RSL15 Getting Started Guide

M-20873-006
March 2024

Table of Contents

Page

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 RSL 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 RSL 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 RSL Central mobile app. RSL 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 RSL 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 RSL 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 RSL Central app to connect.

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

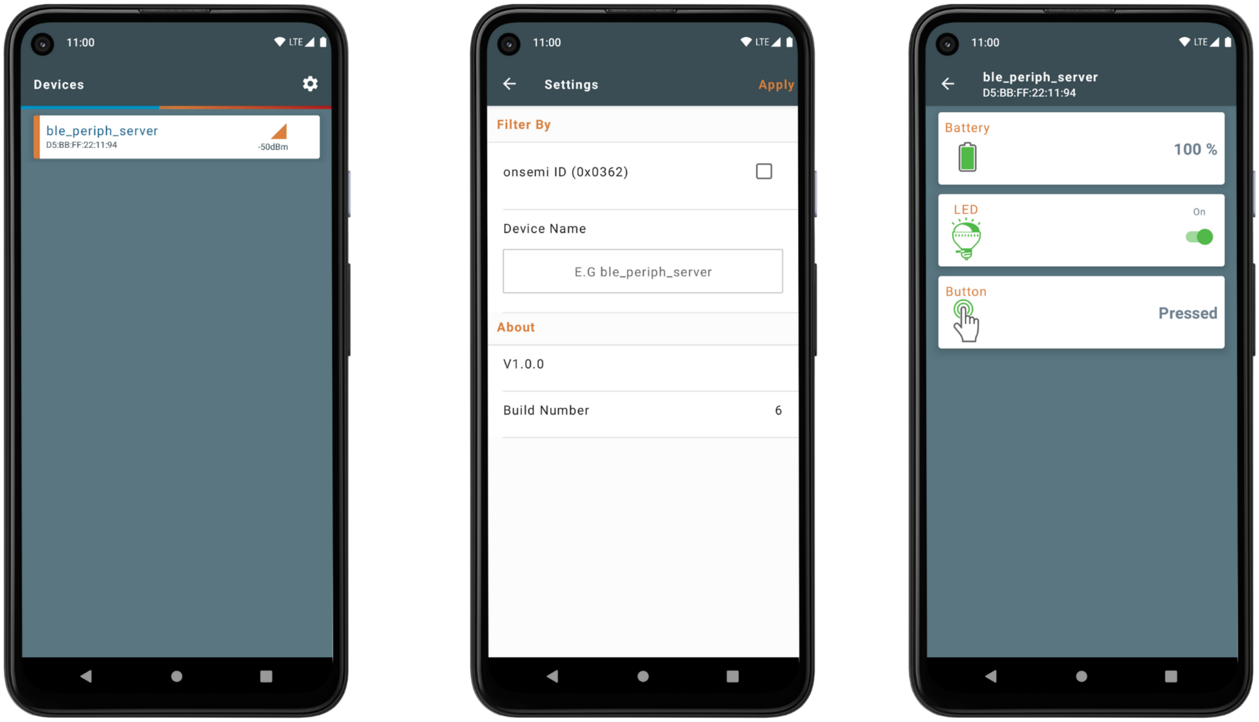


Figure 1. Example screen shots of the RSL Central Mobile App

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).

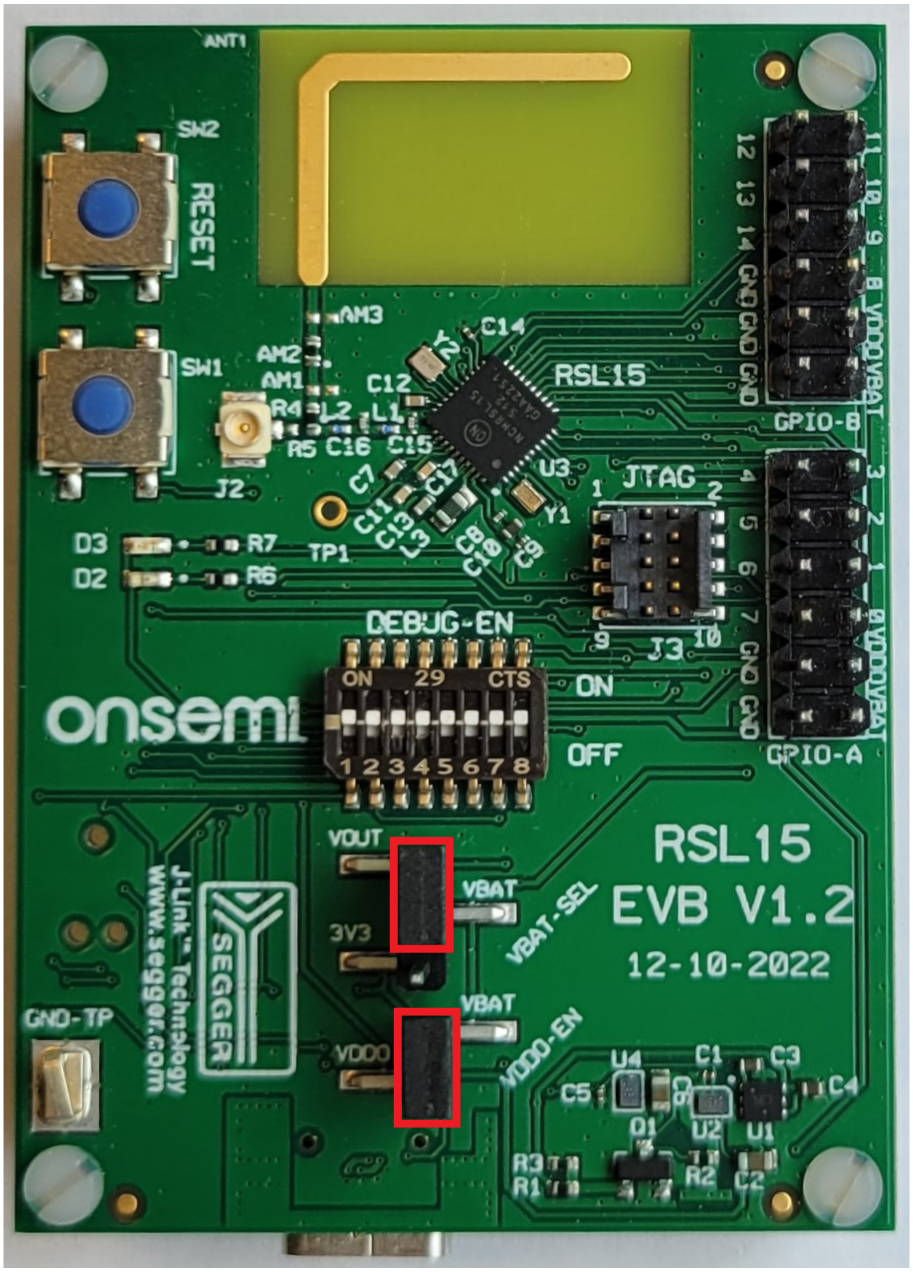


Figure 2. EVB with Default Jumper Configuration

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 RSL 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..

For instructions on installing the onsemi IDE, and details on the J-Link version required for RSL15, 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, by following these steps:
    a. Open the onsemi IDE by going to the Windows Start menu and selecting **onsemi** > **onsemi IDE**.
    b. 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>`).

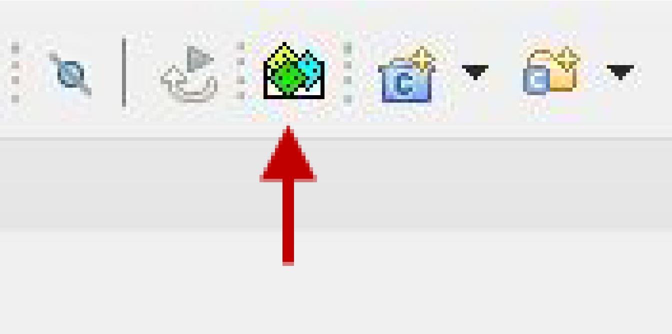
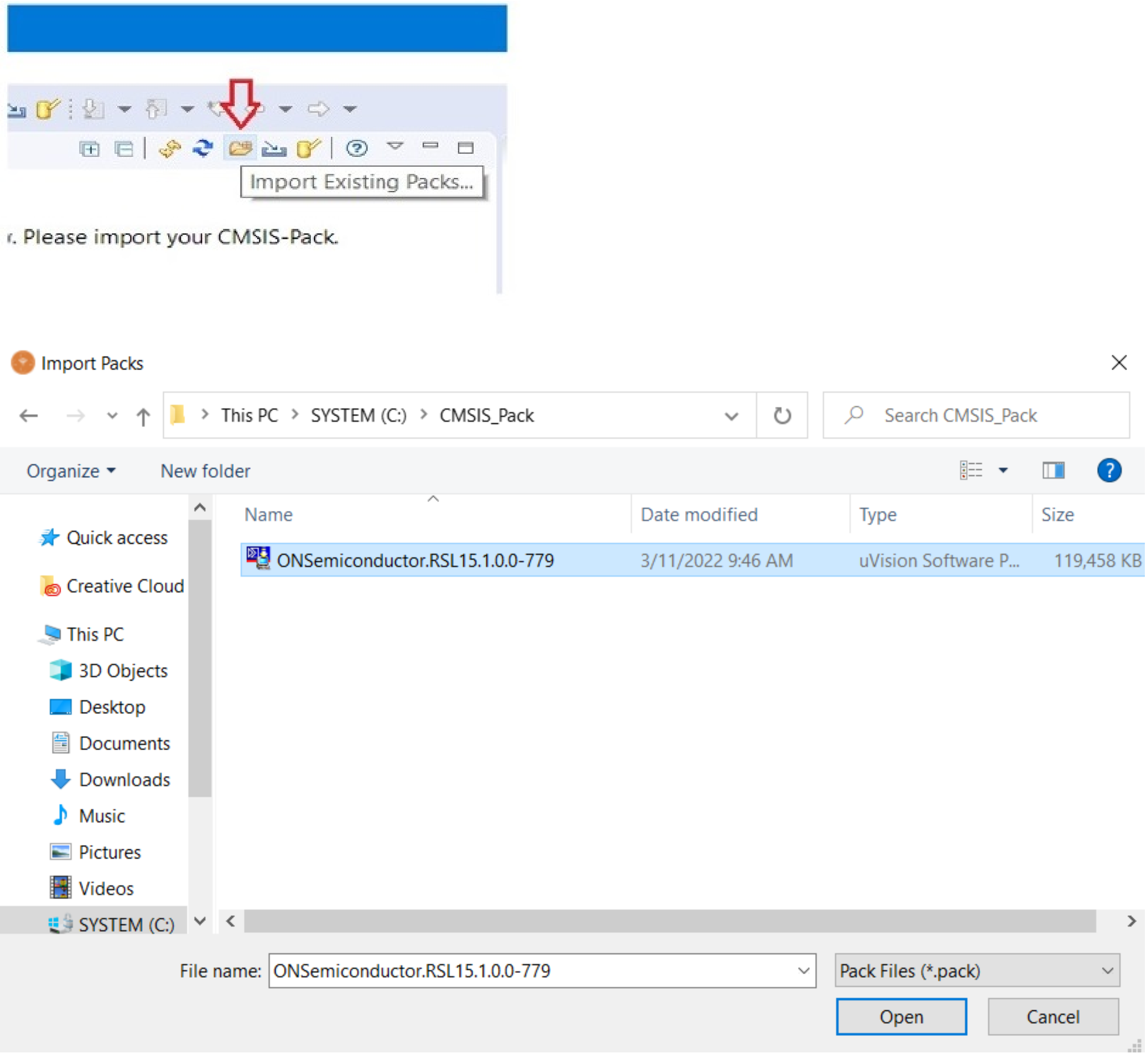


Figure 3. Opening the CMSIS-Pack Manager Perspective

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).)



r. Please import your CMSIS-Pack.

Name	Date modified	Type	Size
ONSemiconductor.RSL15.1.0.0-779	3/11/2022 9:46 AM	uVision Software P...	119,458 KB

File name: ONSemiconductor.RSL15.1.0.0-779

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).

