent.

Table 107 IAP2 Connect command

Item	Description	
Operating code	0x01	
Parameters	bd_addr (6 bytes)	Bluetooth® address of the peer device.

5.13.2 IAP2 Disconnect

The MCU sends this command to disconnect a previously established EA session.

Table 108 IAP2 Disconnect command

Item	**Description**	
Operating code	0x02	
Parameters	Session handle (2 bytes)	The session handle as reported in the IAP2 Connected event (see **IAP2 Connected**).

5.13.3 IAP2 Data

The MCU issues this command to send data over an established EA session.

Upon receiving this command, the CYWxxxxx attempts to allocate a buffer and queue a data packet for transmission. After successfully enqueuing a packet, the CYWxxxxx sends the IAP2 TX Complete event (see **IAP2 TX Complete**). If the queue is full because data is received over the UART faster than it can be delivered to the peer, then the sending of the TX Completed event is delayed until after the packet is successfully enqueued.

Table 109 IAP2 Data command

Item	**Description**	
Operating code	0x03	
Parameters	Session handle (2 bytes)	The session handle as reported in the IAP2 Connected event (see **IAP2 Connected**).
	Data (variable bytes)	Data to be transmitted to the iOS device.

5.13.4 IAP2 Get Auth Chip Info

The MCU sends this command to read the chip information from the authentication coprocessor connected to the CYWxxxxx.

Table 110 IAP2 Get Auth Chip Info command

Item	**Description**
Operating code	0x04
Parameters	-

5.14 Hands-Free AG commands: HCI_CONTROL_GROUP_AG

The Hands-Free AG commands instructs the MCU to establish signaling and audio connections to a peer Hands-Free device. The current version of the protocol defined in this document supports a simple implementation that can be used only for voice control and not for actual calls, conferences, and more.

5.14.1 AG Connect

the MCU sends this command to the CYWxxxxx to establish a Bluetooth® Hands-Free audio gateway connection to a specified device. Upon receiving the command, the CYWxxxxx establishes an ACL data connection, performs an SDP search for the RFCOMM service, establishes a connection with the RFCOMM service, and establishes a signaling connection with the specified Hands-Free device.

After an AG connection is successfully established, the CYWxxxxx will send the AG Connected event (see

AG Connected) back to the MCU.

Table 111 AG Connect Command

Item	Description	
Operating code	0x01	
Parameters	bd_addr (6 bytes)	Bluetooth® address of the peer device.

5.14.2 AG Disconnect

The MCU sends this command to disconnect a previously established AG signaling connection.

Table 112 AG Disconnect command

Item	Description	
Operating code	0x02	
Parameters	Session handle (2 bytes)	The session handle as reported in the AG Connected event (see AG Connected).

5.14.3 AG Audio Connect

The MCU sends this command to establish an audio channel over a previously established AG signaling connection.

Table 113 AG Audio Connect command

Item	Description	
Operating code	0x03	
Parameters	Session handle (2 bytes)	The session handle as reported in the AG Connected event (see AG Connected).

5.14.4 AG Audio Disconnect

The MCU sends this command to disconnect the audio channel previously established over the AG signaling connection.

Table 114 AG Audio Disconnect command

Item	Description	
Operating code	0x04	
Parameters	Session handle (2 bytes)	The session handle as reported in the AG Connected event (see AG Connected).

5.15 AIO Server Commands: HCI_CONTROL_GROUP_AIO_SERVER

The Automation IO (AIO) server commands instructs the MCU to perform various AIO server procedures using the CYWxxxxx.

5.15.1 AIO Digital Input

This command allows the MCU to simulate a change in an AIO server digital input.

Table 115 AIO Digital Input command

Item	**Description**	
Operating code	0x01	
Parameters	Index (1 byte)	Index of digital IO, starting with 0.
	Data (variable bytes)	An array of 2-bit values in a bit field in little endian order, which identifies the state of the digital input. 00: Inactive state 01: Active state 10: Tristate 11: Unknown state

After a CYWxxxxx receives this command, it sets the new value in the database and, if a value/time trigger is set and the condition is met, sends a notification or indication with the new value to the AIO client.

5.15.2 AIO Analog Input

This command allows the MCU to indicate a change in an AIO server analog input value.

Table 116 AIO Analog Input command

Item	Description	
Operating code	0x02	
Parameters	Index (1 byte)	Index of digital IO, starting with 0.
	Data (2 bytes)	The value of the analog signal as an unsigned 16-bit integer.

After a CYWxxxxx receives this command, it sets the new value in the database and, if a value/time trigger is set and the condition is met, sends a notification or indication with the new value to the AIO client.

5.16 AIO Client commands: HCI_CONTROL_GROUP_AIO_CLIENT

The Automation IO client commands instruct the MCU to perform various AIO client procedures using the CYWxxxxx.

5.16.1 AIO Connect

This command instructs the AIO client on a CYWxxxxx to connect to an AIO server.

Table 117 AIO Connect command

Item	Description	
Operating code	0x01	
Parameters	bd_addr (6 bytes)	Bluetooth® address of the AIO server to which a connection is made.

After the CYWxxxxx receives this command, it tries to establish a connection to the specified AIO server. If a Bluetooth® device address is not specified or set to all zeros, it starts LE scanning and connects to the first AIO server it finds. After a connection is established, the CYWxxxxx performs characteristic and characteristic descriptor discoveries.

5.16.2 AIO Read

This command instructs the AIO client on a CYWxxxxx to read a value from the AIO server.

Table 118 AIO Read command

Item	Description	
Operating code	0x02	
Parameters	Type (1 byte)	1: Analog IO 2: Digital IO 3: Aggregate IO

Item	Description	
	Index (1 byte)	Index of the analog, digital, or aggregate IO, starting with 0.

After a CYWxxxxx receives this command, it sends a read request to the AIO server. After a read response comes back from the AIO server, the CYWxxxxx will send the value back to the MCU in an AIO Read Response event (see **AIO Read Response**).

5.16.3 AIO Write

This command instructs the AIO client on a CYWxxxxx to write a value to the AIO server.

Table 119 AIO Write command

Item	Description	
Operating code	0x03	
Parameters	Type (1 byte)	1: Analog IO 2: Digital IO
	Index (1 byte)	Index of the analog or digital IO, starting with 0.
	Data (variable bytes)	An unsigned 16-bit integer for analog IO, or an array of 2-bit values in a bit field for digital IO.

After the CYWxxxxx receives this command, it sends a write request to the AIO server.

5.16.4 AIO Register for Notification

This command instructs the AIO client on a CYWxxxxx to register for notification or indication on the AIO server.

Table 120 AIO Register for Notification command

Item	Description	
Operating code	0x04	
Parameters	Type (1 byte)	1: Analog IO 2: Digital IO 3: Aggregate IO
	Index (1 byte)	Index of the analog or digital IO, starting with 0.
	Value (1 byte)	0: Unregister notification/indication 1: Register for notification 2: Register for indication

After a CYWxxxxx receives this command, it sends a write request to the AIO server to set a client characteristic configuration descriptor. The notification and/or indication configuration is set through a combination of the client characteristic configuration descriptor and the value and/or time trigger settings. See **AIO Set Value Trigger** and **AIO Set Time Trigger** for information regarding setting value and time triggers.

5.16.5 AIO Set Value Trigger

This command instructs the AIO client on a CYWxxxxx to set a value trigger on the AIO server.

Table 121 AIO Set Value Trigger command

Item	Description	
Operating code	0x06	
Parameters	Type (1 byte)	1: Analog IO 2: Digital IO
	Index (1 byte)	Index of the analog or digital IO, starting with 0.
	Condition (1 byte)	0: Value changed 1: Crossed boundary 2: On the boundary 3: Value change exceeds a set value 4: Mask then compare 5: Crossed boundaries 6: On the boundaries 7: No value trigger
	Values (variable bytes)	These bytes are a function of the condition set. They represent one or more boundaries or a set value.

After a CYWxxxxx receives this command, it sends a write request to an AIO server to set a value trigger descriptor.

5.16.6 AIO Set Time Trigger

This command instructs the AIO client on a CYWxxxxx to set a time trigger on the AIO server.

Table 122 AIO Set Time Trigger command

Item	Description	
Operating code	0x07	
Parameters	Type (1 byte)	1: Analog IO 2: Digital IO
	Index (1 byte)	Index of the analog or digital IO, starting with 0.
	Condition (1 byte)	0: No time trigger 1: Periodic 2: Not more often than a set time 3: Value changed N times, where N is a count that can be set.

Item	Description	
	Values (variable bytes)	These bytes are a function of the condition set.

After a CYWxxxxx receives this command, it sends a write request to the AIO server to set a time trigger descriptor.

5.16.7 AIO Set User Description

This command instructs the AIO client on a CYWxxxxx to set the user description on the AIO server.

Table 123 AIO Set User Description command

Item	Description	
Operating code	0x08	
Parameters	Type (1 byte)	1: Analog IO 2: Digital IO
	Index (1 byte)	Index of the analog or digital IO, starting with 0.
	Description (variable bytes)	User description

5.16.8 AIO Disconnect

This command instructs the AIO client on a CYWxxxxx to disconnect from the AIO server.

Table 124 AIO Disconnect command

Item	Description
Operating code	0x09
Parameters	-

After a CYWxxxxx receives this command, it terminates its connection with the AIO server.

5.17 Audio Sink commands: HCI_CONTROL_GROUP_AUDIO_SINK

The Audio Sink commands let the MCU establish an AV sink connection to a peer device over the AVDT protocol.

5.17.1 Audio Sink Connect

The Audio Sink Connect command instructs the CYWxxxxx to establish an AV Sink connection to a specified device. Upon receiving the command, the CYWxxxxx establishes an ACL data connection, performs Service Discovery Protocol (SDP) searches for the A2DP service, and establishes an AVDTP signaling connection. It is source responsibility to discover and configure streaming endpoint.

If the operation succeeds, the CYWxxxxx will send the Audio Sink Connected event to the MCU. If the operation fails, the Audio Sink Connection Failed event, or Audio Sink ServiceNot Found event is sent.

Table 125 Audio Sink Connect command

Item	Description	
Operating code	0x01	
Parameters	Address (6 bytes)	Bluetooth® device address of the peer device.

5.17.2 Audio Sink Disconnect

The Audio Sink Disconnect command instructs the CYWxxxxx to disconnect a previously established AV sink connection.

Table 126 Audio Sink Disconnect command

Item	Description	
Operating code	0x02	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Sink Connected event.

5.17.3 Audio Sink Start

The Audio Sink Start command instructs the CYWxxxxx to send start audio streaming request from the MCU to the remote device. Upon receiving the command, if the CYWxxxxx determines that it's appropriate and necessary, it configures the codec settings, audio route and sends AVDTP START request to peer device.

The MCU can send an Audio Sink Start command only after an audio sink connection to the peer device has been established; that is, after an Audio Sink Connected event has been received (see **Audio Sink Connected**).

If the MCU was streaming data and issued the Audio Sink Stop command (see **Audio Sink Stop**), it should not send another Audio Sink Start command until it receives the Audio sink Stopped event (see **Audio Sink Stopped**).

