AN0822: Simplicity Studio™ 4 User's Guide

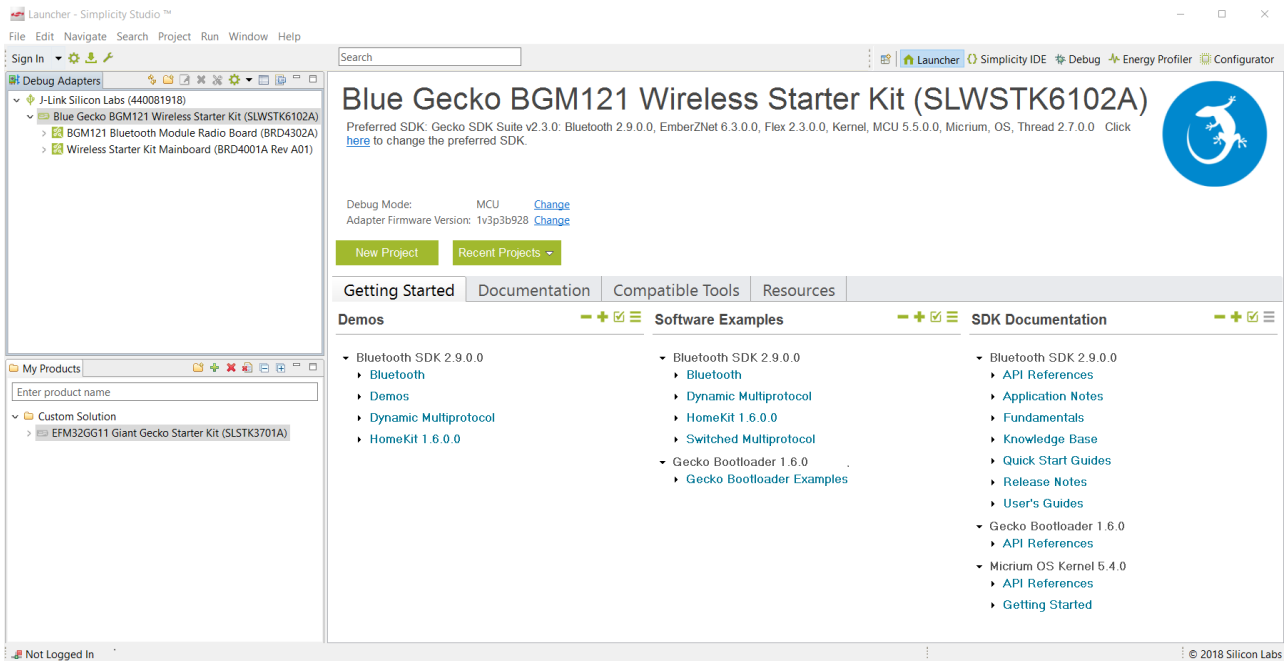
Simplicity Studio greatly reduces development time and complexity with Silicon Labs' EFM32, EFM8, and 8051 MCUs, wireless MCUs, and Zigbee SoCs.

Simplicity Studio can create wireless applications and provides hardware configuration, network analysis, real-time energy debugging, a high-powered IDE, and links to helpful resources, all in one place.

NOTE: This document applies to Simplicity Studio 4 and is no longer being maintained. If you are using Simplicity Studio 5 see the Simplicity Studio 5 User's Guide.

KEY POINTS

- Simplicity Studio makes the development process easier, faster, and more efficient.
- The IDE and integrated tools help optimize designs.
- Getting started with development is quick and easy with Demos and Software Examples.
- Quickly find help and design resources.



 Copyright © 2023 by Silicon Laboratories

1. Relevant Resources

- Simplicity IDE Guide—In the Simplicity Studio IDE, select [**Help**]>[**Help Contents**] to display this guide as well as any installed documentation.
- *AN0821: Simplicity Studio C8051F85x Walkthrough*—This document provides a step-by-step walkthrough for using the Simplicity Configurator and Simplicity IDE tools with the C8051F85x family.
- *AN0823: Simplicity Configurator User's Guide*—In addition to the documentation within the tool itself, this document provides a discussion of the Simplicity Configurator tool.
- Each MCU based product line has a quick start guide or other getting started type document. Reference 12. Where to Next? for the document list. In addition, the "Fundamental" series of documents covers topics that project managers, application designers, and developers should understand before beginning to work on an embedded networking solution using Silicon Labs chips, networking stacks such as EmberZNet PRO or Silicon Labs Bluetooth®, and associated development tools. The documents can be used as a starting place for anyone needing an introduction to developing wireless networking applications, or who is new to the Silicon Labs development environment. The documents can be found under the "Fundamentals" documentation category in Simplicity Studio or by searching for 'UG103' on www.silabs.com. The document numbers are all in the UG103.xx range (for example *UG103.1: Wireless Networking Application Development Fundamentals*).
- Energy Profiler *UG343: Multi-Node Energy Profiler User's Guide* - This document explans how to use the Energy Profiler to measure the power consumption of applications on one or more devices at the same time and how to use code correlation to figure out which sections of a program are consuming the most energy.
- Silicon Labs Videos—See the Simplicity Studio tools in action! Access these videos from the Silicon Labs website (http://www.silabs.com/simplicity-studio).
- Lizard Labs—More information on specific tools included in Simplicity Studio and their usage can be found at http://www.silabs.com/support/training/pages/online-training.aspx.

2. Overview

Simplicity Studio is a launching pad for everything needed to evaluate, configure, and develop with EFM32, EFM8, and 8051 MCUs, Wireless Geckos and MCUs, Zigbee SoCs, and Wireless Modules. The software gathers all of these tools into four categories: Getting Started, Documentation, Compatible Tools, and Resources.

The contents of these sections are contextualized for the selected device or kit so that only the relevant information and tools are shown.

2.1 Part Selection

Selecting a device will change the available tiles and the behavior of each tile in the launcher. The device can be selected by connecting a kit or board or by adding a part to the [**My Products**] window. Examples and documentation can also be found by searching for the device or kit.

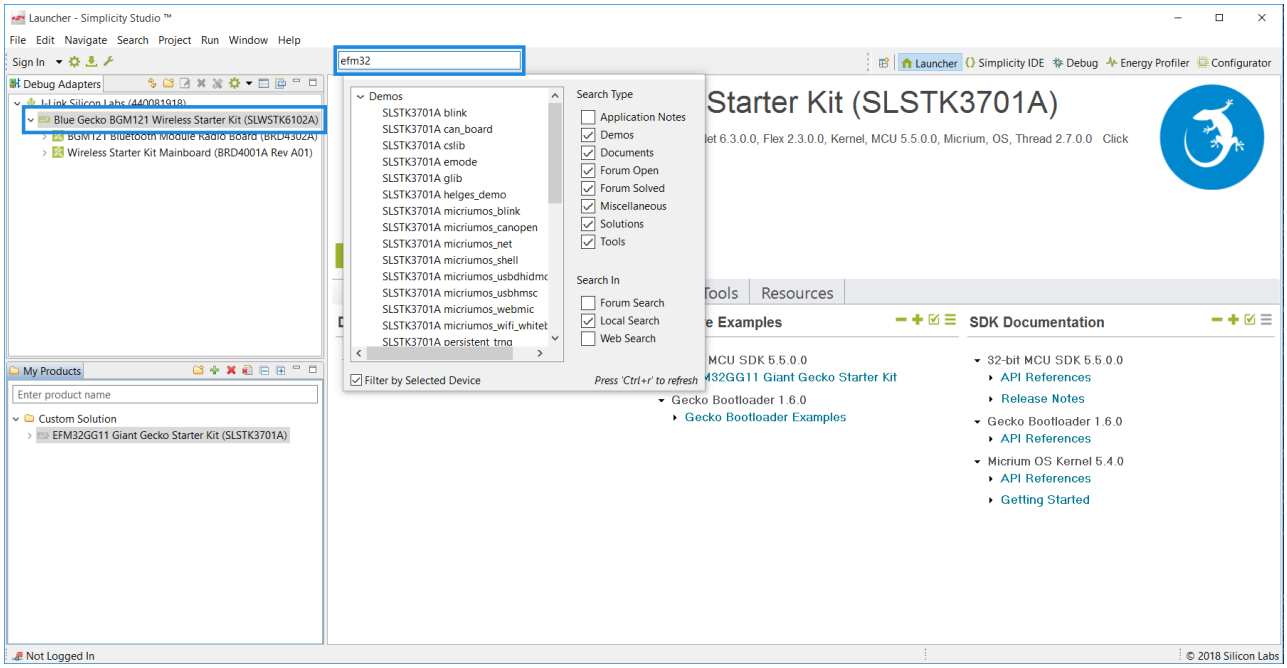


Figure 2.1. Searching for Parts

To automatically detect the appropriate device, connect the board to the PC. Simplicity Studio is notified a new device is present, and if it is a supported device, it will be added to the [**Debug Adapters**] window. Selecting the device from the list will update all of the options in the [**Getting Started**], [**Documentation**], [**Compatible Tools**], and [**Resources**] tabs of the launcher.

To view the available collateral without connecting a board, click the [**Enter product name**] box on the [**My Products**] tab, then start typing a product name and when the desired product appears in the drop down list click it. Any part of the product name or part number can be entered. Examples of what can be typed to start the drop down list are: EFM8, EFM32, EFR32, WSTK, STK, Blue, BRD.

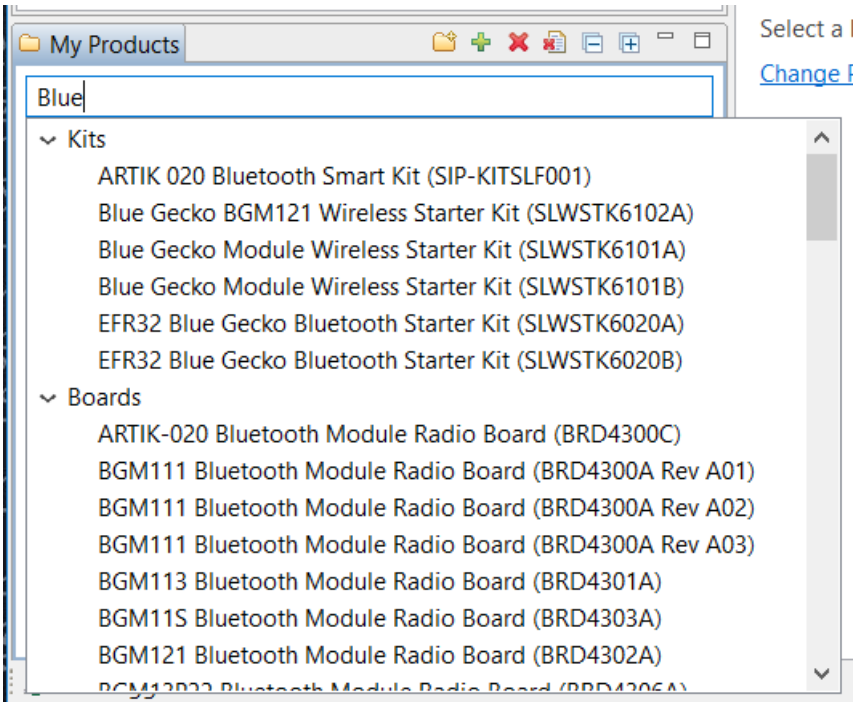


Figure 2.2. Adding to My Products

Clicking on a product from the drop down list will create an entry in the Custom Solution folder in the [**My Products**] area. The entry will automatically be highlighted and the relevant Demos, Software Examples and SDK Documentation will appear on the [**Getting Started**] tab in the main Launcher window.

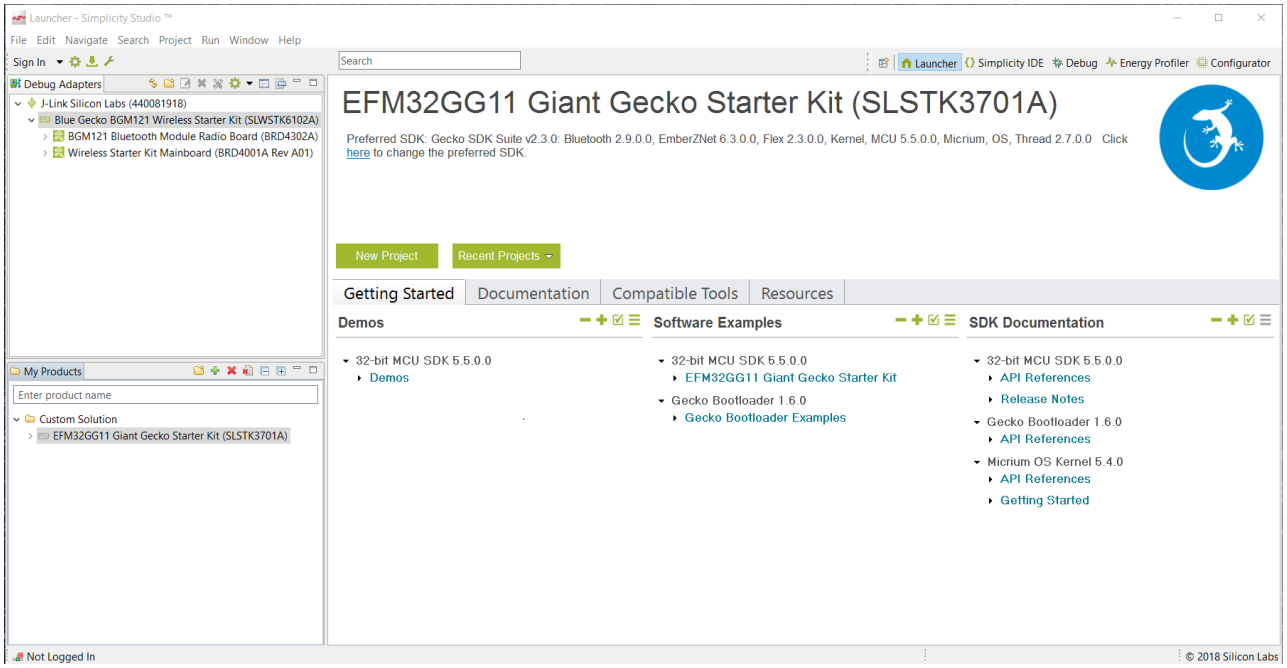


Figure 2.3. My Products Content

Various controls are available for the [**My Products**] tab to add [**Folders**] or [**Products**] and to remove a [**Single Product**] or [**All Products**] from the window.

Figure 2.4. My Products Controls

2.2 Getting Started

The [**Getting Started**] section provides demos, software examples, and application notes. The demos are pre-compiled examples for use with a Starter or Development Kit and they also allow the analysis tools such as the Energy Profiler and Network Analyzer to be used. These resources enable fast and easy development with supported Silicon Labs products.

2.3 Documentation

The [**Documentation**] section provides links to device and kit documentation. In addition, this section links to the available software documentation for the selected device. The documentation categories can be expanded or collapsed to change the scope of the presented documentation.

