AN216512

SPI Implementation for WICED BT Devices

Associated Part Family: CYW20706, CYW20735, CYW207x9

WICED™ Studio 4

To get the latest version of this application note, please visit www.cypress.com/AN216512

This document provides details on the Wireless Internet Connectivity for Embedded Devices (WICED) Serial Peripheral Interface (SPI) transport implementation for WICED Bluetooth devices. The document is for software developers who use WICED Studio to create applications for WICED Bluetooth chips that support the SPI transport, including CYW20706, CYW20735, and CYW207x9. This document refers to CYW20706 throughout, but the information is applicable to all chips with the SPI transport.

Contents

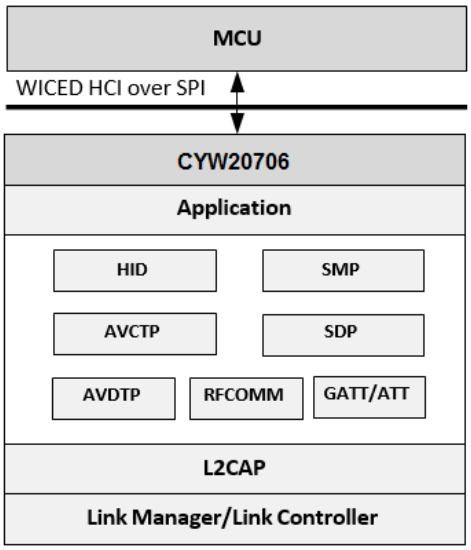
1 Introduction....1
2 IoT Resources....2
3 SPI Transport Applications....2
3.1 watch....2
3.2 hci_uart_spi_bridge....3
4 SPI Master TX Sequence....3
5 SPI Master RX Sequence....5
6 Transaction Collision....7
7 Test Procedure....7
8 References....8
Document History....9
Worldwide Sales and Design Support....10

1 Introduction

This document describes the SPI transport implementation for WICED devices. The WICED Host Controller Interface (HCI) Control Protocol [1] is used for data packets exchanged over the SPI. Data packets consist of a header that includes the length of the payload and the payload itself.

Figure 1 shows the logical location of the WICED HCI over SPI as well as the system protocol stack. The Microcontroller Unit (MCU) acts as an SPI master (Master) while the CYW20706 device acts as an SPI slave (Slave).

Figure 1. WICED HCI over SPI



2 IoT Resources

Cypress provides a wealth of data at http://www.cypress.com/internet-things-iot to help you select the right IoT device for your design, and quickly and effectively integrate the device into the design. Cypress provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. Customers can acquire technical documentation and software from the Cypress Support Community website (http://community.cypress.com/).

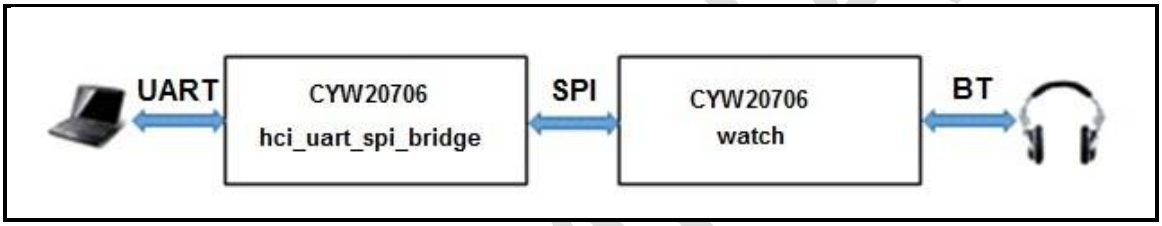
3 SPI Transport Applications

Two sample applications are available as part of WICED Studio that support SPI transport.

- watch
- hci_uart_spi_bridge

These applications can be used in tandem to verify the SPI implementation. Figure 2 shows an example of an SPI test setup using two CYW20706 devices.

Figure 2. SPI Test Setup Using Two CYW20706 Devices



3.1 watch

The watch application implements the SPI Slave, when built for the SPI transport target. Create or use the following make target in WICED Studio:

- watch-BCM920706_P49_SPI download

The watch application supports the following Bluetooth profiles:

- Advanced Audio Distribution Profile (A2DP)
- Generic Attribute Profile (GATT)
- Audio Video Remote Control (AVRC)

Table 1 shows the pins that are configured for SPI signaling on the Slave device.

