



WICED Studio



CYW920735Q60EVB-01 Evaluation Kit User Guide

Associated Part Family: CYW20735

Doc. No.: 002-23764 Rev. **

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

Contents


Safety Information	4
General Safety Instructions	4
ESD Protection	4
Handling Boards	4
1 Introduction	5
1.1 CYW920735Q60EVB-01 EVB Contents.....	5
1.2 CYW920735Q60EVB-01 Board Details.....	6
1.3 WICED Studio Development System	6
1.4 Getting Started	7
1.5 IOT Resources and Technical Support.....	7
1.6 Additional Learning Resources.....	7
1.7 Document Conventions	7
1.8 Acronyms	8
2 WICED Studio	10
2.1 Before You Begin	10
2.2 WICED Studio Overview	10
2.3 Hardware and Software Requirements.....	10
2.4 Development Process	10
2.5 Setting up WICED Studio	10
2.5.1 Install WICED Studio	11
2.5.1.1 Windows.....	11
2.5.1.2 Linux.....	11
2.5.1.3 Mac OS X.....	11
2.5.2 Connect the WICED Evaluation Board	12
2.5.3 Verify Driver Installation.....	13
2.5.3.1 Windows.....	13
2.5.3.2 Linux.....	13
2.5.3.3 Mac OS X.....	13
2.6 Using the WICED Studio IDE	14
2.6.1 WICED Studio IDE UI	14
2.6.2 WICED Studio SDK Directory Structure	15
2.6.3 WICED Studio Code Examples	15
2.6.4 Build and Load a Sample Application	16
2.6.5 Hello Client Peer Application	18
2.6.6 Testing the Hello Sensor Application.....	19
2.6.6.1 Hello Sensor Application Structure.....	19
2.6.6.2 Hello Sensor Application Test Procedure.....	19
2.6.6.2.1 Hello Input Characteristic.....	19
2.6.6.2.2 Hello Configuration Characteristic	20
2.6.7 Viewing Application Trace Messages	20
2.6.7.1 Routing Trace Messages.....	20
2.6.7.2 View Traces Using a Terminal Emulation Program	21
2.6.7.3 View Traces Using the BTSpy Windows Application.....	21
3 Kit Operation	23
3.1 Theory of Operation.....	23
3.2 Jumpers.....	28
3.3 Buttons and Switches	30
3.4 Arduino-Compatible Headers	31


3.5	Other Headers	33
3.6	USB Serial Interface Chip	34
3.7	Kit Power Supply	34
3.8	Test Points	34
3.9	Current Measurement	35
3.10	SWD Debugging	35
3.11	Pin Configuration	35
4	Code Examples	36
4.1	Thermostat	36
4.1.1	Project Description	36
4.1.2	Hardware Connections	36
4.1.3	Flow Chart	37
4.1.4	Verify Output	38
5	Hardware	39
5.1	Carrier Module	39
5.1.1	CYW20735	39
5.1.2	Antenna	39
5.1.3	Crystal	39
5.1.4	External Serial Flash	39
5.2	Base Board	40
5.3	Serial Communication between CYW20735 and FTDI USB-Serial Device	40
5.4	Power	40
5.5	RESET	44
5.6	Thermistor	45
5.7	Motion Sensor	45
5.8	LED	46
5.9	Analog Mic	46
5.10	Push Buttons	47
Appendix A.	CYW20735 Device IO Mapping	48
Document Revision History		51

Safety Information

The CYW920735Q60EVB-01 Evaluation Board (EVB) is intended for use as a development platform for hardware or software in a laboratory environment. The board is an open-system design, which does not include a shielded enclosure. Due to this reason, the board may cause interference to other electrical or electronic devices in close proximity. In a domestic environment, this product may cause radio interference. In such cases, take adequate preventive measures. Also, do not use this board near any medical equipment or RF devices.

Attaching additional wiring to this product or modifying the product operation from the factory default may affect its performance and cause interference with other apparatus in the immediate vicinity. If such interference is detected, suitable mitigating measures must be taken.

	CYW920735Q60EVB-01 contains electrostatic discharge (ESD)-sensitive devices. Electrostatic charges readily accumulate on the human body and any equipment, and can discharge without detection. Permanent damage may occur on devices subjected to high-energy discharges. Proper ESD precautions are recommended to avoid performance degradation or loss of functionality. Store unused CYW920735Q60EVB-01 in the protective shipping package.
---	--

	End-of-Life/Product Recycling This kit has an end-of-life cycle of five years from the year of manufacturing mentioned on the back of the box. Contact your nearest recycler for discarding the kit.
---	--

General Safety Instructions

ESD Protection

ESD can damage boards and associated components. Cypress recommends that you perform procedures only at an ESD workstation. If an ESD workstation is not available, use appropriate ESD protection by wearing an antistatic wrist strap attached to the chassis ground (any unpainted metal surface) on the board when handling parts.

Handling Boards

CYW920735Q60EVB-01 boards are sensitive to ESD. Hold the board only by its edges. After removing the board from its box, place it on a grounded, static-free surface. Use a conductive foam pad if available. Do not slide the board over any surface. Any physical action on CYW920735Q60EVB-01 such as changing wires, jumper settings, or measuring voltages can cause stress on the CYW920735Q60EVB-01 printed circuit board assembly (PCBA). You must ensure that the PCBA has proper support on the bottom side to avoid stress on the PCBA when the EVB is in operation.

1 Introduction

Thank you for your interest in the CYW920735Q60EVB-01 Evaluation Board (EVB). The CYW920735Q60EVB-01 EVB enables customers to evaluate and develop single-chip Bluetooth and Bluetooth Low Energy (BLE) applications using the CYW20735B1, dual-mode Bluetooth 5.0 (BLE and BR) wireless MCU.

The CYW920735Q60EVB-01 EVB can be used with WICED™ Studio to develop and debug your CYW20735 project. The CYW920735Q60EVB-01 EVB offers footprint-compatibility with Arduino shields. In addition, the kit features an onboard programmer and USB-UART chip. The CYW920735Q60EVB-01 EVB supports 1.8 V and 3.3 V operation.

The CYW920735Q60EVB-01 EVB and CYW20735B1 device are supported in WICED Studio. The development system is compatible with Windows, OS X, and Linux operating systems. This document provides instructions for developing sample applications using WICED Studio.

Note: This document applies to WICED Studio 6.2 (or later).

The CYW920735Q60EVB-01 EVB is available through the [Cypress Online Store](https://www.cypress.com/store) or through our distributors.

1.1 CYW920735Q60EVB-01 EVB Contents

The CYW920735Q60EVB-01 EVB includes the following:

- One CYW920735Q60EVB-01 evaluation board
- One USB 2.0 Type-A to Micro-B cable
- One Quick Start Guide



Figure 1-1. CYW920735Q60EVB-01 Kit Contents

Inspect the kit contents. If you find any part missing, contact your nearest Cypress sales office for assistance: www.cypress.com/support.

1.2 CYW920735Q60EVB-01 Board Details

Figure 1-2 shows CYW920735Q60EVB-01 with the following features:

1. CYW20735B1-based carrier board with onboard antenna
2. Expansion headers that are compatible with Arduino shields
3. Support for 1.8 V and 3.3 V operation of the CYW20735 device
4. Two user-controlled LEDs, one push button, one recovery button, and one reset button
5. Onboard micro-USB connector for programming and debug purposes

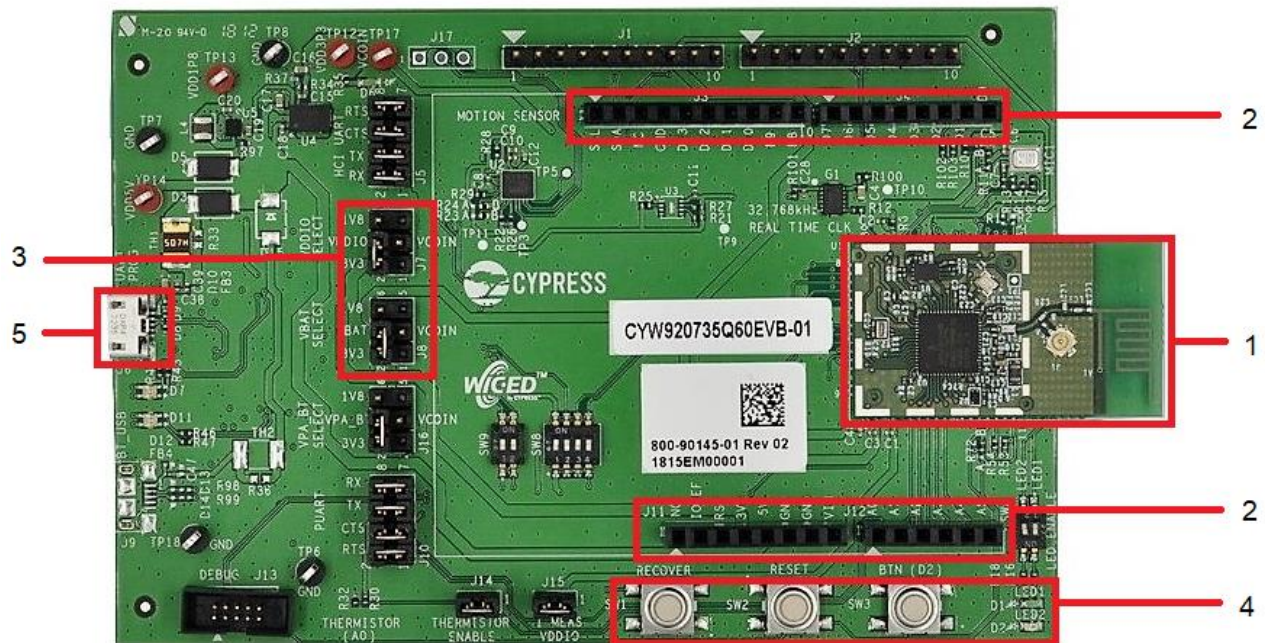


Figure 1-2. CYW920735Q60EVB-01 Evaluation Board

1.3 WICED Studio Development System

The WICED Studio Development System comprises a software development kit (SDK) along with the Eclipse integrated development environment (IDE) to enable development of projects with WICED evaluation boards.

The CYW920735Q60EVB-01 board and WICED Studio can be used for feature evaluation, debugging, and developing Bluetooth applications based on the CYW20735 device.

WICED Studio includes libraries and code examples which can speed up the design and development of user applications.

For detailed information on WICED Studio installation and usage, see [Setting up WICED Studio](#).

1.4 Getting Started

This user guide will provide additional details of the CYW920735Q60EVB-01 EVB:

- The [WICED Studio](#) chapter describes the installation and usage of the kit software. This includes WICED Studio to develop and debug the applications, Hello Client Peer application to test the hello_sensor application and BTSpy to view trace messages.
- The [Kit Operation](#) chapter describes the operation of the kit and how to use its various features.
- The [Code Examples](#) chapter describes code examples that will help you understand how to use an example with the kit.
- The [Hardware](#) chapter describes the design details of the CYW920735Q60EVB-01 EVB hardware blocks.

1.5 IOT Resources and Technical Support

Cypress provides a wealth of wireless product documentation at www.cypress.com/products/wireless-connectivity to help you to select the right IoT device for your design. In addition, a professional community at community.cypress.com/community/wireless supplies developers the latest software and tools to solve common evaluation and integration problems while interacting directly with both Cypress applications engineers and experienced peers.

1.6 Additional Learning Resources

Visit the <http://www.cypress.com/documentation/development-kitsboards/cyw920735q60evb-01-evaluation-kit> webpage for additional learning resources including datasheets and application notes.

1.7 Document Conventions

Convention	Usage
Courier New	Displays source code examples.
Consoles	API and function names (when mentioned within body text) The <code>WICED_BT_TRACE()</code> macro can be used to generate printf-style messages from the application code.
<i>Italics</i>	Displays file names, file locations, and reference documentation: <code>C:\...cd\iccl</code>
File > Open	Represents menu paths: File > Open > New Project
Bold	Displays commands, menu paths, and icon names in procedures: Click the File icon and then click Open .
Times New Roman	Displays an equation: $2 + 2 = 4$
Text in gray boxes	Describes Cautions or unique functionality of the product.

Table 1-1. Document Conventions for Guides

1.8 Acronyms

Acronym	Definition
ADC	Analog to Digital Converter
API	Application Programming Interface
BR	Basic Rate
BT / BLE	Bluetooth / Bluetooth Low Energy
EEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electro-magnetic
ESS	Environment Sensing Service
EVB	Evaluation Board
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GPIO	General Purpose Input Output
HAL	Hardware Abstraction Layer
HCI	Host Controller Interface
I ² C	Inter-Integrated Circuit
IDE	Integrated Development Environment
JRE	Java Runtime Environment
JTAG	Joint Test Action Group
LE	Low Energy
LED	Light Emitting Diode
LHL	Lean High Land
LPO	Low Power Oscillator
MEMS	Micro Electro-Mechanical System
NTC	Negative Temperature Coefficient
PCB	Printed Circuit Board
PUART	Peripheral UART
PWM	Pulse Width Modulation
RF	Radio Frequency
SDK	Software Development Kit
SIG	Special Interest Group
SoC	System-On-Chip
SPI	Serial Peripheral Interface
SWD	Serial Wire Debug
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

Acronym	Definition
VDD	Voltage Drain Drain
WICED	Wireless Internet Connectivity for Embedded Devices
XTAL	Crystal Oscillator

Table 1-2. List of Acronyms used in this Document

2 WICED Studio

This section provides detailed instructions to set up the Cypress Wireless Internet Connectivity for Embedded Devices (WICED; pronounced “wick-ed”) CYW920735Q60EVB-01 evaluation board for use with the Cypress WICED Studio Development System for Bluetooth Classic (aka BR - Basic Rate) and Low Energy (LE) devices.

WICED Studio supports application development using a WICED device such as the CYW20735 provided on the CYW920735Q60EVB-01 kit. The development system is compatible with Windows, Mac OS X, and Linux operating systems. This section describes the software components included in WICED Studio and provides instructions for compiling WICED sample applications using WICED Studio.

This document applies to WICED Studio 6.2 and WICED Bluetooth CYW20735 devices.

2.1 Before You Begin

All Cypress software installations require administrator privileges. Make sure that you have the required privileges on the system for successful installation. Before you install the kit software, close any other Cypress software that is currently running.

2.2 WICED Studio Overview

WICED Studio includes the following:

- Generic profile-level and BT stack level APIs (WICED BT API)
- Sample applications that demonstrate the use of the API
- Drivers to access on-chip peripherals using WICED HAL APIs (for example, UART, SPI, I2C, ADC, PWM)
- WICED BT API documentation
- Utilities to support development, testing, and mass production on Windows, Mac OS X, and Linux environments

2.3 Hardware and Software Requirements

WICED Studio runs on 32- and 64-bit versions of Microsoft Windows, and 64-bit versions of Mac OS X and Linux.

The development computer requires a single USB port to connect to the WICED evaluation board.

2.4 Development Process

WICED Studio is distributed as executable installers for Windows, Mac OS X, and Linux. Follow these steps to prepare and run an application:

- Download and install WICED Studio (see [Install WICED Studio](#)).
- Connect the WICED evaluation board (see [Connect the WICED Evaluation Board](#)).
- Create and load an application (see [Build and Load a Sample Application](#)).

2.5 Setting up WICED Studio

Download WICED Studio from [WICED Software page](#). For more details, please visit [Cypress WICED Products website](#) or [Cypress Customer Support Portal](#).

The WICED Studio distribution is provided as a self-installing executable file inside a zip file. Extract the file to a folder on the local hard drive; do not execute the installer from the zip file. Some customized distributions may also provide a configuration file called *config.eml*, which should be placed in the same folder prior to running the installer.

2.5.1 Install WICED Studio

Note: "x.x.x.x" used in the filenames denotes the WICED Studio version numbers in actual files.

2.5.1.1 Windows

1. Unzip the distribution to a local folder, along with the *config.eml* file, if present.
2. Double-click the *WICED-Studio-x.x.x.x-IDE-Installer.exe* file.
3. Follow the prompts to override or accept the default folders for the Eclipse IDE and WICED Studio SDK files.

2.5.1.2 Linux

1. Unzip the distribution to a local folder, along with the *config.eml* file, if present.
2. Open a terminal window and change directory (`cd`) to the same folder.
3. Apply execute permissions to the installer executable with the following command:

```
chmod +x ./WICED-Studio-x.x.x.x-IDE-Installer.bin
```

4. Launch the installer from the same terminal window:

```
./WICED-Studio-x.x.x.x-IDE-Installer.bin
```
5. Follow the prompts to override or accept the default folders for the Eclipse IDE and WICED Studio SDK files.