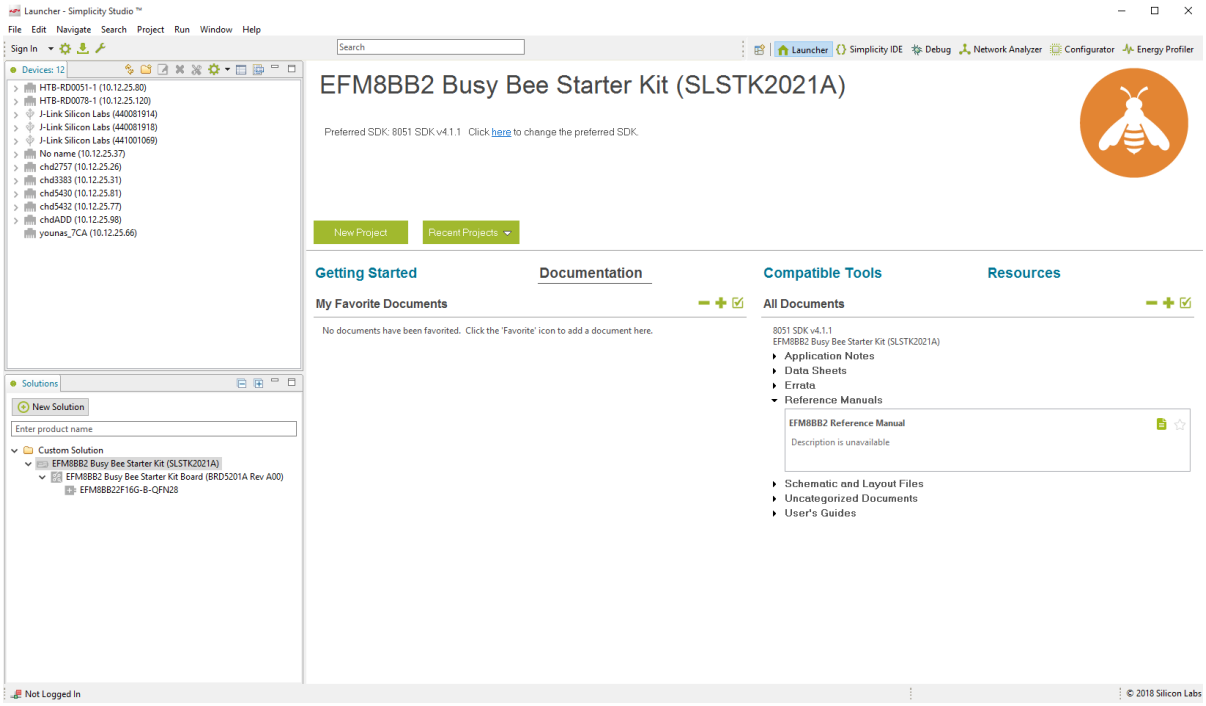


Figure 2.5. Documentation Tab

In addition the categories of documents to be presented can be customized by clicking checkmark icon to further refine the scope of the presented documents.

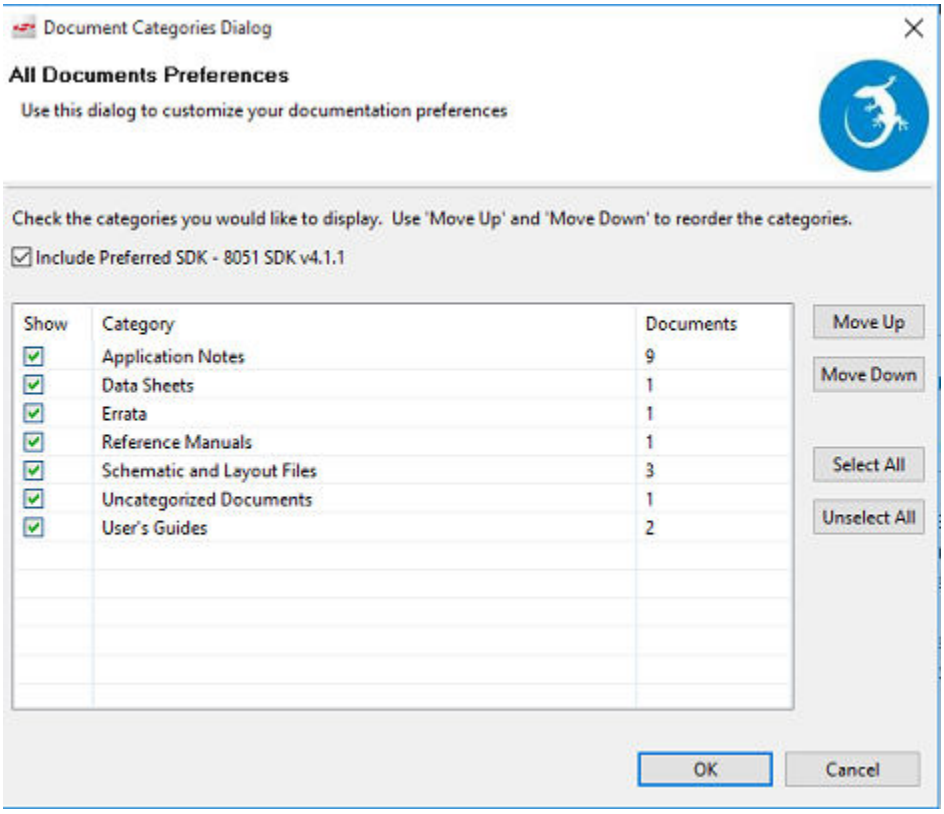


Figure 2.6. Documentation Preferences

2.4 Compatible Tools

The [**Compatible Tools**] section of Simplicity Studio provides software to develop code, configure or download code to a device, and debug the code on hardware. There are also tools to measure energy consumption in real-time, estimate a battery life, download demo code to a device, analyze a network of wireless devices, and configure wireless applications.

The default IDE selection is the Simplicity IDE, which is an Eclipse-based IDE provided with Simplicity Studio. To change the IDE preferences and use another IDE, click the [**Settings**] button and select [**Simplicity Studio**]>[**Preferred IDE**]. All IDEs detected on the system will be available as options.

2.5 Resources

The [**Resources**] section provides links to useful resources, like the Silicon Labs Community or Technical Support.

3. Simplicity IDE

The Simplicity IDE is an Eclipse-based Integrated Development Environment (IDE) enabling code editing, downloading, and debugging for Wireless (including EM35xx), Wireless Geckos, Wireless Modules, EFM32, EFM8, and 8051 devices.

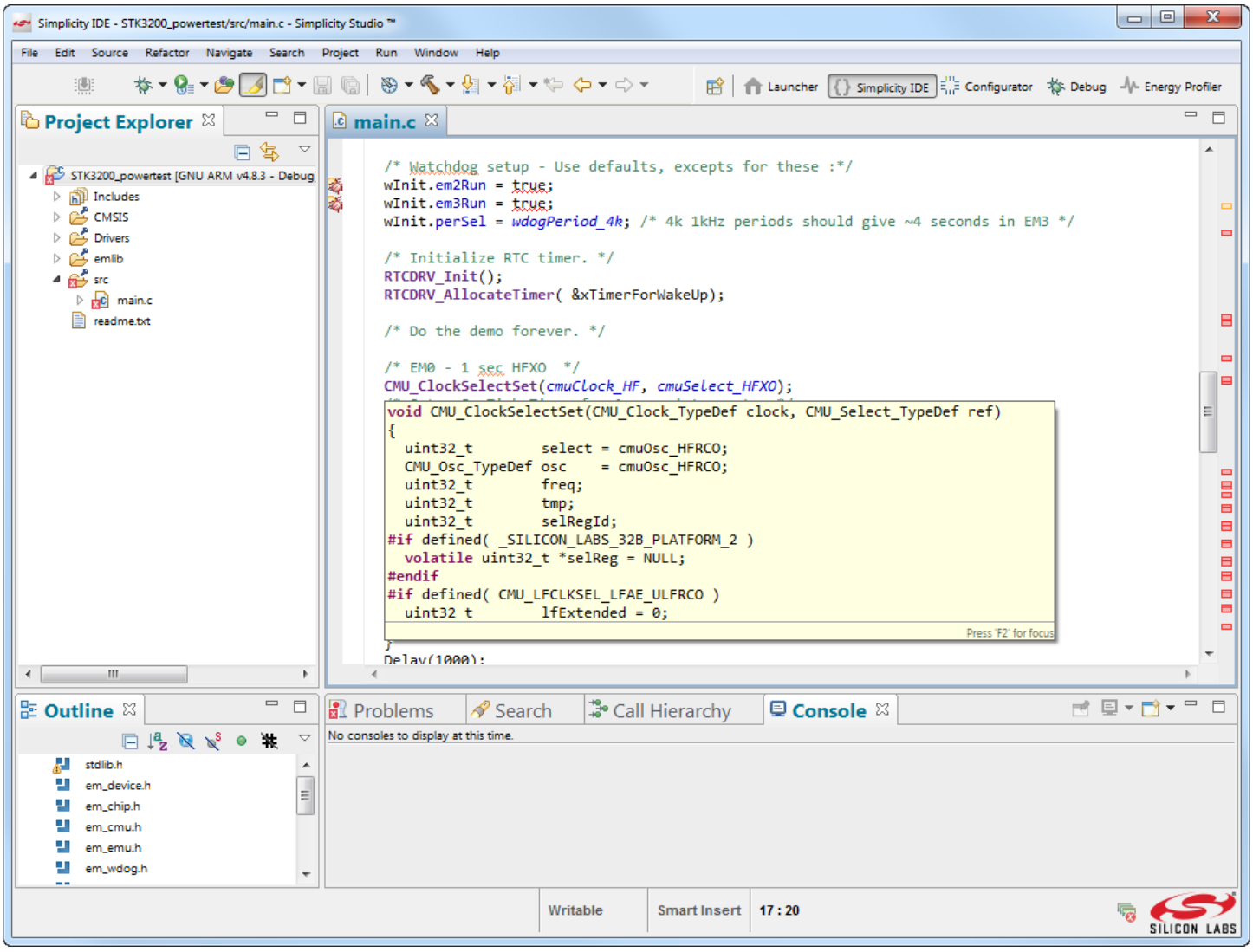


Figure 3.1. Simplicity IDE

Launch the Simplicity IDE from the Simplicity Studio launcher by clicking on the [**Simplicity IDE**] perspective, clicking the [**Software Examples**] links, clicking the [**New Project**] button, or using the [**Import Project**] button in the [**Application Notes**]>[**View All**] dialog.

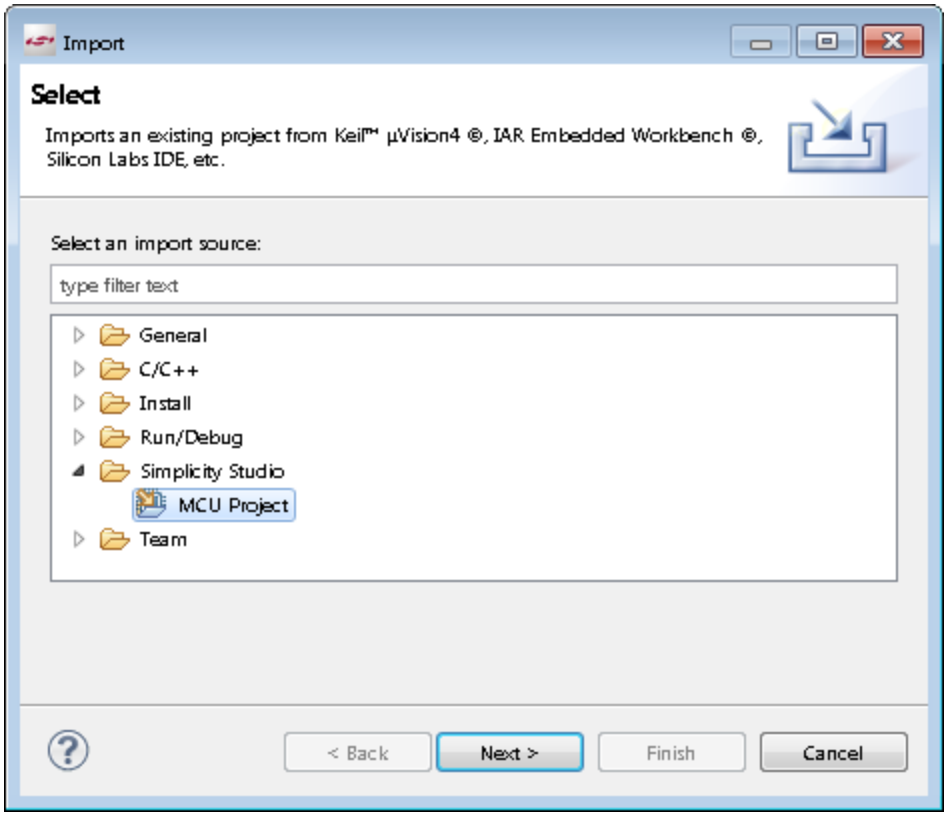
3.1 Projects

A Simplicity IDE project contains files, build options, and project settings. Projects generally exist as a directory containing sub-directories and files. The project structure seen in the IDE in the [**Project Explorer**] view is replicated physically on the disk. However, a project may also contain linked files or directories which are just pointers to files or folders outside of the project directory.

3.2 Importing a Project

To import a project from Keil µVision4, IAR Embedded Workbench, or the 8-bit Silicon Labs IDE into Simplicity IDE:

1. Go to [**File**]>[**Import**].
2. Select [**Simplicity Studio**]>[**MCU Project**].



3. Follow the steps in the wizard to finish importing the project.

The SDK is a collection of header files and examples for the device families. The header files define registers and enumerations for each bitfield within registers.

3.3 Exporting a Project

To share a Simplicity IDE project:

1. Go to [**File**]>[**Export**].
2. Select [**Simplicity Studio**]>[**MCU Projects to Files**].
3. Follow the steps in the wizard to finish exporting the project.

This will enable Simplicity Studio to re-import the project at another time.

3.4 Creating a New Project

There are four project types in Simplicity Studio:

- Empty C Program
- Example—This type of project is based on a pre-existing example.
- Library—A Library project enables the creation of a library, which can then be used as part of another project.
- Simplicity Configurator Program—This type of project enables pin, clock, and peripheral configuration of supported devices. This option may not be present if it has not been installed.

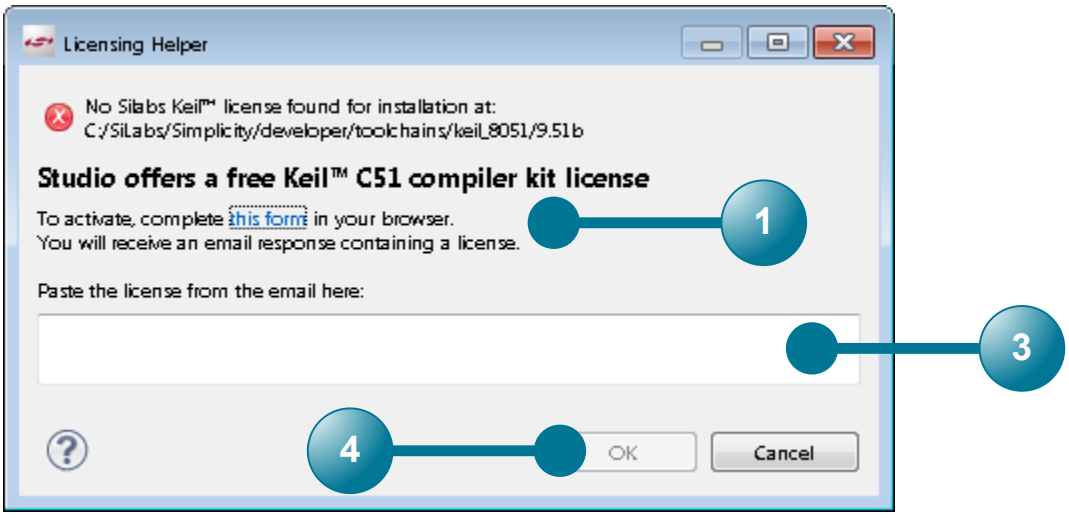
To create a new project in Simplicity IDE:

1. Click the [**Getting Started**]>[**New Project**] button from the launcher.
2. Select the desired [**Kit**], [**Part**], and [**SDK**] from the drop-down menus and click [**Next**].
3. Select the project type and click [**Next**]. For [**Example**] projects, select the example and click [**Next**].
4. Name the project and click [**Next**] or [**Finish**].
5. After clicking [**Next**], the [**Build Configurations**] dialog enables setup of the project build options. The default is [**Debug**] and [**Release**] configurations, which differ by debug symbol and optimization settings.