Table 1. SPI Signaling for Slave Device

Functionality	Direction	Pin
SPI CLK	Input	P3
SPI MOSI	Input	P0
SPI MISO	Output	P25
SPI CS	Input	P2
SPI_SLAVE_READY	Output	P15

Note: The SPI CS pin also functions as a wake source for CYW20706.

The SPI_SLAVE_READY pin is used to both wake the host and indicate to the host that the Slave has data to send.

3.2 hci_uart_spi_bridge

The **hci_uart_spi_bridge** application demonstrates the SPI signaling protocol implementation on the MCU to allow communication over SPI using the WICED HCI Control Protocol [1]. This application implements the SPI Master and operates as a UART-SPI Bridge. The bridge application implements the following functions:

- Receives WICED HCI commands from the MCU/PC over a peripheral UART (PUART) and forwards the commands over SPI to the Slave.
- Receives WICED HCI events from the connected CYW20706 running the watch application over SPI and forwards them over the PUART to the connected MCU/PC.
- Intercepts Audio Data Request events from the watch application and responds with stored sine wave data.

Table 2 shows the pins that are configured for SPI signaling on the Master device.

Table 2. SPI Signaling for Master Device

Functionality	Direction	Pin
SPI CLK	Output	P36
SPI MOSI	Output	P0
SPI MISO	Input	P25
SPI CS	Output	P26
SPI_SLAVE_READY	Input	P12

Note: The SPI_SLAVE_READY pin can function as a wake source for the host/MCU.

4 SPI Master TX Sequence

The following steps describe how to transmit data from the SPI Master to the SPI Slave.

1. **The Master wakes the Slave.**
   a. The Master asserts the CS signal. This wakes/selects the Slave.
   b. The Master waits for the SPI_SLAVE_READY signal (rising edge) from the Slave to transmit the Header.
   c. The Master goes to Step 2 upon receiving the SPI_SLAVE_READY signal from the Slave.
2. **The Master transmits the Header of the WICED HCI packet (Header).**
   a. The Master transmits the Header, which contains the length of the payload, then deasserts the CS signal.
   b. The Master waits for the SPI_SLAVE_READY signal to be deasserted. This confirms that the Slave has received the Header.
   c. The Master waits for the SPI_SLAVE_READY signal from the Slave to transmit the payload.
   d. The Slave receives the Header and deasserts SPI_SLAVE_READY. When the Slave is ready to receive the payload, it asserts the SPI_SLAVE_READY signal.
   e. The Master goes to Step 3 upon receiving the SPI_SLAVE_READY signal from the Slave.
3. **The Master transmits the payload.**
   a. The Master asserts the CS signal, transmits the payload, and then deasserts the CS signal.
   b. The Slave receives the payload and deasserts the SPI_SLAVE_READY signal.

Figure 3 shows the SPI Master TX process and signaling sequence.