If there is a conflict between a pre-existing Java version installed on the system and the Java JRE supplied with the WICED Studio installer, the following error may be encountered during the Linux installation process: "Installer User Interface Mode Not Supported"

To resolve, use the package manager for your Linux distribution (for example, `dnf`, `apt-get`, `yum`) to update Java to the latest version. For example:

```
sudo dnf install java
```

2.5.1.3 Mac OS X

1. Unzip the distribution to a local folder.
2. If there is a *config.eml* file with the distribution, use Finder to copy the file and the *WICED-Studio-x.x.x.x-IDE-Installer.app* folder to another folder. This is needed as a workaround for a known OSX 10.12 install issue.
3. Double-click the *WICED-Studio-x.x.x.x-IDE-Installer.app*.
4. Follow the prompts to override or accept the default folders for the Eclipse IDE and WICED Studio SDK files.

If the installer fails to execute, you may need to install or update Java to resolve any potential conflict between a pre-existing Java version installed on the system and the Java JRE supplied with the WICED Studio installer.

Open xterm, run the `java -version` command. If it fails to return any results or states that you are running version 1.6, then you need to install the Java SE Development Kit 8 (JDK8), which can be found [here](#). Once JDK8 is installed, run the `java -version` command again with xterm. If this command returns "1.6", then you will need to fix the symbolic link using the following commands:

```
rm -f /usr/bin/java
ln -s /System/Library/Frameworks/JavaVM.framework/Versions/Current/Commands/java
/usr/bin/java
```

2.5.2 Connect the WICED Evaluation Board

Figure 2-1 shows the CYW920735Q60EVB-01 WICED evaluation board.

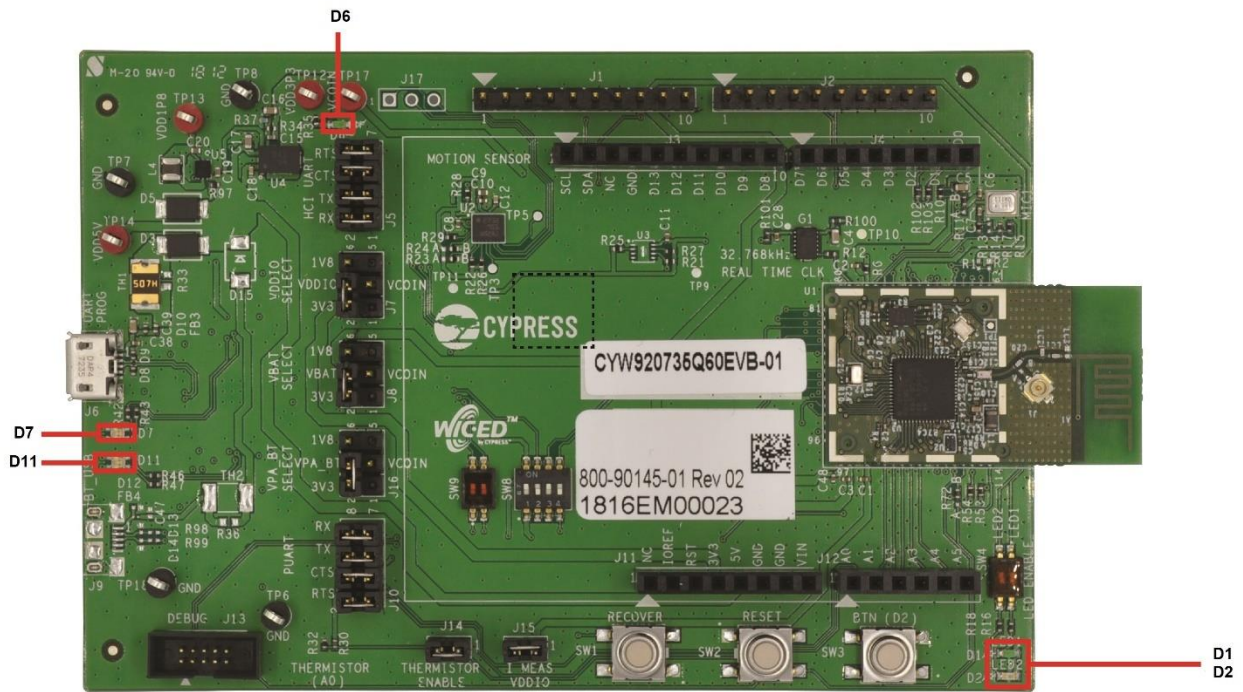


Figure 2-1. CYW920735Q60EVB-01 Evaluation Board

The Micro-USB connector (J6) supports UART connections and provides +5 V power to the board.

Follow these steps before connecting the board and verifying the driver installation:

1. Verify that all the jumpers are in default configuration as shown in [Table 3-1](#) to [Table 3-6](#), so that the Peripheral UART is selected and can display embedded application trace messages. The picture above shows the default jumper locations.
2. Connect J6 of the WICED evaluation board to the development PC with a USB cable. The USB UART driver should load automatically.

See [Jumpers](#) and [Buttons and Switches](#) for complete information on DIP switches and jumper settings.

The LEDs labelled on the board serve the following purposes:

- D6 (Green) indicates that VDD3P3 (3.3 V) power is ON
- D7 (variable color) indicates HCI UART activity
- D11 (variable color) indicates peripheral UART activity
- D1 (Yellow) and D2 (Red) are generic user LEDs controlled by GPIOs

2.5.3 Verify Driver Installation

2.5.3.1 Windows

1. Open the Device Manager (right-click **My Computer**, select **Properties**, and then select **Device Manager**).
2. In the **Device Manager** window, verify that two new USB serial COM ports are listed under **Ports (COM & LPT)**.

Note: In [Figure 2-2](#), the Device Manager identifies the new WICED evaluation board USB serial COM ports as WICED HCI UART and WICED Peripheral UART. Assigned port numbers vary among systems.

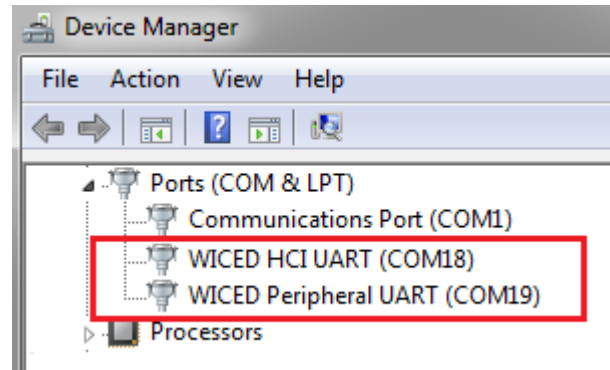


Figure 2-2. Device Manager COM Ports

Note: If an error occurs during driver installation, download new drivers from Windows Update. Verify that you have an Internet connection, disconnect and reconnect the board, and wait for the drivers to automatically install.

If new COM ports for the WICED board do not appear in the Device Manager after the drivers are installed via Windows Update, manually install the drivers from the `Drivers\Windows\wiced_uart` folder of the WICED Studio installation. Double-click the installer `DPInst.exe` for 32-bit Windows and `DPInst_x64.exe` for 64-bit Windows. Alternately, you can run the driver installers from inside WICED Studio; right-click the driver file name and select **Open With > System Editor**.

If the error persists, check all jumper settings (see [Jumpers](#)) on the board and replace the USB cable.

2.5.3.2 Linux

Open a terminal window and verify that two UART ports are listed, usually `/dev/ttyWICED_HCI_UART0` and `/dev/ttyWICED_PUART1`, although numbers may vary.

Note: An additional step may be required when connecting a WICED board to a computer running Linux. On common Linux distributions, the serial UART ports (such as `/dev/ttySx` or `/dev/ttyUSBx` devices, including the `/dev/ttyWICED_xxx` devices installed by WICED Studio) belong to the root user and to the dialout group. Standard users are not allowed to access these devices.

An easy way to allow the current user access to Linux's serial ports is by adding the user to the dialout group. You can do this using the following command:

```
$sudo usermod -a -G dialout $USER
```

Note: For this command to take effect, you must log out and log in again.

2.5.3.3 Mac OS X

Open an xterm window and verify that two UART ports are listed, usually `/dev/tty.usbserial-144A` and `/dev/tty.usbserial-144B`, although numbers may vary. The lower-numbered port is WICED HCI UART, the higher-numbered port is WICED Peripheral UART.

Note: On OS X versions 10.10 or earlier, the Apple version of the FTDI driver that ships with the OS has known issues. Follow the instructions [here](#) and update to a specific version.

On OS X versions 10.11 and later, use the Apple version of the FTDI driver. Remove previous instance of the FTDI version of the driver, if any, using the following commands in an xterm window:

```
sudo rm -rf /Library/Extensions/FTDIUSBSerialDriver.kext
sudo rm -rf /System/Library/Extensions/FTDIUSBSerialDriver.kext
```

Reboot the system after performing the rm commands.

2.6 Using the WICED Studio IDE

This section describes how to use the WICED Studio IDE to create application build targets for the WICED evaluation board and how to download applications to the board.

2.6.1 WICED Studio IDE UI

Double-click the WICED Studio icon on the desktop to start the IDE. Some WICED Studio packages may offer support for multiple devices, and on first-time execution of the IDE, you might be prompted to select the default platform. If prompted, select **20735-B1_Bluetooth**. You can change the default platform later from the **WICED Target Selector** drop-down list in the IDE.

Figure 2-3 shows the WICED Studio UI, illustrating the following functionality available through the IDE:

1. Project Explorer Window - Explore existing applications/firmware and libraries of the SDK.
2. Text Editor Window - Edit your application firmware.
3. Make Target Window - Create and edit Make Targets for the platform to build your application or project.
4. Console Window - View Build messages in the Console window.
5. Help Window - Access Help that contains instructions on building and downloading applications.

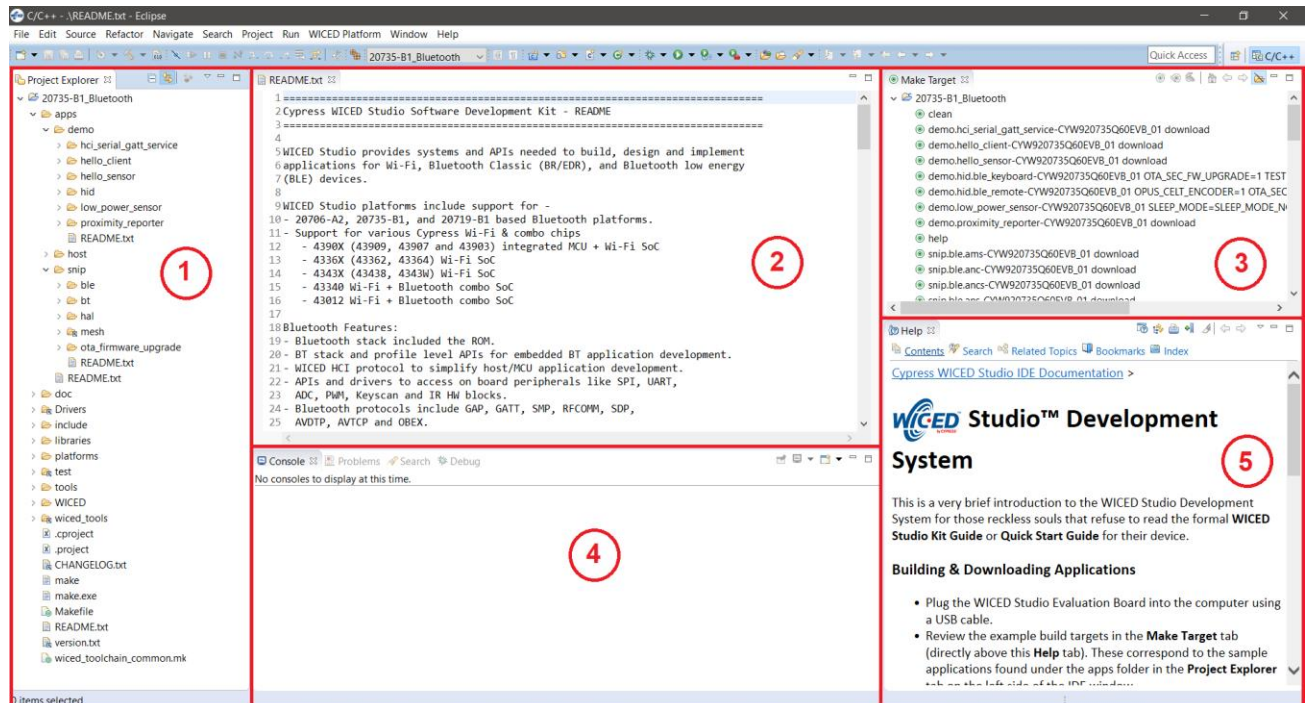


Figure 2-3. WICED Studio IDE

2.6.2 WICED Studio SDK Directory Structure

Table 2-1 lists the directory structure of WICED Studio for the CYW20735 device. WICED Studio may support multiple types of WICED modules depending on the installed components; some modules may share components, files, and folders.

Note: The folder structure presented in the WICED Studio Project Explorer UI window may differ slightly from the directory structure of the files installed on the file system, as some components may be shared between multiple components in common folders on the file system. It is recommended that WICED Studio be used for accessing files and folders rather than directly through the file system. In case of any discrepancies between the structure presented in WICED Studio and the file system, right-click the component in WICED Studio and select **Properties** to view the location of the component in the file system. For example, the ARM_GNU toolchain resides under the *43xxx_Wi-Fi* folder in the filesystem structure, but as used for the CYW20735 device, the *ARM_GNU* folder appears in the *wiced_tools* folder in the WICED Studio Project Explorer UI folder tree.

WICED Studio Project Explorer Directory	Directory Contents
<i>20735-B1_Bluetooth\doc</i>	Reference documentation
<i>20735-B1_Bluetooth\Drivers</i>	USB drivers for the evaluation board
<i>20735-B1_Bluetooth\wiced_tools</i>	Tools including programming tool, and other utilities and scripts
<i>20735-B1_Bluetooth\wiced_tools\ARM_GNU</i>	Toolchain including compiler, linker, and supporting files (libraries and headers)
<i>20735-B1_Bluetooth\apps</i>	Sample applications and the location where your applications will reside.
<i>20735-B1_Bluetooth\build</i>	Output files of built applications (will not be present until a build is performed)
<i>20735-B1_Bluetooth\include</i>	WICED API function prototypes and definitions
<i>20735-B1_Bluetooth\libraries</i>	Sources for various WICED interface libraries
<i>20735-B1_Bluetooth\platforms</i>	Configuration files and information for supported hardware platforms
<i>20735-B1_Bluetooth\test</i>	Tools provided for automation testing
<i>20735-B1_Bluetooth\tools</i>	Common utilities used by the IDE build processes
<i>20735-B1_Bluetooth\WICED</i>	WICED core components

Table 2-1. WICED Studio SDK Directory Structure

2.6.3 WICED Studio Code Examples

Application examples can speed up the design process by serving as templates for development. Code examples are located under the *apps* Folder (in the Project Explorer window), as shown in Figure 2-4.