Alternatively, create a project directly from the Simplicity Studio launcher by clicking the [**Getting Started**]>[**Software Examples**]>[**View All**] link and following the wizard.

Simplicity IDE can detect if the Keil C51 toolchain is not activated. If the [**Licensing Helper**] is displayed, perform the following steps to activate the toolchain:

1. Click the [**this form**] link to open the activation form in a web browser.
2. Complete and submit the form.
3. Once a registration key is received, copy the key into the text box in the [**Licensing Helper**] dialog.
4. Click [**OK**].



The Licensing Helper can also be accessed using the [**Help**]>[**Licensing**] menu option.

3.5 Code Editing Features

Simplicity IDE is a code editing and development environment. The editor includes context highlighting, reference searching, and standard features found in any modern editor.

Open Declaration

In addition to the basic features, Simplicity IDE supports many advanced code-editing features. For example, the IDE automatically indexes all code within the project to support symbol lookup. The code does not have to build completely for the indexer to work, though certain features may not be available if, for example, the main() routine is not declared.

In this example, the [**Open Declaration**] (F3 shortcut) feature quickly finds symbol declarations.

1. Open the file of interest by double-clicking it in the [**Project Explorer**].
2. Right-click on the desired symbol to display the context menu.
3. Click [**Open Declaration**] to quickly navigate to the definition of the symbol (e.g., [**TMR2CN_TF2H**]).
4. Studio will automatically open the file and highlight the line containing the declaration of the symbol.

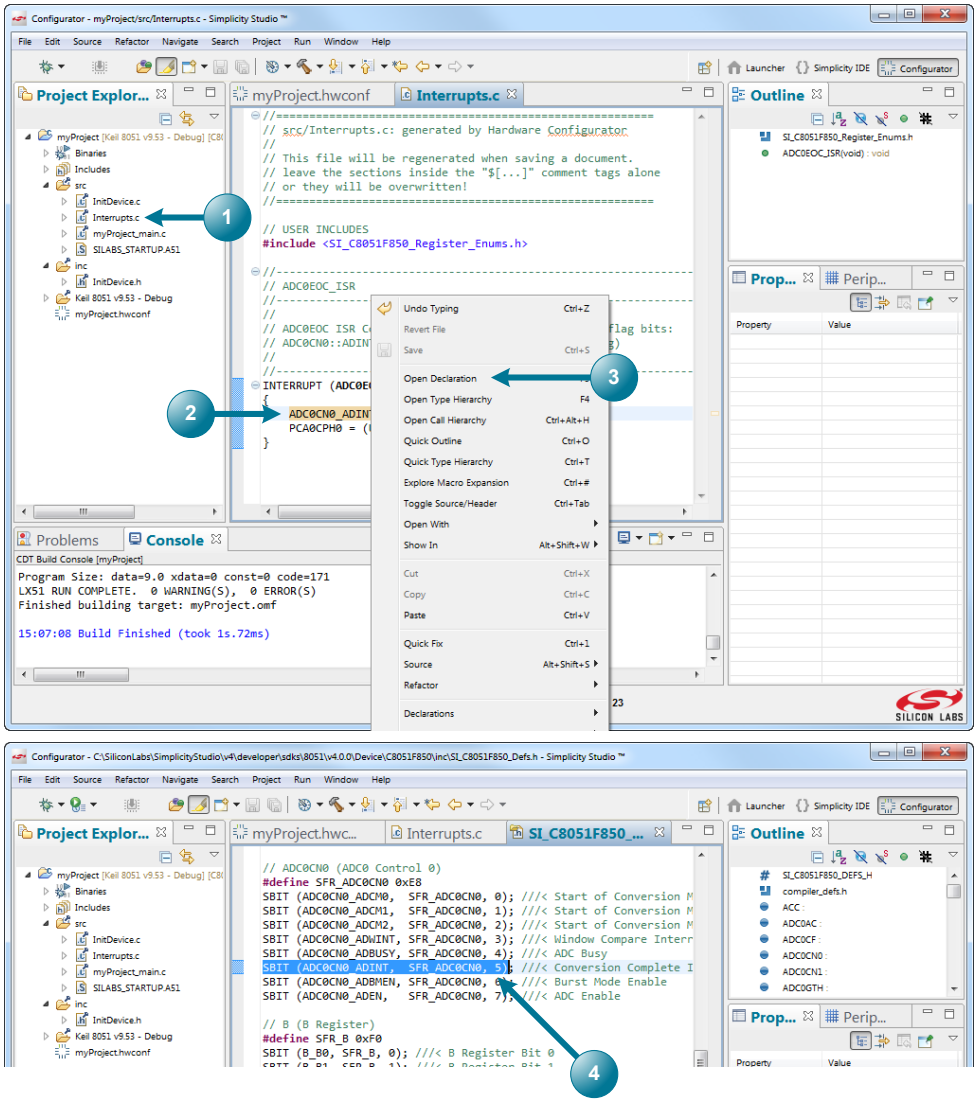


Figure 3.2. Finding Symbol Declarations

Content Assist

Simplicity Studio also supports code completion, a feature called [**Content Assist**]. Content Assist requires that the appropriate header files be included in the file so that the symbols are available. To use [**Content Assist**], type the first few letters of a symbol or include file and press [**Ctrl+Space**] to display a list of symbols that match. For example, to use [**Content Assist**] to display a list of symbols starting with the characters [**P1**]:

1. Open a file and type the desired characters in the file (e.g., [**P1**]).
2. Press [**Ctrl+Space**] to display the [**Content Assist**] list.
3. Use the arrow keys or page up and down keys to look through the list of matching symbols.
4. Pressing [**Enter**] will replace the typed characters with the selected symbol.

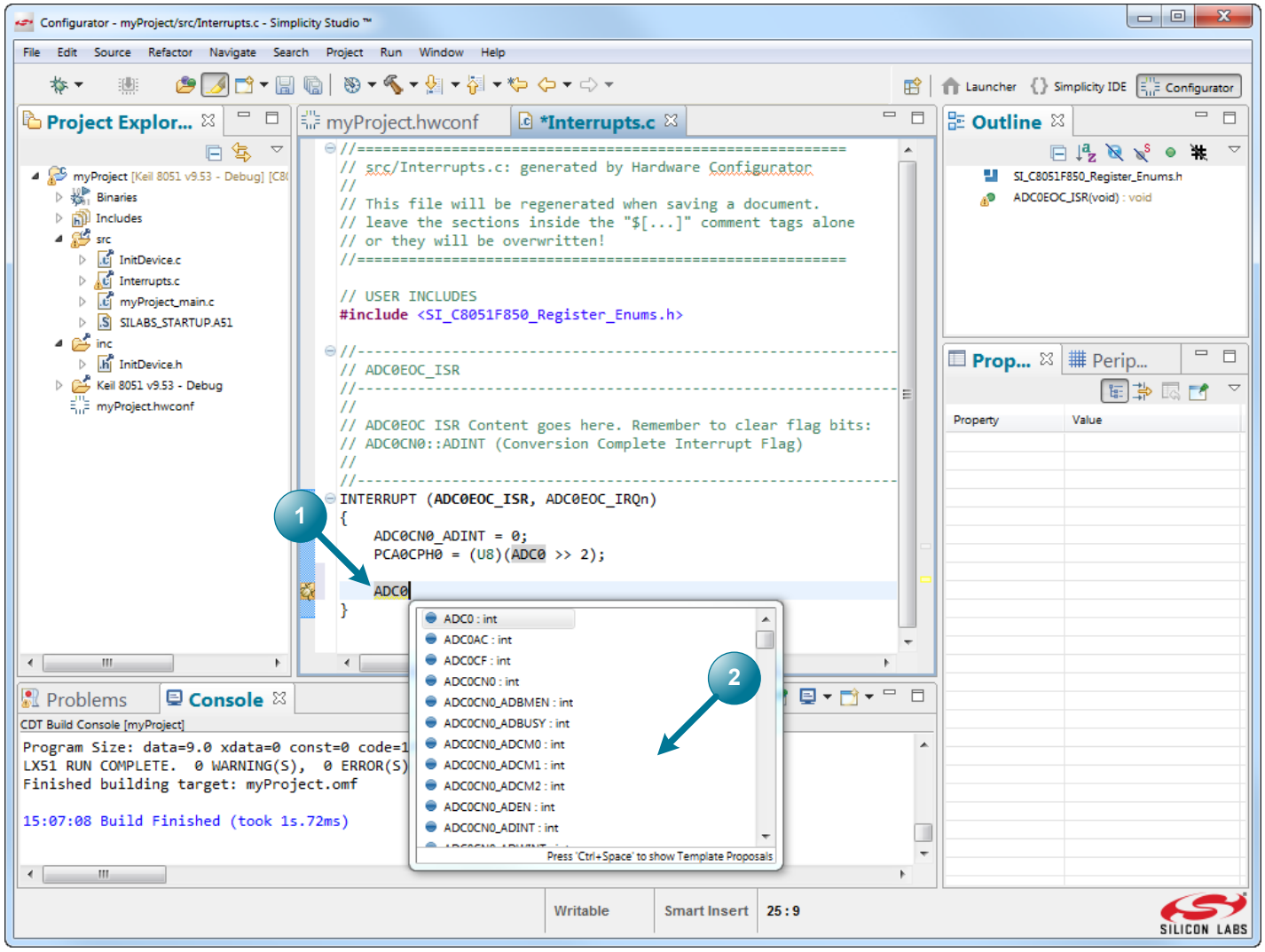


Figure 3.3. Using Content Assist

Link with Editor

The [**Link with Editor**] button synchronizes the editor with the [**Project Explorer**] view, highlighting the file currently selected in the editor.

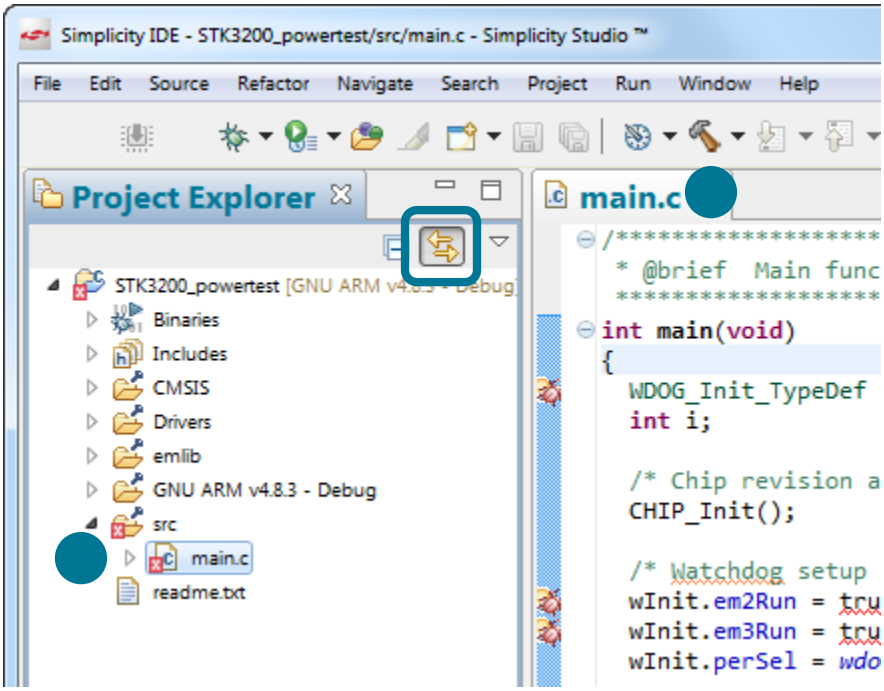


Figure 3.4. Link with Editor

Symbol Expansion

Hovering over a function or macro in the editor will create a hover window with expanded information on that symbol.

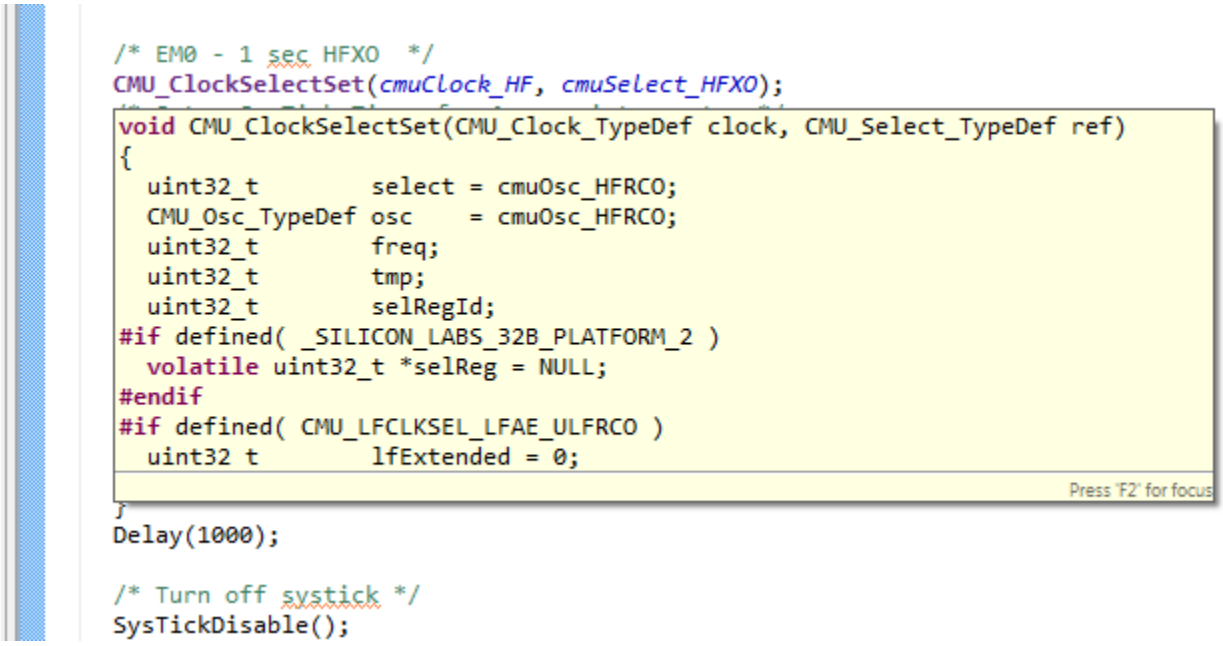


Figure 3.5. Symbol Expansion

Task Lists

The task list will automatically pick up any comments with [**TODO**] in the line and consolidate them into the [**Tasks**] view. To do this:

1. Type [**TODO**] in a comment line with the desired text.
2. Go to [**Window**]>[**Show View**]>[**Other**], type in [**Tasks**], and press [**OK**].
3. The [**TODO**] lines are highlighted by a clipboard in the left-hand blue margin in the editor, and clicking on a line in the [**Tasks**] list will jump to that place in the project.