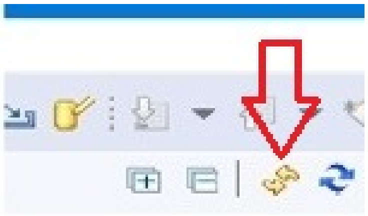


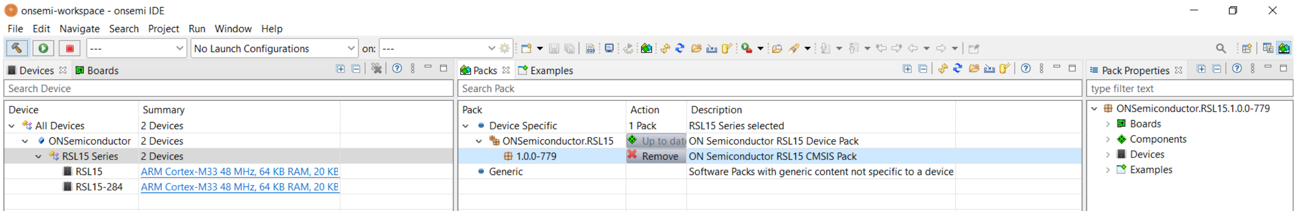
Figure 5. Reload Packs in the CMSIS-Pack root folder

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**.



Device	Summary
All Devices	2 Devices
ONSemiconductor	2 Devices
RSL15 Series	2 Devices
RSL15	ARM Cortex-M33 48 MHz, 64 KB RAM, 20 KE
RSL15-284	ARM Cortex-M33 48 MHz, 64 KB RAM, 20 KE

Pack	Action	Description
Device Specific	1 Pack	RSL15 Series selected
ONSemiconductor.RSL15	Up to dat	ON Semiconductor RSL15 Device Pack
1.0.0-779	Remove	ON Semiconductor RSL15 CMSIS Pack
Generic		Software Packs with generic content not specific to a device

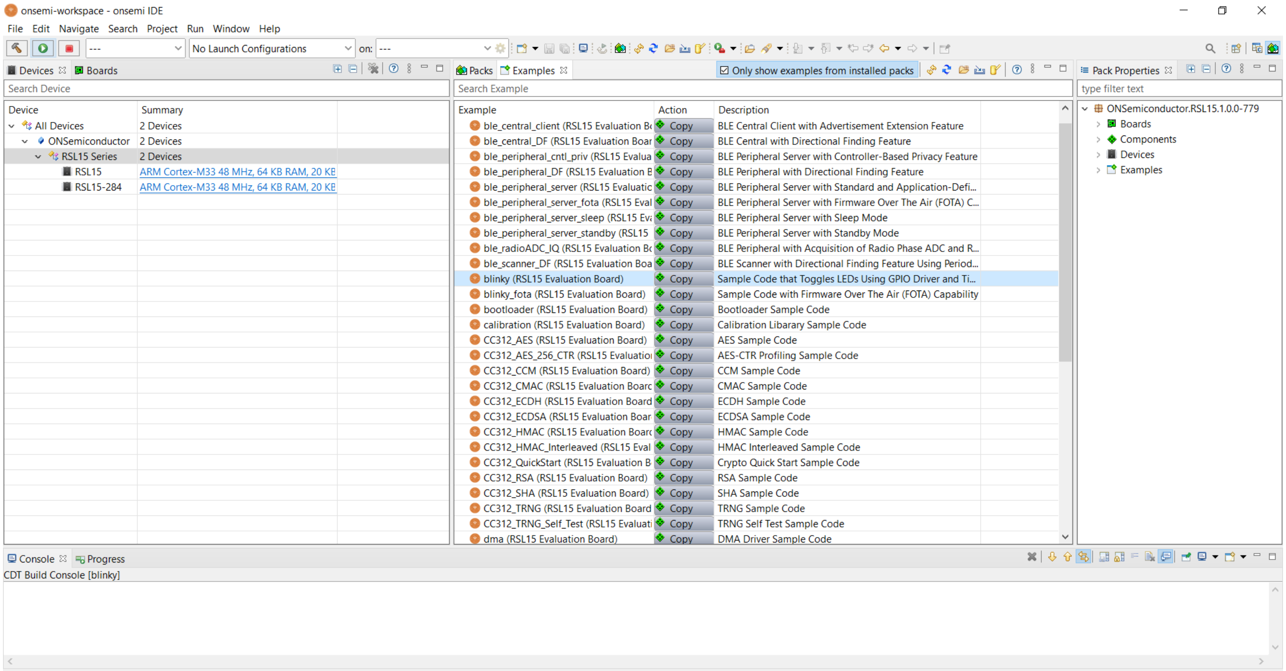
- ONSemiconductor.RSL15.1.0.0-779
  - Boards
  - Components
  - Devices
  - Examples

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. When the confirmation window appears, choose **Copy**.

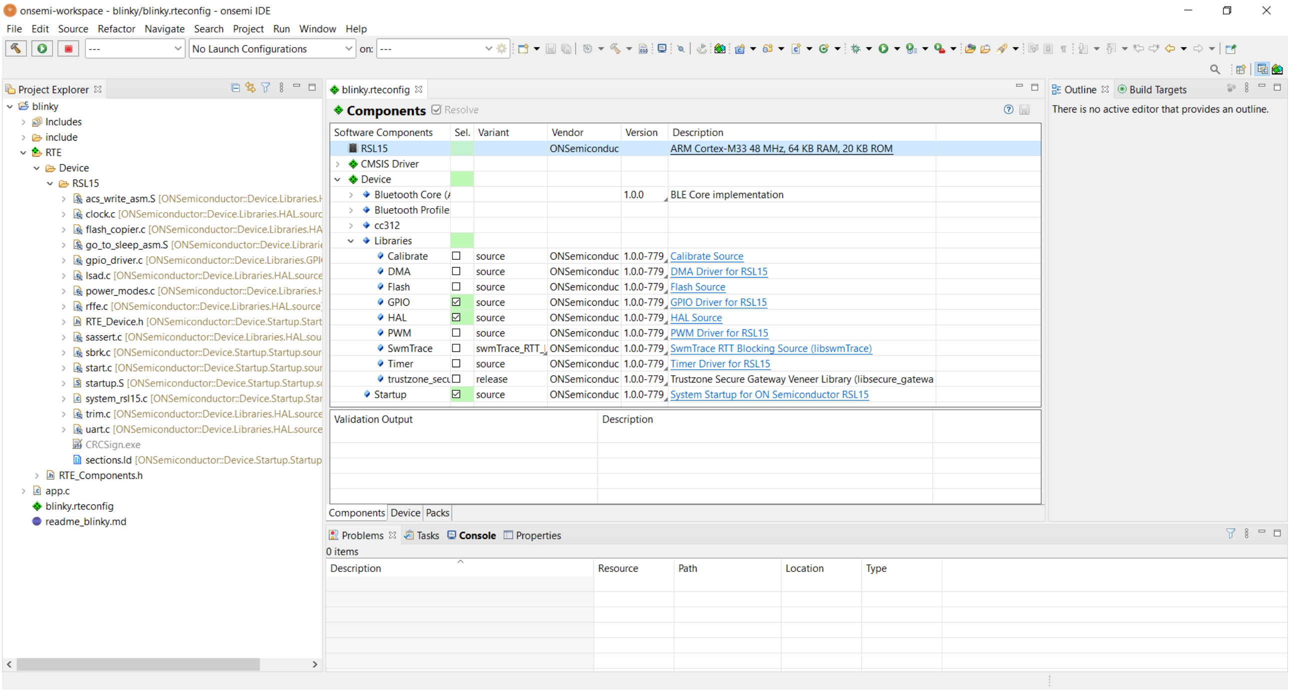

onsemi-workspace - onsemi IDE
File Edit Navigate Search Project Run Window Help
No Launch Configurations
on: ---
Devices
Boards
Packs
Examples
☑ Only show examples from installed packs
Pack Properties
Search Device
Search Example
type filter text
ONSemiconductor.RSL15.1.0.0-779
Boards
Components
Devices
Examples
Console
Progress
CDT Build Console [blinky]

Device	Summary
All Devices	2 Devices
ONSemiconductor	2 Devices
RSL15 Series	2 Devices
RSL15	ARM Cortex-M33 48 MHz, 64 KB RAM, 20 KB
RSL15-284	ARM Cortex-M33 48 MHz, 64 KB RAM, 20 KB

Example	Action	Description
ble_central_client (RSL15 Evaluation Bo	Copy	BLE Central Client with Advertisement Extension Feature
ble_central_DF (RSL15 Evaluation Boar	Copy	BLE Central with Directional Finding Feature
ble_peripheral_cntl_priv (RSL15 Evalua	Copy	BLE Peripheral Server with Controller-Based Privacy Feature
ble_peripheral_DF (RSL15 Evaluation B	Copy	BLE Peripheral with Directional Finding Feature
ble_peripheral_server (RSL15 Evaluatic	Copy	BLE Peripheral Server with Standard and Application-Defi...
ble_peripheral_server_fota (RSL15 Eval	Copy	BLE Peripheral Server with Firmware Over The Air (FOTA) C...
ble_peripheral_server_sleep (RSL15 Eva	Copy	BLE Peripheral Server with Sleep Mode
ble_peripheral_server_standby (RSL15	Copy	BLE Peripheral Server with Standby Mode
ble_radioADC_IQ (RSL15 Evaluation Bo	Copy	BLE Peripheral with Acquisition of Radio Phase ADC and R...
ble_scanner_DF (RSL15 Evaluation Boa	Copy	BLE Scanner with Directional Finding Feature Using Period...
blinky (RSL15 Evaluation Board)	Copy	Sample Code that Toggles LEDs Using GPIO Driver and Ti...
blinky_fota (RSL15 Evaluation Board)	Copy	Sample Code with Firmware Over The Air (FOTA) Capability
bootloader (RSL15 Evaluation Board)	Copy	Bootloader Sample Code
calibration (RSL15 Evaluation Board)	Copy	Calibration Libarary Sample Code
CC312_AES (RSL15 Evaluation Board)	Copy	AES Sample Code
CC312_AES_256_CTR (RSL15 Evaluatio	Copy	AES-CTR Profiling Sample Code
CC312_CCM (RSL15 Evaluation Board)	Copy	CCM Sample Code
CC312_CMAC (RSL15 Evaluation Boarc	Copy	CMAC Sample Code
CC312_ECDH (RSL15 Evaluation Board	Copy	ECDH Sample Code
CC312_ECDSA (RSL15 Evaluation Boar	Copy	ECDSA Sample Code
CC312_HMAC (RSL15 Evaluation Boarc	Copy	HMAC Sample Code
CC312_HMAC_Interleaved (RSL15 Eval	Copy	HMAC Interleaved Sample Code
CC312_QuickStart (RSL15 Evaluation B	Copy	Crypto Quick Start Sample Code
CC312_RSA (RSL15 Evaluation Board)	Copy	RSA Sample Code
CC312_SHA (RSL15 Evaluation Board)	Copy	SHA Sample Code
CC312_TRNG (RSL15 Evaluation Board	Copy	TRNG Sample Code
CC312_TRNG_Self_Test (RSL15 Evaluati	Copy	TRNG Self Test Sample Code
dma (RSL15 Evaluation Board)	Copy	DMA Driver Sample Code

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** in the Components view, 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).)


onsemi-workspace - blinky/blinky.rteconfig - onsemi IDE
File Edit Source Refactor Navigate Search Project Run Window Help
No Launch Configurations
Project Explorer
blinky
Includes
include
RTE
Device
RSL15
acs_write_asm.S [ONSemiconductor::Device.Libraries.H
clock.c [ONSemiconductor::Device.Libraries.HAL.sourc
flash_copier.c [ONSemiconductor::Device.Libraries.HA
go_to_sleep_asm.S [ONSemiconductor::Device.Librari
gpio_driver.c [ONSemiconductor::Device.Libraries.GPI
lsad.c [ONSemiconductor::Device.Libraries.HAL.source
power_modes.c [ONSemiconductor::Device.Libraries.H
rffe.c [ONSemiconductor::Device.Libraries.HAL.source]
RTE_Device.h [ONSemiconductor::Device.Startup.Start
sassert.c [ONSemiconductor::Device.Libraries.HAL.sou
sbrk.c [ONSemiconductor::Device.Startup.Startup.sour
start.c [ONSemiconductor::Device.Startup.Startup.sour
startup.S [ONSemiconductor::Device.Startup.Startup.s
system_rsl15.c [ONSemiconductor::Device.Startup.Star
trim.c [ONSemiconductor::Device.Libraries.HAL.source
uart.c [ONSemiconductor::Device.Libraries.HAL.source
CRCSign.exe
sections.ld [ONSemiconductor::Device.Startup.Startup
RTE_Components.h
app.c
blinky.rteconfig
readme_blinky.md
blinky.rteconfig
Components
Resolve
Components Device Packs
Outline Build Targets
Problems Tasks Console Properties

There is no active editor that provides an outline.

