Chapter 1: Tour of Cypress Bluetooth

Time: ¾ Hour

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

1.1 Tour of ModusToolbox IDE

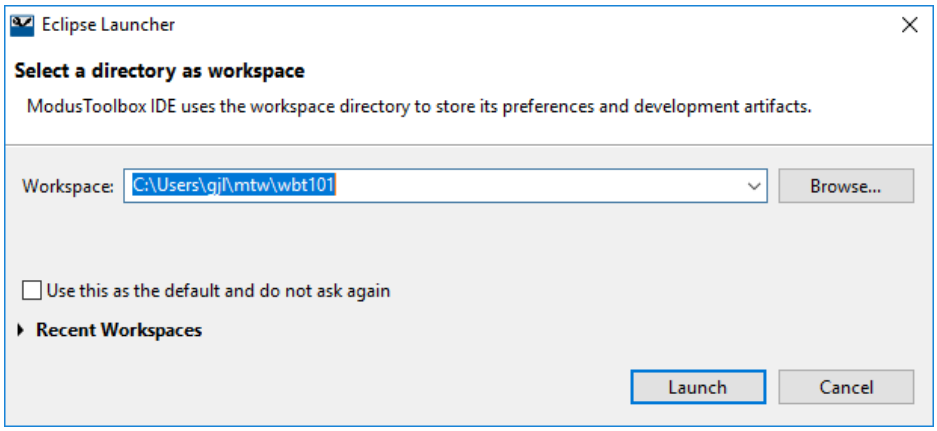
1.1.1 First Look

The software tool we will use is called "ModusToolbox IDE" and it is based on Eclipse.

ModusToolbox is installed, by default, in *C:/Users/<UserName>/ModusToolbox_1.1.* Once installed, the IDE will show up in Windows under *Start->All Programs->ModusToolbox 1.1*.

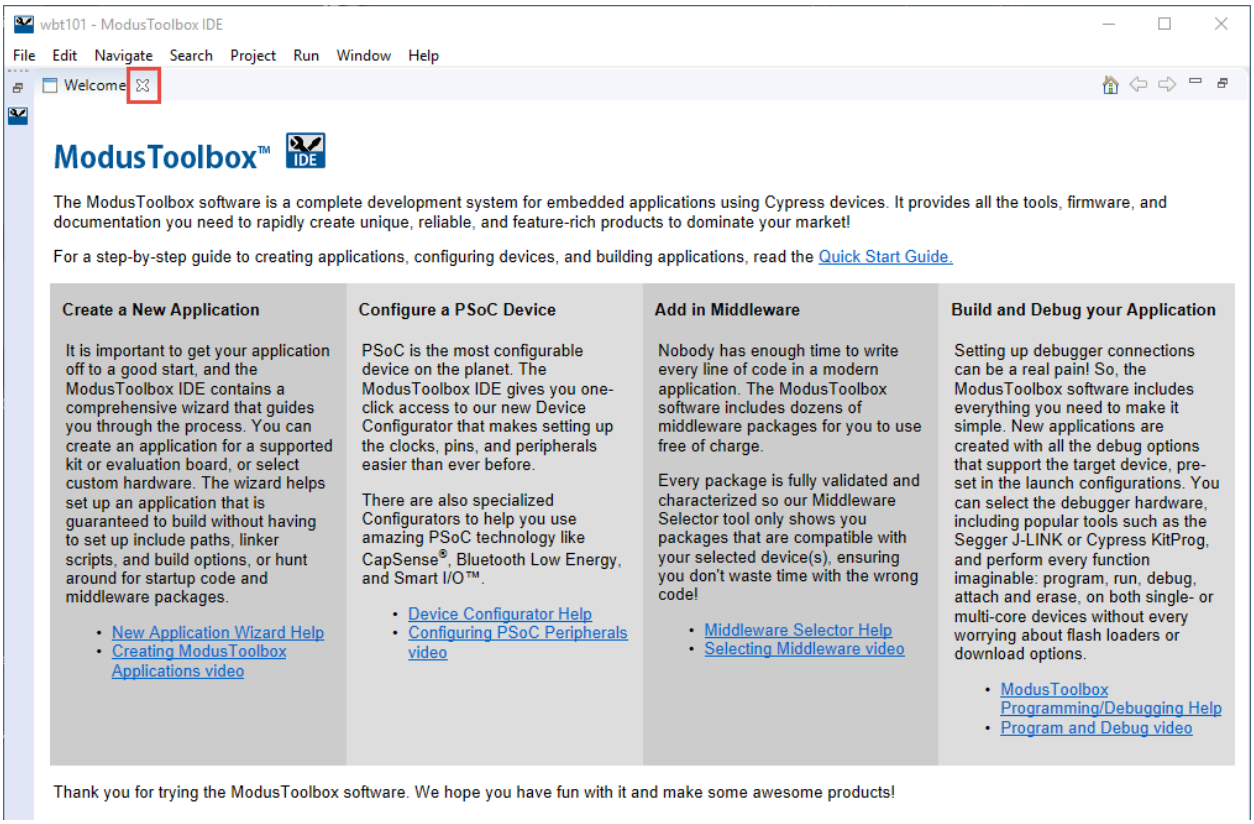
When you first run ModusToolbox IDE, you will create a Workspace to hold your applications. The default Workspace location is *C:/Users/<UserName>/mtw* but you can have as many Workspaces as you want, and you can put them anywhere you want. I usually put each workspace under a folder inside *C:/Users/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

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



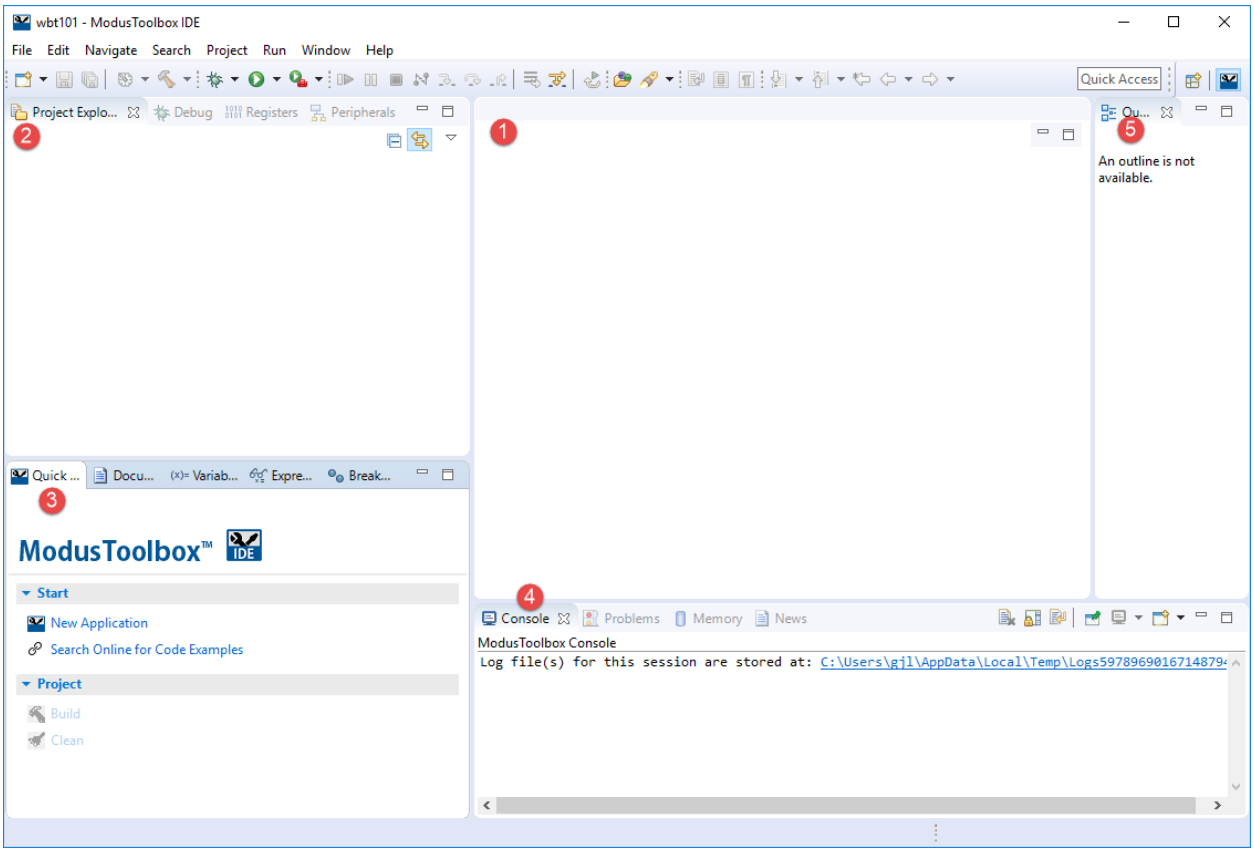
Whenever you want to switch to a different Workspace or create a new one, just use the menu item *File->Switch Workspace* and either select one from the list or select *Other…* to specify the name of an existing or new Workspace.