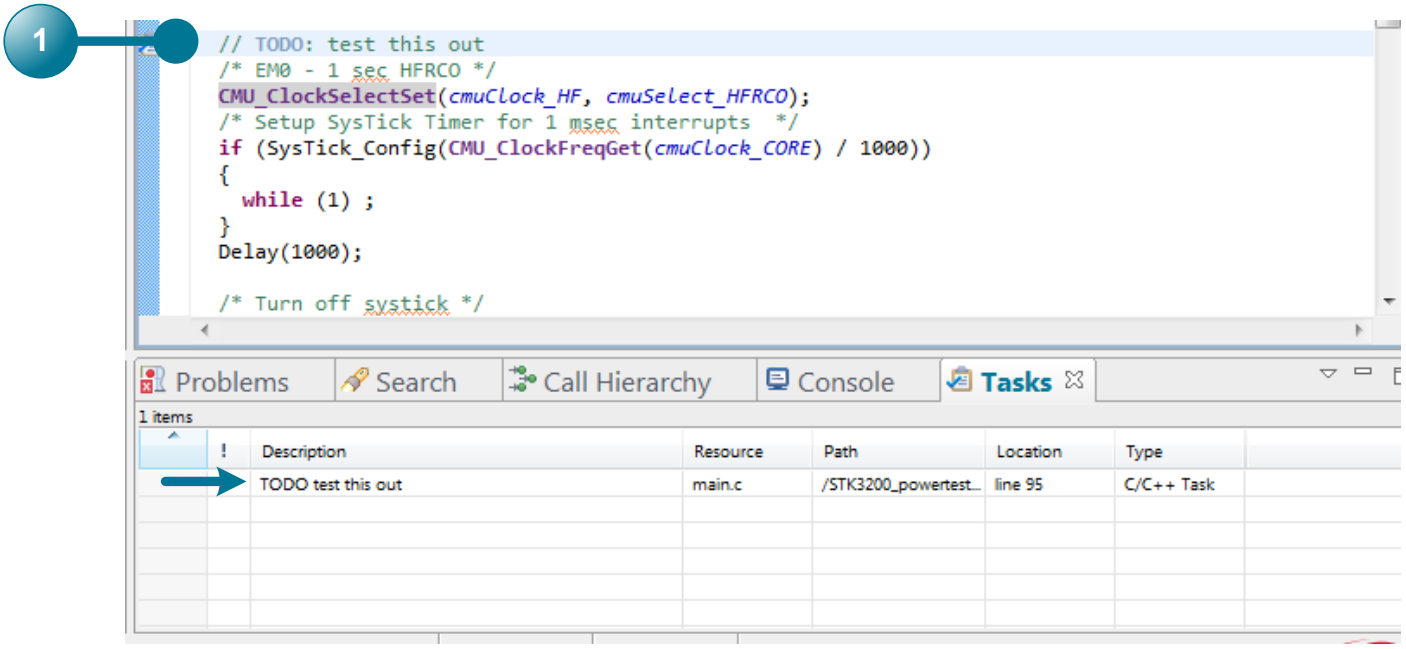


Figure 3.6. Using the Task List

Quick-Access Console

Pressing [**Ctrl+3**] within Simplicity Studio brings up a quick-access console for locating any menu or view within the IDE. For example, press [**Ctrl+3**] and type [**Preferences**]. This lists all of the Preferences menus available within Simplicity Studio. Then, select an option to open the menu.

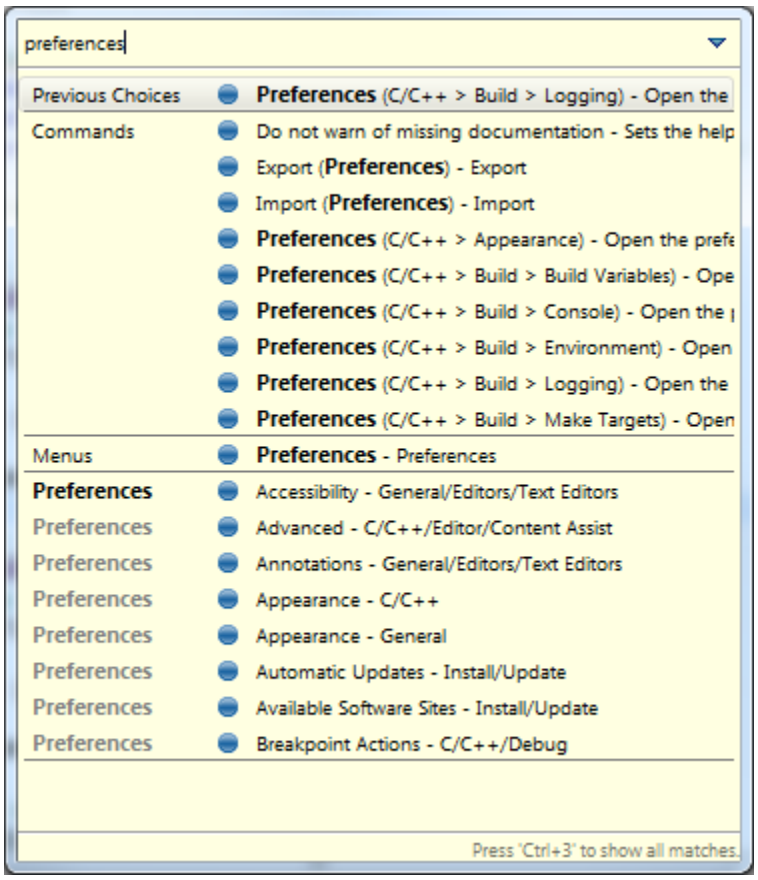


Figure 3.7. Quick-Access Console

Call Hierarchy

The IDE includes a call hierarchy that can help find where functions are called. To find the call hierarchy for a function in development mode, right-click on it and select [**Open Call Hierarchy**].

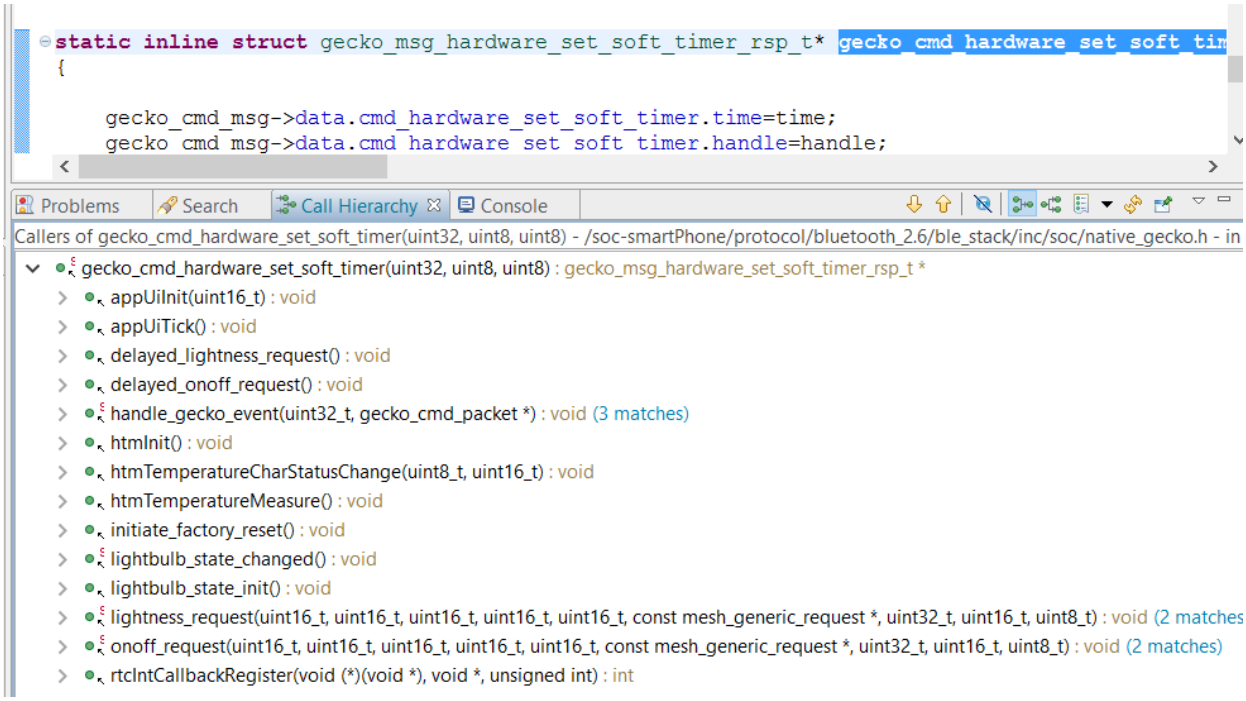


Figure 3.8. Using the Call Hierarchy

4. Debugger

The debugger supplied with Simplicity Studio is based on the Eclipse EDC debugger. It is a full featured debugger that offers the ability to step through code, set breakpoints, examine memory, variables and registers. Silicon Labs supplies the debug models for each of their devices so that all of the registers can be examined and modified using the debugger.

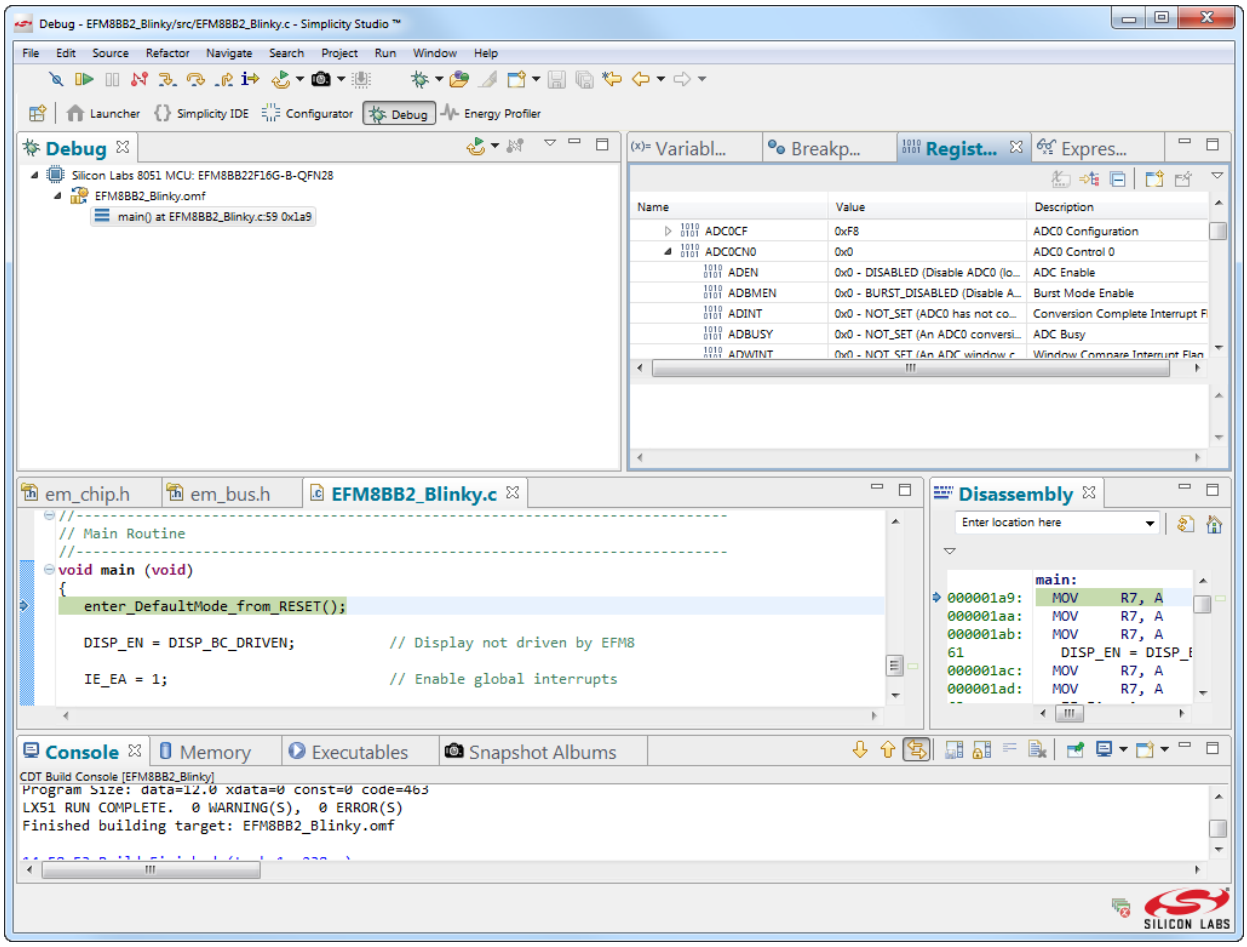


Figure 4.1. Debugger

The debugger perspective can be opened a variety of ways, the most common way is by clicking the debug icon from the Simplicity IDE perspective after a project has gotten far enough along that it will build and link correctly. If a debug session has never been started before a warning that a debug configuration does not exist might appear. In this case click the small down chevron next to the debug icon and then choose "Debug as..." and then select either "Silicon Labs ARM Program" for 32-bit parts or "Silicon Labs 8051 Program" for EFM8 and C8051 parts.

4.1 Common Debug Actions

In a debug session, the IDE switches to the [**Debug**] perspective with the common actions shown in Table 4.1 Simplicity IDE Debug Toolbar on page 18. If a single debug adapter is available, the IDE will automatically download the code to the MCU. If more than one debug adapter is available, the IDE will prompt to select a debugger.

Table 4.1. Simplicity IDE Debug Toolbar

Icon	Command	Description
	Debug	The [**Debug**] button starts a new debug session. An active debug session must be disconnected before starting a new session using the same debug adapter.
	Resume	The [**Resume**] button runs the MCU after reset or after hitting a breakpoint.
	Suspend	The [**Suspend**] button halts the MCU.
	Disconnect	The [**Disconnect**] button terminates the current debug session and disconnects the debug adapter. The IDE will automatically switch back to the Development perspective.
	Reset the Device	The [**Reset the Device**] button performs a hardware reset on the MCU.
	Step Into	The [**Step Into**] button single steps into the first line of a function.
	Step Over	The [**Step Over**] button single steps over a function, executing the entire function.
	Step Return	The [**Step Return**] button steps out of a function, executing the rest of the function.
	Instruction Stepping Mode	The [**Instruction Stepping Mode**] button toggles assembly single stepping. When enabled, single steps will execute a single assembly instruction at a time. See the [**Disassembly**] view for the assembly code corresponding to the source code at the current line of execution.

The [**Debug**] view in the top-left corner displays any active debug sessions. A debug adapter can only support a single debug session at a time. An active debug session must be disconnected before code can be recompiled and a new debug session started.

To set a breakpoint, double-click in the blue bar to the left of the code editor or right-click on a line of code and select [**Add Breakpoint**]. Breakpoints can be managed using the [**Breakpoints**] view in the [**Debug**] perspective. Register contents are viewable and editable using the [**Registers**] view. Memory can be accessed using the [**Memory**] view.