Software Components	Sel.	Variant	Vendor	Version	Description
RSL15			ONSemiconduc		ARM Cortex-M33 48 MHz, 64 KB RAM, 20 KB ROM
CMSIS Driver					
Device					
Bluetooth Core (/				1.0.0	BLE Core implementation
Bluetooth Profile					
cc312					
Libraries					
Calibrate	☐	source	ONSemiconduc	1.0.0-779	Calibrate Source
DMA	☐	source	ONSemiconduc	1.0.0-779	DMA Driver for RSL15
Flash	☐	source	ONSemiconduc	1.0.0-779	Flash Source
GPIO	☑	source	ONSemiconduc	1.0.0-779	GPIO Driver for RSL15
HAL	☑	source	ONSemiconduc	1.0.0-779	HAL Source
PWM	☐	source	ONSemiconduc	1.0.0-779	PWM Driver for RSL15
SwmTrace	☐	swmTrace_RTT_	ONSemiconduc	1.0.0-779	SwmTrace RTT Blocking Source (libswmTrace)
Timer	☐	source	ONSemiconduc	1.0.0-779	Timer Driver for RSL15
trustzone_secu	☐	release	ONSemiconduc	1.0.0-779	Trustzone Secure Gateway Veneer Library (libsecure_gatewa
Startup	☑	source	ONSemiconduc	1.0.0-779	System Startup for ON Semiconductor RSL15

Validation Output	Description

0 items

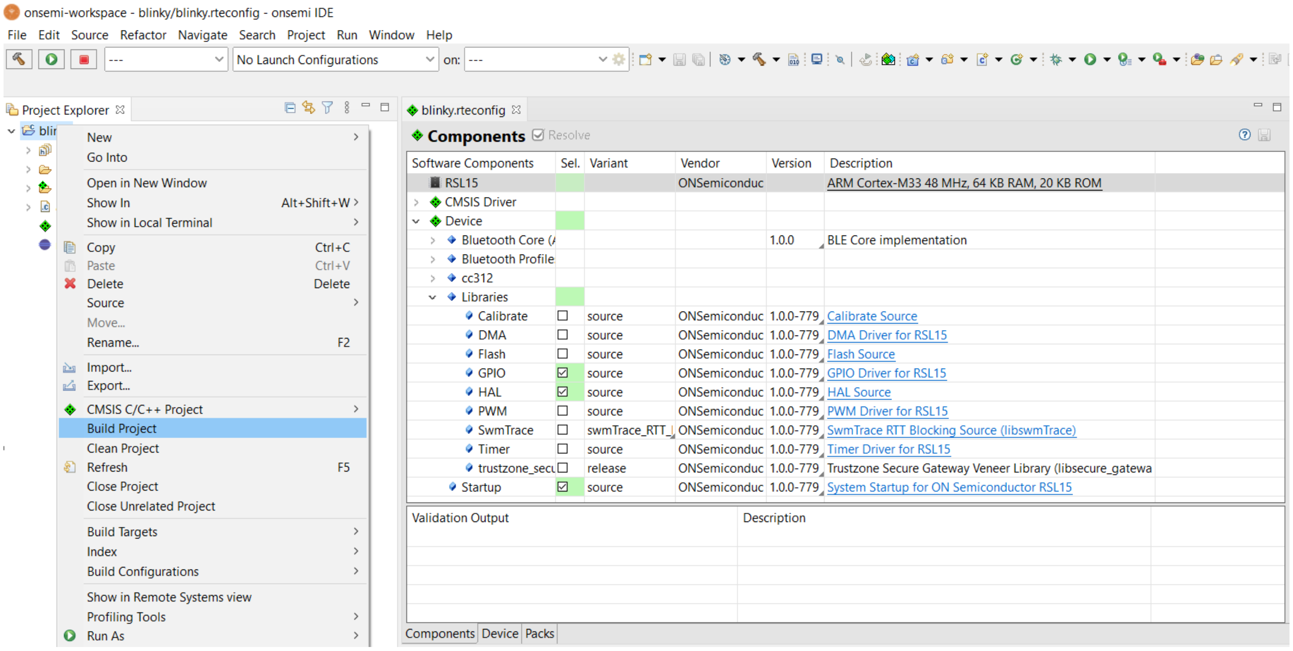
Description	Resource	Path	Location	Type

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. In the Project Explorer view, 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).

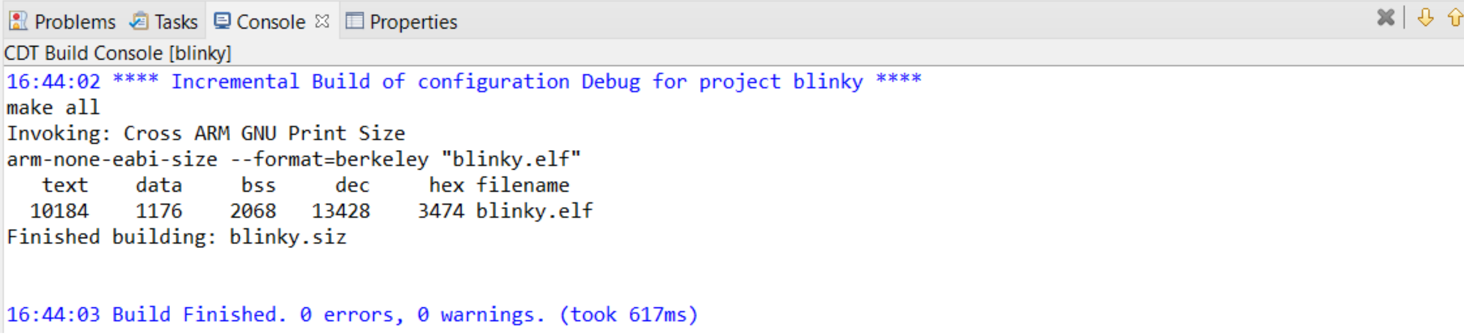


Software Components	Sel.	Variant	Vendor	Version	Description
RSL15			ONSemiconduc		ARM Cortex-M33 48 MHz, 64 KB RAM, 20 KB ROM
CMSIS Driver					
Device					
Bluetooth Core (/				1.0.0	BLE Core implementation
Bluetooth Profile					
cc312					
Libraries					
Calibrate	☐	source	ONSemiconduc	1.0.0-779	Calibrate Source
DMA	☐	source	ONSemiconduc	1.0.0-779	DMA Driver for RSL15
Flash	☐	source	ONSemiconduc	1.0.0-779	Flash Source
GPIO	☑	source	ONSemiconduc	1.0.0-779	GPIO Driver for RSL15
HAL	☑	source	ONSemiconduc	1.0.0-779	HAL Source
PWM	☐	source	ONSemiconduc	1.0.0-779	PWM Driver for RSL15
SwmTrace	☐	swmTrace_RTT_	ONSemiconduc	1.0.0-779	SwmTrace RTT Blocking Source (libswmTrace)
Timer	☐	source	ONSemiconduc	1.0.0-779	Timer Driver for RSL15
trustzone_secu	☐	release	ONSemiconduc	1.0.0-779	Trustzone Secure Gateway Veneer Library (libsecure_gatewa
Startup	☑	source	ONSemiconduc	1.0.0-779	System Startup for ON Semiconductor RSL15

Validation Output	Description

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).



```
16:44:02 **** Incremental Build of configuration Debug for project blinky ****
make all
Invoking: Cross ARM GNU Print Size
arm-none-eabi-size --format=berkeley "blinky.elf"
   text    data     bss     dec     hex filename
  10184    1176    2068   13428    3474 blinky.elf
Finished building: blinky.siz


16:44:03 Build Finished. 0 errors, 0 warnings. (took 617ms)
```

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.

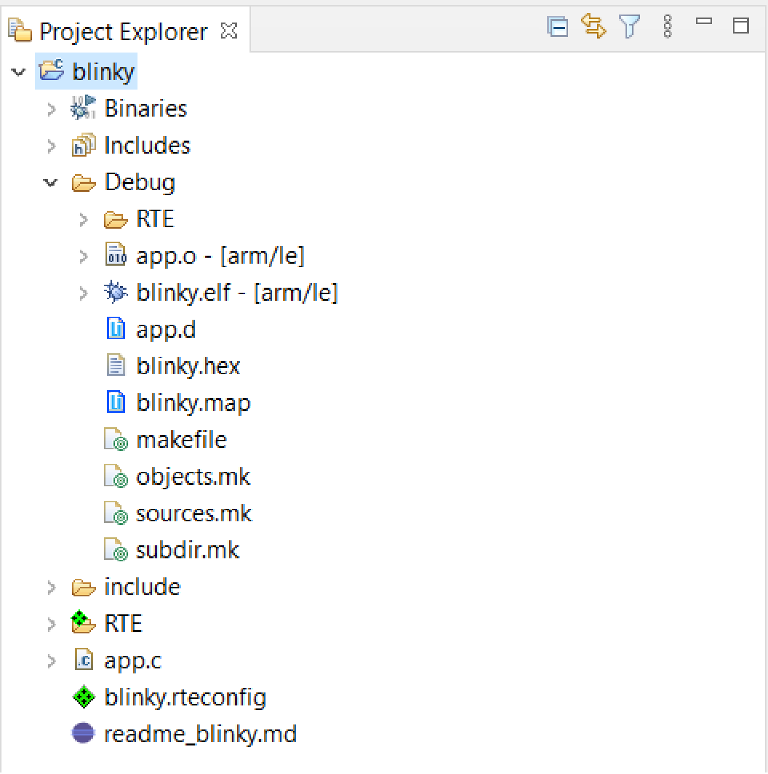


Figure 11. Output Files from Building a Sample Project

4.6 DEBUGGING THE SAMPLE CODE

Debug the application as follows:

1. From the **Launch Configuration** dropdown menu, select **blinky Debug**. From the **Launch Mode** dropdown menu, select **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.

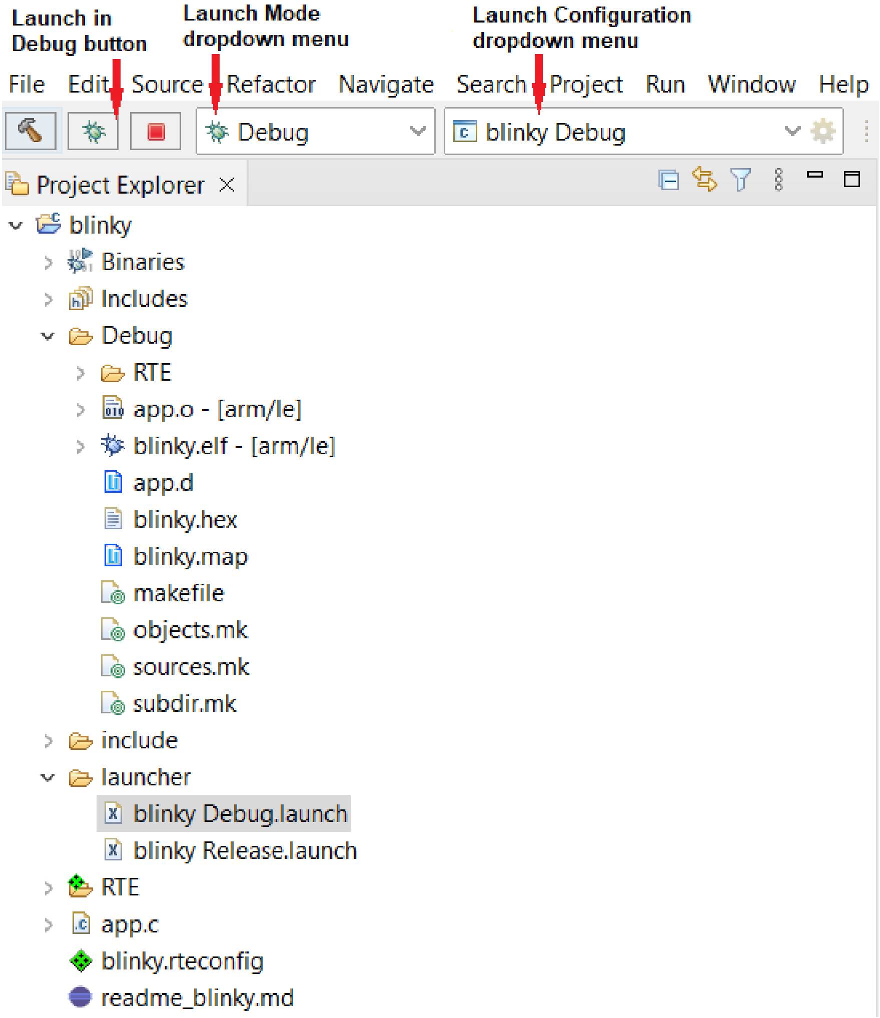


Figure 12. Launching a Debug Session

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.

