Chapter 1: Tour of Cypress Bluetooth

After completing Chapter 1 (this chapter) you will understand a top-level view of the ModusToolbox Bluetooth ecosystem components, including the chips, modules, software, documentation, support infrastructure and development kits. You will have ModusToolbox installed and working on your computer and understand how to program an existing project into a kit.

1.1 Tour of ModusToolbox

ModusToolbox is a set of tools that allow you to develop, program, and debug applications for Cypress MCUs. The Eclipse IDE for ModusToolbox is an Eclipse-based development environment that is included as part of the ModusToolbox software installer, but it is not the only supported environment. There is a command line interface (CLI) that can be used interchangeably with IDE applications. In addition, users can use ModusToolbox software with their own preferred IDE, such as Visual Studio Code or IAR.

For this class, we will focus on the Eclipse IDE for ModusToolbox with a little information on using the CLI. You'll also get a chance to try using Visual Studio Code with ModusToolbox Bluetooth applications in an advanced exercise if you want.

For Windows, ModusToolbox software is installed, by default, in *C:/Users/<UserName>/ModusToolbox.* Once installed, the IDE will show up in Windows under *Start->All Programs->ModusToolbox <version>*.

If you install ModusToolbox in a non-default location, you will need to set the CY_TOOLS_PATHS environment variable for your system to point to the *ModusToolbox/tools_<version>* folder or set that variable in each Makefile. Note: If your home folder has a space in the name, you will need to install in a different location - see the ModusToolbox Installation Guide (available on cypress.com) for details.

1.1.1 Make/ Build Infrastructure

ModusToolbox uses a GNU "make" based system to create, build, program, and debug projects. The installation provides a set of tools required for all the operations to work. Those tools include:

- Gnu Make 3.81 or newer
- Git 2.20 or newer
- Python3
- GCC
- OpenOCD with supports for Cypress devices
- Configurator tools
- Library manager
- New project GUI and CLI utilities
- ModusShell
- Eclipse IDE for ModusToolbox

These tools can be found in the ModusToolbox installation folder under *tools_<version>* except for the Eclipse IDE which is under *ide_<version>*.

The general application development flow is as follows. All these steps can be done using an IDE or on the command line. You can even switch back and forth seamlessly between the two.

- Clone the wiced_btsdk (this is a shared library for all Bluetooth applications in a workspace – more on this later)
- Clone the application (either from a GitHub code example or a local set of files)
- Get the libraries required by the application

- Configure the device and its peripherals using configurators
- Write the application code
- Build/Program/Debug

1.2 Network Proxy Settings

Depending on the location of this class you may need to set up your network proxy settings for ModusToolbox - specifically the Project Creator and Library Manager since they both require internet access. Proxy settings will vary from site to site.

If you are on an internal Cypress network, you can find a list of proxy servers here: https://itsm.cypress.com/solutions/726105.portal. If you are at a non-Cypress site, contact that site's IT department for the proxy name and port.

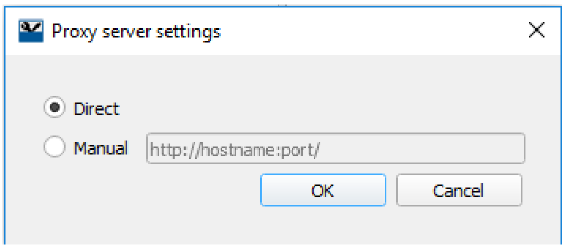
When you run the Project Creator or Library Manager, the first thing it does is look for a remote manifest file. If that file isn't found, you will not be able to go forward. In some cases, it may find the manifest but then fail during the project creation step (during git clone). If either of those errors occur, it is likely due to one of these reasons:

- You are not connected to the internet. In this case, you can choose to use offline content if you have previously downloaded it. Offline content is discussed in the next section.
- You may have incorrect proxy settings (or no proxy settings).

To view/set proxy settings in the Project Creator or Library Manager, use the menu option Settings > Proxy Settings…

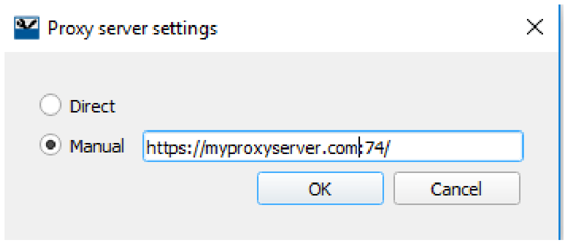
If your network doesn't require a proxy, choose "Direct". If your network requires a proxy choose "Manual". Enter the server name and port in the format http:/hostname:port/.

No Proxy



[✓ Direct] Manual http://hostname:port/

Manual Proxy



Direct [✓ Manual] https://myproxyserver.com:74/

Once you set a proxy server, you can enable/disable it just by selecting Manual or Direct. There is no need to re-enter the proxy server time you connect to a network requiring that proxy server. The tool will remember your last server name.

The settings made in either the Project Creator or Library Manager apply to the other.

1.3 Offline Content

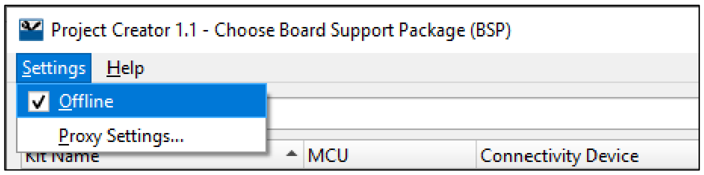
Beginning with ModusToolbox version 2.1, Cypress provides a zipped-up bundle of the GitHub repos to allow you to work offline, such as on an airplane or if for some reason you don't have access to GitHub. To set this up, you must have access to cypress.com in order to download the zip file. After that, you can work offline.

Go to https://www.cypress.com/products/modustoolbox-software-environment and download the "modusttoolbox-offline-content-x.x.x.<build>.zip" file.

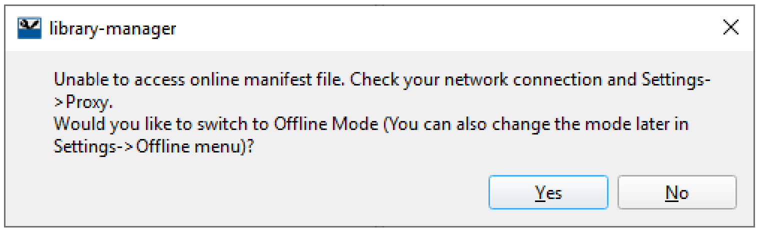
Extract the "offline" directory under the location where you have ModusToolbox installed in a hidden directory named ".modustoolbox". In most cases, this is in your User Home directory. After extracting, the path should be similar to this:

C:\Users\<username>\.modustoolbox\offline

To use the offline content, toggle the setting in the Project Creator and Library Manager tools, as follows:



Also, if you try to open these tools without a connection, a message will ask you to switch to Offline Mode.


library-manager

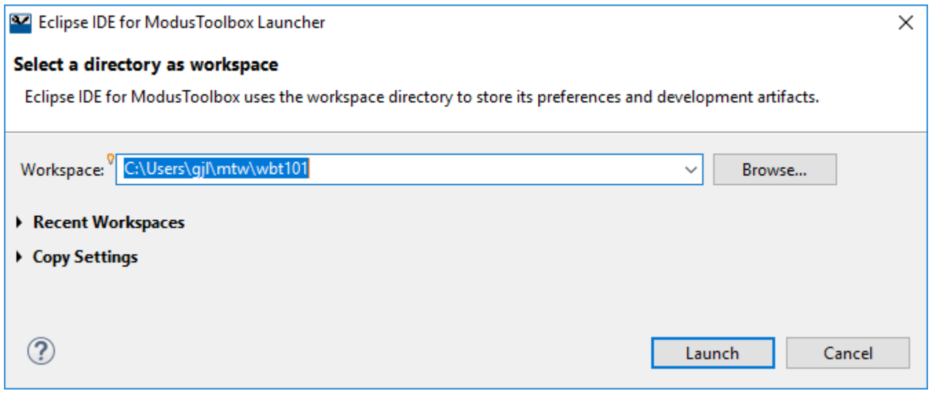
Unable to access online manifest file. Check your network connection and Settings->Proxy.
Would you like to switch to Offline Mode (You can also change the mode later in Settings->Offline menu)?

1.4 Tour of Eclipse IDE for ModusToolbox

1.4.1 First Look

When you first run the Eclipse IDE, you will create a Workspace to hold your applications. The default Workspace location is *C:/Users/<UserName>/mtw*, but you can have as many Workspaces as you want, and you can put them anywhere you want. I usually put each workspace under a folder inside *C:/Users/<UserName>/mtw* such as *wbt101* for the exercises in this class.

Each time you open Eclipse IDE you will need to provide a workspace name such as the following:


Eclipse IDE for ModusToolbox Launcher

Select a directory as workspace

Eclipse IDE for ModusToolbox uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\gjl\mtw\wbt101

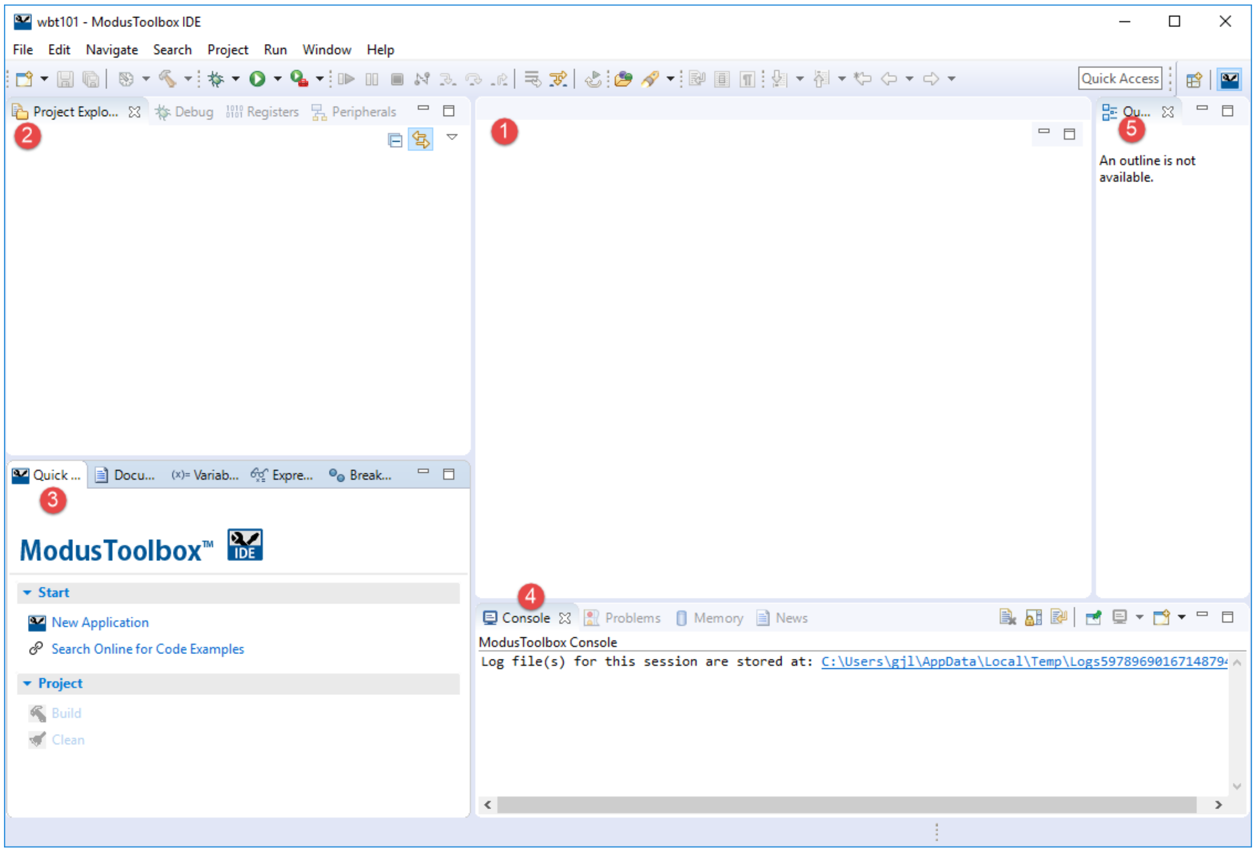
Whenever you want to switch to a different Workspace or create a new one, just use the menu item *File->Switch Workspace* and either select one from the list or select *Other…* to specify the name of an existing or new Workspace.

After clicking Launch, the IDE will start, and you will see the (empty) ModusToolbox perspective.

A perspective in Eclipse is a collection of views. The ModusToolbox perspective combines editing and debugging features. You can also create your own custom perspectives if you want a different set or arrangement of windows.

You can always get back to the ModusToolbox perspective by selecting it from the button in the upper right corner of the IDE, clicking the Open Perspective button and choosing ModusToolbox, or from *Window->Perspective->Open Perspective->Other->ModusToolbox*.

If you unintentionally change something in the perspective and want to reset to the default, you can use *Window->Perspective->Reset Perspective*.



```
ModusToolbox Console
Log file(s) for this session are stored at: C:\Users\gjl\AppData\Local\Temp\Logs59789690167148794
```

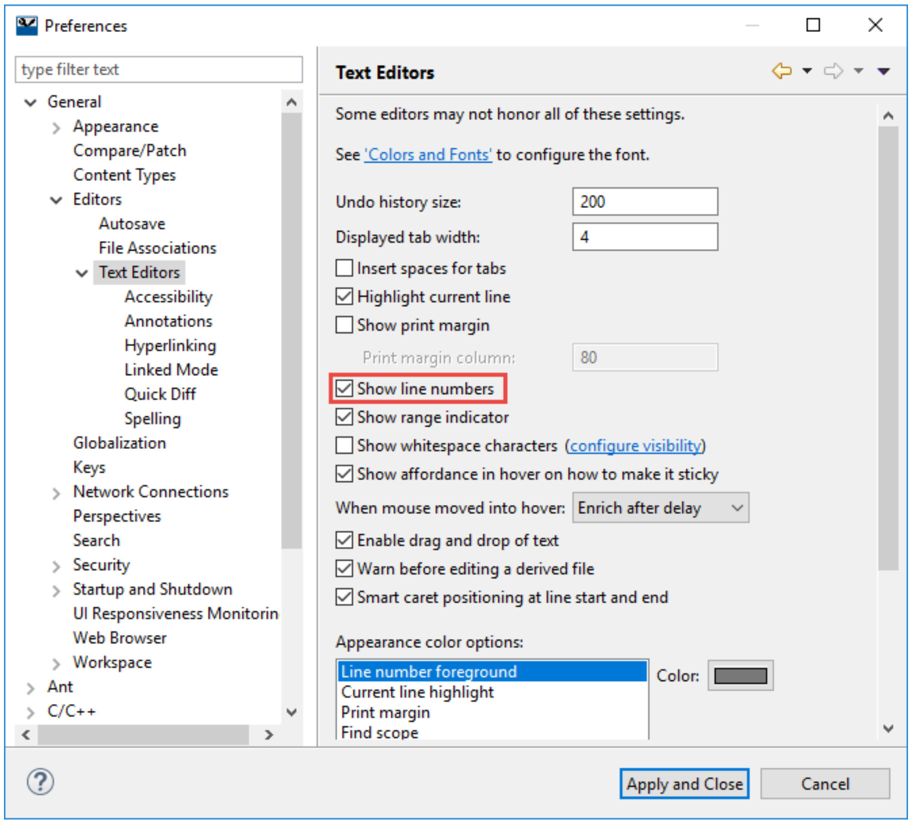
The major views are:

1. File Viewer/Editor
2. Project Explorer
3. Quick Panel
4. Console / Problems
5. Outline

If you close a view unintentionally, you can reopen it from the menu *Window->Show View*. Some of the views are under *Window->Show View->Other…* You can drag and drop windows and resize them as you desire.

1.4.2 Customization

Eclipse is extremely flexible – you can customize almost anything if you know where to look. A good place to start for general Eclipse settings is *Window->Preferences*. One that I always turn on is *General->Editors->Text Editors->Show line numbers* (you can also access this from a pop-up menu by right clicking along the left edge of the code editor window). Most of these settings are at the workspace level but when you create a new workspace there is an option under "Copy Settings" to copy various settings from an existing workspace if desired.


Preferences

type filter text

- General
  - Appearance
  - Compare/Patch
  - Content Types
  - Editors
    - Autosave
    - File Associations
    - Text Editors
      - Accessibility
      - Annotations
      - Hyperlinking
      - Linked Mode
      - Quick Diff
      - Spelling
  - Globalization
  - Keys
  - Network Connections
  - Perspectives
  - Search
  - Security
  - Startup and Shutdown
  - UI Responsiveness Monitorin
  - Web Browser
  - Workspace
- Ant
- C/C++

Text Editors

Some editors may not honor all of these settings.

See 'Colors and Fonts' to configure the font.

Undo history size: 200

Displayed tab width: 4

- [ ] Insert spaces for tabs
- [x] Highlight current line
- [ ] Show print margin

Print margin column: 80

- [x] Show line numbers
- [x] Show range indicator
- [ ] Show whitespace characters (configure visibility)
- [x] Show affordance in hover on how to make it sticky

When mouse moved into hover: Enrich after delay

- [x] Enable drag and drop of text
- [x] Warn before editing a derived file
- [x] Smart caret positioning at line start and end

Appearance color options:

Line number foreground
Current line highlight
Print margin
Find scope

Color: ▬

Apply and Close | Cancel

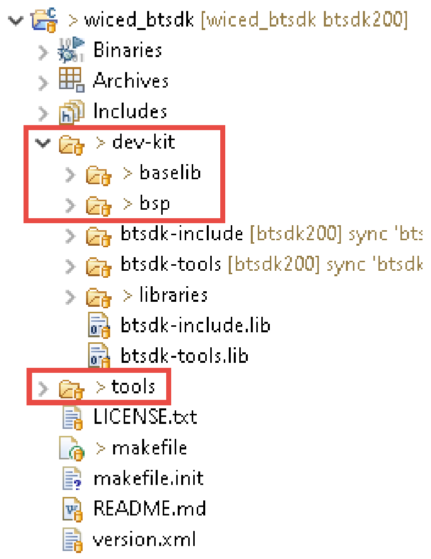
There are also project build settings which we will explore later.

1.4.3 Bluetooth SDK Architecture

The Bluetooth SDK is downloaded from GitHub just like the other project dependencies. In the case of the SDK, it is contained in a separate library project called wiced_btsdk. This project is referenced by all the applications in your workspace. That is, the wiced_btsdk is shared among all applications in a single workspace.