Figure 3. SPI Master TX

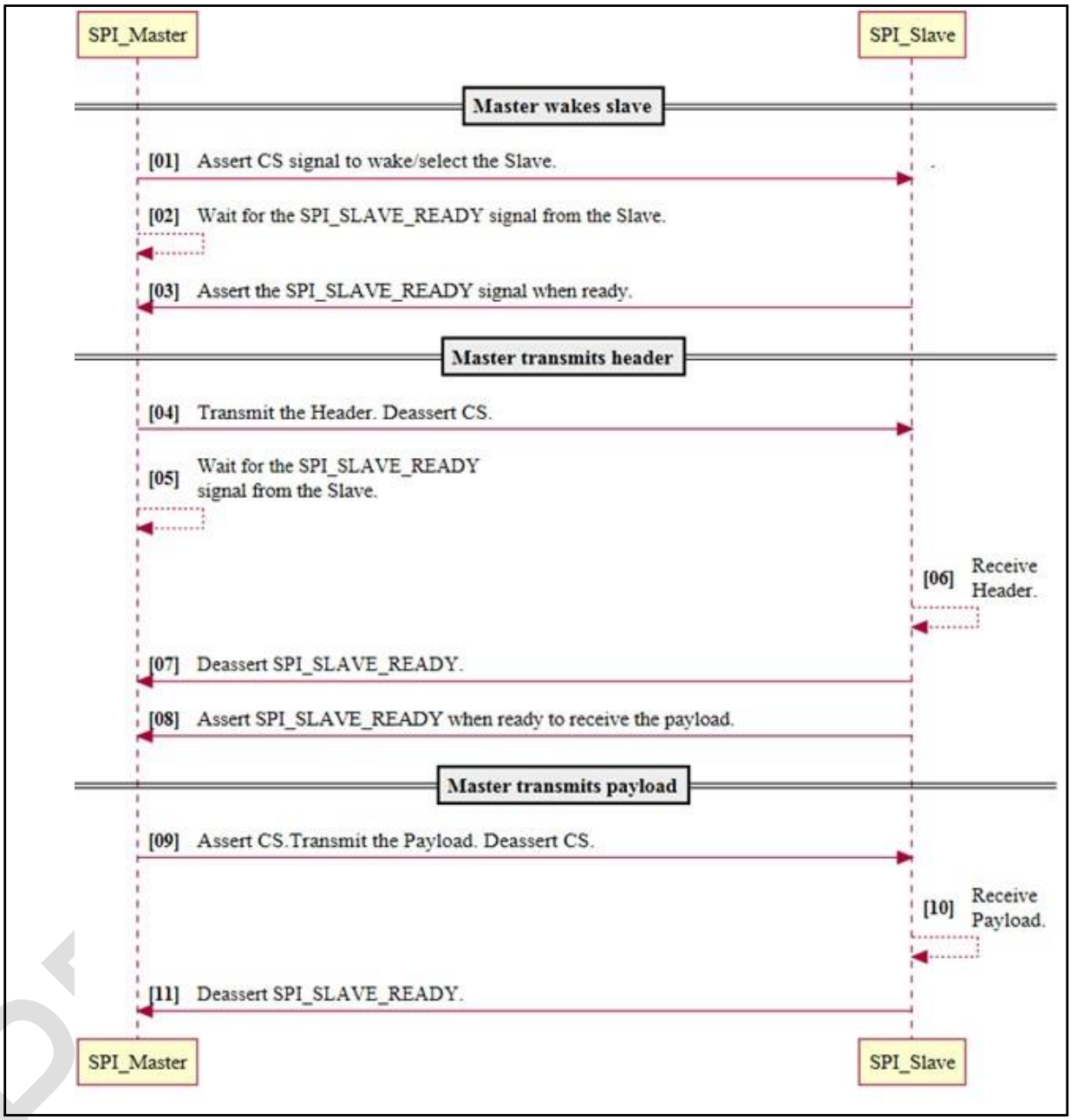
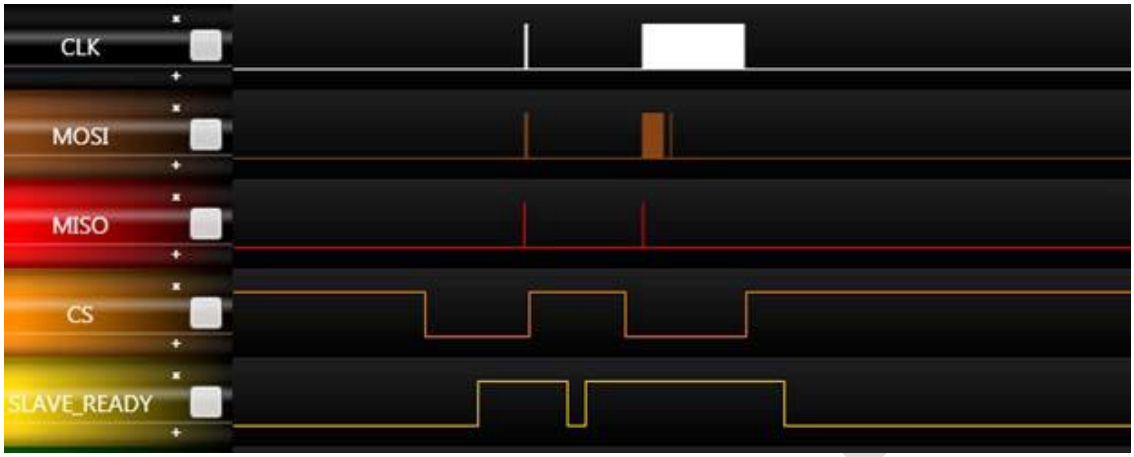


Figure 4 shows the Master TX sequence signaling from a timing perspective.