If the operation is successful, then the CYWxxxxx will send the Audio Sink Started event (see **Audio Sink Started**) back to the MCU. If the operation fails, the Audio Stopped event (see **Audio Sink Stopped**) will be sent.

Table 127 Audio Sink Stop command

Item	Description	
Operating code	0x03	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Sink Connected event.

5.17.4 Audio Sink Stop

The Audio Sink Stop command instructs the CYWxxxxx to stop streaming the audio to the remote device. Upon receiving the command, the CYWxxxxx resets codec and sends AVDTP SUSPEND request to the peer. When the CYWxxxxx receives AVDTP SUSPEND CONFIRM event from peer, it sends the Audio Sink Stopped event (see **Audio Sink Stopped**) to the MCU.

Table 128 Audio Sink Stop command

Item	Description	
Operating code	0x04	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Sink Connected event.

5.17.5 Audio Sink Start Response

The Audio Sink Start Response command instructs the CYWxxxxx to acceptor reject the audio stream sSstart request identified by connection handle.

Table 129 Audio Sink Start Response command

Item	Description	
Operating code	0x05	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Sink Connected event.
	Label (1 byte)	Transaction Label received in Audio Sink Start Indication event.
	Flag (1 byte)	0: Accept Audio Start Streaming Request 1: Reject Audio Start Streaming Request

When the CYWxxxxx receives and processes this command, it attempts to accept or reject the audio stream start request identified by the passed handle.

5.17.6 Audio Sink Change Route

The Audio Sink Change Route command instructs the CYWxxxxx to change the output data route identified by connection handle.

The MCU can send an Audio Sink Change Route command only after an Audio Sink Connection to the peer device has been established and before audio streaming is started. This command should be sent on receiving Audio Sink Codec Configured event.

Table 130 Audio Sink Change Route

Item	Description	
Operating code	0x06	
Parameters	Connection handle (2 bytes)	The connection handle as reported in the Audio Sink Connected event.
	Route (1 byte)	0: I2S - Route the PCM samples over I2S (default) 1: UART - Route the PCM samples over transport 4: Compressed transport - Route the compressed audio data over transport.

5.18 LE COC commands: HCI_CONTROL_GROUP_LE_COC

The LE COC commands let an MCU establish a LE COC connection and send/receive data over LE COC channels.

5.18.1 Connect

The Connect command instructs the CYWxxxxx to establish a LE COC connection to a specified device. Upon receiving the command, the CYWxxxxx establishes a LE COC connection on the PSM set by the MCU.

If the operation succeeds, the CYWxxxxx will send the LE COC Connected event back to the MCU.

Table 131 LE COC Connect

Item	Description	
Operating code	0x01	
Parameters	Address (6 bytes)	Bluetooth® device address of the peer device.

5.18.2 Disconnect

The LE COC Disconnect command instructs the CYWxxxxx to disconnect a previously established COC connection.

Table 132 LE COC Disconnect

Item	Description	
Operating code	0x02	
Parameters	Address (6 bytes)	Bluetooth® device address of the peer device.

5.18.3 Send Data

The LE COC Send Data command instructs the CYWxxxxx to send data to peer on the established COC connection.

Table 133 LE COC Send data

Item	Description	
Operating code	0x03	
Parameters	Data (max 256 bytes)	Data to be sent to peer

5.18.4 Set MTU

The LE COC Set MTU command instructs the CYWxxxxx to indicate the MTU supported to peer during connection establishment.

Table 134 Set MTU

Item	Description	
Operating code	0x04	
Parameters	MTU (2 bytes)	MTU

5.18.5 Set PSM

The LE COC Set PSM command instructs the CYWxxxxx to establish connection on the given PSM.

Table 135 Set PSM

Item	Description	
Operating code	0x05	
Parameters	PSM (2 bytes)	PSM

5.18.6 Enable LE2M

The LE COC Enable LE2M command instructs the CYWxxxxx to use LE 2M PHY instead of 1M PHY for data transfer.

Table 136 Set LE2M

Item	Description	
Operating code	0x06	
Parameters	enable (2 bytes)	1- LE2M enable 2 - LE2M disable

5.19 ANS commands: HCI_CONTROL_GROUP_ANS

The Alert Notification Service (ANS) commands let an MCU perform various ANS-related procedures using the CYWxxxxx.

5.19.1 Set Supported New Alert Category

The Set Supported New Alert Category command instructs the CYWxxxxx to configure the new alerts. New alert type would include simple alert, SMS, Email, news etc.

Table 137 Set Supported New Alert Category

Item	Description	
Operating code	0x01	
Parameters	New alert (2 bytes)	Supported new alert categories.

5.19.2 Set Supported Unread Alert Category

The Set Supported Unread Alert Category command instructs the CYWxxxxx to configure the unread alerts. Unread alert type would include simple alert, SMS, Email, news, and so on.

Table 138 Set Supported Unread Alert Category

Item	Description	
Operating code	0x02	
Parameters	Unread alert (2 bytes)	Supported unread alert categories

5.19.3 Generate Alerts

The Generate Alerts command instructs the CYWxxxxx to generate a specific type of alert. This can be used to generate both New Alert and Unread Alert of a specific category.

Table 139 Generate Alerts

Item	Description	
Operating code	0x03	
Parameters	Alert (1 byte)	Generate the Required type of alert.

5.19.4 Clear Alerts

The Clear Alert command instructs the CYWxxxxx to clear the previously generated alert count. This can be used to clear previously received alert count for both New Alert and Unread Alert of a specific category.

Table 140 Clear Alerts

Item	Description	
Operating code	0x04	
Parameters	Alert (1 byte)	Clear the required type of alert.

5.20 ANC commands: HCI_CONTROL_GROUP_ANC

The Alert Notification Client commands instructs the MCU to perform various ANC related procedures using the CYWxxxxx.

5.20.1 Read Server Supported New Alerts

The Read Server Supported New Alerts command instructs CYWxxxxx to read the supported new alert category from Alert Notification Server.

Note: *New alerts radio button should be selected before issuing the command.*

Table 141 Read Server Supported New Alerts

Item	Description
Operating code	0x01

5.20.2 Read Server Supported Unread Alerts

The Read Server Supported Unread Alerts command instructs CYWxxxxx to read the supported unread alert category from Alert Notification Server.

Note: *Unread alerts radio button should be selected before issuing the command.*

Table 142 Read Server Supported Unread Alerts

Item	Description
Operating code	0x02

5.20.3 Control Alerts

The Control Alert command instructs CYWxxxxx to enable, disable, notify all the pending alerts.

Table 143 Control Alerts

Item	Description	
Operating code	0x03	
Parameters	Command ID (1 byte)	This is the type of alert (new alert/ unread alert) which needs to be sent.
	Alert category (1 byte)	This indicates the specific alert type category. (e.g. Email, SMS, simple alert)

5.20.4 Enable New Alert

The Enable New Alert command instructs CYWxxxxx to enable new alert type notifications.

Table 144 Enable New Alert

Item	Description
Operating code	0x04

5.20.5 Enable Unread Alert

The Enable Unread Alert command instructs CYWxxxxx to enable unread alert type notifications.

Table 145 Enable Unread Alert

Item	Description
Operating code	0x05

5.20.6 Disable New Alert

The Disable New Alert command instructs CYWxxxxx to disable new alert type notifications.

Table 146 Disable New Alert

Item	Description
Operating code	0x06

5.20.7 Disable Unread Alert

The Disable Unread Alert command instructs CYWxxxxx to disable unread alert type notifications.

Table 147 Disable Unread Alert

Item	Description
Operating code	0x07

5.21 DFU commands: HCI_CONTROL_GROUP_DFU

The DFU commands are used to perform Firmware upgrades via HCI.

5.21.1 Get Configuration

This DFU command requests the configuration information from the device that can be useful for firmware upgrades over HCI. Once issued, the expected response is a DFU Configuration event containing the requested data. Currently, the expected response is the efficient size that should be used for subsequent data transfers. This size will vary depending on the flash memory used with the device and kit.

Table 148 Get Configuration command

Item	Description
Operating code	0x00

5.21.2 Write command

This DFU command passes a command to the Firmware upgrade library by calling `hci_fw_upgrade_handle_command()`. The first octet after the operating code is the command code. Depending on the command code further parameters may be needed.

Table 149 Write command

Item	Description	
Operating code	0x01	
Parameters	Prepare download (1 byte) [1]	Prepare for firmware update.
	Download (1 +4 bytes) [2 x x x x]	Start data transfer, where number of bytes to be transferred is provided as “x”.
	Verify (1 + 4 bytes) [3 y y y y]	Verify data transfer. If verifying with CRC-32, final crc value is passed as “y”.
	Abort (1 byte) [7]	Stop firmware download process. Return firmware upgrade library to initial state.

5.21.3 Send Data

This DFU command is used to transfer data to the Firmware Upgrade library. This data is expected to be consecutive blocks of the firmware upgrade binary, except for the last transfer the size should be limited to the value returned by the Get Configuration response event.

Table 150 Send Data command

Item	Description
Operating code	0x02
Parameters	Data (variable bytes)

5.22 Miscellaneous commands: HCI_CONTROL_GROUP_MISC

The miscellaneous commands allow the host to send the general commands as defined by the CYWxxxxx.

5.22.1 Ping Request

This miscellaneous command sends a Ping Request to the CYWxxxxx. The application running on the CYWxxxxx is expected to respond back with a Ping Reply event (see **Ping Request Reply**). The Ping Reply event is expected to return the data sent as part of the Ping Request.

Table 151 Ping Request Command

Item	Description
Operating code	0x01
Parameters	Data (variable bytes)

5.22.2 Get Version

This miscellaneous command requests the CYWxxxxx to report the ModusToolbox™ version used to build the embedded application. The application running on the CYWxxxxx is expected to respond back with a Version Info event (see **Version Info**).

Table 152 Get Version command

Item	Description
Operating code	0x02

6 AIROC™ HCI Control Protocol events

6.1 Device events: HCI_CONTROL_GROUP_DEVICE

The device events are general events and state transitions reported by the CYWxxxxx.

6.1.1 Command Status

The Command Status event indicates to the MCU that execution of the command has been started or that a command has been rejected due to the state of the *hci_control* application.

Table 153 Command Status event

Item	Description	
Operating code	0x01	
Parameters	Status (1 byte)	0: Execution of the command has started.
		1: The command has been rejected because the previous command is still executing.
		2: The Connect command has been rejected because the specified device is already connected.
		3: The Disconnect command has been rejected because the connection is down.
		4: The handle parameter in the command is invalid.
		5: The Discover, read, or write command has been rejected because the previous command has not finished executing.
		6: Invalid parameters passed in the command.
		7: Bluetooth® stack on CYWxxxxx failed to execute the command.
		8: Embedded application loaded on the CYWxxxxx does not support processing of the commands of the requested group
		9: Embedded application loaded on CYWxxxxx does not support the command requested by the MCU.
		10: LE application cannot send notification or indication because the GATT client is not registered.
		11: Out of memory.
		12: Operation disallowed.

6.1.2 AIROC™ Trace

When tracing is enabled (see **Trace Enable**), the CYWxxxxx sends `WICED_BT_TRACE` statements over UART for the MCU to display.