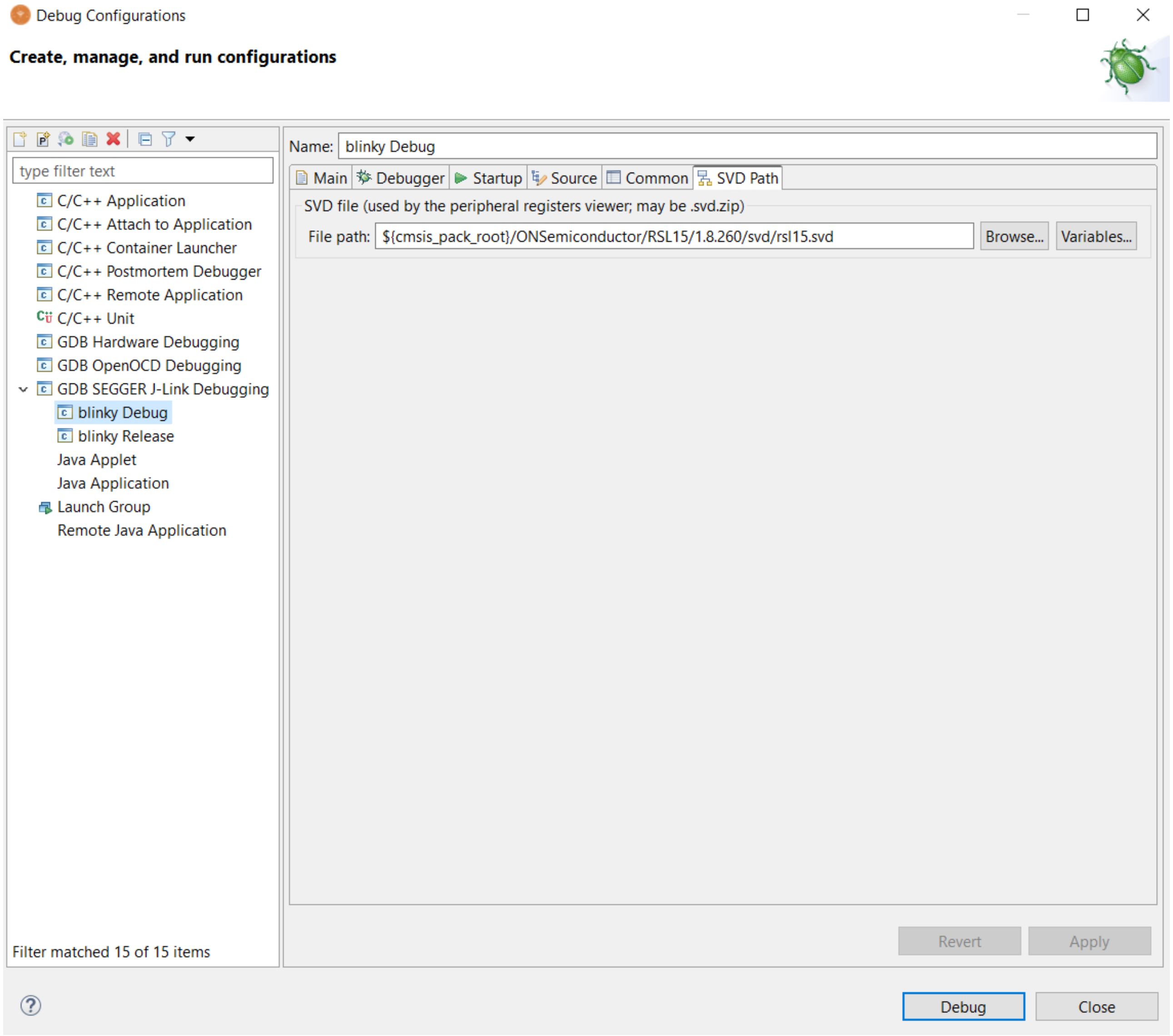
4.7 Viewing Peripheral Registers with the onsemi IDE

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 to use the Peripheral Registers View with the *Blinky* application:

1. If you are not using a onsemi sample application, but one of your own instead, follow these steps to set the location of the system viewer description (SVD) file:
    a. Right click the program name, and choose **Debug As** > **Debug Configuration**.
    b. In the resulting window, choose **GDB SEGGER J-Link Debugging**.
    c. Click on the name of the application you are working with.
    d. In the SVD Path tab, fill in the File Path box with the following:
    `${cmsis_pack_root}/ONSemiconductor/RSL15/`**`<version>`**`/svd/rsl15.svd`
    where **`<version>`** is the CMSIS-Pack version number, and click **Debug**. (See the "Changing the SVD Path" figure (Figure 13).)



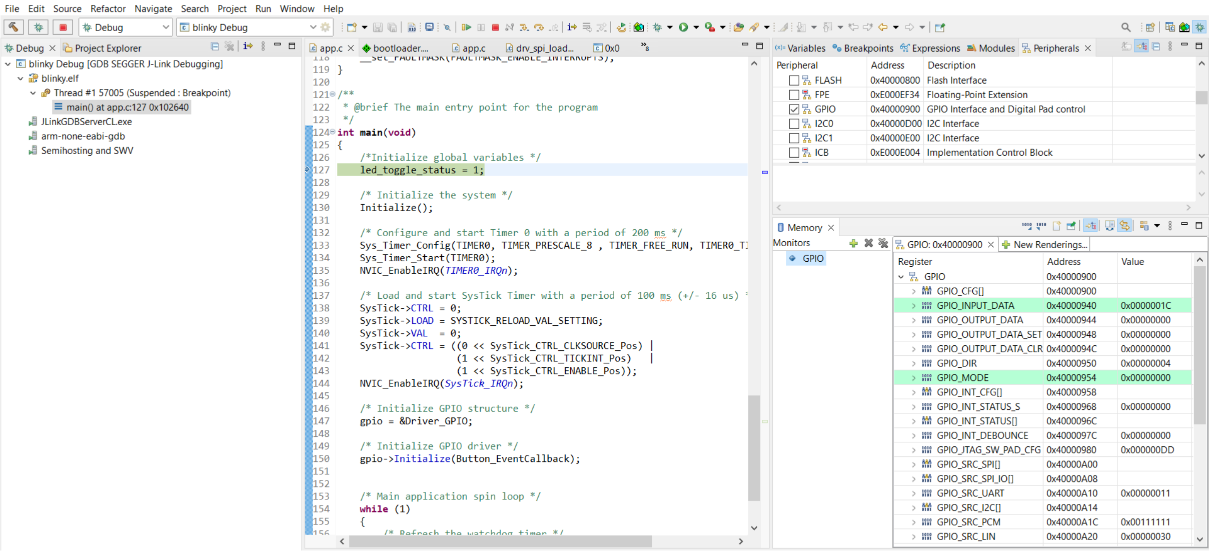
Name: blinky Debug

SVD file (used by the peripheral registers viewer; may be .svd.zip)

File path: ${cmsis_pack_root}/ONSemiconductor/RSL15/1.8.260/svd/rsl15.svd

Figure 13. Changing the SVD Path

2. 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 to open. Now you can see all the RSL15 peripherals displayed.
3. 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 14)) to prevent the console from switching focus away from the Memory window.



blinky Debug [GDB SEGGER J-Link Debugging]
  blinky.elf
    Thread #1 57005 (Suspended : Breakpoint)
      main() at app.c:127 0x102640
  JLinkGDBServerCL.exe
  arm-none-eabi-gdb
  Semihosting and SWV

```
}

/**
 * @brief The main entry point for the program
 */
int main(void)
{
    /*Initialize global variables */
    led_toggle_status = 1;

    /* Initialize the system */
    Initialize();

    /* Configure and start Timer 0 with a period of 200 ms */
    Sys_Timer_Config(TIMER0, TIMER_PRESCALE_8 , TIMER_FREE_RUN, TIMER0_TI
    Sys_Timer_Start(TIMER0);
    NVIC_EnableIRQ(TIMER0_IRQn);

    /* Load and start SysTick Timer with a period of 100 ms (+/- 16 us) *
    SysTick->CTRL = 0;
    SysTick->LOAD = SYSTICK_RELOAD_VAL_SETTING;
    SysTick->VAL  = 0;
    SysTick->CTRL = ((0 << SysTick_CTRL_CLKSOURCE_Pos) |
                     (1 << SysTick_CTRL_TICKINT_Pos)   |
                     (1 << SysTick_CTRL_ENABLE_Pos));
    NVIC_EnableIRQ(SysTick_IRQn);

    /* Initialize GPIO structure */
    gpio = &Driver_GPIO;

    /* Initialize GPIO driver */
    gpio->Initialize(Button_EventCallback);


    /* Main application spin loop */
    while (1)
    {
```

Peripheral	Address	Description
☐ FLASH	0x40000800	Flash Interface
☐ FPE	0xE000EF34	Floating-Point Extension
☑ GPIO	0x40000900	GPIO Interface and Digital Pad control
☐ I2C0	0x40000D00	I2C Interface
☐ I2C1	0x40000E00	I2C Interface
☐ ICB	0xE000E004	Implementation Control Block

Register	Address	Value
GPIO	0x40000900	
GPIO_CFG[]	0x40000900	
GPIO_INPUT_DATA	0x40000940	0x0000001C
GPIO_OUTPUT_DATA	0x40000944	0x00000000
GPIO_OUTPUT_DATA_SET	0x40000948	0x00000000
GPIO_OUTPUT_DATA_CLR	0x4000094C	0x00000000
GPIO_DIR	0x40000950	0x00000004
GPIO_MODE	0x40000954	0x00000000
GPIO_INT_CFG[]	0x40000958	
GPIO_INT_STATUS_S	0x40000968	0x00000000
GPIO_INT_STATUS[]	0x4000096C	
GPIO_INT_DEBOUNCE	0x4000097C	0x00000000
GPIO_JTAG_SW_PAD_CFG	0x40000980	0x000000DD
GPIO_SRC_SPI[]	0x40000A00	
GPIO_SRC_SPI_IO[]	0x40000A08	
GPIO_SRC_UART	0x40000A10	0x00000011
GPIO_SRC_I2C[]	0x40000A14	
GPIO_SRC_PCM	0x40000A1C	0x00111111
GPIO_SRC_LIN	0x40000A20	0x00000030

Figure 14. Peripheral Registers View Perspective in Debug Session After Setting the SVD Path

4. To see or change the GPIO register status, choose **GPIO** and expand the **GPIO** > **GPIO_OUTPUT_DATA** register in the Memory window.
5. Add a breakpoint at this line of code:

```
gpio->ToggleValue((GPIO_SEL_t)(SYSTICK_STATES_GPIO))
```

To add the breakpoint, double-click to the left of the line number. A small circle appears there, indicating that a breakpoint has been set. This is shown in the "Breakpoint Added at a Line of Code" figure (Figure 15).