Figure 4. Master TX Sequence Signaling



5 SPI Master RX Sequence

The following steps describe how to send data from the SPI Slave to the SPI Master.

1. **The Slave interrupts the Master.**
   a. The SPI_SLAVE_READY signal from the Slave indicates data availability from the Slave.
2. **The Master transmits the RX token.**
   a. The Master asserts the CS signal, transmits the RX token, and then deasserts the CS signal.
   b. The Master waits for the SPI_SLAVE_READY signal to be deasserted. This confirms that the Slave has received the RX token.
   c. The Master waits for the SPI_SLAVE_READY signal from the Slave.
   d. The Slave receives the RX token, and then deasserts the SPI_SLAVE_READY signal.
   e. The Slave sets up the data to be transmitted, and then asserts the SPI_SLAVE_READY signal.
   f. The Master goes to Step 3 upon receiving the SPI_SLAVE_READY signal from the Slave.
3. **The Master reads the Header.**
   a. The Master asserts the CS signal, reads the Header, and then calculates the length of the upcoming payload.
4. **The Master reads the payload.**
   a. The Master reads the payload, and then deasserts the CS signal.
   b. The Slave deasserts the SPI_SLAVE_READY signal once the transmission is completed.

The RX token is a special WICED HCI command with opcode and length fields set to 0. The full packet is a 5-byte sequence - 0x19, 0x00, 0x00, 0x00, 0x00. The Master transmits the RX token to indicate that it is ready to receive data. The Slave may transmit the RX token to indicate that it has no data to send.

Figure 5 and Figure 6 show the SPI Master RX process and signaling sequence.