The best practice is to create the wiced_btsdk project first, but in future versions the project will be created automatically for you if it isn't found. It doesn't matter which hardware you select for the wiced_sdk project – one copy will work for all boards – but the name must NOT be changed.

The SDK contains the base libraries for all the supported devices (wiced_btsdk/dev-kit/baselib), board support packages (BSPs) for all the supported boards (wiced_btsdk/dev-kit/bsp), and Bluetooth tools such as ClientControl and BTSpy which you will use in later chapters (wiced_btsdk/tools).


```
> wiced_btsdk [wiced_btsdk btsdk200]
    Binaries
    Archives
    Includes
    > dev-kit
        > baselib
        > bsp
        btsdk-include [btsdk200] sync 'bts
        btsdk-tools [btsdk200] sync 'btsdk
        > libraries
        btsdk-include.lib
        btsdk-tools.lib
    > tools
    LICENSE.txt
    > makefile
    makefile.init
    README.md
    version.xml
```

Since the SDK is shared among all applications in a workspace, any changes to files contained in the SDK will affect all applications in that workspace. This is especially important for the BSP since it contains the device configurator files.

Therefore, if you want to use the device configurator to change settings for an application without affecting other applications, you can do one of three things: (1) override the device configurator files for a single application; (2) create a custom BSP inside the wiced_btsdk; or (3) create a new workspace with a new copy of the wiced_btsdk that you can modify. The first is good when a single application requires a different configuration while 2 is better if you want a single custom configuration for multiple applications. Method 3 is not recommended since it results in modifications to the wiced_btsdk itself, making it more difficult to update versions later.

Override Device Configuration

The steps to override the device configurator files for a single application are:

1. Copy the *wiced_btsdk/dev-kit/bsp/TARGET_<bsp_name>/COMPONENT_bsp_design_modus* folder and its contents to the application's project folder (the folder with the top-level source code). Change the name to something unique that still starts with COMPONENT_. For example:

   COMPONENT_custom_design_modus

2. Disable the configurator files from the BSP and include your custom configurator files by modifying the COMPOENTS in the makefile to change `bsp_design_modus` to whatever name you chose. For example:

   ```
   COMPONENTS += custom_design_modus
   ```

3. Make sure your project is selected and open the device configurator from the Quick Panel. Make the required changes and save/exit the configurator.
   a. It is a good idea to look at the banner in the configurator to verify it is opening the file from the application, not from the SDK.

Custom BSP

The steps to create a custom BSP are:

1. Copy an existing BSP and paste it with a new name in the same folder, which is wiced_btsdk/dev-kit/bsp.
2. Edit the makefile for the application to include the new BSP in the TARGET, SUPPORTED_TARGETS, and TARGET_DEVICE_MAP variables.
3. Make sure your project is selected and open the device configurator from the Quick Panel. Make the required changes and save/exit the configurator.

1.4.4 New Application Creation

When you create a new application from the Eclipse IDE for ModusToolbox, it launches the Project Creator tool. This tool can also be launched independently from the Eclipse IDE. The Project Creator sets up new applications with the required target hardware support, Bluetooth configuration code, middleware libraries, and a "starter" application. Once you close the Project Creator, it imports the newly created project into the Eclipse IDE and sets up for build, program and debug operations. Selecting the "New Application" button in the Quick Panel or the "New ModusToolbox Application" item in *the File->New menu* will launch Project Creator.

When you create a new application, the appropriate files for the application and its libraries are cloned from GitHub (you can use local application files as you will see later, but even in that case the libraries are pulled from a repository). This means that you can always get the latest version of a code example or library (or you can get older versions if you prefer). The downside is that you need an internet connection when creating an application.

Manifests

Manifests are XML files that tell the Project Creator and Library Manager how to discover the list of available boards, example projects, and libraries. There are several manifest files.

- The "super manifest" file contains a list of URLs that software uses to find the board, code example, and middleware manifest files.
- The "app-manifest" file contains a list of all code examples that should be made available to the user.
- The “board-manifest” file contains a list of the boards that should be presented to the user in the new project creation wizards as well as the list of BSP packages that are presented in the Library Manager tool.
- The “middleware-manifest” file contains a list of the available middleware (libraries). Each middleware item can have one or more versions of that middleware available.

To use your own examples, BSPs, and middleware, you need to create manifest files for your content and a super-manifest that points to your manifest files.

Once you have the files created an in a Git repo, place a manifest.loc file in your <user_home>/.modustoolbox directory that specifies the location of your custom super-manifest file (which in turn points to your custom manifest files). For example, a manifest.loc file may have:

```
# This points to the IOT Expert template set
https://github.com/iotexpert/mtb2-iotexpert-manifests/raw/master/iotexpert-super-
manifest.xml
```

Note that you can point to local super-manifest and manifest files by using file:/// with the path instead of https://. For example:

```
file:///C:/MyManfests/my-super-manifest.xml
```

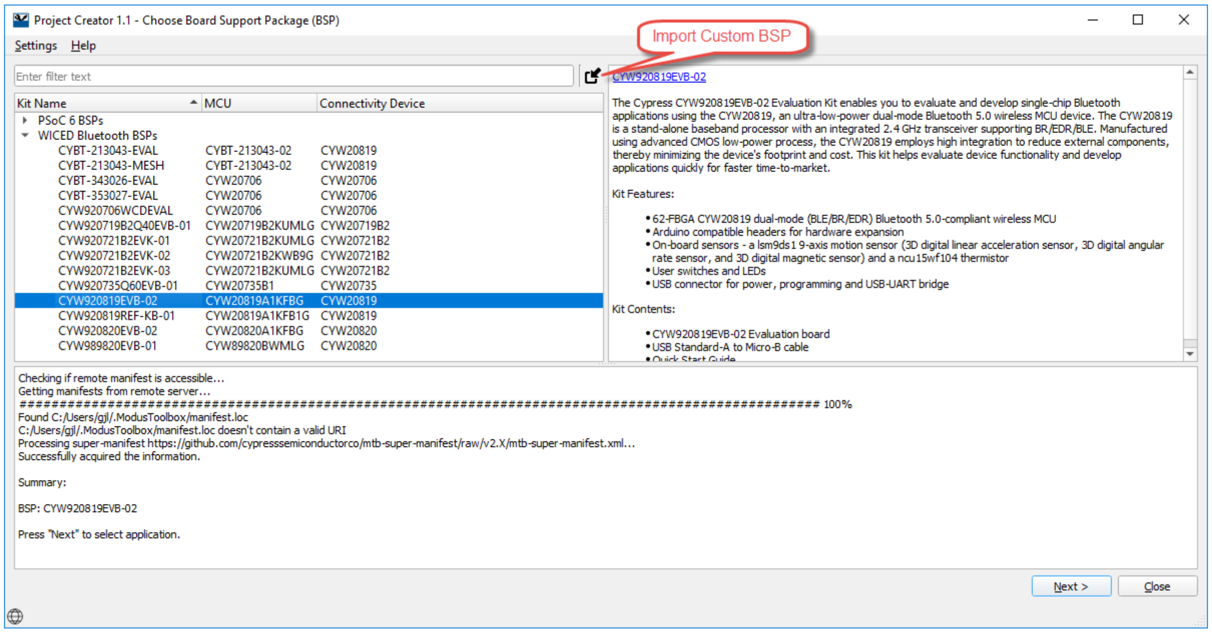
To see examples of the syntax of super-manifest and manifest files, you can look at the Cypress provided files on GitHub:

- Super Manifest: https://github.com/cypresssemiconductorco/mtb-super-manifest
- App Manifest: https://github.com/cypresssemiconductorco/mtb-ce-manifest
- Board Manifest: https://github.com/cypresssemiconductorco/mtb-bsp-manifest
- Middleware Manifest: https://github.com/cypresssemiconductorco/mtb-mw-manifest

Board Support Package

When creating a new application, you must first choose the "Board Support Package (BSP)". Note that it is possible for users to create custom Bluetooth SoC board support packages (BSPs) but that is beyond the scope of this course. In the case of a custom BSP, you can include it in your own Manifest file as described above, or you can use the Import button to specify a local BSP.

The kits used in this course are the CYBT-213043-MESH and CYW920819EVB-02.



Kit Name	MCU	Connectivity Device
PSoC 6 BSPs		
WICED Bluetooth BSPs		
CYBT-213043-EVAL	CYBT-213043-02	CYW20819
CYBT-213043-MESH	CYBT-213043-02	CYW20819
CYBT-343026-EVAL	CYW20706	CYW20706
CYBT-353027-EVAL	CYW20706	CYW20706
CYW920706WCDEVAL	CYW20706	CYW20706
CYW920719B2Q40EVB-01	CYW20719B2KUMLG	CYW20719B2
CYW920721B2EVK-01	CYW20721B2KUMLG	CYW20721B2
CYW920721B2EVK-02	CYW20721B2KWB9G	CYW20721B2
CYW920721B2EVK-03	CYW20721B2KUMLG	CYW20721B2
CYW920735Q60EVB-01	CYW20735B1	CYW20735
CYW920819EVB-02	CYW20819A1KFBG	CYW20819
CYW920819REF-KB-01	CYW20819A1KFB1G	CYW20819
CYW920820EVB-02	CYW20820A1KFBG	CYW20820
CYW989820EVB-01	CYW89820BWMLG	CYW20820

CYW920819EVB-02

The Cypress CYW920819EVB-02 Evaluation Kit enables you to evaluate and develop single-chip Bluetooth applications using the CYW20819, an ultra-low-power dual-mode Bluetooth 5.0 wireless MCU device. The CYW20819 is a stand-alone baseband processor with an integrated 2.4 GHz transceiver supporting BR/EDR/BLE. Manufactured using advanced CMOS low-power process, the CYW20819 employs high integration to reduce external components, thereby minimizing the device's footprint and cost. This kit helps evaluate device functionality and develop applications quickly for faster time-to-market.

Kit Features:

- 62-FBGA CYW20819 dual-mode (BLE/BR/EDR) Bluetooth 5.0-compliant wireless MCU
- Arduino compatible headers for hardware expansion
- On-board sensors - a lsm9ds1 9-axis motion sensor (3D digital linear acceleration sensor, 3D digital angular rate sensor, and 3D digital magnetic sensor) and a ncu15wf104 thermistor
- User switches and LEDs
- USB connector for power, programming and USB-UART bridge

Kit Contents:

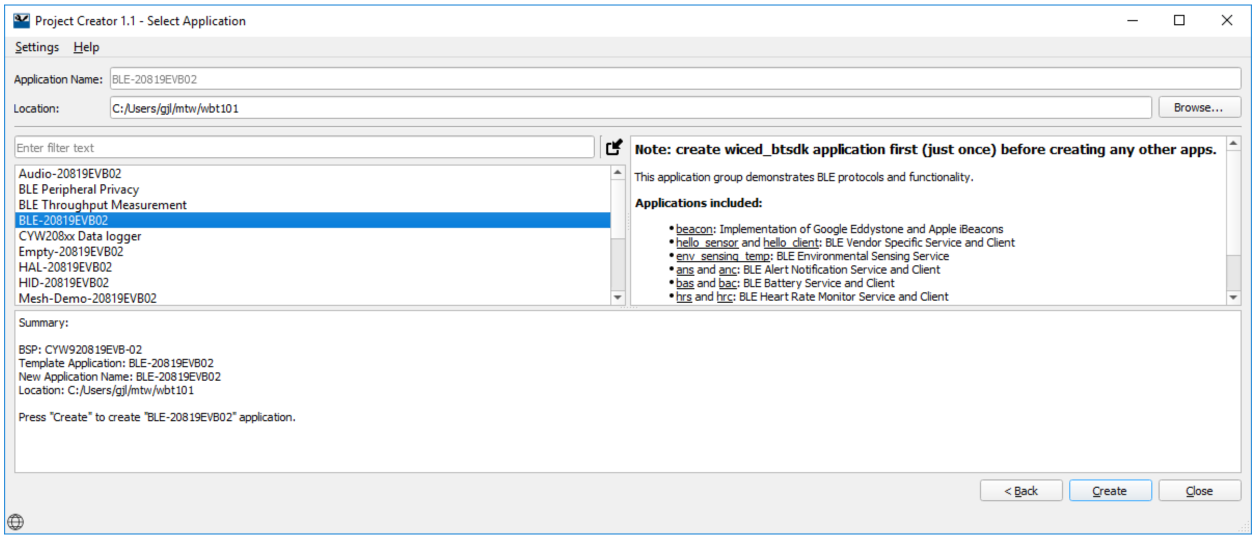
- CYW920819EVB-02 Evaluation board
- USB Standard-A to Micro-B cable
- Quick Start Guide

```
Checking if remote manifest is accessible...
Getting manifests from remote server...
################################################################################################ 100%
Found C:/Users/gjl/.ModusToolbox/manifest.loc
C:/Users/gjl/.ModusToolbox/manifest.loc doesn't contain a valid URI
Processing super-manifest https://github.com/cypresssemiconductorco/mtb-super-manifest/raw/v2.X/mtb-super-manifest.xml...
Successfully acquired the information.

Summary:

BSP: CYW920819EVB-02

Press "Next" to select application.
```

Starter Application

Clicking "Next" lets you choose a starter application from a list of applications that support the selected hardware. You always begin with a starter application. These range in complexity from mostly empty to fully functional demos. For this class we will provide you with template starter applications for many of the exercises. You will access these using the "Import..." button in new application wizard.

You can also choose a location for the application, but if you are using the Eclipse IDE, the location will default to the active workspace which is most likely what you want.



Project Creator 1.1 - Select Application

Settings Help

Application Name: BLE-20819EVB02

Location: C:/Users/gjl/mtw/wbt101

Audio-20819EVB02
BLE Peripheral Privacy
BLE Throughput Measurement
BLE-20819EVB02
CYW208xx Data logger
Empty-20819EVB02
HAL-20819EVB02
HID-20819EVB02
Mesh-Demo-20819EVB02

Note: create wiced_btsdk application first (just once) before creating any other apps.

This application group demonstrates BLE protocols and functionality.

Applications included:

- beacon: Implementation of Google Eddystone and Apple iBeacons
- hello_sensor and hello_client: BLE Vendor Specific Service and Client
- env_sensing_temp: BLE Environmental Sensing Service
- ans and anc: BLE Alert Notification Service and Client
- bas and bac: BLE Battery Service and Client
- hrs and hrc: BLE Heart Rate Monitor Service and Client

Summary:

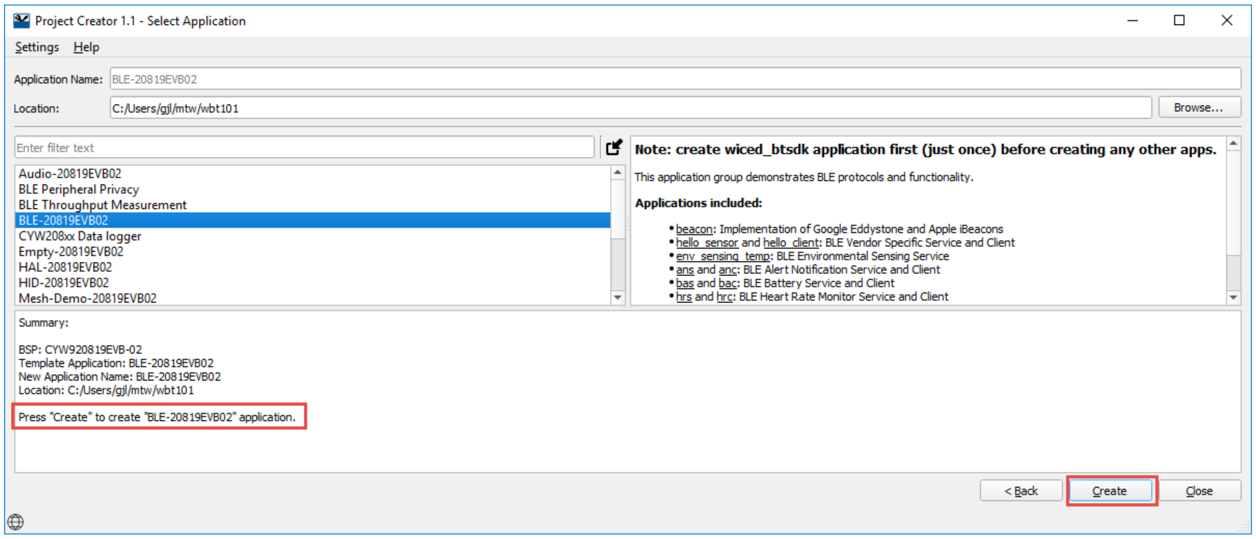
BSP: CYW920819EVB-02
Template Application: BLE-20819EVB02
New Application Name: BLE-20819EVB02
Location: C:/Users/gjl/mtw/wbt101

Press "Create" to create "BLE-20819EVB02" application.

There is no real equivalent to the PSoC Creator empty project, but there is an Empty application that gives you just enough to get started. Unlike the PSoC Creator empty project, using this as a starting point is more of an expert-level approach since it doesn't provide much in the way of code. In most cases, we recommend starting with one of the code examples and modifying it as needed for your application.

Each starter application or set of applications includes a description and a default name, which you can modify if you wish (if you create multiple copies of the same template without changing the name the tool adds an incrementing number to each created project name).

Pressing "Create" will create the application and download its libraries.



Project Creator 1.1 - Select Application

Settings Help

Application Name: BLE-20819EVB02

Location: C:/Users/gjl/mtw/wbt101

Audio-20819EVB02
BLE Peripheral Privacy
BLE Throughput Measurement
BLE-20819EVB02
CYW208xx Data logger
Empty-20819EVB02
HAL-20819EVB02
HID-20819EVB02
Mesh-Demo-20819EVB02

Note: create wiced_btsdk application first (just once) before creating any other apps.

This application group demonstrates BLE protocols and functionality.