After clicking Launch, the IDE will start. When you open a new workspace, the first window you see will be the "Welcome" window which has some getting started information including help documents and short instructional videos.



Once you are done with that window, close it out to 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*.



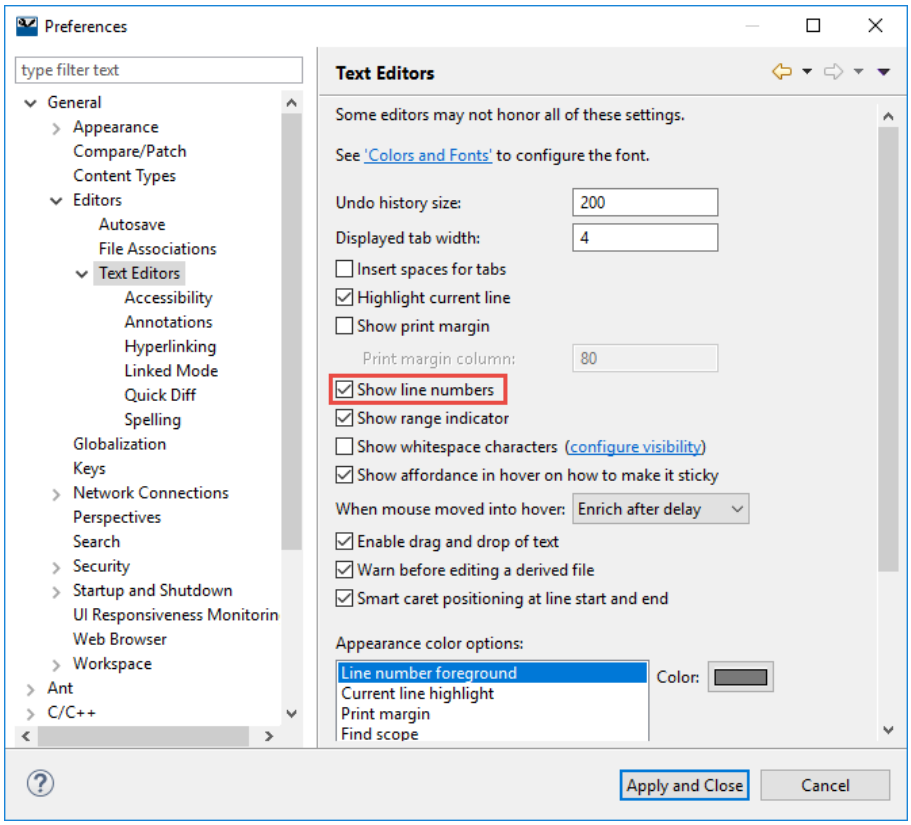
The major views are:

1. File Editor
2. Project Explorer
3. Quick Panel / Documents
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.

1.1.2 Customization

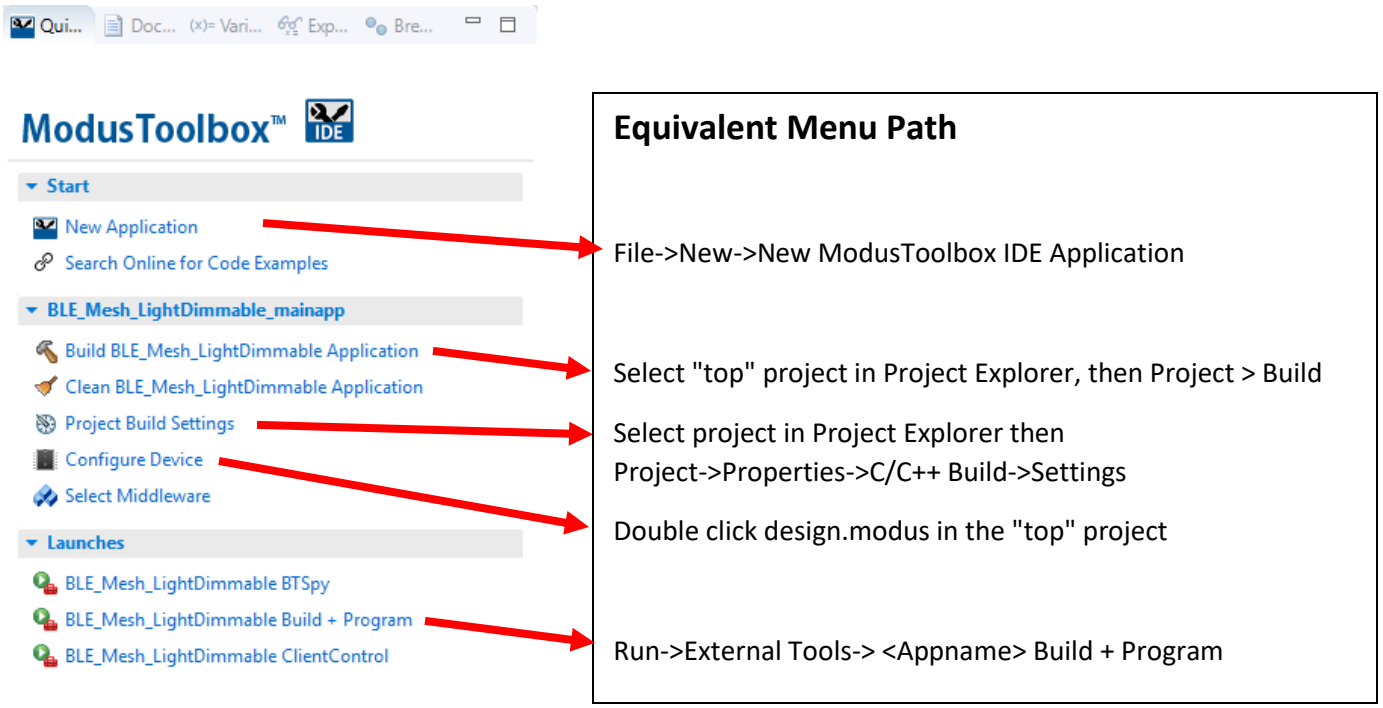
Eclipse is extremely flexible – you can customize almost anything if you know where to look. A good place to start for general Eclipse settings is *Window->Preferences*. One that I always turn on is *General->Editors->Text Editors->Show line numbers*. Most of these settings are at the workspace level.



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

1.1.3 Quick Panel

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, and launches.



1.1.4 Applications and Projects

At this point, we are ready to create an application. In the world of ModusToolbox, an application is "deployable firmware designed for the target hardware" while a project is "a compilation unit, organized into an Eclipse project". An application in ModusToolbox IDE may have more than one Eclipse project.

When you create a new application you always begin with an existing starter application. These range in complexity from mostly empty to a fully functional demo. Some starter applications are built into the IDE, others are available as code examples from GitHub or other locations. For this class we will provide you with template starter applications for many of the exercises.

