RSL15 Getting Started Guide

M-20873-004
June 2023

© SCILLC, 2023
Previous Edition © 2021
"All Rights Reserved"

Table of Contents

Page

RSL15 Getting Started Guide 1

Table of Contents 2

1. Introduction 4

1.1 Summary 4

1.2 Document Conventions 4

1.3 Further Reading 5

2. Using RSL15 Central Mobile App to Establish a Bluetooth Low Energy Connection 7

3. Setting Up the RSL15 Evaluation and Development Board 8

3.1 Configuring and Connecting Hardware 8

4. Getting Started with the onsemi IDE 11

4.1 Software to Download 11

4.2 onsemi IDE Installation Procedure 11

4.3 RSL15 CMSIS-Pack Installation Procedure 11

4.4 Importing the Sample code 13

4.5 Building the Sample Code 15

4.6 Debugging the Sample Code 17

4.7 Viewing Peripheral Registers with the onsemi IDE 18

4.7.1 The Peripheral Registers View Plugin 18

4.7.2 The EmbSys Register View Plugin 21

5. Getting Started with Keil® 24

5.1 Software to Download 24

5.2 Keil Installation Procedure 24

5.3 RSL15 CMSIS-Pack Installation Procedure 24

5.4 Importing the Sample Code 26

5.5 Building the Sample Code 27

5.6 Debugging the Sample Code 29

5.6.1 Preparing J-Link for Debugging 29

5.6.2 Debugging Applications 29

6. Getting Started with IAR 32

6.1 Prerequisite Software: Downloading and Installation 32

6.2 RSL15 CMSIS-Pack Installation Procedure 32

6.3 Building Your First Sample Application with the IAR Embedded Workbench 34

6.3.1 Import the Sample Code 34

6.3.2 Building the Sample Code 37

6.3.3 Debugging the Sample Code 38

6.3.3.1 Debugging Applications 38

7. Resolving External CMSIS-Pack Dependencies 40

7.1 External CMSIS_Pack Dependencies 40

7.2 Resolving External Dependencies 40

8. Accessing Documentation Included with the CMSIS-Pack 43

8.1 Accessing Documentation through the onsemi IDE 43

8.2 Accessing Documentation via the Keil µVision IDE 44

CHAPTER 1

Introduction

1.1 Summary

IMPORTANT: onsemi acknowledges that this document might contain the inappropriate terms “white list", "master" and "slave”. We have a plan to work with other companies to identify an industry wide solution that can eradicate non-inclusive terminology but maintains the technical relationship of the original wording. Once new terminologies are agreed upon, future products will contain new terminology.

This group of topics describes how to begin using the RSL15 Central mobile app and the RSL15 Software Development Kit (SDK) with the RSL15 Evaluation and Development Board (EVB). It provides the prerequisites and instructions necessary to install the relevant software, connect the EVB, and get started.

NOTE: The RSL15 licensing agreement is specified in *Software Use Agreement - use and accept (ONIPLAW 08142020).pdf*, found in the root directory of the installed RSL15 CMSIS-Pack.

NOTE: Admin rights are required to install and run onsemi-provided software, such as the onsemi IDE and BLE Explorer.

1.2 Document Conventions

The following typographical conventions are used in this documentation:

`monospace font`
: Assembly code, macros, functions, registers, defines and addresses.

italics
: File and path names, or any portion of them.

`<angle brackets and bold>`
: Optional parameters and placeholders for specific information. To use an optional parameter or replace a placeholder, specify the information within the brackets; do not include the brackets themselves.

Bold
: GUI items (text that can be seen on a screen).

Note, Important, Caution, Warning

Information requiring special notice is presented in several attention-grabbing formats depending on the consequences of ignoring the information:

NOTE: Significant supplemental information, hints, or tips.

IMPORTANT: Information that is more significant than a Note; intended to help you avoid frustration.

CAUTION: Information that can prevent you from damaging equipment or software.

WARNING: Information that can prevent harm to humans.

Registers:

Registers are shown in `monospace font` using their full descriptors, depending on which core the register is accessing. The full description takes the form **`<PREFIX><GROUP>_<REGISTER>`**.

All registers are accessible from the Arm Cortex-M33 processor.

A register prefix of `D_` is used in the following circumstances:

- In cases where there are multiple instances of a block of registers, the summary of the registers at the beginning of the Register section have slightly different names from the detailed register sections below that table. For example, the `DMA*_CFG0` registers are referred to as `DMA_CFG0` when we are defining bit-fields and settings.

The firmware provides access to these registers in two ways:

- In the flat header files (e.g.: *sk5_hw_flat_cid*.h*), each register is individually accessible by directly using the naming provided in this manual. This is helpful for assembly and low-level C programming.
- In the normal header files (e.g.: *sk5_hw_cid*.h*), each register group forms a structure, with the registers being defined as members within that structure. The structures defined by these header files provide access to registers under the naming conventions `PREFIX_GROUP->REGISTER` (for the structure) and `GROUP->REGISTER` (for the register).
- For more information, see the Hardware Definitions chapter of the *RSL15 Firmware Reference*.

Default settings for registers and bit fields are marked with an asterisk (*).

Any undefined bits must be written to 0, if they are written at all.

Numbers

In general, numbers are presented in decimal notation. In cases where hexadecimal or binary notation is more convenient, these numbers are identified by the prefixes "0x" and "0b" respectively. For example, the decimal number 123456 can also be represented as 0x1E240 or 0b11110001001000000.

Sample Rates

All sample rates specified are the final decimated sample rates, unless stated otherwise.

1.3 Further Reading

The following documents are installed with the RSL15 system, in the default location *C:/Users/***`<your_user_name>`***/AppData/Local/Arm/Packs/ONSemiconductor/RSL15/***`<version_number>`***/documentation*. These manuals are available only in PDF format:

- *Arm TrustZone CryptoCell-312 Software Developers Manual*
- multiple CEVA manuals in the */ceva* folder

For even more information, consult these publicly-available documents:

- *Armv8M Architecture Reference Manual* (PDF download available from https://developer.arm.com/documentation/ddi0553/latest).

- *Arm Cortex-M33 Processor Technical Reference Manual*, revision r1p0, from https://developer.arm.com/documentation/100230/0100
- *Bluetooth Core Specification version 5.2*, available from https://www.bluetooth.com/specifications/adopted-specifications
- TrustZone documentation available from the Arm website at https://developer.arm.com/ip-products/security-ip/trustzone/trustzone-for-cortex-m
- Other ArmCortex-M33 publications, available from the Arm website at https://developer.arm.com/ip-products/processors/cortex-m/cortex-m33

For information about the Evaluation and Development Board Manual and its schematics, go to the RSL15 web page and navigate to the EVB page.

CHAPTER 2

Using RSL15 Central Mobile App to Establish a Bluetooth Low Energy Connection

The RSL15 Evaluation and Development Board (EVB) comes initially programmed with the *ble_peripheral_server* sample. This code allows the RSL15 EVB to establish a Bluetooth Low Energy connection to the RSL15 Central mobile app. RSL15 Central is available for Android at Google Play and iOS at the App Store.

Your RSL15 EVB must have the *ble_peripheral_server* sample loaded to connect to the RSL15 Central mobile app. If the *ble_perhipheral_server* code is modified or removed, it can be restored by following the instructions in this guide to load the *blinky* sample app with your preferred IDE, but load the *ble_peripheral_server* sample instead. The RSL15 Central app scans and establishes a Bluetooth Low Energy connection with the RSL15 EVB. After establishing the connection, the app displays the battery level and button state (SW1) from the RSL15 EVB in real-time. It can also control the LED state on the RSL15 EVB by toggling the LED switch in the app.

Simply power the RSL15 EVB by connecting the USB cable and use the RSL15 Central app to connect.

The "Example screen shots of the Central Mobile App" figure (Figure 1) illustrates the functionality of RSL15 Central:

1. Scan for available Bluetooth Low Energy devices, and select one.
2. Use the Settings screen to filter the Bluetooth Low Energy devices and check the app version.
3. Use the device detail screen to receive battery and button state notifications, and use the LED switch to toggle the LED on the RSL15 EVB.

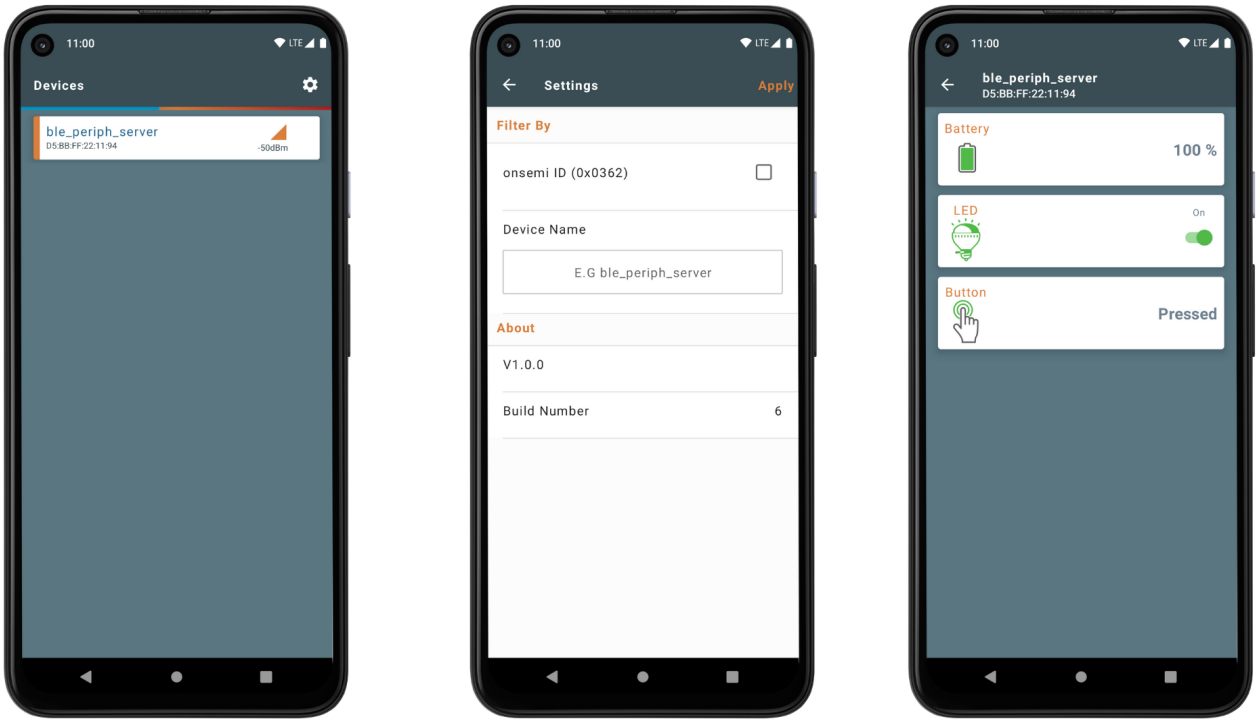

Figure 1. Example screen shots of the Central Mobile App

CHAPTER 3

Setting Up the RSL15 Evaluation and Development Board

This topic shows how to set up the RSL15 Evaluation and Development Board (EVB) to get started.

3.1 Configuring and Connecting Hardware

Verify that the RSL15 EVB has the default jumper configuration, which is VBAT and VOUT pins jumpered on the VBAT-SEL header, and VBAT and VDDO pins jumpered on the VDDO-EN header. See the "EVB with Default Jumper Configuration" figure (Figure 2). The red boxes indicate default jumper positions.

IMPORTANT: Ensure that all of the DIP switches on the VDDO-EN header and the DEBUG-EN switches are in the default ON position, as shown in the "EVB with Default Jumper Configuration" figure (Figure 2).

Figure 2. EVB with Default Jumper Configuration

Once the jumpers are in the correct positions, you can plug the USB cable into the EVB. When powered, the blue LED flashes rapidly (it flashes slower when connected to the RSL15 Central mobile app). See the *RSL15 Evaluation and Development Board User's Guide* for more power options available for the RSL15 EVB.

CHAPTER 4

Getting Started with the onsemi IDE

This topic takes you through the steps for installing and setting up the onsemi IDE and the RSL15 CMSIS-Pack, importing and building sample code, and debugging your first application.

4.1 Software to Download

From www.onsemi.com/RSL15, download:

1. The onsemi IDE Installer
2. RSL15 Firmware Package which contains the RSL15 CMSIS-Pack and Release Notes.

4.2 onsemi IDE Installation Procedure

The onsemi IDE installer comes bundled with the SEGGER® J-Link® installer. J-Link version 7.86c is required for RSL15.

For instructions on installing the onsemi IDE, see *onsemi_IDE_release_notes.pdf* in the *ONSEMI IDE INSTALLER* folder of the unzipped *onsemi IDE Installer* download, available at www.onsemi.com/rsl15.

4.3 RSL15 CMSIS-Pack Installation Procedure

1. Extract the RSL15 Firmware package to a temporary folder.
2. Create a new workspace at, for example, *C:\workspace*, using the onsemi Launcher. Open the onsemi IDE by going to the Windows Start menu and selecting **onsemi** > **onsemi IDE**. From the onsemi IDE Launcher screen, edit the suggested workspace name to a new name if needed, and click **Launch**.

NOTE: We recommend creating a new workspace for each new version of the IDE to ensure compatibility.

3. On the top row of the Workbench perspective, click the "Make the CMSIS Packs manager perspective visible" icon. (See the "Opening the CMSIS-Pack Manager Perspective" figure (Figure 3).)

NOTE: If you have trouble opening the CMSIS-Pack manager perspective, re-install the IDE in your user folder (i.e., *C:\Users*`<user_name>`).

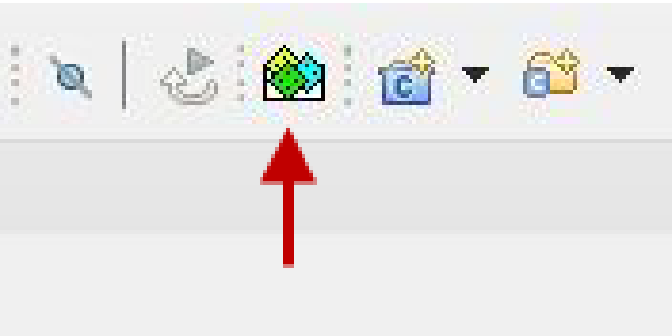

Figure 3. Opening the CMSIS-Pack Manager Perspective

4. Click the **Import Existing Packs icon**, select your pack file (*ONSemiconductor.RSL15.*`<version>`*.pack)*, where `<version>` is a number such as *1.0.0-464*, and click **Open**. (See the "Installing the RSL15 CMSIS-Pack" figure (Figure 4).)

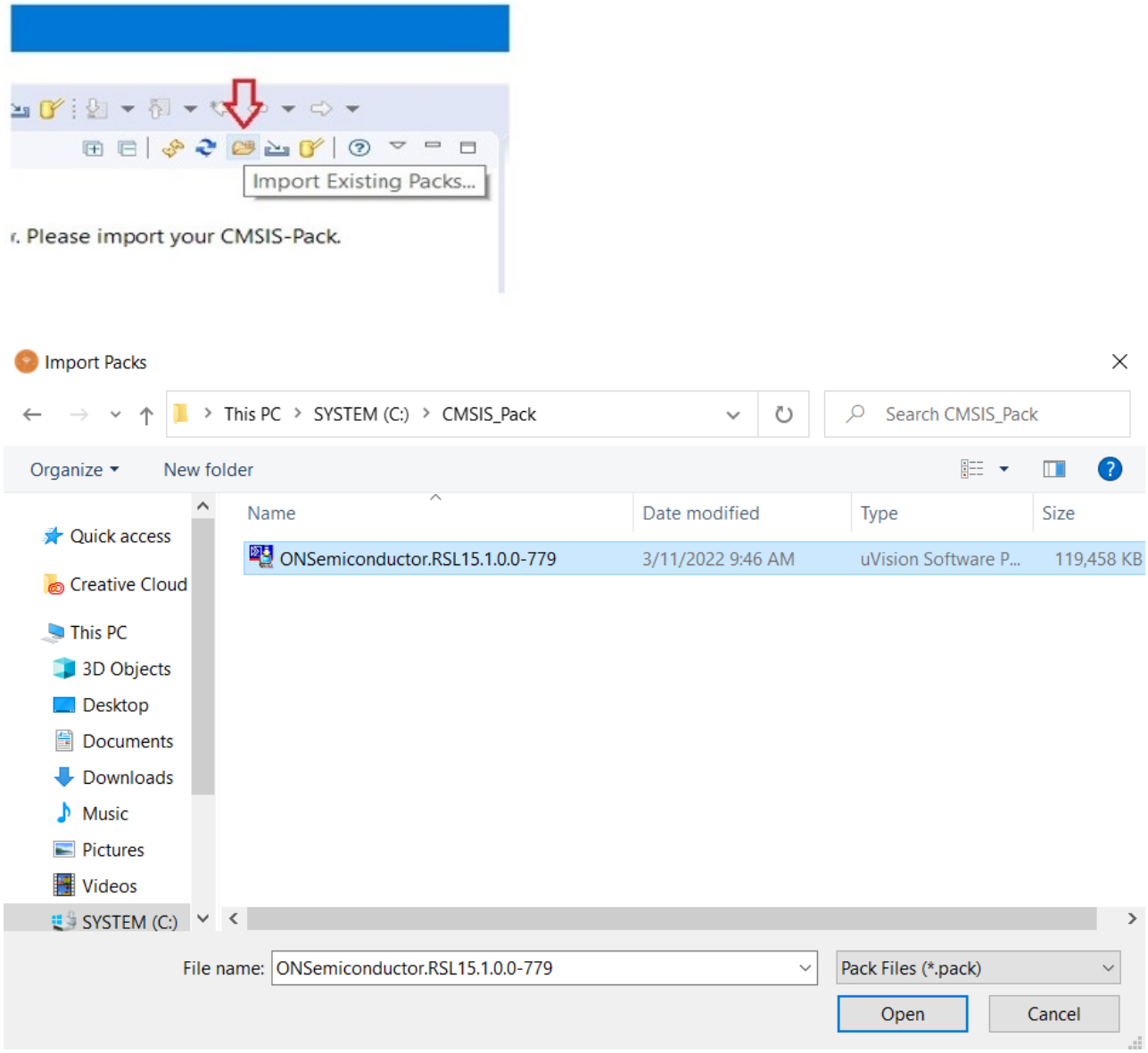

Figure 4. Installing the RSL15 CMSIS-Pack

NOTE: If this is the first time an onsemi CMSIS-Pack is being installed on your system, you are likely to see the following error message:

The CMSIS-Pack root folder "C:/Users/<user_id>/AppData/Local/Arm/Packs" is empty. Please import the CMSIS-Pack.

