Chapter 1: Introduction

After completing this chapter, you will understand what this class is, what topics are covered, and the overall class objectives. You will be introduced to the ModusToolbox™ development ecosystem and will learn how to find documentation online.

Table of contents

Document conventions

Convention	Usage	Example
`Courier New`	Displays code and text commands	`CY_ISR_PROTO(MyISR);` `make build`
Italics	Displays file names and paths	*sourcefile.hex*
[**bracketed, bold**]	Displays keyboard commands in procedures	[**Enter**] or [**Ctrl**] [**C**]
Menu > Selection	Represents menu paths	**File > New Project > Clone**
Bold	Displays GUI commands, menu paths and selections, and icon names in procedures	Click the **Debugger** icon, and then click **Next**.

1.1 What is this class?

This class is a survey of the ModusToolbox™ development ecosystem. The learning objective is to introduce you to all the tools in the ModusToolbox™ ecosystem and help you develop some familiarity with using them. The class is "a mile wide and an inch deep." This should enable you to understand the scope of the development ecosystem and teach you where to find "everything."

This is a "Level 1" class, meaning that it is intended as an entry point to get you started with the ModusToolbox™ ecosystem. Level 2 classes dig deeper to give more detailed training on specific products such as PSoC™ 6 or XMC™ MCUs. Level 3 classes go even further by diving into a complete solution such as Bluetooth®, Wi-Fi, Motor Control, or Machine Learning.

1.2 What is the ModusToolbox™ ecosystem?

Before we talk about what the ModusToolbox™ ecosystem is, let's talk for a second about what it isn't. It isn't an IDE or a compiler. It isn't just software, and it isn't just a set of libraries or an SDK.

The ModusToolbox™ ecosystem is a collection of software and tools designed to work together and with third-party solutions such that you can create the development environment that works best for you. If you want to use the Eclipse IDE, that's fine. If you prefer to use Visual Studio Code (VS Code) instead, that's OK too. Or you can use IAR Embedded Workbench, the Arm® Keil® µVision® IDE, command-line operations, or something else. It's entirely up to you. The same is true for compilers, debuggers, revision control systems, etc. No matter what choices you make, the ModusToolbox™ ecosystem of software and tools work with you.

With the ModusToolbox™ ecosystem, you don't have to choose between closed, proprietary flows that struggle to keep pace with modern innovations and open platforms, or that fail to support the unique features and value of your products. The ModusToolbox™ ecosystem provides the "best of both worlds" platform that delivers a wonderful development experience, increased productivity, and feature-rich, bullet-proof applications.

The ModusToolbox™ ecosystem is divided into Tools and Software assets. Tools are installed as part of the ModusToolbox™ tools install package while Software assets are downloaded as needed for an application.

1.2.1 Tools

Tools refer to programs and services that run on the developer's host computer or in the cloud. For example:

- Eclipse IDE for ModusToolbox™
- Compilers (GCC, Arm®)
- Build System (make, Cygwin)
- Programming and Debug Tools (OpenOCD, PyOCD)
- Configurators and Tuners
- Project Creator
- Library Manager
- Firmware Loader

1.2.2 Software assets

Software assets refers to code that executes on the target device. This includes:

- BSP (Board Support Package)
- CSP (Chip Support Package) integrated into the BSP
- Libraries (e.g., RTOS, Network Stacks, Graphics, etc.)
- Customer or Code Example Application Firmware (i.e. their project or our code example)

1.3 Terminology

The following are definitions for terms that are used throughout the ModusToolbox™ ecosystem and this class. Some of these terms will be covered in more detail later on in this class.

Term	Description
Project	A directory that must contain a Makefile and one or more source files. The Makefile must be built such that it references a board support package (BSP) and can be built using a ModusToolbox™- or BSP-supplied make recipe. The result of the build process is an ELF file.
Application	A directory that contains one or more ModusToolbox™ projects that are meant to build and work together targeting one or more devices in a system. For example, a dual core application may contain two projects; one for each core.
Workspace	A directory that contains one or more ModusToolbox™ Applications and the common *mtb_shared* ModusToolbox™ asset repository.
Asset	A resource that is referenced by a ModusToolbox™ Project that is retrieved using the ModusToolbox™ software `make getlibs` mechanism, which means it is retrieved using a git clone operation.
Board Support Package (BSP)	A ModusToolbox™ asset that supports the target device and board. It provides support to boot a typical application such as startup and linker scripts, provides aliases such as pin names to support cross board common resources, and provides default configurations for some pins, clocks, and peripherals. The customer will typically evolve a standard BSP to support their own board.
Code example	A resource that acts as a starting point for a ModusToolbox™ application. The resource acts as a template and can support one or more BSPs. The template is used to initialize the ModusToolbox™ application directory using a git clone operation during project creation.
Tool	A host platform (Windows, MacOS, Linux) program that is installed by the ModusToolbox™ tools installer and performs a specific function in the ModusToolbox™ development environment.
Tools package	An installer that provides the set of ModusToolbox™ tools that are required to enable the ModusToolbox™ development environment.
Peripheral Driver Library (PDL)	A ModusToolbox™ asset that contains source code that provides low-level support for the hardware found in a given target device. This library should support all of the features of all of the hardware in a given device. There may be some exceptions based on the business needs for a given device.
Hardware Abstraction Layer (HAL)	A ModusToolbox™ asset that contains source code that is an implementation of a standard API that is common across all devices that support the HAL. The intent is to enable code to be written that is portable to all devices that support the HAL.

Term	Description
Configurator	A ModusToolbox™ tool that provides a graphical interface to ease the process of configuring a target device or middleware asset. Configurators generate data structures that are consumed by the build process.
BSP configurator	Hardware configurators that interact directly (via PDL or HAL) with the hardware of the device. Files from these configurators are typically provided by the BSP.
Library configurator	Hardware configurators that are coupled with a firmware library. These configurators generate data structures that are consumed by its associated firmware library. Examples of these include the USB configurator and the Bluetooth® configurator.
Device configurator	A ModusToolbox™ BSP configurator that provides a graphical interface to configure the basic configuration of the target device including clocks, pins, and various hardware IP blocks. Its information is stored in the file *design.modus*.