The *demo* directory contains applications that combine various WICED features into a single application. The *snip* directory contains application snippets that demonstrate how to use various WICED libraries and API functions. The *host* directory contains applications that demonstrate WICED BT API usage for host MCU apps. Located within each subdirectory in the *apps* folder is a *README.txt* that lists and summarizes the applications located within the folder:

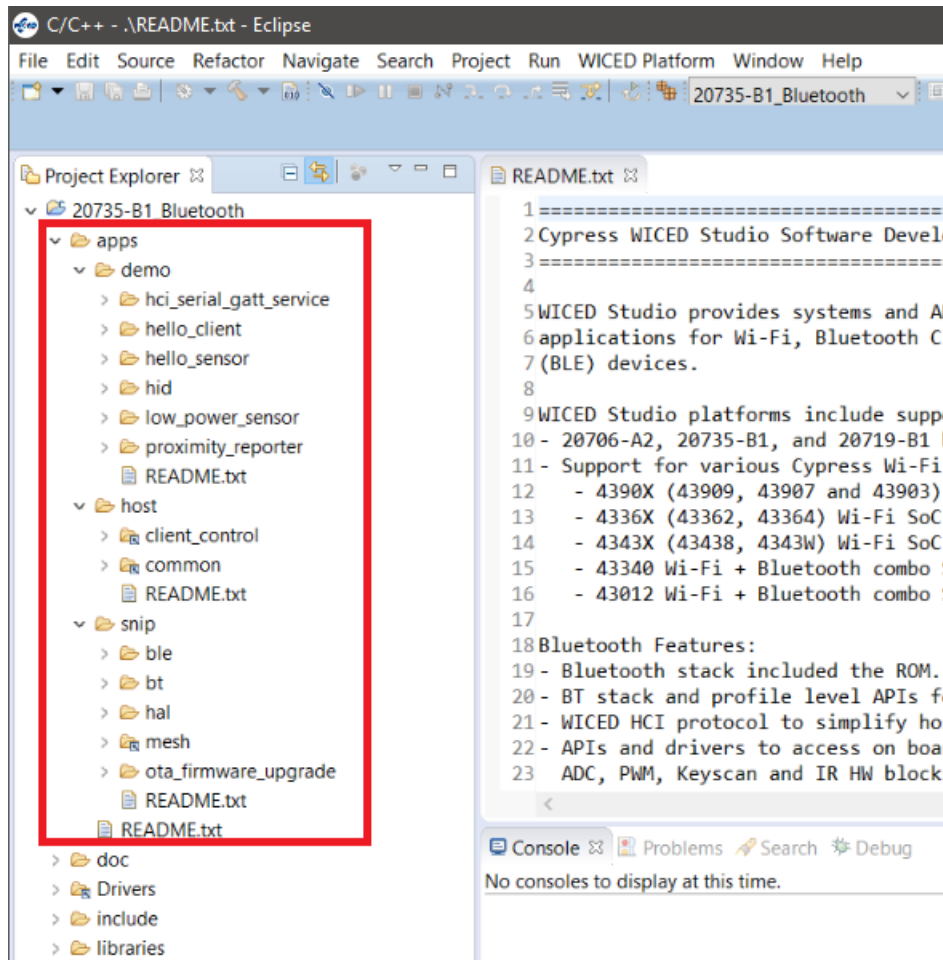


Figure 2-4. Code Examples under apps Folder

For more details on the WICED software stack and APIs, see the documents available in the *doc* folder in the Project Explorer pane of WICED Studio. The *API.html* file contains API documentation, and it is recommended that you open the file using the default system web browser. Right-click *API.html* and select **Open With > System Editor**.

2.6.4 Build and Load a Sample Application

The **Help** pane in the lower-right corner of the IDE (see Figure 2-3) describes how to build and download the sample applications shown in the **Make Target** pane, which is located above the **Help** pane. The **Help** pane also describes how to create new applications and associated make targets based on the samples. The **Make Target** pane contains make targets that are preconfigured for sample applications that run on CYW920735Q60EVB-01 evaluation boards.

The *hello_sensor* sample demonstrates basic LE sensor functionality. The source file *hello_sensor.c* contains information on how to exercise the application with the Hello Client host peer application. You can find the source code for the *hello_sensor* sample and Hello Client host peer application in WICED Studio in the **Project Explorer** pane under *20735-B1_Bluetooth\apps\demo\hello_sensor*. This example shows how to build and run the *hello_sensor* sample application on the CYW20735:

1. Connect the evaluation board to the PC.
2. Verify that the UART port is present after the WICED evaluation board is connected to the PC (see [Verify Driver Installation](#)).
3. In the WICED Studio **Make Target** pane, find the make target *demo.hello_sensor-CYW920735Q60EVB_01 download* in the list. Double-click the Make Target to execute it. The IDE console pane (bottom center of the IDE window) will display the build and download progress.

The build output should look similar to the following:


```
17:34:05 **** Build of configuration Release for project 20735-B1_Bluetooth ****
"C:\Users\<USER>\Documents\WICED-Studio-X.X\20735-B1_Bluetooth\make.exe"
demo.hello_sensor-CYW920735Q60EVB_01 download
Compiling wiced_platform_pin_config.c
Compiling wiced_platform.c
Compiling spar_setup.c
Compiling hello_sensor.c
Compiling wiced_bt_cfg.c
Compiling wiced_platform.c
Compiling wiced_platform_pin_config.c
Compiling lib_installer.c
Linking target ELF
OK, made elf.
..\..\43xxx_Wi-Fi\tools\ARM_GNU\Win32\bin\arm-none-eabi-objdump: section '.aon'
mentioned in a -j option, but not found in any input file
..\..\43xxx_Wi-Fi\tools\ARM_GNU\Win32\bin\arm-none-eabi-objdump: section '.unused'
mentioned in a -j option, but not found in any input file
Call to hello_sensor_spar_crt_setup @ 00211325
OK, made C:/Users/<USER>/Documents/WICED-Studio-6.2.0.41/20735-
B1_Bluetooth/WICED/./build/hello_sensor-CYW920735Q60EVB_01-rom-ram-Wiced-
release/A_20735B1-hello_sensor-rom-ram-spar.cgs. MD5 sum is:
6ff1f72be9bcf59bafd1c0dafa76e477 *./build/hello_sensor-CYW920735Q60EVB_01-rom-ram-
Wiced-release/A_20735B1-hello_sensor-rom-ram-spar.cgs
```

```
-----
Patch code starts at          0x00270400 (RAM address)
Patch code ends at           0x00270E48 (RAM address)
Patch RW/ZI size              56 bytes
Application starts at        0x0020FF64 (RAM address)
Application ends at          0x00211321 (RAM address)
```

```
Patch code size                2632 bytes
Application RAM footprint      5053 bytes
-----
Total RAM footprint            5109 bytes (5.0kiB)
-----
```

```
Converting CGS to HEX...
Conversion complete
```

```
Creating OTA images...
Conversion complete
OTA image footprint in NV is 8545 bytes
Detecting device...
Device found
```

```
Downloading application...
Download complete
```

```
Application running
```

Note: The warnings 'section '.XXX' mentioned in a -j option, but not found in any input file' above are not problems; this is only an indication that the application did not use any data in the mentioned sections.

Note: If the download fails, it is possible the memory on the board has been corrupted by a previously loaded application, or the application used a custom baud rate that the download process does not detect. In that case it may be necessary to reset the board to factory defaults. To do this, first, press and hold the **Recovery** button (SW1), then press the **Reset** button (SW2), release the **Reset** button (SW2), and then release the **Recovery** button (SW1).

Note: Because the sample target includes the download option, the tool will download the firmware to the evaluation board automatically when the build is complete, where it is stored in the serial flash. The board will reset after the download is successful. When the board boots and finds a valid application image in serial flash, it will execute the image.

2.6.5 Hello Client Peer Application

The Hello Client (`hello_client`) peer sample application is provided with WICED Studio to complement the Hello Sensor (`hello_sensor`) application running on the CYW20735 device. You can find the sample application in `20735-B1_Bluetooth\apps\demo\hello_sensor\peer_apps` in the WICED Studio Project Explorer.

The entire source code of the application is provided for Windows, iOS, and Android, along with an executable binary that runs on 32-bit and 64-bit Windows 10 systems. The application is also compatible with Windows 8 and 8.1, but not with lower versions. These steps demonstrate the use of the Windows-based Hello Client application; however, the process is similar for all host operating systems:

1. On the Windows host PC, open the **Settings** screen (usually from **Start**), select **Devices**, and then **Bluetooth**.
2. Click **Add a device** and wait while the PC searches for the devices in range.
3. Select the **Hello Sensor** device, which will appear as "Hello" (this is the embedded application running on CYW20735).
4. Click **Pair** and wait for the device connection to complete.
5. Run `HelloClient.exe` on the Windows host PC. The executable is in the WICED Studio Project Explorer under `apps\demo\hello_sensor\peer_apps\Windows\HelloClient\Release\<x64|x86>`.

If multiple devices have been paired, a Hello Client Select Device window similar to that shown in [Figure 2-5](#). appears. The window shows a list of Bluetooth device addresses for any Hello Sensor applications running on evaluation boards that have been paired to the PC. You can select the correct device from the list displayed in this window.

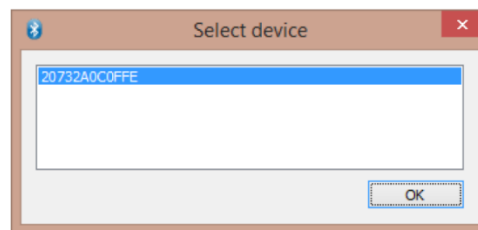


Figure 2-5. Hello Client Select Device Window

6. Select the correct device (if it is not already selected) and click **OK** to initiate a connection to the evaluation board. The connection process may take 5 to 10 seconds.

If only one device has been paired, the selection screen will not appear.

Note: By default, 20735-B1 LE applications use LE random addressing. If desired, this can be changed to use a specific address, or you may specify a portion of the address to be specific and a portion to be generated randomly. To specify all or part of the Bluetooth device address, edit the file `20735-B1_Bluetooth\apps\demo\hello_sensor\wiced_bt_cfg.c` in WICED Studio Project Explorer and disable LE random addressing by changing the setting to "WICED_BT_CFG_DEFAULT_RANDOM_ADDRESS_NEVER_CHANGE" in the following line:

```
.rpa_refresh_timeout = WICED_BT_CFG_DEFAULT_RANDOM_ADDRESS_CHANGE_TIMEOUT,
```

When LE random addressing is disabled, the Bluetooth device address programmed in the evaluation board comes from the file `20735-B1_Bluetooth\platforms\20735_SFLASH.btp` in the WICED Studio Project Explorer.

To change the Bluetooth device address, open the file and modify the `DLConfigBD_ADDRBase` variable. This value can alternately be overwritten in the make target of the application by setting `BT_DEVICE_ADDRESS=xxxxxxxxxxx`, where `xxxxxxxxxxx` is a 12-digit hexadecimal value representing the Bluetooth device address. You can substitute the digits with asterisk (*) characters, in which case WICED Studio will generate random hexadecimal values for the asterisks using the host PC MAC address, so that the random digits will always be the same from a single PC, but different for other PCs.

2.6.6 Testing the Hello Sensor Application

2.6.6.1 Hello Sensor Application Structure

The Hello Sensor application provides paired devices with the following information:

- A Hello Service (a proprietary service) with two proprietary characteristics:
 - The value of the Hello Input characteristic is incremented each time button **SW3** (see [Figure 3-3](#)) on the EVB is pressed. The value (read-only) may be retrieved from the Hello Client application using one of the following methods:
 - Manually, by clicking **Read** on the HelloClient PC application.
 - Whenever the value changes by pressing **SW3**, “**Allow Notifications**” must be selected from the **Value** drop-down list to allow automatic notifications.
 - The Hello Configuration read-write characteristic is used to configure the number of times an LED on the evaluation board blinks (see [Figure 3-3](#)) when the value is changed and whenever button **SW3** is pressed.
- A Device Information Service that provides information including:
 - Manufacturer Name
 - Model Number
 - System ID
- A Battery Service that provides a battery-level indication

The application sends dummy battery level value and increments for every GATT read operation of the battery level. In order to get the actual values from the battery, please refer to `include/hal/wiced_hal_batmon.h` file for using ADC to measure battery level.

2.6.6.2 Hello Sensor Application Test Procedure

To test the application with a WICED evaluation board, follow the instructions provided for each of the Hello Service characteristics (see [Figure 2-6](#)).

2.6.6.2.1 Hello Input Characteristic

1. Select **Allow Notifications** from the **Value** drop-down list for Hello Input.
2. Press button **SW3** on the WICED board (see [Figure 3-3](#)). The Hello X message is displayed in the **Value** field. Each time the button is pressed, the message number increments.

2.6.6.2.2 Hello Configuration Characteristic

1. Change the **Value** field for the Hello Configuration to **5** and then click **Write**. The Red LED on the board blinks 5 times.
2. Press button **SW3** on the WICED board (see [Figure 3-3](#)). The LED on the board again blinks five times.

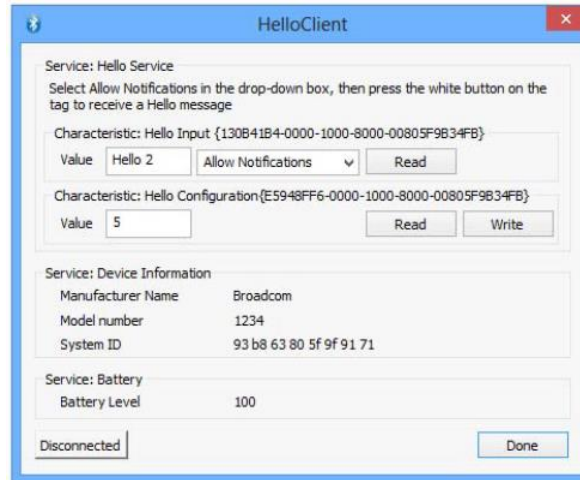


Figure 2-6. HelloClient Services Window

2.6.7 Viewing Application Trace Messages

You can view trace messages from the application if the application is compiled with the `WICED_BT_TRACE_ENABLE` compile flag defined. You can define compile flags in the application `makefile`. In the `hello_sensor` application, the `WICED_BT_TRACE_ENABLE` flag is enabled by default (see `makefile.mk` in the `hello_sensor` application source folder in WICED Studio).

With the `WICED_BT_TRACE_ENABLE` compile flag defined, the `WICED_BT_TRACE()` macro can then be used to generate printf-style messages from the application code.

2.6.7.1 Routing Trace Messages

You can route trace messages to any of the following destinations using the `wiced_set_debug_uart()` API with the appropriate parameter:

- HCI UART serial port – `wiced_set_debug_uart (WICED_ROUTE_DEBUG_TO_HCI_UART)`
- PUART serial port – `wiced_set_debug_uart (WICED_ROUTE_DEBUG_TO_PUART)`
- BTSpY Windows application – `wiced_set_debug_uart (WICED_ROUTE_DEBUG_TO_WICED_UART)`

The BTSpY option allows Bluetooth protocol trace messages to be viewed alongside application traces. See [View Traces Using the BTSpY Windows Application](#) for information on viewing traces with the BTSpY option.

The quickest and simplest way to view application traces is to use the HCI UART or PUART options with a terminal emulation program connected to the appropriate UART port. See [View Traces Using a Terminal Emulation Program](#) for information on terminal settings for the UART route options.

Note: Downloads from WICED Studio use the HCI UART serial port, so if you are also using the HCI UART serial port for debug messages, do not open the HCI UART port with the terminal emulation program until after the application download has completed in WICED Studio to avoid contention. The PUART serial port has no such contention, so the `hello_sensor` sample application uses the PUART by default. The other options (HCI_UART and WICED_UART) are provided in the source file as comments. You can comment out the PUART lines and uncomment either the HCI_UART or WICED_UART line if you want to try one of the other options.

2.6.7.2 View Traces Using a Terminal Emulation Program

To view traces with a terminal emulation program (such as Tera Term or PuTTY on Windows, PuTTY on Linux, or SerialTools on Mac OS X), start the terminal emulation program and open a connection to the desired UART port (either HCI or Peripheral, as specified in the call to the `wiced_set_debug_uart()` API) using the following settings:

Baud Rate: 115200^a
Data: 8 bit
Parity: None
Stop: 1 bit
Flow Control: None
Terminal ID: VT100
New-Line Receive: Auto

Press **Reset** (SW2) on the WICED evaluation board to view the application startup messages. For example:

```
Hello Sensor Start
hello_sensor_management_cback: 15
local keys read from NVRAM result: 0
hello_sensor_management_cback: 0
hello_sensor_application_init
wiced_bt_gatt_register: 0
wiced_bt_gatt_db_init 0
hello_sensor_load_keys_for_address_resolution 00 00 00 00 00 00 result:40
hello_sensor_management_cback: 17
Advertisement State Change: 3
wiced_bt_start_advertisements 0
hello_sensor_timeout: 1, ft:959
hello_sensor_timeout: 2, ft:1944
hello_sensor_timeout: 3, ft:2927
```

2.6.7.3 View Traces Using the BTSpY Windows Application

By routing application traces with the `wiced_set_debug_uart (WICED_ROUTE_DEBUG_TO_WICED_UART)` API call, you can view traces using the separate BTSpY and ClientControl utilities. These Windows applications are supplied with WICED Studio and are available in the `wiced_tools` and `apps\host\client_control` folders respectively.

The ClientControl utility connects to the HCI UART port and receives trace message packets. These message packets are then sent through a socket connection to the BTSpY utility, where they can be displayed and logged.