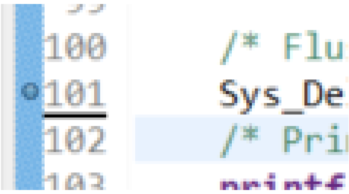
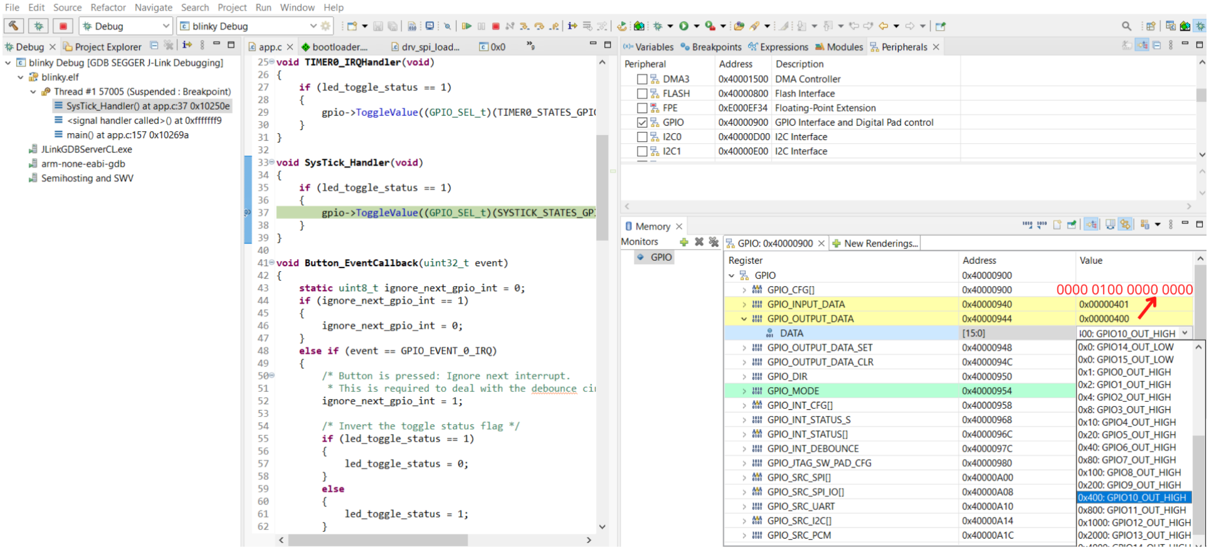


Figure 15. Breakpoint Added at a Line of Code

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

6. 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 16)).



blinky Debug [GDB SEGGER J-Link Debugging]
  blinky.elf
    Thread #1 57005 (Suspended : Breakpoint)
      SysTick_Handler() at app.c:37 0x10250e
      <signal handler called>() at 0xfffffff9
      main() at app.c:157 0x10269a
  JLinkGDBServerCL.exe
  arm-none-eabi-gdb
  Semihosting and SWV

```
void TIMER0_IRQHandler(void)
{
    if (led_toggle_status == 1)
    {
        gpio->ToggleValue((GPIO_SEL_t)(TIMER0_STATES_GPI(
    }
}

void SysTick_Handler(void)
{
    if (led_toggle_status == 1)
    {
        gpio->ToggleValue((GPIO_SEL_t)(SYSTICK_STATES_GPI
    }
}

void Button_EventCallback(uint32_t event)
{
    static uint8_t ignore_next_gpio_int = 0;
    if (ignore_next_gpio_int == 1)
    {
        ignore_next_gpio_int = 0;
    }
    else if (event == GPIO_EVENT_0_IRQ)
    {
        /* Button is pressed: Ignore next interrupt.
         * This is required to deal with the debounce ci
        ignore_next_gpio_int = 1;

        /* Invert the toggle status flag */
        if (led_toggle_status == 1)
        {
            led_toggle_status = 0;
        }
        else
        {
            led_toggle_status = 1;
        }
```

	Peripheral	Address	Description
☐	DMA3	0x40001500	DMA Controller
☐	FLASH	0x40000800	Flash Interface
☐	FPE	0xE000EF34	Floating-Point Extension
☑	GPIO	0x40000900	GPIO Interface and Digital Pad control
☐	I2C0	0x40000D00	I2C Interface
☐	I2C1	0x40000E00	I2C Interface

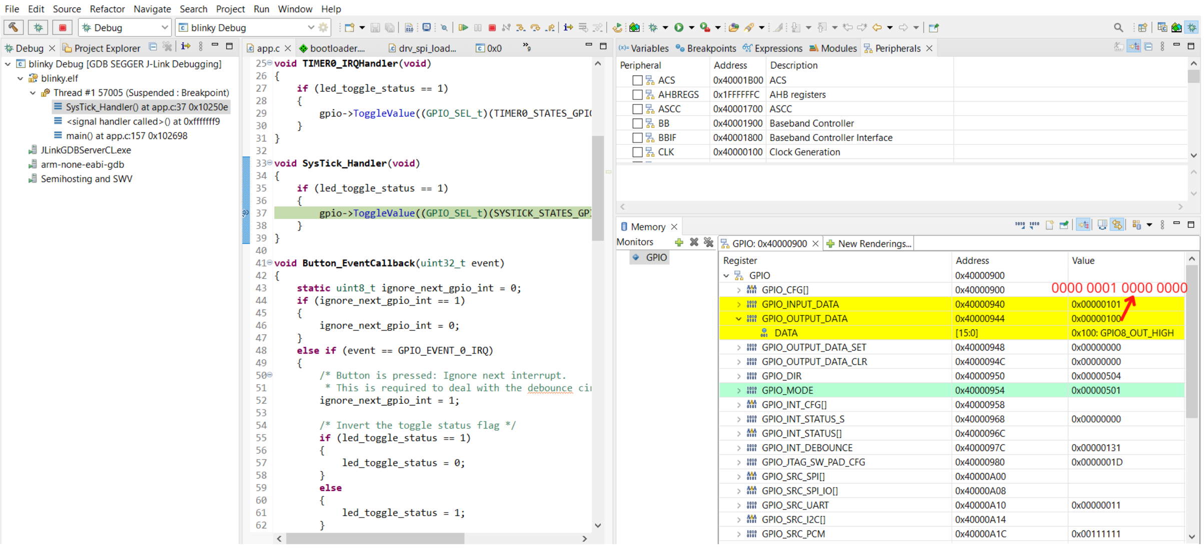
0000 0100 0000 0000

Register	Address	Value
GPIO	0x40000900	
GPIO_CFG[]	0x40000900	
GPIO_INPUT_DATA	0x40000940	0x00000401
GPIO_OUTPUT_DATA	0x40000944	0x00000400
DATA	[15:0]	400: GPIO10_OUT_HIGH
GPIO_OUTPUT_DATA_SET	0x40000948	
GPIO_OUTPUT_DATA_CLR	0x4000094C	
GPIO_DIR	0x40000950	
GPIO_MODE	0x40000954	
GPIO_INT_CFG[]	0x40000958	
GPIO_INT_STATUS_S	0x40000968	
GPIO_INT_STATUS[]	0x4000096C	
GPIO_INT_DEBOUNCE	0x4000097C	
GPIO_JTAG_SW_PAD_CFG	0x40000980	
GPIO_SRC_SPI[]	0x40000A00	
GPIO_SRC_SPI_IO[]	0x40000A08	
GPIO_SRC_UART	0x40000A10	
GPIO_SRC_I2C[]	0x40000A14	
GPIO_SRC_PCM	0x40000A1C	

0x0: GPIO14_OUT_LOW
0x0: GPIO15_OUT_LOW
0x1: GPIO0_OUT_HIGH
0x2: GPIO1_OUT_HIGH
0x4: GPIO2_OUT_HIGH
0x8: GPIO3_OUT_HIGH
0x10: GPIO4_OUT_HIGH
0x20: GPIO5_OUT_HIGH
0x40: GPIO6_OUT_HIGH
0x80: GPIO7_OUT_HIGH
0x100: GPIO8_OUT_HIGH
0x200: GPIO9_OUT_HIGH
0x400: GPIO10_OUT_HIGH
0x800: GPIO11_OUT_HIGH
0x1000: GPIO12_OUT_HIGH
0x2000: GPIO13_OUT_HIGH

Figure 16. Toggling RSL15 GPIO Using the Peripheral Registers View: Before

7. 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 17) illustrates the view after making the change. You can observe that the LED (GPIO8 and GPIO10) on your board changes state.



blinky Debug [GDB SEGGER J-Link Debugging]
blinky.elf
Thread #1 57005 (Suspended : Breakpoint)
SysTick_Handler() at app.c:37 0x10250e
<signal handler called>() at 0xfffffff9
main() at app.c:157 0x102698
JLinkGDBServerCL.exe
arm-none-eabi-gdb
Semihosting and SWV

```
void TIMER0_IRQHandler(void)
{
    if (led_toggle_status == 1)
    {
        gpio->ToggleValue((GPIO_SEL_t)(TIMER0_STATES_GPI
    }
}

void SysTick_Handler(void)
{
    if (led_toggle_status == 1)
    {
        gpio->ToggleValue((GPIO_SEL_t)(SYSTICK_STATES_GP
    }
}

void Button_EventCallback(uint32_t event)
{
    static uint8_t ignore_next_gpio_int = 0;
    if (ignore_next_gpio_int == 1)
    {
        ignore_next_gpio_int = 0;
    }
    else if (event == GPIO_EVENT_0_IRQ)
    {
        /* Button is pressed: Ignore next interrupt.
         * This is required to deal with the debounce ci
        ignore_next_gpio_int = 1;

        /* Invert the toggle status flag */
        if (led_toggle_status == 1)
        {
            led_toggle_status = 0;
        }
        else
        {
            led_toggle_status = 1;
        }
```

Peripheral	Address	Description
☐ ACS	0x40001B00	ACS
☐ AHBREGS	0x1FFFFFFC	AHB registers
☐ ASCC	0x40001700	ASCC
☐ BB	0x40001900	Baseband Controller
☐ BBIF	0x40001800	Baseband Controller Interface
☐ CLK	0x40000100	Clock Generation

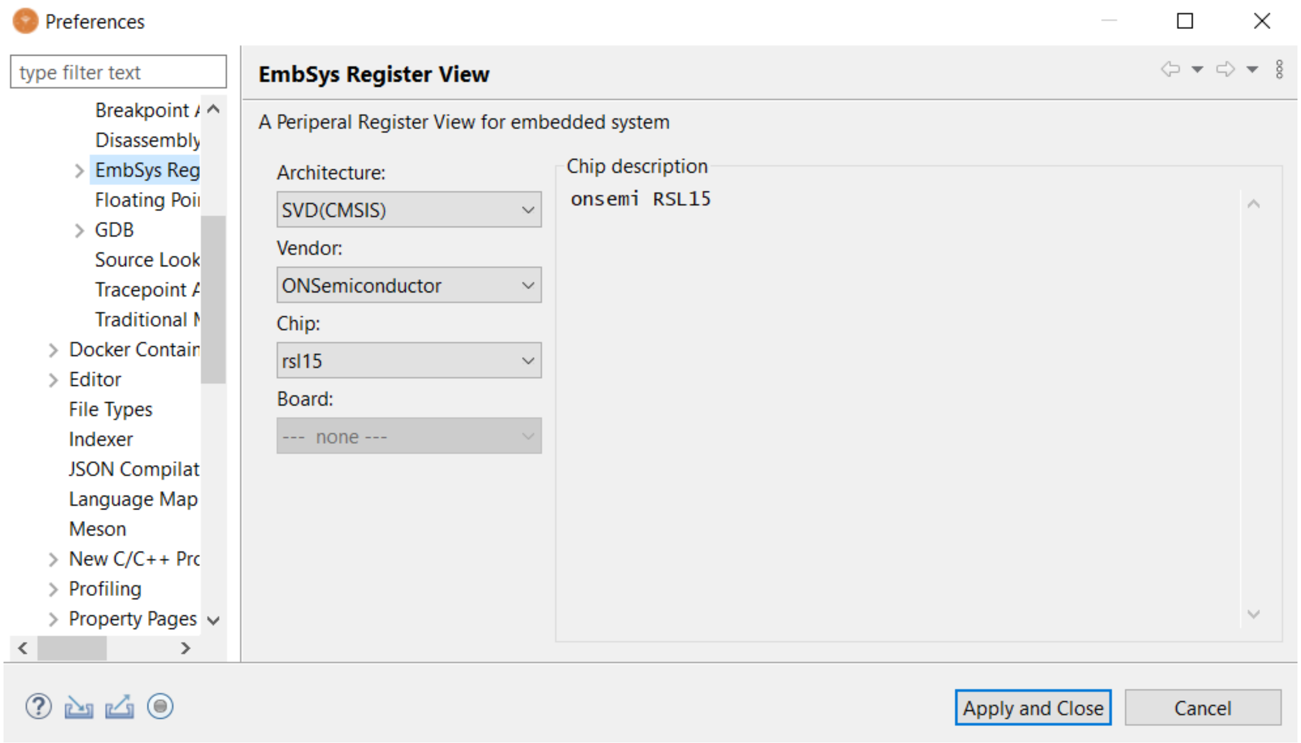
Register	Address	Value
GPIO	0x40000900	
GPIO_CFG[]	0x40000900	
GPIO_INPUT_DATA	0x40000940	0x00000101
GPIO_OUTPUT_DATA	0x40000944	0x00000100
DATA	[15:0]	0x100: GPIO8_OUT_HIGH
GPIO_OUTPUT_DATA_SET	0x40000948	0x00000000
GPIO_OUTPUT_DATA_CLR	0x4000094C	0x00000000
GPIO_DIR	0x40000950	0x00000504
GPIO_MODE	0x40000954	0x00000501
GPIO_INT_CFG[]	0x40000958	
GPIO_INT_STATUS_S	0x40000968	0x00000000
GPIO_INT_STATUS[]	0x4000096C	
GPIO_INT_DEBOUNCE	0x4000097C	0x00000131
GPIO_JTAG_SW_PAD_CFG	0x40000980	0x0000001D
GPIO_SRC_SPI[]	0x40000A00	
GPIO_SRC_SPI_IO[]	0x40000A08	
GPIO_SRC_UART	0x40000A10	0x00000011
GPIO_SRC_I2C[]	0x40000A14	
GPIO_SRC_PCM	0x40000A1C	0x00111111

Figure 17. 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 18):
    - **Architecture:** `SVD(CMSIS)`
    - **Vendor:** `ONSemiconductor`
    - **Chip:** `rsl15`
5. Click **Apply and Close**.