1.4 Supported products

The ModusToolbox™ ecosystem supports many Infineon product families, and more are being added all the time. In fact, by the time you read this, there may already be more products supported than is shown here.

Device	Description
PSoC™ 6 MCU	These are 32-bit dual-core Arm® Cortex®-M4 and Cortex®-M0+ MCUs. They include a rich set of digital and analog peripherals including out industry-leading CapSense™ capacitive-sensing technology. These devices can be paired with Infineon's AIROC™ Wi-Fi, AIROC™ Bluetooth®, or AIROC™ combo radio modules to create secure, low-power, feature-rich IoT products.
PSoC™ 4 MCU	These are 32-bit Arm® Cortex®-M0 and Cortex®-M0+ microcontrollers. As with PSoC™ 6 MCUs, most devices include CapSense capacitive-sensing technology as well as many other digital and analog peripherals.
XMC™ MCU	These are 32-bit industrial microcontrollers including Arm® Cortex®-M0 or Cortex®-M4F CPUs. Some devices include up to 6 cores for large scale industrial systems.
PMG1 USB-C Devices	This is a family of high-voltage microcontrollers with USB-C power delivery. The devices include an Arm® Cortex®-M0/M0+ CPU based on the PSoC™ 4 MCU architecture and a USB-C PD controller along with analog and digital peripherals.
AIROC™ Bluetooth®	This is a family of fully compliant Bluetooth® devices that support basic rate (BR), extended data rate (EDR), and low energy (LE).

1.5 Supported IDEs and reference flows

As described earlier, the ModusToolbox™ ecosystem isn't an IDE. Rather, it supports multiple IDEs so that you can work the way you want. There are four IDEs that are supported explicitly by the ModusToolbox™ ecosystem, but others can also be used by starting from the command line and adding the necessary IDE support.

The IDEs that are supported by the ModusToolbox™ ecosystem out of the box are the Eclipse IDE for ModusToolbox™, Microsoft Visual Studio Code (VS Code), IAR Embedded Workbench, and Arm® Keil® µVision®. Each of these can be selected as the target IDE during application creation which will create the files needed for the chosen IDE. Files for IDEs can also be created from the command line for an existing application.

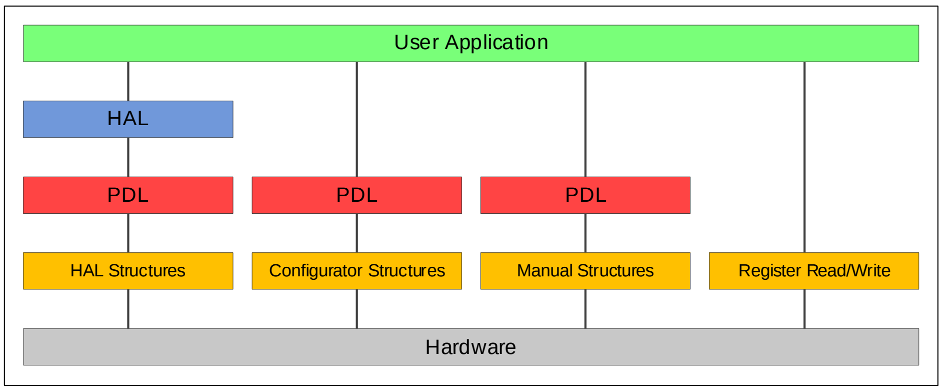
IDE	Description
Eclipse IDE for ModusToolbox™	This selection creates Eclipse project files *.project* and *.cproject* and a directory containing launch configurations called *.mtbLaunchConfigs* to allow easy program and debug operations. These files are intended to be used with the customization of Eclipse that is installed by the ModusToolbox™ tools installer.
VS Code	This selection generates json files in the *.vscode* directory containing projects settings, launch configurations, and task configurations to allow easy program and debug operations. It also creates VS Code workspace file called *<app_name>.code-workspace* that allows the application and its associated libraries to be viewed together
IAR Embedded Workbench	This selection generates the file *<app_name>.ipcf* that is used to open/create the project inside IAR Embedded workbench.
Arm® Keil® µVision®	This selection generates the files *<app_name>.cpdsc*, *<app_name>.gpdsc*, and *<app_name>.cprj* that are necessary for µVision.

Note: *The Eclipse IDE for ModusToolbox™ is a customization of Eclipse that includes some additional menus and windows, as you will see later. You can use standard Eclipse, but those features will not be available.*

1.6 Application layers

Many devices supported by the ModusToolbox™ ecosystem provide methods with different levels of abstraction to interact with the hardware. These range from direct register read/write with complete control to higher level abstraction layers that simplify the interface and provide portability between devices and families.

The description below is specifically for PSoC™ 6 MCUs, but other product families use the same or similar concepts. There are four distinct ways for a PSoC™ 6 MCU application to interact with the hardware as shown in the following diagram:



- **HAL Structures**: Application code uses the HAL, which interacts with the PDL through structures created by the HAL.
- **Configurator Structures**: Application code uses PDL via structures created by a configurator.
- **Manual Structures**: Application code uses PDL through structures created manually.
- **Register Read/Write**: Application code uses direct register read and writes.

Note: *A single application may use different methods for different peripherals.*

1.6.1 HAL

Using the HAL is more portable than the other methods. It is the preferred method for simpler functions and those that don't have extremely strict flash size limitations. It is a high-level interface to the hardware that allows many common functions to be done quickly and easily. This allows the same code to be used even if there are changes to pin assignments, different devices in the same family, or even to a different family that may have radically different underlying architectures.