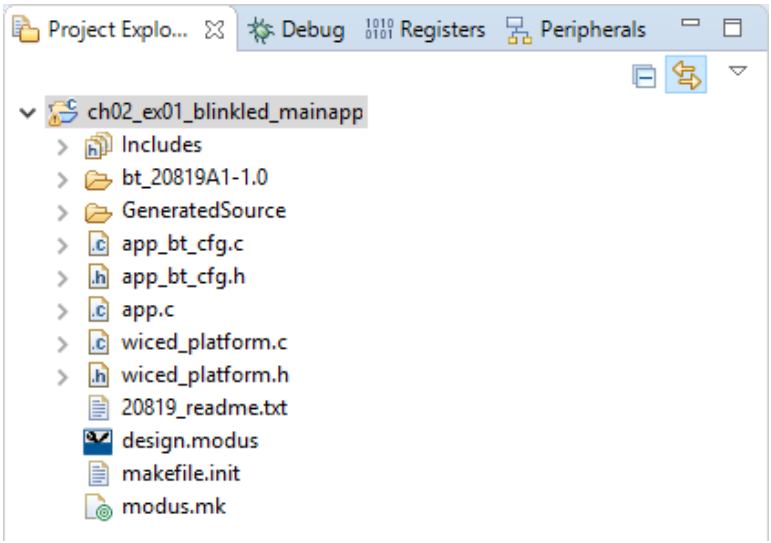
We will go through the details on the different ways to create a new application in the next chapter, but for now let's look at what an application will look like in the IDE once it is created.

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.

Project Explorer

In the project explorer window, you will see the project and all its associated files. WICED Bluetooth applications will consist of a single project.



The top-level C-file name (the one containing application_start) will depend on the application that you started from. In the example shown above, the top-level file is called app.c. Often the top-level file will have the same name as the application.

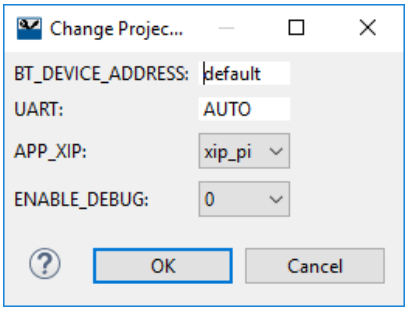
Bluetooth applications will have app_bt_cfg.c and app_bt_cfg.h which contain Bluetooth configuration information such as advertising intervals, buffer pool sizes, etc.

The files wiced_platform.c and wiced_platform.h contain board specific configuration and functions. Additional board specific information is contained in the GeneratedSource folder. This information is created by the device configurator which we will see in a minute. In ModusToolbox 2.0 and beyond, the wiced_platform.c and wiced_platform.h files will go away, and all configuration information will be contained in the configurator files.

The modus.mk file is used in the application creation process – it defines everything that ModusToolbox IDE needs to know to create the application.

Application Settings

If you right-click on an application in the project explorer and choose "Change Application Settings…" you will see a window like the following. This has settings that can be changed for the application.

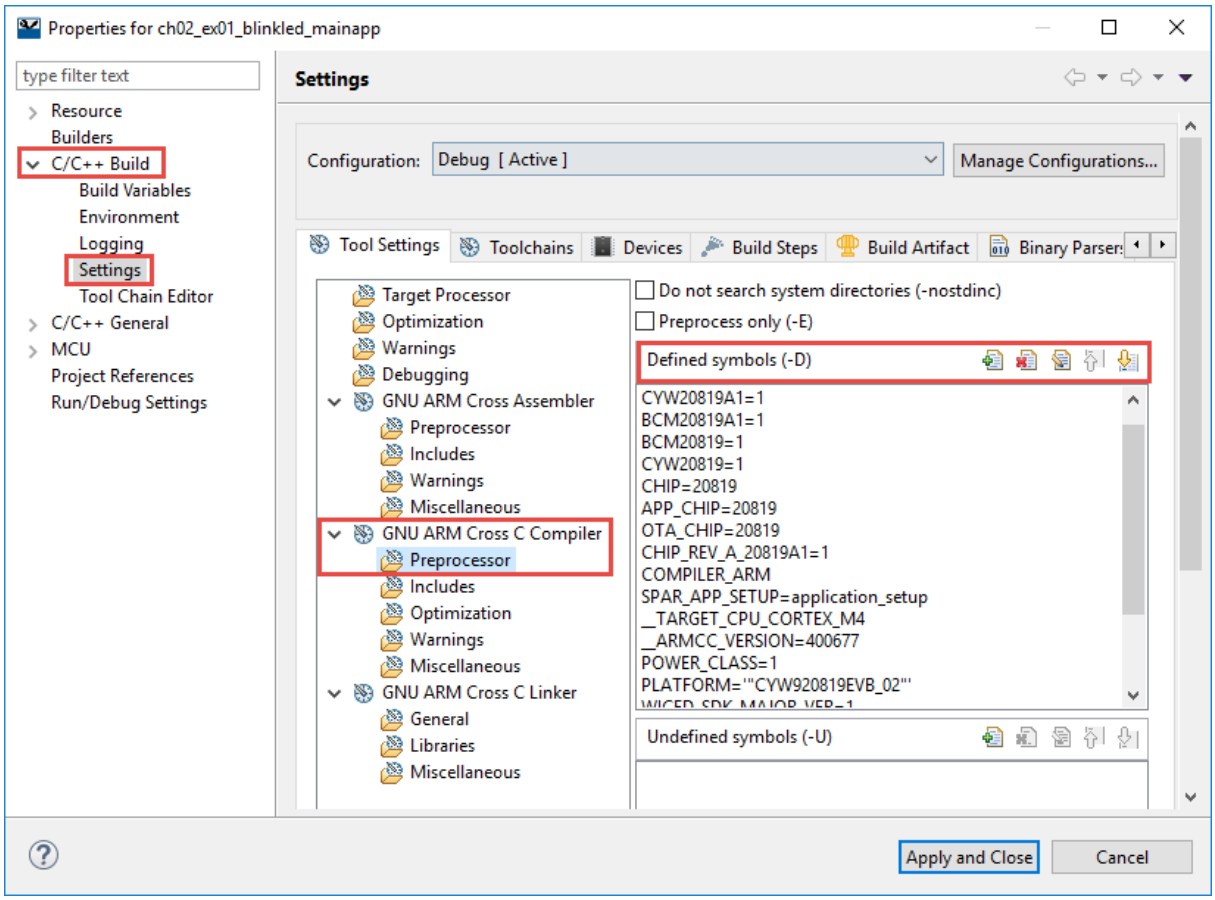


The settings entered here are typically used in the code to enable or disable some feature or may be used by the build process (such as specifying the UART to use for programming).

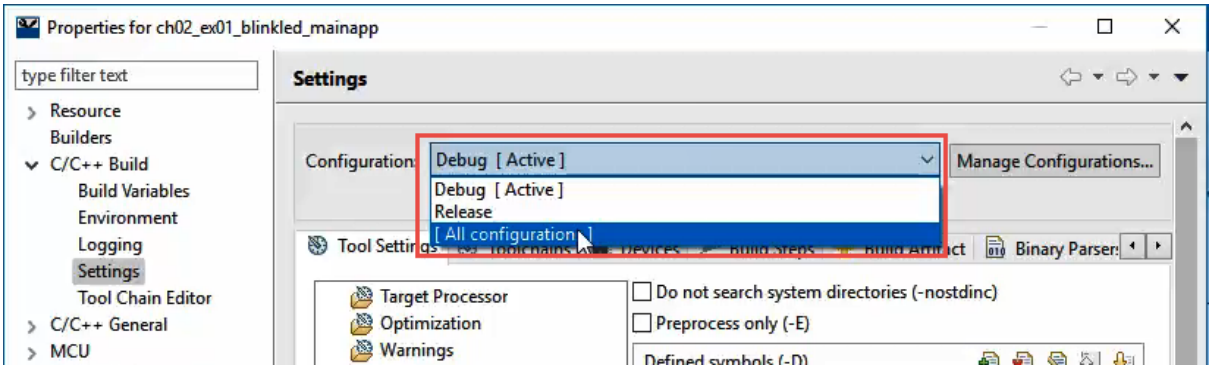
Project Build Settings

Build settings are project specific. You can access them from the Quick Panel, by right-clicking on the project and selecting *Properties*, or from the menu item *Project->Properties*. You will notice that there are a LOT of properties. Most of them are fine with default values but just know that just about anything can be customized.