Applications included:

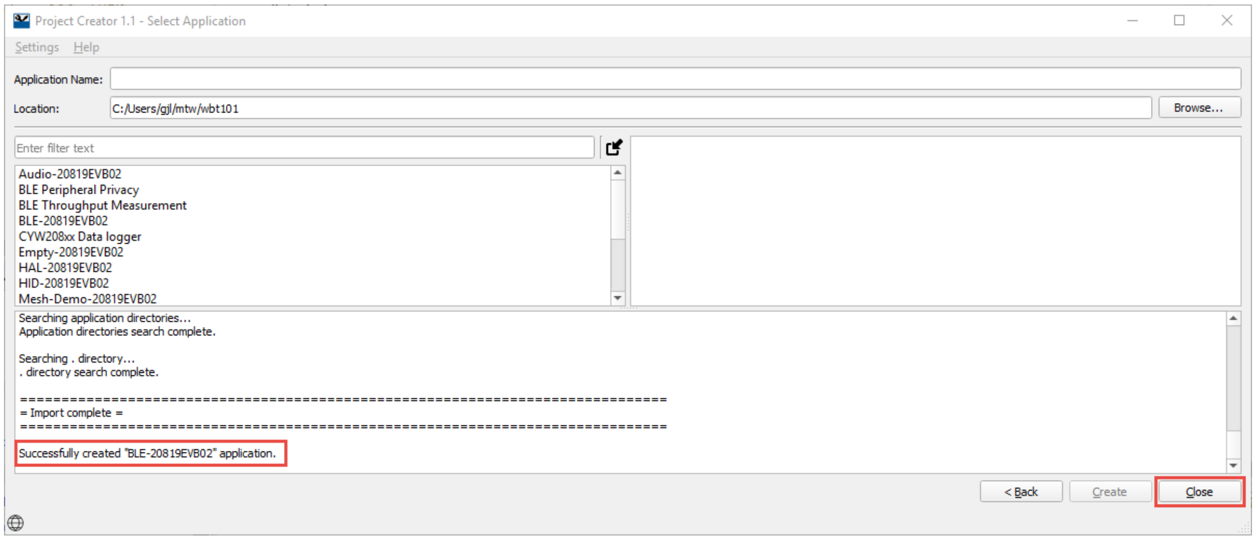
- beacon: Implementation of Google Eddystone and Apple iBeacons
- hello_sensor and hello_client: BLE Vendor Specific Service and Client
- env_sensing_temp: BLE Environmental Sensing Service
- ans and anc: BLE Alert Notification Service and Client
- bas and bac: BLE Battery Service and Client
- hrs and hrc: BLE Heart Rate Monitor Service and Client

Summary:

BSP: CYW920819EVB-02
Template Application: BLE-20819EVB02
New Application Name: BLE-20819EVB02
Location: C:/Users/gjl/mtw/wbt101

Press "Create" to create "BLE-20819EVB02" application.

Once the application creation is complete, you can select and create additional applications if desired. When you have created all the applications that you want, click the "Close" button. This will return control to the Eclipse IDE which will import the new application(s).



Location: C:/Users/gjl/mtw/wbt101

Audio-20819EVB02
BLE Peripheral Privacy
BLE Throughput Measurement
BLE-20819EVB02
CYW208xx Data logger
Empty-20819EVB02
HAL-20819EVB02
HID-20819EVB02
Mesh-Demo-20819EVB02

```
Searching application directories...
Application directories search complete.

Searching . directory...
. directory search complete.

================================================================================
= Import complete =
================================================================================

Successfully created "BLE-20819EVB02" application.
```

Once the application has been created, if you don't see all the expected files, right click on the application name in the Workspace explorer and choose "Refresh".

Application Sets

Most WICED BT starter applications are really a "set" of applications instead of an individual application. For example, the BLE_20819EVB02 starter application contains 12 different BLE applications that are all created when you use that as the starter application. For that reason, the name you provide is used as the prefix of the applications for that set. For example, BLE_20819EVB02 will give you projects called:

BLE_20819EVB02.anc
BLE_20819EVB02.ans
BLE_20819EVB02.bac
BLE_20819EVB02.bas
BLE_20819EVB02.beacon
…

On disk, this is done using hierarchy. For the example described above, the directory structure is:

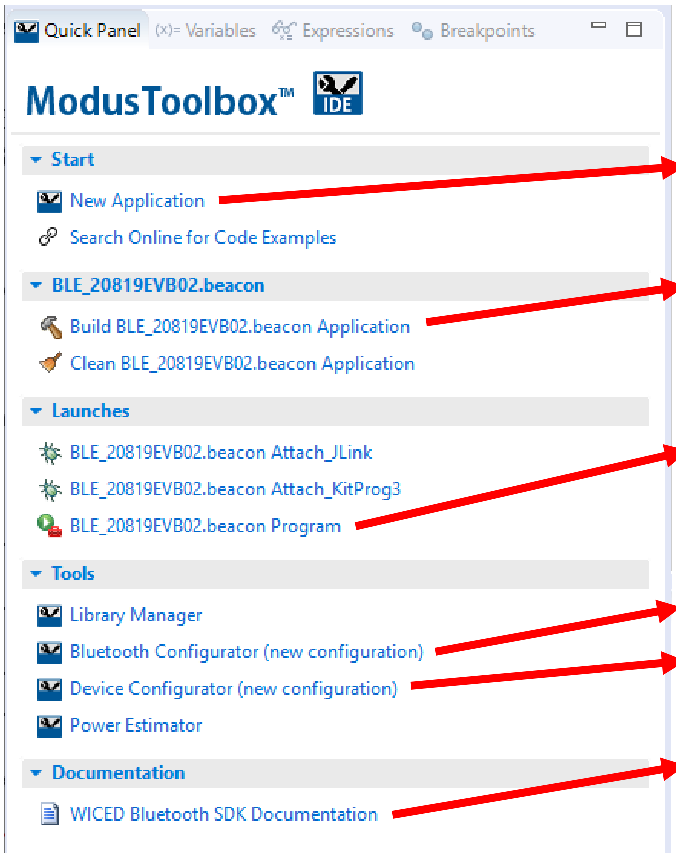
BLE_20819EVB02/ble/anc/<application_files>
BLE_20819EVB02/ble/ans/<application_files>
BLE_20819EVB02/ble/bac/<application_files>
BLE_20819EVB02/ble/bas/<application_files>
BLE_20819EVB02/ble/beacon/<application_files>
…

The lowest level of the hierarchy is where the individual applications are stored. Got to that directory if you want to use the command line interface. Note that the middle level of the directory hierarchy (ble in this example) is not used in the Eclipse application name.

For the exercises in this class, we will use single application templates, so we will not include the extra hierarchy. In that case, the makefile is modified slightly from the starter applications – more on that later.

1.4.5 Quick Panel

Once you have created a new application the Quick Panel is populated with common commands so that you don't have to hunt for them. There are top level commands, application level commands, links to configurators, and links to launches (for program and attaching for debugging). There is also a section with links to documentation.


Quick Panel (x)= Variables Expressions Breakpoints

ModusToolbox™ IDE

Start
- New Application
- Search Online for Code Examples

BLE_20819EVB02.beacon
- Build BLE_20819EVB02.beacon Application
- Clean BLE_20819EVB02.beacon Application

Launches
- BLE_20819EVB02.beacon Attach_JLink
- BLE_20819EVB02.beacon Attach_KitProg3
- BLE_20819EVB02.beacon Program

Tools
- Library Manager
- Bluetooth Configurator (new configuration)
- Device Configurator (new configuration)
- Power Estimator

Documentation
- WICED Bluetooth SDK Documentation

Equivalent Menu Path

File -> New -> New ModusToolbox Application

Select application project in Project Explorer, then Project -> Build

Run->External Tools -> External Tools Configurations -> Program -> <appname>

Double click *.cybt in the application project

Double click design.modus in the BSP

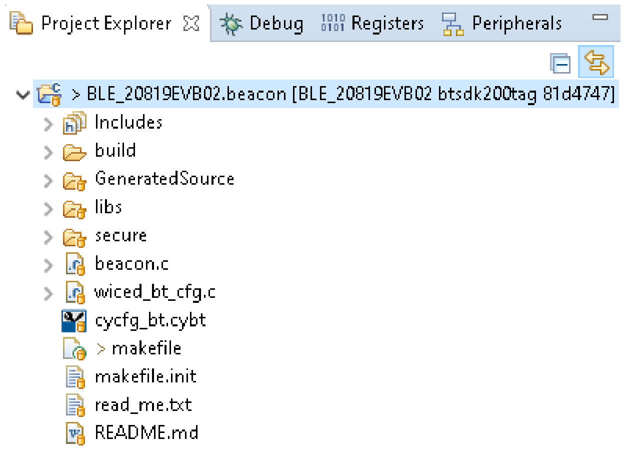
Help -> ModusToolbox General Documentation
Help -> Eclipse IDE for ModusToolbox Documentation

1.4.6 Applications

At this point, we are ready to start developing the application. As mentioned before, a WICED Bluetooth application is the application project and the wiced_btsdk project. We looked briefly at wiced_btsdk earlier, so let's look at an application project now.

Project Explorer

In the project explorer window, you will see your application project and all its associated files.



```
> BLE_20819EVB02.beacon [BLE_20819EVB02 btsdk200tag 81d4747]
    Includes
    build
    GeneratedSource
    libs
    secure
    beacon.c
    wiced_bt_cfg.c
    cycfg_bt.cybt
    > makefile
    makefile.init
    read_me.txt
    README.md
```

The key parts of a project are:

- A folder with the name of the project
- Readme files (read_me.txt and/or README.md)
- Application configurator files (e.g. cycfg_bt.cybt for Bluetooth)
- A GeneratedSource folder with the output files from the application level configurators
- A libs folder
- makefile
- Stack configuration files (e.g. wiced_bt_cfg.c or app_bt_cfg.c)
- Application C source files

Readme

The first file to open in the file editor window will be the readme file included with the application, if there is one. This gives general information about the platform or the application that you started with.

Application Configurator Files

Files for configurators are distinguished by their file extension. For example, any file that ends with *.cybt* is a Bluetooth configurator file.

GeneratedSource Folder

When you save from a configurator, the tool generates firmware in the GeneratedSource folder. The following are the most interesting/useful files (e.g. cycfg_gatt_db.c and cycfg_gatt_db.h for BLE).

Remember that the Device Configuration files (generated from design.modus) go in the BSP so they will NOT be in the GeneratedSource folder in the application unless you override the location.

libs

For Bluetooth starter applications, this folder just contains a file named index.html. It is used to populate the links in the Documents tab in the quick panel.

makefile

The makefile is used in the application creation process – it defines everything that ModusToolbox needs to know to create/build/program the application. This file is interchangeable between the IDE and Command Line Interface so once you create an application, you can go back and forth between an IDE and CLI at will.

Various build settings can be set in the makefile by the user to change the build system behavior. These can be "make" settings or they can be settings that are passed to the compiler. Some examples are:

- Target Device (`TARGET=CYW920819EVB-02`)
- Relative path to the wiced_btsdk (`CY_SHARED_PATH=$(CY_APP_PATH)/../../../wiced_btsdk`)
- Absolute path to the wiced_btsdk (`CY_SHARED_PATH_ABS=$(CURDIR)/../../../wiced_btsdk`)
- Method to generate the Bluetooth Device Address (`BT_DEVICE_ADDRESS?=default`)
- Enable OTA Firmware Upgrade (`OTA_FW_UPGRADE?=1`)
- Compiler Setting to Enabling Debug Trace Printing (`CY_APP_DEFINES+= -DWICED_BT_TRACE_ENABLE`)

You will get to experiment with some of these settings in later chapters.

In particular note the ../../../ in the CY_SHARED_PATH and CY_SHARED_PATH_ABS. This accounts for the hierarchy in the Bluetooth starter applications. In the templates we use in later chapters, the hierarchy will be removed, so those will be changed to just ../

Stack Configuration Files

Many of the templates you will use in this class include app_bt_cfg.c and app_bt_cfg.h, which create static definitions of the stack configuration and buffer pools. You will edit the stack configuration, for example, to optimize the scanning and advertising parameters. The buffer pools determine the availability of various sizes of memory blocks for the stack, and you might edit those to optimize performance and RAM usage.

Note: The actual file names may vary in some code examples, but the definitions of the wiced_bt_cfg_settings struct and wiced_bt_cfg_buf_pools array are required.

Application C file

All starter applications include at least one C source file that starts up the application (from the application_start() function) and then implements other application functionality. The actual name of this file varies according to the starter application used to create the application. It is usually the same name as the template but there are exceptions to that rule. For example, in the templates provided for this class this top-level file is always called app.c.

Note: MESH examples are implemented with a library called mesh_app_lib. The library includes the application_start() function in mesh_application.c inside the library. This is because the application itself is very regimented, and the developer does not really "own" the way devices interact. The user part of the application is restricted to activity supported by the device's capabilities, such as dimming the light with a PWM or controlling a door lock solenoid.

The application C file begins with various #include lines depending on the resources used in your application. These header files can be found in the SDK under wiced_btsdk/dev-kit/baselib/20819A1/include. A few examples are shown below. The first 4 are usually required in any project. The "hal" includes are only needed if the specific peripheral are used in the project, e.g. wiced_hal_i2c.h is required if you are using I2C.

```
#include "wiced.h"                   // Basic formats like stdint, wiced_result_t, WICED_FALSE, WICED_TRUE
#include "wiced_platform.h"          // Platform file for the kit
#include "sparcommon.h"              // Common application definitions
#include "wiced_bt_stack.h"          // Bluetooth Stack
#include "wiced_bt_dev.h"            // Bluetooth Management
#include "wiced_bt_ble.h"            // BLE
#include "wiced_bt_gatt.h"           // BLE GATT database
#include "wiced_bt_uuid.h"           // BLE standard UUIDs
#include "wiced_rtos.h"              // RTOS
#include "wiced_bt_app_common.h"     // Miscellaneous helper functions including wiced_bt_app_init
#include "wiced_transport.h"         // HCI UART drivers
#include "wiced_bt_trace.h"          // Trace message utilities
#include "wiced_timer.h"             // Built-in timer drivers
#include "wiced_hal_i2c.h"           // I2C drivers
#include "wiced_hal_adc.h"           // ADC drivers
#include "wiced_hal_pwm.h"           // PWM drivers
#include "wiced_hal_puart.h"         // PUART drivers
#include "wiced_rtos.h"              // RTOS functions
#include "wiced_hal_nvram.h"         // NVRAM drivers
#include "wiced_hal_wdog.h"          // Watchdog
#include "wiced_spar_utils.h"        // Required for stdio functions such as snprint and snpritf
#include <stdio.h>                   // Stdio C functions such as snprint and snpritf
```

After the includes list you will find the application_start() function, which is the main entry point into the application. That function typically does a minimal amount of initialization, starts the Bluetooth stack and registers a stack callback function by calling *wiced_bt_stack_init()*. Note that the configuration parameters from app_bt_cfg.c are provided to the stack here. The callback function is called by the stack whenever it has an event that the user's application might need to know about. It typically controls the rest of the application based on Bluetooth events.

Most application initialization is done once the Bluetooth stack has been enabled. That event is called BTM_ENABLED_EVT in the callback function. The full list of events from the Bluetooth stack can be found in the file wiced_btsdk/dev-kit/baselib/20819A1/include/wiced_bt_dev.h.

A minimal C file for an application will look something like this:

```
#include "sparcommon.h"
#include "wiced_platform.h"
#include "wiced_bt_dev.h"
```

```
#include "wiced_bt_stack.h"
#include "app_bt_cfg.h"

wiced_bt_dev_status_t  app_bt_management_callback( wiced_bt_management_evt_t event,
wiced_bt_management_evt_data_t *p_event_data );

void application_start(void)
{
    wiced_bt_stack_init( app_bt_management_callback, &wiced_bt_cfg_settings,
                         wiced_bt_cfg_buf_pools );
}

wiced_result_t app_bt_management_callback( wiced_bt_management_evt_t event,
                                           wiced_bt_management_evt_data_t *p_event_data )
{
    wiced_result_t status = WICED_BT_SUCCESS;

    switch( event )
    {
    case BTM_ENABLED_EVT:            // Bluetooth Controller and Host Stack Enabled

        if( WICED_BT_SUCCESS == p_event_data->enabled.status )
        {
           /* Initialize and start your application here once the BT stack is running */
        }
        break;

    default:
        break;
    }

    return status;
}
```

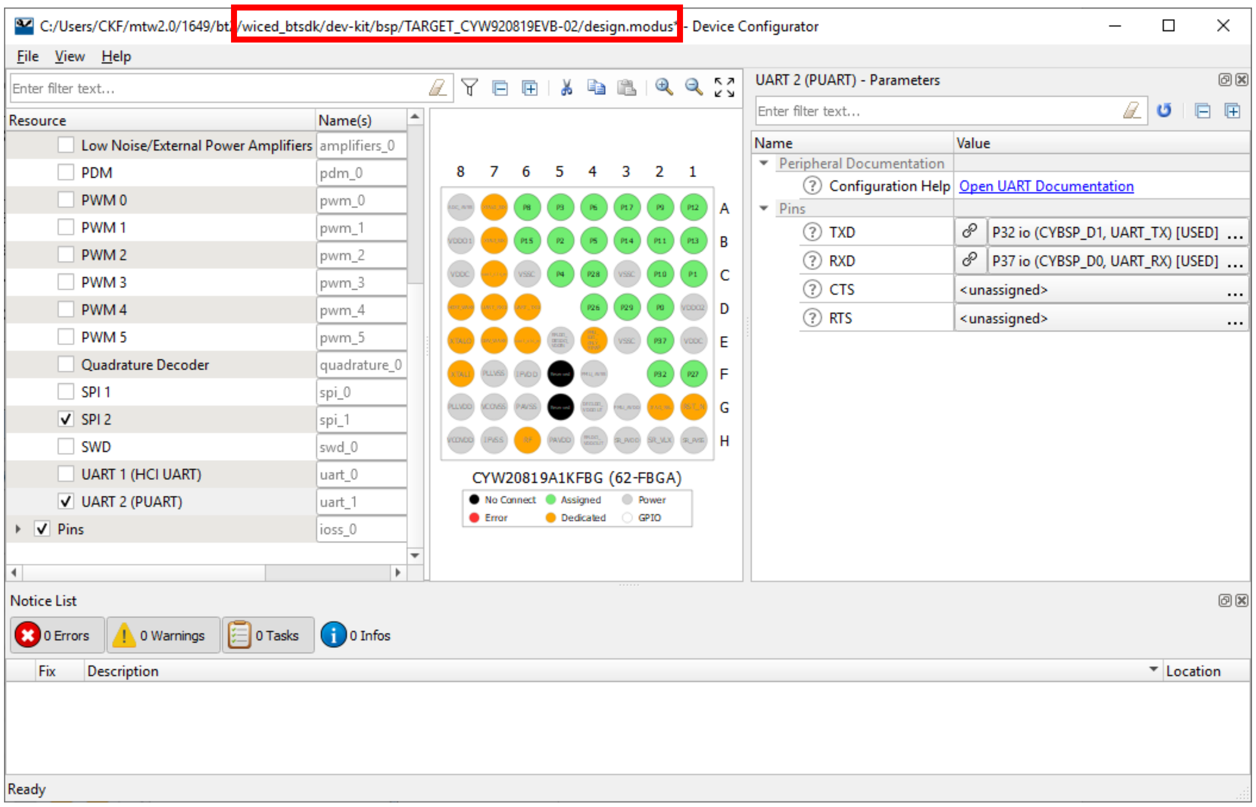
1.4.7 Device Configuration

The design.modus file is the database of configuration choices for the device. It is the source document for the Device Configurator tool which presents a list of pins and peripherals on the device that you can set up. As mentioned earlier, the design.modus file is shared by all applications in a workspace that use a given BSP. Keep that in mind if you chose to edit the design.modus file.