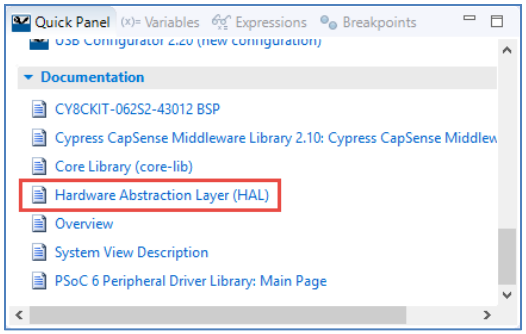
The advantages include:

- Easy hardware changes. Just change the pin assignment in the BSP and the code remains the same. For example, if LED1 changes from P0_0 to P0_1, the code remains the same as long as the code uses the name LED1 with the HAL. The only change is to the BSP pin assignment.
- Easy migration to a different device if there are changes to product requirements.
- Ability to use the same code base across multiple projects and generations, even if underlying architectures are different.

The disadvantages include:

- The HAL may not support every feature that the hardware has. It supports the most common features but not all of them to maintain simplicity.
- The HAL will use additional flash space. The additional flash depends on which HAL APIs are used.

After creating a PSoC™ 6 MCU project, there is a link to the HAL documentation in the Quick Panel under "Documentation."



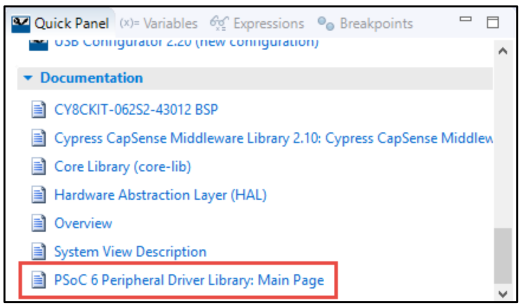
Note: *Many of the libraries provided by Infineon take advantage of the HAL so that they can work seamlessly with a wide variety of Infineon products.*

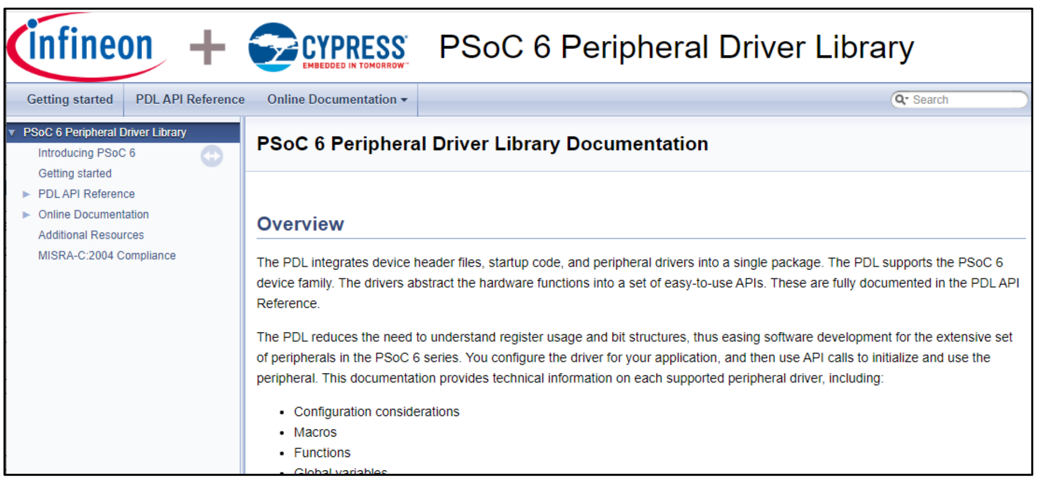
1.6.2 PDL

The PDL is a lower-level interface to the hardware (but still simpler than direct register access) that supports all hardware features. Usually the PDL goes hand-in-hand with configurators. Since the PDL interacts with the hardware at a lower level it is less portable between devices, especially those with different architectures.

The advantages/disadvantages are the exact opposite of those for the HAL. The main advantage is that it provides access to every hardware feature.

After creating a PSoC™ 6 MCU project, there is a link to the PDL documentation in the Quick Panel under "Documentation."





1.6.3 Configurators

Configurators make initial setup easier for hardware accessed using the PDL. The configurators create structures that the PDL requires without you needing to know the exact composition of each structure, and they create the proper structure based on your selections. Configurators will be discussed in more detail in later chapters.

If you use the HAL for a peripheral, it will create the necessary structures for you, so you should NOT use a configurator to set them up. The HAL structure is accessible, and once you initialize a peripheral with the HAL you can view and even modify that structure (that is, a HAL object). Keep in mind that the underlying structures are hardware-specific, so you may be sacrificing portability if you modify the structure manually. There are a few exceptions. For example, it is reasonable to configure system items (such as clocks) and use them with the HAL.

1.7 Installation

As you learned earlier, the ModusToolbox™ ecosystem consists of tools and software assets. The tools are installed onto your computer using an install package while software assets are downloaded as needed from GitHub.

Among other tools, the ModusToolbox™ tools install package includes the Eclipse IDE for ModusToolbox™. If you want to use a different IDE such as VS Code, IAR Embedded Workbench, or Arm® Keil® µVision®, you will need to install that separately.

Python is required for ModusToolbox™ tools. For Windows, the ModusToolbox™ tools installer includes the required version of Python. This will be installed alongside the other ModusToolbox™ tools so that it will not interfere with any other versions that you may have installed.

For MacOS and Linux, Python is often pre-installed on the system. If it isn't, you will need to install it.

Note: *Python 3.7.7 has been tested with the class material. Newer versions may work but they may not have been tested.*

For this class, we will install the ModusToolbox™ tools, Python (if necessary), and VS Code. Instructions are provided in the first exercise.