To resolve this, click the icon shown in the "Reload Packs in the CMSIS-Pack root folder" figure (Figure 5).

Figure 5. Reload Packs in the CMSIS-Pack root folder

NOTE: The default CMSIS-Pack location is:

C:\Users`<user_id>`*\AppData\Local\Arm\Packs\ONSemiconductor\RSL15*`<version>`*\

where `<user_id>` is your userid on your system, and `<version>` is the RSL15 version number.

5. The RSL15 CMSIS-Pack now appears in the list of installed packs. In the **Devices** tab, if you expand **All Devices** > **ONSemiconductor** > **RSL15 Series**, you can see **RSL15** listed there. You can manage your installed packs in the **Packs** tab. Expanding **ONSemiconductor** > **RSL15** and selecting the device makes the **Pack Properties** tab display the details of the RSL15 CMSIS-Pack (see the "Pack Manager Perspective after RSL15 CMSIS-Pack is Installed" figure (Figure 6)). Selecting the device in the Devices tab also displays the RSL15 information in **Pack Properties**.

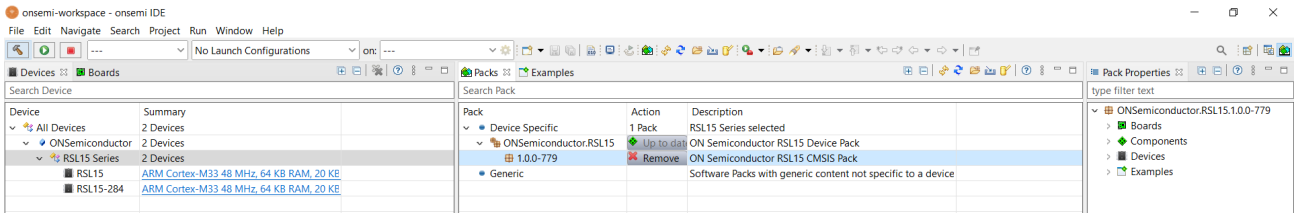

Figure 6. Pack Manager Perspective after RSL15 CMSIS-Pack is Installed

4.4 Importing the Sample code

Import the sample as follows:

1. In the Pack Manager perspective, click on the **Examples** tab to list all the example projects included in the RSL15 CMSIS-Pack. (See the "Pack Manager Perspective: Examples Tab" figure (Figure 7).)
2. Choose the example project called **Blinky**, and click the **Copy** button to import it into your workspace.

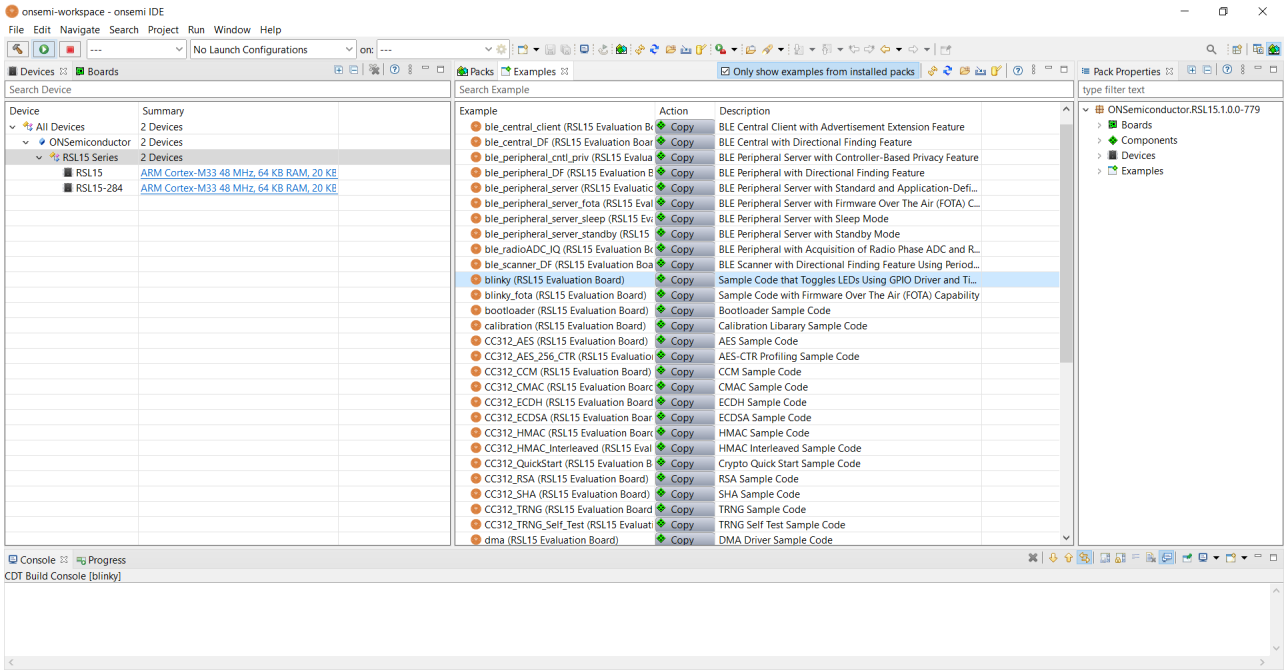

Figure 7. Pack Manager Perspective: Examples Tab

3. The C/C++ perspective opens and displays your newly copied project. In the **Project Explorer** panel, you can expand your project folder and explore the files inside your project. Double-click on the file names to open them. In the Components view, the *blinky.rteconfig* file displays software components. If you expand **Device** > **Libraries**, you can see the **HAL** library and the **Startup** components selected for *blinky*. (See the "RTE Configuration for the Blinky Example Project in the onsemi IDE" figure (Figure 8).)

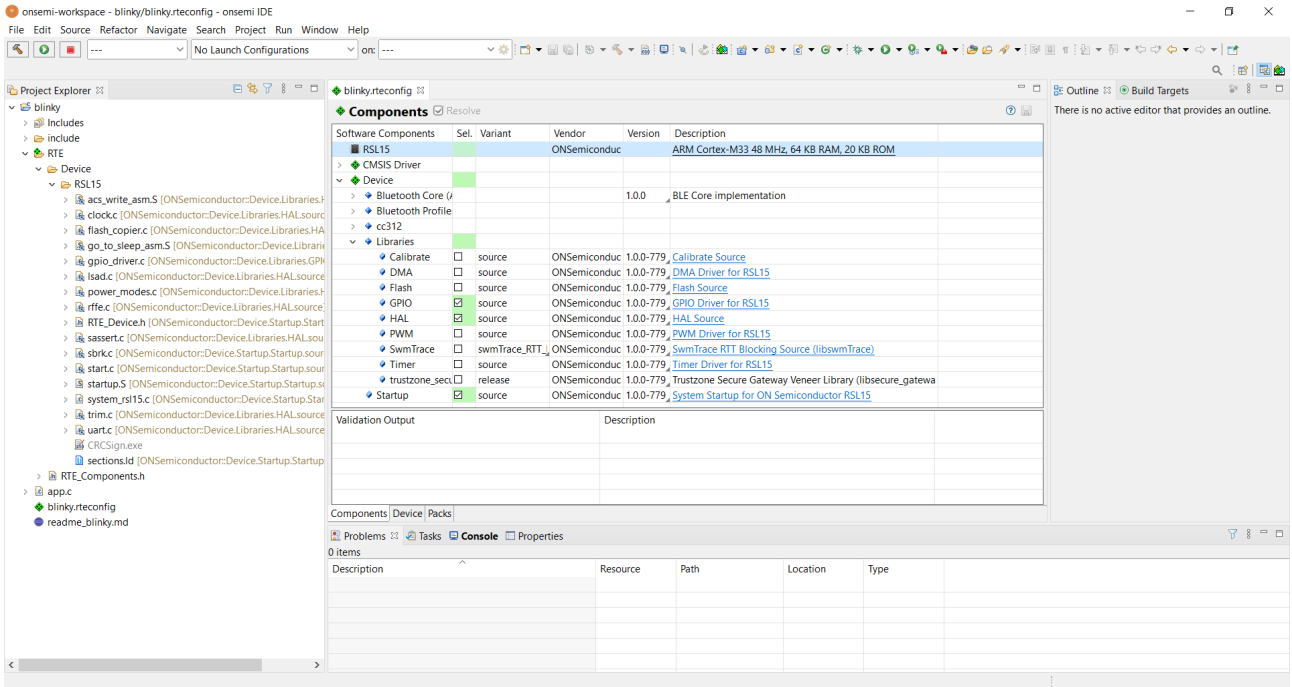

Figure 8. RTE Configuration for the Blinky Example Project in the onsemi IDE

4.5 Building the Sample Code

Follow these steps to build the sample code:

1. Right click on the folder for *blinky* and click **Build Project**. Alternatively, you can select the project and click the **Build Project icon**, which looks like a hammer, as shown in the "Beginning to Build a Project in the onsemi IDE" figure (Figure 9).

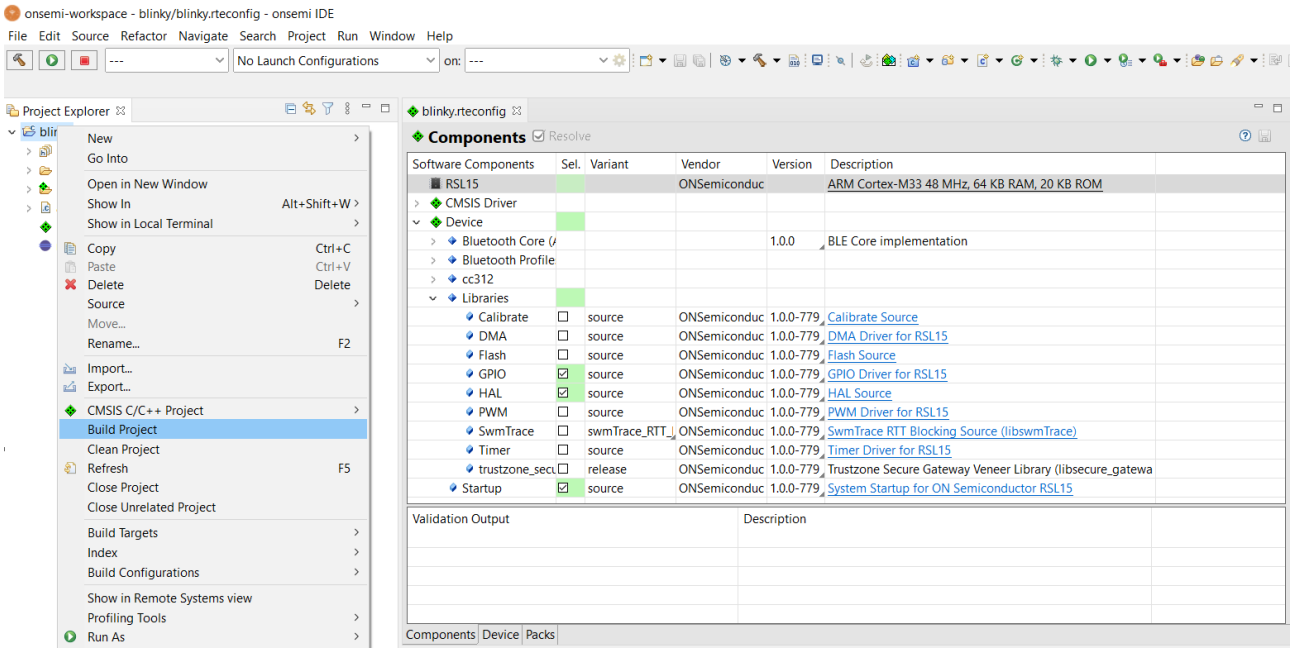

Figure 9. Beginning to Build a Project in the onsemi IDE

2. When the build is running, the output of the build is shown in the onsemi IDE C/C++ Development Tooling (CDT) Build Console, as illustrated in the "Example of Build Output" figure (Figure 10).

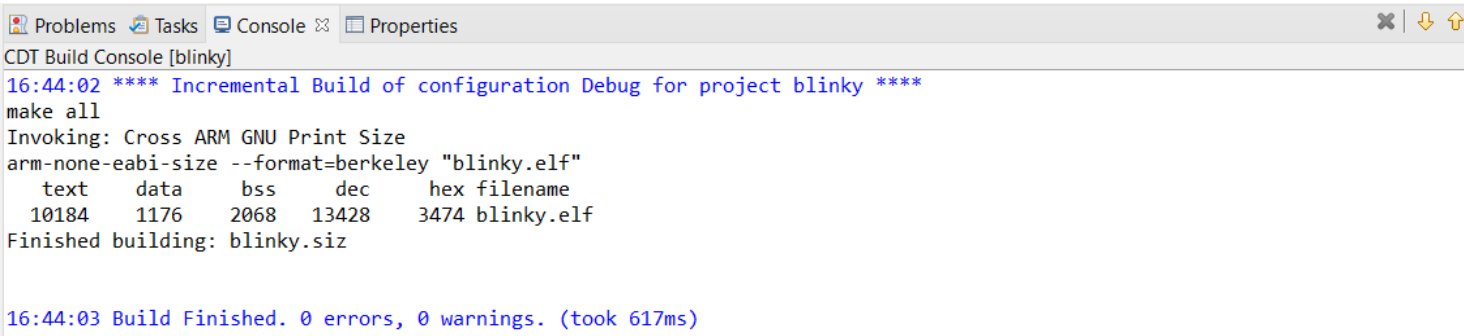

Figure 10. Example of Build Output

3. The key resulting output in Project Explorer, in the **Debug** folder, includes:

- *blinky.hex*: hex file for loading into flash memory
- *blinky.elf*: Arm® executable file, run from RAM, used for debugging
- *blinky.map*: map file of the sections and memory usage

These files are shown in the "Output Files from Building a Sample Project" figure (Figure 11).

NOTE: You might need to refresh the project to see the three built output files. To do so, right-click on the project name *blinky* and choose **Refresh** from the menu.

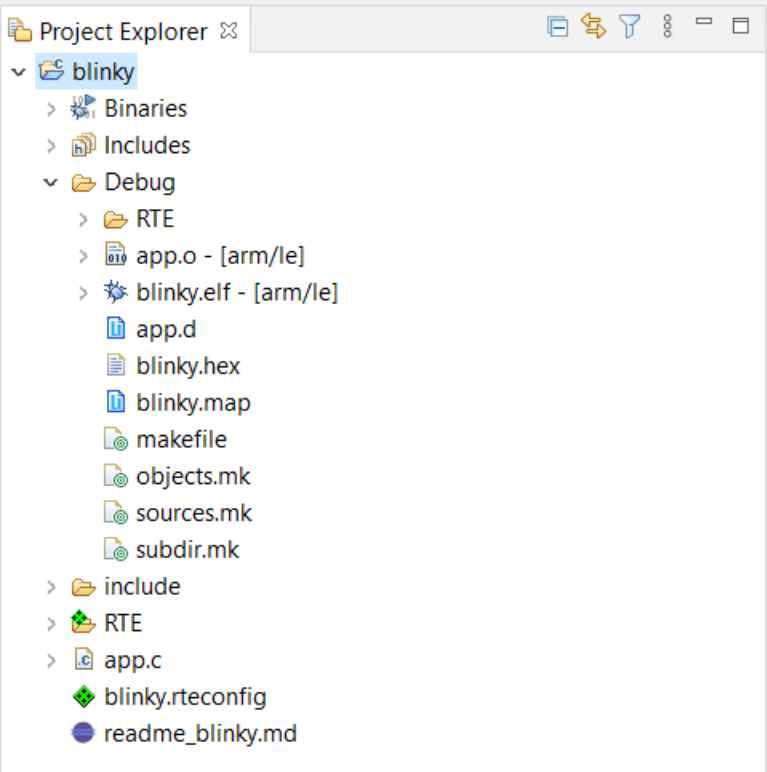

Figure 11. Output Files from Building a Sample Project

4.6 Debugging the Sample Code

Debug the application as follows:

1. From the **Launch Mode** dropdown menu, select **Debug**. From the **Launch Configuration** dropdown menu, select **blinky Debug**. Both are shown in the "Launching a Debug Session" figure (Figure 12)).
2. Press the **Launch in Debug** button (also shown in the "Launching a Debug Session" figure (Figure 12)) to launch the debug session. J-Link downloads the *blinky* sample code to RSL15's flash memory.