The BTSpY trace option provides an advantage over the UART options. Applications may configure the stack to generate Bluetooth protocol trace messages showing all activity between the stack and the controller over the virtual HCI interface embedded in the CYW20735 device. The Bluetooth protocol trace messages will be encoded into WICED HCI message packets and sent along with application trace messages (from `WICED_BT_TRACE()`), which can be encoded and displayed by the BTSpY utility. You can then view the application trace messages in sequence with the corresponding Bluetooth protocol trace messages.

^a The Hello Sensor application uses 115200 as the default baud rate for the PUART, where traces are routed by default in that application. Other applications in WICED Studio may use different trace destinations and baud rates, which may be found in each sample application's initialization of the `wiced_transport_cfg_t` structure, used in the call to the `wiced_transport_init()` API for routing traces to the HCI UART port, and in any calls to the `wiced_hal_puart_set_baudrate()` API overriding the default rate for routing traces to the PUART port.

To configure the stack to generate Bluetooth protocol trace messages, applications must route trace messages to the ClientControl/BTSpy receivers by calling `wiced_set_debug_uart` (`WICED_ROUTE_DEBUG_TO_WICED_UART`), and define a callback function and register it with the stack using the `wiced_bt_dev_register_hci_trace()` API. The callback function implementation should call the `wiced_transport_send_hci_trace()` API with the data received in the callback. See the Hello Sensor sample implementation of `hello_sensor_hci_trace_cback()` in the `hello_sensor.c` file, for an example.

To view application and Bluetooth protocol traces in BTSpy:

1. Define `WICED_BT_TRACE_ENABLE` in the application `makefile.mk` file. For example, add the line:

```
C_FLAGS += -DWICED_BT_TRACE_ENABLE
```

2. Ensure that the application calls `wiced_set_debug_uart` (`WICED_ROUTE_DEBUG_TO_WICED_UART`).

Note: Hello sensor calls this API in `hello_sensor.c`, to route to the PUART by default. However, commented out code is provided to change this to the `WICED_UART` instead.

3. Compile and download the application to the evaluation board.
4. Press **Reset** (SW2) on the WICED evaluation board to reset the HCI UART after downloading, before opening the port with ClientControl.
5. Run the ClientControl utility from `apps\host\client_control`. You can open the utility from inside WICED Studio. Right-click the utility and select **Open With > System Editor**.
6. In the ClientControl utility, select the HCI UART port in the UI, and set the baud rate to the baud rate used by the application for the HCI UART (3000000 for Hello Sensor; for other applications, see the initialization of the `wiced_transport_cfg_t` structure used in the call to the `wiced_transport_init()` API).
7. Run the BTSpy utility from `wiced_tools`. You can open the utility from inside WICED Studio. Right-click the utility and select **Open With > System Editor**.
8. In the ClientControl utility, open the HCI UART COM port by clicking the Open Port button.
9. Press **Reset** (SW2) on the WICED evaluation board to reset the kit and view trace messages in the BTSpy window.

3 Kit Operation

This section provides detailed instructions to set up the Cypress Wireless Internet Connectivity for Embedded Devices (WICED; pronounced "wick-ed") CYW920735Q60EVB-01 evaluation board for use with the Cypress WICED Studio Development System for Bluetooth Classic (aka BR - Basic Rate) and Low Energy (LE) devices.

This chapter introduces you to the CYW920735Q60EVB-01 EVB and the features that will be used as part of kit operation. This chapter also discusses features such as the Bluetooth connection and programming/debugging as well as the USB-UART bridge device that can be used to communicate with the CYW20735-B1 device on this EVB.

3.1 Theory of Operation

The CYW920735Q60EVB-01 is built around the CYW20735-B1 device. Figure 3-1 shows the block diagram of the CYW20735-B1 device. See the CYW20735-B1 datasheet for the details on device features.

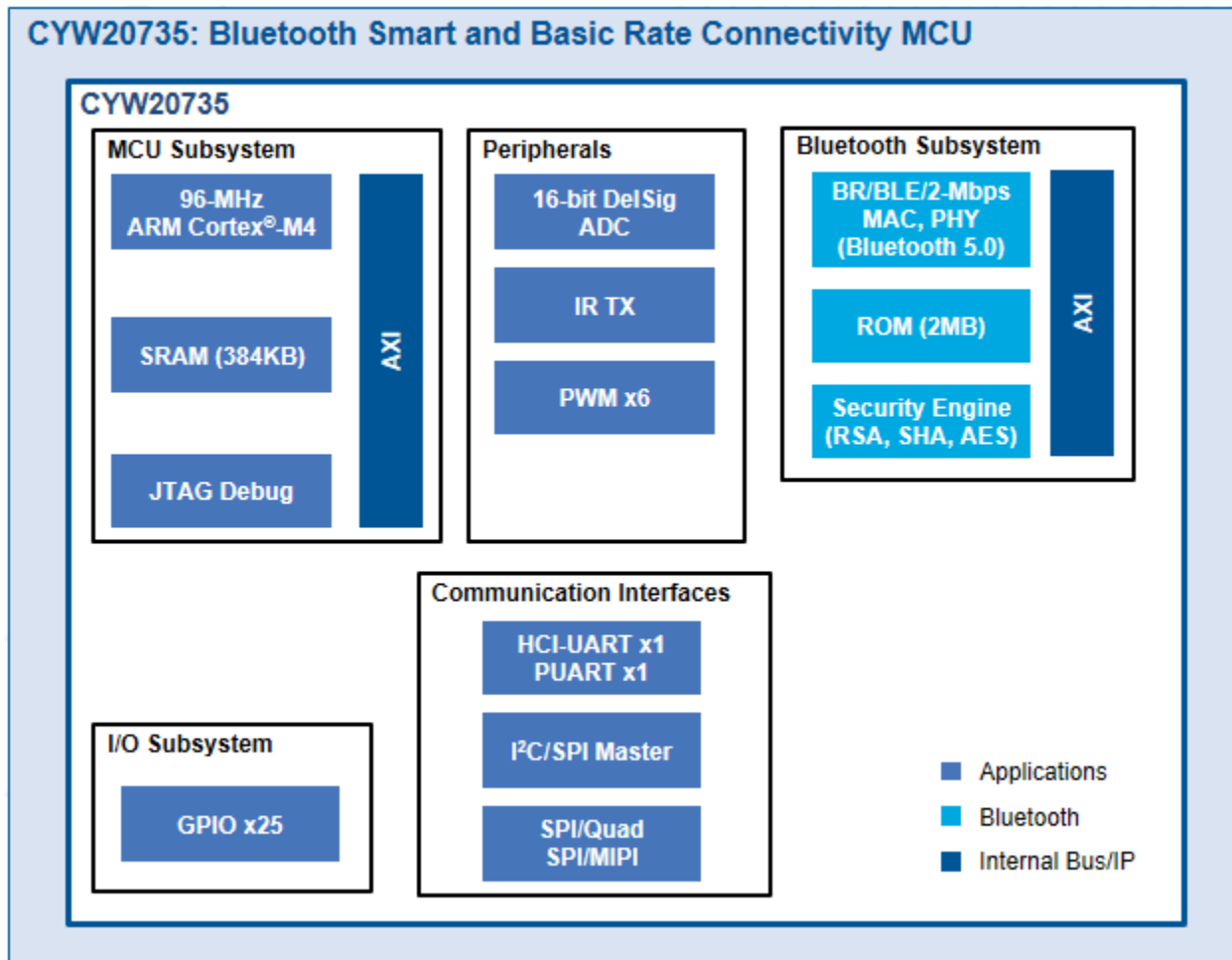


Figure 3-1. Block Diagram of CYW20735-B1 Device

Figure 3-2 illustrates the block diagram of the CYW920735Q60EVB-01 EVB. This board contains a CYW20735-B1 Bluetooth device and a USB-Serial interface/debugger. The kit features Arduino form-factor compatible headers, which enables Arduino shields to be plugged on top, extending its capabilities. It also features one user switch, one reset switch, one recovery switch, two user LEDs, a thermistor, and a motion sensor.

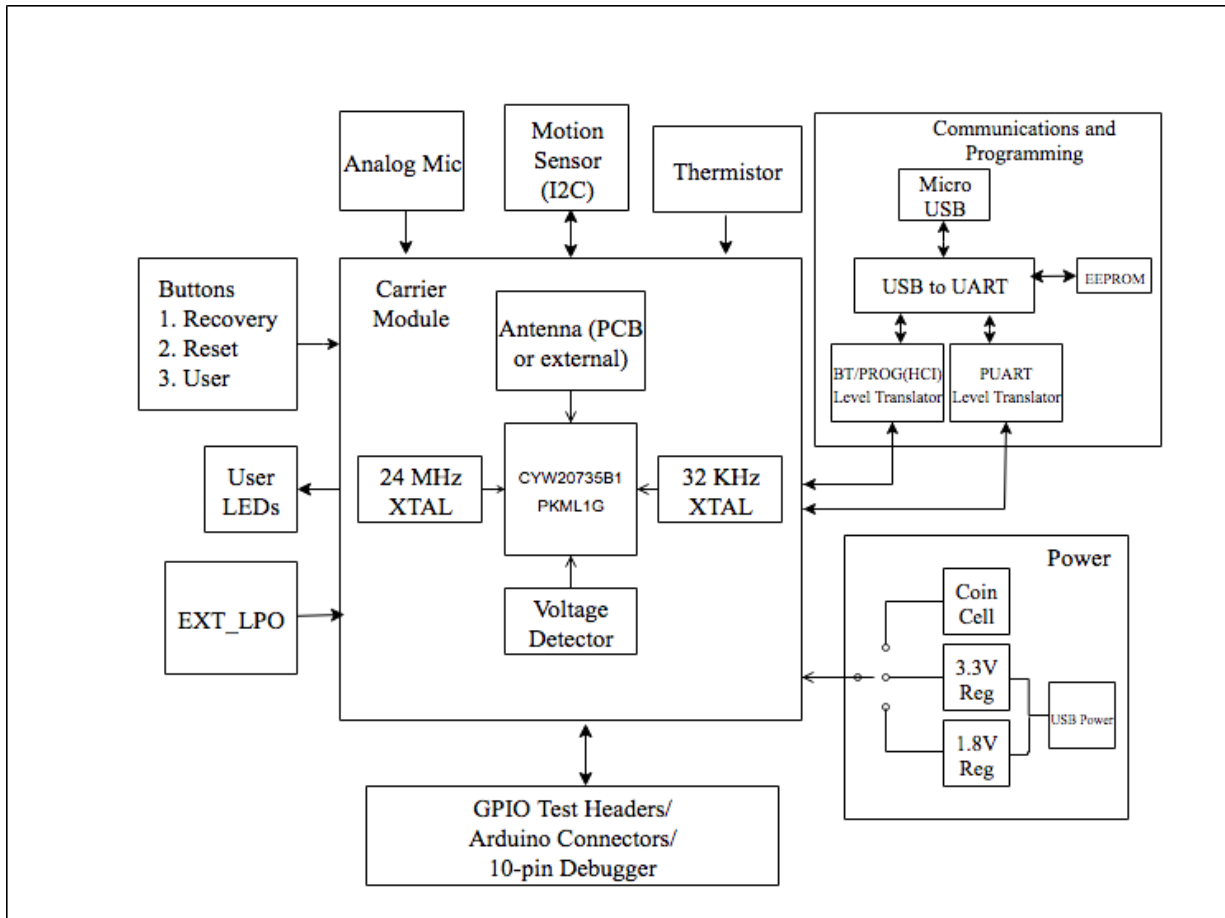


Figure 3-2. Block Diagram of CYW920735Q60EVB-01 EVB

Figure 3-3 and Figure 3-4 show the markup of the CYW920735Q60EVB-01 board. See the list below for a description of the numbered items.

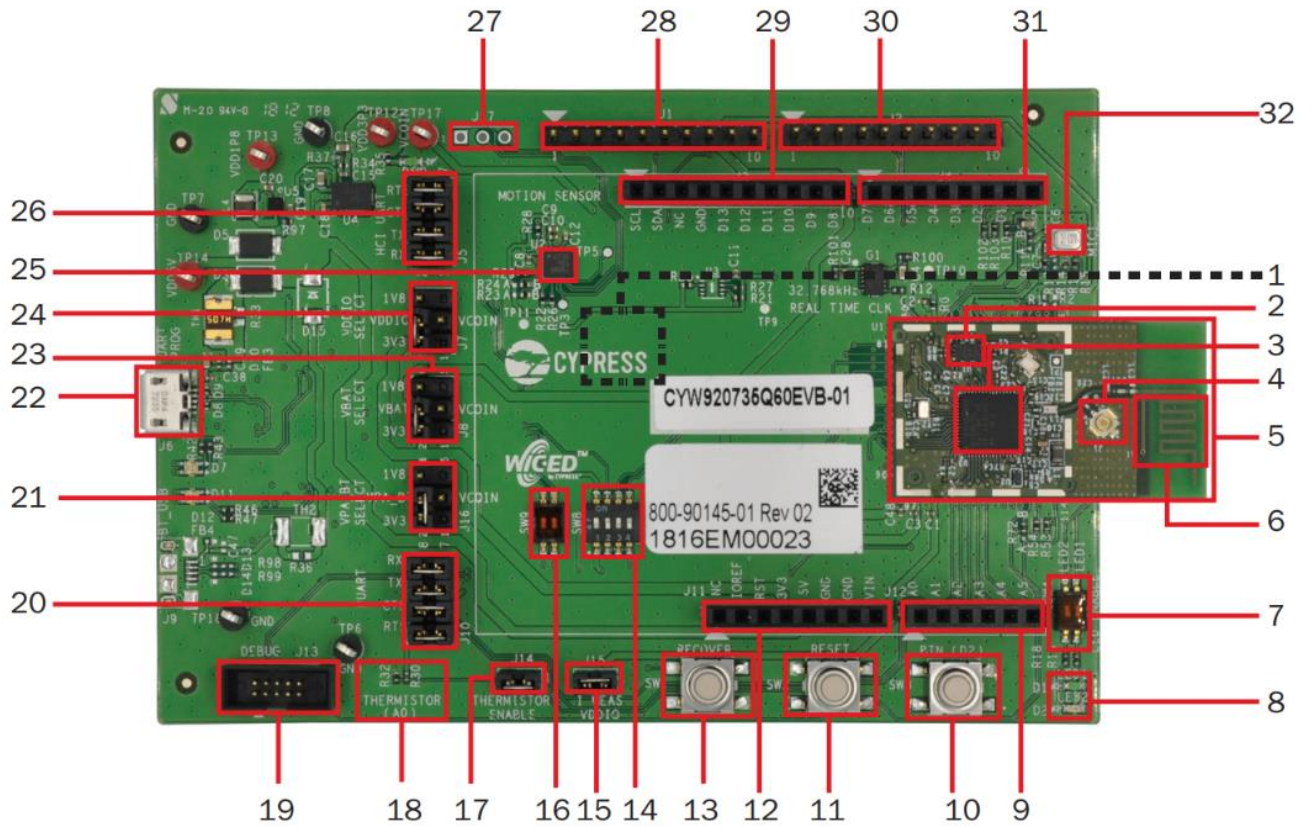


Figure 3-3. CYW920735Q60EVB-01 Evaluation Board

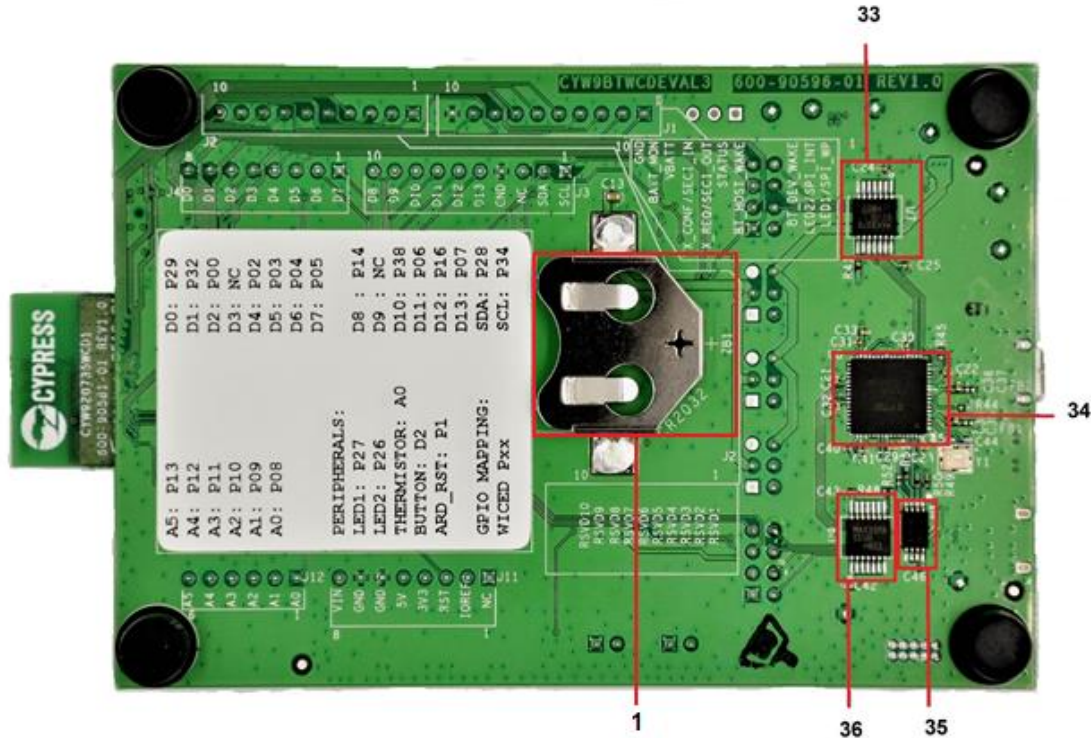


Figure 3-4. CYW920735Q60EVB-01 Evaluation Board (Back View)