Note: *If you install ModusToolbox™ in a non-default location, once the installation completes you must set the CY_TOOLS_PATHS environment variable as described in the ModusToolbox™ Installation Guide. See the section titled "Installing in non-default location".*

Note: *One of the key components of ModusToolbox™ is the make system which does not support spaces in file or pathnames. Therefore, you must install ModusToolbox™ in a location with no spaces in the path. If your home directory contains spaces, you must install in a different location such as C:\ModusToolbox. In that case, once the installation is done, you must create two additional directories and to set four environment variables as described in the ModusToolbox™ Installation Guide. See the section titled "Installing with spaces in user home directory".*

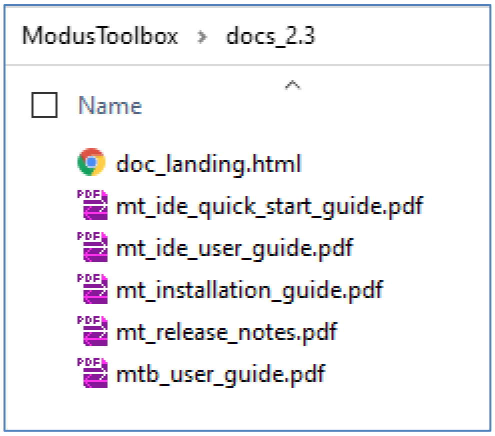
1.8 Documentation

There is a lot of documentation for the ModusToolbox™ ecosystem including tool documentation, asset documentation, code example documentation, and developer community posts. On top of that, our devices and solutions have their own documentation as well. All of it is available online and some is available offline as well. A brief summary of some of the documentation that's out there is provided below.

1.8.1 Tool documentation

The tool documentation is provided as part of the installation. The files can be found in <ModusToolbox™ Installation Directory>/ModusToolbox/docs_<version>. The installation directory defaults to the user's home directory.

The documentation includes an HTML file with links to online versions of all tool documents as well as local copies of the release notes, installation guide and user guide. It also has a quick start guide and user guide for the customized Eclipse IDE for ModusToolbox™. For example:



All of these documents are available from the **Help** menu inside the Eclipse IDE for ModusToolbox™ under **ModusToolbox™ General Documentation** and **Eclipse IDE for ModusToolbox™ Documentation**.

From the web, you can go to https://www.infineon.com/cms/en/design-support/tools/sdk/modustoolbox-software/ and look at the **Documentation** section.

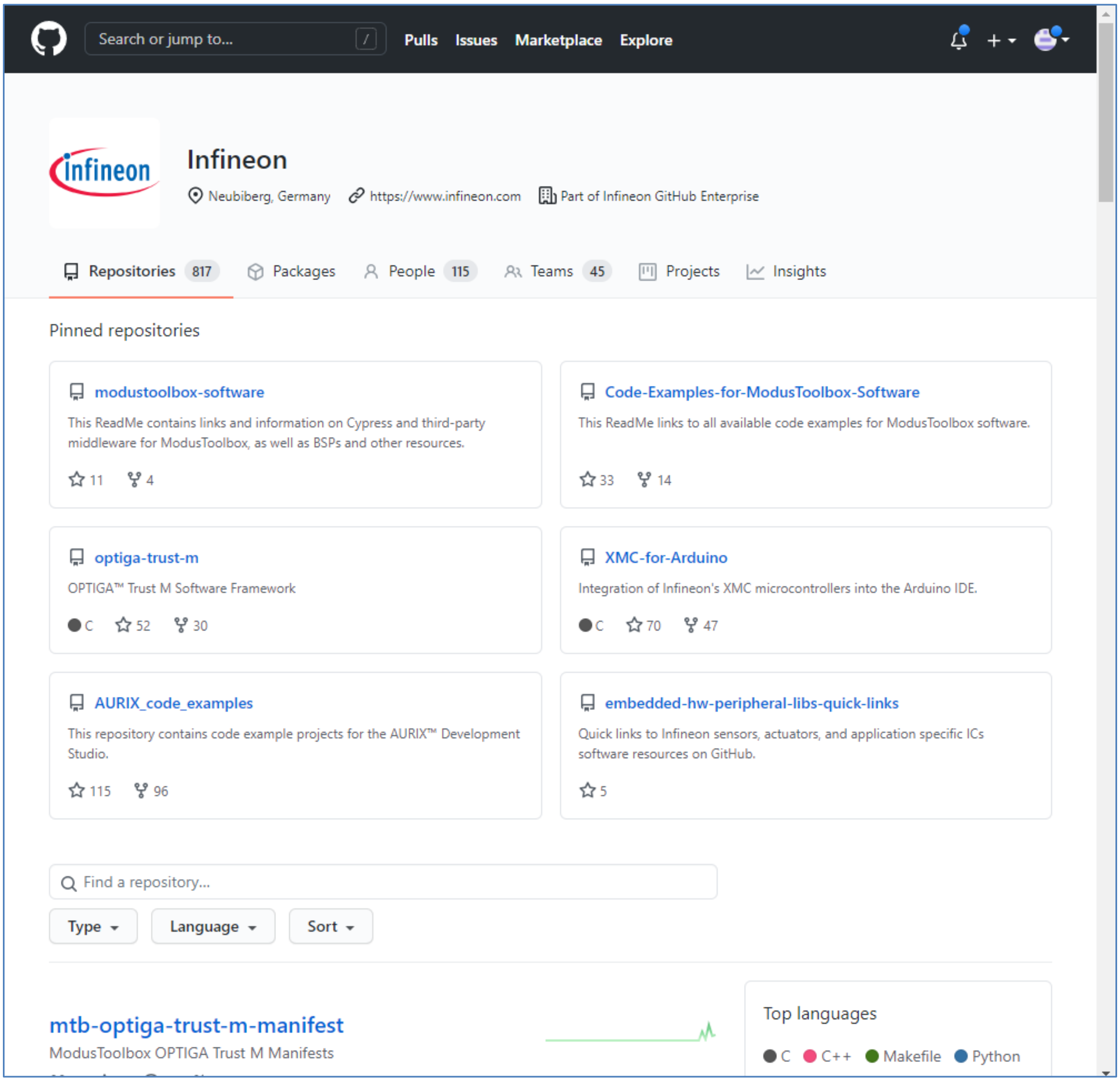
1.8.2 Asset documentation

Assets such as BSPs, libraries and code examples each have their own documentation. Each one has a *README.md* file that can be viewed directly on GitHub or locally once the asset is downloaded using a Markdown viewer or from the Eclipse IDE for ModusToolbox™.