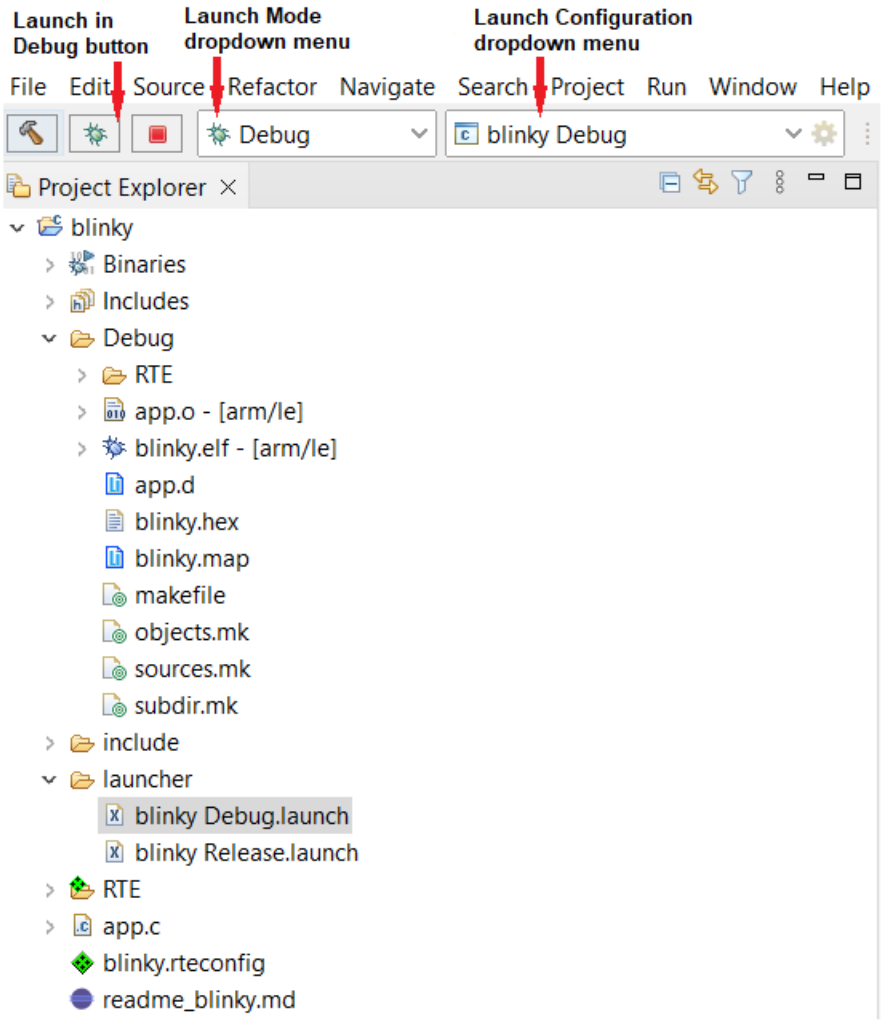

Figure 12. Launching a Debug Session

3. The Debug perspective opens and the application runs up to the first breakpoint in `main`. Press F8 or click the Resume icon ( ) to resume the execution of the application. In the RSL15 EVB, you can see that a green and a blue LED are both blinking, at different rates. When push-button SW1 is pressed, the blinking of these two LEDs is disabled or enabled.

4.7 Viewing Peripheral Registers with the onsemi IDE

4.7.1 The Peripheral Registers View Plugin

The onsemi IDE includes a peripheral register view plugin that enables you to visualize and modify all of the RSL15 registers during a debug session.

The following steps demonstrate how touse the Peripheral Registers View with the *Blinky* application:

1. In the Debug perspective, when the application runs up to the first breakpoint in main, open the Peripherals window view by navigating to **Window** > **Show View** > **Other** > **Debug** > **Peripherals** and clicking **Open**. Now you can see all the RSL15 peripherals displayed.
2. In the Peripherals window, select **GPIO**. Open the **Memory** window to monitor the RSL15 peripheral. Read only registers are highlighted in green. It is a good idea to drag your Memory window and place it side-by-side with your source code view (see the "Peripheral Registers View Perspective in Debug Session After Setting the SVD Path" figure (Figure 13)) to prevent the console from switching focus away from the Memory window.

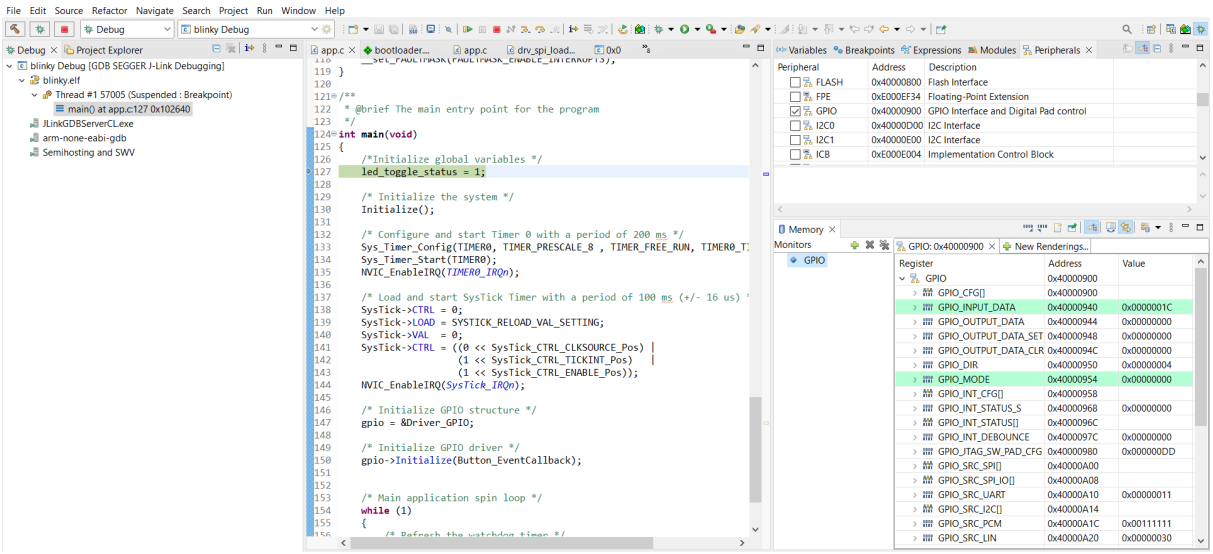

Figure 13. Peripheral Registers View Perspective in Debug Session After Setting the SVD Path

3. To see or change the GPIO register status, choose **GPIO** and expand the **GPIO** > **GPIO_OUTPUT_DATA** register in the Memory window.
4. Add a breakpoint at this line of code:
   `gpio->ToggleValue((GPIO_SEL_t)(SYSTICK_STATES_GPIO))`

   This breakpoint prevents the application from getting stuck in the main loop while you step through.

5. Press F8 and observe that the register bits 8 and 10 toggle their state when `gpio->ToggleValue((GPIO_SEL_t)(SYSTICK_STATES_GPIO))` and `gpio->ToggleValue((GPIO_SEL_t)(TIMER0_STATES_GPIO))` are executed (in this case, from 0x400 to 0x100). The register turns yellow to indicate that you have activated real-time monitoring for it (see the "Toggling RSL15 GPIO Using the Peripheral Registers View: Before" figure (Figure 14)).

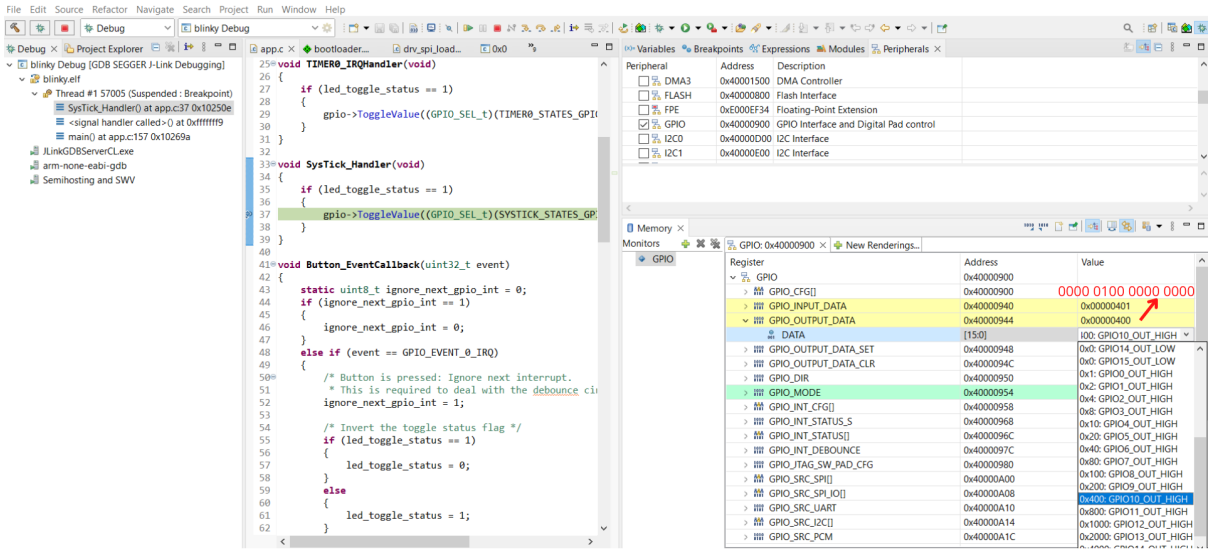

Figure 14. Toggling RSL15 GPIO Using the Peripheral Registers View: Before

6. To manually change the register value, click on the Value field of the GPIO register to change the (HIGH/LOW) state of GPIO8 and GPIO10. The "Toggling RSL15 GPIO Using the Peripheral Registers View: After" figure (Figure 15) illustrates the view after making the change. You can observe that the LED (GPIO8 and GPIO10) on your board changes state.

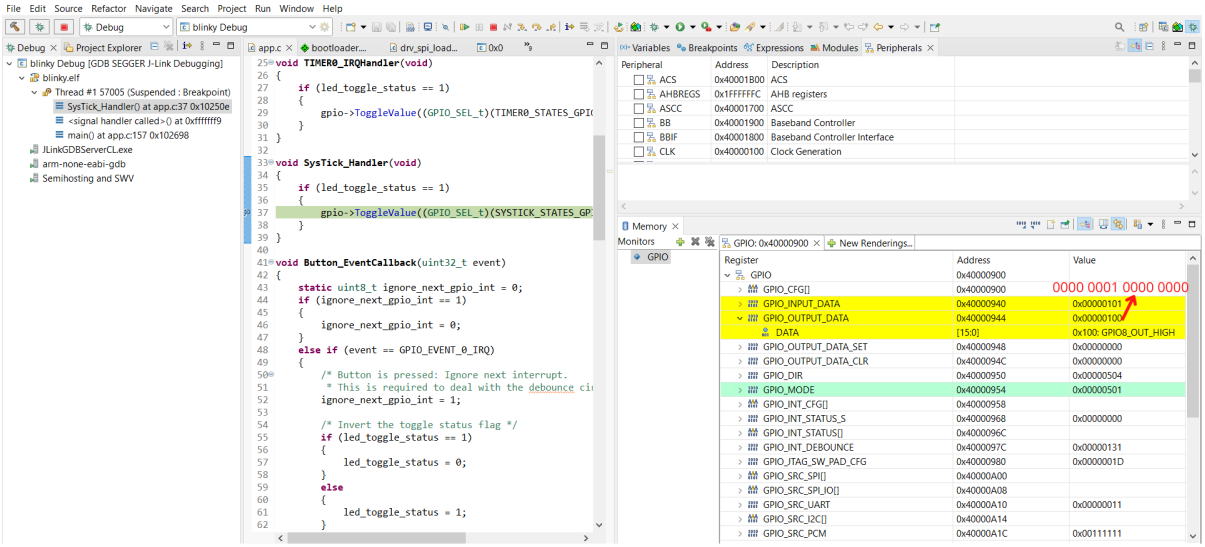

Figure 15. Toggling RSL15 GPIO Using the Peripheral Registers View: After

4.7.2 The EmbSys Register View Plugin

The onsemi IDE includes an EmbSys Register View plugin for viewing RSL15 registers during a debug session. To set up the plugin before using it, perform the following steps:

1. Make sure the latest version of the onsemi IDE is installed.
2. In the IDE, choose **Window** > **Preferences**.
3. Navigate in the left panel to select **C/C++** > **Debug** > **EmbSys Register View**.
4. In the drop-down menus for the following fields, make these selections, as shown in the "Setting up the EmbSys Register View Plugin" figure (Figure 16):
   - **Architecture:** `SVD(CMSIS)`
   - **Vendor:** `ONSemiconductor`
   - **Chip:** `rsl15`
5. Click **Apply and Close**.

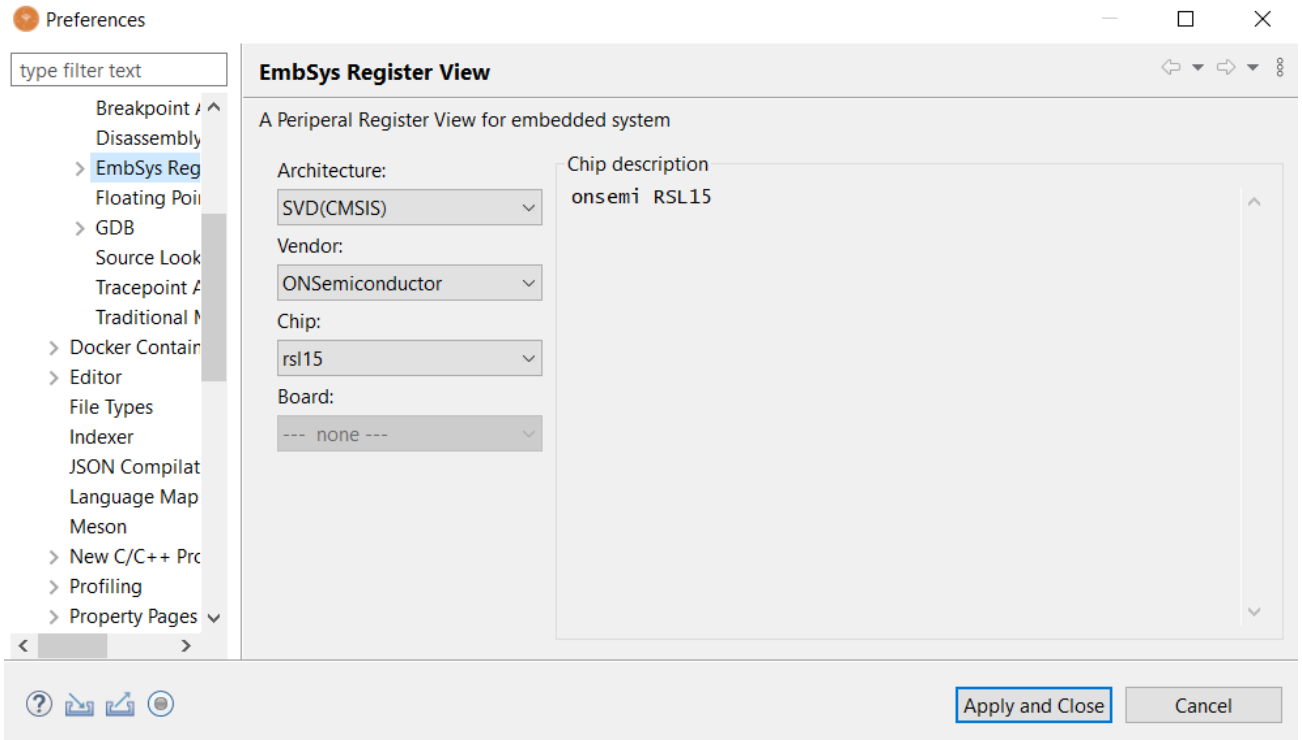

Figure 16. Setting up the EmbSys Register View Plugin

To open the plugin for viewing the registers, perform the following steps:

1. In the IDE, choose **Window** > **Show View** > **Other**.
2. Type `EmbSys` in the filter search.
3. Select **EmbSys Registers** and click **Open**.

The EmbSys Registers View is now ready to be used during a debug session. Any register that needs monitoring must be marked in the plugin's window. To mark a register, double click on it; it turns green to that show it has been selected. Once marked, the register's value appears in the view when a debug session has been suspended.

NOTE: To mark a group of registers at the same time, double click on the parent of a register group; this marks all its child registers for monitoring, as shown in the "Marking a Register Group for Monitoring" figure (Figure 17).