Table 154 AIROC™ Trace event

Item	**Description**
Operating code	0x02
Parameters	WICED_BT_TRACE statements (ASCII string)

6.1.3 HCI Trace

When tracing is enabled (see **Trace Enable**), the CYWxxxxx sends binary data with the HCI commands, events, and data over UART for the MCU to display.

Table 155 HCI Trace event

Item	**Description**	
Operating code	0x03	
Parameters	Type (1 byte)	0: HCI event 1: HCI command 2: Incoming HCI data 3: Outgoing HCI data
	Raw HCI bytes (variable bytes)	Data formatted according to the Bluetooth® Core Specification Vol. 2 Part E. [2]

6.1.4 NVRAM Data

For the situations when the CYWxxxxx does not have internal persistent storage, an application running on the CYWxxxxx can send data to the MCU in the NVRAM Data events.

Table 156 NVRAM Data event

Item	**Description**	
Operating code	0x04	
Parameters	nvram_id (2 bytes)	ID of the NVRAM information chunk
	nvram_data (variable bytes)	Data corresponding to the nvram_id

6.1.5 Device Started

The *hci_control* application sends a Device Started event at the end of application initialization. Upon receiving the event, the MCU can assume that there are no active connections. The application logic determines the initial Bluetooth® LE scanning or advertising state and whether the Bluetooth® device is discoverable and/or connectable.

Table 157 Device Started event

Item	Description
Operating code	0x05
Parameters	-

6.1.6 Inquiry result

The *hci_control* application sends an Inquiry Result event when the CYWxxxxx is performing the inquiry procedure and information is received about a discoverable peer device.

Table 158 Inquiry result event

Item	Description
Operating code	0x06
Parameters	Address (6 bytes)
	Class of device (CoD) (3 bytes)
	RSSI (1 byte)
	Extended inquiry response (EIR) data (variable bytes)

6.1.7 Inquiry complete

The *hci_control* application sends an Inquiry Complete event on completion of the inquiry process.

Table 159 Inquiry Complete Event

Item	Description
Operating code	0x07
Parameters	-

6.1.8 Pairing Completed

The *hci_control* application sends a Pairing Completed event when a secure bond with the peer device has been established or when an attempt to establish a bond has failed.

Table 160 Pairing Complete event

Item	Description	
Operating code	0x08	
Parameters	Pairing result (1 byte):	0: Success 1: Passkey Entry Failure 2: OOB Failure 3: Pairing Authentication Failure

Item	Description	
		4: Confirm Value Failure 5: Pairing Not Supported 6: Encryption Key Size Failure 7: Invalid Command 8: Pairing Failure Unknown 9: Repeated Attempts 10: Internal Pairing Error 11: Unknown I/O Capabilities 12: SMP Initialization Failure 13: Confirmation Failure 14: SMP Busy 15: Encryption Failure 16: Bonding Started 17: Response Timeout 18: Generic Failure 19: Connection Timeout
	Bluetooth® device address (6 bytes)	

6.1.9 Encryption Changed

The *hci_control* application sends an Encryption Changed event when a link to the peer device has been encrypted or when encryption has been turned OFF.

Table 161 Encryption Changed event

Item	Description	
Operating code	0x09	
Parameters	Encryption status (1 byte):	0: Encryption enabled Else: Encryption disabled
	Bluetooth® device address (6 bytes)	

6.1.10 Connected Device Name

The application running on the CYWxxxxx can send this command to inform the MCU of the friendly name of the connected device.

Table 162 Connected Device Name event

Item	Description
Operating code	0x0A
Parameters	A variable length UTF-8 string representing a peer's device name.

6.1.11 User Confirmation Request

The application running on the CYWxxxxx device can be written to support numerical-comparison pairing or require a user permission to pair with another device. For these cases, the application sends this event to the MCU.

Table 163 User Confirmation Request event

Item	Description
Operating code	0x0B
Parameters	Bluetooth® device address (6 bytes)
	Numeric comparison code (4 bytes)

6.1.12 Device Error

The CYWxxxxx sends this event when it runs into a situation where it cannot proceed and needs to reset to recover. This can occur if the controller or the embedded application detects that it has entered a bad state.

Table 164 Device Error event

Item	Description	
Operating code	0x0C	
Parameters	Application error code (1 byte)	Error code reported by application
	Firmware error code (1 byte)	Error code reported by controller

6.1.13 Local Bluetooth® Device Address

The CYWxxxxx sends this event in response to the Read Local Bluetooth® Device Address command.

Table 165 Local Bluetooth® Device Address event

Item	Description
Operating code	0x0D
Parameters	A 6-byte Bluetooth® device address

6.1.14 Maximum Number of Paired Devices Reached

The CYWxxxxx sends this event if the maximum number of keys stored for paired devices is reached. When this event occurs, the CYWxxxxx will also disable pairing since there are no more buffers available to store more pairing keys. The host will need to delete one or more NVRAM entries and enable pairing to pair with more devices.

Table 166 Maximum Number of Paired Devices Reached event

Item	Description
Operating code	0x0E
Parameters	-

6.1.15 Buffer Pool Usage Statistics

The CYWxxxxx sends this event when the Read Buffer Pool Usage Statistics is received. The Buffer Pool Usage Statistics event message provides the buffer pool usage since the start of the application running on the CYWxxxxx. This event message provides all buffer pool information such as the number of buffers allocated at the instance of receiving the Read Buffer Pool Usage Statistics command, the maximum number of buffers in use at a given time since the start of the system, and the total number of buffers in a pool. The actual number of pools are application dependent.

Table 167 Buffer Pool Usage Statistics event

Item	Description
Operating code	0x0F
Parameters	Pool ID (1 byte)
	Pool buffer size (2 byte)
	Current allocated count (2 bytes)
	Maximum allocated count (2 bytes)
	Total allocated count (2 bytes)

6.1.16 Key Press Notification event

The CYWxxxxx sends this event when the user has been entered or erased the keys on the peer device during passkey entry protocol pairing process.

Table 168 Key Press Notification event

Item	Description
Operating code	0x10
Parameters	Bluetooth® device address (6 bytes)
	1-byte Pass key entry notification type: 0 - PASSKEY_ENTRY_STARTED 1 - PASSKEY_DIGIT_ENTERED 2 - PASSKEY_DIGIT_ERASED 3 - PASSKEY_DIGIT_CLEARED 4 - PASSKEY_ENTRY_COMPLETED

6.1.17 Connection Status event

The CYWxxxxx sends this event when ACL connection status changed (i.e. ACL connection up/down).

Table 169 Connection Status event

Item	Description
Operating code	0x11
Parameters	1-byte Connection status: 0 – Not connected 1 - Connected
	1-byte reason 0 – success, else

Item	Description
	HCI error codes defined as per Core Bluetooth® specification.

6.2 LE events—HCI_CONTROL_GROUP_LE

The LE events are related to the Bluetooth® LE GAP profile and reported by the CYWxxxxx.

6.2.1 LE Command Status

This event indicates to the MCU that LE command execution has started or that a command has been rejected due to the state of the *hci_control* application.

Table 170 LE Command Status event

Item	Description	
Operating code	0x01	
Parameters	Status (1 byte)	See **Command Status**

6.2.2 LE Scan Status

The *hci_control* application sends a Scan Status event when the CYWxxxxx enters a new scanning state. A scanning state transition can be caused by a received LE Scan Command or internal application or stack logic.

Table 171 LE Scan Status event

Item	Description	
Operating code	0x02	
Parameters	State[1]	0 No scan
		1 High-duty-cycle scan
		2 Low-duty-cycle scan

6.2.3 LE Advertisement Report

The *hci_control* application sends an LE Advertisement Report event when the CYWxxxxx is scanning and it receives an advertisement or a scan response from a peer device.

Table 172 LE Advertisement Report event

Item	Description
Operating code	0x03
Parameters	Event type indicating the type of advertisement report (1 byte)
	Address type indicating the Bluetooth® address type (1 byte)
	Bluetooth® device address (6 bytes)
	RSSI of the advertisement (1 byte)
	Advertisement data (variable bytes)

[1] The high-duty-cycle and low-duty-cycle scan parameters for each state are defined in the wiced_bt_cfg.c file, which is included in every application.

6.2.4 LE Advertisement State

The *hci_control* application sends an LE Advertisement State event when the CYWxxxxx enters a new advertisement state. An advertisement state change can be caused by an LE Advertise Command received from the MCU or by internal application or stack logic.

Table 173 LE Advertisement State event

Item	Description	
Operating code	0x04	
Parameters	State[1]	0 Not Discoverable
		1 High-duty-cycle discoverable
		2 Low-duty-cycle discoverable

6.2.5 LE Connected

The *hci_control* application sends the LE Connected event when the CYWxxxxx establishes a connection with a peer Bluetooth® LE device identified by address type and address. The connection handle identifies the connection and can be used in consecutive requests to disconnect or transfer data. If the Role parameter is zero, then the CYWxxxxx performs in the central role in a newly established connection. Otherwise, the CYWxxxxx performs as a peripheral. If the CYWxxxxx is performing as a GATT client, then the MCU can issue the GATT Command Read Request, GATT Command Write, or GATT Command Write Request commands to send data to the peer. Otherwise, the GATT Command Notify or GATT Command Indicate commands should be used.

Table 174 LE Connected event

Item	Description	
Operating code	0x05	
Parameters	Type (1 byte)	Bluetooth® device address type.
	Address (6 bytes)	Bluetooth® device address
	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Role (1 byte)	The role is either peripheral or central.

[1] The advertisement intervals and durations for each state are defined in the wiced_bt_cfg.c file, which is included in every application.

6.2.6 LE Disconnected

When the Bluetooth® LE connection with a peer device is disconnected, the *hci_control* application sends the LE Disconnected event. The connection handle and disconnection reason are passed as parameters.

Table 175 LE Disconnected event

Item	Description	
Operating code	0x06	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Disconnection reason (1 byte)	-

6.2.7 LE Identity Address

When the LE Get Identity Address is called, the resolved Identity Address of the peer is returned via this event message.

Table 176 LE Identity Address event

Item	Description	
Operating code	0x07	
Parameters	Address (6 bytes)	Resolved Identity address

6.2.8 LE Peer MTU

When the CYWxxxxx receives a Client MTU Request, this event will be passed to the MCU indicating the negotiated MTU size.

Table 177 LE Peer MTU Event

Item	Description	
Operating code	0x08	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event
	MTU size (2 bytes)	

6.2.9 LE Connection Parameters

When the CYWxxxxx receives a connection update complete event from a peer device, this LE Connection Parameters event will be passed to the MCU indicating the negotiated connection parameters or error code reflected by the status byte.

Table 178 LE Connection Parameters event

Item	Description	
Operating code	0x09	
Parameters	Status (1 byte)	0: Success, Else: Failure
	Peer address (6 bytes)	
	Connection interval (2 bytes)	
	Connection latency (2 bytes)	
	Supervision timeout (2 bytes)	

6.3 GATT events

The GATT events are related to the GATT profile and reported by the CYWxxxxx.

6.3.1 GATT Command Status

This event indicates to the MCU that GATT command execution has started or that a command has been rejected due to the state of the *hci_control* application.