1. **Coin Cell Holder (ZB1):** This is a coin cell battery holder located on the bottom side of the development kit.
2. **8 Mb Serial Flash (U1):** This is a SPI Flash of 8-Mbit to store applications and patch codes.
3. **CYW20735(U1, U3):** The Bluetooth (BR- Basic Rate) and Bluetooth Low Energy 5 qualified System-On-Chip from Cypress is the heart of this development kit.
4. **External Antenna Connector (U1.J1):** The external antenna connector is an RF connector fed from the BT_RF pad of the CYW20735 followed by a band pass filter. Please see Hardware design guidelines for more details.
5. **Carrier Module (U1):** The carrier module has the CYW20735 SoC on it. A Bluetooth antenna is etched on the carrier module PCB. The UART and GPIOs are brought out from the SoC pads to interface with the base board.
6. **PCB Antenna (A1):** The PCB antenna is the EM wave radiating part of the evaluation board which is fed from the BT_RF pad of the CYW20735 followed by a band pass filter and an Antenna matching circuit.
7. **LED Enable Switch (SW4):** These DIP switches are used to connect/disconnect the user controlled LEDs from the CYW20735 device.
8. **LEDs (D1 and D2):** These onboard LEDs can be controlled by the CYW20735 device. The LEDs are active LOW, so the pins must be driven to ground to turn ON the LEDs.
9. **Arduino Header (J12):** The Arduino-compatible I/O header brings out pins from CYW20735 to interface with Arduino shields.
10. **User Button (SW3):** This button can be used to provide an input to the CYW20735 device. Note that the button connects the CYW20735 pin to ground when pressed, so the CYW20735 pin must be configured as a digital input with resistive pull-up for detecting the button press.
11. **Reset Button (SW2):** This button can be used to reset the device.
12. **Arduino Header (J11):** The Arduino-compatible I/O header brings out pins from CYW20735 to interface with Arduino shields.
13. **Recovery Button (SW1):** This button is used to put the device in recovery mode. To put the device in recovery mode, press and hold the recovery button, press and release the reset button, then release the recovery button.

14. **Recovery Button and Peripheral Power Switch (SW8):** The DIP switch allows the user to disconnect the supply to onboard peripheral devices like motion sensor, thermistor, analog mic and real-time clock by disabling VDDP. It also allows configuration of the Recovery button connection. The recovery button connection should not be changed for this EVB – the other options are only used on other variants of the base board.
15. **VDDIO Current Measurement Header (J15):** This jumper is used to power the carrier board. To measure the current consumed by the carrier module, remove this jumper and connect an ammeter to the two pins.
16. **SWD/GPIO Switch (SW9):** This DIP switch allows the user to route the functionality of GPIO/SWD lines to either J4 Arduino Header or J13 debug header.
17. **Thermistor Enable Header (J14):** This jumper can be used to connect or disconnect the thermistor from the CYW20735 device.
18. **Thermistor (R30):** The onboard thermistor is a NTC analog thermistor that can be used to measure temperature.
19. **Debug Header (J13):** J13 is a 10-pin interface header that can be used to connect an external debugger via SWD.
20. **PUART Header (J10):** This header can be used to connect or disconnect the PUART pins from the USB-Serial device.
21. **VPA_BT Select Header (J16):** This header is used to select the VPA_BT (Power Amplifier supply) power source. The possible selections are 3.3 V, 1.8 V, or VCOIN which is the coin cell battery holder on the back of the board. Typically, VPA_BT should always be 3.3 V to get the maximum power levels mentioned in the datasheet.
22. **USB Connector (J6):** J6 is a micro USB female connector for connecting a USB cable to a PC.
23. **VBAT Select Header (J8):** This header is used to select the VBAT (Core power supply) power source. The possible selections are 3.3 V, 1.8 V, or VCOIN which is the coin cell battery holder on the bottom side of the board.
24. **VDDIO Select Header (J7):** This header is used to select the VDDIO power source. The possible selections are 3.3 V, 1.8 V, or VCOIN which is the coin cell battery holder on the bottom side of the board.
25. **Motion Sensor (U2):** This is an I²C-based 9-axis inertial module.
26. **HCI UART Header (J5):** This header can be used to connect or disconnect HCI UART from the USB-Serial device.
27. **Motion Sensor Interrupt Test-point (J17):** J17 has 3 test points for the interrupts from the motion sensor (U2).
28. **WICED Header (J1):** This header brings out some pins of the CYW20735 device that are not connected to the Arduino headers. These pins can be used for testing or for custom applications.
29. **Arduino Header (J3):** The Arduino-compatible I/O header brings out pins from CYW20735 to interface with Arduino shields.
30. **WICED Header (J2):** This header brings out some pins of the CYW20735 device that are not connected to the Arduino headers. These pins can be used for testing or for custom applications.
31. **Arduino Header (J4):** The Arduino-compatible I/O header brings out pins from CYW20735 to interface with Arduino shields.
32. **Analog Mic (MIC1):** This is a MEMS microphone capable of recording voice which outputs to the *micn* and *micp* pins of the CYW20735.
33. **BT UART Voltage Level translator (U7):** This voltage level translator IC allows interoperability of devices with different high-level voltage and low-level voltage for input and output operations.
34. **USB to UART converter (U8):** This is a two channel USB to serial UART IC for communicating between the CYW920735 and a PC.
35. **EEPROM (U10):** This EEPROM contains the configuration information for the USB to UART Converter for acting as USB Slave.
36. **PUART Voltage Level translator (U9):** This voltage level translator IC allows the interoperability of devices with different high-level voltage and low-level voltage for input and output operations.

3.2 Jumpers

Table 3-1 to Table 3-7 list the jumper settings on the CYW920735Q60EVB-01 kit.

Jumper J5 (HCI UART)	Default State	Connection on CYW20735 Device	Description
1 and 2	Shorted	UART_RXD	Connects the CYW20735's BT_UART_RX pin to FTDI chip via level translators for USB to UART functionality
3 and 4	Shorted	UART_TXD	Connects the CYW20735's BT_UART_TX pin to FTDI chip via level translators for USB to UART functionality
5 and 6	Shorted	UART_CTS_L	Connects the CYW20735's BT_UART_CTS_L pin to FTDI chip via level translators for USB to UART functionality
7 and 8	Shorted	UART_RTS_L	Connects the CYW20735's BT_UART_RTS_L pin to FTDI chip via level translators for USB to UART functionality

Table 3-1. Jumper J5 Pin Configurations

Jumper J7 (VDDIO selection)	Default State	Connection on CYW20735 Device	Description
2 and 4	Shorted	LHL_VDDIO, BT_VDDIO	Short these pins to supply 3.3 V to VDDIO of the CYW20735 device, as well as all peripherals and sensors.
4 and 6	Open		Short these pins to supply 1.8 V to VDDIO of the CYW20735 device, as well as all peripherals and sensors.
3 and 4	Open		Short these pins to supply VDDIO of the CYW20735 device, as well as all peripherals and sensors from the coin cell supply (VCOIN).

Table 3-2. Jumper J7 Pin Configurations

Jumper J8 (VBAT selection)	Default State	Connection on CYW20735 Device	Description
2 and 4	Shorted	SR_VDDBAT3V	Short these pins to supply 3.3 V to VBAT of the CYW20735 device. Also, use this jumper to measure the current consumption of VBAT when using 3.3 V supply.
4 and 6	Open		Short these pins to supply 1.8 V to VBAT of the CYW20735 device. Also, use this jumper to measure the current consumption of VBAT when using 1.8 V supply.
3 and 4	Open		Short these pins to use the coin cell holder to supply VBAT of the CYW20735 device. Also, use this jumper to measure the current consumption of VBAT when using the coin cell supply (VCOIN).

Table 3-3. Jumper J8 Pin Configurations

Jumper J10 (Peripheral UART)	Default State	Connection on CYW20735 Device	Description
1 and 2	Shorted	P11	Connects the CYW20735's P11 pin to FTDI chip via level translators for USB to UART functionality
3 and 4	Shorted	P10	Connects the CYW20735's P10 pin to FTDI chip via level translators for USB to UART functionality
5 and 6	Shorted	P32	Connects the CYW20735's P32 pin to FTDI chip via level translators for USB to UART functionality
7 and 8	Shorted	P29	Connects the CYW20735's P29 pin to FTDI chip via level translators for USB to UART functionality

Table 3-4. Jumper J10 Pin Configurations

Jumper J14 (Thermistor connect/disconnect)	Default State	Connection on CYW20735 Device	Description
1 and 2	Shorted	P8	Short this jumper to connect thermistor to CYW20735.

Table 3-5. Jumper J14 Pin Configurations

Jumper J15 (VDDIO current measurement)	Default State	Connection on CYW20735 Device	Description
1 and 2	Shorted	LHL_VDDIO, BT_VDDIO	Short this jumper to supply power to the IO Domain (VDDIO) of the CYW20735. Also, use this jumper to measure the current consumption of the IO Domain.

Table 3-6. Jumper J15 Pin Configurations

Jumper J16 (VPA_BT current measurement)	Default State	Connection on CYW20735 Device	Description
2 and 4	Shorted	PALDO_VDDIN_5V**	Short these pins to supply 3.3 V to power amplifier (VPA) of the CYW20735 device. Also, use this jumper to measure the current consumption of VPA_BT when using 3.3 V supply.
4 and 6	Open		Short these pins to supply 1.8 V to power amplifier (VPA) of the CYW20735 device. Also, use this jumper to measure the current consumption of VPA_BT when using 1.8 V supply.
3 and 4	Open		Short this jumper to supply power to the power amplifier (VPA) of the CYW20735. Also, use this jumper to measure the current consumption of VPA_BT when using the coin cell supply (VCOIN).

Table 3-7. Jumper J16 Pin Configurations

** Please note that PALDO_VDDIN_5V should be a minimum of 2.5 V and a maximum of 3.63 V to get the desired power levels. If PALDO_VDDIN_5V is less than 2.5 V, then the power amplifier will not serve its intended purpose. Please make sure that VPA_BT select header is 3.3 V to get the desired power amplification. Please see the datasheet for more details.

NOTE: VDDIO must be greater or equal to VBAT. The CYW20735 uses an on-chip low voltage detector to shut down the chip when supply voltage (VBAT) drops below the operating range. The Shutdown voltage (V_{SHUT}) lies between a minimum of 1.625 V and a maximum of 1.775 V. Please see the datasheet for more details.

3.3 Buttons and Switches

Buttons	Pressed Value	Connection on CYW20735 Device	Description
SW1	GND	RSVD8(SF_SPI_MOSI)	Recovery Button (SW8 configuration need to be as per Table 3-10)
SW2	GND	RST_N	Active low Reset Button
SW3	GND	P0	User application button

Table 3-8. Button Functionality

The DIP switch SW4 enables or disables the two onboard user LEDs. By default, both LED1 and LED2 are enabled.

DIP SW4	Default State	Connection on CYW20735 Device	Description
1	ON	P27	Enables LED1
2	ON	P26	Enables LED2

Table 3-9. SW4 DIP Switches Configuration

The SW8 is a 4 pole DIP switch. Pole 1 of SW8 allows the user to disconnect the VDDIO from the onboard peripherals like Analog mic, LEDs, Real Time clock, 9-axis Motion sensor and Thermistor. Pole 2, 3 and 4 of the SW8 allows the user to connect particular pins to the recovery button. The selections for poles 2, 3, and 4 should not be changed for the CYW920735Q60EVB-01 EVB – they are only used for other variations of the base board.

DIP SW8	Default State	Connection on CYW20735 Device	Description
1	ON	NIL	Enables VDDP (Onboard Peripherals)
2	OFF	BT_UART_CTS_L	Enables BT_UART_CTS_L to recovery button
3	ON	SF_SPI_MOSI	Enables RSVD8 (SF_SPI_MOSI) to recovery button (CYW920735Q60EVB-01 should only use this option)
4	OFF	SDA	Enables SDA to recovery button

Table 3-10. SW8 DIP Switches Configuration

The SW9 is a 2 pole DIP switch. Pole 1 allows P2 to be switched between SWDCK functionality for the debugger interface and GPIO functionality of the Arduino header. Pole 2 allows P3 to be switched between SWDIO functionality for the debugger interface and GPIO functionality of the Arduino header.

DIP SW9	Default State	Connection on CYW20735 Device	Description
1	ON	P2	Connects P2 to the Arduino header pin D4 if the switch pole is in ON state. Connects P2 to the debugger header if the switch pole is in OFF state.
2	ON	P3	Connects P3 to the Arduino header D5 if the switch pole is in OFF state. Connects P3 to the debugger header if the switch pole is in OFF state.

Table 3-11. SW9 DIP Switches Configuration

3.4 Arduino-Compatible Headers

J3, J4, J11, and J12 are the Arduino compatible headers.

Header J3	Arduino PIN	Connection on CYW20735 Device	WICED Enum Name	Description
1	SCL	P34	WICED_P34	I2C SCL
2	SDA	P28	WICED_P28	I2C SDA
3	AREF	NC		Analog Reference
4	GND	GND		Ground
5	D13	P7	WICED_P07	General purpose GPIO
6	D12	P16	WICED_P16	General purpose GPIO
7	D11	P6	WICED_P06	General purpose GPIO
8	D10	P38	WICED_P38	General purpose GPIO
9	D9	NC		NC
10	D8	P14	WICED_P14	General purpose GPIO

Table 3-12. Header J3 Pin Configurations

Header J4	Arduino PIN	Connection on CYW20735 Device	WICED Enum Name	Description
1	D7	P5	WICED_P05	General purpose GPIO
2	D6	P4	WICED_P04	General purpose GPIO
3	D5	P3	WICED_P03	General purpose GPIO
4	D4	P2	WICED_P02	General purpose GPIO
5	D3	NC		NC
6	D2	P0	WICED_P00	User Button
7	D1	P32	WICED_P32	PUART_TX
8	D0	P29	WICED_P29	PUART_RX

Table 3-13. Header J4 Pin Configurations

Header J11	Arduino PIN	Connection on CYW20735 Device	Description
1	NC	NC	NC
2	IOREF	VDDIO	IO reference pin used by shields to determine IO voltage. Connected to VDDIO on this board
3	ARD_RST	BT_RST/ P1	Arduino Reset (R72 install position A) (See RESET) / P1 (R72 install position B)
4	VDD3P3	VDD3P3	3.3 V supply to the Arduino Shield
5	VDD5V	VDD5V	5 V supply to the Arduino Shield
6	GND	GND	GND
7	GND	GND	GND
8	NC	NC	NC

Table 3-14. Header J11 Pin Configurations

Header J12	Arduino PIN	Connection on CYW20735 Device	WICED Enum Name	Description
1	A0	P8	WICED_P08	General purpose GPIO
2	A1	P9	WICED_P09	General purpose GPIO
3	A2	P10	WICED_P10	General purpose GPIO
4	A3	P11	WICED_P11	General purpose GPIO
5	A4	P12	WICED_P12	General purpose GPIO
6	A5	P13	WICED_P13	General purpose GPIO

Table 3-15. Header J12 Pin Configurations

3.5 Other Headers

J1 and J2 are test headers which bring out certain pins of the CYW20735 for testing

Header J1	Connection to Header pin	Connection on CYW20735 Device	WICED Enum Name	Description
1	LED1/SF_WP	P27	WICED_P27	LED1 connection
2	LED2/SPI_INT	P26	WICED_P26	LED2 or SPI interrupt connection
3	BT_DEV_WAKE	NC		No Connect
4	BT_HOST_WAKE	HOST_WAKE		Signal to wake up host
5	STATUS	NC		No Connect
6	TX_REQ/SECI_OUT	NC		No Connect
7	TX_CONF/SECI_IN	NC		No Connect
8	VBATT	NC		No Connect
9	BATT_MON	NC		No Connect
10	GND	GND		Ground

Table 3-16. Header J1 Pin description

Header J2	Connection to Header pin	Connection on CYW20735 Device	Description
1	RSVD1	NC	No Connect
2	RSVD2	NC	No Connect
3	RSVD3	NC	No Connect
4	RSVD4	NC	No Connect
5	RSVD5	NC	No Connect
6	RSVD6	NC	No Connect
7	RSVD7	SF_SPI_MISO	SPI (MISO) External Flash Memory
8	RSVD8	SF_SPI_MOSI	SPI (MOSI) External Flash Memory
9	RSVD9	SF_SPI_CS_L	SPI (Slave Select) External Flash Memory
10	RSVD10	SF_SPI_CLK	SPI (Clock) External Flash Memory

Table 3-17. Header J2 Pin description

J13 is a 10-pin debugger header to debug the CYW920735Q60EVB-01 using SWD.