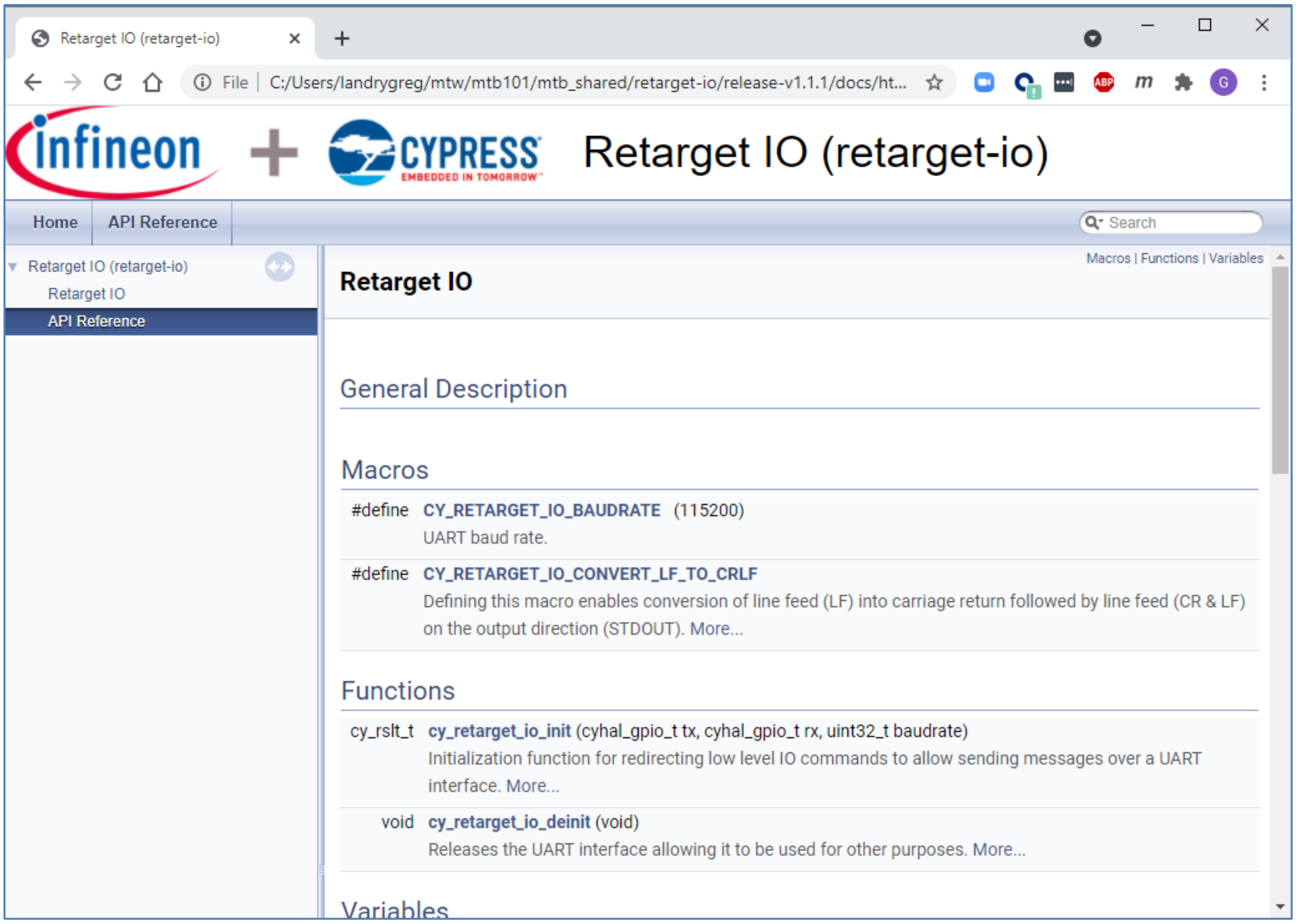
The main GitHub site is:

https://github.com/Infineon

From there you can search for repos by name or you can select one of the links to filter out code examples, or other software assets.



Most assets also have a directory named *docs* containing an HTML file with the API reference and other general information. For example, the retarget-io library has the following HTML file:



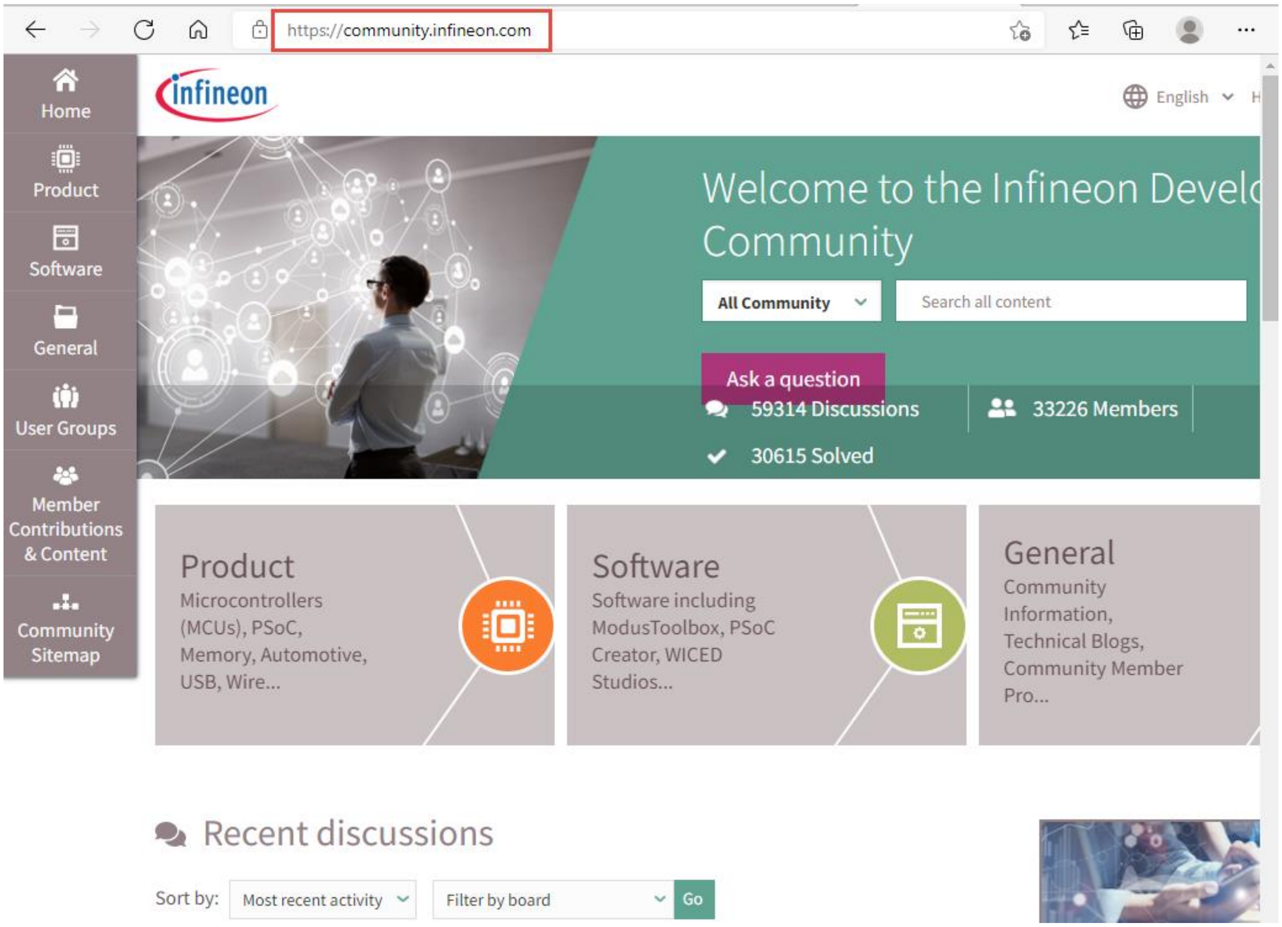
Note: *If you want to view the HTML file directly from GitHub, you can use the following URL where <asset_name> is replaced with the name of the asset (assuming the documentation is in docs/html):*