Table 179 GATT Command Status event

Item	Description	
Operating code	0x01	
Parameters	Status (1 byte)	See **Command Status**

6.3.2 GATT Discovery Complete

The GATT Discovery Complete event indicates to an MCU that all results from a previously issued GATT Discover Services, GATT Discover Characteristics, or GATT Discover Descriptors command have been delivered. After receiving this event, the MCU can start a new discovery procedure.

Table 180 GATT Discovery Complete event

Item	Description	
Operating code	0x02	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the Bluetooth® LE Connected event.

6.3.3 GATT Service Discovered

While performing a service discovery, the *hci_control* application sends the GATT Service Discovered event for every service found on a peer device. The connection handle identifies the connection to the peer device. The start and end handles identify the handles used by the service. The UUID identifies the remote service and can be either 2 or 16 bytes.

Table 181 GATT Service Discovered event

Item	Description	
Operating code	0x03	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	UUID (2 or 16 bytes)	The UUID of the discovered service.
	Start handle (2 bytes)	The start handle of the service.
	End handle (2 bytes)	The end handle of the service.

6.3.4 GATT Characteristic Discovered

While performing a characteristic discovery, the *hci_control* application sends the GATT Characteristic Discovered event for every characteristic discovered on the peer device. The connection handle identifies the connection to the peer device. The value handle can be used by the MCU in consecutive GATT Read, GATT Write command, GATT Write Request, GATT Notify, or GATT Indicate calls to send data to the peer. The UUID identifies the remote characteristic and can be either 2 or 16 bytes.

Table 182 GATT Characteristic Discovered event

Item	Description	
Operating code	0x04	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Characteristic handle (2 bytes)	-
	UUID (2 or 16 bytes)	The UUID of the characteristic found.
	Characteristic properties (1 byte)	A bit mask of the properties supported by the discovered characteristic.
	Value handle (2 bytes)	The characteristic-value handle that can be used in consecutive reads and write.

6.3.5 GATT Descriptor Discovered

While performing a characteristic descriptor discovery, the *hci_control* application sends the GATT Descriptor Discovered event for every characteristic descriptor discovered on the peer device. The connection handle identifies the connection to the peer device. The handle can be used by the MCU in consecutive GATT Read or GATT Write Request commands to set or get a descriptor value. The UUID identifies the remote descriptor and can be either 2 or 16 bytes.

Table 183 GATT Descriptor Discovered event

Item	Description	
Operating code	0x05	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	UUID (2 or 16 bytes)	The descriptor UUID.
	Handle (2 bytes)	The descriptor handle, which can be used in subsequent reads and writes.

6.3.6 GATT Event Read Request

The GATT Event Read Request can be sent to the MCU to provide the value of the specific attribute. The connection handle identifies the connection to the peer device requesting the operation and the attribute handle identifies the attribute requested by the peer device. Upon receiving this request, the MCU should send the GATT Command Read Response (see **GATT Command Read Response**).

Table 184 GATT Event Read Request

Item	Description	
Operating code	0x06	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Attribute handle (2 bytes)	The attribute handle of the value being read.

See **Figure 5** for a message sequence example where the GATT Event Read Request is used.

6.3.7 GATT Event Read Response

The GATT Event Read Response indicates to the MCU that the execution of the GATT Command Read Request has completed. The event includes the received data. The connection handle identifies the connection to the peer device for which the read procedure has been performed.

Table 185 GATT Event Read Response

Item	Description
Operating code	0x07

Item	Description	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Data (variable bytes	-

See **Figure 4** and **Figure 5** for message sequence examples where the GATT Event Read Response is used.

6.3.8 GATT Event Write Request

The GATT Event Write Request indicates to the MCU that a write request from a connected peer has been received. The connection handle identifies the connection of the peer device that issued the write request and the attribute handle identifies the characteristic to be written.

The CYWxxxxx application can be designed to wait for the GATT Command Write Response (see **GATT Command Write Response**) or to reply automatically to indicate the success of the write operation to the peer. Waiting for the GATT Command Write Response is required when the MCU needs to be able to reject peer write attempts.

Table 186 GATT Event Write Request

Item	Description	
Operating code	0x08	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Attribute handle (2 bytes)	The attribute handle of the value being written.
	Data (variable bytes)	-

See **Figure 8** for a message sequence example where the GATT Event Write Request is used.

6.3.9 GATT Event Write Response

The GATT Event Write Response indicates to the MCU that the execution of a GATT Command Write, GATT Command Write Request, GATT Command Notify, or GATT Command Indicate has completed. The event includes the result of the write operation. The connection handle identifies the connection to the peer device for which the procedure has been performed.

For the GATT Command Write Request and GATT Command Indicate commands, issuance of the GATT Event Write Response indicates that the write has completed and that the peer has confirmed receiving the data. For the GATT Command Write and GATT Command Notify commands, issuance of the GATT Event Write Response indicates that the buffer has been allocated and a command has been scheduled for transmission.

Table 187 GATT Event Write Response

Item	Description	
Operating code	0x09	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Result (1 byte)	-

See **Figure 8** for a message sequence example where the GATT Event Write Response is used.

6.3.10 GATT Event Indication

The GATT Event Indication event passes data received from a peer-sent GATT Indication to the MCU. The connection handle identifies the connection to the peer device from which the GATT Indication was received. The attribute handle identifies the characteristic value or descriptor to which data has been written.

The application running on the CYWxxxxx can behave in one of the following two ways after receiving a GATT Indication:

- It can reply automatically (with the success).
- In a flow-controlled scenario, it can pass the event up to the MCU and wait for the GATT Command Indicate Confirm from the MCU before replying.
- LE Connection Parameters event

Table 188 GATT Event Indication event

Item	Description	
Operating code	0x0A	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Attribute handle (2 bytes)	This the handle of the attribute being accessed.
	Data (variable bytes)	-

See **Figure 10** for a message sequence example where the GATT Event Indication is used.

6.3.11 GATT Event Notification

The GATT Event Notification forwards data received from a peer-sent GATT Command Notify to the MCU. The connection handle identifies the connection to the peer device from which the GATT Command Notify was received. The attribute handle identifies the characteristic value to which data has been written.

Table 189 GATT Event Notification event

Item	Description	
Operating code	0x0B	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Attribute handle (2 bytes)	This is the handle of the attribute being accessed.

See **Figure 9** for a message sequence example where the GATT Event Notification is used.

6.3.12 GATT Event Read Error

The GATT Event Read Error message will be sent to the MCU in the case where a GATT Read Request command resulted in an error. This event message will include the received read result GATT error code, for example, Insufficient authentication.

Table 190 GATT Event Read Error

Item	Description	
Operating code	0x0C	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Read result (1 byte)	Received GATT error code.

6.3.13 GATT Event Write Request Error

The GATT Event Write Request Error message will be sent to the MCU in the case where a GATT Write Request command resulted in an error. This event message will include the received read result GATT error code, for example, Insufficient authentication.

Table 191 GATT Event Write Request Error

Item	Description	
Operating code	0x0D	
Parameters	Connection handle (2 bytes)	This is the connection handle reported in the LE Connected event.
	Read result (1 byte)	Received GATT error code.

6.4 HF events: HCI_CONTROL_GROUP_HF

These events sent by the CYWxxxxx pertain to the functionality of the Hands-Free profile.

6.4.1 HF Open

This event is sent when an RFCOMM connection is established with an AG. At this point, the Service Level Connection (SLC) is still not established, so commands cannot yet be sent. The Bluetooth® device address and connection handle are passed as parameters. The connection handle can be used by the MCU to send commands or to identify a peer device that caused the event.

Table 192 HF Open event

Item	Description
Operating code	0x01
Parameters	Connection handle (2 bytes)
	Bluetooth® device address of the AG (6 bytes)
	Status (1 byte)

6.4.2 HF Close

This event is sent when an RFCOMM connection with an AG is closed.

Table 193 HF Close event

Item	Description
Operating code	0x02
Parameters	Connection handle (2 bytes)

6.4.3 HF Connected

This event is sent when the Hands-Free device and the AG have completed the protocol exchange necessary to establish an SLC. At this point, the application can send any commands to the CYWxxxxx.

Table 194 HF Connected event

Item	Description
Operating code	0x03
Parameters	Connection handle (2 bytes)
	32-bit mask of AG supported features

6.4.4 HF Audio Open

This event is sent when an audio connection with an AG is opened.

Table 195 HF Audio Open event

Item	Description
Operating code	0x04
Parameters	Connection handle (2 bytes)

6.4.5 HF Audio Close

This event is sent when an audio connection with an AG is closed.

Table 196 HF Audio Close event

Item	Description
Operating code	0x05
Parameters	Connection handle (2 bytes)

6.4.6 HF Audio Connection Request

This event is sent to the MCU on receiving an audio connection request from the AG. The MCU shall use the HF Accept/Reject Audio Connection command to accept/reject the connection request.

Table 197 HF Audio Connection Request event

Item	Description
Operating code	0x06
Parameters	Bluetooth® device address of the AG (6 bytes)
	SCO index (2 bytes)

6.4.7 HF Response

The HF Response events are sent when a response is received from the AG for a command sent by the application.

Table 198 HF Response event format

Item	Description
Operating code	See **Table 199**
Parameters	Connection handle (2 bytes)
	Numeric value (2 bytes)
	Optional supporting character string

Table 199 shows various available values for the operating code, numeric value, and optional string parameters of **Table 198**.

Table 199 HF Response event details

Operating Code		Numeric value	Optional string
Code	Description		
0x20	OK response	Command index of last command	-
0x21	Error response	Command index of last command	-
0x22	Extended error response	Command index of last command	Error code
0x23	Incoming call	-	-
0x24	Speaker gain	0–15	-

Operating Code		Numeric value	Optional string
Code	Description		
0x25	Microphone gain	0–15	-
0x26	Incoming call waiting	-	The calling party's number and number type. For example: "nnnnn, 128"
0x27	Call hold	0: Release all held calls 1: Release all active calls 2: Swap active and held calls 3: Hold active call	-
0x28	AG indicators	-	The AG indicators
0x29	Caller phone number	-	The caller's number
0x2A	AG indicator changed	-	The indicator number [1-7] and value. For example: "1,2" 1: Service indicator 2: Call status indicator 3: Call set up status indicator 4: Call hold status indicator 5: Signal Strength indicator 6: Roaming status indicator 7: Battery charge indicator
0x2B	Number attached to voice tag	-	Phone number. For example: "nnnnnn"
0x2C	Voice recognition status	0: VR disabled in AG 1: VR enabled in AG	-
0x2D	In-band ring tone	0: No AG in-band ring tone 1: AG provides in-band ring tone	-
0x2E	Subscriber number	-	The subscriber number and number type. For example: "nnnnn, 128"
0x2F	Call hold status	0: AG put incoming call on hold 1: AG accepted held incoming call 2: AG rejected held incoming call	-

Operating Code		Numeric value	Optional string
Code	Description		
0x30	Operator information	-	-
0x31	Active call list	-	List of active calls
0x32	Supported HF indicators	-	-
0x33	Bluetooth® Codec Selection	1: CVSD Codec 2: MSBC Codec	-
0x34	Unknown AT response	-	The unknown response that was received from the AG.