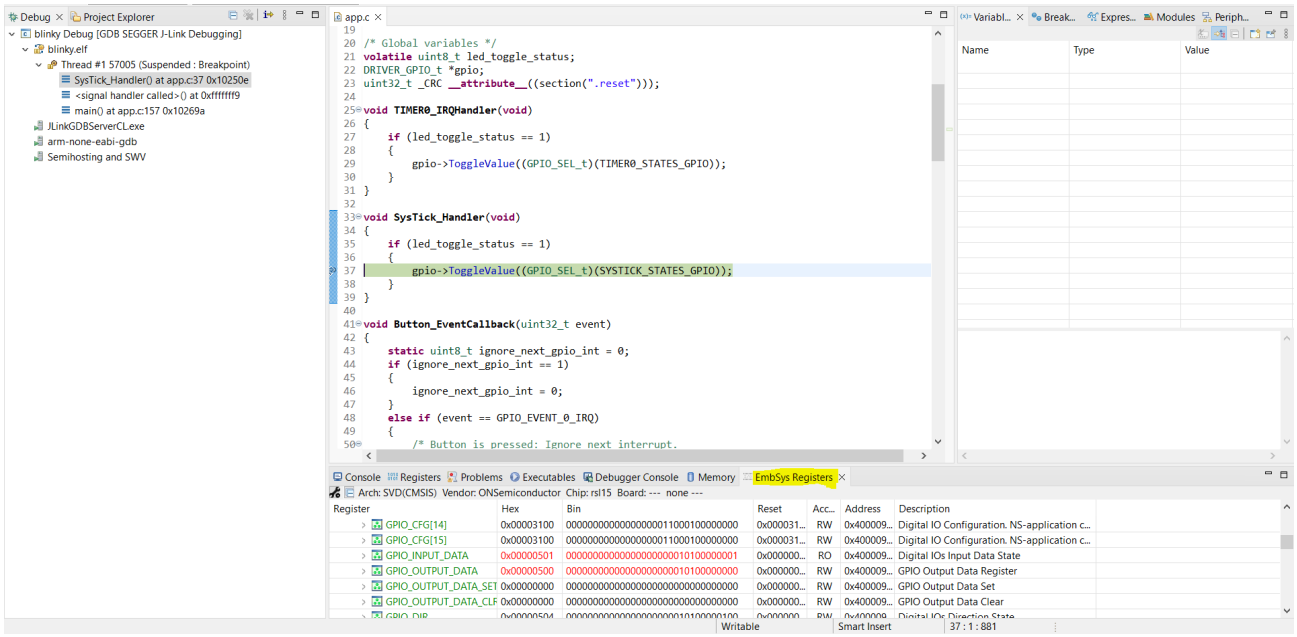

Figure 17. Marking a Register Group for Monitoring

CHAPTER 5

Getting Started with Keil®

This topic shows how you can set up and start using RSL15 with the Arm Keil µVision® IDE.

5.1 Software to Download

Download the following:

1. The Keil µVision IDE from the Keil website.
2. RSL15 Firmware Package, which contains the CMSIS-Pack and Release Notes, from www.onsemi.com/rsl15.

5.2 Keil Installation Procedure

Follow the Keil µVision IDE installation instructions provided by Keil. After the Keil IDE is installed, you need to update the SEGGER J-Link software:

1. Go the Start menu and type `J-Link DLL Updater V`**`<J-Link_version>`**, where **`<J-Link_version>`** is 7.86c or later, and press Enter.
2. Select the **Keil MDK-ARM** checkbox and click **OK**.

The RSL15 device is now available in the Keil IDE.

5.3 RSL15 CMSIS-Pack Installation Procedure

To install the RSL15 CMSIS-Pack:

1. Extract the RSL15 Firmware package to a temporary folder
2. Open the Keil µVision IDE and navigate to **Project** > **Manage** > **Pack Installer** or click on the icon shown in the "Pack Installer Icon" figure (Figure 18).

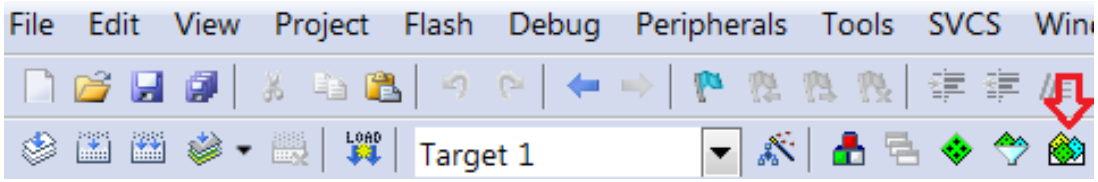

Figure 18. Pack Installer Icon

3. Click on **File** > **Import** (see the "Importing the RSL15 CMSIS-Pack" figure (Figure 19)), select your pack file *ONSemiconductor.RSL15.***`<version>`***.pack*, and click **Open** (see the "Installing the RSL15 CMSIS-Pack for the Keil µVision IDE" figure (Figure 20)). **<version>** is the RSL15 version, such as 2.2.347.

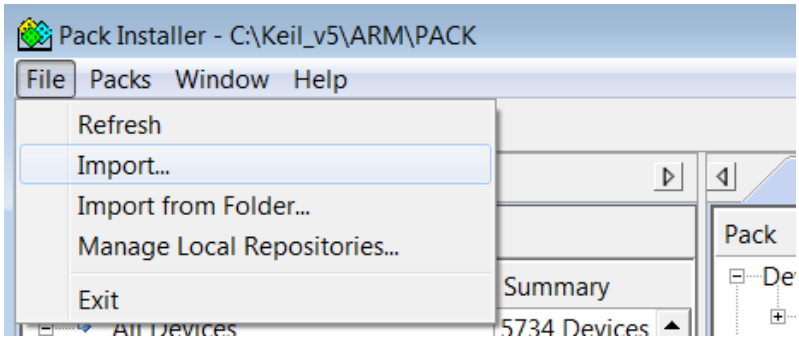

Figure 19. Importing the RSL15 CMSIS-Pack

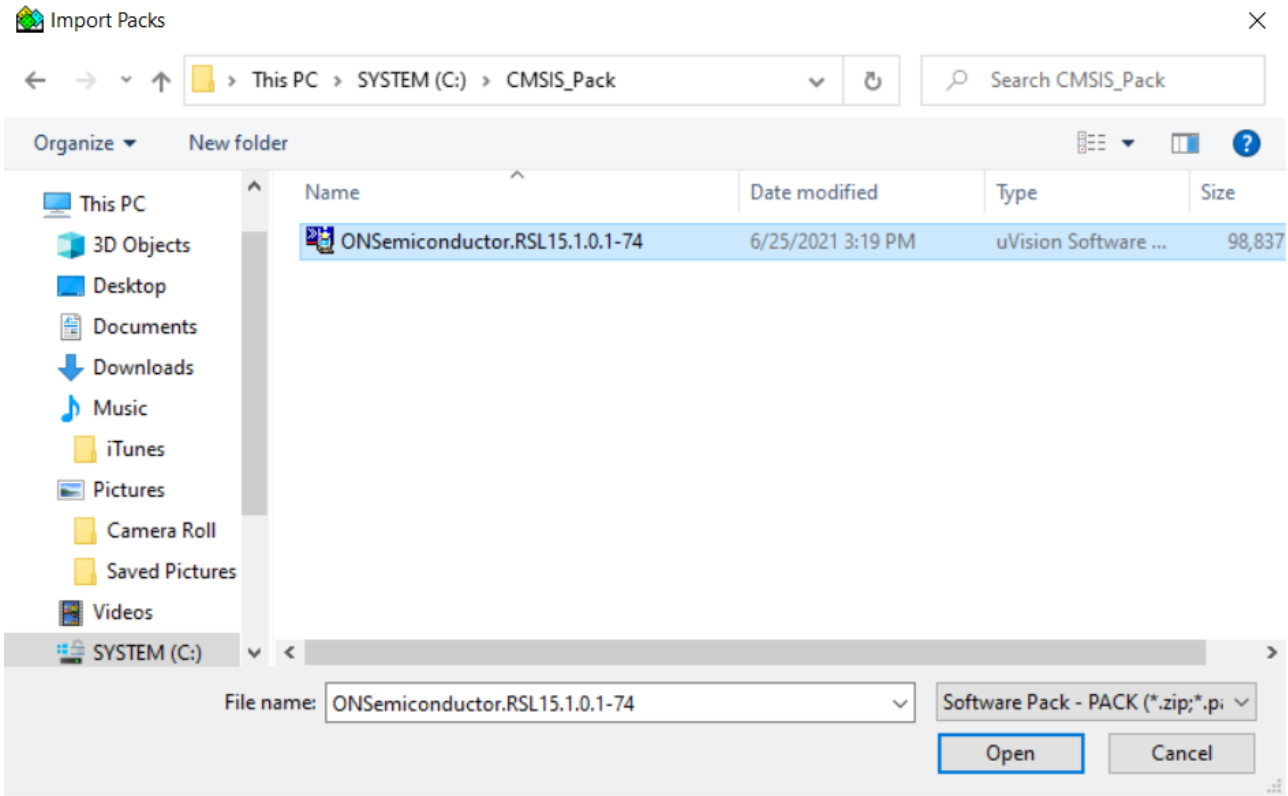

Figure 20. Installing the RSL15 CMSIS-Pack for the Keil µVision IDE

4. The IDE installs the RSL15 CMSIS-Pack in the *%LOCALAPPDATA%\Arm\Packs* folder.
5. After installation, use **File** > **Refresh** as shown in the "Refresh Pack after installation" figure (Figure 21) to update your pack proprieties.

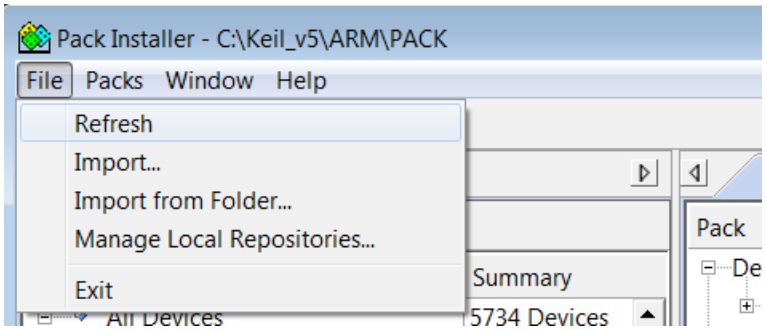

Figure 21. Refresh Pack after installation

6. The RSL15 CMSIS-Pack now appears in the list of installed packs. In the **Devices** tab, if you expand **All Devices** > **ONSemiconductor** > **RSL15 Series**, you can see RSL15 listed there. You can manage your installed packs in the **Packs** tab. Expanding **ONSemiconductor** > **RSL15** makes the **Pack Properties** tab display the details of the RSL15 CMSIS-Pack. the "Pack Installer after RSL15 CMSIS-Pack is Installed in the Keil µVision IDE" figure (Figure 22) illustrates what the Pack Installer perspective looks like after installation.

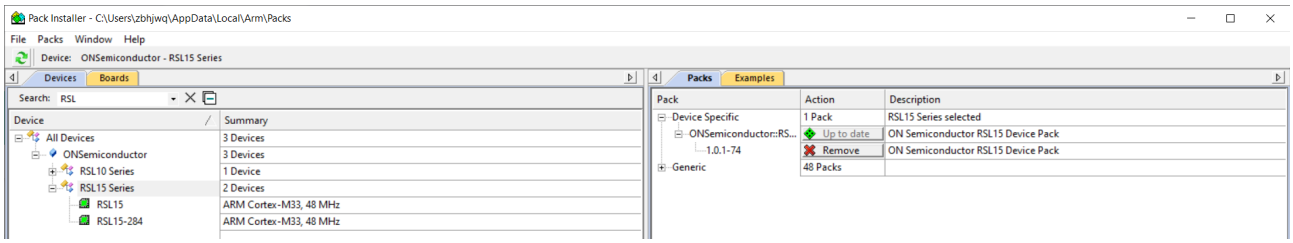

Figure 22. Pack Installer after RSL15 CMSIS-Pack is Installed in the Keil µVision IDE

5.4 Importing the Sample Code

To import the sample code:

1. In the Pack installer, click on the **Examples** tab to list all the example projects included in the RSL15 CMSIS-Pack.
2. Choose the example project called *blinky*, and click the **Copy** button to import it into your workspace (see the "Pack Manager Perspective: Examples Tab" figure (Figure 23)). Choose a destination folder for a copy of the sample code. We recommend leaving the **Launch µVision** checkbox checked.

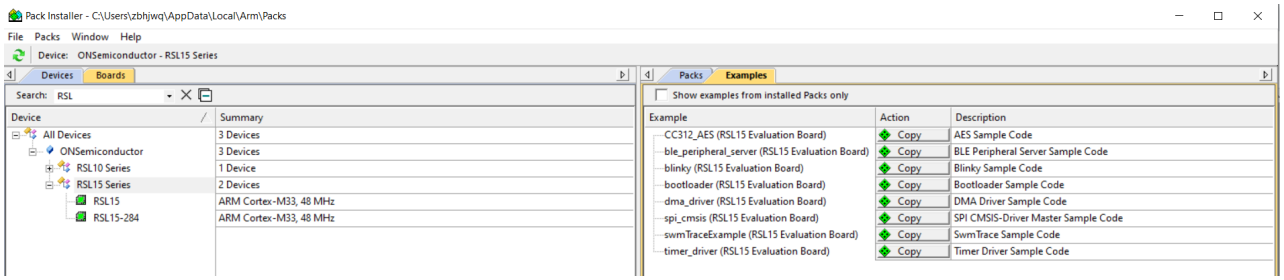

Figure 23. Pack Manager Perspective: Examples Tab

For the *blinky* sample, the Startup, HAL and GPIO components are preconfigured with the source variant, so the source code of these libraries is included directly (see the "RTE Configuration for the Blinky Example Project in the Keil µVision IDE" figure (Figure 24)).

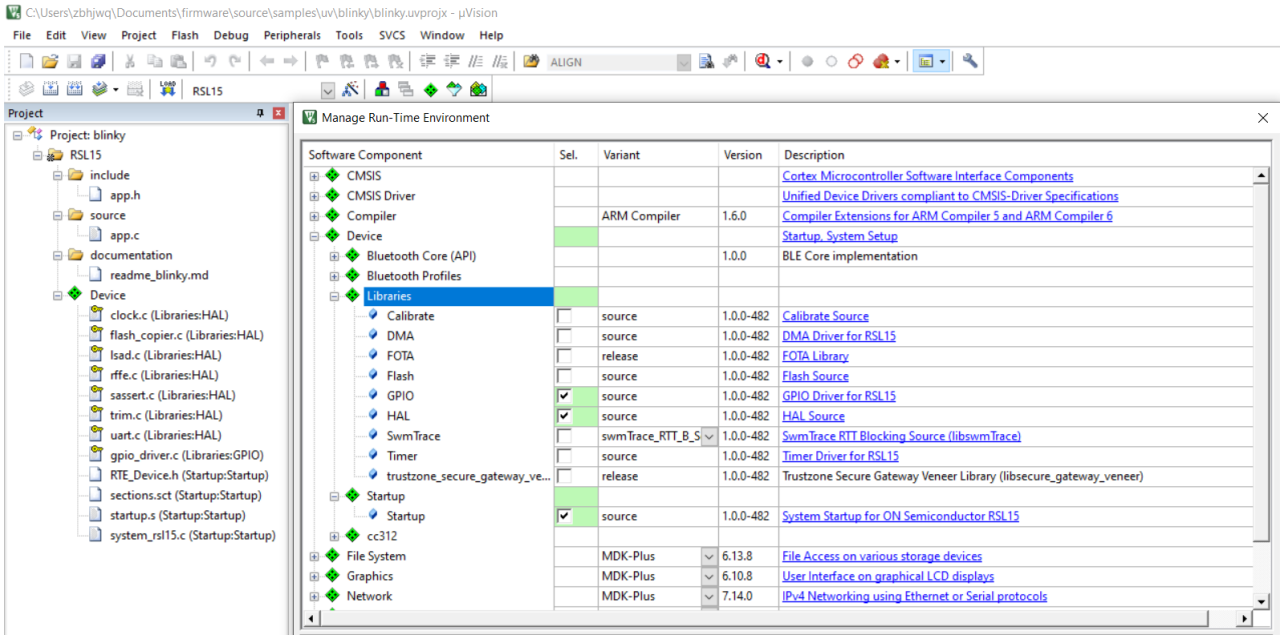

Figure 24. RTE Configuration for the Blinky Example Project in the Keil µVision IDE

5.5 Building the Sample Code

Build the sample code as follows:

1. Right click on **RSL15** and **choose Rebuild all target files**. Alternatively, you can use the icon shown in the "Starting to Build a Project in the Keil µVision IDE" figure (Figure 25).

NOTE: Compiling with Link-Time Optimization (LTO) enabled might result in some unexpected results. For more information on LTO, see this LTO article in the Arm Community website.

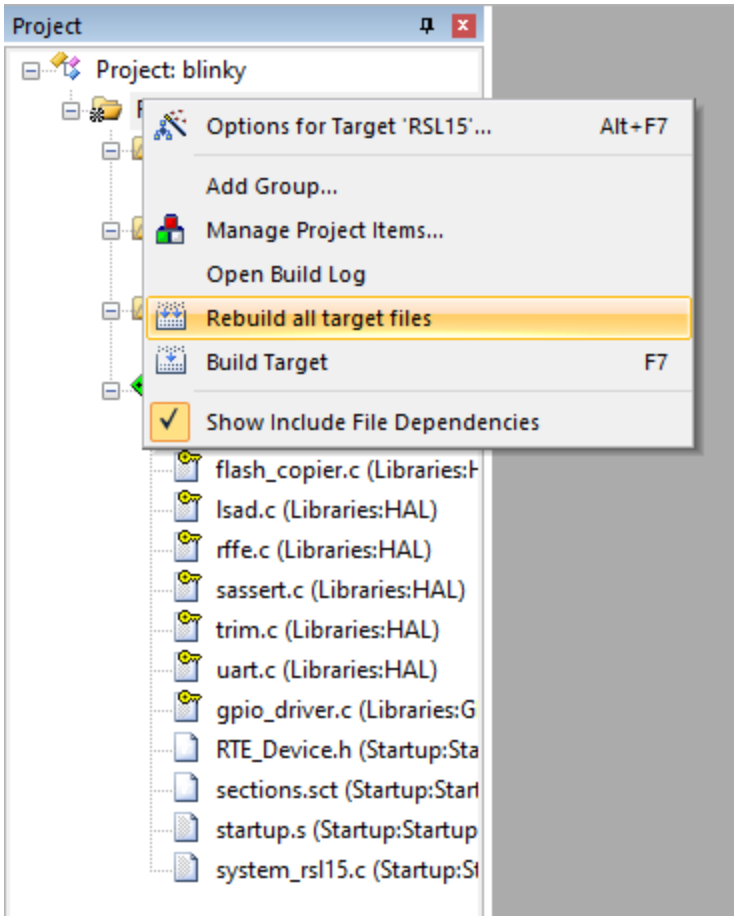

Figure 25. Starting to Build a Project in the Keil µVision IDE

2. When the build is running, the output of the build is shown in the Build Output view in the IDE, as illustrated in the "Example of Build Output" figure (Figure 26).

```
Build Output
Rebuild started: Project: blinky
*** Using Compiler 'V6.15', folder: 'C:\Keil_v5\ARM\ARMCLANG\Bin'
Rebuild target 'RSL15'
compiling lsad.c...
compiling sassert.c...
compiling rffe.c...
compiling clock.c...
compiling uart.c...
compiling system_rsl15.c...
compiling app.c...
compiling gpio_driver.c...
compiling trim.c...
compiling flash_copier.c...
assembling startup.s...
linking...
Program Size: Code=5980 RO-data=244 RW-data=68 ZI-data=4108
FromELF: creating hex file...
".\Objects\blinky.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:03
```

Figure 26. Example of Build Output

3. The key resulting output in Project Explorer in the IDE includes:
   - *blinky.hex*: HEX file for loading into Flash memory
   - *blinky.axf*: Arm executable file, run from RAM, used for debugging
   - *blinky.map*: map file of the sections and memory usage

5.6 Debugging the Sample Code

5.6.1 Preparing J-Link for Debugging

Before debugging with J-Link, go to *C:\Keil_v5\ARM\Segger* and make sure that the folder contains a *JL2CM3.dll* file. As well, make sure that you have installed a compatible version of J-Link.