https://infineon.github.io/<asset_name>/html/index.html

The HTML files associated with the assets that are part of an application are listed in the Quick Panel in the Eclipse IDE for ModusToolbox™ when that application is selected. The Quick Panel links can be used to launch the HTML files in your default browser directly from inside the Eclipse IDE for ModusToolbox™.

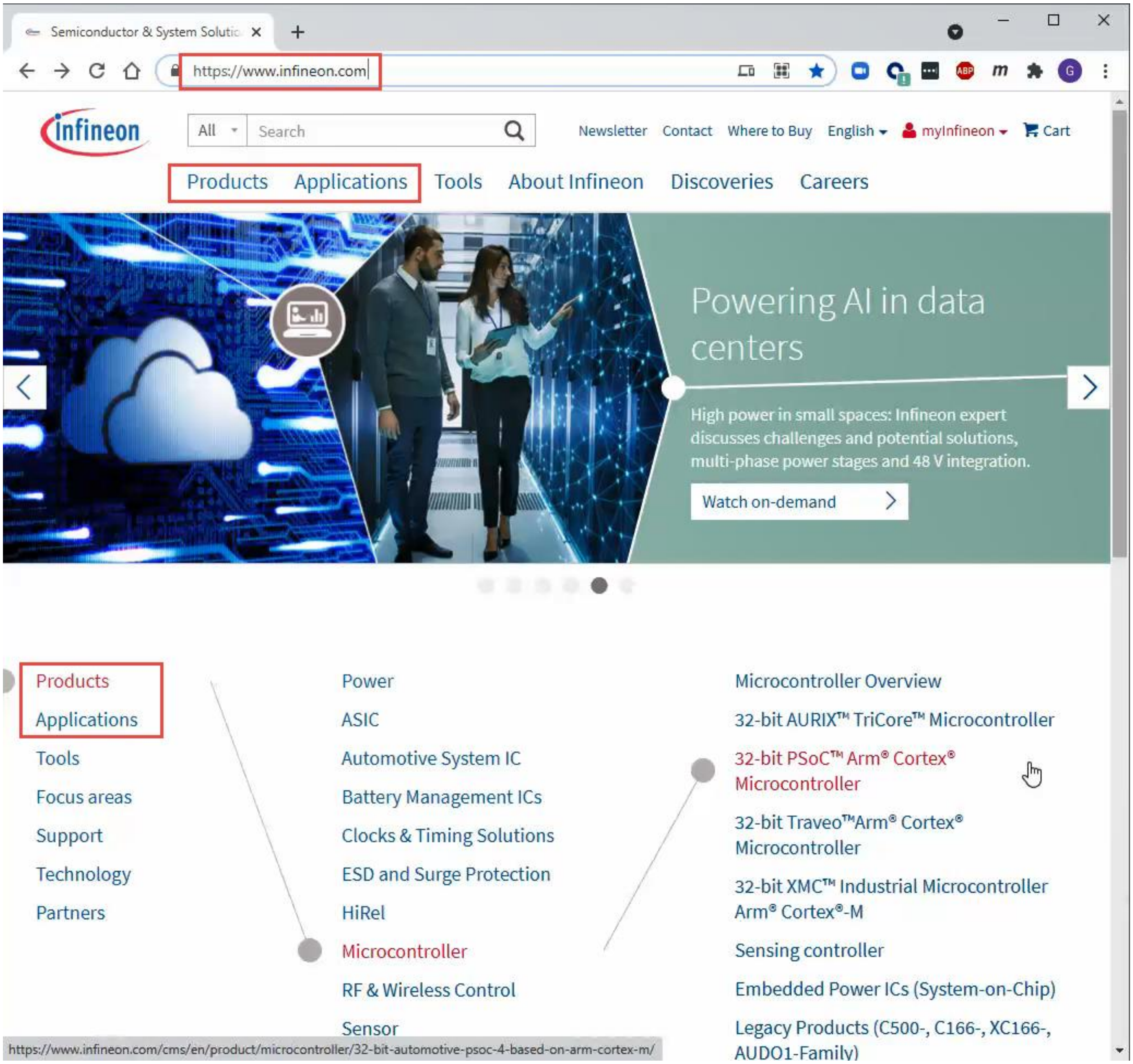
1.8.3 Developer community

There is an active community that can be found at https://community.infineon.com. Once you have an account (which is free), you can review existing posts and start your own discussions. Infineon has applications engineers who follow the community to answer your questions.