6.5 SPP events— HCI_CONTROL_GROUP_SPP

These events sent by the CYWxxxxx pertain to the functionality of the Serial Port Profile (SPP).

6.5.1 SPP Connected

This event is sent when an SPP connection has been established with a peer device. The Bluetooth® device address and connection handle are passed as parameters. The connection handle can be used by the MCU for future commands to send commands or data and to identify a peer device that has sent data.

Table 200 SPP Connected event

Item	Description
Operating code	0x01
Parameters	Bluetooth® device address (6 bytes)
	Connection handle (2 bytes)

6.5.2 SPP Service Not Found

This event is sent when a CYWxxxxx is able to connect to a peer device and perform SDP discovery, but the SPP service is not found.

Table 201 SPP Service Not Found event

Item	Description
Operating code	0x02
Parameters	-

6.5.3 SPP Connection Failed

A CYWxxxxx sends this event when a connection attempt requested by an MCU is unsuccessful.

Table 202 SPP Connection Failed event

Item	Description
Operating code	0x03
Parameters	-

6.5.4 SPP Disconnected

This event is sent when an SPP connection has been dropped.

Table 203 SPP Disconnected event

Item	Description
Operating code	0x04
Parameters	Connection handle (2 bytes)

6.5.5 SPP TX Complete

A CYWxxxxx sends this event after a data packet received from an MCU, in an SPP send data command, has been queued for transmission. The MCU should not send another data packet until it has received this event for the previous packet.

Table 204 SPP TX Complete event

Item	Description
Operating code	0x05
Parameters	Connection handle (2 bytes)
	Result (1 byte) 0 = Success, other result codes defined in ModusToolbox™ header file *wiced_bt_rfcomm.h* `wiced_bt_rfcomm_result_t` enum

6.5.6 SPP RX Data

A CYWxxxxx forwards SPP data received from a peer device in the SPP RX Data event.

Table 205 SPP RX Data event

Item	Description
Operating code	0x06
Parameters	Connection handle (2 bytes)
	Data received from the peer

6.5.7 SPP Command Status

This event indicates to the MCU that a SPP command execution has started or that a command has been rejected due to the state of the *hci_control* application.

Table 206 SPP Command Status event

Item	Description
Operating code	0x07
Parameters	Status (1 byte)
	See **Command Status**

6.6 Audio events—HCI_CONTROL_GROUP_AUDIO

These events sent by the CYWxxxxx pertain to audio (A2DP) profile functionality.

6.6.1 Audio Command Status

This event indicates to the MCU that an Audio command execution has started or that a command has been rejected due to the state of the *hci_control* application.

Table 207 Audio Command Status event

Item	Description
Operating code	0x01
Parameters	Status (1 byte)
	See **Command Status**

6.6.2 Audio Connected

This event is sent when an audio connection has been established with a peer device. The Bluetooth® device address and connection handle are passed as parameters. The connection handle can be used by the MCU to send commands or data, and to identify a peer device that has sent data.

The Absolute Volume Capable flag indicates to the MCU whether a peer device can accept commands to set the volume.

Table 208 Audio Connected event

Item	Description	
Operating code	0x02	
Parameters	Address (6 bytes)	Bluetooth® device address of peer.
	Connection handle (2 bytes)	The handle to use during command and data exchanges.
	Absolute volume capable (1 byte)	1: Peer can accept commands to set volume. 0: Peer cannot accept commands to set volume.

6.6.3 Audio Service Not Found

A CYWxxxxx sends this event when it is able to connect to a peer device and perform SDP discovery, but there is no A2DP service.

Table 209 Audio Service Not Found event

Item	Description
Operating code	0x03
Parameters	-

6.6.4 Audio Connection Failed

A CYWxxxxx sends this event when a connection attempt requested by the MCU is unsuccessful.

Table 210 Audio Connection Failed event

Item	Description
Operating code	0x04
Parameters	-

6.6.5 Audio Disconnected

A CYWxxxxx sends this event when an audio connection has been dropped.

Table 211 Audio Disconnected event

Item	Description	
Operating code	0x05	
Parameters	Connection handle (2 bytes)	
	Status (1 byte)	Reflects the internal stack API operation; should always be 0 for success.
	Reason (1 byte)	Reflects the HCI Disconnect reason

6.6.6 Audio Data Request

A CYWxxxxx sends this event when an audio stream is configured to send audio data over UART. The host is expected to maintain and send the number of packets requested as well as the number of bytes per packet.

Table 212 Audio Data Request event

Item	Description	
Operating code	0x06	
Parameters	Bytes per packet (2 bytes)	
	Number of packets (1 byte)	
	Total number of packets requested (2 bytes)	Total number of audio packets requested since the start of audio streaming, including the current number of packets request
	Total number of packets received (2 bytes)	Total number of audio packets received from the MCU

6.6.7 Audio Started

A CYWxxxxx sends this event when an audio stream has been started by an MCU-sent Audio Start command (see **Audio Start**).

Table 213 Audio Started event

Item	Description
Operating code	0x07
Parameters	Connection handle (2 bytes)

6.6.8 Audio Stopped

A CYWxxxxx sends this event when an audio stream has been stopped by an MCU-sent an audio stop command (see **Audio Stop**).

Table 214 Audio Stopped event

Item	Description
Operating code	0x08
Parameters	Connection handle (2 bytes)

6.6.9 Audio Statistics

A CYWxxxxx sends this event in response to the Audio Read Statistics command (see Audio Read Statistics).

Table 215 Audio Statistics event

Item	Description
Operating code	0x09
Parameters	Duration (4 bytes): Duration of the a2dp streaming in which stats were captured
	Table [11] (44 bytes): Delay between packet sent OTA and ack received Table [0] (4 bytes): Number of packets having delay between 0 – 9 msec Table [1] (4 bytes): Number of packets having delay between 10 - 19 msec … Table [9] (4 bytes): Number of packets having delay between 90 - 99 msec Table [10] (4 bytes): Number of packets having delay > 100 msec

6.7 AV Remote Control Controller events: HCI_CONTROL_GROUP_AVRC_CONTROLLER

6.7.1 AVRC Controller Connected

A CYWxxxxx sends the AVRC Connected event to an MCU when a peer device establishes an AVRC connection or after a connection requested by an AVRC Connect command has been successfully established.

Table 216 AVRC Controller Connected event

Item	Description	
Operating code	0x01	
	bd_addr (6 bytes)	Bluetooth® address of the connected player.
	Status (1 byte)	Status of the connection establishment event. If 0, then the connection has been established successfully.
	Session handle (2 bytes)	The session handle as reported in the AVRC Connected event.

6.7.2 AVRC Controller Disconnected

A CYWxxxxx sends the AVRC Disconnected event to an MCU to indicate that the AVRC connection has been terminated.

Table 217 AVRC Controller Disconnected event

Item	Description	
Operating code	0x02	
Parameters	Session handle (2 bytes)	The session handle as reported in the AVRC C qonnected event.

6.7.3 AVRC Controller Current Track Info

A CYWxxxxx sends this event when it receives information about new attributes of the track playing on the connected player. Each attribute reported by the player will be passed to the MCU in a separate AVRC Controller Current Track Info event.

Table 218 AVRC Controller Current Track Info event

Item	Description	
Operating code	0x03	
Parameters	Session handle (2 bytes)	The session handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

Item	Description	
	Status (1 byte)	AVRC Response Status
	Attribute ID (1 byte)	1: Title 2: Artist 3: Album 4: Track number 5: Number of tracks 6: Genre 7: Playing time
	Attribute length (2 bytes)	The length of the attribute data string.
	Data (variable bytes)	Attribute data string.

6.7.4 AVRC Controller Play Status

A CYWxxxxx sends the AVRC Controller Play Status event when a connected player reports a change in player status.

Table 219 AVRC Controller Play Status event

Item	Description	
Operating code	0x04	
Parameters	Session handle (2 bytes)	The session handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).
	Play status (1 byte)	0: Stopped 1: Playing 2: Paused 3: Forward seek 4: Reverse seek 255: Error

6.7.5 AVRC Controller Play Position

A CYWxxxxx sends an AVRC Controller Play Status event when a connected player reports a change in the play position.

Table 220 AVRC Controller Play Position event

Item	Description	
Operating code	0x05	
Parameters	Session handle (2 bytes)	The session handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).
	Play position (4 bytes)	The play position in milliseconds since the beginning of the track.

6.7.6 AVRC Controller Track Change

A CYWxxxxx sends an AVRC Controller Track Change event when a connected player reports a track change. It is incumbent upon the MCU to request the updated track information.

Table 221 AVRC Controller Track Change event

Item	Description	
Operating code	0x06	
Parameters	Session handle (2 bytes)	The session handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

6.7.7 AVRC Controller Track End

A CYWxxxxx sends an AVRC Controller Track End event when a connected player reports reaching the end of a track.

Table 222 AVRC Controller Track End event

Item	Description	
Operating code	0x07	
Parameters	Session handle (2 bytes)	The session handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

6.7.8 AVRC Controller Track Start

A CYWxxxxx sends an AVRC Controller Track Start event when a connected player reports starting a new track.

Table 223 AVRC Controller Track Start event

Item	Description	
Operating code	0x08	
Parameters	Session handle (2 bytes)	The session handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).

6.7.9 AVRC Controller Settings Available

A CYWxxxxx sends an AVRC Controller Settings Available event to report the player settings available for the connected player.

Table 224 AVRC Controller Settings Available event

Item	Description	
Operating code	0x09	
	Session handle (2 bytes)	The session handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).
	Settings (variable bytes)	An array of bytes indicating which attributes are supported by the connected player. Any value set in these bytes indicates that the setting is supported. The bits indicate the possible values for each setting: 1: The player supports an Equalizer. Bit 0: Unused Bit 1: Off supported Bit 2: On supported 2: The player supports Repeat mode. Bit 0: Unused Bit 1: Off supported Bit 2: Single track repeat supported Bit 3: All track repeat supported Bit 4: Group repeat supported 3: The player supports Shuffle mode. Bit 0: Unused Bit 1: Off supported Bit 2: All track shuffle supported Bit 4: Group shuffle supported 4: The player supports Scan mode.

Item	Description	
		Bit 0: Unused Bit 1: Off supported Bit 2: All track scan supported

6.7.10 AVRC Controller Setting Change

A CYWxxxxx sends an AVRC Controller Setting Change event to report the initial value or a setting change on a connected player.

Table 225 AVRC Controller Setting Change event

Item	Description	
Operating code	0x0A	
Parameters	Session handle (2 bytes)	The session handle as reported in the AVRC Connected event (see **AVRC Controller Connected**).
	Number of Settings (1 byte)	Number of ID-value pairs
	Setting ID (1 byte)	The following values indicate the ID of the player setting: 1: Equalizer. 2: Repeat mode. 3: Shuffle mode. 4: Scan mode.
	Setting value (1 byte)	For ID = 1 (Equalizer): 1: On 2: Off For ID = 2 (Repeat mode): 1: Off 2: Repeat a single track 3: Repeat all tracks 4: Repeat a group of tracks For ID = 3 (Shuffle mode): 1: Off 2: Shuffle all tracks 3: Shuffle a group of tracks For ID = 4 (Scan mode): 1: Off 2: Scan all tracks 3: Scan a group of tracks

6.7.11 AVRC Controller Player Change

A CYWxxxxx sends an AVRC Controller Player change event to report a change in the named connected player.