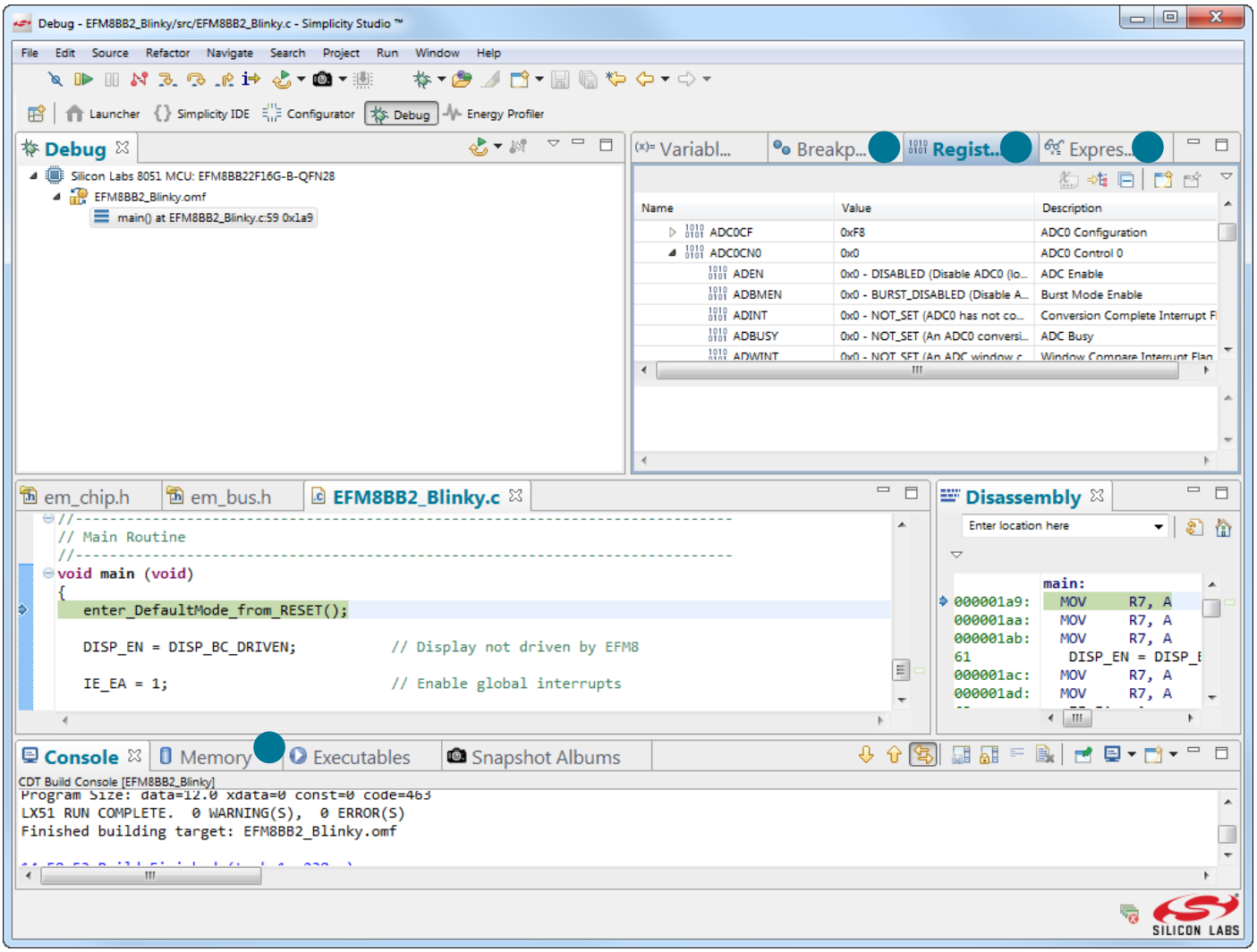


Figure 4.2. Navigating the IDE while Debugging

4.2 Debugger Call Hierarchy and Snapshots

The debugger shows the call stack and call hierarchy that can help debug and find where functions are called.

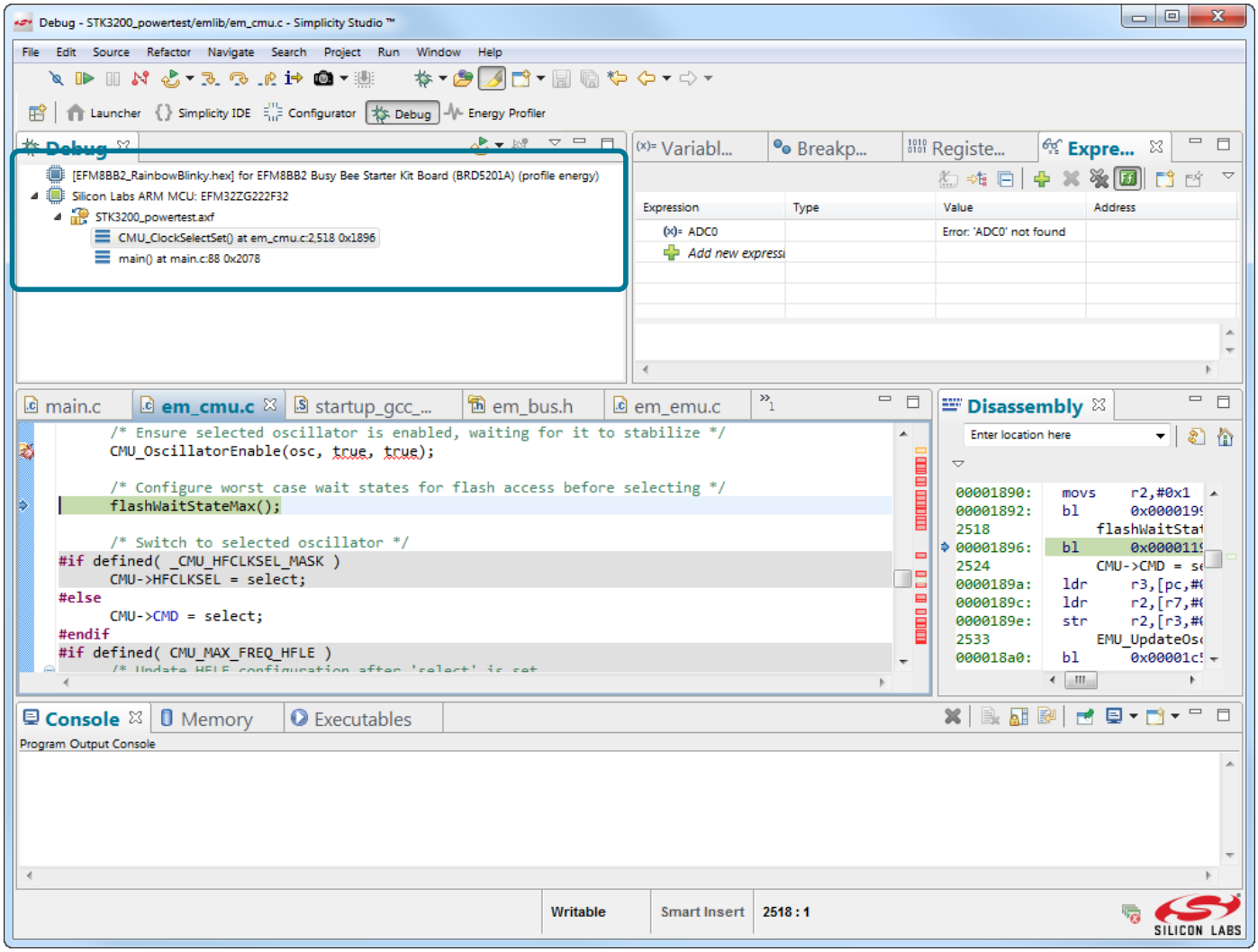


Figure 4.3. Debugger Call Hierarchy

Snapshots The [**[Snapshot]**] feature is available while debugging saves the values of the registers on a device at a particular point in time. Take add a snapshot: 1. Click the [**[Snapshot]**] button when the core is halted and the register values are constant and accessible by the debug hardware. 2. Each [**[Snapshot]**] is listed in the [**[Snapshot Albums]**] view. This feature is useful when comparing the states of two systems next to each other or looking at the state of the hardware at separate points in time. To view a snapshot, switch to the [Debug] perspective and go to [**[Window]>[Show View]>[Other]**] and select [**[Snapshot Albums]**]. In the list, right click on a [Snapshot] and select [**[Launch Snapshot]**]. This will add a debug session in the Debug window as it was at the time of the [**[Snapshot]**] with a Snapshot label at the start of the debug configuration, but it is not an active debug session on hardware. To switch back to a debug session on hardware, select the Silicon Labs ARM debug configuration in the Debug window if a hardware debug session was active or else go to [**[Run]>[Debug]**] if a session is not active.

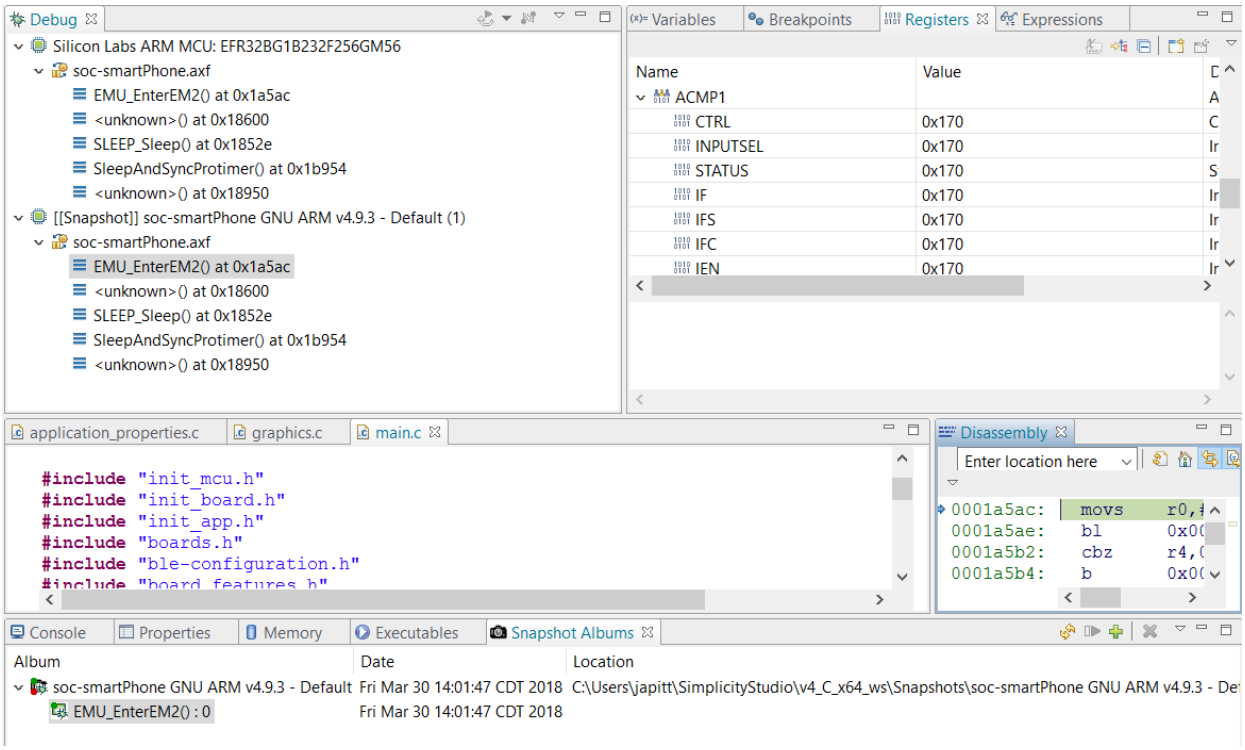


Figure 4.4. Debug Snapshot Session

5. Development Flow Examples

The following sections show simple examples of development flow for an EFM32 and an EFM8 projects.

5.1 Running Blinky with an EFM32 Device

To create a project based on the Blinky example using the Zero Gecko starter kit:

1. From the main Simplicity Studio launcher, click the [**New Project**] button.
2. In the first [**Project setup**] page of the project creation wizard:
   a. Select [**EFM32 Zero Gecko 3200 Starter Kit board**] in the [**Boards**] drop-down menu.
   b. The wizard automatically selects the [**EFM32ZG222F32**] device in the [**Part**] drop-down menu.
   c. Select the SDK in the [**SDK**] drop-down menu.
   d. Click [**Next**].
   e. Select the [**Example**] radio button.
   f. Click [**Next**].
3. Select the [**STK3200_blink**] example from the list and click [**Next**].
4. Accept the default [**Project name**] or give it a custom name and then click [**Finish**].
5. Click the project in the [**Project Explorer**] view and click the [**Debug**] button to build and download the code to the hardware.
6. Click the [**Resume**] button to start running the example. The LED should blink.
7. Press the [**Disconnect**] button to return to the [**Development**] perspective.

5.2 Running Blinky with an EFM8 or 8-bit Device

To create a project based on the Blinky example using the EFM8BB1 starter kit:

1. From the main Simplicity Studio launcher, click the [**New Project**] button.
2. In the first [**Project setup**] page of the project creation wizard:
   a. Select [**EFM8BB1 Busy Bee Starter Kit Board**] in the [**Boards**] drop-down menu.
   b. The wizard automatically selects the [**EFM8BB10F8G**] device in the [**Part**] drop-down menu.
   c. Select the SDK in the [**SDK**] drop-down menu.
   d. Click [**Next**].
   e. Select the [**Example**] radio button and click [**Next**].
3. Select the [**EFM8BB1 Blinky**] example from the list and click [**Next**].
4. Accept the default [**Project name**] or else give the project a custom name and click [**Finish**].
5. Click the project in the [**Project Explorer**] view and click the [**Debug**] button to build and download the code to the hardware.
6. Click the [**Resume**] button to start running the example. The LED should blink.
7. Press the [**Disconnect**] button to return to the [**Development**] perspective.

6. Energy Profiler

The [**Energy Profiler**] is available from the Simplicity Studio Compatible Tools area and enables Advanced Energy Monitoring. This tool enables power profiling of code in real-time by measuring the power consumption. Some devices also support linking the power consumed with the associated line of code.

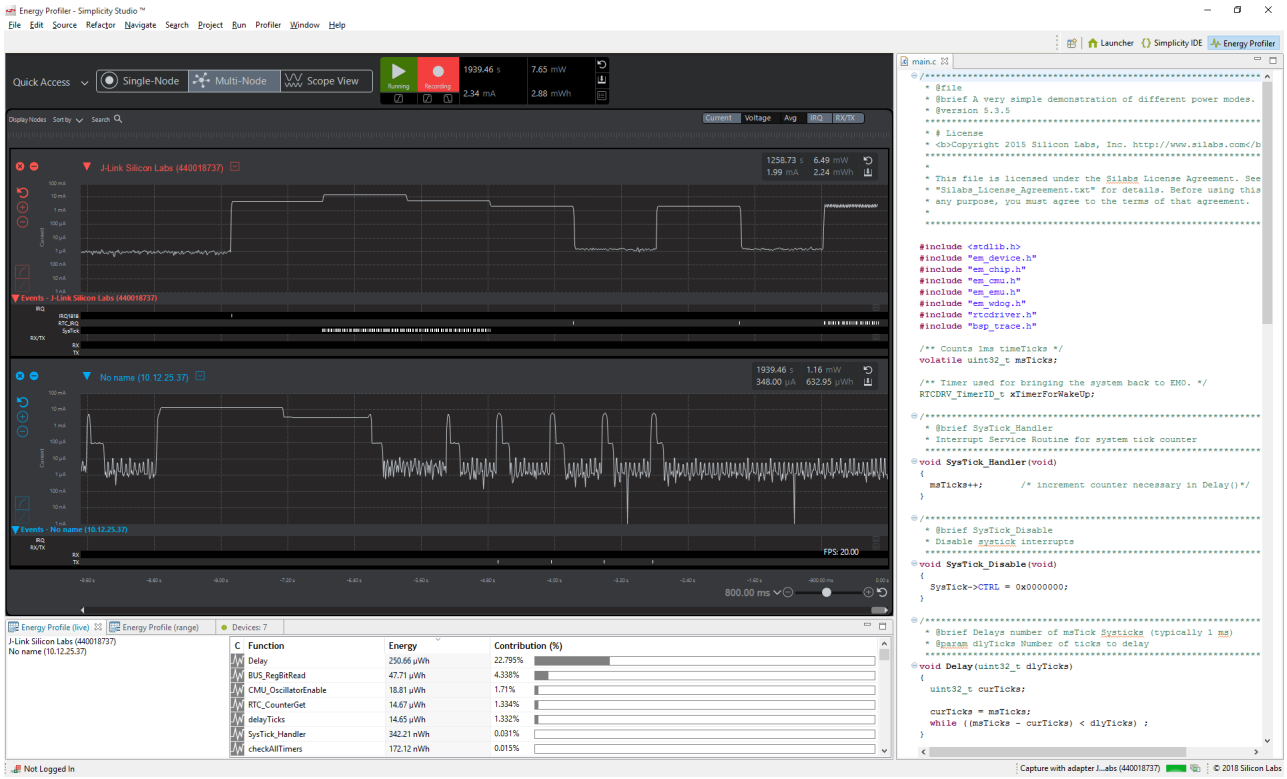


Figure 6.1. Energy Profiler

More information on this tool can be found by reading *UG343: Multi-Node Energy Profiler User's Guide*.