1.8.4 Device and solution documentation

Device datasheets and solution documentation can be found on the web. Start at https://www.infineon.com and use either the **Products** or **Applications** tab to find what you are looking for.



1.9 Exercises

Exercise 1: Install the software

In this exercise, you will install the ModusToolbox™ tools, Python (if necessary), and VS Code.

Python

☐ 1. **Windows**: Skip this step. Python will be installed along with the other ModusToolbox™ tools.

☐ 2. **MacOS**:

   a. Open a command terminal and type the following command:
   ```
   python --version
   ```
   b. If it returns Python 3.7.7 or later, skip to section **Error! Reference source not found.** to install the ModusToolbox™ tools.
   c. If you need to install Python, you will need to use Homebrew. If you don't have that, can get it from https://brew.sh.
   d. Once Homebrew is installed, just run the following from a command terminal:
   ```
   brew install python3
   ```

☐ 3. **Linux**:

   a. Open a command terminal and type the following command:
   ```
   python --version
   ```
   b. If it returns Python 3.7.7 or later, skip to section **Error! Reference source not found.** to install the ModusToolbox™ tools.
   c. If you need to install Python, use your distribution's package manager. For example, on Ubuntu the command is:
   ```
   sudo apt install python3
   ```

ModusToolbox™ tools

☐ 1. Download the latest ModusToolbox™ tools installation package from this website: https://www.infineon.com/cms/en/design-support/tools/sdk/modustoolbox-software/

Note: Once you reach the Infineon Developer Center, it is easiest to use the "Download" button to download ModusToolbox™. If you use the "Install" button you will need to login to or register for an Infineon account and then download/install the Infineon Developer Center Launcher before installing ModusToolbox™

☐ 2. Go to Documents > Getting Started and open the ModusToolbox™ Installation Guide. Follow the instructions for your operating system.

Note: For Windows, be sure to run the installer with administrator privileges so that the necessary pre-requisites and drivers can be installed.

Note: If you install ModusToolbox™ in a non-default location, once the installation completes you must set the CY_TOOLS_PATHS environment variable as described in the ModusToolbox™ Installation Guide. See the section titled "Installing in non-default location".

Note: One of the key components of ModusToolbox™ is the make system which does not support spaces in file or pathnames. Therefore, if your home directory contains spaces you must install in a different location such as C:\ModusToolbox. In that case, once the installation is done, you must create two additional directories and to set four environment variables as described in the ModusToolbox™ Installation Guide section titled "Installing with spaces in user home directory".

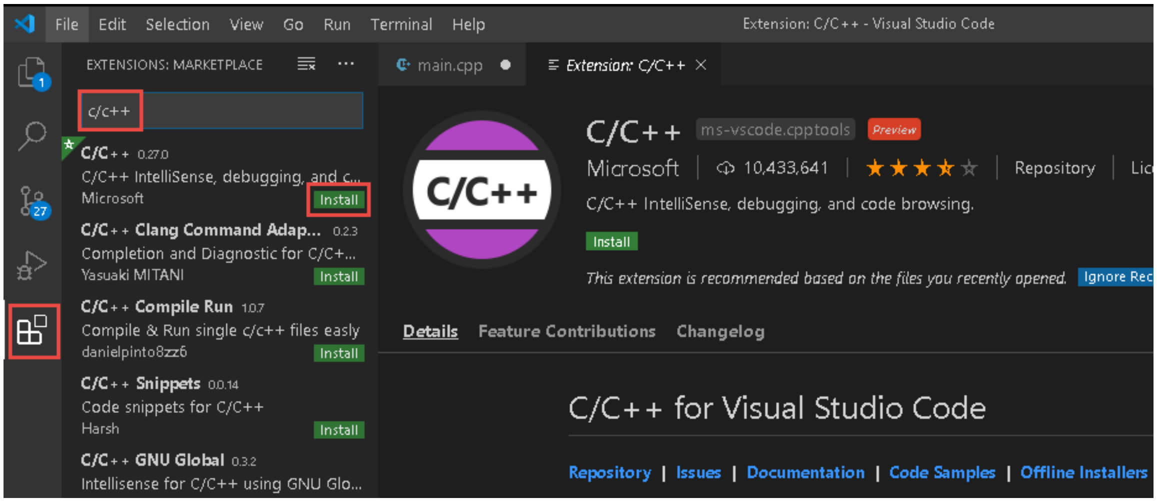
VS Code

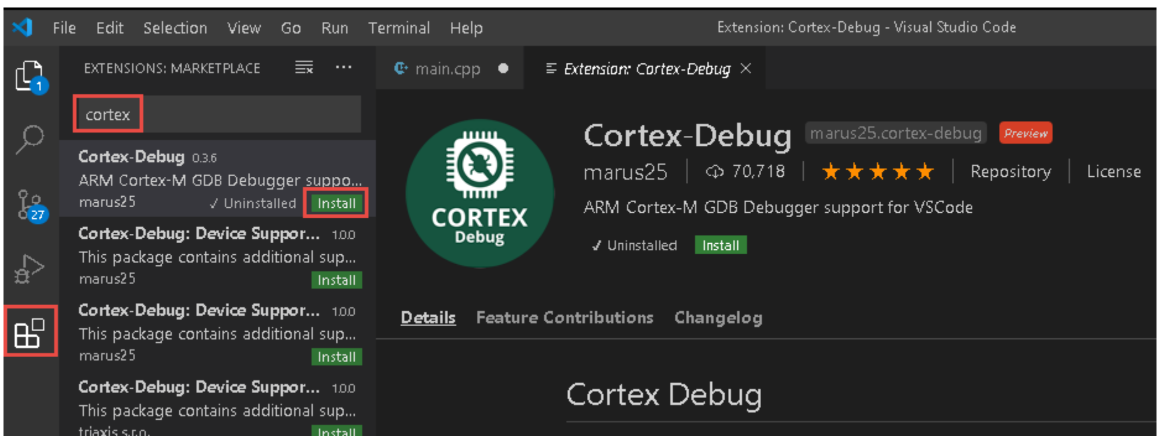
☐ 1. Download the software from this website:
https://code.visualstudio.com

☐ 2. Follow the instructions for your operating system at:
https://code.visualstudio.com/docs/setup/setup-overview

☐ 3. Once VS Code is installed, run it and install the C/C++ and Cortex® Debug extensions.