For build settings, you will find them under the left window hierarchy of *C/C++ Build->Settings*. Inside the Settings window there are tabs for *Tool Settings*, *Toolchains*, etc. Under *Tool Settings* there is additional hierarchy containing settings for *Optimization*, *Assembler*, *Compiler*, etc. For example, you can add #define processor directives to your project under *GNU ARM Cross C Compiler->Preprocessor*.



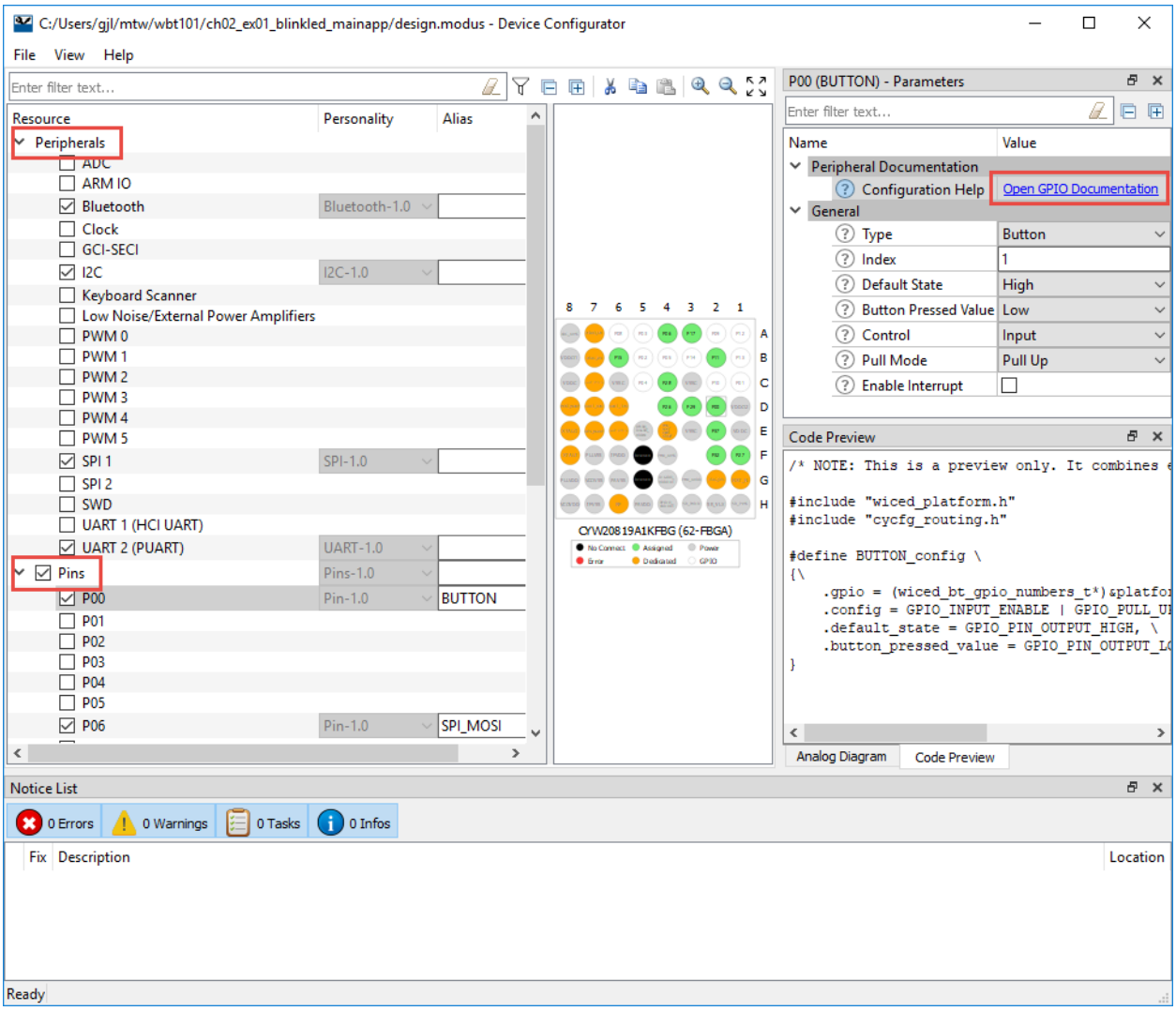
The build settings are specific to a build configuration (e.g. Debug vs. Release). If you want to make changes to all configurations at once, you can choose [All Configurations] from the drop-down menu.



To select the active configuration, right-click on the project and select *Build Configurations->Set Active* or use the menu item *Project->Build Configurations->Set Active*. Again, remember that build configurations are specific to a project, NOT an application. If your application has multiple projects and you want to set the active configuration for all of them, select them all in the project explorer window first (use Shift-Click on Control-Click) and then set the active configuration.

Configure Device

Based on the board that you select when you create an application, many items are pre-configured. However, you can change the configuration using the Configure Device link in the Quick Panel or by double-clicking on the design.modus file in the top project for your application. This will launch the Device Configurator.



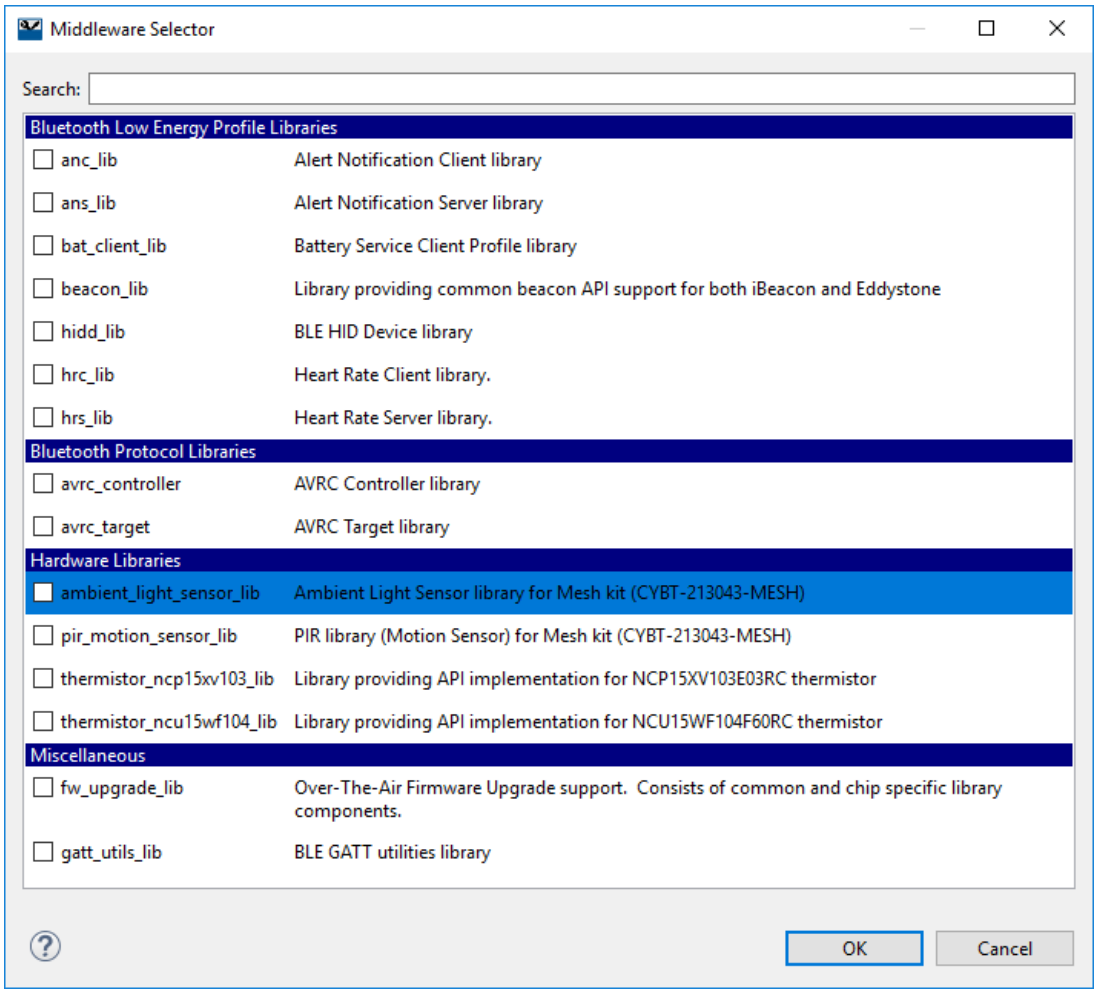
As you can see there are sections for Peripherals and Pins on the left. When you enable a peripheral or pin, the upper right-hand panel allows you to select configuration options and to open the documentation for that element. In some cases, you can launch a secondary configurator from that window (Bluetooth is one of those cases).