5.6.2 Debugging Applications

The IDE's debug configurations are already set in the CMSIS-Pack. To debug an application:

1. Make sure the EVB is connected to the PC via a USB cable.
2. Select **Debug** > **Start/Stop Debug Session** or click the icon shown in the "Start/Stop Debug Session Icon" figure (Figure 27).

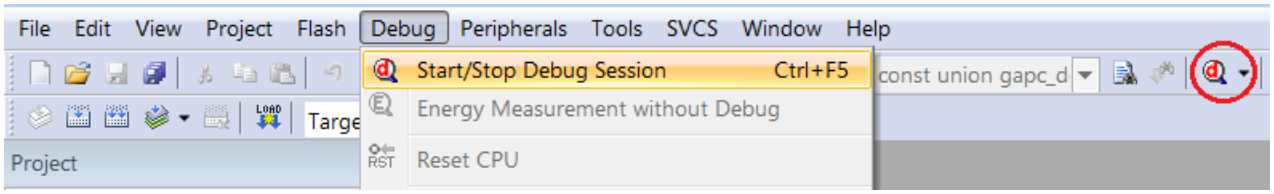

Figure 27. Start/Stop Debug Session Icon

3. The application runs up to the first breakpoint in main, as shown in the "Debug Session in the Keil µVision IDE" figure (Figure 28). You can press F11 multiple times to step through the code and observe that the LED changes its state when the application executes the line `gpio->ToggleValue(TIMER0_STATES_GPIO)`.

.

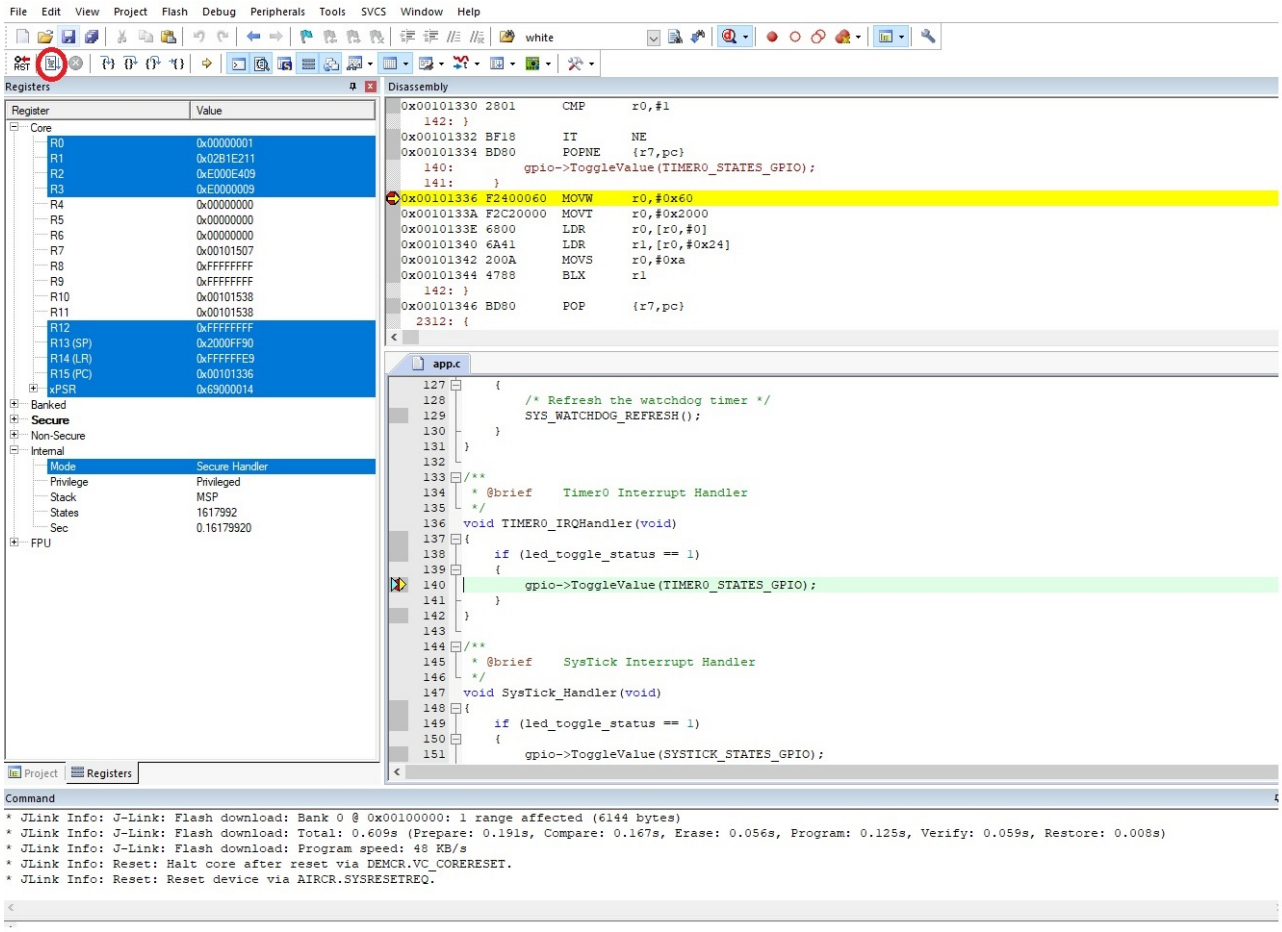

Figure 28. Debug Session in the Keil µVision IDE

NOTE: Debug configurations are preconfigured for the sample applications in the CMSIS-Pack. Flash downloading through the Download icon (as shown in the "Download Button Not Supported for J-Link" figure (Figure 29)) or F8 is not supported for J-Link.

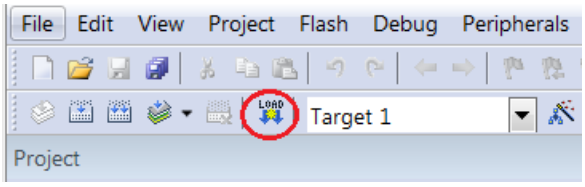

Figure 29. Download Button Not Supported for J-Link

CHAPTER 6

Getting Started with IAR

This topic shows how you can set up and start using RSL15 with the IAR Embedded Workbench® IDE from IAR Systems®.

6.1 Prerequisite Software: Downloading and Installation

1. Download and install the IAR Embedded Workbench from the IAR Systems Website, using the vendor's instructions.
2. After the IAR IDE is installed, you need to update the SEGGER® J-Link® software:
   a. Go the Start menu and type `J-Link DLL Updater V`**`<J-Link_version>`**, where **`<J-Link_version>`** is 7.86c or later, and press Enter.
   b. Select the **IAR Embedded Workbench for ARM** checkbox and click **OK**.
3. Download the RSL15 Software Package from www.onsemi.com/RSL15 and extract the RSL15 CMSIS-Pack (*onsemi.RSL15.***<version>***.pack*) to any temporary folder.

6.2 RSL15 CMSIS-Pack Installation Procedure

To install the RSL15 CMSIS-Pack:

1. Open the IAR Embedded Workbench and expand **File** > **New Workspace** to open a new workspace, then go to **File** > **Save Workspace As** and choose the location for your workspace.
2. Navigate to **Project** > **CMSIS Pack Manager**, or click on the icon shown in the "Pack Installer Icon" figure (Figure 30).

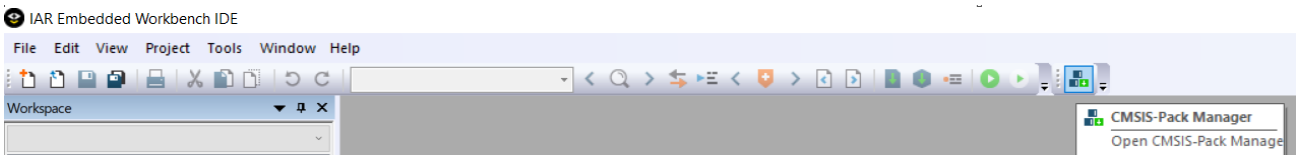

Figure 30. Pack Installer Icon

3. Click on **CMSIS Manager** > **Import Existing Packs**, select your pack file *ONSemiconductor.RSL15.***`<version>`***.pack*, and click **Open** (see the "Installing the RSL15 CMSIS_Pack for the IAR Embedded Workbench IDE" figure (Figure 31)). <version> is the RSL15 version.

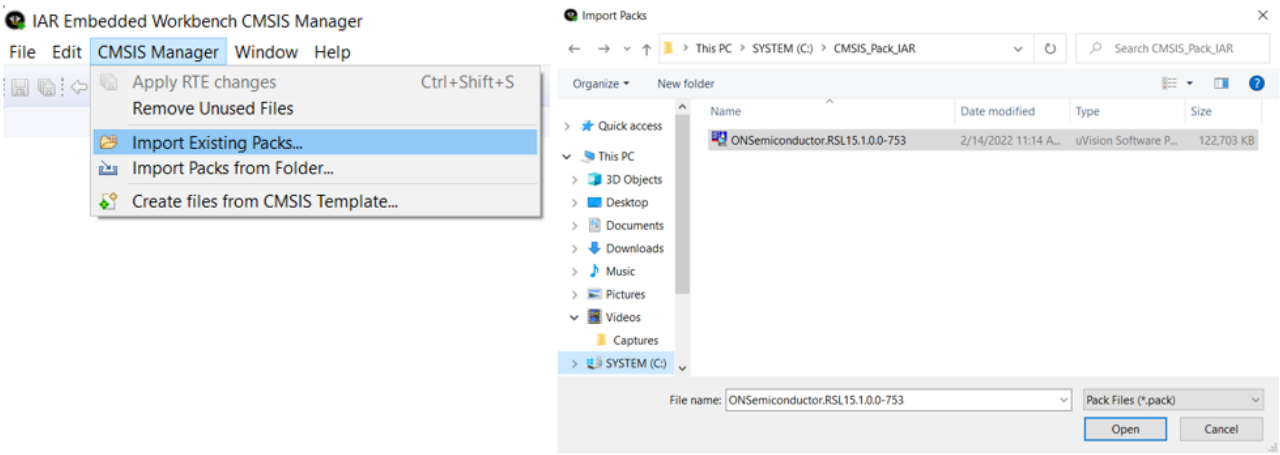

Figure 31. Installing the RSL15 CMSIS_Pack for the IAR Embedded Workbench IDE

4. The IDE prompts you to read and accept the license agreement, then installs the RSL15 CMSIS-Pack in the CMSIS-Pack root folder.
5. After installation, click on the refresh icon with yellow arrows, which shows the text **Reload Packs in the CMSIS Pack root folder** when you hover over it with your cursor, in the Packs tab (as shown in the "Refresh Pack after Installation" figure (Figure 32)), to update your pack proprieties.

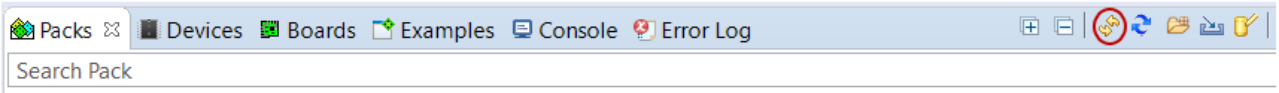

Figure 32. Refresh Pack after Installation

6. In the **Devices** tab, expand **All Devices** > **ONSemiconductor** > **RSL15 Series**, and select **RSL15** from the list. The RSL15 CMSIS-Pack now appears in the list of installed packs in the **Packs** tab. Expanding **ONSemiconductor.RSL15** makes the **Pack Properties** tab display the details of the RSL15 CMSIS-Pack. The "IAR Embedded Workbench CMSIS Manager after RSL15 CMSIS-Pack is Installed" figure (Figure 33) illustrates what the **Pack Manager** perspective looks like after installation.

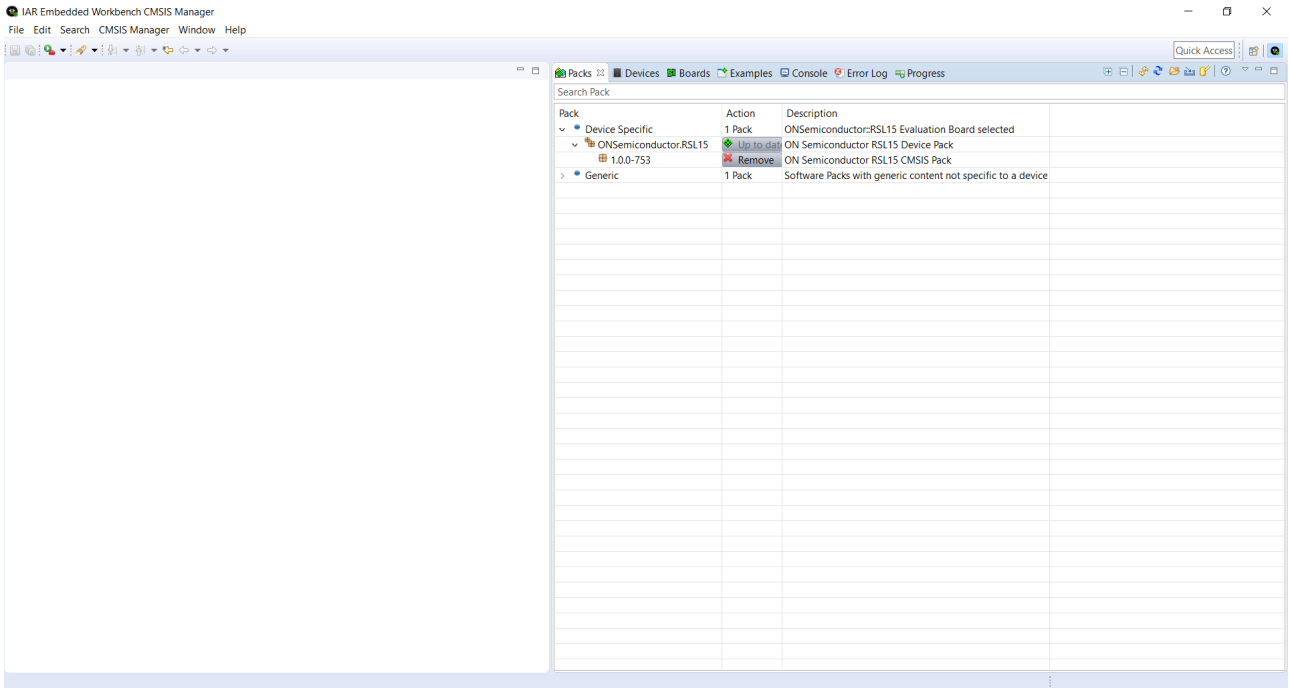

Figure 33. IAR Embedded Workbench CMSIS Manager after RSL15 CMSIS-Pack is Installed

6.3 Building Your First Sample Application with the IAR Embedded Workbench

This section guides you through importing and building your first sample application, named *blinky*. This application makes the LED (DIO6) blink on the RSL15 Evaluation and Development Board. The procedure described in this section assumes that you have installed the SDK.

For more information about the sample applications, see the *readme* files accompanying the code.

6.3.1 Import the Sample Code

Import the sample code to your workspace as follows:

1. In the IDE's **CMSIS Manager**, click on the **Examples** tab to list all the example projects included in the RSL15 CMSIS-Pack.
2. Choose the example project called **blinky**, and click the **Copy** button to import it into your workspace (see the "IAR Embedded Workbench CMSIS Manager: Examples Tab" figure (Figure 34)). Choose a destination folder for a copy of the sample code.