From the GUI, click the **Extensions** button and then search for `C/C++` and `Cortex Debug` one at a time and click the **Install** button. For example:





Exercise 2: Create a developer community account

The developer community is a very useful resource to get answers to just about any question related to the ModusToolbox™ ecosystem or Infineon devices. If you don't already have an account, you will create one now.

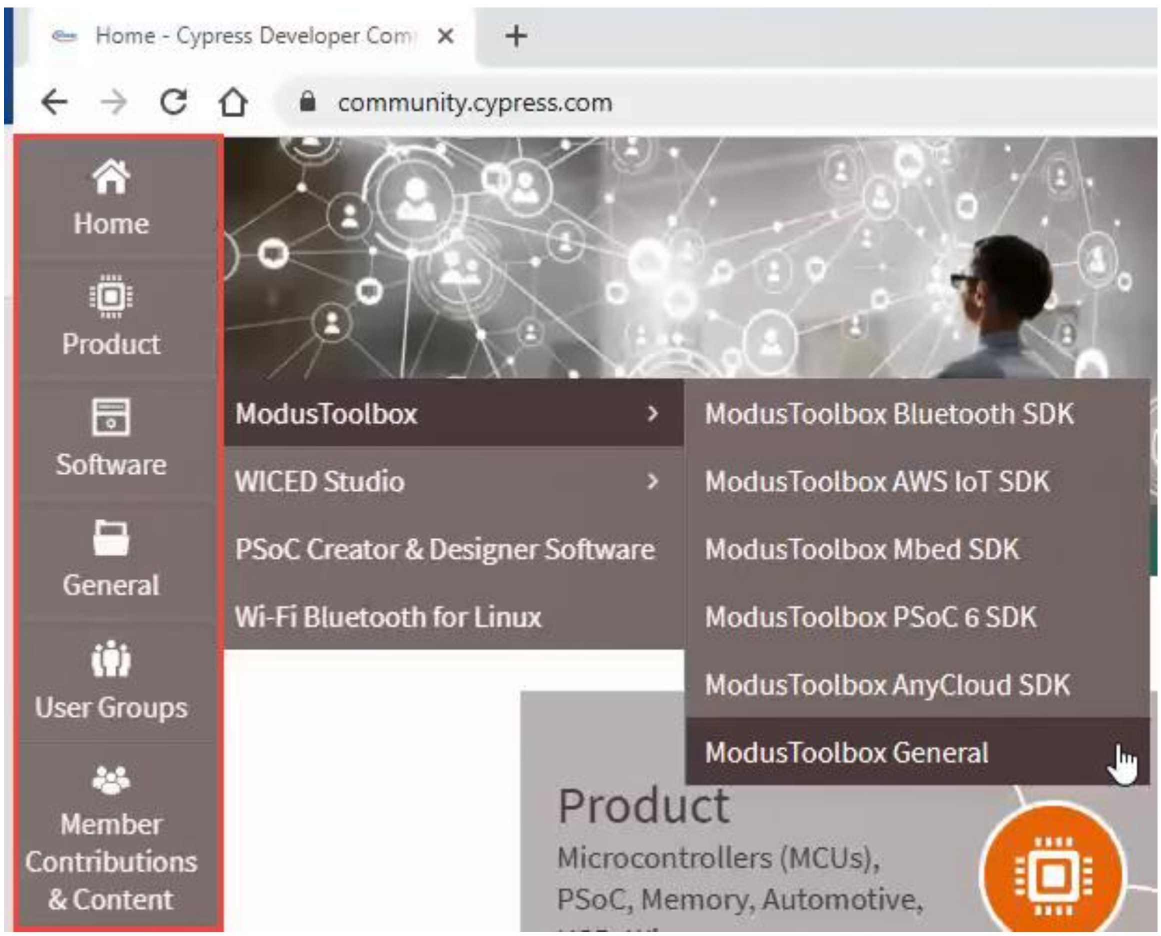
☐ 1. Open a web browser and go to https://community.infineon.com.

☐ 2. Click the Sign In button in the upper right corner.

☐ 3. If you have an account already, sign in. If not, register for a free account.

☐ 4. Once you are logged in, explore the discussions to find topics that interest you.

You can use the bar along the left side to help narrow your search to specific products, software tools, general topics such as blogs and knowledge base articles, and even member contributed code examples.



Exercise 3: Look at online documentation

In this exercise, you will become familiar with how to find online tool, asset, code example and product documentation.

☐ 1. Open a web browser and go to https://www.infineon.com/cms/en/design-support/tools/sdk/modustoolbox-software/.

☐ 2. Go to the **Documents** section.

☐ 3. Expand all and explore the list of documents.

Note: *Make sure you are logged in to your Infineon account so that you see all available documentation.*

☐ 4. Go to https://github.com/Infineon.

☐ 5. Click on the **modustoolbox-software** link followed by the **libraries** link.

☐ 6. Explore the list of libraries and look at a few of the *README.md* files.

☐ 7. Go to https://github.com/Infineon again.

☐ 8. Click on the **Code-Examples-for-ModusToolobox-Software** link followed by link of your choice such as PSoC™ 6 MCU, XMC™, PMG1, AnyCloud, etc.

☐ 9. Explore the list of code examples and look at a few of the *README.md* files.

☐ 10. Go to https://www.infineon.com.

☐ 11. Use the **Products** menu to explore documentation for a product.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Published by

Infineon Technologies AG

81726 Munich, Germany