Table 226 AVRC Controller Player Change event

Item	Description
Operating code	0x0B
Parameters	Name (n bytes). character string that identifies the player by name.

6.7.12 AVRC Controller Command Status

This event indicates to the MCU that an AVRC command execution has started or that a command has been rejected due to the state of the hci_control application.

Table 227 AVRC Controller Command Status event

Item	Description
Operating code	0xFF
Parameters	Status (1 byte). See **Command Status**.

6.8 AV Remote Control Target events: HCI_CONTROL_GROUP_AVRC_TARGET

6.8.1 AVRC Target Connected

A CYWxxxxx device sends the AVRC Target Connected event to an MCU when a peer device establishes an AVRC connection or after a connection requested by an AVRC Target Connect command is successfully established.

Table 228 AVRC Target Connected event

Item	Description	
Operating code	0x01	
Parameters	bd_addr (6 bytes).	Bluetooth® address of the connected player.
	Connection handle (2 bytes)	The connection handle for the AVRC connection.

6.8.2 AVRC Target Disconnected

A CYWxxxxx sends the AVRC Target Disconnected event to an MCU to indicate that the AVRC connection has been terminated.

Table 229 AVRC Target Disconnected event

Item	Description	
Operating code	0x02	
Parameters	Connection handle (2 bytes).	The connection handle as reported in the AVRC Connected event.

6.8.3 AVRC Target Play

The CYWxxxxx sends this event to the MCU when a play command is received from a connected AVRC controller.

Table 230 AVRC Target Play event

Item	Description
Operating code	0x03
Parameters	Connection handle (2 bytes)

6.8.4 AVRC Target Stop

The CYWxxxxx sends this event to the MCU when a stop command is received from a connected AVRC controller.

Table 231 AVRC Target Stop event

Item	Description
Operating code	0x04
Parameters	Connection handle (2 bytes)

6.8.5 AVRC Target Pause

The CYWxxxxx sends this event to the MCU when a pause command is received from a connected AVRC controller.

Table 232 AVRC Target Pause event

Item	Description
Operating code	0x05
Parameters	Connection handle (2 bytes)

6.8.6 AVRC Target Next Track

The CYWxxxxx sends this event to the MCU when a next track command is received from a connected AVRC controller.

Table 233 AVRC Target Next Track event

Item	Description
Operating code	0x06
Parameters	Connection handle (2 bytes)

6.8.7 AVRC Target Previous Track

The CYWxxxxx sends this event to the MCU when a previous track command is received from a connected AVRC controller.

Table 234 AVRC Target Previous Track event

Item	Description
Operating code	0x07
Parameters	Connection handle (2 bytes)

6.8.8 AVRC Target Begin Fast Forward event

The CYWxxxxx sends this event to the MCU when a connected AVRC controller starts fast-forward operation. The target application should continue the fast-forward operation until the End Fast Forward event is received.

Table 235 AVRC Target Begin Fast Forward event

Item	Description
Operating code	0x08
Parameters	Connection handle (2 bytes)

6.8.9 AVRC Target End Fast Forward

The CYWxxxxx sends this event to the MCU when a connected AVRC controller terminates fast-forward operation.

Table 236 AVRC Target End Fast Forward event

Item	Description
Operating code	0x09
Parameters	Connection handle (2 bytes)

6.8.10 AVRC Target Begin Rewind

The CYWxxxxx sends this event to the MCU when a connected AVRC controller starts rewind operation. The MCU should continue the rewind operation until the AVRC Target End Rewind event is received.

Table 237 AVRC Target Begin Rewind event

Item	Description
Operating code	0x0A
Parameters	Connection handle (2 bytes)

6.8.11 AVRC Target End Rewind

The CYWxxxxx sends this event to the MCU when a connected AVRC controller terminates rewind operation.

Table 238 AVRC Target End Rewind event

Item	Description
Operating code	0x0B
Parameters	Connection handle (2 bytes)

6.8.12 AVRC Target Volume Level

The CYWxxxxx sends this event to the MCU when it receives a volume-level indication from a connected AVRC controller.

Table 239 AVRC Target Volume Level event

Item	Description
Operating code	0x0C
Parameters	Connection handle (2 bytes)

Item	Description
	Volume level (1 byte). The percentage (0 to 100) of the maximum volume level of the local audio player to be set.

6.8.13 AVRC Target Repeat Settings

The CYWxxxxx sends this event to the MCU when a connected remote controller changes the player repeat attribute settings value.

Table 240 AVRC Target Repeat Settings event

Item	Description	
Operating code	0x0D	
Parameters	Setting value (1 byte)	The following are possible values:
		0x01: Off
		0x02: Single track repeat
		0x03: All track repeat
		0x04: Group repeat

6.8.14 AVRC Target Shuffle Settings

The CYWxxxxx sends this event to the MCU when a connected remote controller changes the player shuffle attribute settings value.

Table 241 AVRC Target Shuffle Setting event

Item	Description	
Operating code	0x0E	
Parameters	Setting value (1 byte)	The following are possible values:
		0x01: Off
		0x02: All track shuffle
		0x03: Group shuffle

6.8.15 AVRC Target Command Status

This event indicates to the MCU that an AVRC command execution has started or that a command has been rejected due to the state of the *hci_control* application.

Table 242 AVRC Target Command Status event

Item	Description	
Operating code	0xFF	
Parameters	Status (1 byte)	The following are possible values:
		See **Command Status**

6.9 HID Device events: HCI_CONTROL_GROUP_HIDD

These events sent by the CYWxxxxx pertain to HID device profile functionality.

6.9.1 HID Opened

This event is sent when a HID connection has been fully established with a peer device, including control and interrupt channels.

Table 243 HID Opened event

Item	Description
Operating code	0x01
Parameters	-

6.9.2 HID Virtual Cable Unplugged

The CYWxxxxx sends this event when a connected host sends a Virtual Cable Unplug message over the HID Control channel.

Table 244 HID Virtual Cable Unplugged event

Item	Description
Operating code	0x02
Parameters	-

6.9.3 HID Data

The CYWxxxxx sends a HID data event after receiving a HID report on either the control or interrupt channel.

Table 245 HID Data event

Item	Description
Operating code	0x03
Parameters	Report type (1 byte)
	Report data (variable bytes)

6.9.4 HID Closed

The CYWxxxxx sends this event when a HID connection has been disconnected.

Table 246 HID Closed event

Item	Description
Operating code	0x04
Parameters	Reason (1 byte)

6.10 AIO Server events: HCI_CONTROL_GROUP_AIO_SERVER

These events sent by a CYWxxxxx pertain to AIO server functionality.

6.10.1 AIO Digital Output

This event sends a digital output value to an MCU.

Table 247 AIO Digital Output event

Item	Description	
Operating code	0x01	
Parameters	Index (1 byte)	Digital IO index, starting with 0.
	Data (variable bytes)	An array of 2-bit values in a bit field in little endian order.

6.10.2 AIO Analog Output

This event sends an analog output value to an MCU.

Table 248 AIO Analog Output event

Item	Description	
Operating code	0x01	
Parameters	Index (1 byte)	Analog IO index, starting with 0.
	Data (variable bytes)	The value of the analog signal as an unsigned 16-bit integer.

6.11 AIO Client events: HCI_CONTROL_GROUP_AIO_CLIENT

These events sent by a CYWxxxxx pertain to AIO client functionality.

6.11.1 AIO Command Status

This event indicates to the MCU that AIO command execution has started or that a command was rejected due to the state of the application.

Table 249 AIO Command Status event

Item	Description	
Operating code	0x01	
Parameters	Status (1 byte)	0: Command execution has started. 1: Command rejected because the previous command is still executing. 2: Connect command rejected; the specified device is already connected. 3: Disconnect command rejected because the connection is down.

Item	Description	
		4: Characteristic is not found. 5: Characteristic descriptor is not found. 6: Invalid parameters passed in the command

6.11.2 AIO Connected

This event instructs the MCU that a connection with an AIO server had been created.

Table 250 AIO Connected event

Item	Description
Operating code	0x02
Parameters	Device address (6 bytes)

6.11.3 AIO Read Response

This event sends a read response to an MCU.

Table 251 AIO Read Response event

Item	Description	
Operating code	0x03	
Parameters	Status (1 byte)	0: Success. 2: Read not permitted.
	Data (variable bytes)	An unsigned 16-bit integer for analog IO or an array of 2-bit values in a bit field for digital IO.

6.11.4 AIO Write Response

This event sends a write response to the MCU.

Table 252 AIO Write Response event

Item	Description	
Operating code	0x04	
Parameters	Status (1 byte)	0: Success. 3: Write not permitted.
	Data (variable bytes)	An unsigned 16-bit integer for analog IO or an array of 2-bit values in a bit field for digital IO.

6.11.5 AIO Input

The AIO client sends this event to the MCU after it receives notification about an IO module input change on the server.

Table 253 AIO Input event

Item	Description	
Operating code	0x05	
Parameters	Type (1 byte)	1: Analog IO. 2: Digital IO.
	Index (1 byte)	Analog or digital IO index, starting with 0.
	Data (variable bytes)	An unsigned 16-bit integer for analog IO or an array of 2-bit values in a bit field for digital IO.

6.11.6 AIO Disconnected

This event informs the MCU that an AIO server has been disconnected.

Table 254 AIO Disconnected event

Item	Description
Operating code	0x06
Parameters	Reason (1 byte)

6.12 Current Time events: HCI_CONTROL_GROUP_TIME

6.12.1 Time Update

An application running on a CYWxxxxx sends this event to the MCU when it can to connect to a peer device and retrieve the current time via a current-time service or when a current-time service running on a peer device sends a time update notification (for example, a notification that daylight savings time [DST] has taken an effect).

The date and time values are the local date and time reported by the server device. The time that the server device provides is normally the correct time for the location adjusted for time zone and DST.

Table 255 Time Update event

Item	Description	
Operating code	0x01	
Parameters	Year (2 bytes)	Current year
	Month (1 byte)	Current month
	Day (1 bytes)	Current day of month
	Hour	Current hour

Item	Description	
	(1 byte)	
	Minutes (1 byte)	Current minutes
	Seconds (1 byte)	Current seconds
	Exact time 256 (1 byte)	Current seconds fraction. LSB = 1/256 seconds.
	Day of week (1 byte)	Current day of the week: 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday 7: Sunday
	Adjust Reason (1 byte)	Bit field indicating the reason for the change in the time on the server. Bit 0: Manual time update Bit 1: External reference time update Bit 2: Time zone change Bit 3: Daylight savings time change

6.13 Test events: HCI_CONTROL_GROUP_TEST

The Test events pertain to the test command functionality to allow the host to execute various tests on the CYWxxxxx.

6.13.1 Encapsulated HCI event

While in the Test mode, the application encapsulates all the HCI events received from the controller in the Encapsulated HCI events and sends them to the MCU.