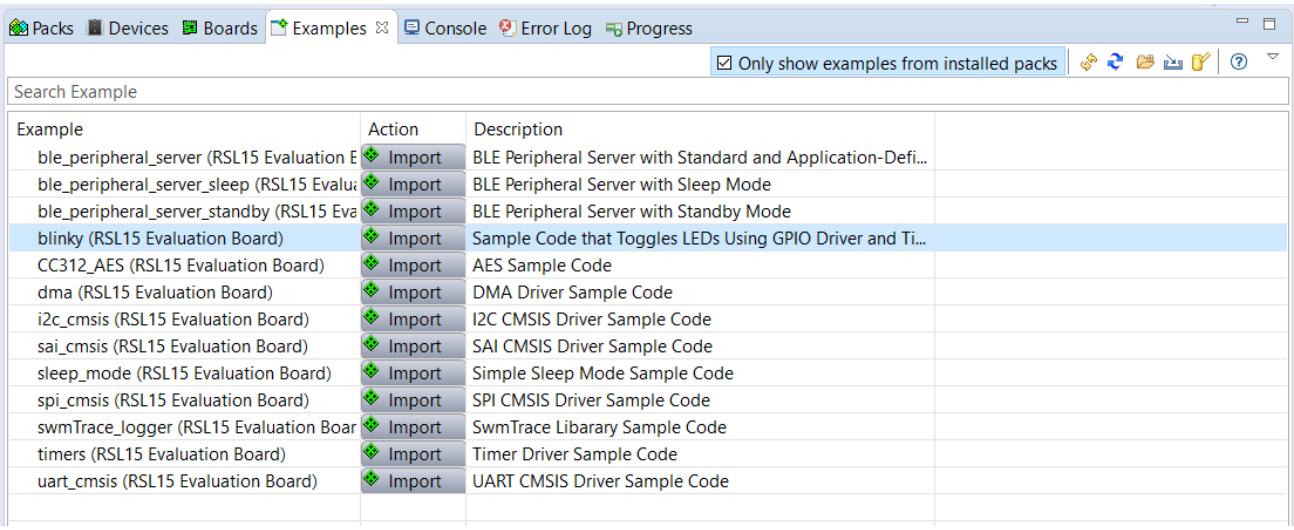

Figure 34. IAR Embedded Workbench CMSIS Manager: Examples Tab

For the blinky sample, the Startup, HAL and GPIO components are preconfigured with the source variant, so the source code of these libraries is included directly (see the "RTE Configuration for the Blinky Example Project in the IAR Embedded Workbench CMSIS Manager Window" figure (Figure 36) and the "RTE Configuration for the Blinky Example Project in the IAR Embedded Workbench Window" figure (Figure 35)).

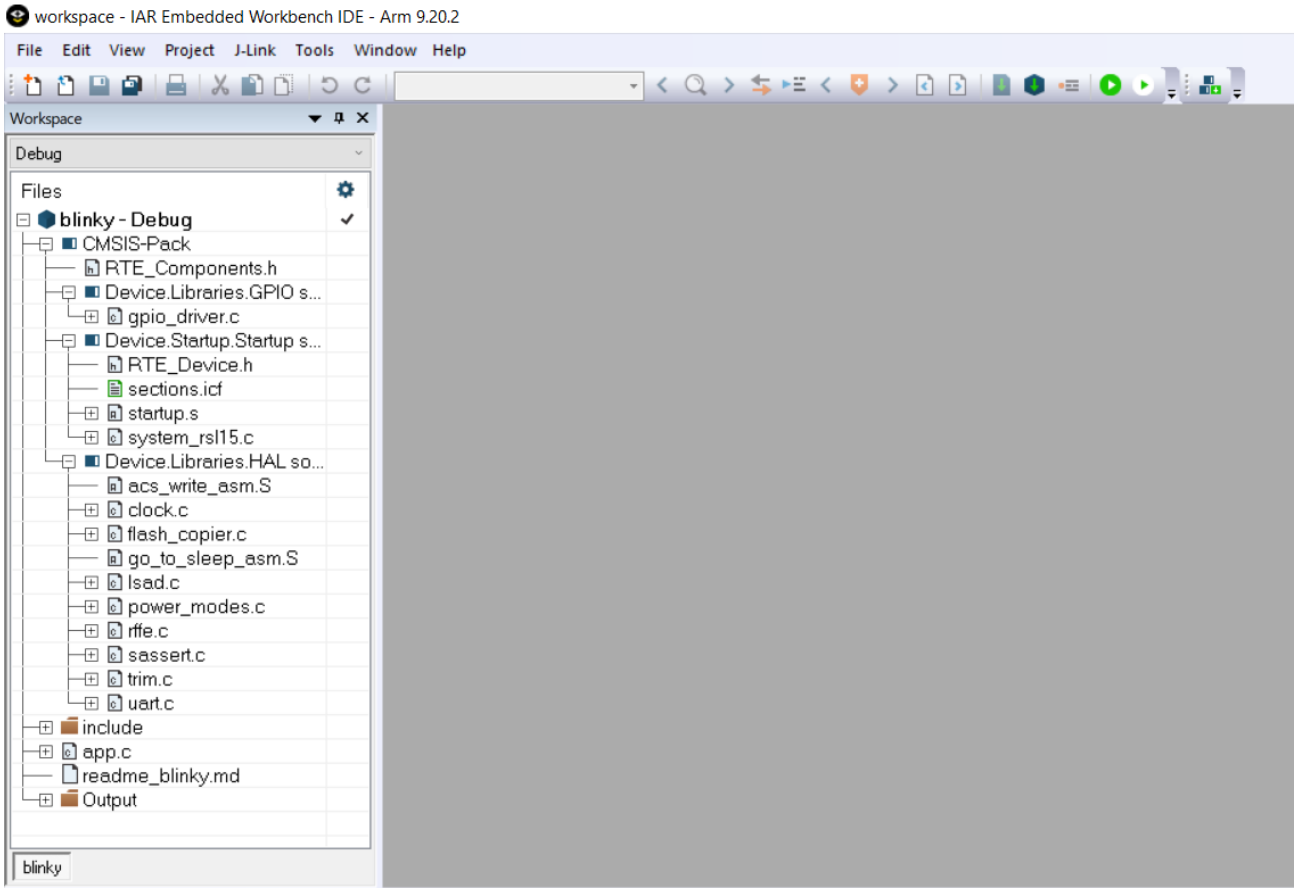

Figure 35. RTE Configuration for the Blinky Example Project in the IAR Embedded Workbench Window

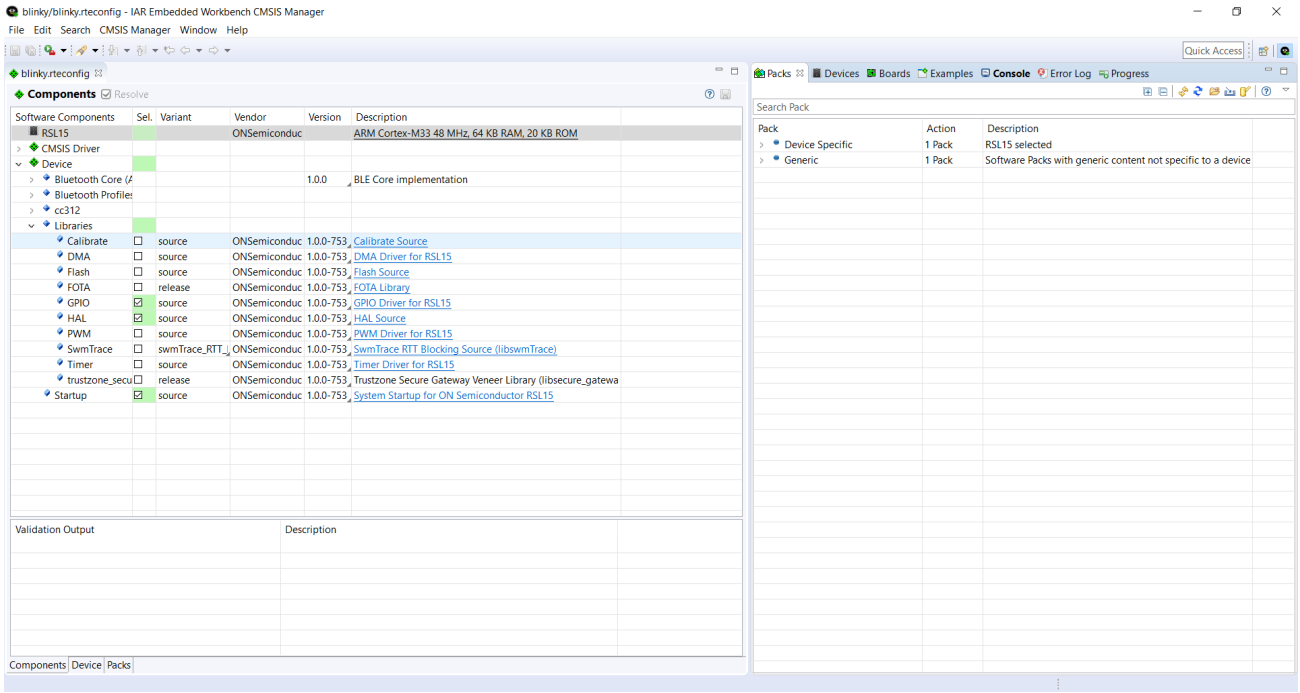

Figure 36. RTE Configuration for the Blinky Example Project in the IAR Embedded Workbench CMSIS Manager Window

6.3.2 Building the Sample Code

To build the sample code:

1. Right click on the folder for **blinky** and choose **Rebuild All**. Alternatively, you can use the icon shown in the "Starting a Project Build in the IAR Embedded Workbench" figure (Figure 37).

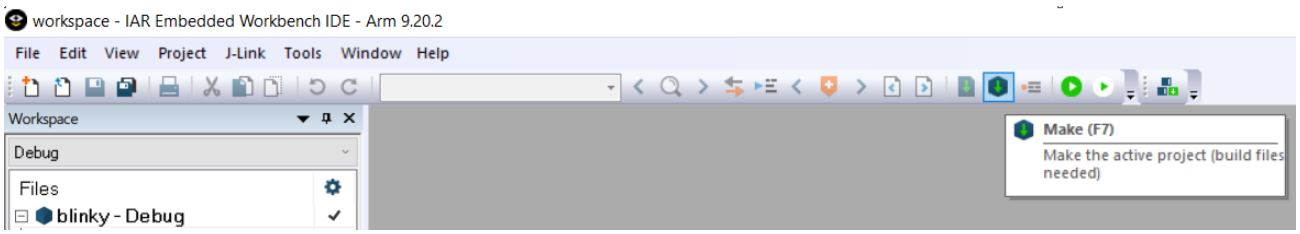

Figure 37. Starting a Project Build in the IAR Embedded Workbench

2. When the build is running, the output of the build is displayed in the Build Output view in the IDE, as illustrated in the "Example of Build Output" figure (Figure 38).

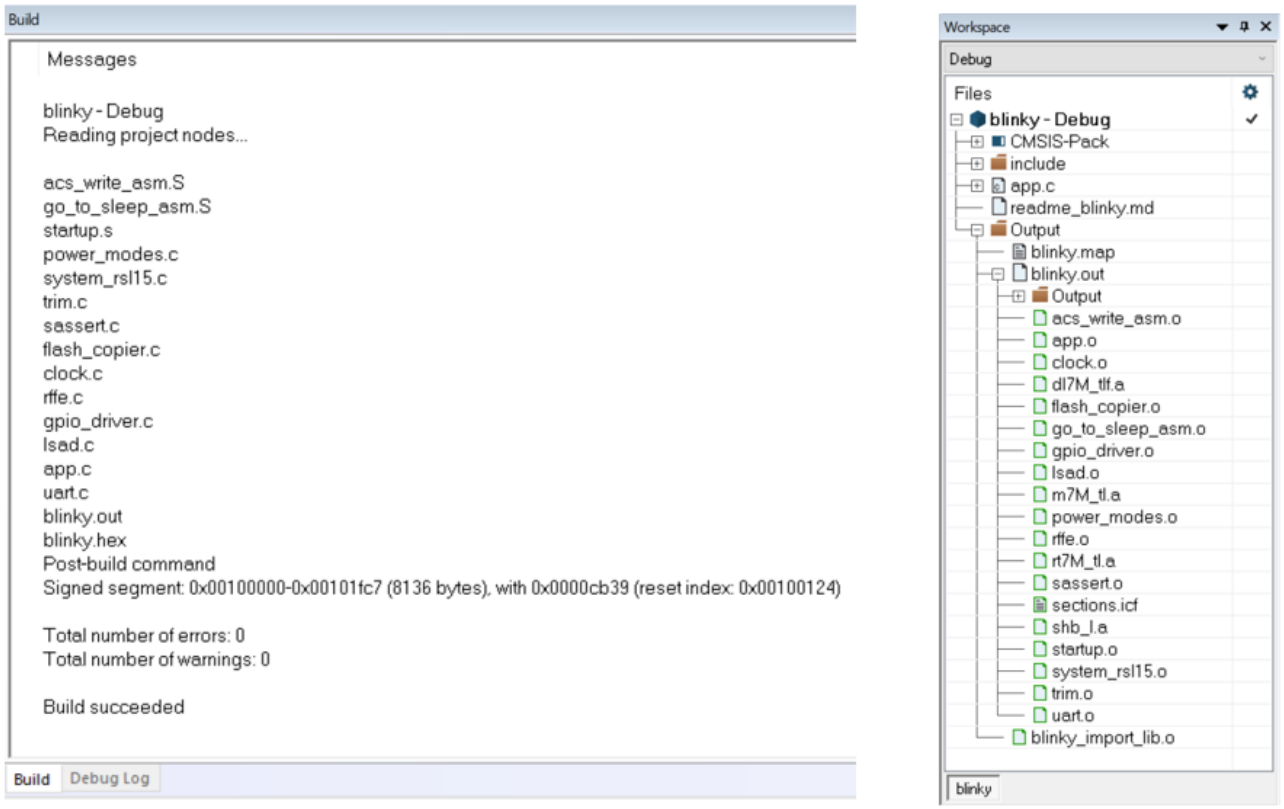

Figure 38. Example of Build Output

3. The key resulting output shown in Project Explorer in the IDE includes:
   - *blinky.hex*: HEX file for loading into flash memory
   - *blinky.out*: Arm executable file, used for debugging
   - *blinky.map*: map file of the sections and memory usage

6.3.3 Debugging the Sample Code

6.3.3.1 Debugging Applications

IDE debug configurations are already set in the CMSIS pack. To debug an application:

1. Make sure the EVB is connected to the PC via a micro USB cable.
2. Select **Project** > **Download and Debug**, or click the icon shown in the "Start/Stop Debug Session Icon" below, then accept the J-Link pop-up dialog in order to use the flash breakpoints (as shown in the "J-Link "Out of breakpoints" Pop-up Dialog" on the next page).

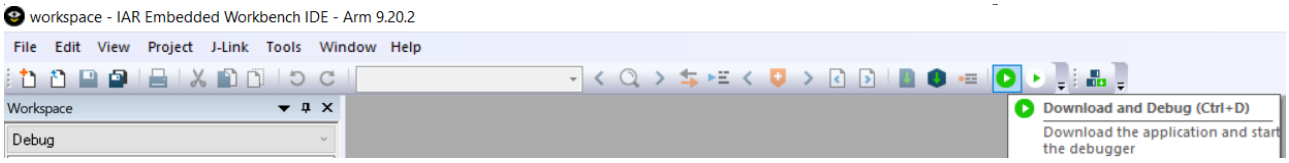

Figure 39. Start/Stop Debug Session Icon

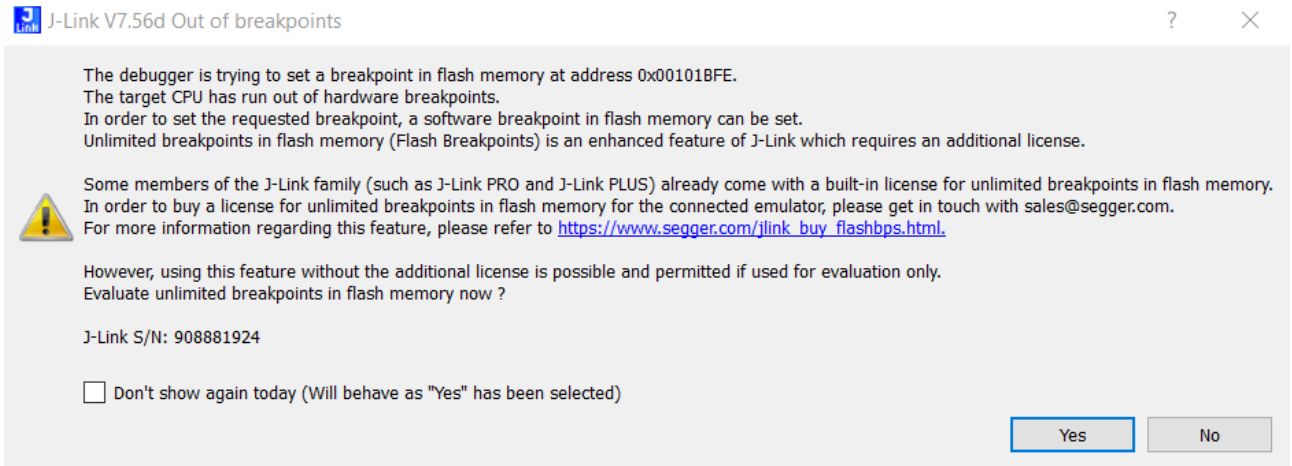

Figure 40. J-Link "Out of breakpoints" Pop-up Dialog

3. The application runs up to the first breakpoint in *main*. You can press F5 or click the Run icon (as shown in Figure 38) multiple times to step through the code and observe that the LED changes its state when the application executes the line `gpio->ToggleValue(SYSTICK_STATES_GPIO)`. To stop the debug session, press the Stop icon.