As described earlier, you can override the design.modus for a single application. You will use this method in the exercises in the next chapter.

To launch the Device Configurator click on the link in the Quick Panel or double-click on the design.modus file either in the BSP (wiced_btsdk/dev-kit/bsp/<BSP Name>) or in your application if you are using the override method. As you can see there are sections for Peripherals and Pins on the left. When you enable a peripheral or pin, the upper right-hand panel allows you to select configuration options and to open the documentation for that element. In some cases, you can launch a secondary configurator from that window (Bluetooth is one of those cases).

It is a good idea to look at the path at the top of the configurator window to verify you are editing the file from the expected location (either in the BSP or in the application).



C:/Users/CKF/mtw2.0/1649/bt/wiced_btsdk/dev-kit/bsp/TARGET_CYW920819EVB-02/design.modus* - Device Configurator

Resource	Name(s)
☐ Low Noise/External Power Amplifiers	amplifiers_0
☐ PDM	pdm_0
☐ PWM 0	pwm_0
☐ PWM 1	pwm_1
☐ PWM 2	pwm_2
☐ PWM 3	pwm_3
☐ PWM 4	pwm_4
☐ PWM 5	pwm_5
☐ Quadrature Decoder	quadrature_0
☐ SPI 1	spi_0
✓ SPI 2	spi_1
☐ SWD	swd_0
☐ UART 1 (HCI UART)	uart_0
✓ UART 2 (PUART)	uart_1
✓ Pins	ioss_0

UART 2 (PUART) - Parameters

Name	Value
Peripheral Documentation	
Configuration Help	Open UART Documentation
Pins	
TXD	P32 io (CYBSP_D1, UART_TX) [USED]
RXD	P37 io (CYBSP_D0, UART_RX) [USED]
CTS	<unassigned>
RTS	<unassigned>

Notice List

0 Errors 0 Warnings 0 Tasks 0 Infos

Fix	Description	Location

Ready

Once you save the configurator information (*File->Save*) it creates/updates the Generated Source files in the project BSP or the local copy if you are using the override method.

1.4.8 Library Manager

There is a Library Manager that is intended to add/remove individual libraries from an application. However, for BT applications, the libraries are all currently shared (from wiced_btsdk) so the library manager isn't particularly useful (at least not yet).

Instead, the way to add a new library to an application is as follows (you will get to try this out in later chapters):

1. Add the appropriate include in your source code.
2. Add the library name to the COMPONENTS variable in the makefile.
3. Add the path to the library to the SEARCH_LIBS_AND_INCLUDES variable in the makefile.

For example, to add the over-the-air (OTA) firmware upgrade library, you would add:

1. source code:

```
#include "wiced_bt_ota_firmware_upgrade.h"
```

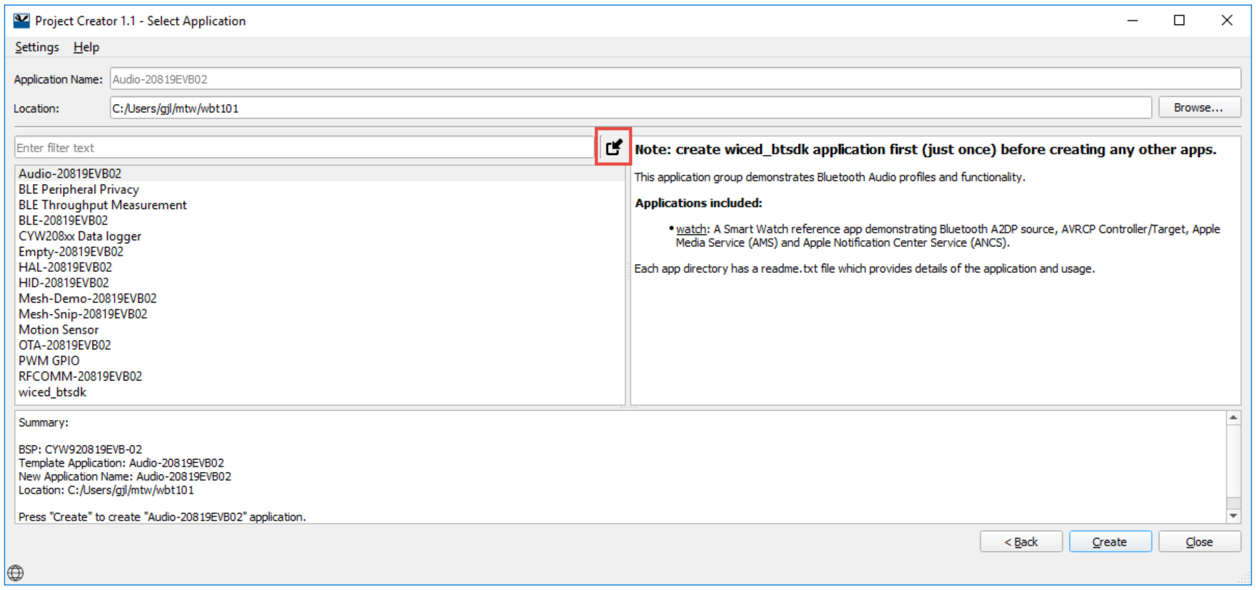
2. makefile:

```
COMPONENTS+=fw_upgrade_lib
.
.
.
SEARCH_LIBS_AND_INCLUDES+=$(CY_SHARED_PATH)/dev-kit/libraries/btsdk-ota
```

Note: for future reference, the required makefile edits are also shown at the bottom of the README.md file in the wiced_btsdk project and the BT starter applications.

1.4.9 Renaming / Copying Projects

You can easily create a new application from an existing one using the "Import..." button in the new application wizard after selecting the target hardware.



Application Name: Audio-20819EVB02

Location: C:/Users/gjl/mtw/wbt101

Enter filter text

Audio-20819EVB02
BLE Peripheral Privacy
BLE Throughput Measurement
BLE-20819EVB02
CYW208xx Data logger
Empty-20819EVB02
HAL-20819EVB02
HID-20819EVB02
Mesh-Demo-20819EVB02
Mesh-Snip-20819EVB02
Motion Sensor
OTA-20819EVB02
PWM GPIO
RFCOMM-20819EVB02
wiced_btsdk

Note: create wiced_btsdk application first (just once) before creating any other apps.

This application group demonstrates Bluetooth Audio profiles and functionality.

Applications included:

- watch: A Smart Watch reference app demonstrating Bluetooth A2DP source, AVRCP Controller/Target, Apple Media Service (AMS) and Apple Notification Center Service (ANCS).

Each app directory has a readme.txt file which provides details of the application and usage.

Summary:

BSP: CYW920819EVB-02
Template Application: Audio-20819EVB02
New Application Name: Audio-20819EVB02
Location: C:/Users/gjl/mtw/wbt101

Press "Create" to create "Audio-20819EVB02" application.

After clicking the button, just select the application folder to import (the folder containing the makefile).

<u>If you do this for a single application that has additional hierarchy (such as most of the starter application code examples) you need to modify the path in the makefile after importing since the import will not maintain the additional hierarchy</u>. You will see an error about the wiced_btsdk not being available. To fix this, the CY_SHARED_PATH and CY_SHARED_PATH_ABS will need to be modified to the following:

```
CY_SHARED_PATH=$(CY_APP_PATH)/../wiced_btsdk

CY_SHARED_PATH_ABS=$(CURDIR)/../wiced_btsdk
```

After updating the makefile, click the *Generate Launches for <application>* in the Quick Panel to get the Launches populated.

<u>Note that if you provide the top-level folder of an entire set of applications, the hierarchy will be maintained so everything will work as-is and it will import all of the applications in the set at once</u>.

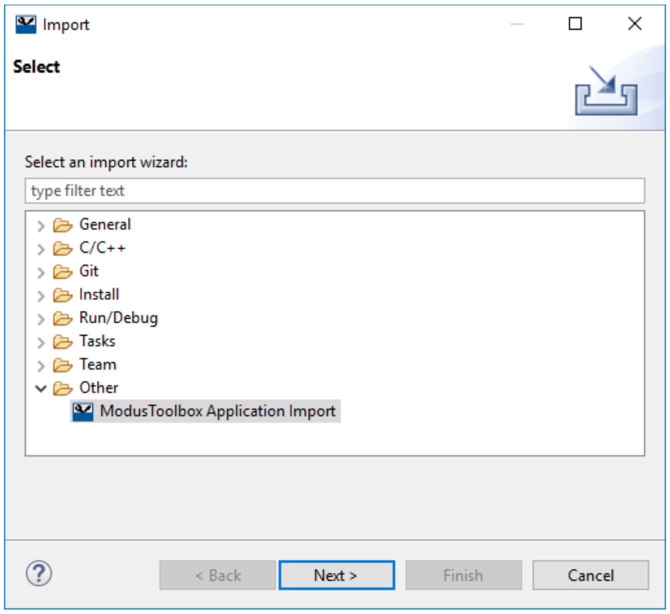
Alternately, you can rename a single project from inside the Eclipse IDE by right clicking on the project in the project explorer and selecting "Rename" (don't rename wiced_btsdk though – that will break things). This renames the project as it shows up in Eclipse, but it does NOT change the name of the folders in the filesystem. The Eclipse project name can be found in the *.project* file inside the project folder.

You can also copy/paste but you must keep the relative path to the wiced_btsdk the same (or else change the path in the makefile to match the new relative path). This is difficult to get just right inside the IDE so it is easier to copy/paste from a file window (Windows explorer, etc.) and then update the project name in the *.project* file. Once you do that, you can import the project into Eclipse using *File > Import > General > Existing Projects into Workspace*.

After either a rename or a copy/paste, the Launches section for the new project in the Quick Panel will be empty. The launches can be recreated by using *Generate Launches for <application>* in the Quick Panel.

1.4.10 Importing Applications

If you have an application that you want to import into the Eclipse IDE (for example from a colleague, or an application created on the command line that you decide you want to open in the IDE), you can use **File > Import… > Other > ModusToolbox Application Import** from the IDE. Note that this imports the application in-place - it does NOT copy it into your workspace. If you want the resulting application to be in your workspace, make sure you put it in the workspace folder before importing it.



After clicking **Next >**, browse to the location of the application's directory, select it, and click **Finish**.

Remember that the path to the wiced_btsdk project must be specified in the makefile so if the relative hierarchy of the newly imported application is not the same as it was previously, you will need to edit the CY_SHARED_PATH and CY_SHARED_PATH_ABS variables in the makefile.

1.4.11 Launch Configs

A few important notes regarding the launch configurations inside Eclipse:

There is a directory inside the application called *.mtbLaunchConfigs*. This is where the launch configurations are located for an application. There are cases where you will end up with old launch configurations in that directory which can confuse Eclipse. You may see no launch configs, the wrong launch configs, or multiple sets of launch configs.

In case you see that behavior, it is safe to blow away the .mtbLanuncConfigs directory at any time and then just click on "Generate Launches for <project name>" in the Quick Panel.

Some of the cases where you may end up with stale or multiple sets of launch configs:

1. If you rename an application directory before importing.
2. If you rename an application inside of Eclipse (i.e. Right-click > Rename)
3. If you create a new ModusToolbox application and point to an existing application as the starter template.

1.5 Command Line Interface (CLI)

In addition to the Eclipse IDE for ModusToolbox, there is a command line interface that can be used for all operations including downloading applications, running configurators, building, programming, and even debugging.

The make/build system works the same in both environments so once you have an application setup in the IDE, you can go back and forth between the command line and IDE at will.

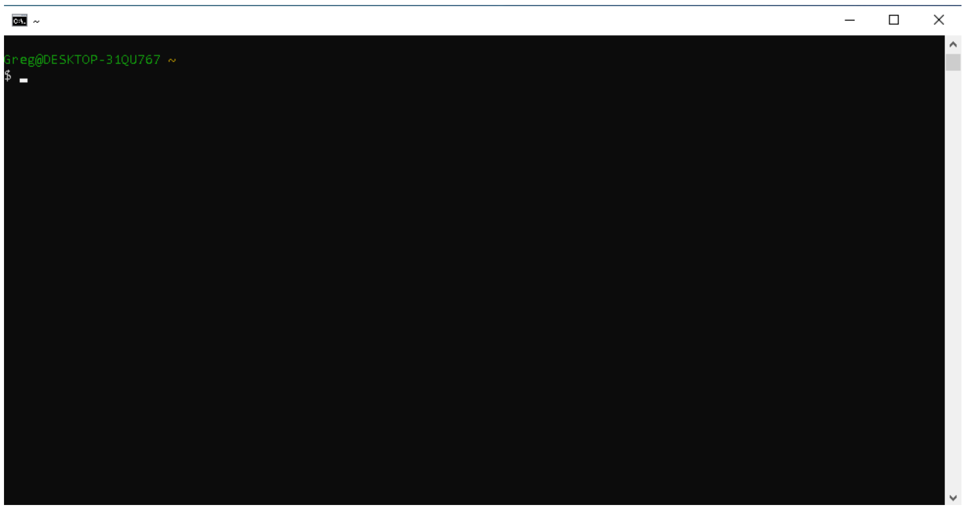
If you start from the IDE, the application is automatically capable of command line operations. If you start from the command line, you can use the *ModusToolbox Application Import* command shown earlier. In either case, once it is set up, you can use both methods interchangeably depending on what you want to do.

Note that you must NOT have Cygwin in your path. If you do, the ModusToolbox CLI may not work properly.

1.5.1 Modus Shell

Modus Shell is a Cygwin environment that is set up for ModusToolbox. If you already have Cygwin installed and it has make and git, you can use Cygwin directly. If not, Modus Shell is a very convenient option.

To run Modus Shell, go to the tools directory (<install>/ModusToolbox/tools_<version>/modus-shell) and double click on Cygwin.bat. This will launch the shell. It is Linux based, so you can use any Linux commands.



```
Greg@DESKTOP-31QU767 ~
$ 
```

1.5.2 Downloading an Application from a Repository

As discussed previously, Project Creator is used when creating a new application. It is launched from the Eclipse IDE, but for command line operations, you can run it stand-alone. In fact, there is both a GUI version of this tool (project-creator.exe) and a command-line version (project-creator-cli.exe). The GUI version is the same application used in the Eclipse IDE and is a very easy way to setup an application which can then be used in the command line interface. The command line version requires a few arguments, but if you run it with no arguments it will print help information.

Both tools can be found in the ModusToolbox installation at:
<install_folder>/ModusToolbox/tools_<version>/project-creator

If you want to setup an application manually without using one of the project creator tools, you can use Git and make commands to download an application and its dependencies from their repositories (typically on GitHub). You can find all of the available code examples at: https://github.com/cypresssemiconductorco/Code-Examples-BT-SDK-for-ModusToolbox.

For example, you could use these commands from Modus Shell to download and build the Empty Bluetooth code example (the make commands will be explained in the next section):

Make a Directory to Hold the Applications (i.e. a Workspace)

```
mkdir my_workspace

cd my_workspace
```

Download the BTSDK (only need to do this once per workspace)

```
git clone https://github.com/cypresssemiconductorco/wiced_btsdk

cd wiced_btsdk

make getlibs

cd ..
```

Download the application (or application group)

```
git clone https://github.com/cypresssemiconductorco/mtb-examples-CYW920819EVB-02-btsdk-empty

cd Empty_20819EVB02/template/empty_wiced_bt

make getlibs
```

Build the Application

```
make build
```

1.5.3 Modus CLI Commands

To run ModusToolbox CLI commands, you must first change to the directory with your application's project (the directory with the makefile). For example:

cd C:/Users/Greg/mtw/wbt101/BLE_20819EVB02/ble/beacon

Once in the application's project folder, you can run make with various arguments such as:

make help
Print top level information and options of the make system.

make *help CY_HELP=build*
Print help information about the build make target.

make build
Build the application.

make build VERBOSE=true
Build the application with verbose output messages.

make program
Build the application and then program it to the target device.

make qprogram
Program the application using the existing build output files.

make clean
Clean the application.

make getlibs
Search for library files in the project (*.lib) and download the associated libraries.

make config_bt
Run the Bluetooth configurator. If an existing .cybt file exists it will open it. If not, it will allow you to create a new one once the tool launches.

make config
Run the device configurator. This will open the configurator file from the BSP for your target device. Remember that the BSP is in a location that may be shared between applications so be careful when editing this file.

There are many other options and variables available. Explore the make help output to find out what else is possible.

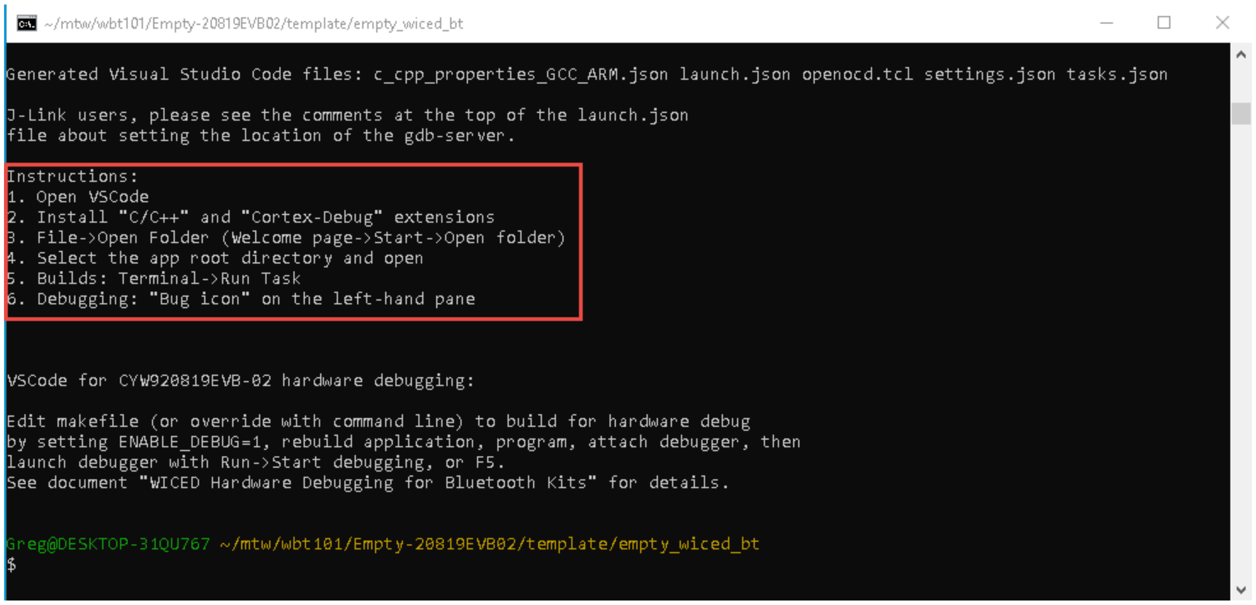
1.6 Visual Studio Code (VS Code)

One alternative to using the Eclipse-based IDE is a code editor program called VS Code. This tool is quickly becoming a favorite editor for developers. The ModusToolbox command line knows how to make all the files required for VS Code to edit, build, and program a ModusToolbox program.