Header J13	Connection to Header pin	Connection on CYW20735 Device	WICED Enum Name	Description
1	VDDIO	VDDIO		VDDIO reference
2	D5/SWDIO	P3	WICED_P03	Serial Wire Debug Input Output
3	GND	GND		Ground
4	D4/SWDCK	P2	WICED_P02	Serial Wire Debug Clock
5	GND	GND		Ground
6	NC	NC		No Connect
7	NC	NC		No Connect

Header J13	Connection to Header pin	Connection on CYW20735 Device	WICED Enum Name	Description
8	NC	NC		No Connect
9	NC	NC		No Connect
10	BT_RST	RST_N		CYW20735 device reset

Table 3-18. Header J13 Pin description

3.6 USB Serial Interface Chip

An FT-2232-HQ chip is used for onboard programming, debugging, and USB-Serial functionality. It connects to the computer over a USB interface and connects to the CYW20735-B1 device through the HCI UART and PUART pins.

3.7 Kit Power Supply

The kit can be powered by one of two power sources: USB or coin cell battery.

As shown in [Figure 3-2](#) the USB power is connected to two Buck regulators, one regulating the voltage to 1.8 V and the other to 3.3 V. A coin cell battery can be directly connected without the need of a regulator. See [Table 3-2](#), [Table 3-3](#), [Table 3-6](#) and [Table 3-7](#) to understand the jumper settings for power selection.

3.8 Test Points

All power domains are brought out as test points for easy probing.

Label	Description
T6, TP7, TP8, TP18	Test points for Ground
TP12	Test point for VDD3P3
TP13	Test point for VDD1P8
TP14	Test point for VDD5V
TP17	Test point for VCOIN

Table 3-19. Test points for power domains available in CYW920735Q60EVB-01

J17 is a 3-pin test-point for testing purposes of the 9-axis motion sensor (U2) interrupts.

Test-Point J17	Connection to Header pin	Description
1	INT1_A/G	Interrupt 1 from Accelerometer/Gyroscope
2	INT2_A/G	Interrupt 2 from Accelerometer/Gyroscope
3	INT_M	Interrupt from Magnetometer

Table 3-20. Test-point J17 description

3.9 Current Measurement

The CYW20735 device has 3 power domains; VDDIO to power the Lean High Land (LHL) pins, VPA_BT to power the integrated power amplifier and VBAT to power the core. The total current consumption by the device is the sum of the current consumed by the VDDIO, VPA_BT and the VBAT domains. To measure the current consumed by the VDDIO domain, an ammeter can be connected across jumper J15. To measure the current consumed by the VBAT domain, connect an ammeter across pin 4 and one of pins 2, 3, or 6 (depending on the power source) of jumper J8. To measure the current consumed by the VPA_BT domain, connect an ammeter across pin 4 and one of pins 2, 3, or 6 (depending on the power source) of jumper J16.

3.10 SWD Debugging

WICED Studio supports multiple ARM-JTAG adapters for debugging Bluetooth products like the CYW20735. Debugging is possible on CYW920735Q60EVB-01 through SWD (Serial Wire Debug) signals. SWD is a two-wire interface that uses SWDIO (SWD Input Output) and SWDCK (Serial Wire Clock) for debugging the device. These two lines can be brought out to any of the LHL GPIOs on the CYW20735. Please refer to [Table 3-11](#) to enable SWD pins to the debug connector (J13).

Once the hardware configurations are done, please follow the debugger guide for debugging your application using WICED Studio.

3.11 Pin Configuration

The GPIOs on the CYW20735 device can be muxed to various peripherals. For more information on the peripherals that can be routed to the various GPIOs, see the device datasheet. For this board, WICED Studio initializes the GPIOs to the platform default configuration. For example, P34 and P28 are configured as I2C SCL and I2C SDA, respectively. This is done in the file `20735-B1_Bluetooth\platforms\wiced_platform_pin_config.c`. To override the default configuration for an application, add your own pin configuration to the `app` folder and name the file as `<app_name>__pin_config.c`. WICED Studio provides a SuperMux Wizard GUI to configure GPIO pin function mapping and associated configuration, to automatically generate this `<app_name>__pin_config.c` file, and to integrate the code into an existing application. For more information on this tool, see the user guide *WICED-SuperMux-Wizard-User-Manual* located at `20735-B1_Bluetooth\doc`.

4 Code Examples

This chapter explains the functionality of the CYW20735 Thermostat code example. This code example can be found in the folder `20735-B1_Bluetooth\apps\snip\ble\env_sensing_temp`. This chapter also describes the hardware connections and how to verify the output. You can find more code examples and their details in WICED Studio under the `apps` folder.

4.1 Thermostat

4.1.1 Project Description

This project demonstrates the ADC and BLE capability of the CYW20735 device. A negative temperature coefficient type of thermistor is used to measure ambient temperature. The thermistor present on the CYW920735Q60EVB-01 Evaluation board is PN NCU15W104F60RC from Murata. It is a 100 K Ω thermistor at 25 °C with 1% accuracy. The thermistor is connected to GPIO P8 on the CYW20735 device. The ADC is used to measure the change in voltage to determine the temperature. The temperature value is sent over BLE to a client, and on the PUART as DEBUG TRACE messages to a Terminal Emulator such as TeraTerm. This project demonstrates the following features:

- ADC usage
- BLE Environment Sensing Service (ESS)
- Debug Trace messages
- Connection with any central device

The Environmental Sensing Profile in this project consists of one Bluetooth SIG defined service, the ESS. This project implements only the temperature characteristic from this service. This characteristic supports notification, which allows the GATT server to send data to the connected client device whenever new data is available.

The project consists of the following files:

- `thermistor_app.c`

This file contains the `application_start` function which is the entry point and execution of firmware application. It also contains the following functionality:

- System initialization
- Reading the ADC values and converting to temperature
- Callback function registered with the BLE stack to process BLE events from the stack
- GATT events handler
- Function to send temperature value notifications to a GATT client
- Send Debug Trace messages over PUART

- `thermistor_gatt_db.c/h`

These files contain the GATT database definition.

- `wiced_app_cfg.c`

This file contains the BLE stack configuration parameters.

- `thermistor_temp.db.c/h`

These files contain the function to map resistance to temperature values of the thermistor using a lookup table (from the thermistor datasheet).

4.1.2 Hardware Connections

All the jumpers should be in the default configuration as shown in [Table 3-6](#). The board should be connected to a PC using the Micro-B USB cable. Also, make sure that the Thermistor enable (J14) and VDDP enable (SW8.1) jumpers are in the correct positions.

4.1.3 Flow Chart

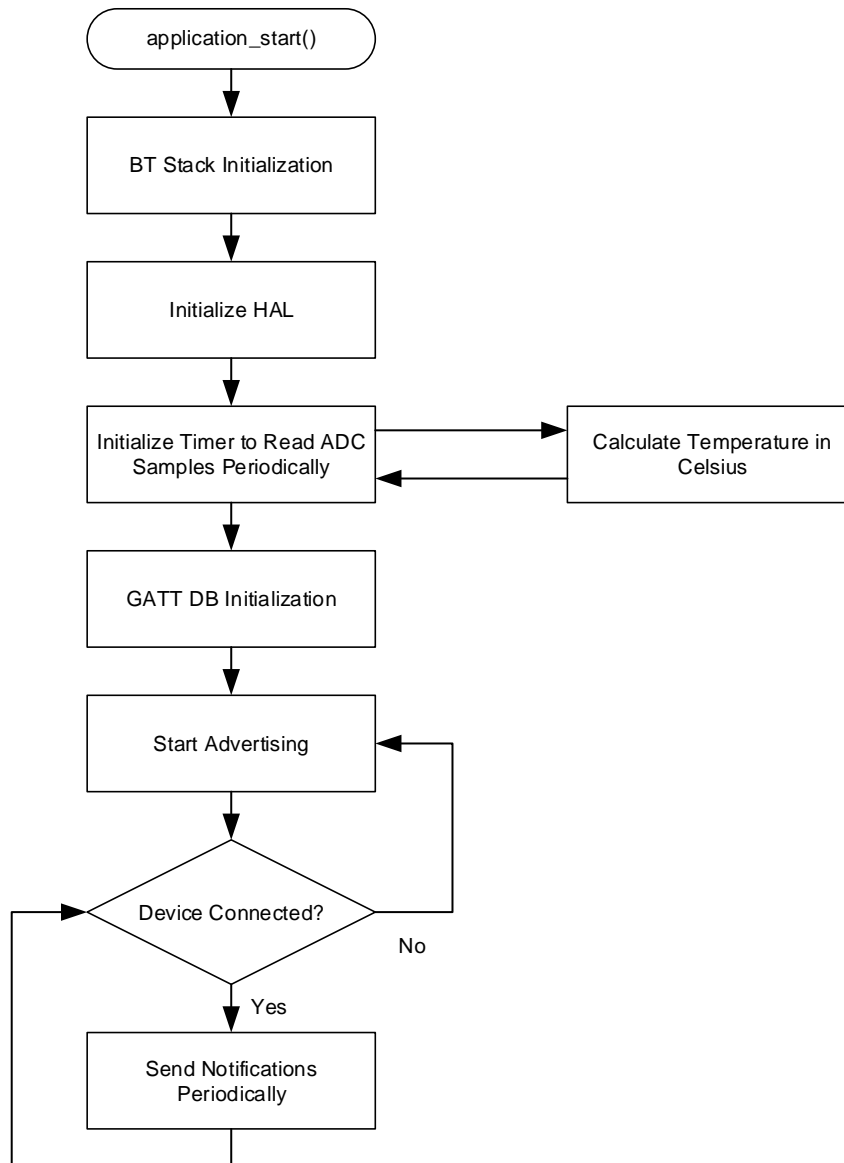


Figure 4-1. thermistor_app Flowchart

4.1.4 Verify Output

To verify the output, follow these steps:

1. Program the kit with the `env_sensing_temp` example project. See [Build and Load a Sample Application](#) to understand how to program the board.
2. Open a Terminal Emulator program such as TeraTerm with the settings shown in [Figure 4-2](#). Note that the COM port will be different for each user. See [Windows](#) to find the PUART COM port number.

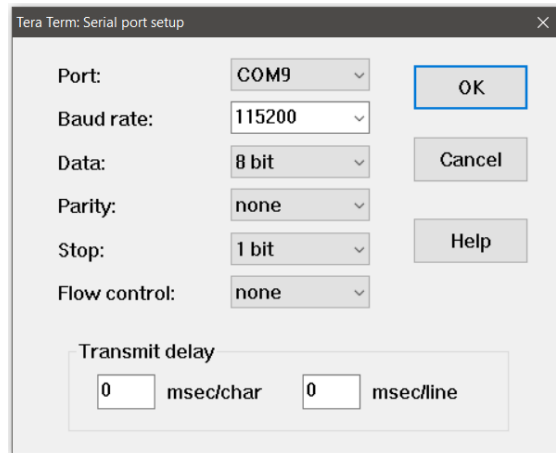


Figure 4-2. Terminal Emulator settings

3. You will see the resistance of the thermistor and the equivalent temperature at an interval of five seconds as shown in [Figure 4-3](#).
4. If you connect to the device using a suitable BLE client (such as a smartphone running an app that supports the ESS profile) you will see that the temperature value is sent to the client at an interval of five seconds. If no client is connected, you will see a “Connection is not up” message instead.

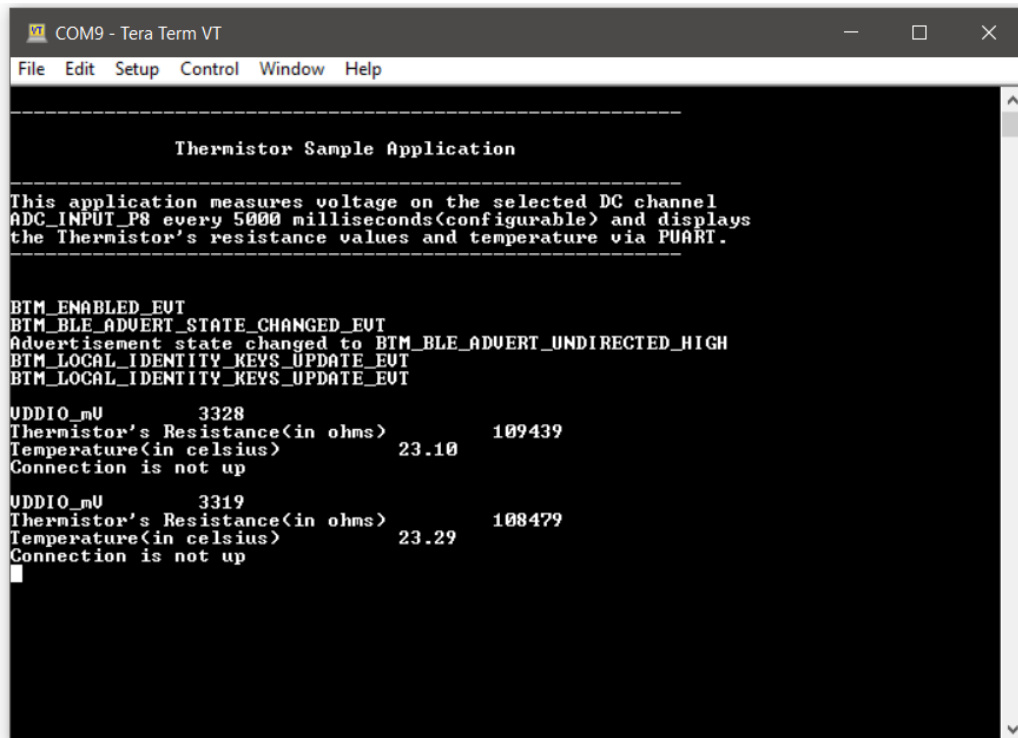


Figure 4-3. `env_sensing_temp` output

5 Hardware

This chapter describes the CYW920735Q60EVB-01 EVB hardware and its different blocks, such as reset control, Arduino-compatible headers, and module connectors.

The schematics for the Base Board and Carrier Module can be found in the following path of WICED Studio Project Explorer:

20735-B1_Bluetooth\doc\CYW920735Q60EVB_01-CarrierBoard-Schematic.pdf

20735-B1_Bluetooth\doc\CYW920735Q60EVB_01-BaseBoard-Schematic.pdf

5.1 Carrier Module

The base board of the CYW920735Q60EVB-01 kit is designed to be modular so that many devices can be used with the same base board. The actual device (CYW20735 in this case) is on the carrier module. The carrier module interface is a generic interface that can be used across many devices. See [Appendix A](#) for a detailed interface description. The Bluetooth antenna is printed on the carrier module PCB. The UART signals and GPIOs are brought out to the module pins to interface with the base board.

5.1.1 CYW20735

The CYW920735Q60EVB-01 kit employs the CYW20735B1KUMLG part which is a 60-QFN package. This board uses all 23 LHL IOs provided by the 60-QFN package. This board utilizes the internal power amplifier for the radio.

5.1.2 Antenna

A PCB antenna is printed on the carrier module. This antenna is matched to 50Ω when the CYW920735Q60EVB-01 kit is sitting on a table. [Table 5-1](#) lists the S11 measurement. See AN91445 – Antenna Design and RF Layout Guidelines for additional information.