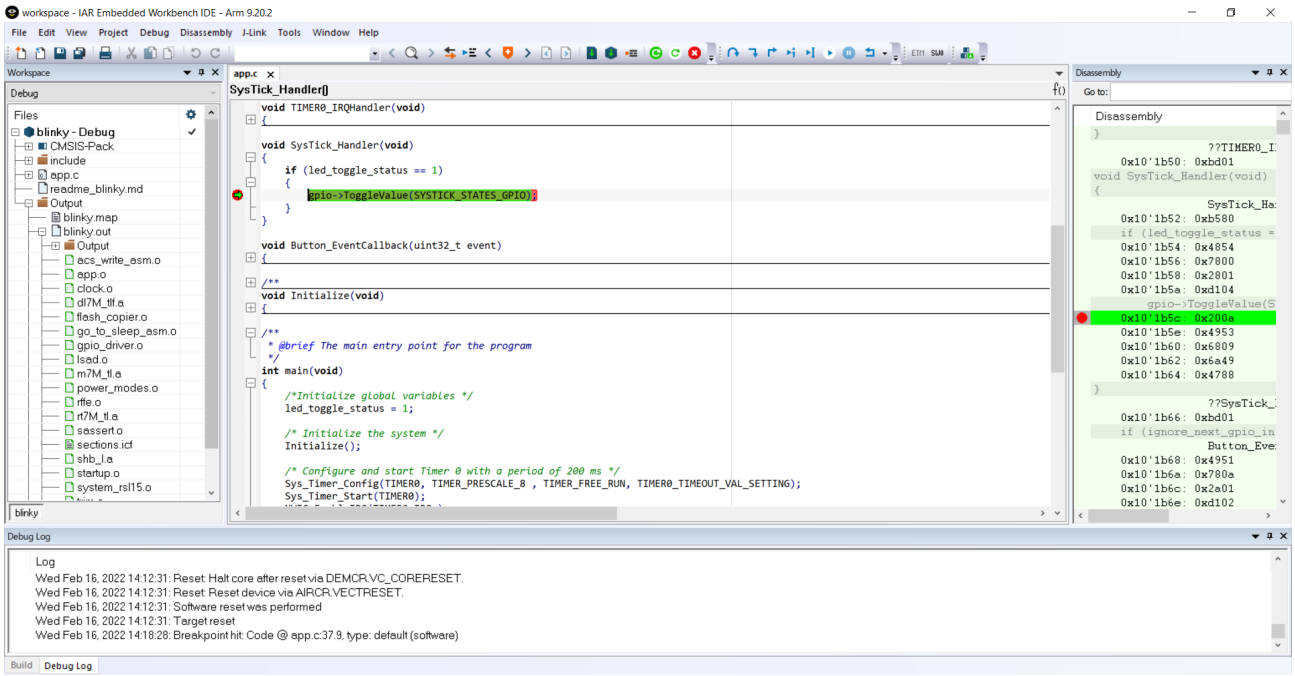

Figure 41. Debug Session in the IAR Embedded Workbench

CHAPTER 7

Resolving External CMSIS-Pack Dependencies

This topic explains CMSIS-Pack dependencies, and how to resolve them so that "component is missing" errors are avoided when importing sample code from the RSL15 CMSIS-Pack into the workspace.

7.1 External CMSIS_Pack Dependencies

Some of the RSL15 sample applications depend on software components from external vendors. For example, applications that make use of CMSIS-Drivers or FreeRTOS depend on CMSIS-Packs provided by Arm. The dependencies are displayed in the RTE Configuration. See Section 7.2 “Resolving External Dependencies” on page 40 for an example.

7.2 Resolving External Dependencies

The following instructions show how to identify and resolve external dependencies in RSL15 sample applications using the CMSIS-Pack manager.

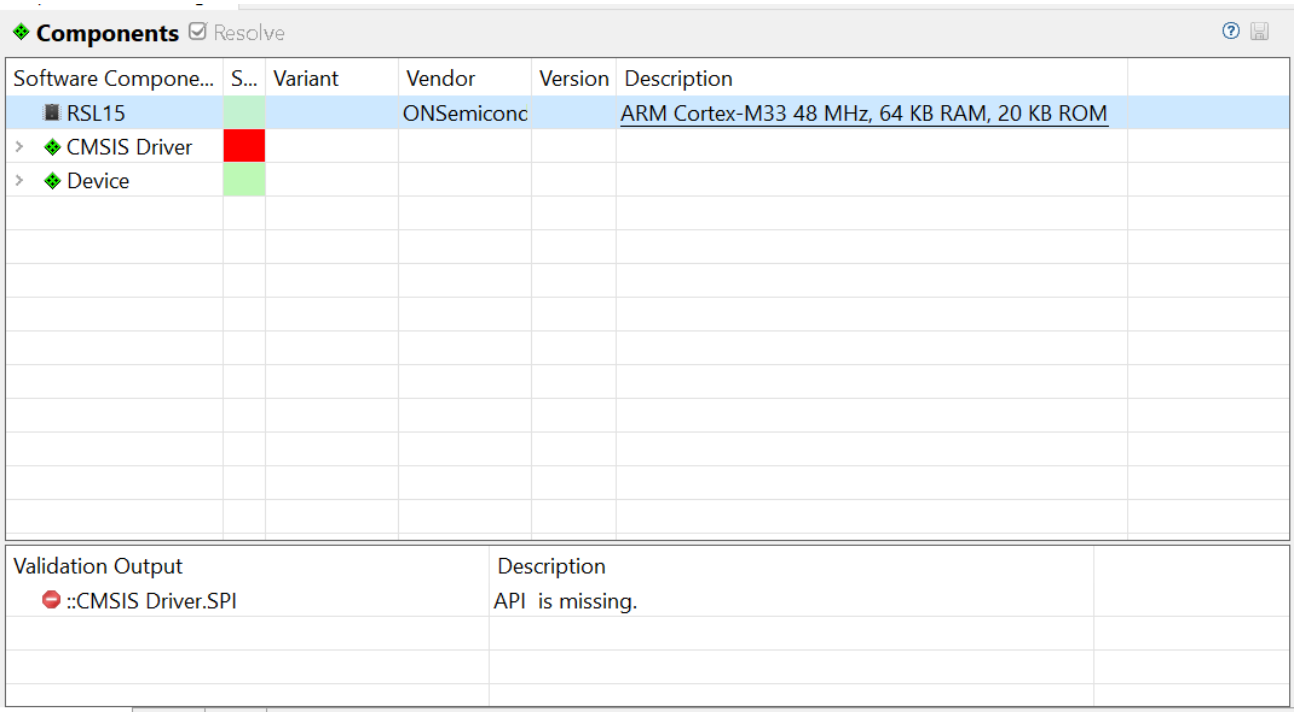

Figure 42. RTE Configuration Perspective Before Resolving Pack Dependencies

The "RTE Configuration Perspective Before Resolving Pack Dependencies" figure (Figure 42) shows the RTE Configuration view when Pack dependencies are unresolved. To resolve Pack dependencies, follow these steps:

1. In the CMSIS-Pack Manager perspective, click on the **Check for Updates on Web** button (see the "Check for Updates on Web Button" figure (Figure 43)).

Figure 43. Check for Updates on Web Button

The "Installing the Arm CMSIS-Pack" figure (Figure 44) shows an example of the Packs tab after checking for updates.

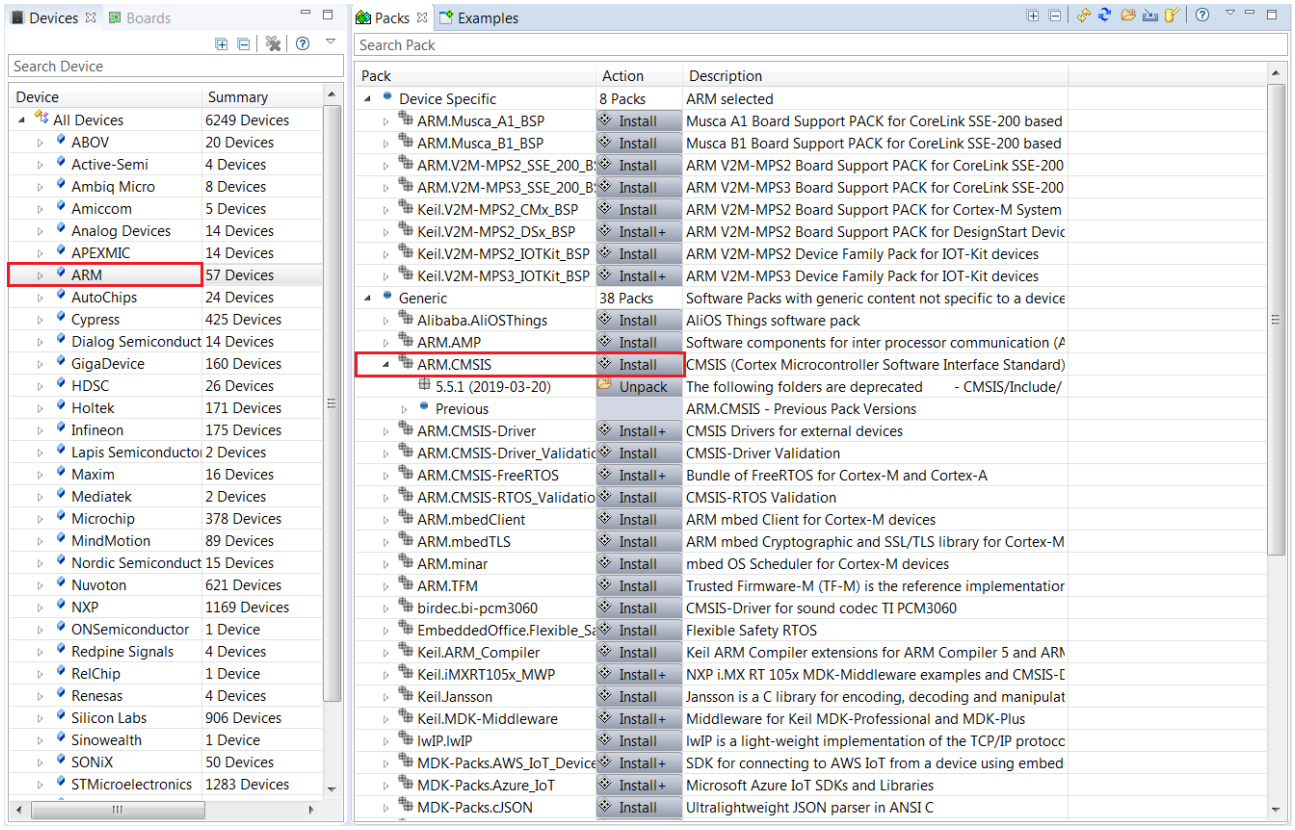

Figure 44. Installing the Arm CMSIS-Pack

2. To manually install a CMSIS-Pack, select the **Packs** tab and search for the required CMSIS-Pack (in this example, we installed the **ARM.CMSIS** pack); click the **Install** button (shown in the "Installing the Arm CMSIS-Pack" figure (Figure 44)). Alternatively, follow the next steps to automatically resolve any Pack dependencies that are missing.
3. Open the **.rteconfig* file; in the **Packs** tab, select the **Resolve Missing Packs** button (see the "Resolve Missing Packs Icon" figure (Figure 45)).

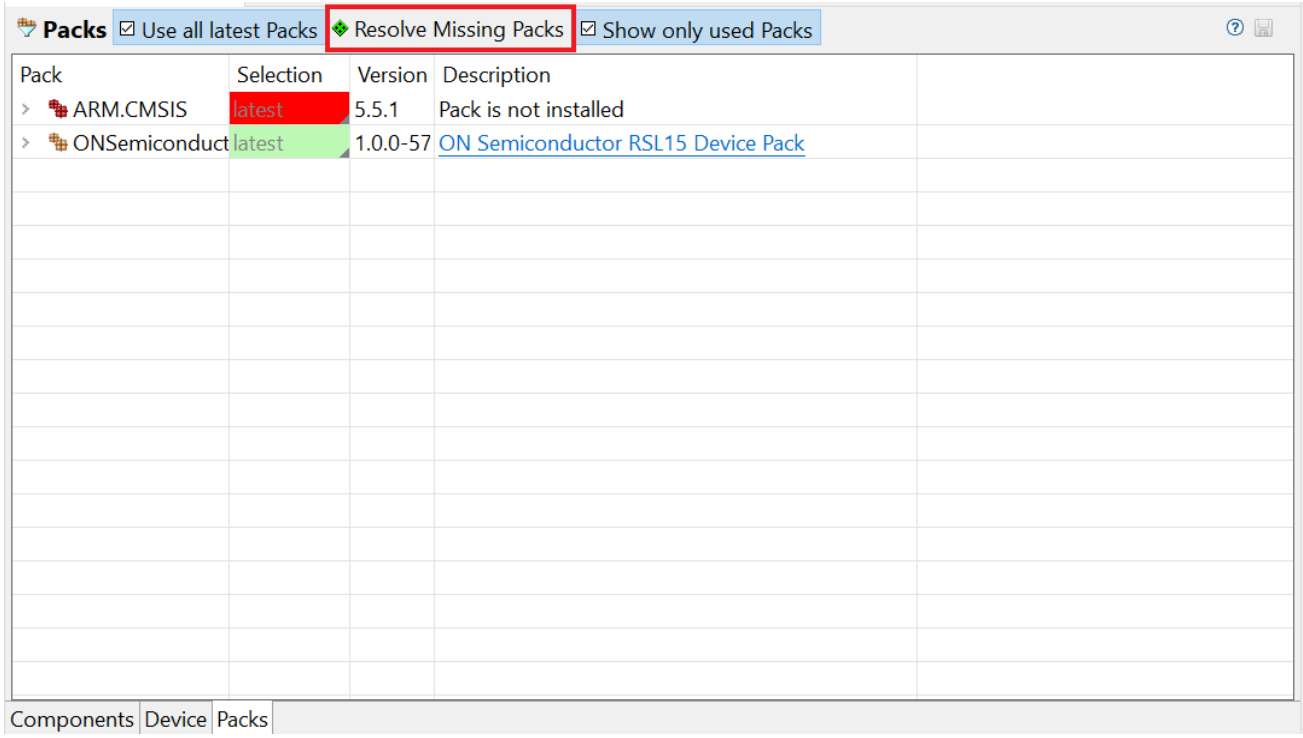

Figure 45. Resolve Missing Packs Icon

4. The IDE prompts you to read and accept the license agreement, then installs the missing Packs. The "RTE Configuration Perspective After Resolving Pack Dependencies" figure (Figure 46) illustrates the RTE configuration after resolving missing Packs.

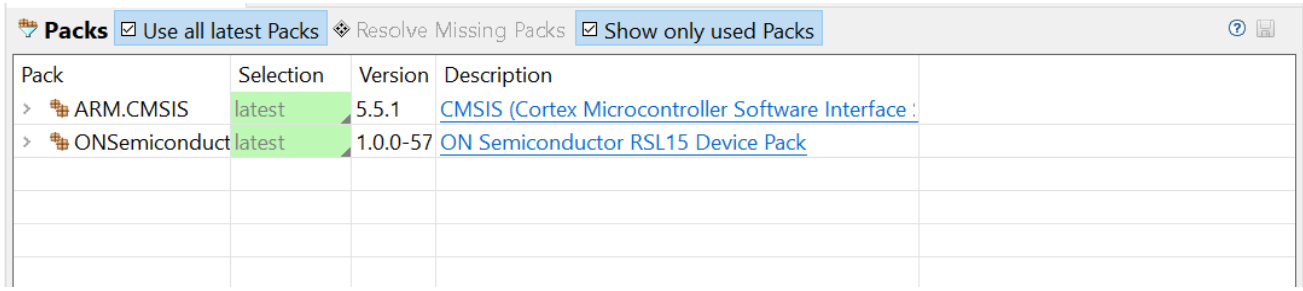

Figure 46. RTE Configuration Perspective After Resolving Pack Dependencies

CHAPTER 8

Accessing Documentation Included with the CMSIS-Pack

Documentation is included in the CMSIS-Pack, and is found in the *\documentation* folder at your RSL15 install location, *C:\Users*<**user_id**>*\AppData\Local\Arm\Packs\ONSemiconductor\RSL15*<**version**>\, where <**user_id**> is your userid on your system and <**version**> is the RSL15 version number.

The documentation can be accessed directly from an IDE.

8.1 Accessing Documentation through the onsemi IDE

To access the documentation via the onsemi IDE, follow these steps:

1. From the C/C++ perspective, open any RTE configuration file, such as *blinky.rteconfig* (see the "Opening the .rteconfig file in the onsemi IDE" figure (Figure 47)).