7. Hardware Configurator

Hardware Configurator is part of Simplicity Studio and greatly simplifies peripheral initialization by presenting peripherals and peripheral properties in a graphical user interface.

The majority of the initialization firmware can be generated by selecting peripherals and property values from combo boxes or entering register values in text boxes. Depending on the product family, some peripherals provide calculators, such as baud rate calculators, timer overflow rate calculators, and SPI clock rate calculators, that can be used to automatically confirm the necessary reload register value to generate the specified clock rate. Configurator also provides real-time validation of properties to ensure that a configuration is valid before downloading code to the MCU.

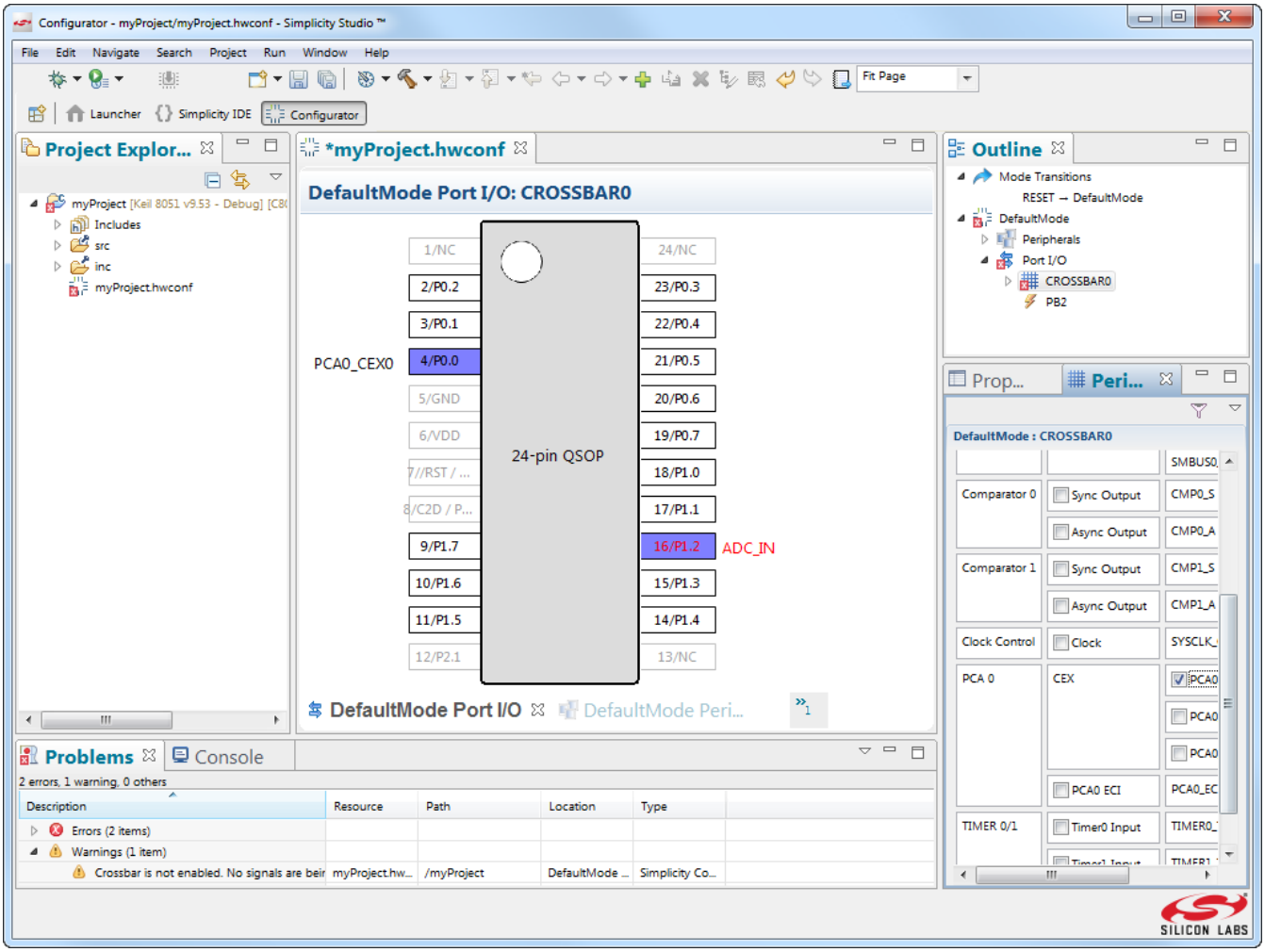


Figure 7.1. Simplicity Studio Hardware Configurator

More information on how to use Simplicity Configurator can be found in the following application notes:

- For EFM8 and C8051 products: *AN0823: Simplicity Configurator User Guide*.
- For EFR32 products using the EmberZNet (Zigbee), Thread or Flex SDKs: *AN1115: Configuring Peripherals for 32-Bit Devices in Simplicity Studio*.

Application notes can be found on the Silicon Labs webpage (http://www.silabs.com/32bit-appnotes or http://www.silabs.com/8bit-appnotes) or in Simplicity Studio using the [**Application Notes**] section from the Launcher Getting Started or Documentation tabs.

8. Network Analyzer

The Simplicity Network Analyzer enables debugging of complex wireless systems. This tool captures a trace of wireless network activity that can be examined in detail live or at a later time.

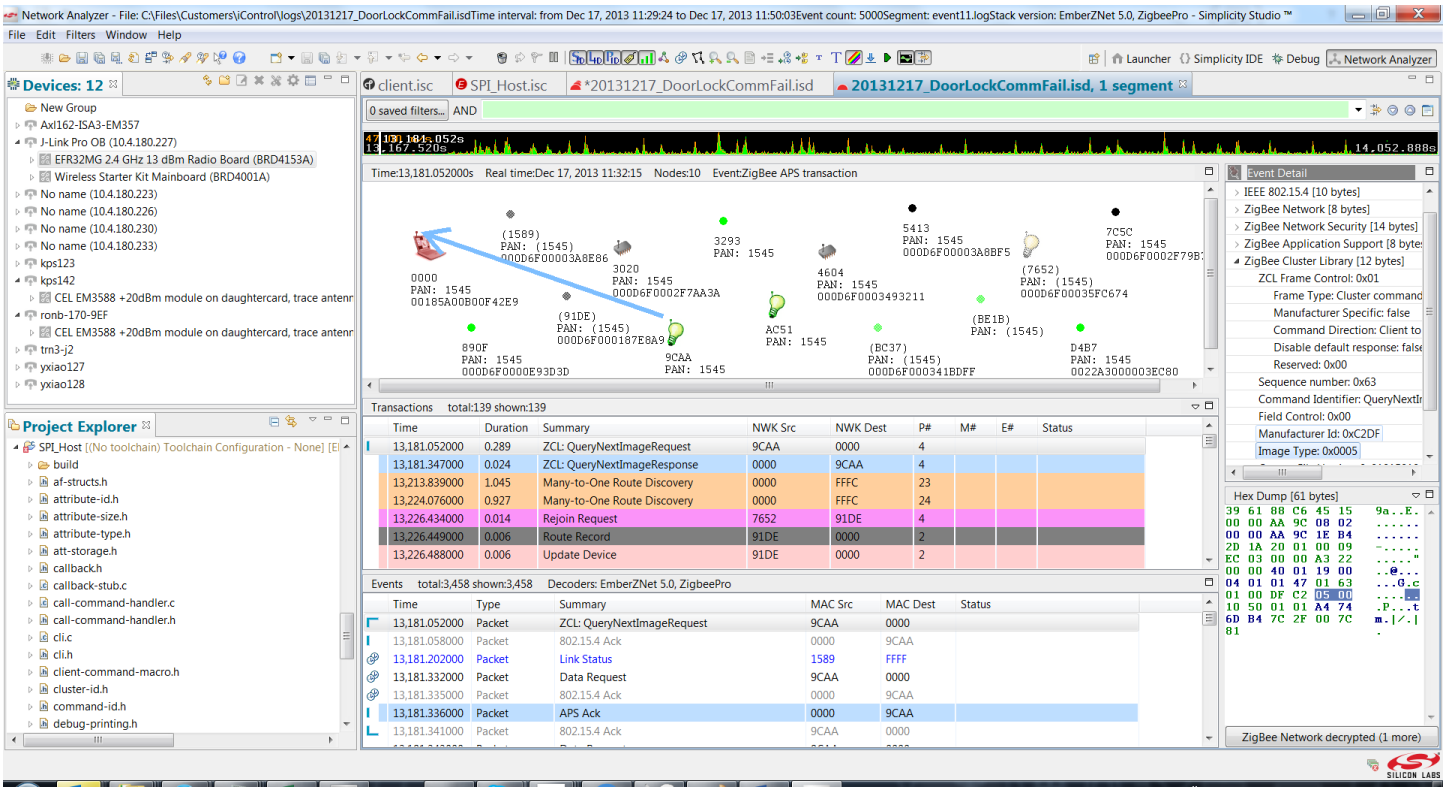


Figure 8.1. Simplicity Network Analyzer

The Simplicity Network Analyzer incorporates all the features of Silicon Labs' Ember Desktop software. More than simply a packet sniffer, the Network Analyzer works with the data sniffer interface on the Silicon Labs wireless chips to provide direct feedback from the baseband radio of each device, allowing any supported radio to report detailed packet transmission and reception data, such as timestamps, link quality (or LQI), receive sensitivity (or RSSI), and CRC pass/fail results, all without any software overhead.

With Simplicity Studio, any PTI-enabled Silicon Labs platform can record the radio activity regardless of the application firmware that is being used, so there's no need to have a dedicated sniffer device installed to catch the traffic. The Network Analyzer also enables capture from multiple sources simultaneously into the same log file without falsely duplicating packets. This enables the developer to compare how well different radios in the network heard the same transmission.

In cases when detail is not desired, Network Analyzer makes it easier to understand the workings of a complex wireless protocol. Related packet events are automatically grouped into a [**Transactions**] pane within the capture view, allowing for quicker parsing of what's happening during that portion of the traffic log. Quickly access statistics like total duration, number of related packets, number of point-to-point and end-to-end retries, and unexpected conditions like requests with missing responses or deliveries where expected acknowledgments are missing.

8.1 Filtering Events and Data

Filters can be applied to live or pre-captured traffic and can be entered via the [**Expression Builder**] by selecting one of the suggested filters in the context menu of the [**Transactions**] or [**Events**] pane or by manually entering or editing text in the [**Filter Bar**].

Filters can be chained together using regular expressions to produce more specific results. Filters can also double as search criteria with the rest of the surrounding events for context.

When the developer finds an item of interest, add a bookmark for future reference. This is great for annotating captures during debugging or collaborating with other users.

8.2 Logging Results

Simplicity Network Analyzer can record software-based debug activity from the firmware on the embedded networking devices, allowing application developers a glimpse into which firmware events correlate closely in time with certain network-related events and transactions.

Stepping through a time-sensitive embedded networking application with a traditional debugger can disrupt the networking stack's real-time behavior, but with Network Analyzer, a firmware engineer can get a transcript of all networking stack API calls and stack callback functions as they happen, which can be a big help in shedding light on why a certain network activity occurred or why it had a particular outcome. Silicon Labs even provides a Virtual UART interface to allow for traditional printf output through Serial Wire debug lines to enable application printf statements the stack API and callback activity.

Other critical software activities, such as resets, assert() violations, and stack startup, are also captured in this [**Events**] listing, so developers have the maximum amount of information about what happened at runtime. This recorded information can also be sent to Silicon Labs technical support to quickly resolve problems or answer questions.

8.3 More Information

More information on the Simplicity Network Analyzer tool can be found in "UG104: Testing and Debugging Ember Applications" in Chapter 3. This document can be found on the Silicon Labs website: https://www.silabs.com/documents/public/user-guides/UG104-TestDebugEmberApps.pdf

9. Application Builder

Simplicity Studio shortens the software design cycle and reduces time-to-market with Application Builder (AppBuilder), a graphical interface that works in harmony with Silicon Labs' Wireless SDKs and the Gecko Bootloader. The functionality provided by Application Builder varies depending on the SDK being used.