Figure 5. SPI Master RX

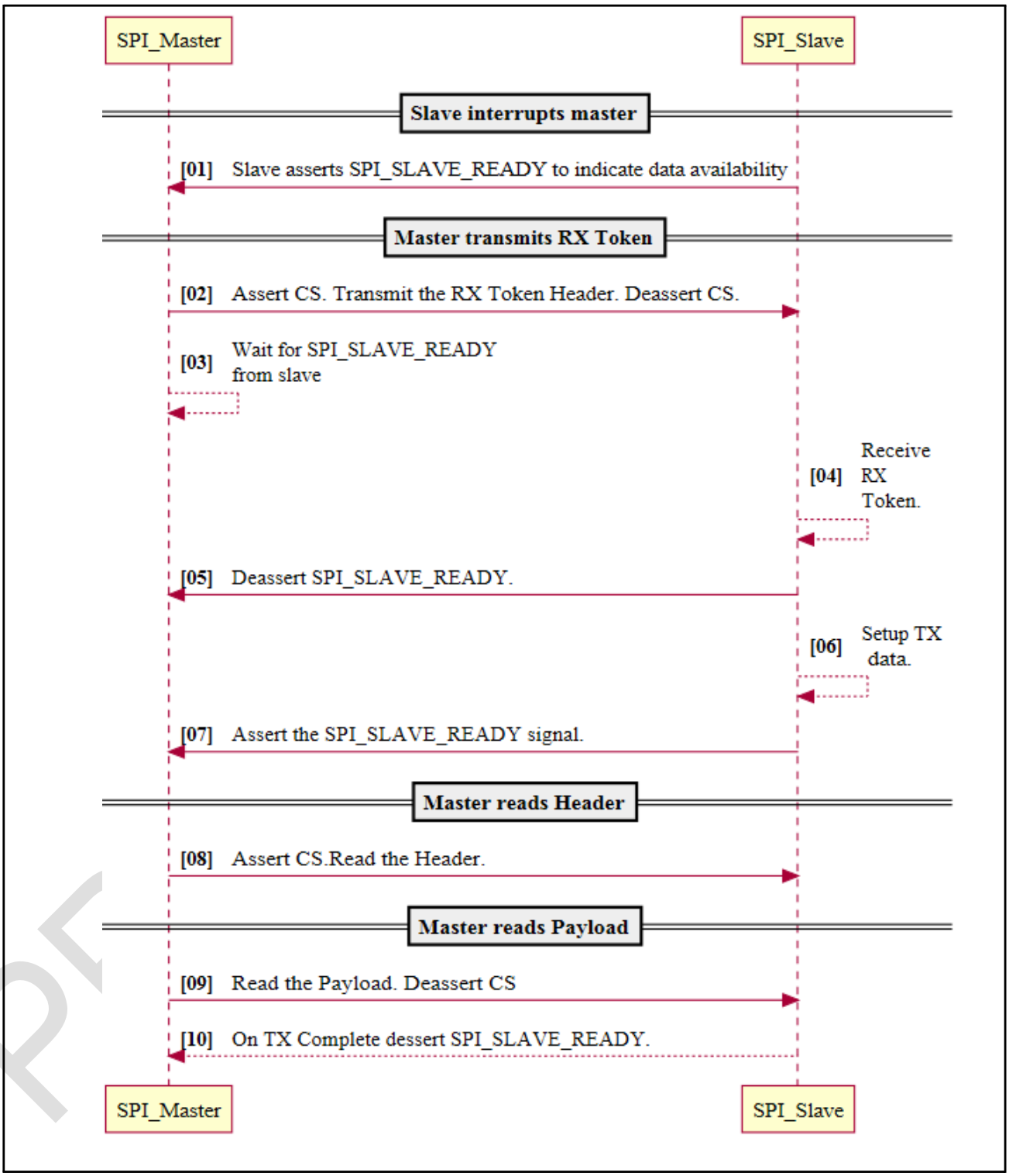


Figure 6. Master RX Sequence Signaling

6 Transaction Collision

In the described sequence, if the Master has data to send continuously, it will never grant the Slave the opportunity to transmit. Upon determining that the Slave has deasserted the SPI_SLAVE_READY signal at the end of the transaction (see step 3 from "SPI Master TX Sequence"), the Master will assert the CS signal and consider the asserted SPI_SLAVE_READY signal as an indication that the Slave is ready to receive, rather than concluding that the Slave wants to send data.

As a recommendation to avoid this problem, after the Master transmits the data payload, the Master may wait for 1 ms before starting a new transmission. If the Master sees the SPI_SLAVE_READY signal asserted during this back-off period, it starts the receive sequence even if it has data to send.

Note: If the Master sends an RX token and the Slave has no data to send, then the Slave sends another RX token to the Master to indicate that no data is available.

7 Test Procedure

1. Connect two CYW20706 devices via SPI, connecting the matching input and output signals for the Master and Slave (See Section 3, Table 1, and Table 2).
2. Build and download the **watch** WICED Studio sample application to the CYW20706 device that acts as the SPI Slave (See Section 3, and the WICED Studio Quick Start Guide for the chip you are using, for example CYW20706 [3]).
3. Build and download the **hci_uart_spi_bridge** WICED Studio sample applications to the CYW20706 device that acts as the SPI Master (See Section 3, and the WICED Studio Quick Start Guide for the chip you are using, for example CYW20706 [3]).
4. Execute the watch Client Control application provided in WICED Studio:

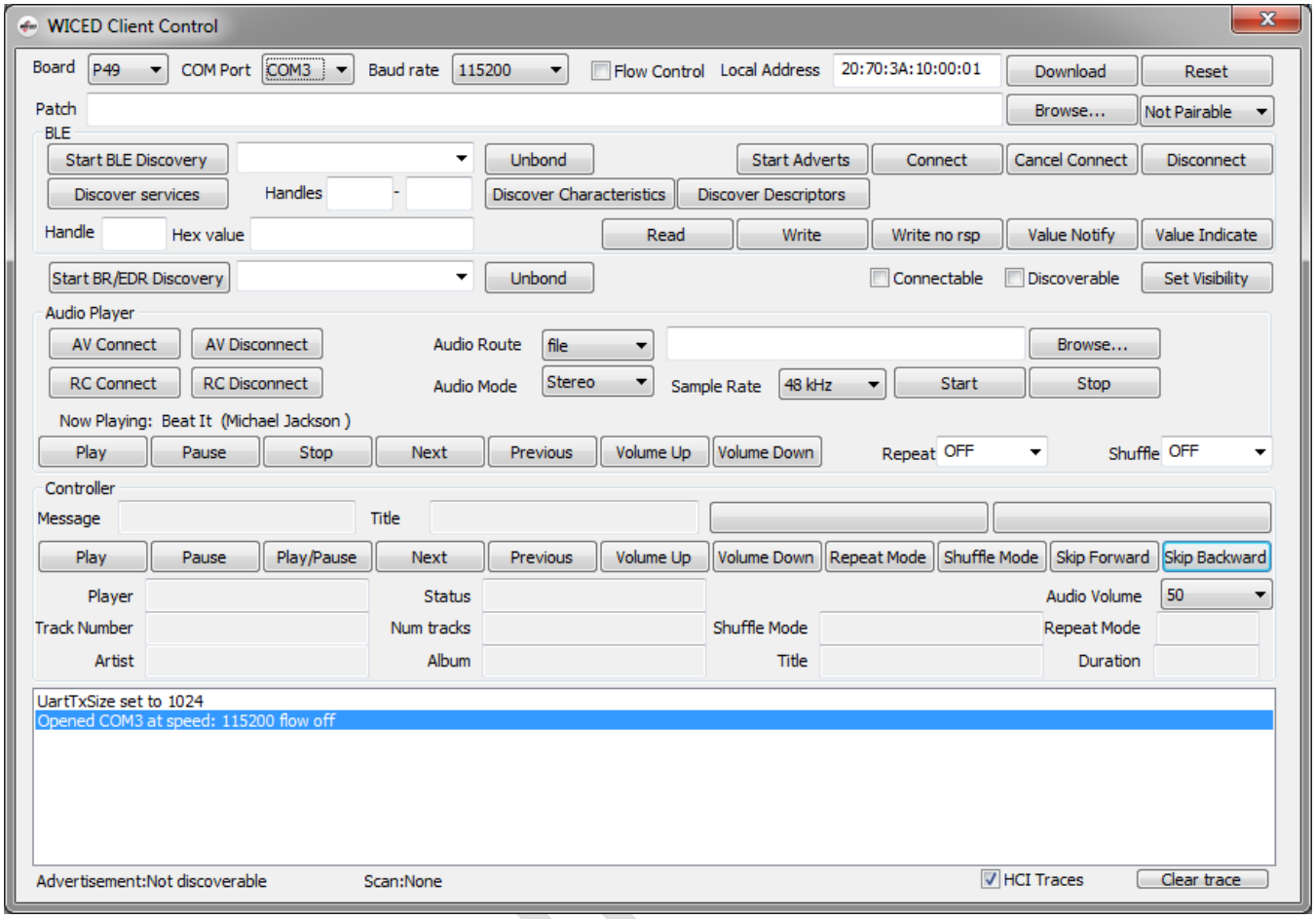
   <WICED-Studio>\20706-A2_Bluetooth\apps\watch\ClientControl\Release\ClientControl.exe

   Note: The CYW20706 device, when connected to the PC, enumerates two UARTs. The first is HCI UART and the second is Peripheral UART (PUART).

   Select the PUART port on the CYW20706 device running the **hci_uart_spi_bridge** application at 115,200 baud rate without flow control.

Figure 7 shows the interface that is displayed after the Client Control application is executed.

Figure 7. WICED Client Control Watch Application Interface



2. Reset the CYW20706 device running the **hci_uart_spi_bridge** application.
3. Reset the CYW20706 device running the **watch** application.

A *Device Started* message should be displayed in the trace window of the Client Control application. This indicates that the SPI transport is up and running.

Follow the standard test procedure to establish a connection between the CYW20706 running the **watch** application and a Bluetooth A2DP Sink device and play music. Refer to the application notes in *the hci_control.c* file that is provided as part of the watch application (in the beginning of this file, there are steps that describe how to test this application).

Note: Only sine wave audio stored in the **hci_uart_spi_bridge** application can be played over the air.

8 References

[1] AN216618 - WICED HCI Control Protocol

[2] AN216535 - CYW92070xV3_EVAL Hardware User Manual

[3] AN218191 - WICED Studio Quick Start Guide for BT CYW20706

[4] Bluetooth Core Specification, Version 4.2 (https://www.bluetooth.com/specifications/adopted-specifications)

Document History

Document Title: AN216512 – SPI Implementation for WICED BT Devices

Document Number: 002-16512

Revision	Submission Date	Description of Change
*A	01/12/2017	Updates to template and formatting - Preliminary
**	10/17/2016	Initial release - Preliminary

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709