To use Visual Studio Code, first create an application using one of the methods covered previously. Then, from the command line enter the following command inside the application's folder:

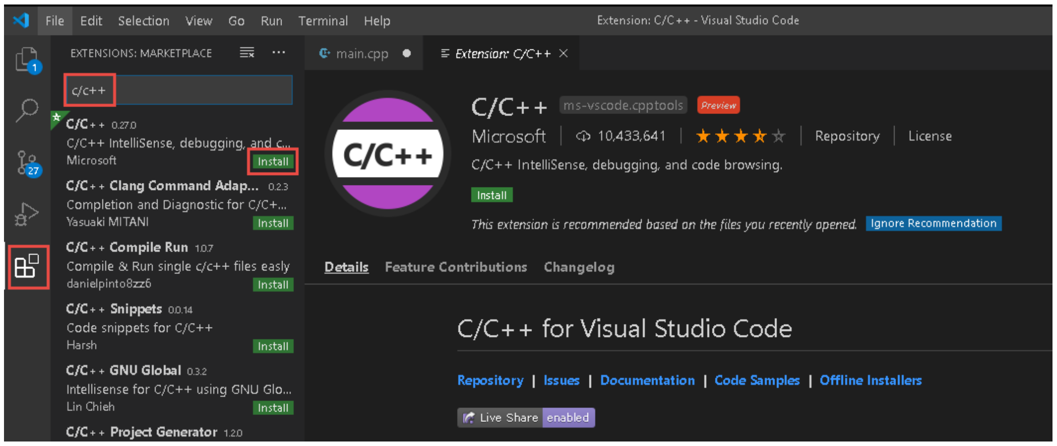
```
make vscode
```

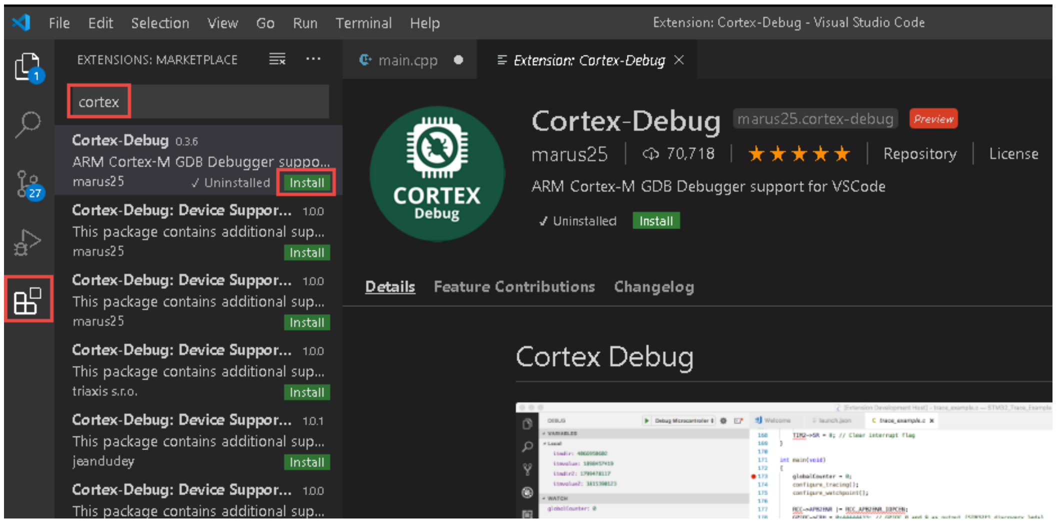
The output will look like the following. The message at the bottom of the command window tells you the next steps to take. These steps are explained in detail below.



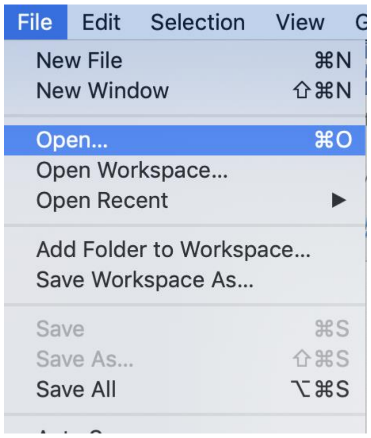
```
~/mtw/wbt101/Empty-20819EVB02/template/empty_wiced_bt

Generated Visual Studio Code files: c_cpp_properties_GCC_ARM.json launch.json openocd.tcl settings.json tasks.json

J-Link users, please see the comments at the top of the launch.json
file about setting the location of the gdb-server.

Instructions:
1. Open VSCode
2. Install "C/C++" and "Cortex-Debug" extensions
3. File->Open Folder (Welcome page->Start->Open folder)
4. Select the app root directory and open
5. Builds: Terminal->Run Task
6. Debugging: "Bug icon" on the left-hand pane


VSCode for CYW920819EVB-02 hardware debugging:

Edit makefile (or override with command line) to build for hardware debug
by setting ENABLE_DEBUG=1, rebuild application, program, attach debugger, then
launch debugger with Run->Start debugging, or F5.
See document "WICED Hardware Debugging for Bluetooth Kits" for details.


Greg@DESKTOP-31QU767 ~/mtw/wbt101/Empty-20819EVB02/template/empty_wiced_bt
$
```

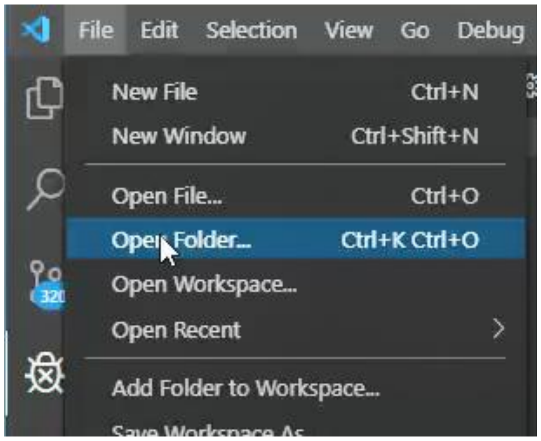
When you start VS Code, install the C/C++ and Cortex-Debug extensions if you don't already have them.


File Edit Selection View Go Run Terminal Help
Extension: C/C++ - Visual Studio Code
EXTENSIONS: MARKETPLACE
c/c++
main.cpp
Extension: C/C++
C/C++ 0.27.0
C/C++ IntelliSense, debugging, and c...
Microsoft
Install
C/C++ Clang Command Adap... 0.2.3
Completion and Diagnostic for C/C+...
Yasuaki MITANI
Install
C/C++ Compile Run 1.0.7
Compile & Run single c/c++ files easly
danielpinto8zz6
Install
C/C++ Snippets 0.0.14
Code snippets for C/C++
Harsh
Install
C/C++ GNU Global 0.3.2
Intellisense for C/C++ using GNU Glo...
Lin Chieh
Install
C/C++ Project Generator 1.2.0
C/C++
ms-vscode.cpptools
Preview
Microsoft
10,433,641
Repository
License
C/C++ IntelliSense, debugging, and code browsing.
Install
This extension is recommended based on the files you recently opened.
Ignore Recommendation
Details
Feature Contributions
Changelog
C/C++ for Visual Studio Code
Repository | Issues | Documentation | Code Samples | Offline Installers
Live Share enabled

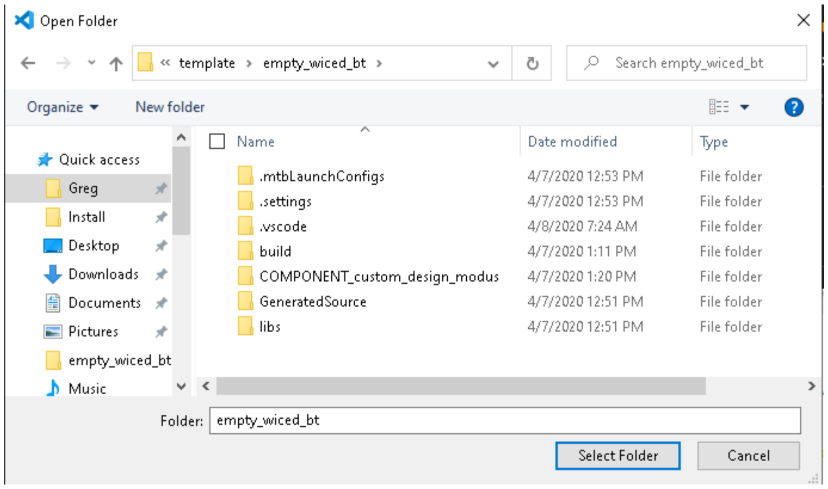

File Edit Selection View Go Run Terminal Help
Extension: Cortex-Debug - Visual Studio Code
EXTENSIONS: MARKETPLACE
cortex
main.cpp
Extension: Cortex-Debug
Cortex-Debug 0.3.6
ARM Cortex-M GDB Debugger suppo...
marus25
✓ Uninstalled
Install
Cortex-Debug: Device Suppor... 1.0.0
This package contains additional sup...
marus25
Install
Cortex-Debug: Device Suppor... 1.0.0
This package contains additional sup...
marus25
Install
Cortex-Debug: Device Suppor... 1.0.0
This package contains additional sup...
triaxis s.r.o.
Install
Cortex-Debug: Device Suppor... 1.0.1
This package contains additional sup...
jeandudey
Install
Cortex-Debug: Device Suppor... 1.0.0
This package contains additional sup...
CORTEX Debug
Cortex-Debug
marus25.cortex-debug
Preview
marus25
70,718
Repository
License
ARM Cortex-M GDB Debugger support for VSCode
✓ Uninstalled
Install
Details
Feature Contributions
Changelog
Cortex Debug

Open the project in Visual Studio Code using **File > Open** (on MacOS) or **File > Open Folder** (on Windows).



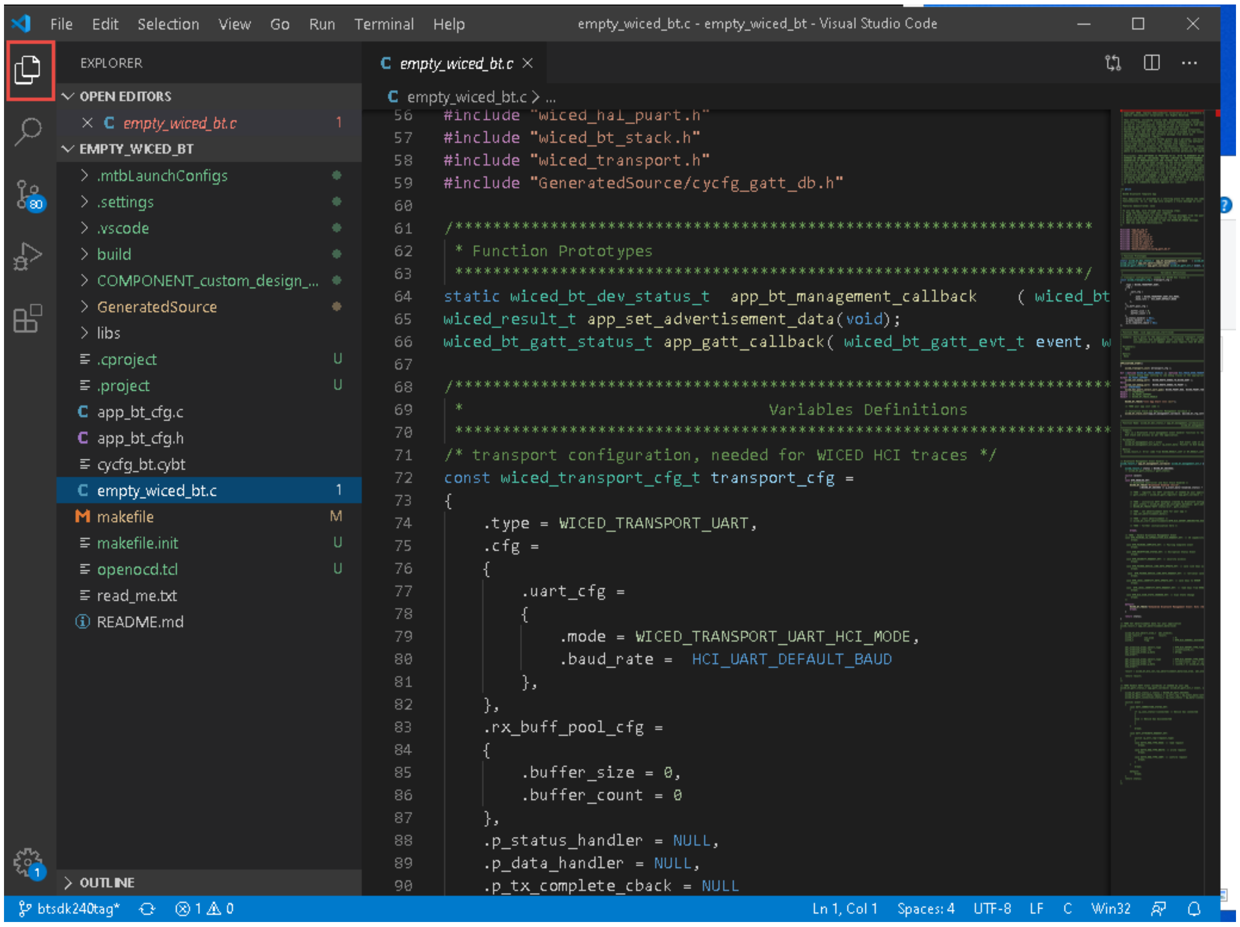


Then, navigate to the directory where your project resides and click **Open** (MacOS) or **Select Folder** (Windows).



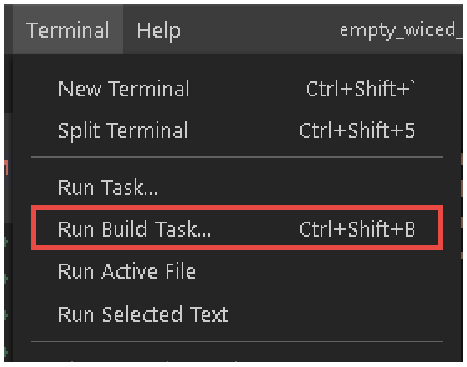
Name	Date modified	Type
.mtbLaunchConfigs	4/7/2020 12:53 PM	File folder
.settings	4/7/2020 12:53 PM	File folder
.vscode	4/8/2020 7:24 AM	File folder
build	4/7/2020 1:11 PM	File folder
COMPONENT_custom_design_modus	4/7/2020 1:20 PM	File folder
GeneratedSource	4/7/2020 12:51 PM	File folder
libs	4/7/2020 12:51 PM	File folder

Your windows should look something like this (make sure you have the Explorer tab selected on the left side). You can open the files by clicking on them in the Explorer window.

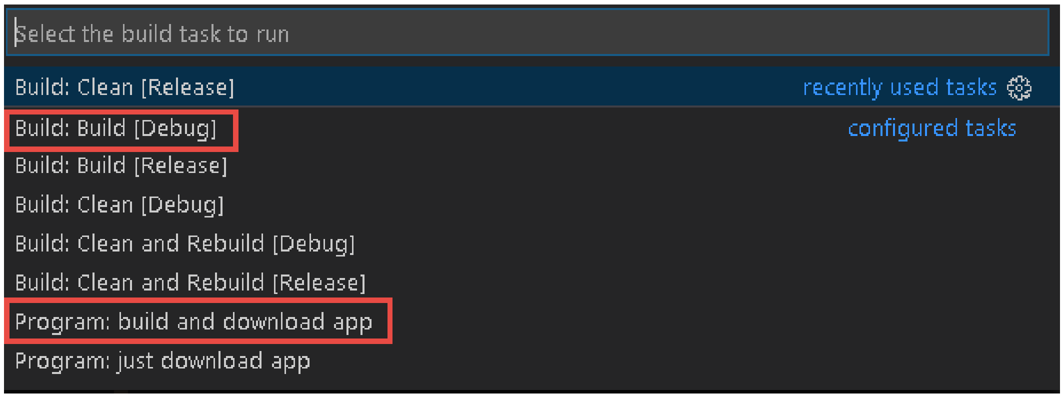


```
#include "wiced_hal_puart.h"
#include "wiced_bt_stack.h"
#include "wiced_transport.h"
#include "GeneratedSource/cycfg_gatt_db.h"

/*******************************************************************
 * Function Prototypes
 *******************************************************************/
static wiced_bt_dev_status_t  app_bt_management_callback    ( wiced_bt
wiced_result_t app_set_advertisement_data(void);
wiced_bt_gatt_status_t app_gatt_callback( wiced_bt_gatt_evt_t event, w

/*******************************************************************
 *                      Variables Definitions
 *******************************************************************
/* transport configuration, needed for WICED HCI traces */
const wiced_transport_cfg_t transport_cfg =
{
    .type = WICED_TRANSPORT_UART,
    .cfg =
    {
        .uart_cfg =
        {
            .mode = WICED_TRANSPORT_UART_HCI_MODE,
            .baud_rate =  HCI_UART_DEFAULT_BAUD
        },
    },
    .rx_buff_pool_cfg =
    {
        .buffer_size = 0,
        .buffer_count = 0
    },
    .p_status_handler = NULL,
    .p_data_handler = NULL,
    .p_tx_complete_cback = NULL
```

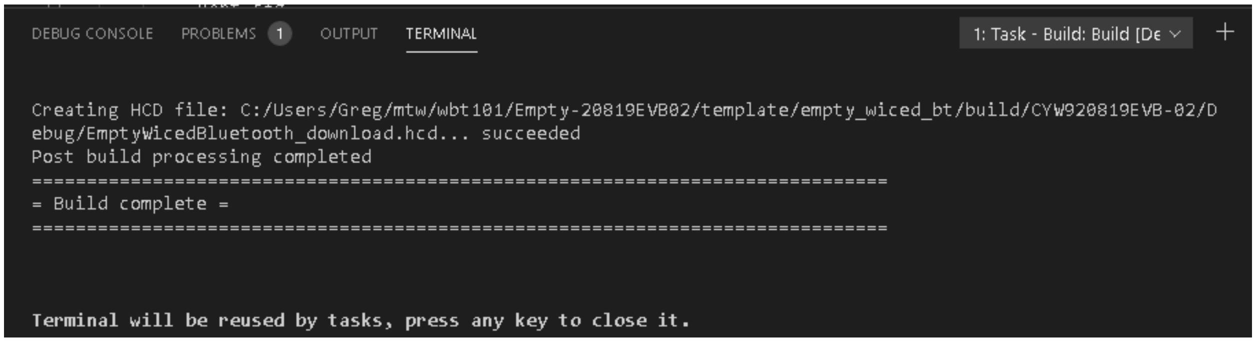
In order to build your project, select **Terminal > Run Build Task…**


Terminal
Help
empty_wiced_
New Terminal Ctrl+Shift+`
Split Terminal Ctrl+Shift+5
Run Task...
Run Build Task... Ctrl+Shift+B
Run Active File
Run Selected Text

Then, select the appropriate task such as **Build: Build Debug** or **Program: build and download app**.