Once you save the configurator information (*File->Save*) it creates/updates the Generated Source files in the project.

We will explore the configurators in much more detail in later chapters.

Select Middleware

The Select Middleware link in the Quick Panel allows you to enable/disable various middleware libraries that are supported for the device you have chosen. For example, you will notice that there are middleware libraries for the ambient light and PIR motion sensors that are included on the CYBT-213043 -MESH kit.



When you enable middleware and click OK, the project will be updated with a *libraries* folder which will contain the code for the chosen middleware packages.

1.1.5 Sharing Applications

As discussed above, creation of a new application relies on a starter application (i.e. template). The starter application includes a modus.mk file and the source files. If the application uses custom device configuration settings, it will also include design.modus and wiced_platform.h files.

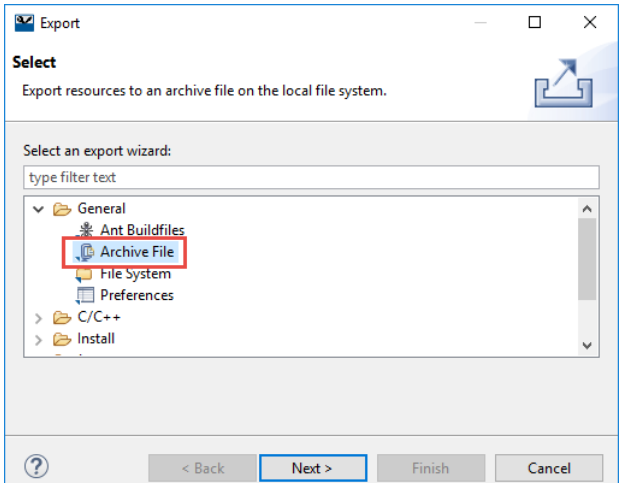
The modus.mk has the instructions on how to build the project. You can do the same thing for your own applications if you desire but remember that the modus.mk is NOT automatically updated when you make changes to an application, so you may need to update it manually. Also note that if you need

custom device configuration settings, you must specify the path to your custom design.modus and wiced_platform.h files in the list of source files in modus.mk. Otherwise, application creation will use the default files from the board support package (which we will discuss in the next chapter).

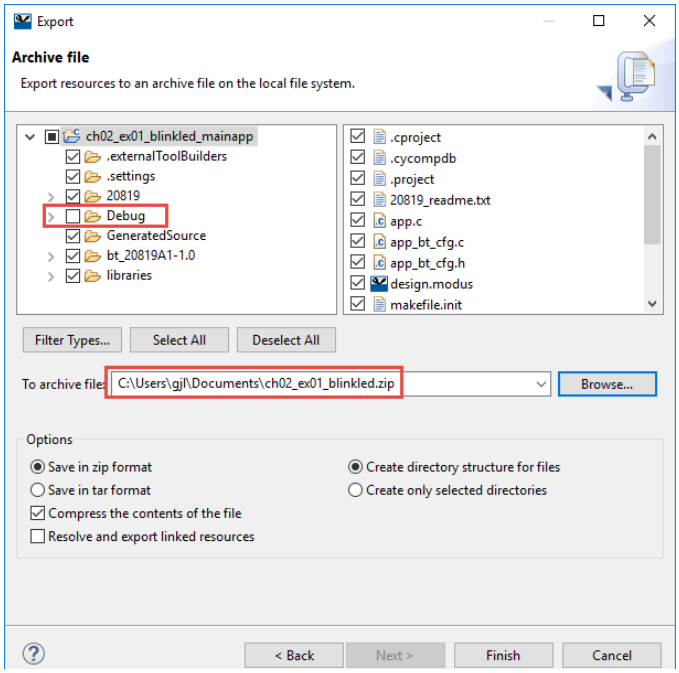
Another way to share an application is to export it from ModusToolbox IDE into an archive file. This captures a full copy of the application that can then be imported as existing projects into another workspace. The steps are:

Export to Archive

1. Select the project or projects to be exported
2. Choose *File->Export->General->Archive File* and click Next



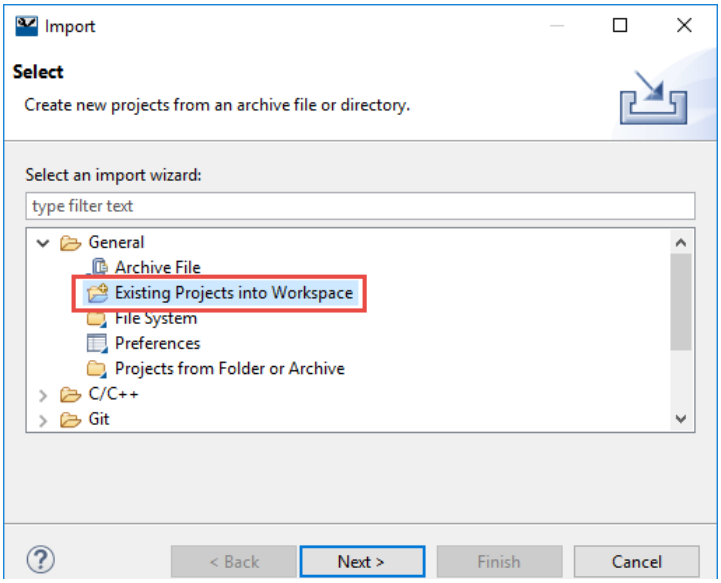
3. Uncheck the Debug and Release folders if they exist to save space (these are build output files)
4. Enter the desired file path/name



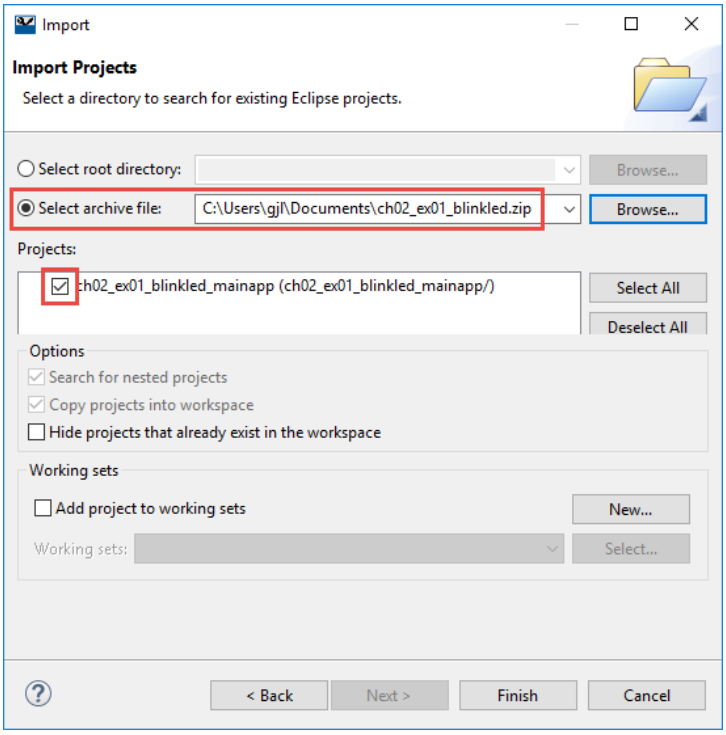
5. Click Finish

Import from Archive

1. In the new workspace choose *File->Import->General->Existing Projects into Workspace*
   a. DO NOT choose ~~*File->Import->General->Archive File*~~*!*



2. Choose *Select archive file* and specify the path to the zip file.
3. Check the box next to the project(s) you want to import.