EmbSys Register View

A Periperal Register View for embedded system

Architecture: SVD(CMSIS)

Vendor: ONSemiconductor

Chip: rsl15

Board: --- none ---

Chip description: `onsemi RSL15`

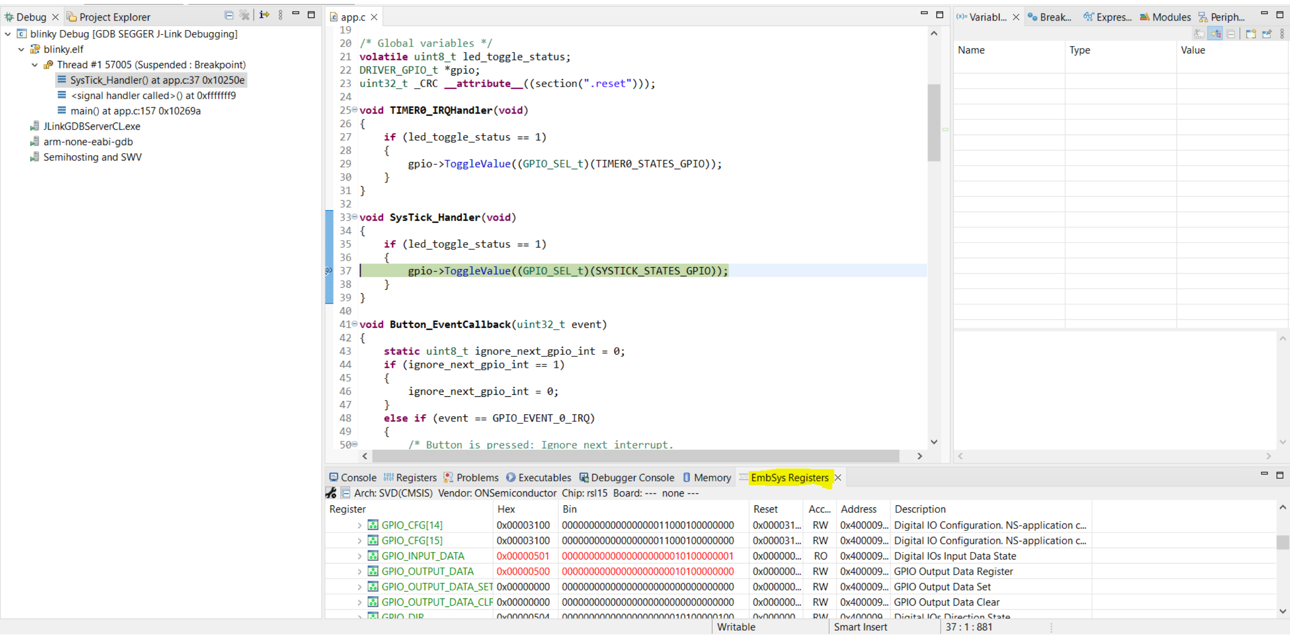
Figure 18. Setting up the EmbSys Register View Plugin

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 19).



blinky Debug [GDB SEGGER J-Link Debugging]
- blinky.elf
  - Thread #1 57005 (Suspended : Breakpoint)
    - SysTick_Handler() at app.c:37 0x10250e
    - <signal handler called>() at 0xfffffff9
    - main() at app.c:157 0x10269a
- JLinkGDBServerCL.exe
- arm-none-eabi-gdb
- Semihosting and SWV

```c

/* Global variables */
volatile uint8_t led_toggle_status;
DRIVER_GPIO_t *gpio;
uint32_t _CRC __attribute__((section(".reset")));

void TIMER0_IRQHandler(void)
{
    if (led_toggle_status == 1)
    {
        gpio->ToggleValue((GPIO_SEL_t)(TIMER0_STATES_GPIO));
    }
}

void SysTick_Handler(void)
{
    if (led_toggle_status == 1)
    {
        gpio->ToggleValue((GPIO_SEL_t)(SYSTICK_STATES_GPIO));
    }
}

void Button_EventCallback(uint32_t event)
{
    static uint8_t ignore_next_gpio_int = 0;
    if (ignore_next_gpio_int == 1)
    {
        ignore_next_gpio_int = 0;
    }
    else if (event == GPIO_EVENT_0_IRQ)
    {
        /* Button is pressed: Ignore next interrupt.
```

Arch: SVD(CMSIS) Vendor: ONSemiconductor Chip: rsl15 Board: --- none ---

Register	Hex	Bin	Reset	Acc...	Address	Description
GPIO_CFG[14]	0x00003100	00000000000000000011000100000000	0x000031...	RW	0x400009...	Digital IO Configuration. NS-application c...
GPIO_CFG[15]	0x00003100	00000000000000000011000100000000	0x000031...	RW	0x400009...	Digital IO Configuration. NS-application c...
GPIO_INPUT_DATA	0x00000501	00000000000000000000010100000001	0x000000...	RO	0x400009...	Digital IOs Input Data State
GPIO_OUTPUT_DATA	0x00000500	00000000000000000000010100000000	0x000000...	RW	0x400009...	GPIO Output Data Register
GPIO_OUTPUT_DATA_SET	0x00000000	00000000000000000000000000000000	0x000000...	RW	0x400009...	GPIO Output Data Set
GPIO_OUTPUT_DATA_CLR	0x00000000	00000000000000000000000000000000	0x000000...	RW	0x400009...	GPIO Output Data Clear
GPIO_DIR	0x00000504	00000000000000000000010100000100	0x000000...	RW	0x400009...	Digital IOs Direction State

Figure 19. 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 the version of J-Link recommended for the Keil µVision IDE, 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 20).

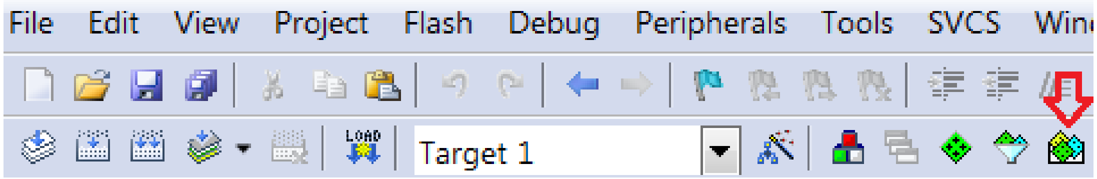


Figure 20. Pack Installer Icon

3. Click on **File** > **Import** (see the "Importing the RSL15 CMSIS-Pack" figure (Figure 21)), 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 22)). **<version>** is the RSL15 version, such as 2.2.347.

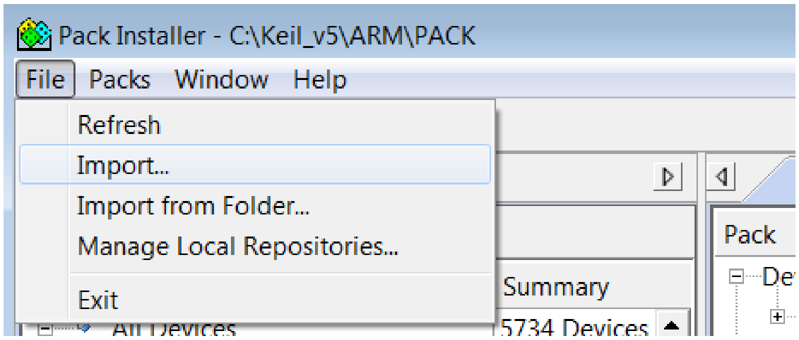


Figure 21. Importing the RSL15 CMSIS-Pack

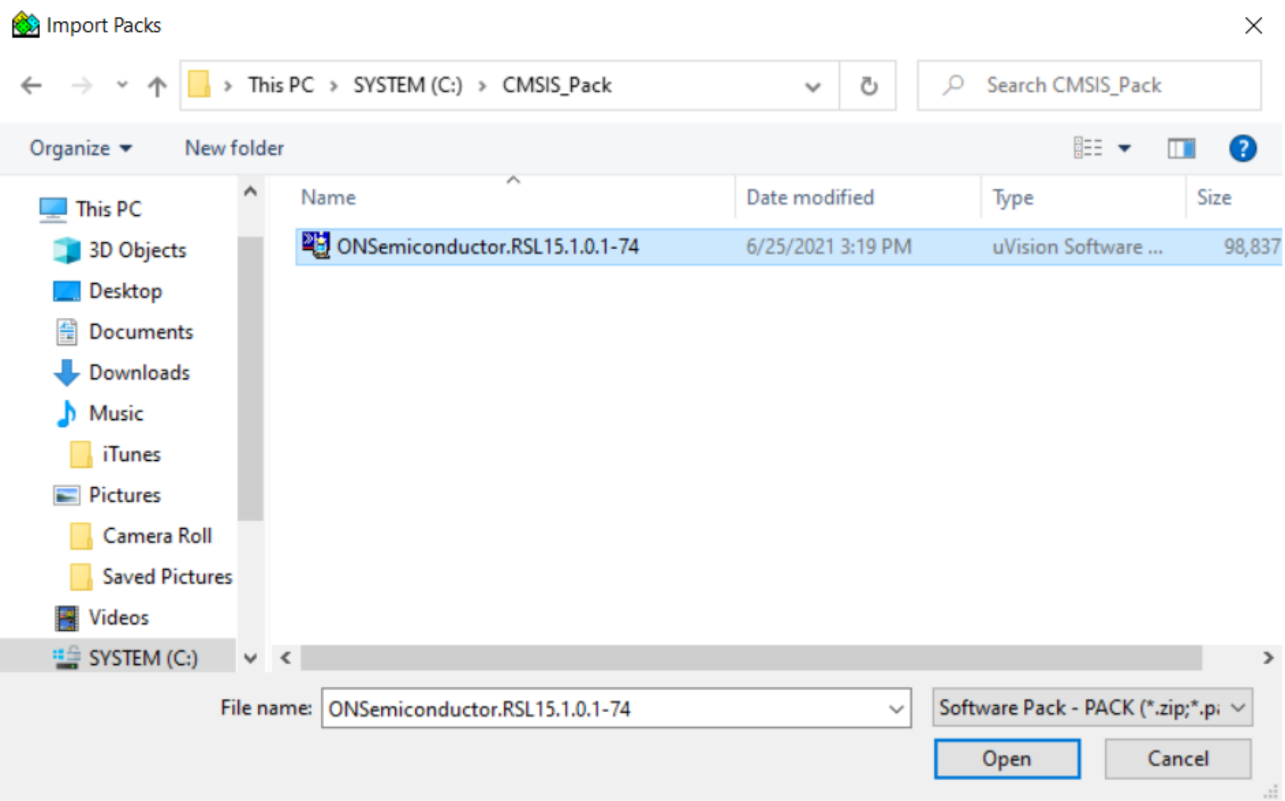


Figure 22. Installing the RSL15 CMSIS-Pack for the Keil µVision IDE

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 23) to update your pack proprieties.

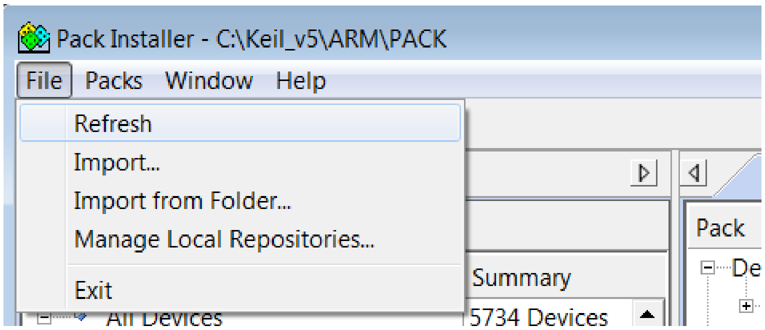
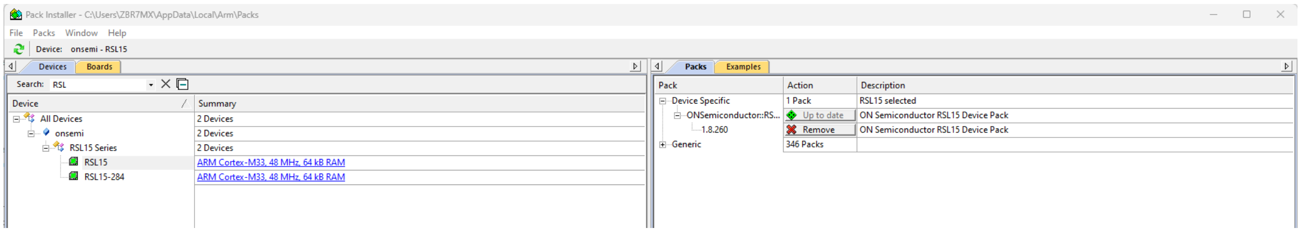


Figure 23. Refresh Pack after installation

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 **onsemi** > **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 24) illustrates what the Pack Installer perspective looks like after installation.