2402 MHz	2441 MHz	2480 MHz
-13.2 dB	-14.6 dB	-12.6 dB

Table 5-1. Antenna S11 measurement

5.1.3 Crystal

The CYW20735 carrier module has two crystals onboard. A 24 MHz crystal (XTAL) is the main crystal. This XTAL must have an accuracy of +/-20 ppm as defined by the Bluetooth specification. A 32.768 kHz crystal provides accurate timing during low power operation. See the CYW20735 datasheet for crystal requirements.

5.1.4 External Serial Flash

The CYW20735 Carrier module has an 8Mbit serial flash on board to store applications and patch codes. The Gigadevice GD25WD80CEIG is the 8Mbit serial flash used in the CYW920735Q60EVB-01 kit. The WICED SDK supports JEDEC's CFI (Common Flash Interface) compatible SPI Flash chips.

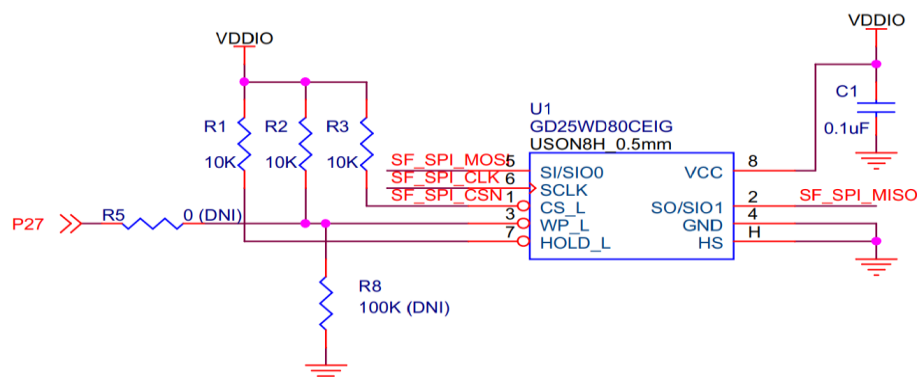


Figure 5-1. External Serial Flash

5.2 Base Board

The CYW9BTCDEVAL1 is a baseboard on which the CYW20735 Carrier Module is soldered.

5.3 Serial Communication between CYW20735 and FTDI USB-Serial Device

The onboard FTDI device is a two-channel USB-Serial converter. The USB-serial pins of the FTDI device are hard-wired to the HCI UART and PUART pins of the CYW20735 device. An EEPROM device is connected to the FTDI USB-Serial device to store its configuration.

5.4 Power

The power supply system on this board is versatile, allowing the input supply to come from the following sources:

- 1.8 V or 3.3 V from the onboard USB connector
- 3 V from a coin cell battery

Figure 5-2 shows the power architecture of the evaluation board.

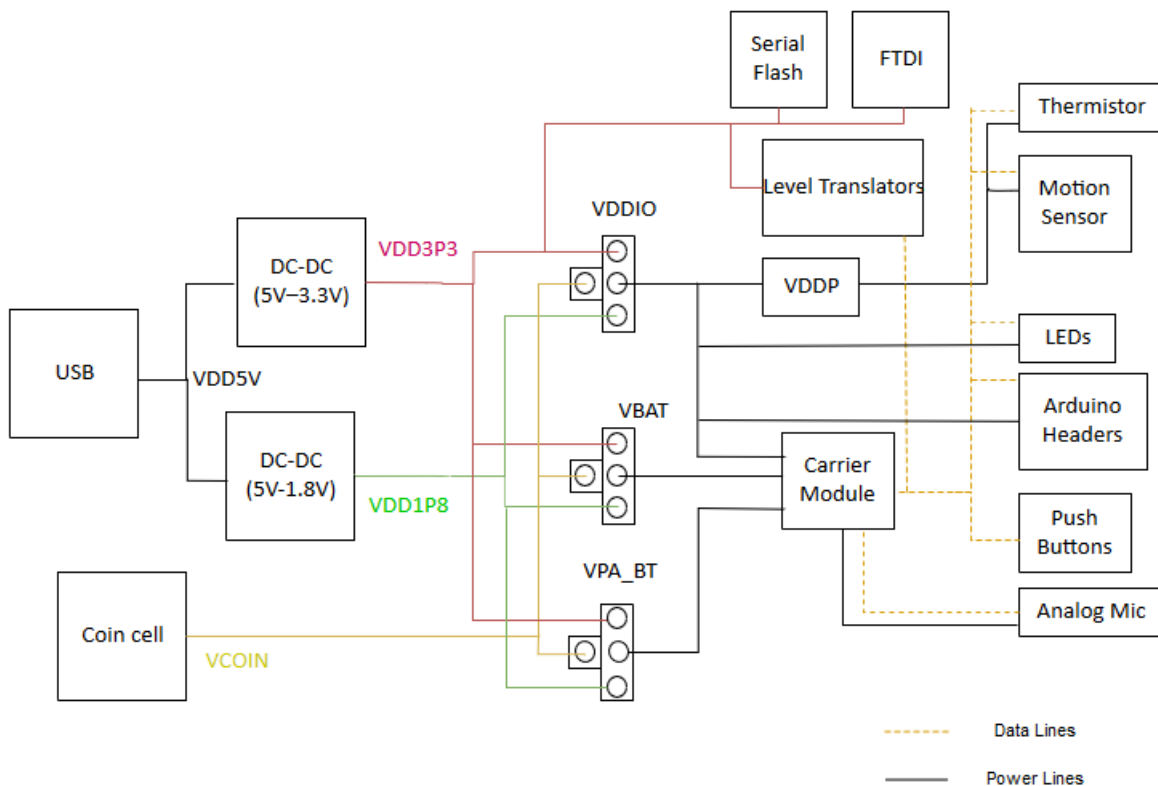


Figure 5-2. Power Architecture

Power supply options can be selected via jumper settings on J7, J8 and J16. See [Table 3-2](#), [Table 3-3](#), and [Table 3-7](#) for the different jumper settings for J7, J8 and J16. The resistors R25 and R21 are the pull-up resistors for I2C lines to motion sensor i.e. SCL and SDA. Please note that SW8.1(VDDP) should be turned on for any I2C devices to be connected because the pull-up voltages for SCL and SDA are supplied from VDDP.

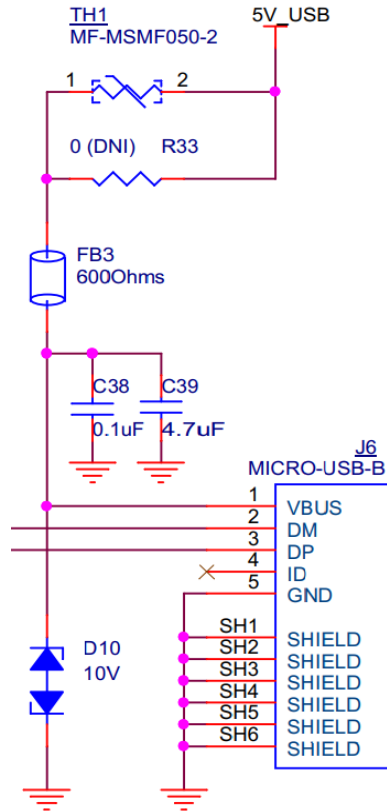


Figure 5-3. 5 V Power Supply from USB

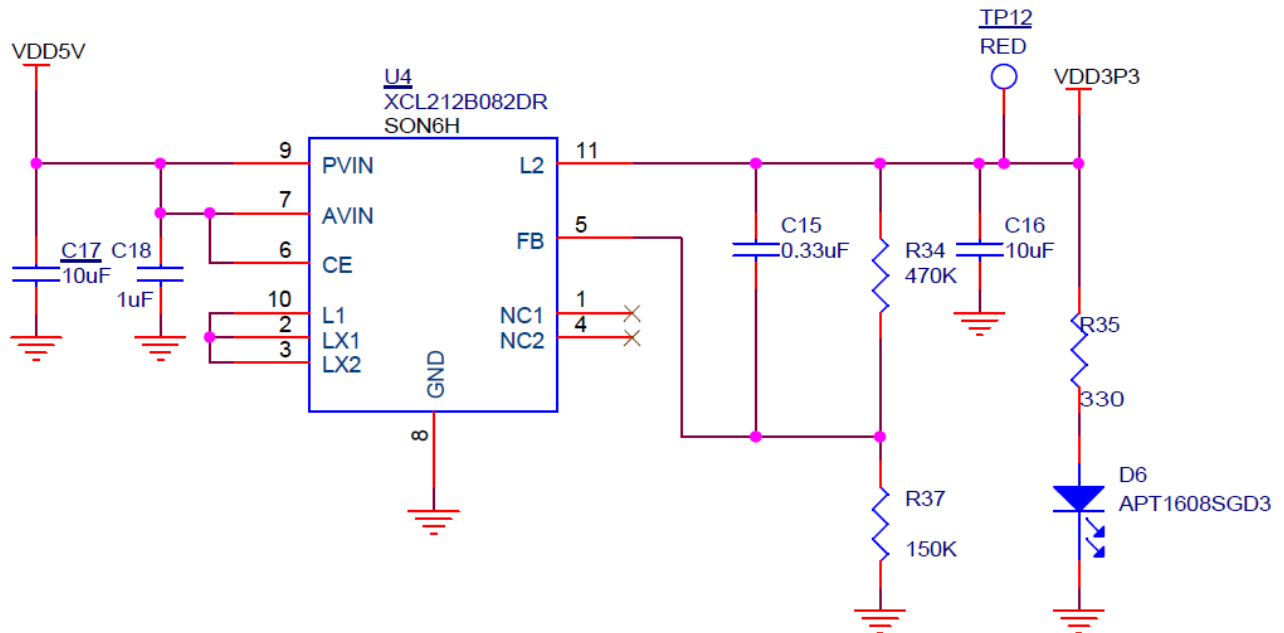


Figure 5-4. 3.3 V Regulator Circuit

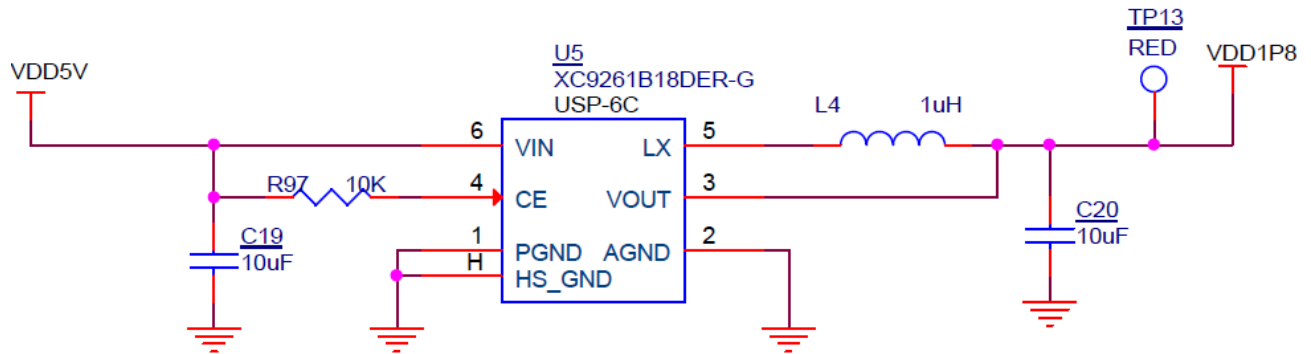


Figure 5-5. 1.8 V Regulator Circuit

VDDIO Supply Options:
 Short J7 Pos 2-4: Power VDDIO from VDD3P3
 Short J7 Pos 4-6: Power VDDIO from VDD1P8
 Short J7 Pos 3-4: Power VDDIO from Coin Cell

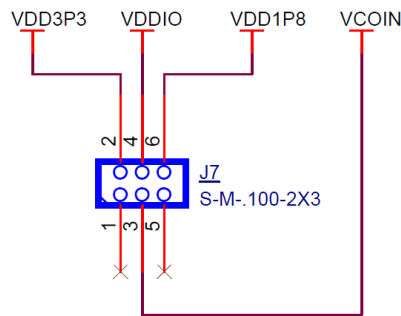


Figure 5-6. Jumper J7 for VDDIO Selection

VBAT Supply Options:
 Short J8 Pos 2-4: Power VBATT from VDD3P3
 Short J8 Pos 4-6: Power VBATT from VDD1P8
 Short J8 Pos 3-4: Power VBATT from Coin Cell

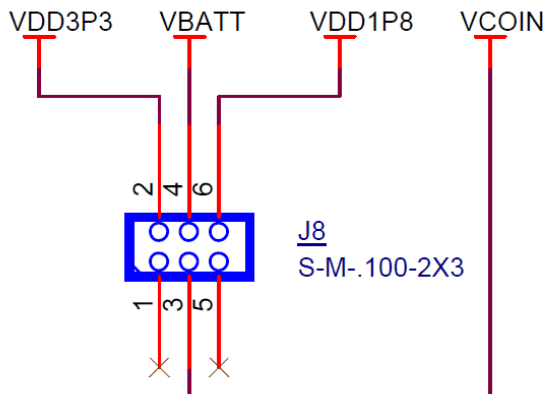


Figure 5-7. Jumper J8 for VBAT Selection

VPA_BT Supply Options:
 Short J16 Pos 2-4: Power VPA_BT from VDD3P3
 Short J16 Pos 4-6: Power VPA_BT from VDD1P8
 Short J16 Pos 3-4: Power VPA_BT from Coin Cell
 See kit user guide for list of supported operation modes.

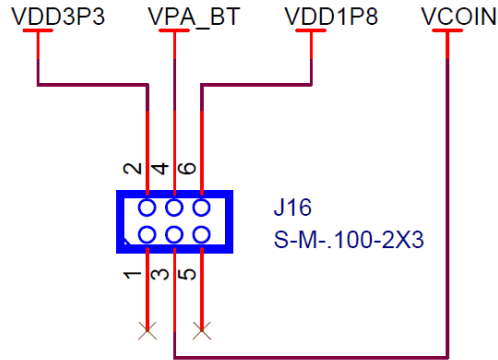
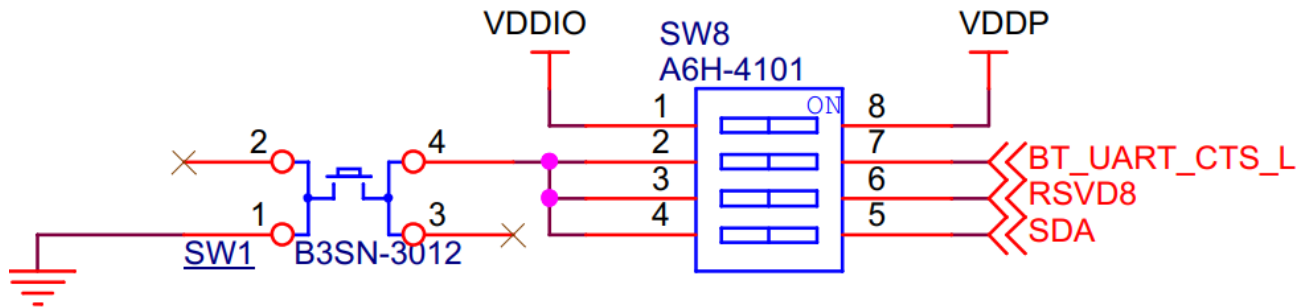


Figure 5-8. Jumper J16 for VPA_BT Selection



SW8:
 Pos 1 on: Connect VDDIO to VDDP
 Pos 2 on: Select BT_UART_CTS_L as recovery BTN
 Pos 3 on: Select RSVD8 ($\overline{SF_MOSI}$) as recovery BTN
 Pos 4 on: Select SDA as recovery BTN

Figure 5-9. DIP Switch SW8 functionality

5.5 RESET

The reset circuit on the board consists of a Reset button (SW2) connected to ground and a voltage detector reset IC. The RESET_N pin on the CYW20735 should be released 50 ms or more after the VDDIO supply voltage has stabilized. The voltage detector IC is used to provide this delay.