Select the build task to run
Build: Clean [Release]
recently used tasks
Build: Build [Debug]
configured tasks
Build: Build [Release]
Build: Clean [Debug]
Build: Clean and Rebuild [Debug]
Build: Clean and Rebuild [Release]
Program: build and download app
Program: just download app

You should see the normal compiler output in the console at the bottom of VS Code:

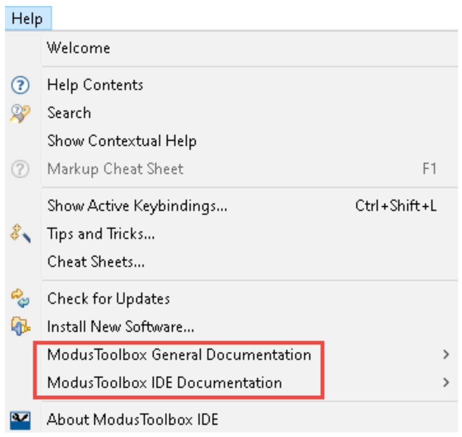


```
Creating HCD file: C:/Users/Greg/mtw/wbt101/Empty-20819EVB02/template/empty_wiced_bt/build/CYW920819EVB-02/D
ebug/EmptyWicedBluetooth_download.hcd... succeeded
Post build processing completed
================================================================================
= Build complete =
================================================================================


Terminal will be reused by tasks, press any key to close it.
```

1.7 Tour of Documentation

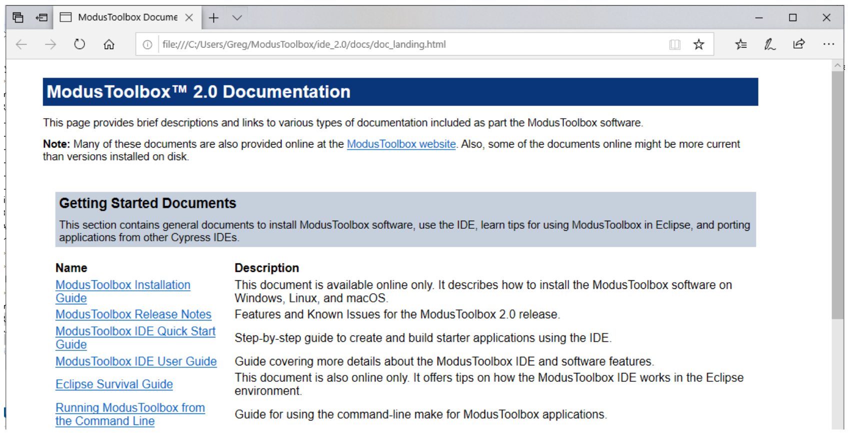
1.7.1 In the Eclipse IDE for ModusToolbox

In the Eclipse IDE Help menu there are 2 items of particular interest:



ModusToolbox General Documentation

The first item of interest has a link to the ModusToolbox Documentation Index which is a web page that looks like this:


ModusToolbox Docume
file:///C:/Users/Greg/ModusToolbox/ide_2.0/docs/doc_landing.html

ModusToolbox™ 2.0 Documentation

This page provides brief descriptions and links to various types of documentation included as part the ModusToolbox software.

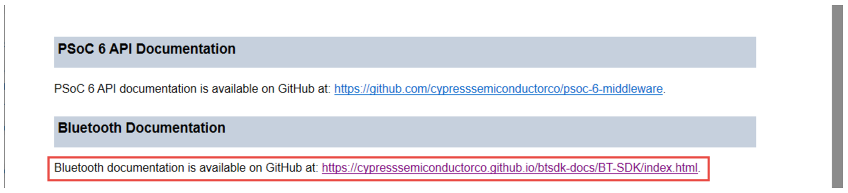
Note: Many of these documents are also provided online at the ModusToolbox website. Also, some of the documents online might be more current than versions installed on disk.

Getting Started Documents

This section contains general documents to install ModusToolbox software, use the IDE, learn tips for using ModusToolbox in Eclipse, and porting applications from other Cypress IDEs.

Name	Description
ModusToolbox Installation Guide	This document is available online only. It describes how to install the ModusToolbox software on Windows, Linux, and macOS.
ModusToolbox Release Notes	Features and Known Issues for the ModusToolbox 2.0 release.
ModusToolbox IDE Quick Start Guide	Step-by-step guide to create and build starter applications using the IDE.
ModusToolbox IDE User Guide	Guide covering more details about the ModusToolbox IDE and software features.
Eclipse Survival Guide	This document is also online only. It offers tips on how the ModusToolbox IDE works in the Eclipse environment.
Running ModusToolbox from the Command Line	Guide for using the command-line make for ModusToolbox applications.

At the bottom of that page you will find a link to the BT SDK API guide.



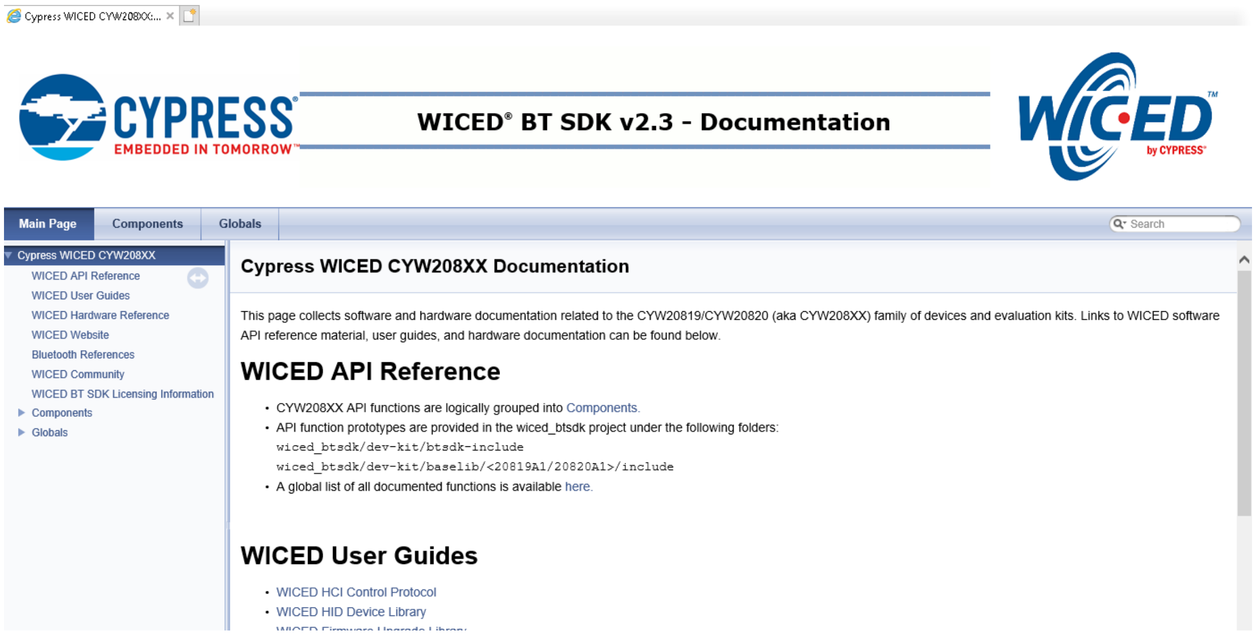
PSoC 6 API Documentation

PSoC 6 API documentation is available on GitHub at: https://github.com/cypresssemiconductorco/psoc-6-middleware.

Bluetooth Documentation

Bluetooth documentation is available on GitHub at: https://cypresssemiconductorco.github.io/btsdk-docs/BT-SDK/index.html.

The Bluetooth Documentation link takes you to a page where you select from a list of Bluetooth devices. This page is also available directly from a link in the Quick Panel. Once you select your device, you will get to this page:



Cypress WICED CYW208XX Documentation

This page collects software and hardware documentation related to the CYW20819/CYW20820 (aka CYW208XX) family of devices and evaluation kits. Links to WICED software API reference material, user guides, and hardware documentation can be found below.

WICED API Reference

- CYW208XX API functions are logically grouped into Components.
- API function prototypes are provided in the wiced_btsdk project under the following folders:
  `wiced_btsdk/dev-kit/btsdk-include`
  `wiced_btsdk/dev-kit/baselib/<20819A1/20820A1>/include`
- A global list of all documented functions is available here.

WICED User Guides

- WICED HCI Control Protocol
- WICED HID Device Library

The documentation on the right is useful for specific topics while the section on the left is invaluable for understanding the API functions, data types, enumerations, macros, etc.

Eclipse IDE Documentation

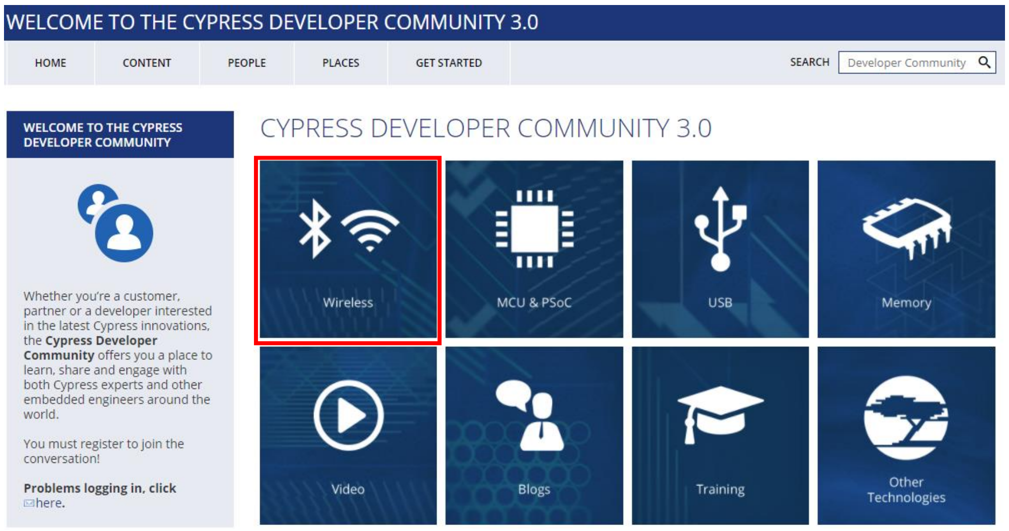
The second item on the Help Menu of interest contains documents for:

- Quick Start Guide
- User Guide
- Eclipse IDE Survival Guide

The last item contains tips and tricks on using the Eclipse IDE with Cypress devices intended for those who have relatively little experience with Eclipse.

1.7.2 On the Web

Navigating to "www.cypress.com > Design Support > Community" will take you to the following site (the direct link is https://community.cypress.com/welcome):


WELCOME TO THE CYPRESS DEVELOPER COMMUNITY 3.0

HOME CONTENT PEOPLE PLACES GET STARTED SEARCH Developer Community

WELCOME TO THE CYPRESS DEVELOPER COMMUNITY

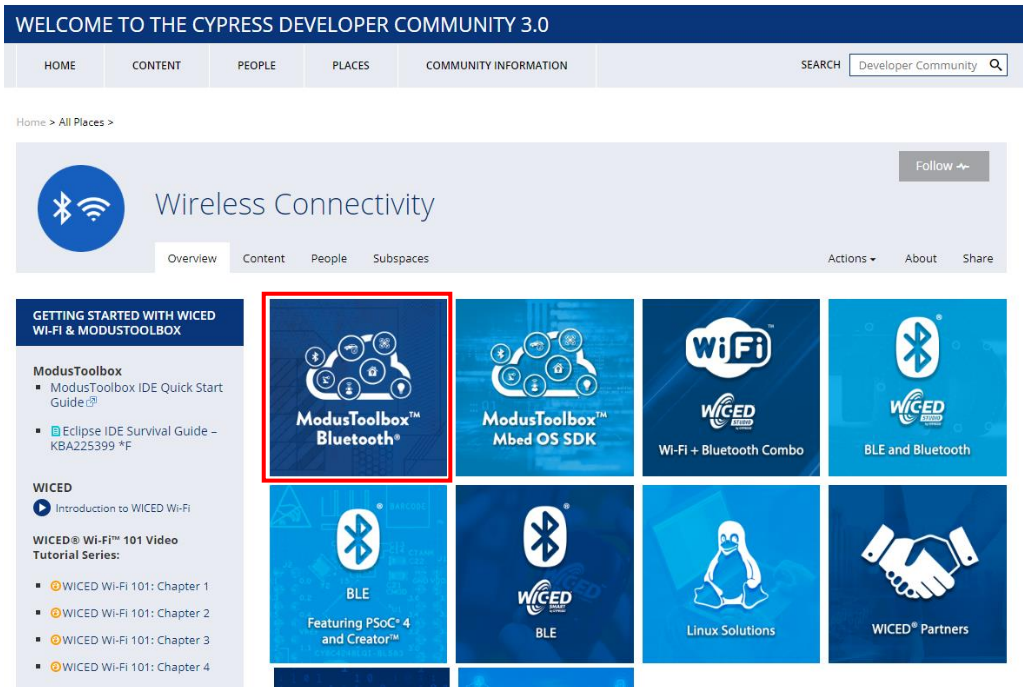
Whether you're a customer, partner or a developer interested in the latest Cypress innovations, the **Cypress Developer Community** offers you a place to learn, share and engage with both Cypress experts and other embedded engineers around the world.

You must register to join the conversation!

Problems logging in, click here.

CYPRESS DEVELOPER COMMUNITY 3.0

Clicking on the *Wireless* icon will take you to the page as shown below. From this page, you will find links to pages that allow you to download ModusToolbox, search for answers, ask questions, etc.


WELCOME TO THE CYPRESS DEVELOPER COMMUNITY 3.0

HOME CONTENT PEOPLE PLACES COMMUNITY INFORMATION SEARCH Developer Community

Home > All Places >

Wireless Connectivity

Follow

Overview Content People Subspaces Actions About Share

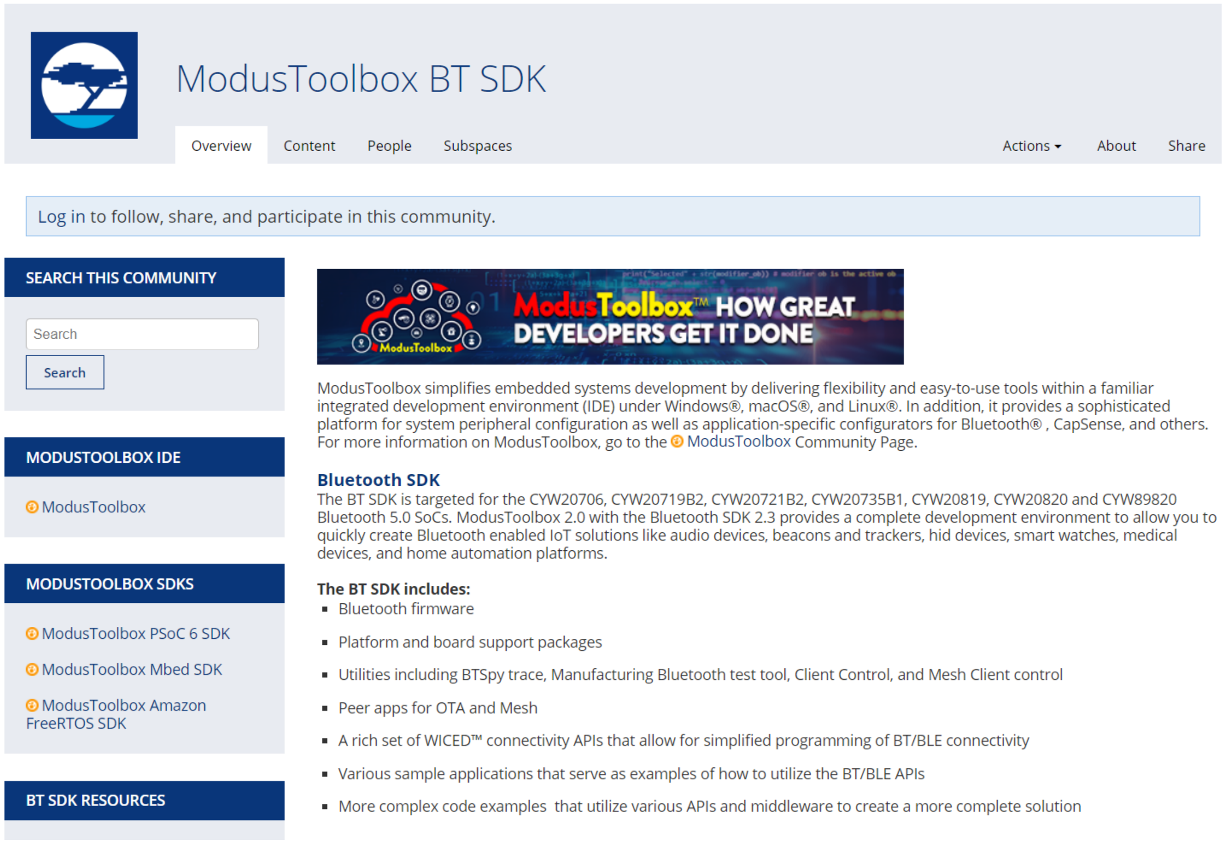
GETTING STARTED WITH WICED WI-FI & MODUSTOOLBOX

ModusToolbox
- ModusToolbox IDE Quick Start Guide
- Eclipse IDE Survival Guide – KBA225399 *F

WICED
Introduction to WICED Wi-Fi

WICED® Wi-Fi™ 101 Video Tutorial Series:
- WICED Wi-Fi 101: Chapter 1
- WICED Wi-Fi 101: Chapter 2
- WICED Wi-Fi 101: Chapter 3
- WICED Wi-Fi 101: Chapter 4

From the Wireless Connectivity page, click on ModusToolbox Bluetooth to get to the page where you can find downloads for ModusToolbox and information on the BT SDK as well as links to product guides, code examples, videos, etc.