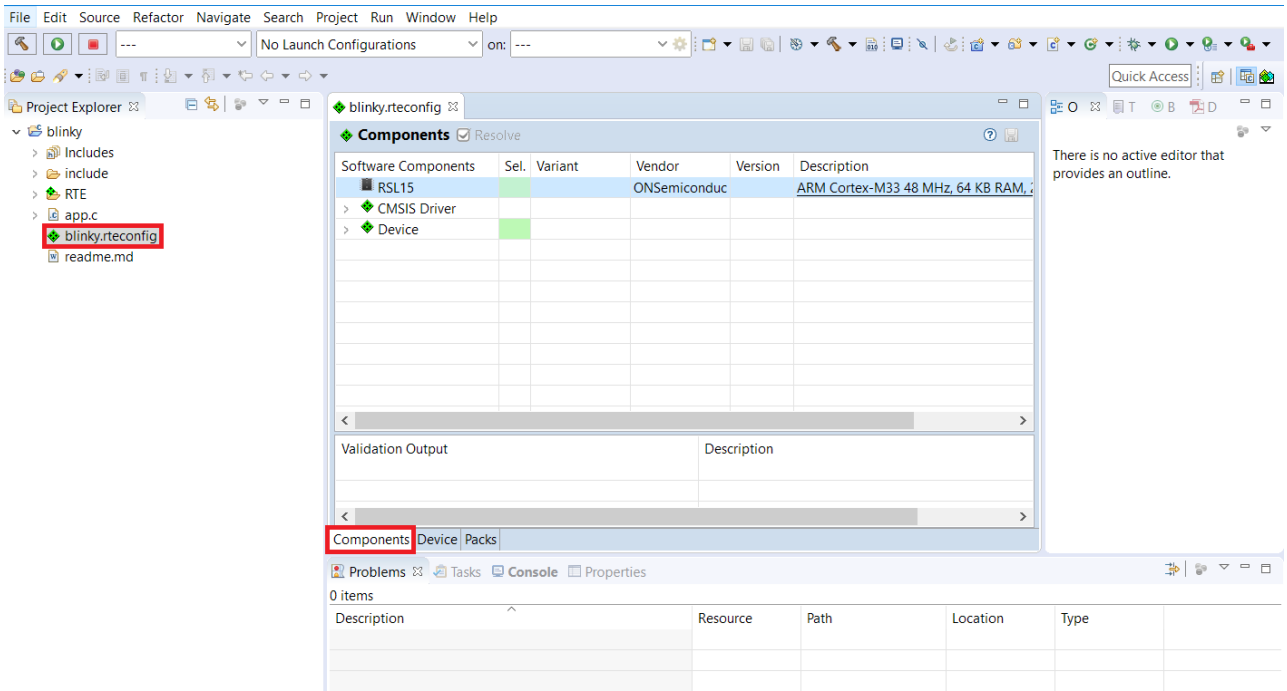

Figure 47. Opening the .rteconfig file in the onsemi IDE

2. Select the **Device** tab (see the "Accessing Documentation with the Device Tab" figure (Figure 48)).

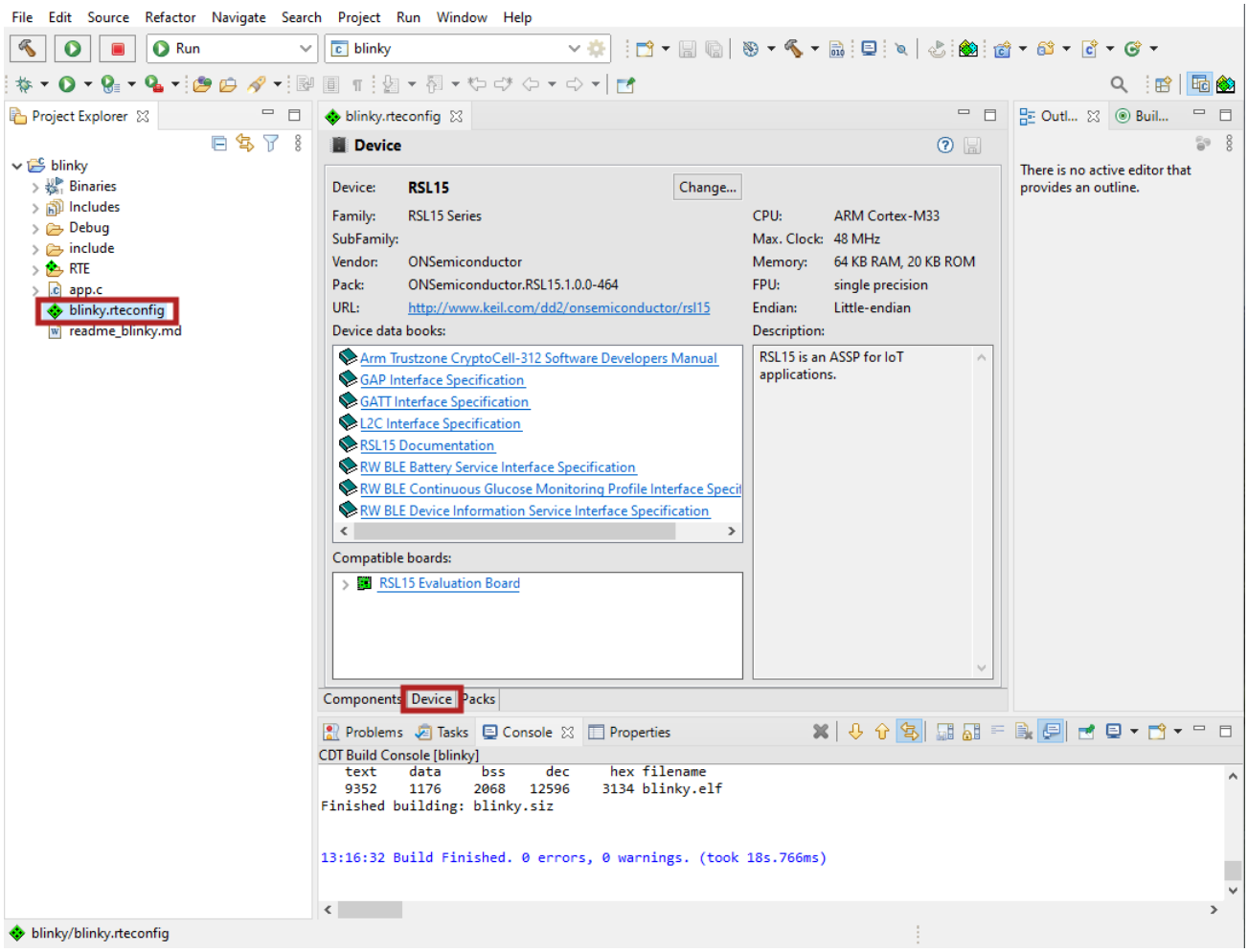

Figure 48. Accessing Documentation with the Device Tab

From this list you can open the documentation files, which are marked with green book icons.

8.2 Accessing Documentation via the Keil µVision IDE

1. To open component-specific documentation in Keil µVision, you first need to click on the **Manage Run-Time Environment** icon (see the "Keil's Mange Run-Time Environment Icon" figure (Figure 49)).

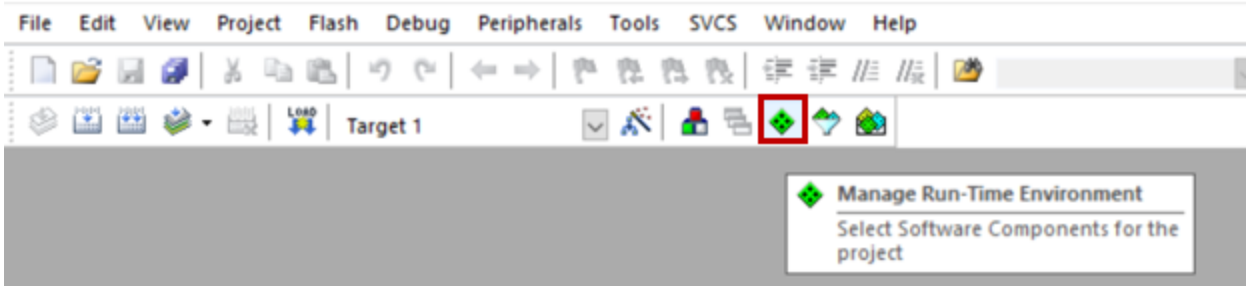

Figure 49. Keil's Mange Run-Time Environment Icon

2. Next, select the software component you are interested in (such as the Startup component chosen in the "Keil µVision Run-TIme Environment" figure (Figure 50)).

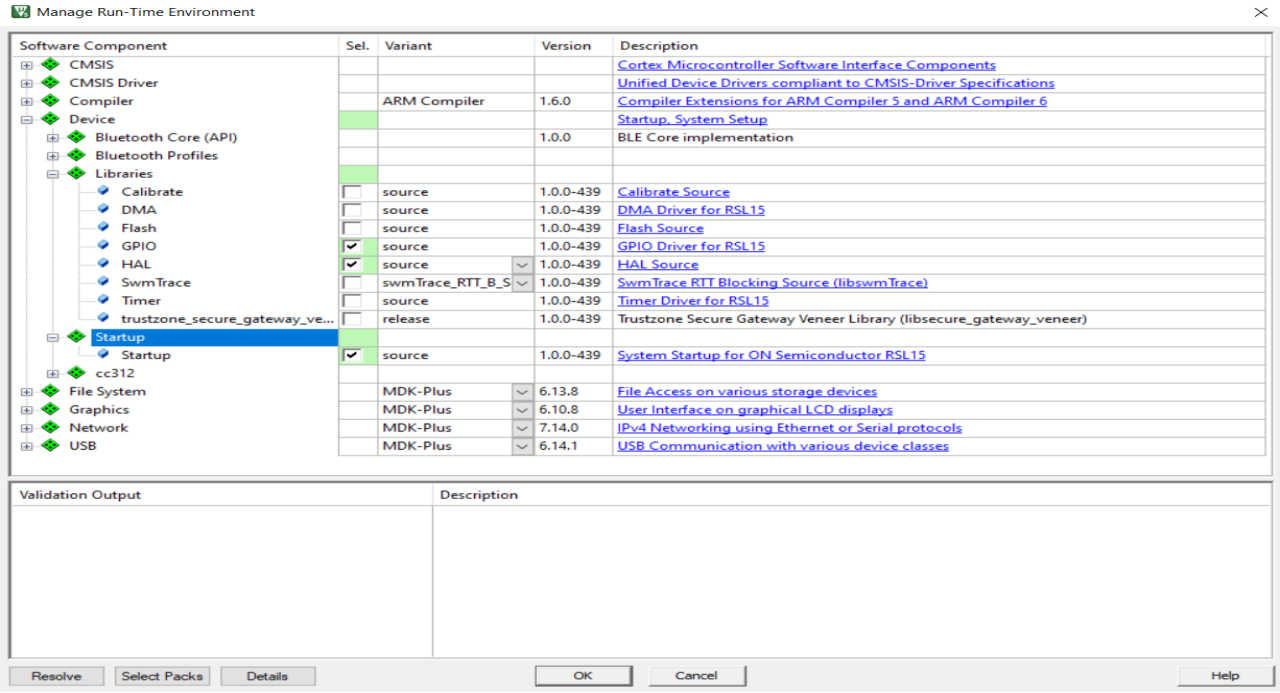

Figure 50. Keil µVision Run-TIme Environment

3. Documentation can be opened by navigating to **View** > Open **Books Window**, or to **Help** > Open **Books Window** (see the "Keil Books Window Showing Documentation" figure (Figure 51)).

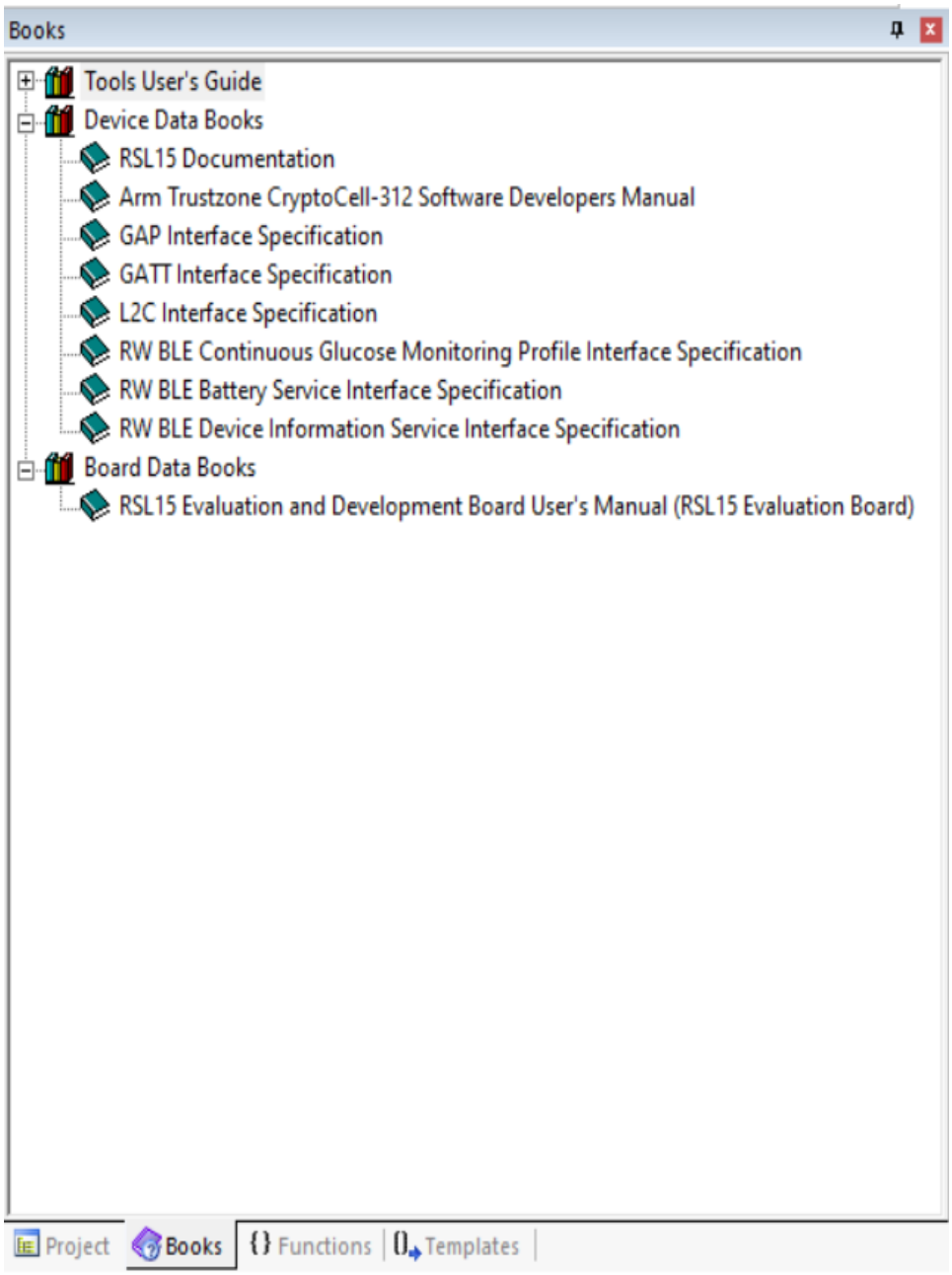

Figure 51. Keil Books Window Showing Documentation

From this list you can open the documentation files, which are marked with green book icons.

Windows is a registered trademark of Microsoft Corporation. Arm, Cortex, Keil, and uVision are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All other brand names and product names appearing in this document are trademarks of their respective holders.

onsemi and the onsemi logo are trademarks of Semiconductor Components Industries, LLC dba onsemi or its subsidiaries in the United States and/or other countries. onsemi owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of onsemi's product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent-Marking.pdf. onsemi is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.

The evaluation board/kit (research and development board/kit) (hereinafter the "board") is not a finished product and is as such not available for sale to consumers. The board is only intended for research, development, demonstration and evaluation purposes and should as such only be used in laboratory/development areas by persons with an engineering/technical training and familiar with the risks associated with handling electrical/mechanical components, systems and subsystems. This person assumes full responsibility/liability for proper and safe handling. Any other use, resale or redistribution for any other purpose is strictly prohibited.

The board is delivered "AS IS" and without warranty of any kind including, but not limited to, that the board is production-worthy, that the functions contained in the board will meet your requirements, or that the operation of the board will be uninterrupted or error free. onsemi expressly disclaims all warranties, express, implied or otherwise, including without limitation, warranties of fitness for a particular purpose and non-infringement of intellectual property rights.

onsemi reserves the right to make changes without further notice to any board.

You are responsible for determining whether the board will be suitable for your intended use or application or will achieve your intended results. Prior to using or distributing any systems that have been evaluated, designed or tested using the board, you agree to test and validate your design to confirm the functionality for your application. Any technical, applications or design information or advice, quality characterization, reliability data or other services provided by onsemi shall not constitute any representation or warranty by onsemi, and no additional obligations or liabilities shall arise from onsemi having provided such information or services.

The boards are not designed, intended, or authorized for use in life support systems, or any FDA Class 3 medical devices or medical devices with a similar or equivalent classification in a foreign jurisdiction, or any devices intended for implantation in the human body. Should you purchase or use the board for any such unintended or unauthorized application, you shall indemnify and hold onsemi and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that onsemi was negligent regarding the design or manufacture of the board.

This evaluation board/kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and may not meet the technical requirements of these or other related directives.

FCC WARNING – This evaluation board/kit is intended for use for engineering development, demonstration, or evaluation purposes only and is not considered by onsemi to be a finished end product fit for general consumer use. It may generate, use, or radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment may cause interference with radio communications, in which case the user shall be responsible, at its expense, to take whatever measures may be required to correct this interference.

onsemi does not convey any license under its patent rights nor the rights of others.

LIMITATIONS OF LIABILITY: onsemi shall not be liable for any special, consequential, incidental, indirect or punitive damages, including, but not limited to the costs of requalification, delay, loss of profits or goodwill, arising out of or in connection with the board, even if onsemi is advised of the possibility of such damages. In no event shall onsemi's aggregate liability from any obligation arising out of or in connection with the board, under any theory of liability, exceed the purchase price paid for the board, if any.

For more information and documentation, please visit www.onsemi.com.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:

Literature Distribution Center for ON Semiconductor

19521 E. 32nd Pkwy, Aurora, Colorado 80011 USA

Phone: 303-675-2175 or 800-344-3860 Toll Free USA/Canada

Fax: 303-675-2176 or 800-344-3867 Toll Free USA/Canada

Email: orderlit@onsemi.com

N. American Technical Support:

800-282-9855 Toll Free USA/Canada

Europe, Middle East and Africa Technical Support:

Phone: 421 33 790 2910

ON Semiconductor Website: www.onsemi.com

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative

M-20873-004