Table 256 Encapsulated HCI event

Item	Description	
Operating code	0x10	
Parameters	HCI event (variable bytes)	Fully formatted HCI event

6.14 ANCS events: HCI_CONTROL_GROUP_ANCS

The Apple Notification Control Service (ANCS) events pertain to the ANCS commands that let the MCU perform various ANCS-related procedures using the CYWxxxxx. See the Apple ANCS specification [3] for more information.

6.14.1 ANCS Notification

An application running on a CYWxxxxx sends this event to the MCU when it receives a notification from a connected iOS device.

Table 257 ANCS Notification event

Item	Description	
Operating code	0x01	
Parameters	Notification UID (4 bytes)	Notification Unique Identifier
	Event ID (1 byte)	0: Notification added 1: Notification modified 2: Notification removed
	Category (1 bytes)	0: Other 1: Incoming call 2: Missed call 3: Voicemail 4: Social 5: Schedule 6: Email 7: News 8: Health and fitness 9: Business and finance 10: Location 11: Entertainment
	Flags (1 byte)	Bit mask of event flags Bit 0: silent Bit 2: Important Bit 3: Preexisting Bit 4: Positive action possible Bit 5: Negative action possible
	Title (variable bytes)	Zero terminated UTF8 string with notification title.
	Message (variable bytes)	Zero terminated UTF8 string with notification message.
	Positive Action (variable bytes)	Zero terminated UTF8 string with positive action that can be performed by the MCU.
	Negative Action (variable bytes)	Zero terminated UTF8 string with negative action that can be performed by the MCU.

6.14.2 ANCS Command Status

This event indicates to the MCU that ANCS command execution has started or a command has been rejected due to the state of the application.

Table 258 ANCS Command Status event

Item	Description	
Operating code	0x02	
Parameters	Status (1 byte)	See **Command Status**

6.14.3 ANCS Service Found

This event indicates to the MCU that the ANCS service has been found on the given LE Connection Handle.

Table 259 ANCS Service Found event

Item	Description	
Operating code	0x03	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.

6.14.4 ANCS Connected

This event indicates to the MCU that ANCS service has started. The MCU can expect to start receiving ANCS Notification events after the ANCS Connected event has occurred.

Table 260 ANCS Connected event

Item	Description	
Operating code	0x04	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.
	Result (1 byte)	Provides additional status information, see **Command Status**.

6.14.5 ANCS Disconnected

This event indicates to the MCU that ANCS service has stopped or has been unsubscribed. ANCS Notification events shall not occur after the ANCS service has been disconnected.

Table 261 ANCS Disconnected event

Item	Description	
Operating code	0x05	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.

Item	Description	
	Result (1 byte)	Provides additional status information, see **Command Status**

6.15 AMS events: HCI_CONTROL_GROUP_AMS

The Apple Media Service (AMS) events pertain to the AMS commands that let an MCU perform various AMS-related procedures using the CYWxxxxx. See the Apple developer AMS specification **[4]** for more information:

6.15.1 AMS Command Status

This event indicates to the MCU that AMS command execution has started or a command has been rejected due to the state of the application.

Table 262 AMS Command Status event

Item	Description	
Operating code	0x01	
Parameters	Status (1 byte)	See **Command Status**

6.15.2 AMS Service Found

This event indicates to the MCU that the AMS service has been found on the given LE Connection Handle.

Table 263 AMS Service Found event

Item	Description	
Operating code	0x02	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.

6.15.3 AMS Connected

This event indicates to the MCU that AMS service has started.

Table 264 AMS Connected event

Item	Description	
Operating code	0x03	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.
	Status (1 byte)	See **Command Status**

6.15.4 AMS Disconnected

This event indicates to the MCU that AMS service has stopped or has been unsubscribed.

Table 265 ANCS Disconnected event

Item	Description	
Operating code	0x04	
Parameters	Connection handle (2 bytes)	The connection handle reported in the LE Connected event.
	Status (1 byte)	See **Command Status**

6.16 Alert events: HCI_CONTROL_GROUP_ALERT

6.16.1 Alert Notification

An application running on a CYWxxxxx forwards alerts received from a peer device in this event.

Table 266 Alert Notification event

Item	Description	
Operating code	0x01	
Parameters	Alert level (1 byte)	Alert level requested by the peer device. 0: No alert 1: Medium alert 2: High alert

6.17 iAP2 events: HCI_CONTROL_GROUP_IAP2

The CYWxxxxx uses Apple iPod Accessory Protocol (iAP2) events to provide an MCU with protocol status changes and data received over an iAP2 External Accessory (EA) session.

6.17.1 IAP2 Connected

This event is sent when an EA session has been established with a peer device. The Bluetooth® device address and connection handle are passed as parameters. The connection handle can be used by the MCU when sending subsequent commands or data and for identifying a peer device that has sent data.

This event can be sent for a connection originated by the MCU or by a peer iOS device.

Table 267 IAP2 Connected event

Item	Description	
Operating code	0x01	
Parameters	bd_addr (6 bytes)	Bluetooth® device address of the peer device with which an EA session has been established.

Item	Description	
	Handle (2 bytes)	iAP2 EA session handle.

6.17.2 IAP2 Service Not Found

A CYWxxxxx sends this event when it is able to connect to a peer device and perform SDP discovery, but the iAP2 service is not found.

Table 268 IAP2 Service Not Found event

Item	Description
Operating code	0x02
Parameters	-

6.17.3 IAP2 Connection Failed

The CYWxxxxx sends this event when a connection attempt requested by the MCU is unsuccessful.

Table 269 IAP2 Connection Failed event

Item	Description
Operating code	0x03
Parameters	-

6.17.4 IAP2 Disconnected

This event is sent when a previously established EA session is disconnected.

Table 270 IAP2 Disconnected event

Item	Description	
Operating code	0x04	
Parameters	Connection handle (2 bytes)	Connection handle reported in an IAP2 Connected event.

6.17.5 IAP2 TX Complete

A CYWxxxxx sends this event after a data packet received from an MCU in an IAP2 Send Data command has been queued for transmission. After sending the IAP2 Send Data command, the MCU should not send another data packet until it has received this event.

Table 271 IAP2 TX Complete event

Item	Description	
Operating code	0x05	
Parameters	Connection handle (2 bytes)	Connection handle reported in an IAP2 Connected event.

6.17.6 IAP2 RX Data

A CYWxxxxx sends this event to forward iAP2 data received from a peer device during an EA session.

Table 272 IAP2 RX Data event

Item	Description	
Operating code	0x06	
Parameters	Connection handle (2 bytes)	Connection handle reported in an IAP2 Connected event.
	Data (variable bytes)	Data received from a peer.

6.17.7 IAP2 Auth Chip Info

The CYWxxxxx sends this event after successfully processing an IAP2 Get Auth Chip Info command with chip information received from the authentication coprocessor.

Table 273 IAP2 Auth Chip Info event

Item	Description	
Operating code	0x07	
Parameters	Device version (1 byte)	Device version reported by the auth chip
	Firmware version (1 byte)	Firmware version reported by the auth chip
	Protocol version (Major) (1 byte)	Protocol version reported by the auth chip
	Protocol version (Minor) (1 byte)	Protocol version reported by the auth chip
	Device ID (4 bytes)	Device identification reported by the auth chip

6.17.8 IAP2 Auth Chip Certificate

The CYWxxxxx sends this event after successfully receiving IAP2 Auth Chip Certificate.

Table 274 IAP2 Auth Chip Certificate event

Item	Description	
Operating code	0x08	
Parameters	Data (variable byte)	Auth chip certificate

6.17.9 IAP2 Auth Chip Signature

The CYWxxxxx sends this event after successfully receiving IAP2 Auth Chip Signature.

Table 275 IAP2 Auth Chip Signature event

Item	Description	
Operating code	0x09	
Parameters	Data (variable byte)	Auth chip signature

6.18 AG events: HCI_CONTROL_GROUP_AG

These events sent by the CYWxxxxx pertain to the functionality of the Hands-Free Profile Audio Gateway.

6.18.1 AG Open

This event is sent when RFCOMM connection is established with a Hands-Free device. At this point, the Service Level Connection (SLC) is still not established, so commands cannot be sent. The Bluetooth® device address and connection handle are passed as parameters. The connection handle can be used by the MCU to send commands or to identify a peer device that caused the event.

Table 276 AG Open event

Item	Description	
Operating code	0x01	
Parameters	Connection handle (2 bytes)	Connection handle reported in an AG Connected event.
	Address (6 bytes)	Bluetooth® device address of the AG.
	Status (1 byte)	-

6.18.2 AG Close

This event is sent when the RFCOMM connection with a Hands-Free device is closed.

Table 277 AG Close event

Item	Description	
Operating code	0x02	
Parameters	Connection handle (2 bytes)	Connection handle reported in an AG Connected event.

6.18.3 AG Connected

This event is sent when the Hands-Free device and the AG have completed the protocol exchange necessary to establish an SLC. At this point, the application can send a command to establish an audio connection to the CYWxxxxx.

Table 278 AG Connected event

Item	Description	
Operating code	0x03	
Parameters	Connection handle (2 bytes)	Connection handle reported in an AG Connected event.
	Mask (4 bytes)	Mask of Hands-Free supported features.

6.18.4 AG Audio Open

This event is sent when an audio connection with a Hands-Free device is opened.

Table 279 AG Audio Open event

Item	Description	
Operating code	0x04	
Parameters	Connection handle (2 bytes)	Connection handle reported in an AG Connected event.

6.18.5 AG Audio Close

This event is sent when an audio connection with a Hands-Free device is closed.

Table 280 AG Audio Close event

Item	Description	
Operating code	0x05	
Parameters	Connection handle (2 bytes)	Connection handle reported in an AG Connected event.

6.19 Audio Sink events: HCI_CONTROL_GROUP_AUDIO_SINK

These events sent by the CYWxxxxx pertain to audio (A2DP) profile functionality.

6.19.1 Audio Sink Command Status

This event indicates to the MCU that an Audio Sink command execution has started or a command has been rejected due to the state of the *hci_control* application.

Table 281 Audio Sink Command Status event

Item	Description
Operating code	0x01
Parameters	Status (1 byte)
	See **Command Status**

6.19.2 Audio Sink Connected

This event is sent when an audio sink connection has been established with a peer device. The Bluetooth® device address and connection handle are passed as parameters. The connection handle can be used by the MCU to send commands or data, and to identify a peer device that has sent data.

Table 282 Audio Sink Connected event

Item	Description	
Operating code	0x02	
Parameters	Address (6 bytes)	Bluetooth® device address of peer.
	Connection handle (2 bytes)	The handle to use during command and data exchanges.

6.19.3 Audio Sink Service Not Found

A CYWxxxxx sends this event when it can connect to a peer device and perform SDP discovery, but there is no A2DP service.

Table 283 Audio Sink Service Not Found event

Item	Description
Operating code	0x03
Parameters	-

6.19.4 Audio Sink Connection Failed

A CYWxxxxx sends this event when a connection attempt requested by the MCU is unsuccessful.

Table 284 Audio Sink Connection Failed event

Item	Description
Operating code	0x04
Parameters	-

6.19.5 Audio Sink Disconnected

A CYWxxxxx sends this event when an audio sink connection has been dropped.