ModusToolbox BT SDK

Overview Content People Subspaces Actions About Share

Log in to follow, share, and participate in this community.

SEARCH THIS COMMUNITY

Search

Search

MODUSTOOLBOX IDE

ModusToolbox

MODUSTOOLBOX SDKS

ModusToolbox PSoC 6 SDK

ModusToolbox Mbed SDK

ModusToolbox Amazon FreeRTOS SDK

BT SDK RESOURCES

ModusToolbox simplifies embedded systems development by delivering flexibility and easy-to-use tools within a familiar integrated development environment (IDE) under Windows®, macOS®, and Linux®. In addition, it provides a sophisticated platform for system peripheral configuration as well as application-specific configurators for Bluetooth® , CapSense, and others. For more information on ModusToolbox, go to the ModusToolbox Community Page.

Bluetooth SDK

The BT SDK is targeted for the CYW20706, CYW20719B2, CYW20721B2, CYW20735B1, CYW20819, CYW20820 and CYW89820 Bluetooth 5.0 SoCs. ModusToolbox 2.0 with the Bluetooth SDK 2.3 provides a complete development environment to allow you to quickly create Bluetooth enabled IoT solutions like audio devices, beacons and trackers, hid devices, smart watches, medical devices, and home automation platforms.

The BT SDK includes:

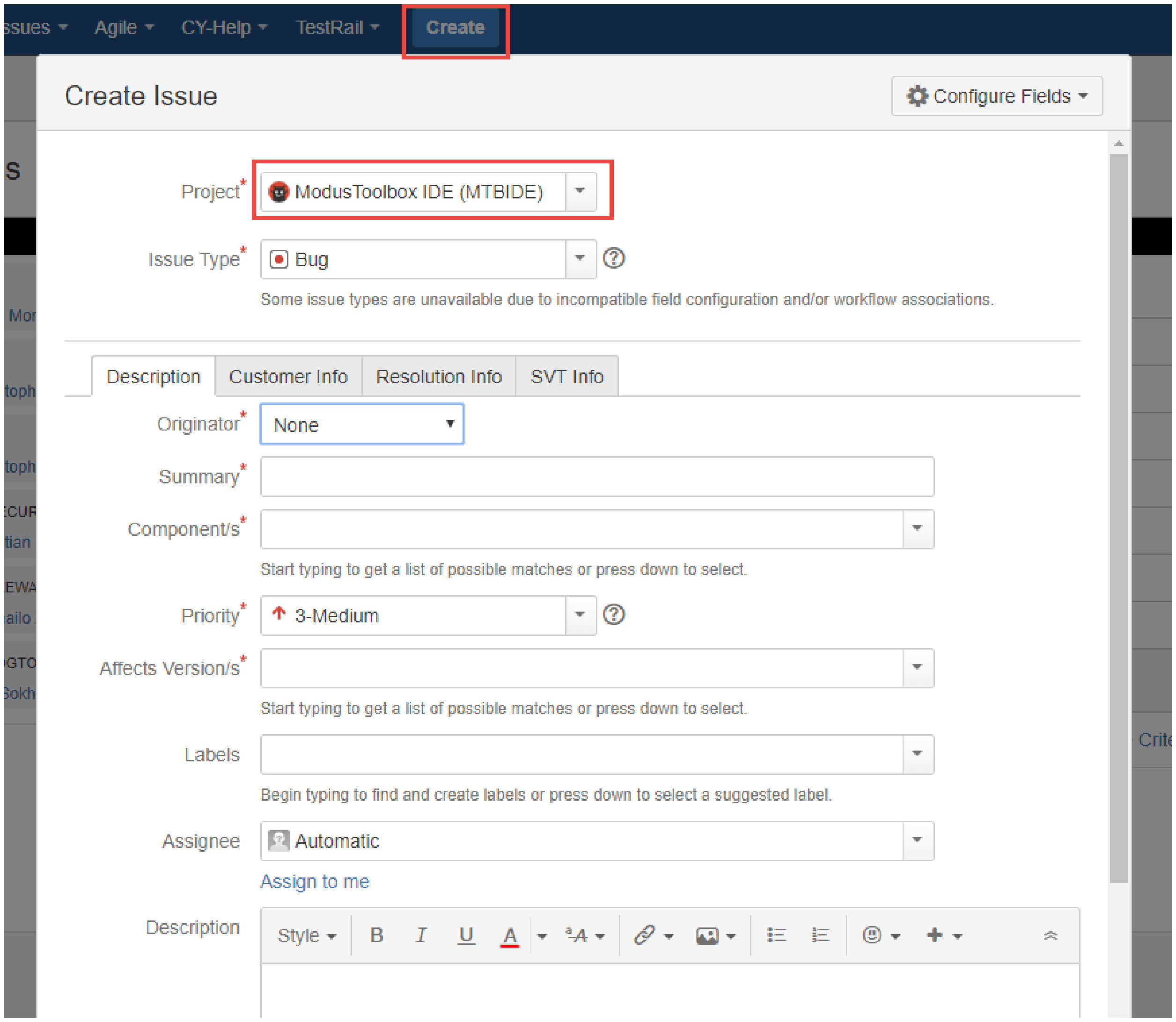
- Bluetooth firmware
- Platform and board support packages
- Utilities including BTSpy trace, Manufacturing Bluetooth test tool, Client Control, and Mesh Client control
- Peer apps for OTA and Mesh
- A rich set of WICED™ connectivity APIs that allow for simplified programming of BT/BLE connectivity
- Various sample applications that serve as examples of how to utilize the BT/BLE APIs
- More complex code examples that utilize various APIs and middleware to create a more complete solution

1.8 Reporting Issues

If you are a Cypress employee and you find an issue in ModusToolbox (bug, missing or confusing documentation, enhancement request), please use a "JIRA" to report it:

jira.cypress.com

Click on Create to start submitting a JIRA. Use the project type of "MTBIDE" or "BTSDK", depending on the issue and then fill in as many details as you can to report the issue.



Create Issue

Project* ModusToolbox IDE (MTBIDE)

Issue Type* Bug

Some issue types are unavailable due to incompatible field configuration and/or workflow associations.

Originator* None

Summary*

Component/s*

Start typing to get a list of possible matches or press down to select.

Priority* 3-Medium

Affects Version/s*

Start typing to get a list of possible matches or press down to select.

Labels

Begin typing to find and create labels or press down to select a suggested label.

Assignee Automatic

Description

Non-Cypress employees can ask questions and report issues in the developer community.

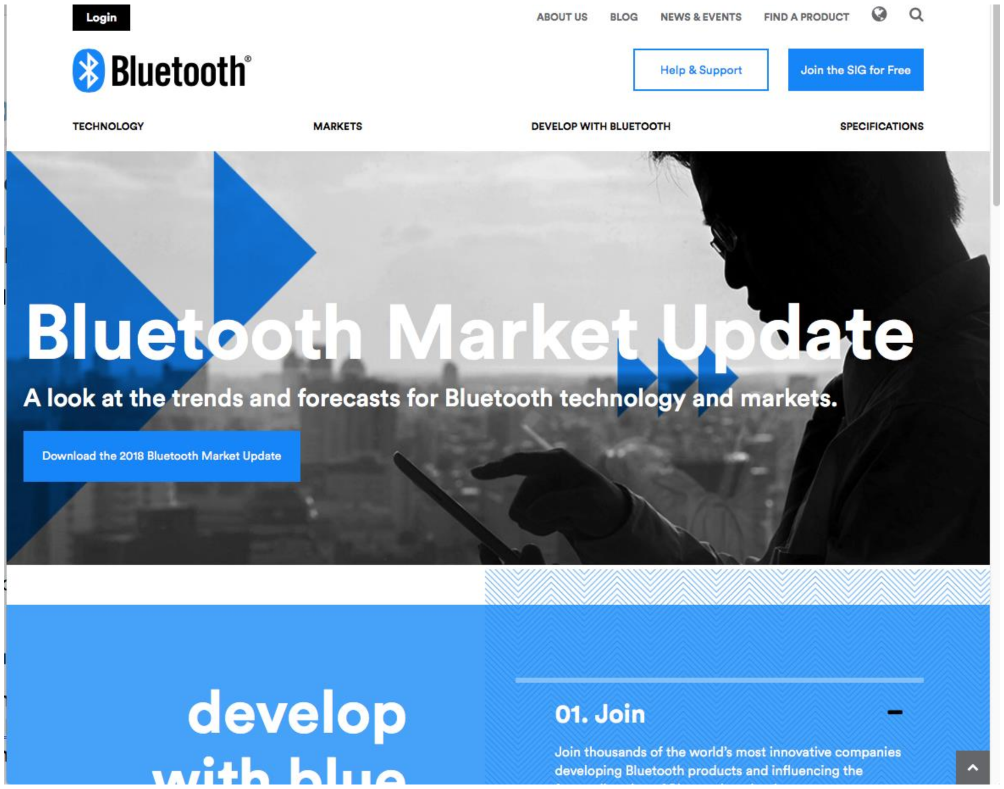
1.9 Tour of Bluetooth

Bluetooth is a short-range wireless standard that runs on the 2.4 GHz ISM (Industrial, Scientific, and Medical) band modulation. It is controlled by the Bluetooth Special Interest Group (SIG).

Discussions about Bluetooth are typically divided into *Classic Bluetooth* and *Bluetooth Low Energy*.

1.9.1 The Bluetooth Special Interest Group (SIG)

The Bluetooth Special Interest Group is an industry consortium that owns the specifications for Bluetooth. All the Bluetooth documentation is available at www.bluetooth.org. You can register for an account on that website.

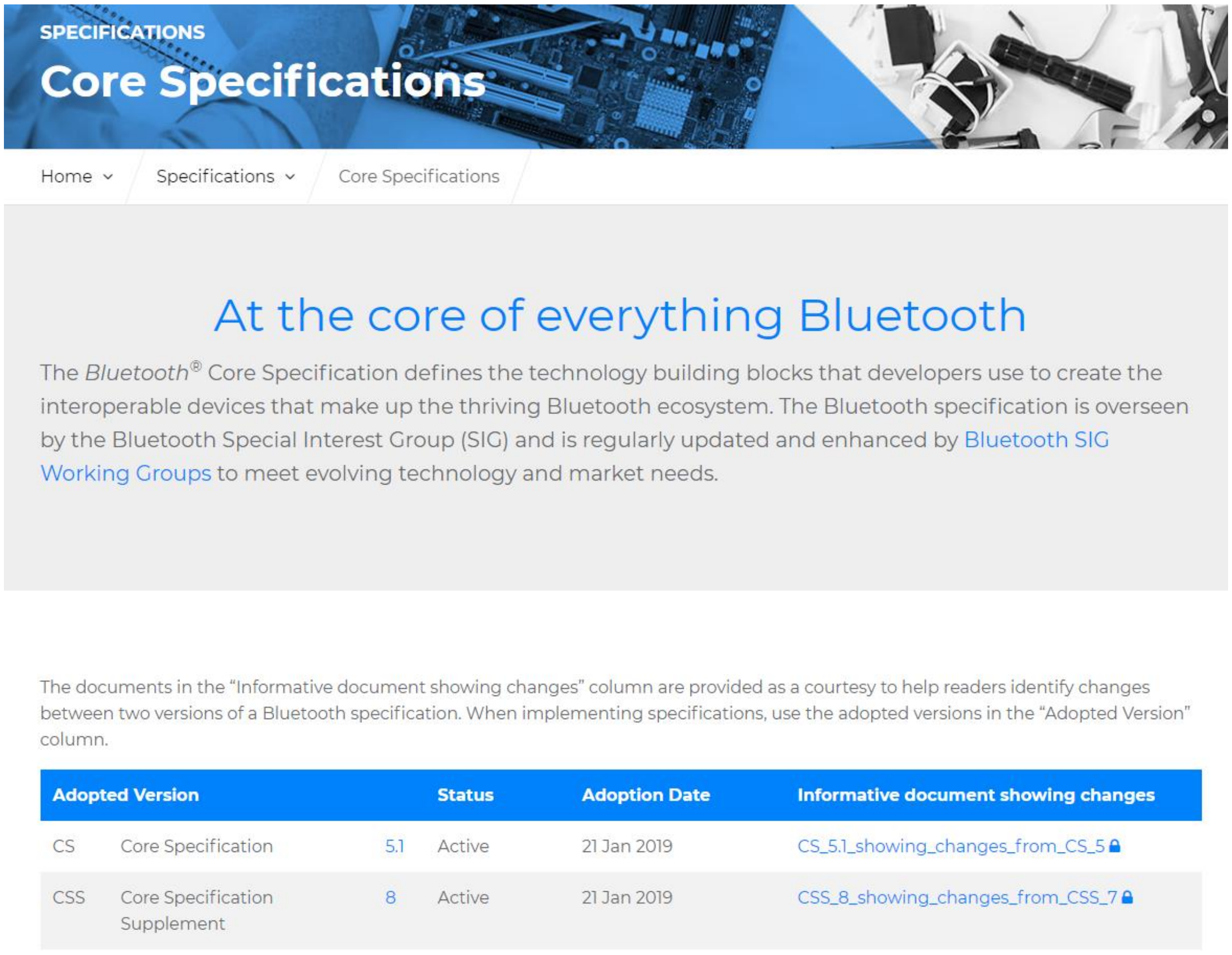


Bluetooth Market Update

A look at the trends and forecasts for Bluetooth technology and markets.

Join thousands of the world's most innovative companies developing Bluetooth products and influencing the

The current Bluetooth Specification is Version 5.1 is a ~3000 page long document that can be downloaded from the Bluetooth SIG website at https://www.bluetooth.com/specifications/bluetooth-core-specification



At the core of everything Bluetooth

The *Bluetooth*® Core Specification defines the technology building blocks that developers use to create the interoperable devices that make up the thriving Bluetooth ecosystem. The Bluetooth specification is overseen by the Bluetooth Special Interest Group (SIG) and is regularly updated and enhanced by Bluetooth SIG Working Groups to meet evolving technology and market needs.

The documents in the "Informative document showing changes" column are provided as a courtesy to help readers identify changes between two versions of a Bluetooth specification. When implementing specifications, use the adopted versions in the "Adopted Version" column.

Adopted Version			Status	Adoption Date	Informative document showing changes
CS	Core Specification	5.1	Active	21 Jan 2019	CS_5.1_showing_changes_from_CS_5 🔒
CSS	Core Specification Supplement	8	Active	21 Jan 2019	CSS_8_showing_changes_from_CSS_7 🔒

1.9.2 Classic Bluetooth

Classic Bluetooth uses 79 channels with a channel spacing of 1 MHz. It has three main speeds – Basic Rate (BR) and two Extended Data Rates (EDR). Each of these uses a different modulation scheme.

Mode	**Speed**	**Modulation**
Basic Rate	1 Mbps	GFSK (Gaussian Frequency Shift Keying)
Extended Data Rate	2 Mbps	π/4 DQPSK (Differential Quadrature Phase Shift Keying)
Extended Data Rate	3 Mbps	8DPSK (Octal Differential Phase Shift Keying)

The range is dependent on the transmission power which is divided into four classes:

Class	**Max Permitted Power**		**Typical Range**
	(mW)	(dBm)	**(m)**
1	100	20	100
2	2.5	4	10
3	1	0	1
4	0.5	-3	0.5

1.9.3 Bluetooth Low Energy

Bluetooth Low Energy (BLE) uses 40 channels with a channel spacing of 2 MHz (and so it shares the same range of frequencies with Bluetooth Classic). It provides much lower power consumption than Classic Bluetooth. Lower power is not achieved by reducing range (i.e. transmission power) but rather by staying actively connected for short bursts and being idle most of the time. This requires devices to agree on a connection interval. This connection interval can be varied to trade off the frequency of data transmitted vs. power. Therefore, BLE is excellent for data that can be sent in occasional bursts such as sensor states (i.e. temperature, state of a door, state of a light, etc.) but is not good for continuous streaming of data such as audio. BLE typically transmits data up to 1 Mbps, but 2 Mbps can be achieved in Bluetooth version 5 with shorter range.