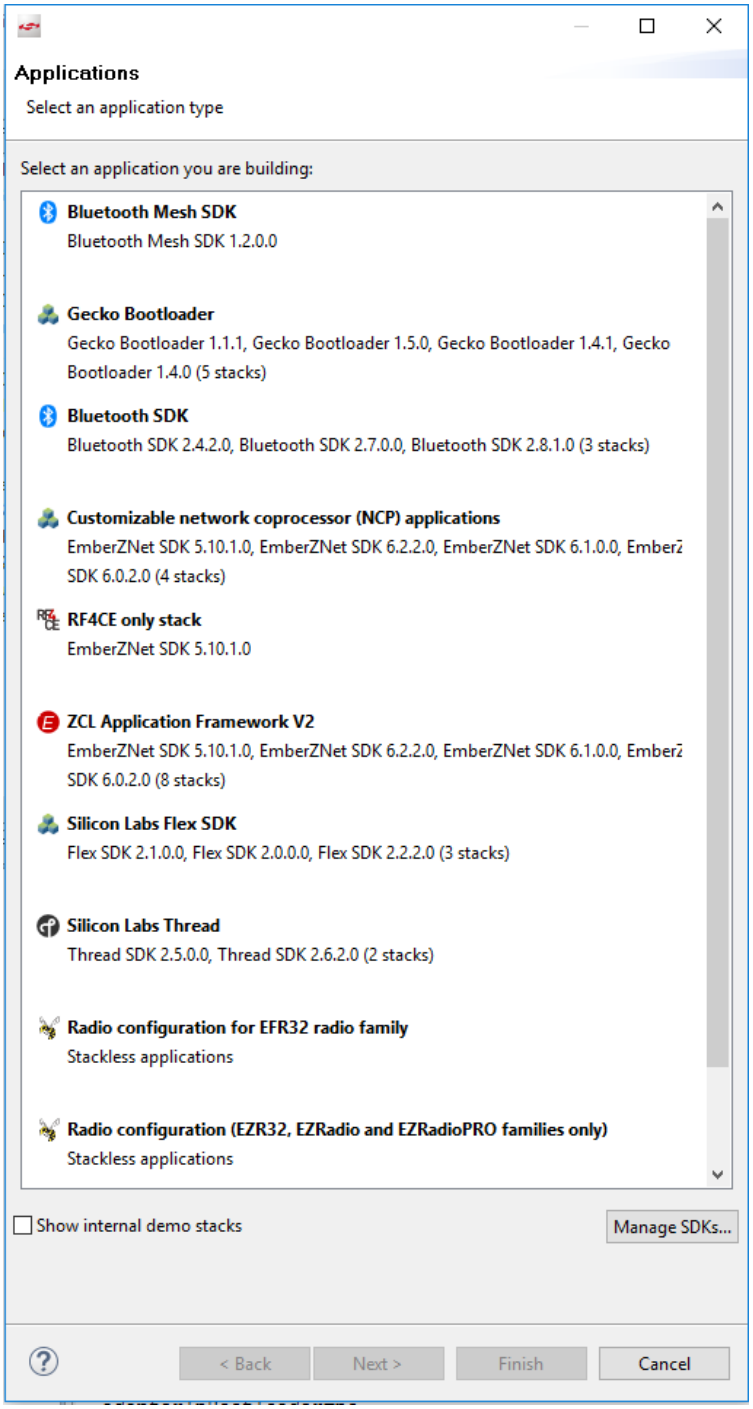


Figure 9.1. Application Builder Types

9.1 EmberZNet (Zigbee), Thread and Flex SDK-Based Applications and Gecko Bootloader Projects

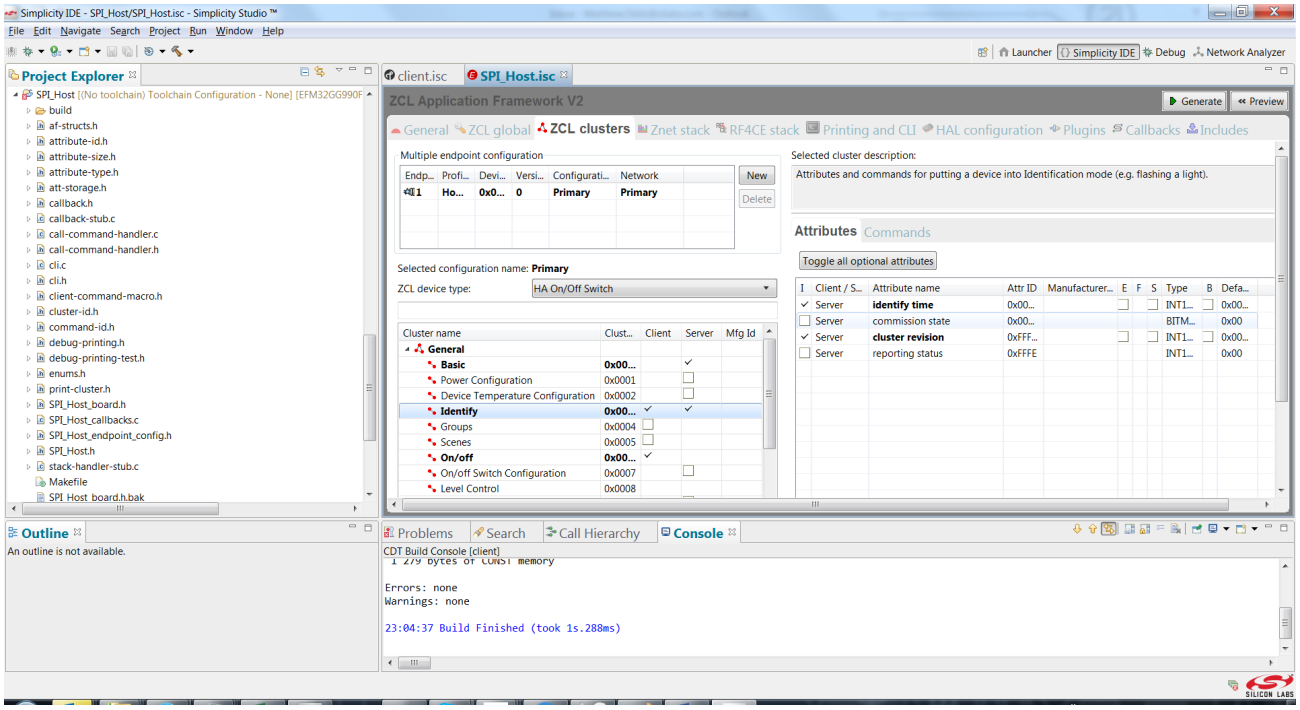


Figure 9.2. Simplicity Application Builder

The Silicon Labs Application Builder, or AppBuilder, allows a developer to start a new project based on an existing framework of best-practice application state machine code developed and tested by Silicon Labs. This framework sits on top of the wireless networking stack to interface with the Hardware Abstraction Layer (or HAL) and provide application layer functionality, including the following:

- Start-up routines.
- Mechanisms for finding, joining, or forming networks.
- High-level APIs for creating, parsing, and handling message payloads.
- Configuration of the networking stack.
- Command-line interfaces for control of the program.
- Human-readable debug output tailored to the needs of the developer.
- Incorporation of customer-provided libraries and code modules.

The AppBuilder tool is modular enough to accommodate different data sets for different versions of application protocols, so you can choose exactly the right version of application-level specs when it really matters for interoperability.

Within a given application layer, like Zigbee's ZCL or Zigbee Cluster Library, you can pick from any of the defined device types to get the desired mixture of client and server functionality, or you can build your own custom device definition from any of the available client and server clusters.

Device-wide, stack-level settings like network node type and security method can easily be set via the picklists in AppBuilder, and all relevant code is automatically included, with all unnecessary code being automatically removed in the final output. You can even use the Simplicity Studio Application Builder to configure two different networks on the same chip, such as Zigbee PRO and RF4CE.

9.1.1 Callback Interface

AppBuilder is very versatile, with one example being the callback interface, which allows user applications to be notified about important events or make decisions based on runtime state. Software designers can choose which callbacks are needed for their application and can leave the rest disabled so that only the chosen callbacks appear in the user's `Callbacks` C file. The rest are safely defined as empty stubs within the framework until they are enabled.

Callbacks are the places where custom application code can be added on top of Silicon Labs' existing framework to give that application unique behaviors and decide how it will react. Within the callback implementations, developers can utilize the entire HAL and stack APIs as well as a complete set of Application Framework-specific APIs that often provide high-level wrappers around complex HAL or stack functionality. These APIs are documented in an Application Framework API Guide and be found in use in existing sample code. Like all of the Silicon Labs Application Framework, full source code in C is provided to help understand the implementation of these APIs.

Overall, there are over 200 callbacks to choose from. In addition, Application Builder automatically provides a unique callback for every incoming message type supported by the device's application protocol, enabling the application to handle each incoming command frame exactly as desired.

9.1.2 Customization

To add proprietary extensions to a standard application protocol like ZCL, just describe the custom clusters, commands, attributes, and device types in an XML file that follows the Silicon Labs schema, and point AppBuilder to the XML file. The next time a configuration based on that stack is created or edited, those manufacturer-specific enhancements will appear as clusters and attributes that can be enabled, and callbacks will automatically be provided to handle any proprietary incoming commands declared in the XML data. AppBuilder will even create macros that firmware can use to quickly craft an outgoing message in the proprietary format, enabling the assembly of a command frame and the passing of arguments from the API to populate parameters of that command.

9.1.3 Plugins

The [**Plugins**] tab in AppBuilder also enables application acceleration. This tab has over 100 plugins that each implement one or more callbacks on top of the Application Framework to provide a sophisticated state machine for complex features like message fragmentation, RF4CE device profiles, or external dataflash storage for downloading firmware updates. Source code for all plugins is available, so the code is there as a point of reference for customized implementations.

Many plugins implement functionality for a specific Zigbee ZCL cluster, such as Level Control, Groups, or Key Establishment, so the plugin can be enabled and, in many cases, provide all the needed messaging required to pass ZigBee application layer compliance tests. Other plugins implement a software state machine and expect developers to integrate the hardware behavior into that state machine to get the device to physically react to the change in software state. Still other plugins are provided as sample code that can be used like a test harness to exercise functionality otherwise not available in the system.

9.1.4 Generating a Project

After making selections for any portions of the application the developer wishes to customize, AppBuilder will generate a software project with customized header files and array definitions in C code to represent the desired application behavior. The resulting project can be built from within Simplicity IDE to produce a binary that can be loaded onto the target wireless SoC.

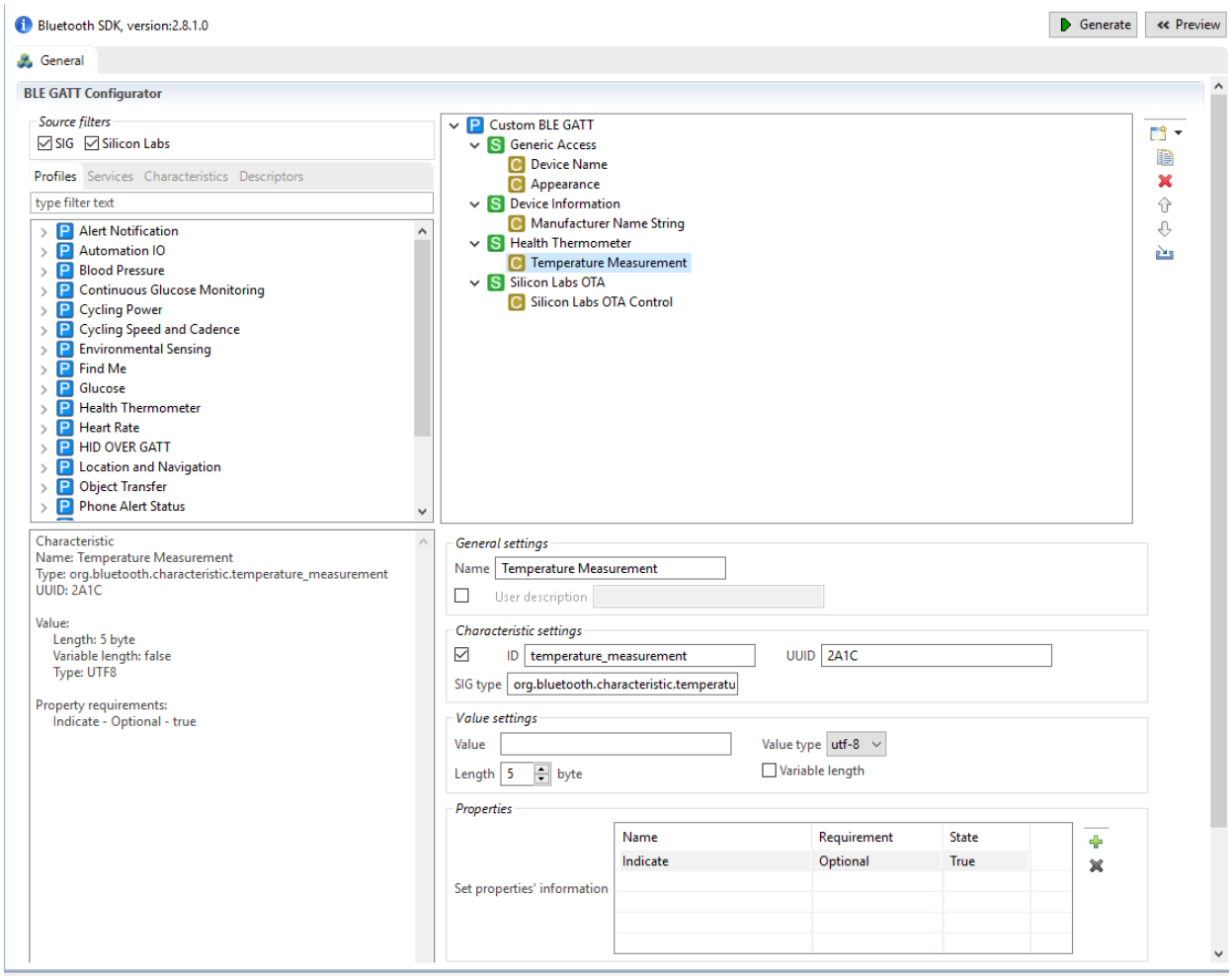
This project generation is specific to the stack version and target chip where the configuration is done, but the Silicon Labs Application Framework abstracts many of these differences within its APIs and state machines, so AppBuilder configurations can be reused across different stack versions or chip platforms. This makes it much easier to transition from one version of a stack release to the next.

9.1.5 More Information

More information on the Simplicity AppBuilder tool can be found in "UG102: Ember Application Framework Developer's Guide." This document can be found on the Silicon Labs website: http://www.silabs.com/Support%20Documents/TechnicalDocs/UG102.pdf.

9.2 Bluetooth SDK Based Applications

For Bluetooth SDK based applications, AppBuilder provides a graphical way to specify the Bluetooth profiles and services the application supports.



More information about specifying the Bluetooth profiles can be found in the following documents:

- *UG136: Silicon Labs Bluetooth ® C Application Developer's Guide*
- *UG118: Blue Gecko Bluetooth® Profile Toolkit Developer's Guide*

The GATT (Generic Attribute Profile) editor section of *QSG139: Bluetooth® Development with Simplicity Studio*

10. Simplicity Studio Help

Simplicity Studio includes detailed help information and device documentation within the tool. The help contains descriptions for each dialog window. To view the documentation for a dialog, click the question mark icon in the window:

This will open a pane specific to the dialog with additional details.

The documentation within the tool can also be viewed by going to [**Help**]>[**Help Contents**] or [**Help**]>[**Search**]. Simplicity Studio also contains several interactive walkthroughs called [**Cheat Sheets**]. These tutorials provide guidance through basic usage scenarios within the IDE. Access these tutorials by going to [**Help**]>[**Cheat Sheets**].

11. New to Eclipse?

Simplicity Studio is an Eclipse-based application, so for those unfamiliar with Eclipse this section details some terminology and other information about Eclipse that will make it easier for you to transition to Simplicity Studio. Eclipse is based on the concept of a "Workbench" which is the desktop development environment. The Workbench allows various tools to be easily integrated as different resources. From the Eclipse.org website "each Workbench window contains one or more perspectives. Perspectives contain views and editors and control what appears in certain menus and tool bars." So the main tools available in Simplicity Studio are different perspectives. The perspective shown when Simplicity Studio is first installed is called the "Launcher". Most of the chapters in this Users' Guide are describing a different perspective available in Simplicity Studio (Simplicity IDE, Debugger, Energy Profiler, etc).

Simplicity Studio also relies on the Eclipse C/C++ Development Toolkit (CDT) to provide the majority of the functionality in the Simplicity IDE and Debugger.

11.1 Views and Perspectives

Perspectives are a saved set and configuration of views. Each view within a perspective provides information specific to the active perspective's task. For example, while debugging a project, the [**Debug**] perspective shows the [**Debug**] view with stack trace information, [**Registers**] view, [**Breakpoints**] view, [**Expressions**] view, and [**Disassembly**] view.

Swap between perspectives using the buttons in the upper-right corner of the IDE. The perspectives available by default are as follows:

- Launcher—Clicking the [**Launcher**] perspective will reopen the main Simplicity Studio launcher page. Click the Simplicity IDE tile to return to the Development perspective.
- Simplicity IDE—The [**Simplicity IDE**] perspective to switch to the default perspective designed for code editing and development.
- Configurator—The [**Configurator**] perspective is used by the Simplicity Configurator tool, which enables peripheral and pin configuration for devices.
- Debug—Starting a debug session will automatically switch the IDE to the [**Debug**] perspective, which includes useful debugging tools like the [**Registers**] and [**Disassembly**] views.
- Energy Profiler—Clicking the [**Energy Profiler**] perspective will automatically switch to the Energy Profiler tool, which uses the AEM circuitry on Starter Kits to measure power consumption.