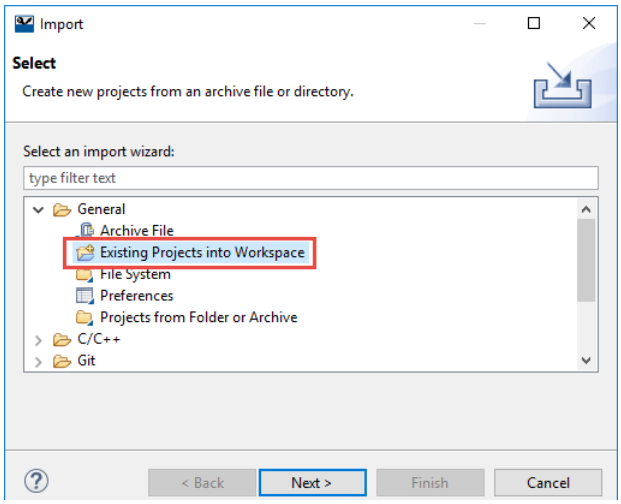


4. Click Finish

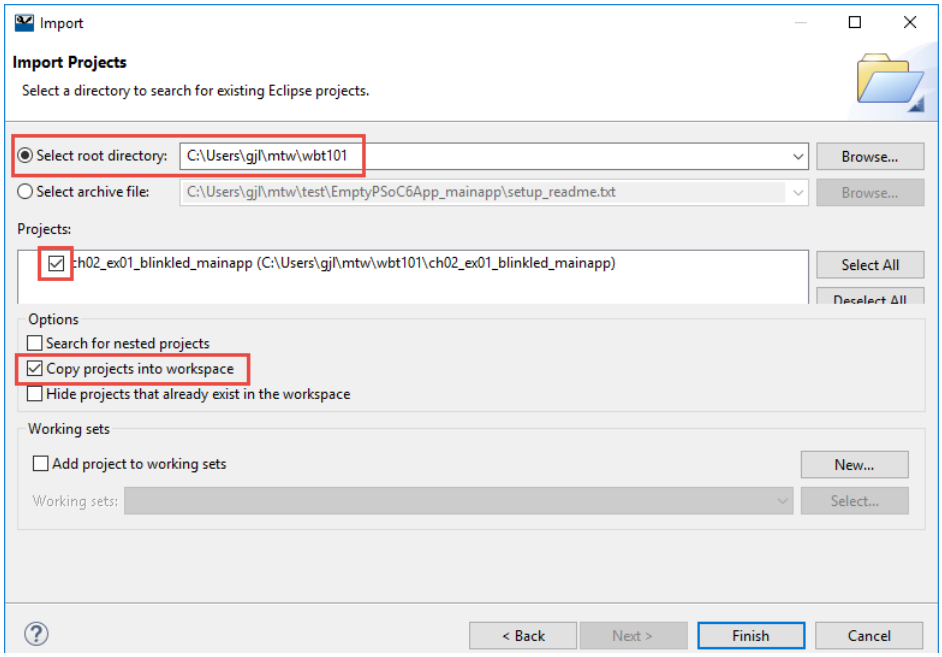
Yet another way to share a full application is to just use the filesystem path to the application when importing existing projects. This works the same as the previous option except that an archive is not created first. The steps are:

Import from Filesystem

1. In the new workspace choose *File->Import->General->Existing Projects into Workspace* and click Next.



2. Choose the path to the projects to be imported.
    a. This can either be the path to a workspace the path to an individual project.
3. Check the box next to the project tor projects you want to import.
4. Check the box *Copy projects into workspace*.

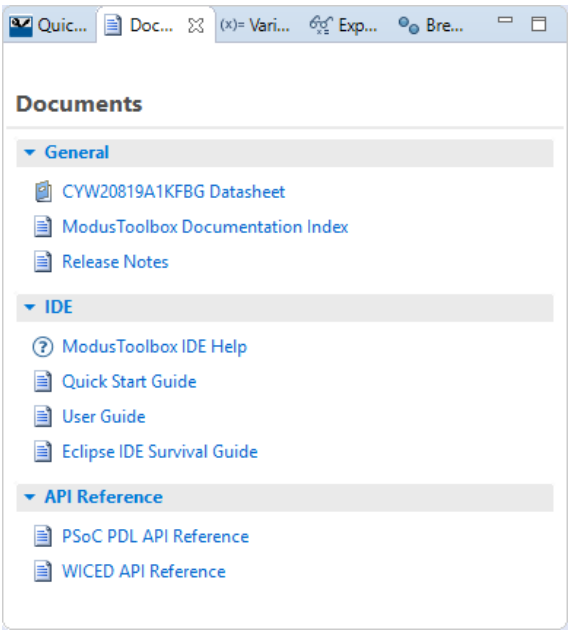


5. Click Finish

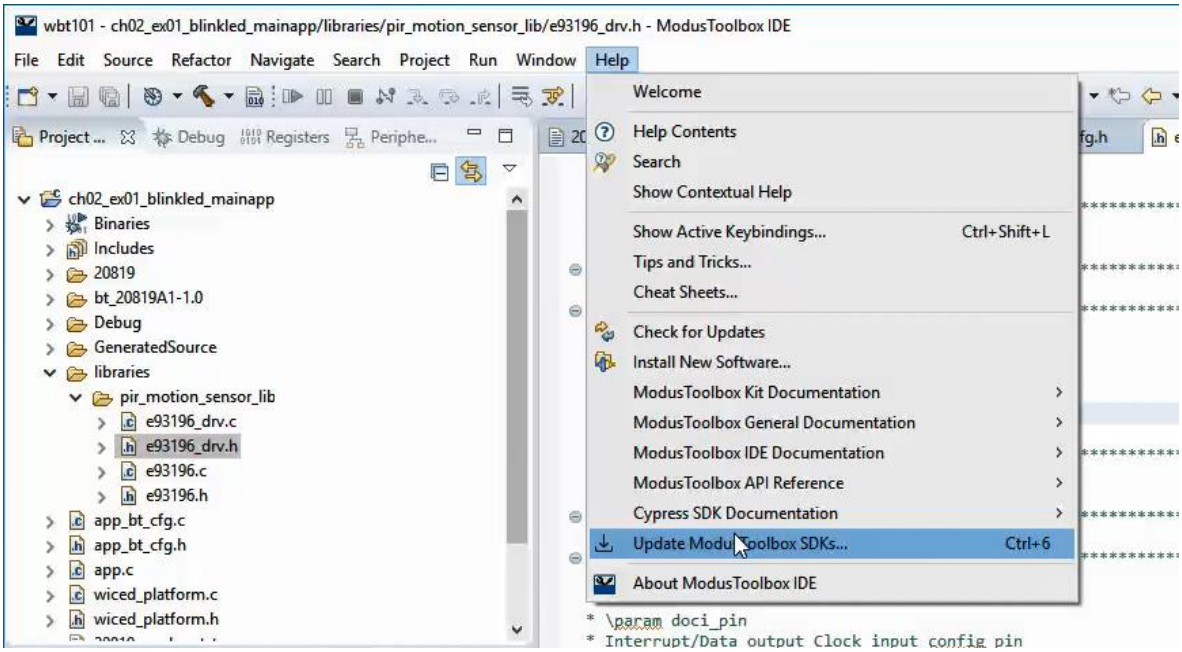
1.2 Tour of Documentation

1.2.1 In ModusToolbox IDE

Next to the Quick Panel tab is a tab that contains links to documentation. In includes documentation for the selected device, documentation for the IDE, and API references.



The *Help* menu has most of those links as well as links to kit and SDK documentation. The Help menu also has an item to check for updates to the installed SDKs.