1.9.4 Bluetooth History

Bluetooth Spec	Year	Major Features
1.0	1999	Initial standard.
1.1	2002	Many bug fixes. Addition of RSSI and non-encrypted channels.
1.2	2003	Faster connection and discovery. Adaptive Frequency Hopping (AFH) Host Control Interface (HCI) Addition of flow control and retransmission.
2.0 + EDR	2004	Addition of EDR (up to 3 Mbps).
2.1 + EDR	2007	Addition of Secure Simple Pairing (SPP) and enhanced security. Extended Inquiry Response (EIR).
3.0 + HS	2009	Addition of HS which uses Bluetooth for negotiation and establishment, then uses an 802.11 link for up to 24 Mbps. This is called Alternative MAC/PHY (AMP). Addition of Enhanced Retransmission Mode (ERTM) and Streaming Mode (SM) for reliable and unreliable channels.
4.0 + LE	2010	Addition of BLE. Addition of Generic Attribute Profile (GATT). Addition of Security Manager (SM) with AES encryption.
4.1	2013	Incremental software update.
4.2	2014	LE secure connections with data packet length extension. Link Layer privacy. Internet Protocol Support Profile (SPP) version 6.
5	2016	LE up to 2 Mbps for shorter range, or 4x range with lower data rate. LE increased packet lengths to achieve 8x data broadcasting capacity.
5.1	2019	Mesh-based model hierarchy Angle of Arrival (AoA) and Angle of Departure (AoD) fo rtracking Advertising Channel Index GATT Cacheing

1.10 Tour of Chips

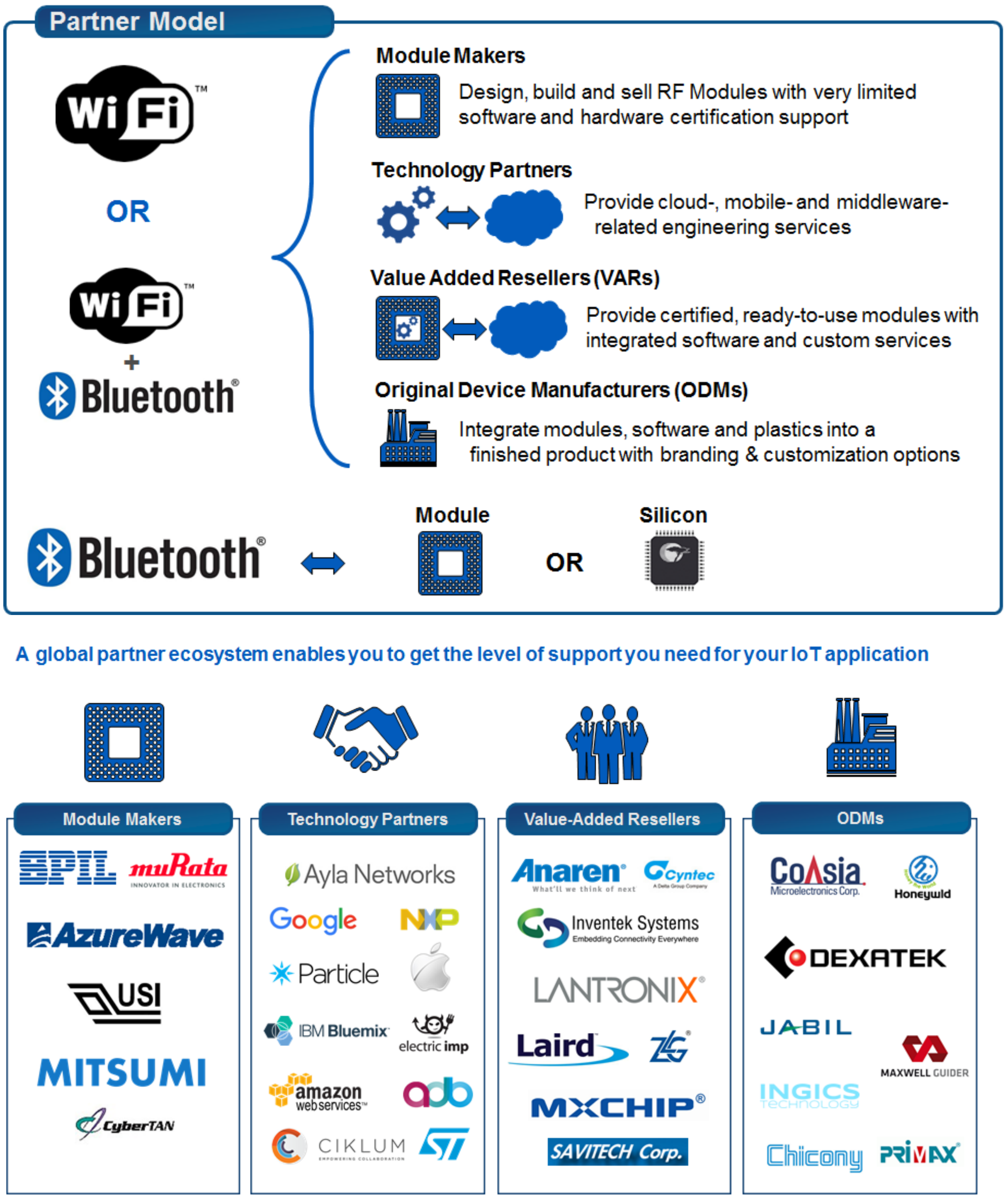
Device	Key Features	Notes
CYW20706	• Bluetooth BR, EDR and LE 5.x • ARM Cortex-M3 • 848 kB ROM • 352 kB RAM (data and patches) • 2 kB NVRAM	WICED Studio
CYW20719	• Bluetooth BR, EDR and LE 5.x • 2 Mbps LE v5 • 96 MHz ARM Cortex-M4 • Single Precision FPU • 2 MB ROM • 1 MB On-Chip Flash • 512 kB RAM	ModusToolbox
CYW20819	• Bluetooth BR, EDR and LE 5.x • BR/EDR 2 Mbps and 3Mbps • LE 2Mbps • Ultra-low power • 96 MHz ARM Cortex-M4 • 1 MB ROM • 256 kB On-Chip Flash • 176 kB RAM	ModusToolbox
PSoC 4 BLE	• BLE 4.2 • 48 MHz ARM Cortex-M0 • 256 kB On-Chip Flash • 32 kB RAM	PSoC Creator
PSoC 6 BLE	• BLE 5.x • 150 MHz ARM Cortex-M4 & M0+ • 1MB or 2 MB On-Chip Flash • 288 KB RAM	ModusToolbox PSoC Creator

This class covers only the WICED Bluetooth SoC devices, not the PSoC BLE devices.

The Cypress CYW20819 is an ultra-low power (ULP), highly integrated, and dual-mode Bluetooth wireless MCU. By leveraging the all-inclusive development platform ModusToolbox, it allows you to implement the industry's smallest-footprint, lowest-power Bluetooth Low Energy (BLE) and dual mode Bluetooth applications quickly. CYW20819 is a Bluetooth 5.x compliant SoC with support for Bluetooth Basic Rate (BR), Enhanced Data Rate (EDR), and BLE.

The CYW20819 employs the highest level of integration to eliminate all critical external components, thereby minimizing the device's footprint and the costs associated with implementing Bluetooth solutions. A 96 MHz CM4 CPU coupled with 256 kB on-chip flash and 1 MB ROM for stack and profiles offers significant processing power and flash space to customers for their applications. CYW20819 is the optimal solution for a range of battery-powered single/dual mode Bluetooth internet of things applications such as home automation, HID, wearables, audio, asset tracking, and so on.

1.11 Tour of Partners


Partner Model
Wi Fi
OR
Wi Fi
+
Bluetooth
Module Makers
Design, build and sell RF Modules with very limited software and hardware certification support
Technology Partners
Provide cloud-, mobile- and middleware-related engineering services
Value Added Resellers (VARs)
Provide certified, ready-to-use modules with integrated software and custom services
Original Device Manufacturers (ODMs)
Integrate modules, software and plastics into a finished product with branding & customization options
Bluetooth
Module
OR
Silicon
A global partner ecosystem enables you to get the level of support you need for your IoT application
Module Makers
SPIL
muRata
AzureWave
USI
MITSUMI
CyberTAN
Technology Partners
Ayla Networks
Google
NXP
Particle
IBM Bluemix
electric imp
amazon web services
CIKLUM
Value-Added Resellers
Anaren
Cyntec
Inventek Systems
LANTRONIX
Laird
MXCHIP
SAVITECH Corp.
ODMs
CoAsia Microelectronics Corp.
Honeywld
DEXATEK
JABIL
MAXWELL GUIDER
INGICS TECHNOLOGY
Chicony
PRIMAX

An IoT Selector Guide including partner modules available can be found in the Community at:

https://community.cypress.com/docs/DOC-3021

1.12 Tour of Development Kits

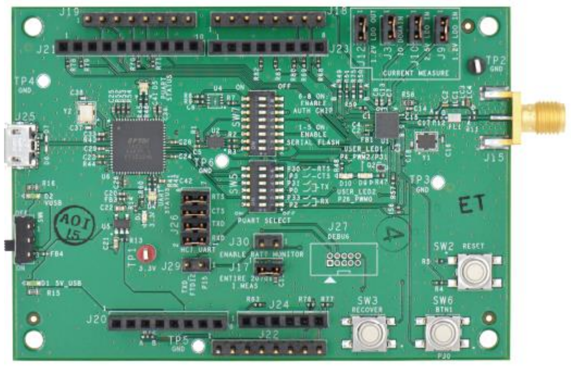
1.12.1 Cypress CYW920819EVB-02

- Bluetooth 5.x plus 2 Mbps LE from v5
- 96 MHz ARM Cortex-M4
- Integrated transceiver
- 1 MB ROM, 256 kB On-Chip Flash, 176 kB SRAM
- 1 User Button, 2 User LEDs
- USB JTAG Programmer/Debugger

1.12.2 Cypress CYBT-213043-MESH

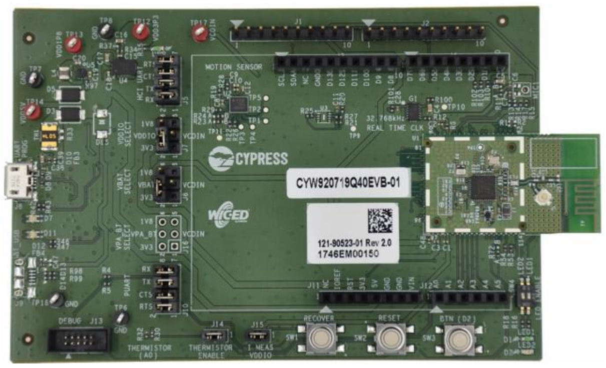
- Bluetooth Mesh kit with 20819 module
- Each kit contains 4 boards to evaluate mesh networks
- 1 User Button, RGB LED, ambient light sensor, PIR motion sensor

1.12.3 Cypress CYW920706WCDEVAL



- Monolithic, Single-chip, Bluetooth 5.x + HS
- ARM Cortex-M3 processor
- Integrated transceiver
- 848 kB ROM, 352 kB SRAM (data and patches), 2 kB NVRAM, 512 kB External Serial Flash
- 1 User Button, 2 User LEDs
- USB JTAG Programmer/Debugger

1.12.4 Cypress CYW920719Q40EVB-01



- Bluetooth 5.x plus 2 Mbps LE from v5
- 96 MHz ARM Cortex-M4
- Integrated transceiver
- 2 MB ROM, 1 MB On-Chip Flash, 512 kB SRAM
- 1 User Button, 2 User LEDs
- USB JTAG Programmer/Debugger

1.13 Exercise(s)

Exercise - 1.1 Create a Forum Account

1. Go to https://community.cypress.com/welcome
2. Click "Log in" from the top right corner of the page and login to your Cypress account. If you do not have an account, you will need to create one first.
3. Once you are logged in, click the "Wireless" icon and then explore.

Exercise - 1.2 Start the Eclipse IDE for ModusToolbox, and Explore the Documentation

1. Run the Eclipse IDE and create a new workspace.
2. Explore the different documents available in the Help menu such as the Eclipse IDE for *ModusToolbox Help*, *Quick Start Guide, User Guide* and *Eclipse IDE Survival Guide*.
3. Open and explore the *ModusToolbox Documentation Index* page. Be sure to look at the *CYW920819EVB-02 Kit* link and the *Bluetooth Documentation -> CYW208XX API Reference*.

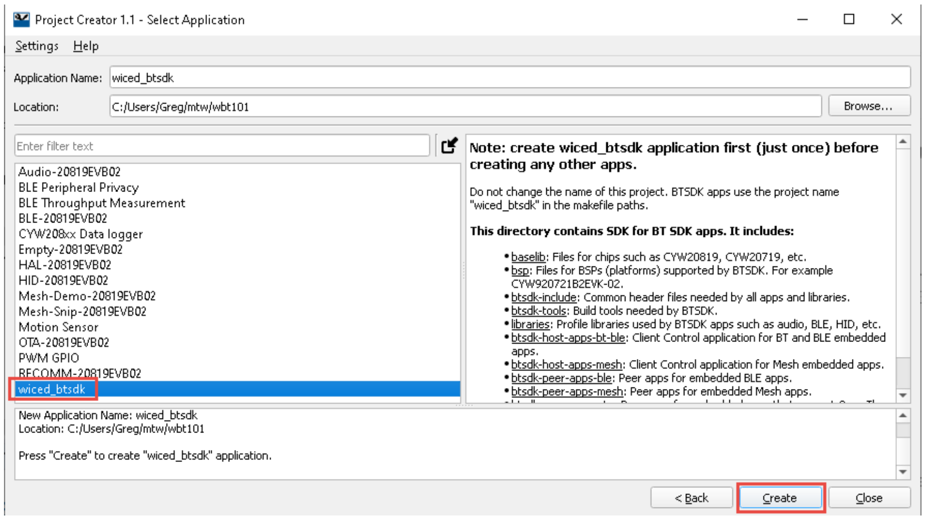
Questions to answer:

1. Where is the WICED API documentation for the PWM located?

Exercise - 1.3 Program a Simple Application

In this exercise, you will install the wiced_btsdk, create a new application, and then build/program it to a kit.

1. In the Quick Panel tab click "New Application".
2. If you are on a Cypress internal network, set the required Proxy settings. See section 1.2
3. Select the CYW920819EVB-02 kit and click "Next >".
4. Select "wiced_btsdk" from the list of applications and click "Create". It will take a few minutes to create the application.



Application Name: wiced_btsdk
Location: C:/Users/Greg/mtw/wbt101

Note: create wiced_btsdk application first (just once) before creating any other apps.

Do not change the name of this project. BTSDK apps use the project name "wiced_btsdk" in the makefile paths.

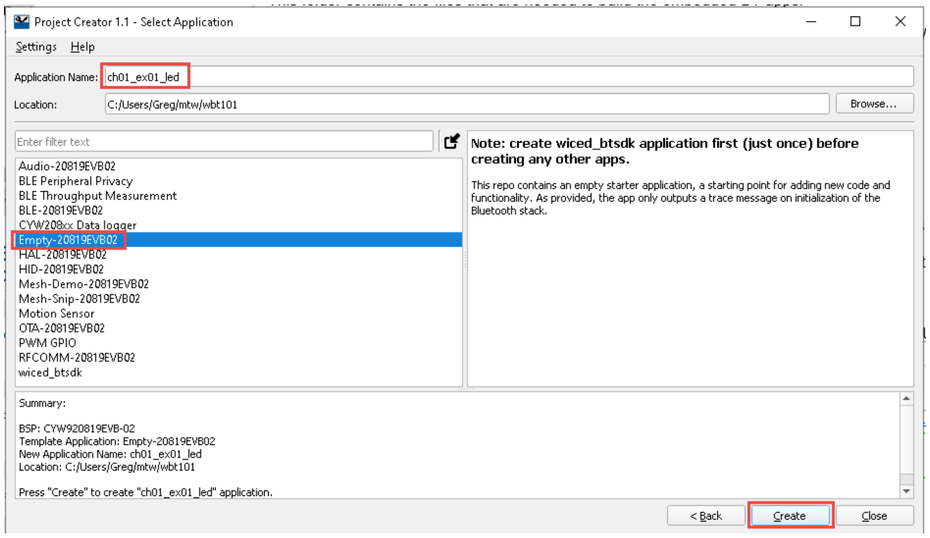
This directory contains SDK for BT SDK apps. It includes:

- baselib: Files for chips such as CYW20819, CYW20719, etc.
- bsp: Files for BSPs (platforms) supported by BTSDK. For example CYW920721B2EVK-02.
- btsdk-include: Common header files needed by all apps and libraries.
- btsdk-tools: Build tools needed by BTSDK.
- libraries: Profile libraries used by BTSDK apps such as audio, BLE, HID, etc.
- btsdk-host-apps-bt-ble: Client Control application for BT and BLE embedded apps.
- btsdk-host-apps-mesh: Client Control application for Mesh embedded apps.
- btsdk-peer-apps-ble: Peer apps for embedded BLE apps.
- btsdk-peer-apps-mesh: Peer apps for embedded Mesh apps.

New Application Name: wiced_btsdk
Location: C:/Users/Greg/mtw/wbt101

Press "Create" to create "wiced_btsdk" application.

5. Once the wiced_btsdk application has been created, select "Empty-20819EVB02".
6. Change the application name to **ch01_ex01_led** and click "Create".



Application Name: ch01_ex01_led
Location: C:/Users/Greg/mtw/wbt101

Note: create wiced_btsdk application first (just once) before creating any other apps.

This repo contains an empty starter application, a starting point for adding new code and functionality. As provided, the app only outputs a trace message on initialization of the Bluetooth stack.

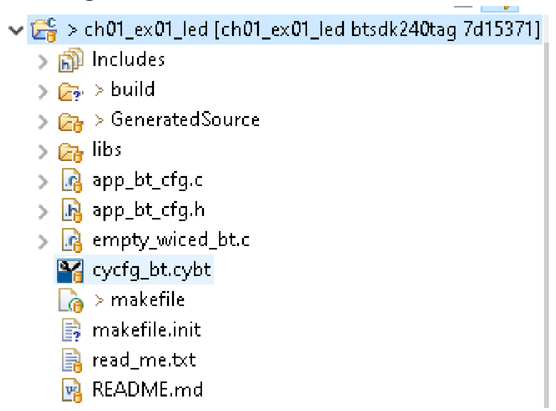
Summary:

BSP: CYW920819EVB-02
Template Application: Empty-20819EVB02
New Application Name: ch01_ex01_led
Location: C:/Users/Greg/mtw/wbt101

Press "Create" to create "ch01_ex01_led" application.

Note: you can call the application anything you like but it will really help if you maintain an alphabetically sortable naming scheme – you are going to create quite a few projects.

7. Once the ch01_ex01_led application has been created, click "Close". The applications will now be imported into the Eclipse IDE. When that's done, the Project Explorer window in Eclipse should look like the following:


> ch01_ex01_led [ch01_ex01_led btsdk240tag 7d15371]
Includes
> build
> GeneratedSource
libs
app_bt_cfg.c
app_bt_cfg.h
empty_wiced_bt.c
cycfg_bt.cybt
> makefile
makefile.init
read_me.txt
README.md

8. Open the top-level C file which in this case is called empty_wiced_bt.c.
9. Add the following line in the application_start function:

```
wiced_hal_gpio_set_pin_output(WICED_GPIO_PIN_LED_1, GPIO_PIN_OUTPUT_LOW);
```

10. Connect your CYW920819EVB-02 kit to a USB port on your computer.
11. In the Quick Panel, look in the "Launches" section and click the "ch01_ex01_led Program" link.
12. Once the build and program operations are done, you should see "Download succeeded" in the Console window and LED1 (the yellow user LED) should be on.

Exercise - 1.4 Use the Command Line

In this exercise you will use the command line to modify/build/program the same application as the previous exercise.

1. Open ModusShell (Windows) or a Terminal (MacOS and Linux).
2. Go to the directory containing the application from the previous exercise.
3. Edit the file empty_wiced_bt.c with a file editor of your choice to change to turn the LED OFF instead of ON.
4. Run the appropriate make command to build and then program your kit.

Exercise - 1.5 (Advanced) Use Visual Studio Code

In this exercise you will use Visual Studio Code to modify/build/program the same application as the previous exercise.

1. Open ModusShell (Windows) or a Terminal (MacOS and Linux).
2. Go to the directory containing the application from the previous exercise.
3. Run the make vscode command to create the VS Code files.
4. Open the application in VS Code.
5. Edit the file empty_wiced_bt.c to turn the LED ON instead of OFF.
6. Run the appropriate build task to build and program your kit.