Device	Summary
All Devices	2 Devices
onsemi	2 Devices
RSL15 Series	2 Devices
RSL15	ARM Cortex-M33, 48 MHz, 64 kB RAM
RSL15-284	ARM Cortex-M33, 48 MHz, 64 kB RAM

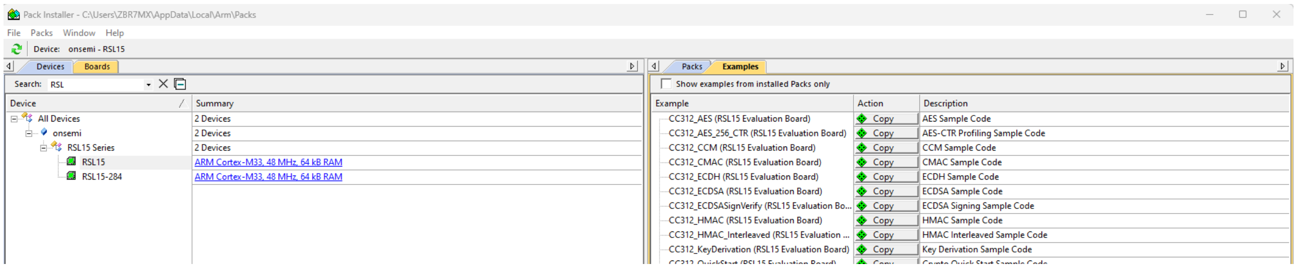
Pack	Action	Description
Device Specific	1 Pack	RSL15 selected
ONSemiconductor::RS...	Up to date	ON Semiconductor RSL15 Device Pack
1.8.260	Remove	ON Semiconductor RSL15 Device Pack
Generic	346 Packs	

Figure 24. 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 25)). Choose a destination folder for a copy of the sample code. We recommend leaving the **Launch µVision** checkbox checked.



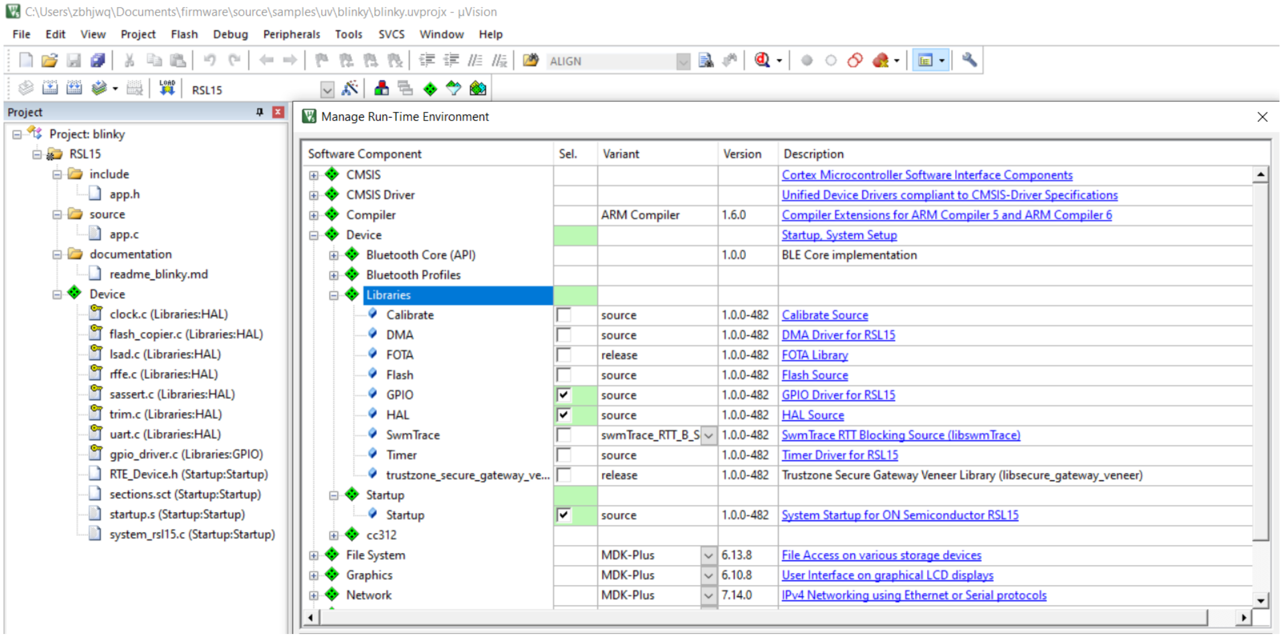
Device	Summary
All Devices	2 Devices
onsemi	2 Devices
RSL15 Series	2 Devices
RSL15	ARM Cortex-M33, 48 MHz, 64 kB RAM
RSL15-284	ARM Cortex-M33, 48 MHz, 64 kB RAM

☐ Show examples from installed Packs only

Example	Action	Description
CC312_AES (RSL15 Evaluation Board)	Copy	AES Sample Code
CC312_AES_256_CTR (RSL15 Evaluation Board)	Copy	AES-CTR Profiling Sample Code
CC312_CCM (RSL15 Evaluation Board)	Copy	CCM Sample Code
CC312_CMAC (RSL15 Evaluation Board)	Copy	CMAC Sample Code
CC312_ECDH (RSL15 Evaluation Board)	Copy	ECDH Sample Code
CC312_ECDSA (RSL15 Evaluation Board)	Copy	ECDSA Sample Code
CC312_ECDSASignVerify (RSL15 Evaluation Bo...	Copy	ECDSA Signing Sample Code
CC312_HMAC (RSL15 Evaluation Board)	Copy	HMAC Sample Code
CC312_HMAC_Interleaved (RSL15 Evaluation ...	Copy	HMAC Interleaved Sample Code
CC312_KeyDerivation (RSL15 Evaluation Board)	Copy	Key Derivation Sample Code
CC312_QuickStart (RSL15 Evaluation Board)	Copy	Crypto Quick Start Sample Code

Figure 25. 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 26)).


C:\Users\zbhjwq\Documents\firmware\source\samples\uv\blinky\blinky.uvprojx - µVision
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
ALIGN
RSL15
Manage Run-Time Environment

Project
- Project: blinky
  - RSL15
    - include
      - app.h
    - source
      - app.c
    - documentation
      - readme_blinky.md
    - Device
      - clock.c (Libraries:HAL)
      - flash_copier.c (Libraries:HAL)
      - lsad.c (Libraries:HAL)
      - rffe.c (Libraries:HAL)
      - sassert.c (Libraries:HAL)
      - trim.c (Libraries:HAL)
      - uart.c (Libraries:HAL)
      - gpio_driver.c (Libraries:GPIO)
      - RTE_Device.h (Startup:Startup)
      - sections.sct (Startup:Startup)
      - startup.s (Startup:Startup)
      - system_rsl15.c (Startup:Startup)

Software Component	Sel.	Variant	Version	Description
CMSIS				Cortex Microcontroller Software Interface Components
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler		ARM Compiler	1.6.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
Bluetooth Core (API)			1.0.0	BLE Core implementation
Bluetooth Profiles				
Libraries				
Calibrate	☐	source	1.0.0-482	Calibrate Source
DMA	☐	source	1.0.0-482	DMA Driver for RSL15
FOTA	☐	release	1.0.0-482	FOTA Library
Flash	☐	source	1.0.0-482	Flash Source
GPIO	☑	source	1.0.0-482	GPIO Driver for RSL15
HAL	☑	source	1.0.0-482	HAL Source
SwmTrace	☐	swmTrace_RTT_B_S	1.0.0-482	SwmTrace RTT Blocking Source (libswmTrace)
Timer	☐	source	1.0.0-482	Timer Driver for RSL15
trustzone_secure_gateway_ve...	☐	release	1.0.0-482	Trustzone Secure Gateway Veneer Library (libsecure_gateway_veneer)
Startup				
Startup	☑	source	1.0.0-482	System Startup for ON Semiconductor RSL15
cc312				
File System		MDK-Plus	6.13.8	File Access on various storage devices
Graphics		MDK-Plus	6.10.8	User Interface on graphical LCD displays
Network		MDK-Plus	7.14.0	IPv4 Networking using Ethernet or Serial protocols

Figure 26. 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 27).

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.

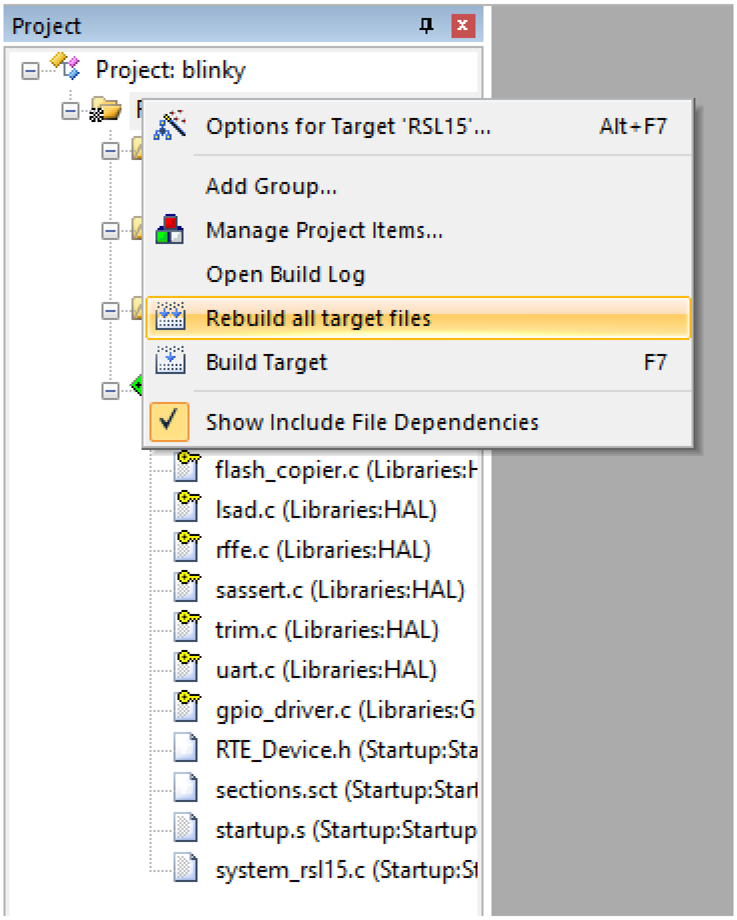
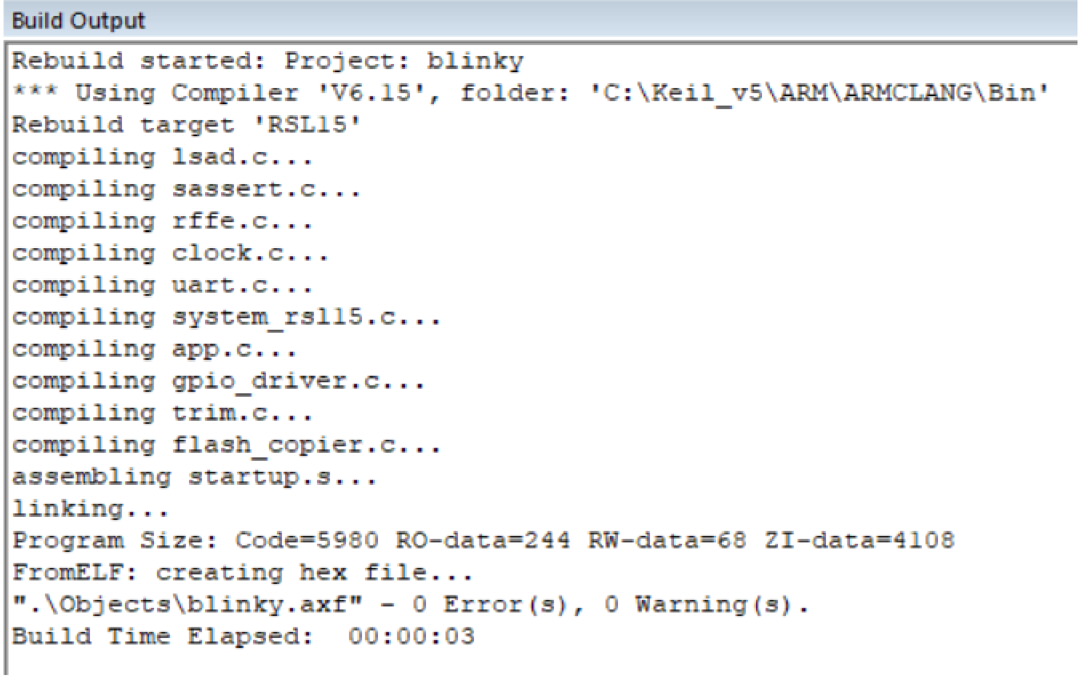