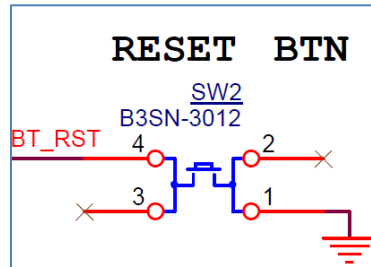


Figure 5-9. Reset Button Circuit

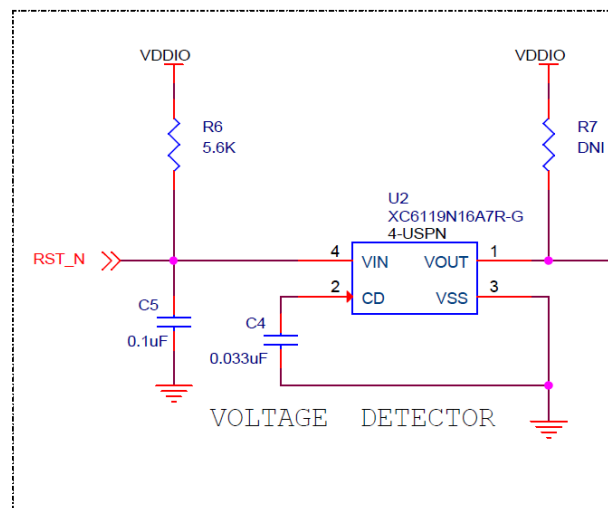


Figure 5-10. Voltage detector circuit in the carrier module

The reset button (SW2) always connects to the BT_RST input on the device so that the device can be reset by pushing the button.

In the default configuration of R72 shown below (position B-C), device pin P1 will be routed to the ARD_RST header so the reset button will not toggle the Arduino header reset pin, but device pin P1 can be used to monitor or drive the Arduino header reset pin.

When R72 is in position (B-C), the reset button is routed to the Arduino header reset pin in addition to the device so both the device and shields can be reset by pressing SW2. Alternately in this case, the device can be reset by driving the Arduino header reset pin low allowing for an external reset source.

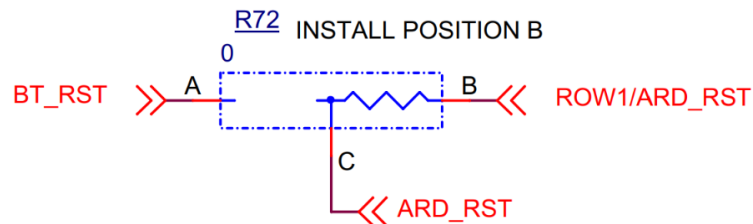


Figure 5-11. Reset to Arduino Header

5.6 Thermistor

The thermistor circuit is a simple voltage divider circuit consisting of an NTC thermistor that is 100 KΩ at 25 °C and a fixed 100 KΩ resistor. The divided voltage is fed in to A0 and the voltage level determines the ambient temperature. The part number of the thermistor used on this kit is PN NCU15W104F60RC.

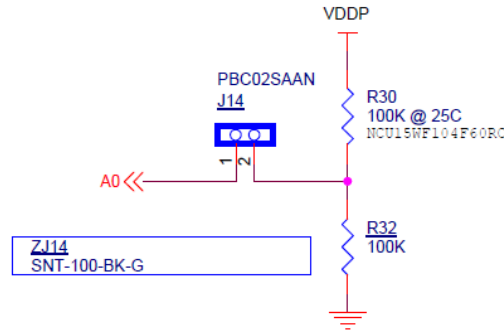


Figure 5-12. Thermistor Circuit

5.7 Motion Sensor

The CYW920735Q60EVB-01 has an onboard 9-axis motion sensor (LSM9DS1). It has three acceleration channels, three angular rate channels, and three magnetic field channels. The CYW20735 device communicates with this sensor over I²C. The I²C address to access the accelerometer and gyroscope is 0xD4 and to access the magnetometer, address is 0x38. The sensor supports 100 kHz and 400 kHz speed. See the LSM9DS1 datasheet to understand how to access the registers.

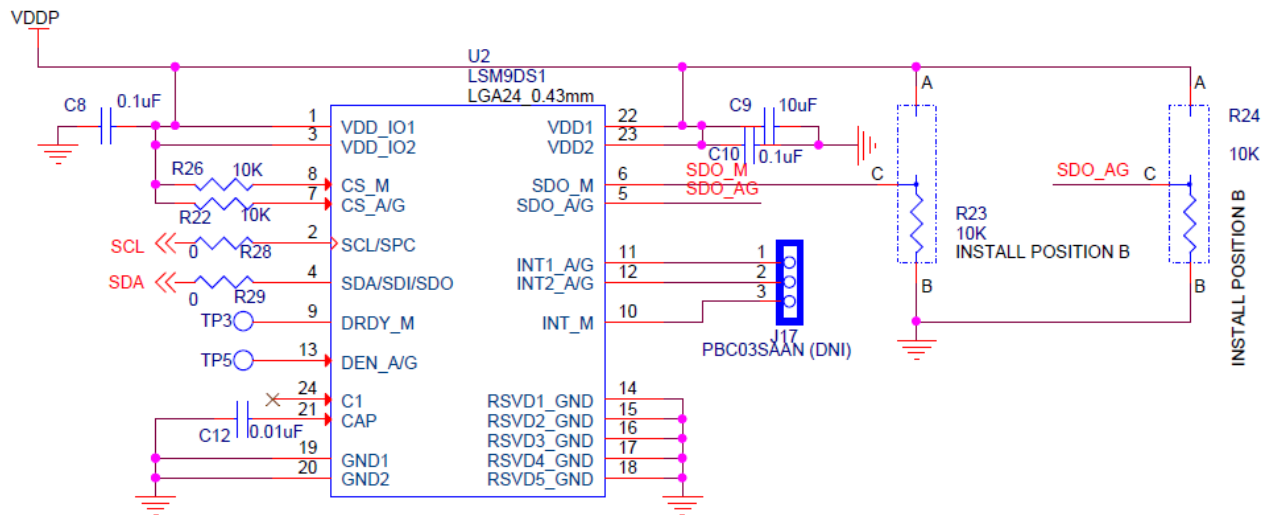


Figure 5-13. Motion Sensor Circuit

5.8 LED

There are two onboard user LEDs on this kit. LED1 is controlled by P27. LED2 is controlled by P26. You can enable or disable the LEDs using DIP switch SW4. See Table 3-9 for DIP switch configuration. TLMY1000-GS08 (Yellow) requires a typical voltage of at least 1.8 V (can be as high as 2.6 V) to operate. LTST-C190CKT (Red) requires a typical voltage of at least 1.8 V (can be as high as 2.4 V) to operate.

Note: The LEDs are initialized by default on bootup. See the file `20735-B1_Bluetooth/platforms/CYW920735Q60EVB_01/wiced_platform.h` for the LED enumeration.

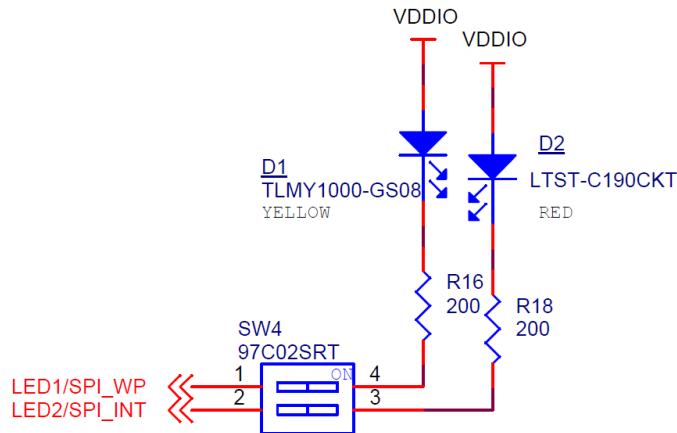


Figure 5-14. LED Circuit

5.9 Analog Mic

There is an onboard Analog microphone that can be used as voice input. MIC1 is a MEMS microphone and it is omnidirectional. The operating voltage of MIC1 is 1.5V to 3.63V. MIC1 can be powered by either VDDP or by the CYW20735's MIC_BIAS pad. The MIC_BIAS output voltage depends on the supply voltage to MIC_AVDD pad of CYW20735. Typical MIC_BIAS output voltage is 2.1V when supply voltage(MIC_AVDD) is 2.5V. Please refer to the datasheet for more details on ADC Electrical Specifications. In CYW920735Q60EVB-01 kit, the MIC_AVDD is powered by VDDIO which can have either 1.8V or 3.3V supply voltage. The R11 resistor's position will determine the supply voltage to MIC1. The default install position of R11 is B-C which supplies the MIC_BIAS from CYW20735 to MIC1. In order to change the supply voltage of MIC1 to VDDP, please change the install position of R11 to A-C.

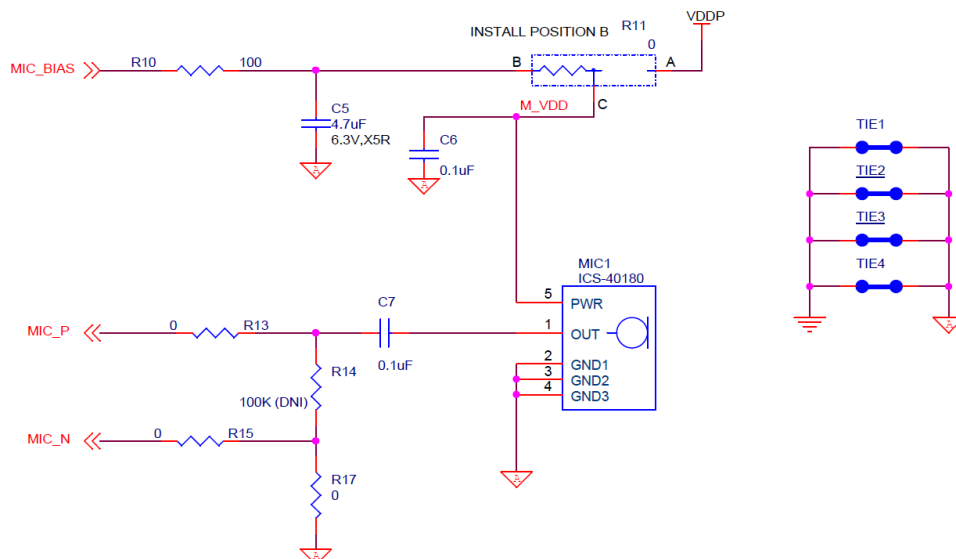


Figure 5-15. Analog Mic

5.10 Push Buttons

The CYW920735Q60EVB-01 has a reset button, recovery button, and a user button. See the [RESET](#) section for details on the Reset button. See the [Build and Load a Sample Application](#) section for details on using the recovery button during kit programming. One user button (SW3) is connected to the P0 pin of the CYW20735 device.

Note: The user button is initialized by default on bootup. See the *20735-B1_Bluetooth/platforms/CYW920735Q60EVB_01/wiced_platform.h* file for button enumeration.

Appendix A. CYW20735 Device IO Mapping

Table A.1 maps the CYW20735 device IOs to headers and sensors on the baseboard. It also lists the carrier module interface definition.

Carrier Module Pin Number	Carrier Module Pin Name	CYW20735 Pin	Base Board connection 1	Base Board connection 2	Base Board connection 3	WICED Enum Name
HS5	GND_HS5					
63	BATT_MON	NC		J1.9		
64	RSVD10	SF_SPI_CLK		J2.10		
65	RSVD9	SF_SPI_CS_L		J2.9		
66	RSVD8	SF_SPI_MOSI	SW8.6 (DIP Switch ON for Recovery Button)	J2.8		
67	RSVD7	SF_SPI_MISO		J2.7		
68	GND_68	Ground	Ground			
69	BT_UART_RXD	BT_UART_RX		J5.1		
70	BT_UART_TXD	BT_UART_TX		J5.3		
71	BT_UART_RTS_N	BT_UART_RTS		J5.7		
72	BT_UART_CTS_N	BT_UART_CTS	SW8.7 (DIP Switch ON for Recovery Button)	J5.5		
73	GND_73	Ground	Ground			
74	MIC_BIAS	MIC_BIAS	Mic (Power Supply)		MIC1	
75	MICP	MIC_P	Mic (Output)		MIC1	
76	MICN	MIC_N	Mic (Output)		MIC1	
77	MIC_AVDD	MIC_AVDD	VDDIO			
78	GND_78	Ground	Ground			
79	EXT_LPO	EXT_LPO	RTC			
80	GND_80	Ground	Ground			
HS6	GND_HS6	Ground	Ground			
81	COL7/SPI_CS/D10	P38		J3.8		WICED_P38
82	COL8/SPI_MISO/D12	P16		J3.6		WICED_P16
83	ROW2/DMIC_CLK/D4	P2	SW9.2 (DIP Switch)	J4.4 (ON for Arduino Headers)	J13.2 (OFF for Debugger)	WICED_P02
84	ROW3/DMIC_DATA/D5	P3	SW9.1 (DIP Switch)	J4.3 (ON for Arduino Headers)	J13.4 (OFF for Debugger)	WICED_P03
85	ROW4/PCM_OUT/I2S_D0/D6	P4		J4.2		WICED_P04
86	RSVD6	Reserved		J2.6		
87	GND_87	Ground	Ground			
88	ROW5/PCM_IN/I2S_DI/D7	P5		J4.1		WICED_P05
89	ROW6/SPI_MOSI/D11	P6		J3.7		WICED_P06

Carrier Module Pin Number	Carrier Module Pin Name	CYW20735 Pin	Base Board connection 1	Base Board connection 2	Base Board connection 3	WICED Enum Name
90	ROW7/SPI_CLK/D13	P7		J3.5		WICED_P07
91	RSVD5	Reserved		J2.5		
92	COL0/A0	P8	J14.1 Jumper	J12.1	Thermistor	WICED_P08
93	COL6/MCLK/D8	P14		J3.10		WICED_P14
94	GND_94	Ground	Ground			
95	BT_USB_DP	NC		J9.3		
96	BT_USB_DM	NC		J9.2		
HS7	GND_HS7	Ground	Ground			
97	VPA_BT	VPA_BT	VPA_BT	J16.4		
98	VBATT	VBAT	VBATT	J8.4		
99	GND_99	Ground	Ground			
100	VDDIO	VDDIO	VDDIO	J15.1		
101	BT_REG_ON	NC	NC			
102	COL1/A1	P9		J12.2		WICED_P09
103	PUART_CTS_N/COL2/A2	P10		J12.3, J10.3		WICED_P10
104	PUART_RTS_N/COL3/A3	P11		J12.4, J10.1		WICED_P11
105	COL4/PCM_CLK/I2S_CLK/A4	P12		J12.5		WICED_P12
106	COL5/PCM_SYNC/I2S_WS/A5	P13		J12.6		WICED_P13
107	GND_107	Ground				
108	LED4/D9	NC		J3.9		
109	LED3/D3	NC		J4.5		
110	ROW0/BUTTON/D2	P0	SW3 (User Button)	J4.6		WICED_P00
111	ROW1/ARD_RST	P1	If Install position of R72 is in B position, it routes P1 pin	J11.3		WICED_P01
112	SCL	P34		J3.1	Motion sensor	WICED_P34
113	IR_TX	NC	NC			
114	BT_RST	RST_N	SW2 (RST Button)	J13.10		
HS8	GND_HS8	Ground	Ground			
115	SDA	P28	SW8.5 (DIP Switch ON for Recovery Button)	J3.2	Motion sensor	WICED_P28
116	LED2/SPI_INT	P26	SW4.2 DIP Switch	J1.2		WICED_P26
117	LED1/SF_WP	P27	SW4.1 DIP Switch	J1.1		WICED_P27
118	PUART_TXD/D1	P32		J13.4, J4.7, J10.5		WICED_P32
119	PUART_RXD/D0	P29		J13.2, J4.8, J10.7		WICED_P29
120	GND_120	Ground	Ground			

Carrier Module Pin Number	Carrier Module Pin Name	CYW20735 Pin	Base Board connection 1	Base Board connection 2	Base Board connection 3	WICED Enum Name
121	RSVD4	Reserved		J2.4		
122	RSVD3	Reserved		J2.3		
123	RSVD2	Reserved		J2.2		
124	RSVD1	Reserved		J2.1		
125	GND_125	Ground	Ground			
126	STATUS	NC		J1.5		
127	TX_REQ/BT_SECI_OUT	NC		J1.6		
128	TX_CONF/BT_SECI_IN	NC		J1.7		
129	BT_HOST_WAKE	HOST_WAKE		J1.4		
130	BT_DEV_WAKE	NC		J1.3		

Table A-1. Carrier Module Interface and Pin Connections

Document Revision History

Document Title: CYW920735Q60EVB-01 Evaluation Kit User Guide

Document Number: 002-23764

Revision	ECN	Origin of Change	Issue Date	Description of Change
**	6168755	SIRK	05/17/2018	Initial release.

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](#) | [PSoC 6 MCU](#)

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

© Cypress Semiconductor Corporation, 2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.