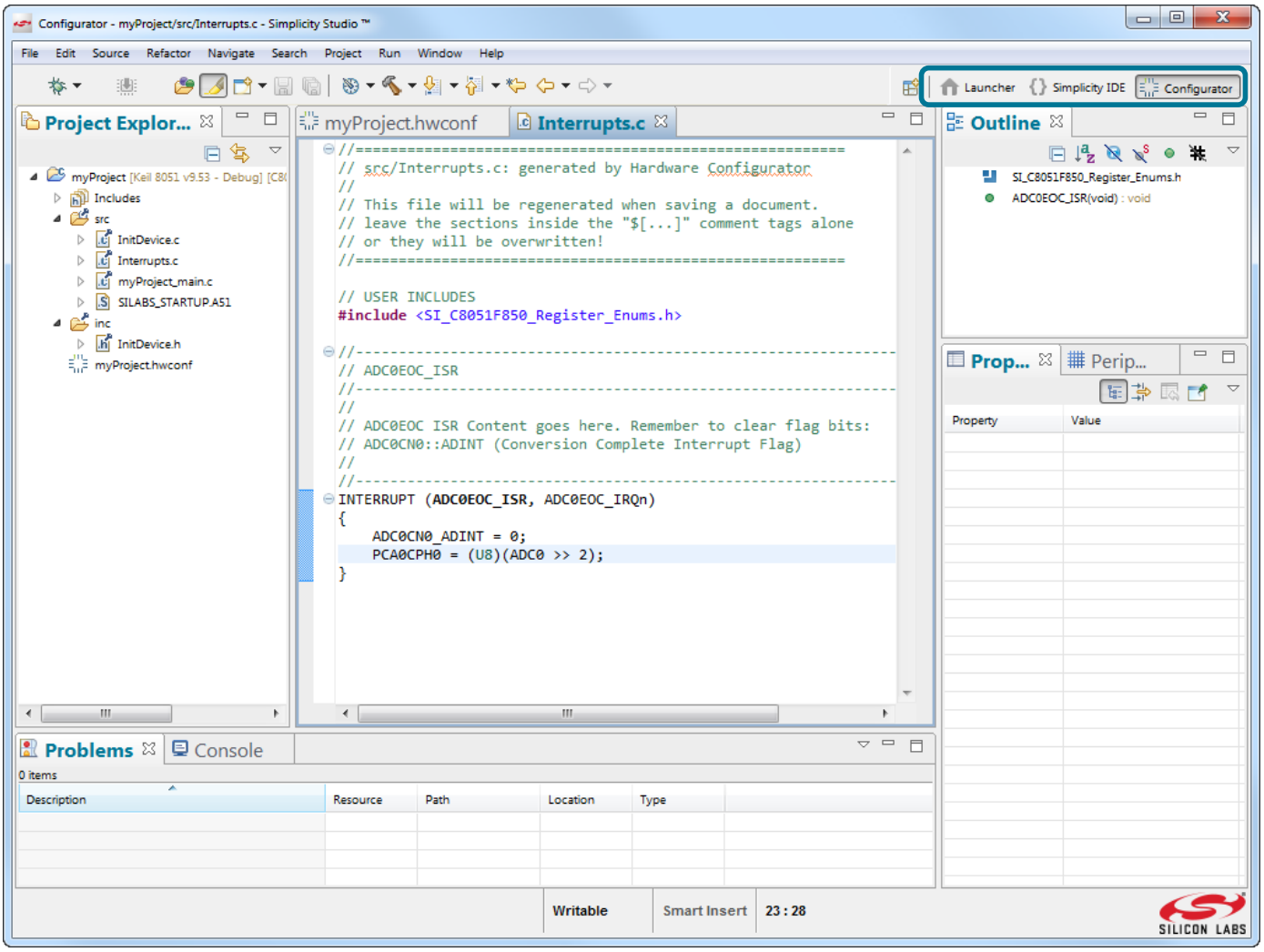


Figure 11.1. Perspectives Within the Simplicity IDE

11.2 Workspaces

A workspace in the Simplicity IDE is a grouping of active projects shown in the [**Project Explorer**] view and is tied to a physical location on disk. The workspace contains the top-level IDE settings, including the global defaults, view window positions, and the projects in the workspace. The workspace information is contained in the metadata (.metadata) subdirectory in the workspace directory. Any projects added to the workspace will be copied to this location. The default workspace location can be seen and modified by going to [**File**]>[**Switch Workspace**]>[**Other**].

11.3 Preferences

Eclipse is a highly customizable application framework and the settings the user applies are called "Preferences". If you want to change the way that Simplicity Studio looks or acts, there is a good chance that there is a preference that allows that change. The Preference menu is accessed by either clicking the "Settings" icon (Green Gear) on the Launcher perspective or from the main menu [Window] > [Preferences] (on a Mac this would be [Studio] > [Preferences]). Examples of Preferences that are available are:

- [Window] > [Preferences] > [General] > [Workspace] and the "Show workspace path in window title". This option is very useful if you work with multiple workspaces
- [Window] > [Preferences] > [C/C++] > [Code Style] > [Formatter] This preference dialog allows setting the color and appearance of code in the editor:

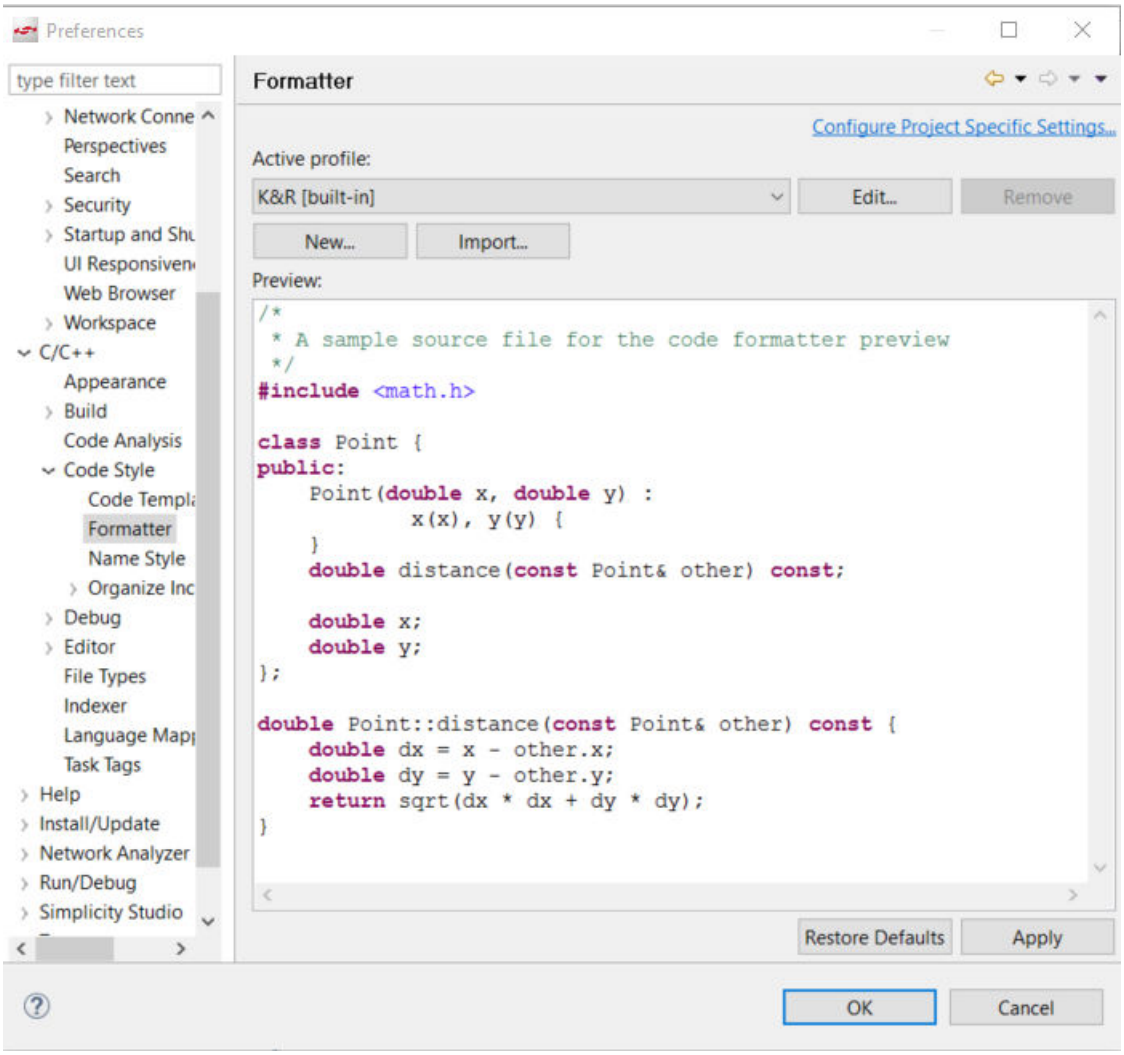


Figure 11.2. Code Style Formatter Preferences

- [Window] > [Preferences] > [Simplicity Studio] > [SDKs] This preference dialog shows a list of the SDKs that are installed in Simplicity Studio and selecting one from the list displays various properties about the SDK including supported parts. Also SDKs that are installed external to Simplicity Studio can be added on this preference page.

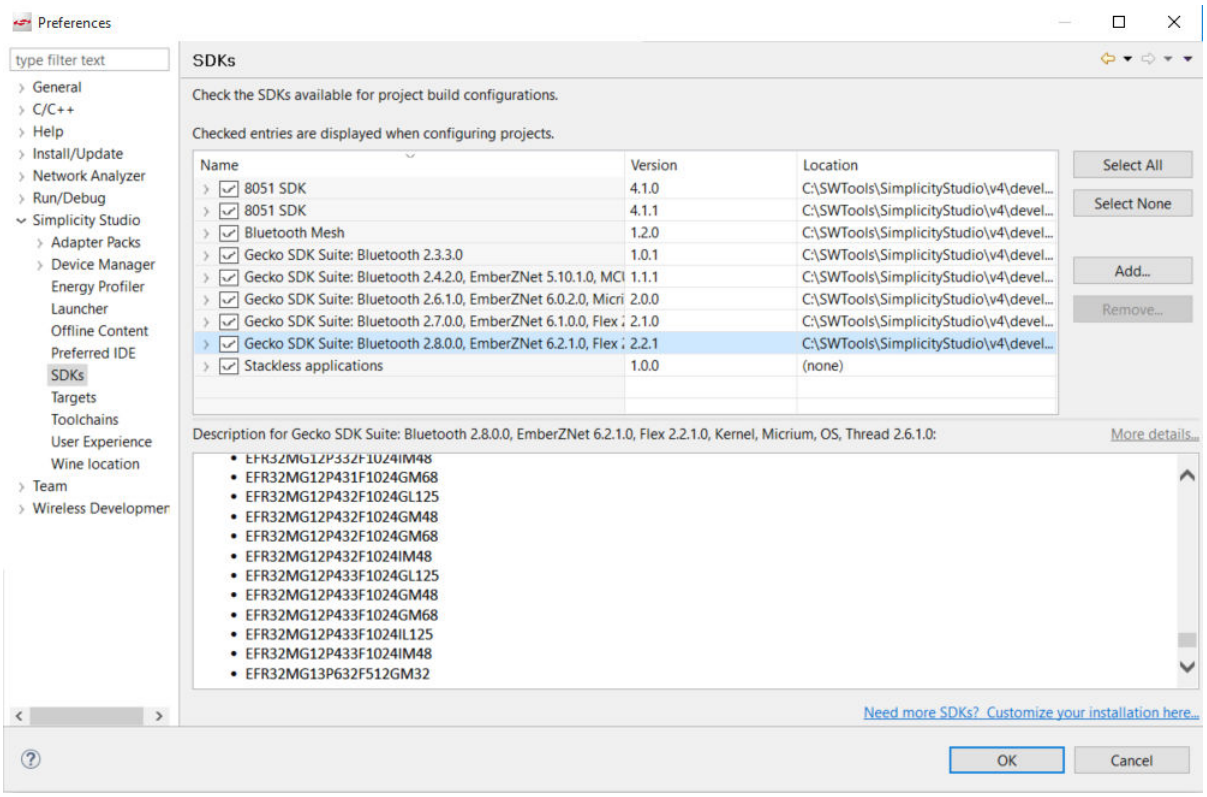


Figure 11.3. Simplicity Studio SDKs Preferences

12. Where to Next?

The following table lists the starting document to refer to for the various Silicon Labs products that Simplicity Studio supports.

Product Family	Document	Link	Main Perspective
EFM8 and C8051	*AN0821: Simplicity Studio C8051F85x Walkthrough*	https://www.silabs.com/documents/public/application-notes/AN0821-simplicity-studio-c8051f85x-walkthrough.pdf	Simplicity IDE
EFM32 Series 0*	*AN0009.0: Getting Started with EFM32 and EZR32 Series 0*	https://www.silabs.com/documents/public/application-notes/an0009.0-efm32-ezr32-series-0-getting-started.pdf	Simplicity IDE
EFM32 Series 1**	*AN0009.1: Getting Started with EFM32 and EFR32 Series 1*	https://www.silabs.com/documents/public/application-notes/an0009.1-efr32-efm32-series-1-getting-started.pdf	Simplicity IDE
EFR32BG	*QSG108: Getting Started with Silicon Labs Bluetooth® Software*	https://www.silabs.com/documents/login/quick-start-guides/qsg108-bluetooth-software-getting-started.pdf	Simplicity IDE
EFR32FG	*QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32™) Portfolio*	https://www.silabs.com/documents/public/quick-start-guides/qsg138-flex-efr32.pdf	Application Builder
EFR32MG	*QSG106: Getting Started with EmberZNet PRO* *QSG113: Getting Started with Silicon Labs Thread*	https://www.silabs.com/documents/public/quick-start-guides/qsg106-efr32-zigbee-pro.pdf https://www.silabs.com/documents/public/quick-start-guides/qsg113-efr32-thread.pdf	Application Builder
Interface	*AN721: USBXpress™ Device Configuration and Programming Guide*	https://www.silabs.com/documents/public/application-notes/AN721.pdf	Xpress Configurator

* Note: MCU Series 0 consists of the following:
- EFM32 Gecko (EFM32G)
- EFM32 Giant Gecko (EFM32GG)
- EFM32 Happy Gecko (EFM32HG)
- EFM32 Wonder Gecko (EFM32WG)
- EFM32 Leopard Gecko (EFM32LG)
- EFM32 Tiny Gecko (EFM32TG)
- EFM32 Zero Gecko (EFM32ZG)

Wireless MCU Series 0 consists of the following:
- EZR32 Wonder Gecko (EZR32WG)
- EZR32 Leopard Gecko (EZR32LG)
- EZR32 Happy Gecko (EZR32HG)

**Note: MCU Series 1 consists of the following:
- EFM32 Jade Gecko (EFM32JG1/EFM32JG12)
- EFM32 Pearl Gecko (EFM32PG1/EFM32PG12
- EFM32 Giant Gecko (EFM32GG11)
- EFM32 Tiny Gecko (EFM32TG11)

Wireless SoC Series 1 Consists of the following:
- EFR32 Blue Gecko (EFR32BG1/EFR32BG12/EFR32BG13/EFR32BG14)
- EFR32 Flex Gecko (EFR32FG1/EFR32FG12/EFR32FG13/EFR32FG14)
- EFR32 Mighty Gecko (EFR32MG1/EFR32MG12/EFR32MG13/EFR32MG14)

13. Other Tools

More information on the other tools available in Simplicity Studio can be found at http://www.silabs.com/simplicity-studio.

14. Revision History

Revision 0.6

December, 2023
- Upated title on first page.
- Added note to front page.

Revision 0.5

June, 2018
- Updated Sections 1 and 2 for revised Launcher views.

Revision 0.5

June, 2018
- Updated Sections 1 and 2 for revised Launcher views.

Revision 0.4

April, 2018
- Updated overall document flow and screenshots and procedures for current version of Simplicity Studio v4.

Revision 0.3

June, 2016
- Updated screenshots and procedures for Simplicity Studio v4.

Revision 0.2

February, 2015
- Updated formatting.
- Updated screenshots for Simplicity Studio v3.

Revision 0.1

February, 2014
- Initial revision.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

Disclaimer
Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.
Note: This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit www.silabs.com/about-us/inclusive-lexicon-project

Trademark Information
Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals®, WiSeConnect , n-Link, ThreadArch®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, Precision32®, Simplicity Studio®, Telegesis, the Telegesis Logo®, USBXpress® , Zentri, the Zentri logo and Zentri DMS, Z-Wave®, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com