Table 285 Audio Sink Disconnected event

Item	Description
Operating code	0x05
Parameters	Connection handle (2 bytes)

6.19.6 Audio Sink Started

A CYWxxxxx sends this event when an audio stream has been started by the MCU that sent the Audio Sink Start command (see **Audio Sink Start**) or accept Audio Start Streaming request sent by peer (see **Audio Sink Start Response**).

Table 286 Audio Sink Started event

Item	Description
Operating code	0x06
Parameters	Connection handle (2 bytes)

6.19.7 Audio Sink Stopped

A CYWxxxxx sends this event when an audio stream has been stopped by the MCU that sent the Audio Stop command (see **Audio Sink Connect**) or peer remote device stop audio streaming.

Table 287 Audio Sink Stopped event

Item	Description
Operating code	0x07
Parameters	Connection handle (2 bytes)

6.19.8 Audio Sink Codec Configured

A CYWxxxxx sends this event when it receives AVDT SETCONFIG or AVDT RECONFIG command from peer audio source device in stream configuration process. On receiving this command, MCU can configure the codec setting based on the received parameter.

Table 288 Audio Sink Codec Configured event

Item	Description	
Operating code	0x08	
Parameters	Codec Id (1 byte)	0: SBC 1: MPEG-1, 2 2: MPEG-2, 4 0xFF: Vendor Specific
	Sampling frequency (1 byte)	0x80: 16kHz 0x40: 32kHz 0x20: 44.1kHz 0x10: 48kHz

Item	Description	
	Channel mode (1 byte)	0x08: Mono 0x04: Dual 0x02: Stereo 0x01: Joint Stereo
	Block length (1 byte)	0x80: 4 blocks 0x40: 8 blocks 0x20: 12 blocks 0x10: 16 blocks
	Number of sub-bands (1 byte)	0x08: 4 0x04: 8
	Allocation method (1 byte)	0x02: SNR 0x01: Loudness
	Max bit pool (1 byte)	
	Min bit pool (1 byte)	

6.19.9 Audio Sink Start Indication

This event is sent to the MCU on receiving an audio sink start request from the audio source device. The MCU will use the **Audio Sink Start Response** command to acceptor reject the audio stream start request.

Table 289 Audio Sink Start Indication event

Item	Description
Operating code	0x09
Parameters	Connection handle (2 bytes)
	Label (1 bytes)

6.19.10 Audio Sink Data

This event is sent to the MCU on receiving audio steaming data from the audio source, when audio route is not I2S. Data can be encoded or decoded based on the audio route selected (see **Audio Sink Start Response**).

Table 290 Audio Sink Data event

Item	Description
Operating code	0x0A
Parameters	Data (variable length)

6.20 LE COC events: HCI_CONTROL_GROUP_LE_COC

6.20.1 Connected

This event indicates to MCU that LE COC connection is established successfully.

Table 291 Connected event

Item	Description
Operating code	0x01
Parameters	Peer BDA

6.20.2 Disconnected

This event indicates to MCU that LE COC connection is disconnected.

Table 292 Disconnected event

Item	Description
Operating code	0x02
Parameters	Peer BDA

6.20.3 Received Data

This event indicates to MCU that data is received from peer over the LE COC connection.

Table 293 Data Received event

Item	Description
Operating code	0x03
Parameters	Data received

6.20.4 Transfer complete

This event indicates to MCU that the data transfer to peer over the established LE COC connection is successful.

Table 294 Transfer Complete event

Item	Description
Operating code	0x04
Parameters	0 – Success / 1 - Failure

6.20.5 Advertisement Status

This event indicates to MCU that current status of the advertisements (whether advertising is enabled/disabled).

Table 295 Advertisement Status event

Item	Description
Operating code	0x05
Parameters	0 – ADV OFF

6.21 ANS events: HCI_CONTROL_GROUP_ANS

These events sent by the CYWxxxxx pertain to ANS profile functionality.

6.21.1 Command Status

This event indicates to the MCU that an ANS command execution has started or a command has been rejected due to the state of the *hci_control* application.

Table 296 ANS Command Status event

Item	Description
Operating code	0x01
Parameters	Status (1 byte)
	See **Command Status**

6.21.2 ANS Enabled event

This event indicates to MCU that an ANS functionality has been initialized with the current supported categories.

Table 297 ANS Enabled event

Item	Description
Operating code	0x02
Parameters	Current enabled alert category (2 bytes)

6.21.3 Connection Up

This event indicates to MCU that the connection with Alert Notification Client is established.

Table 298 ANS Connection Up event

Item	Description
Operating code	0x03
Parameters	-

6.21.4 Connection Down

This event indicates to the MCU that Alert Notification Server is disconnected from Alert Notification Client.

Table 299 ANS Connection Down event

Item	Description
Operating code	0x04
Parameters	-

6.22 ANC events: HCI_CONTROL_GROUP_ANC

These events sent by the CYWxxxxx pertain to an ANC profile functionality.

6.22.1 ANC Enabled event

This event indicates to the MCU that connection with Alert Notification Server is established.

Table 300 ANC Enabled event

Item	Description
Operating code	0x01
Parameters	-

6.22.2 Server Supported New Alerts

This event indicates to MCU that the Read Server Supported New Alerts is complete.

Table 301 Server Supported New Alerts event

Item	Description	
Operating code	0x02	
Parameters	Status (1 byte)	See **Command Status**
	Supported new alerts (2 bytes)	The Server supported new alerts received on complete of Read Server Supported New Alerts.

6.22.3 Server Supported Unread Alerts

This event indicates to the MCU that the Read Server Supported Unread Alerts is complete.

Table 302 Server Supported Unread Alerts event

Item	Description	
Operating code	0x02	
Parameters	Status (1 byte)	See **Command Status**
	Supported unread alerts (2 bytes)	The Server supported unread alerts received on complete of Read Server Supported Unread Alerts.

6.22.4 Control Alerts

This event indicates to the MCU that the Control Alerts configuration for a specific Alert type is complete.

Table 303 Control Alerts event

Item	Description	
Operating code	0x04	
Parameters	Status (1 byte)	See **Command Status**
	Command ID (1 byte)	The type of alert command type for which the alert should be enabled.
	Category ID (1 byte)	The type of alert category which should be enabled.

6.22.5 Enable New Alerts

This event indicates to the MCU that Enabling New Alerts is complete.

Table 304 Enable New Alerts event

Item	Description
Operating code	0x05
Parameters	Status (1 byte)

6.22.6 Disable New Alerts

This event indicates to the MCU that Disabling New Alerts is complete.

Table 305 Disable New Alerts event

Item	Description
Operating code	0x06
Parameters	Status (1 byte)

6.22.7 Enable Unread Alerts

This event indicates to the MCU that Enable Unread Alerts is complete.

Table 306 Enable Unread Alerts event

Item	Description
Operating code	0x07
Parameters	Status (1 byte)

6.22.8 Disable Unread Alerts

This event indicates to the MCU that Disabling Unread Alerts is complete.

Table 307 Disable Unread Alerts event

Item	Description
Operating code	0x08
Parameters	Status (1 byte)

6.22.9 ANC Disabled Event

This event indicates to the MCU that there is a disconnection with the Alert Notification Server.

Table 308 ANC Disabled event

Item	Description
Operating code	0x09
Parameters	-

6.22.10 Command Status

This event indicates to the MCU, the command status for the requested operation.

Table 309 Command Status event

Item	Description
Operating code	0x09
Parameters	Status (1 byte)

6.23 DFU events: HCI_CONTROL_GROUP_DFU

These events are sent in response to the DFU command group.

6.23.1 DFU Configuration Reply event

This DFU event is sent as a response to the Get Configuration command (see **Get Configuration**).

Table 310 DFU Configuration Reply event

Item	Description
Operating code	0x01
Parameters	Transfer size (4 bytes)

6.23.2 DFU Write Command Prepare event

This DFU event is sent when the firmware upgrade library is prepared for starting the process.

Table 311 DFU Write Command Prepare event

Item	Description
Operating code	0x02

6.23.3 DFU Send Data Complete event: HCI_CONTROL_DFU_EVENT_DATA

This DFU event is sent in response to a data transfer command. The event indicates that the library is ready for the next data transfer command.

Table 312 DFU Send Data Complete event

Item	Description
Operating code	0x03

6.23.4 DFU Write Command Verify event

This DFU event is sent when the verification process is started in response to a verify command. The time it will take for verification to complete depends on the upgrade image size flash read time and computation required for verification. Verification methods supported include CRC-32 or ECDSA signature.

Table 313 DFU Write Command Verify event

Item	Description
Operating code	0x04

6.23.5 DFU Verification Complete event

If the verification process completes successfully, then this DFU event is sent. After a verification success, the device will reboot to use the upgraded firmware.

Table 314 DFU Verification Complete event

Item	Description
Operating code	0x05

6.23.6 DFU Aborted event

This DFU event is sent in response to an abort command, or when the verification has failed. After the abort event is received, the device can restart the firmware upgrade process.

Table 315 DFU Aborted event

Item	Description
Operating code	0x06

6.24 Miscellaneous events: HCI_CONTROL_GROUP_MISC

These events sent by the CYWxxxxx pertain to miscellaneous group of commands.

6.24.1 Ping Request Reply

This miscellaneous event is sent when the host sends a Ping Request (see **Ping Request**). The CYWxxxxx device responds with the exact data received in the Ping Request.

Table 316 Ping Request Reply event

Item	Description
Operating code	0x01
Parameters	Data (variable bytes)

6.24.2 Version Info

The Version Info miscellaneous event is sent in reply to the MCU sending Get Version command (see **Get Version**).

Table 317 Version Info event

Item	Description
Operating code	0x02
Parameters	Major version (1 byte)
	Minor version (1 byte)
	Revision number (1 byte)
	Build number (2 bytes)
	Chip ID (3 bytes)
	Unused (1 byte – obsoleted parameter)

For example, an application that runs on a CYW20819 with power class 1 and built using ModusToolbox™ version 1.1.0.225 would report 0x01, 0x01, 0x00, 0xE1, 0x00, 0x53, 0x51, 0x00, 0x00.

References

[1] **CYW920819EVB-02 Evaluation kit user guide**

[2] **Bluetooth® Core specification, version 4.2**

[3] **Apple ANCS specification**

[4] **Apple AMS specification**

Revision history

Document version	Date of release	Description of changes
**	2017-03-24	Initial release in template, updates for current WICED Studio, download sections expanded.
*A	2017-08-17	Updated board names referenced in the document
*B	2019-02-21	Updated for ModusToolbox™ Updated to include CYW20819
*C	2019-04-23	Removed Associated Part Family
*D	2019-05-21	Obsoleted a parameter in Version Info.
*E	2019-10-15	Updated for ModusToolbox™ 2.0
*F	2019-12-06	Updated Introduction Updated Downloading an application to serial
*G	2020-10-01	Updated Audio Connected and Audio Disconnected parameters
*H	2020-11-23	Add DFU over HCI Moved to Infineon Template
*I	2021-04-15	Updated Scope and purpose of the document
*J	2021-11-30	Branding updates, remove ClientControl information

Edition 2021-11-30

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to www.cypress.com/support

Document reference

002-16618 Rev. *J