1.2.2 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):

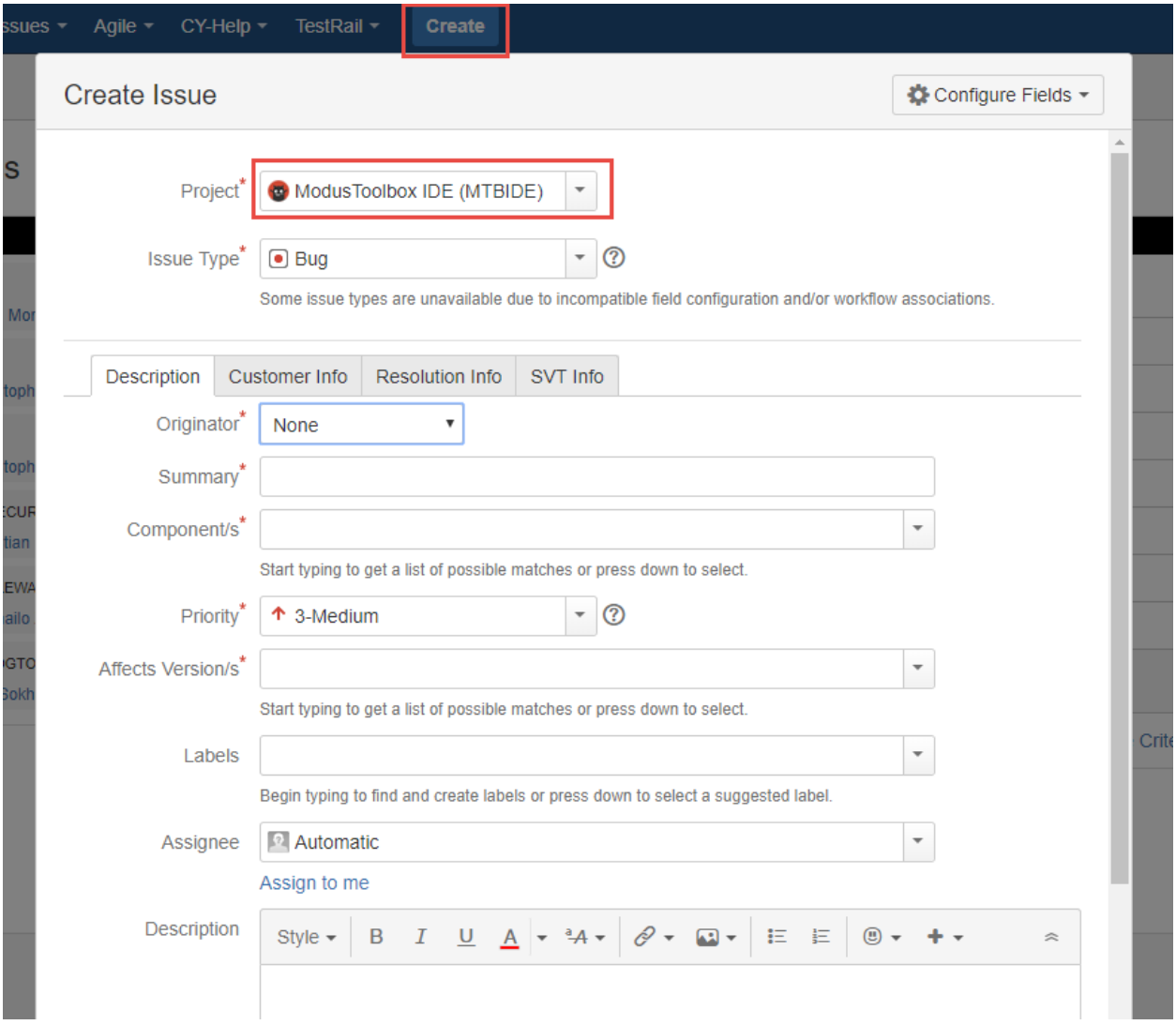
Clicking on the *Wireless* icon will take you to the page as shown below. From this page, you will find links to pages that allow you to download ModusToolbox, search for answers, ask questions, etc.

1.3 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 "ModustToolbox IDE" and then fill in as many details as you can to report the issue.



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

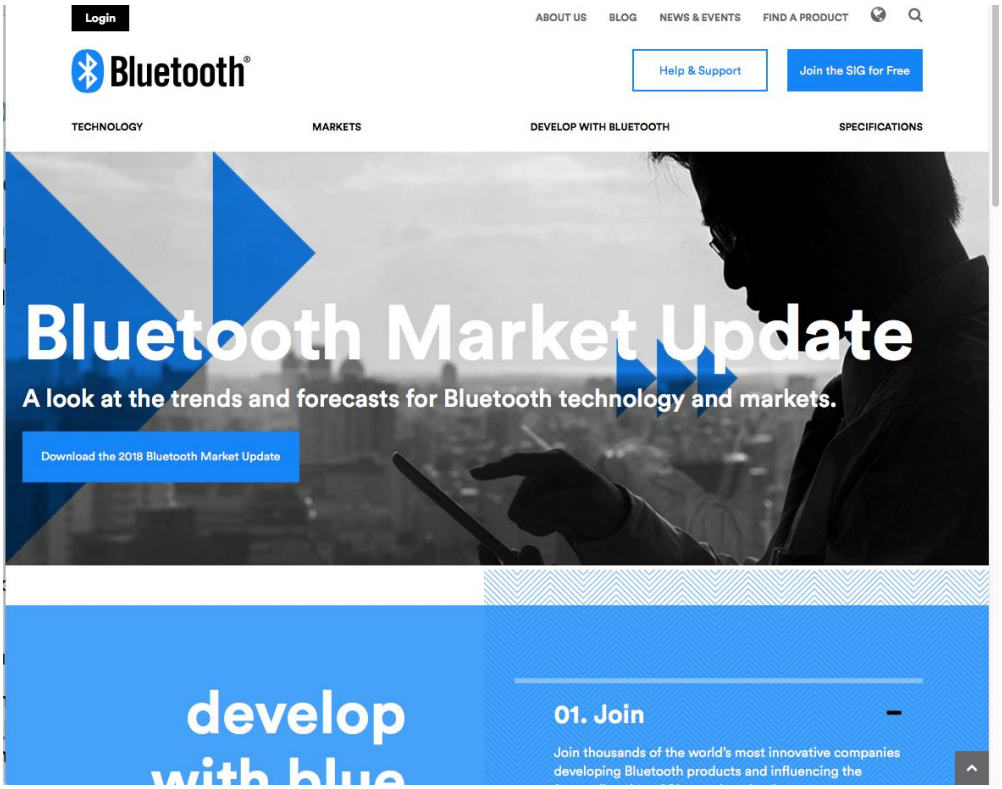
1.4 Tour of Bluetooth

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).

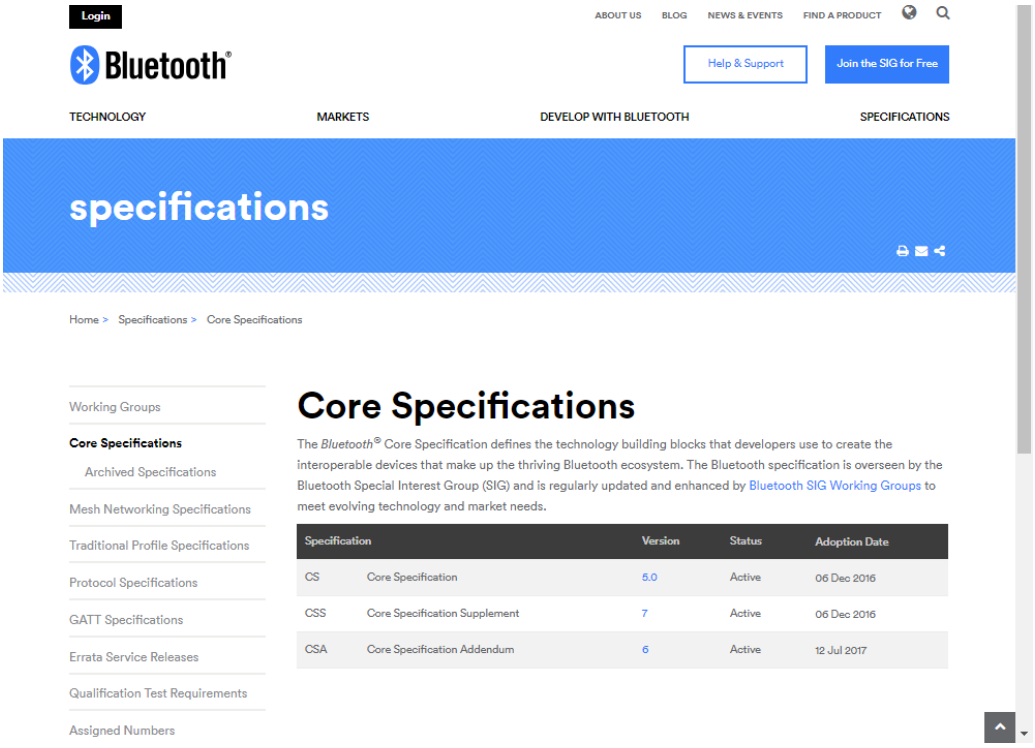
Discussions about Bluetooth are typically divided into *Classic Bluetooth* and *Bluetooth Low Energy*.

1.4.1 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.



The current Bluetooth Specification is Version 5.0 is a 2822 page long document that can be downloaded from the Bluetooth SIG website at https://www.bluetooth.com/specifications/bluetooth-core-specification



1.4.2 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 (m)
	(mW)	(dBm)	
1	100	20	100
2	2.5	4	10
3	1	0	1
4	0.5	-3	0.5

1.4.3 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.

Another name for Bluetooth Low Energy is "Bluetooth Smart". Devices that support both Classic Bluetooth and BLE are sometimes called "Bluetooth Smart Ready".

1.4.4 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.

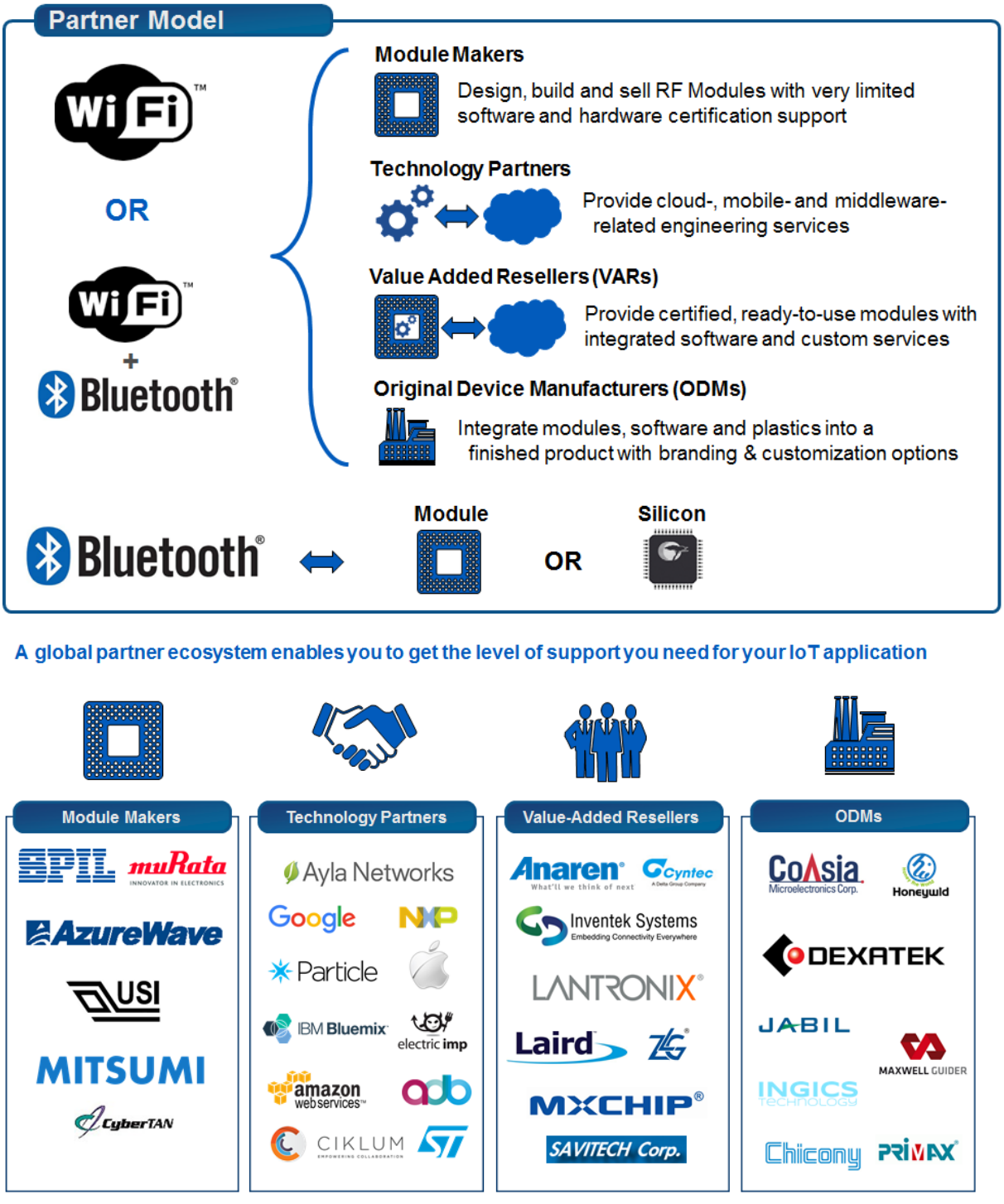
1.5 Tour of Chips

Device	Key Features	Notes
CYW20706	• Bluetooth BR, EDR and LE 5.0 • ARM Cortex-M3 • 848 kB ROM • 352 kB RAM (data and patches) • 2 kB NVRAM	
CYW20719	• Bluetooth BR, EDR and LE 5.0 • 2 Mbps LE v5 • 96 MHz ARM Cortex-M4 • Single Precision FPU • 2 MB ROM • 1 MB On-Chip Flash • 512 kB RAM	
CYW20819	• Bluetooth BR, EDR and LE 5.0 • 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	

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 5.0 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.

1.6 Tour of Partners



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

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

1.7 Tour of Development Kits

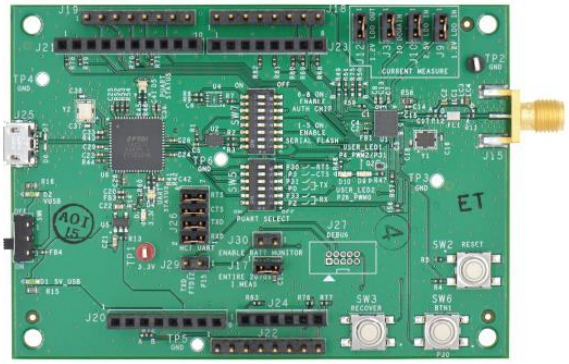
1.7.1 Cypress CYW920819EVB-02

- Bluetooth 5.0 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

1.7.2 Cypress CYBT-213043-MESH

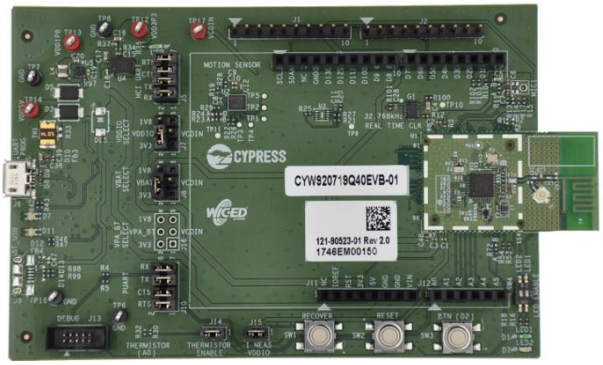
- Bluetooth Mesh kit with 20819 module
- Each kit contains 4 boards to evaluate mesh networks
- 1 User Button, RGB LED, ambient light sensor, PIR motion sensor

1.7.3 Cypress CYW920706WCDEVAL



- Monolithic, Single-chip, Bluetooth 5.0 + 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

1.7.4 Cypress CYW920719Q40EVB-01



- Bluetooth 5.0 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

1.8 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 ModusToolbox IDE and Explore the documentation

1. Open the ModusToolbox IDE.
2. Select the Documents tab in the lower-left panel.
3. Explore the different documents available such as the *ModusToolbox IDE Help*, *Quick Start Guide, User Guide*, *Eclipse IDE Survival Guide* and *WICED API Reference*

Questions to answer:

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

Exercise - 1.3 Download the Bluetooth Spec Version 5.0

The spec can be found at: www.bluetooth.org

Questions to Answer:

1. What is the organization scheme used in the spec?

2. On what page does the Attribute protocol specification start?