Project
Project: blinky
Options for Target 'RSL15'...
Alt+F7
Add Group...
Manage Project Items...
Open Build Log
Rebuild all target files
Build Target
F7
Show Include File Dependencies
flash_copier.c (Libraries:H
lsad.c (Libraries:HAL)
rffe.c (Libraries:HAL)
sassert.c (Libraries:HAL)
trim.c (Libraries:HAL)
uart.c (Libraries:HAL)
gpio_driver.c (Libraries:G
RTE_Device.h (Startup:Sta
sections.sct (Startup:Start
startup.s (Startup:Startup
system_rsl15.c (Startup:St

Figure 27. 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 28).



```
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 28. 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 29).

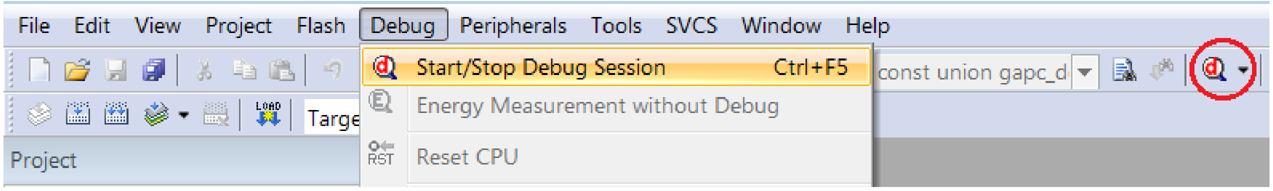
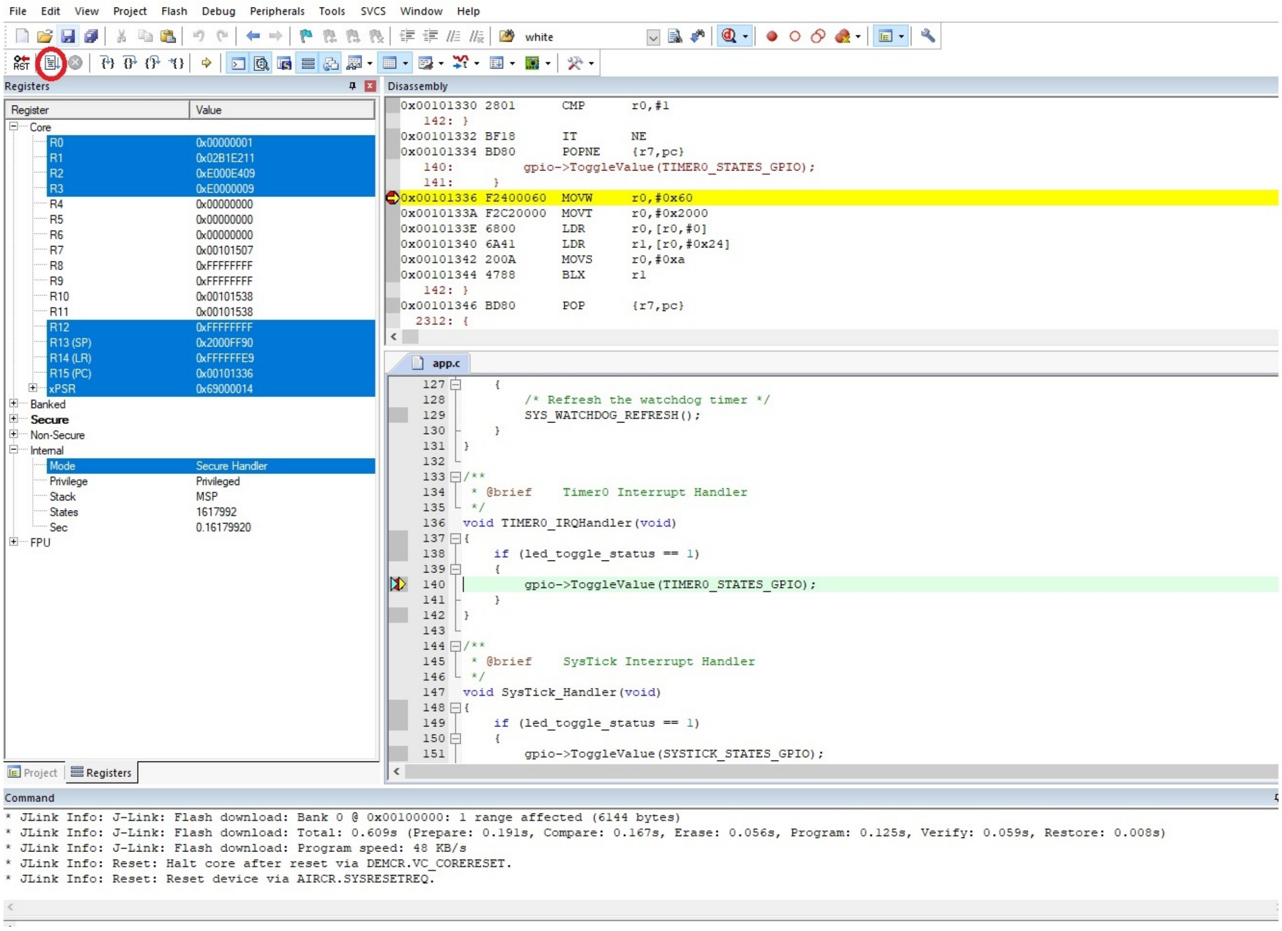


Figure 29. Start/Stop Debug Session Icon

3. The application runs up to the first breakpoint in main, as shown in the "Debug Session in the Keil µVision IDE" figure (Figure 30). 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)`.

   .



Register	Value
Core	
R0	0x00000001
R1	0x02B1E211
R2	0xE000E409
R3	0xE0000009
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00101507
R8	0xFFFFFFFF
R9	0xFFFFFFFF
R10	0x00101538
R11	0x00101538
R12	0xFFFFFFFF
R13 (SP)	0x2000FF90
R14 (LR)	0xFFFFFFE9
R15 (PC)	0x00101336
xPSR	0x69000014
Banked	
Secure	
Non-Secure	
Internal	
Mode	Secure Handler
Privilege	Privileged
Stack	MSP
States	1617992
Sec	0.16179920
FPU	

```
0x00101330 2801      CMP      r0,#1
   142: }
0x00101332 BF18      IT       NE
0x00101334 BD80      POPNE    {r7,pc}
   140:           gpio->ToggleValue(TIMER0_STATES_GPIO);
   141:       }
0x00101336 F2400060  MOVW     r0,#0x60
0x0010133A F2C20000  MOVT     r0,#0x2000
0x0010133E 6800      LDR      r0,[r0,#0]
0x00101340 6A41      LDR      r1,[r0,#0x24]
0x00101342 200A      MOVS     r0,#0xa
0x00101344 4788      BLX      r1
   142: }
0x00101346 BD80      POP      {r7,pc}
  2312: {
```

```
    {
        /* Refresh the watchdog timer */
        SYS_WATCHDOG_REFRESH();
    }
}

/**
 * @brief    Timer0 Interrupt Handler
 */
void TIMER0_IRQHandler(void)
{
    if (led_toggle_status == 1)
    {
        gpio->ToggleValue(TIMER0_STATES_GPIO);
    }
}

/**
 * @brief    SysTick Interrupt Handler
 */
void SysTick_Handler(void)
{
    if (led_toggle_status == 1)
    {
        gpio->ToggleValue(SYSTICK_STATES_GPIO);
```

```
* JLink Info: J-Link: Flash download: Bank 0 @ 0x00100000: 1 range affected (6144 bytes)
* JLink Info: J-Link: Flash download: Total: 0.609s (Prepare: 0.191s, Compare: 0.167s, Erase: 0.056s, Program: 0.125s, Verify: 0.059s, Restore: 0.008s)
* JLink Info: J-Link: Flash download: Program speed: 48 KB/s
* JLink Info: Reset: Halt core after reset via DEMCR.VC_CORERESET.
* JLink Info: Reset: Reset device via AIRCR.SYSRESETREQ.
```

Figure 30. 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 31)) or F8 is not supported for J-Link.

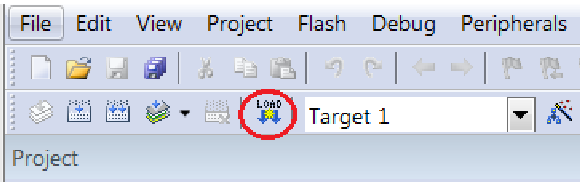


Figure 31. Download Button Not Supported for J-Link

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 the version of J-Link recommendeded for the IAR Embedded Workbench, 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 32).

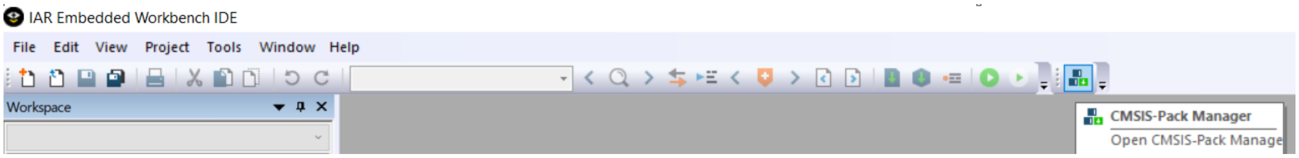


Figure 32. Pack Installer Icon

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 33)). <version> is the RSL15 version.

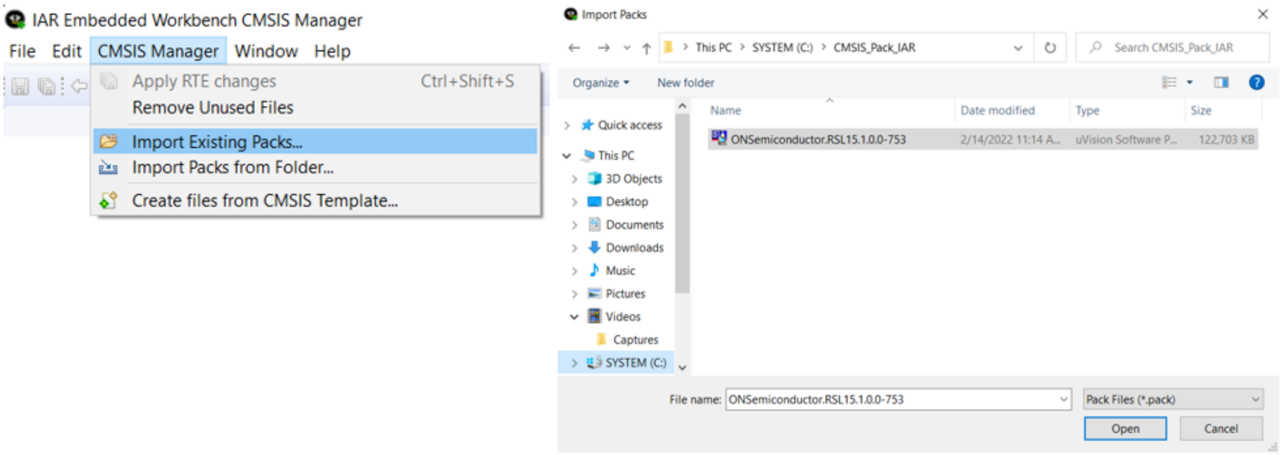


Figure 33. Installing the RSL15 CMSIS_Pack for the IAR Embedded Workbench IDE

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 34)), to update your pack proprieties.

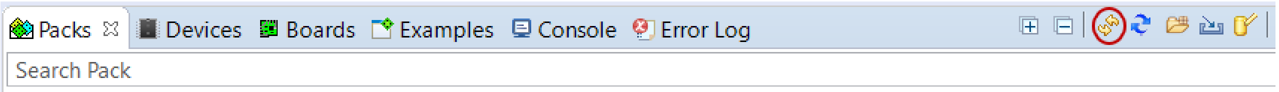
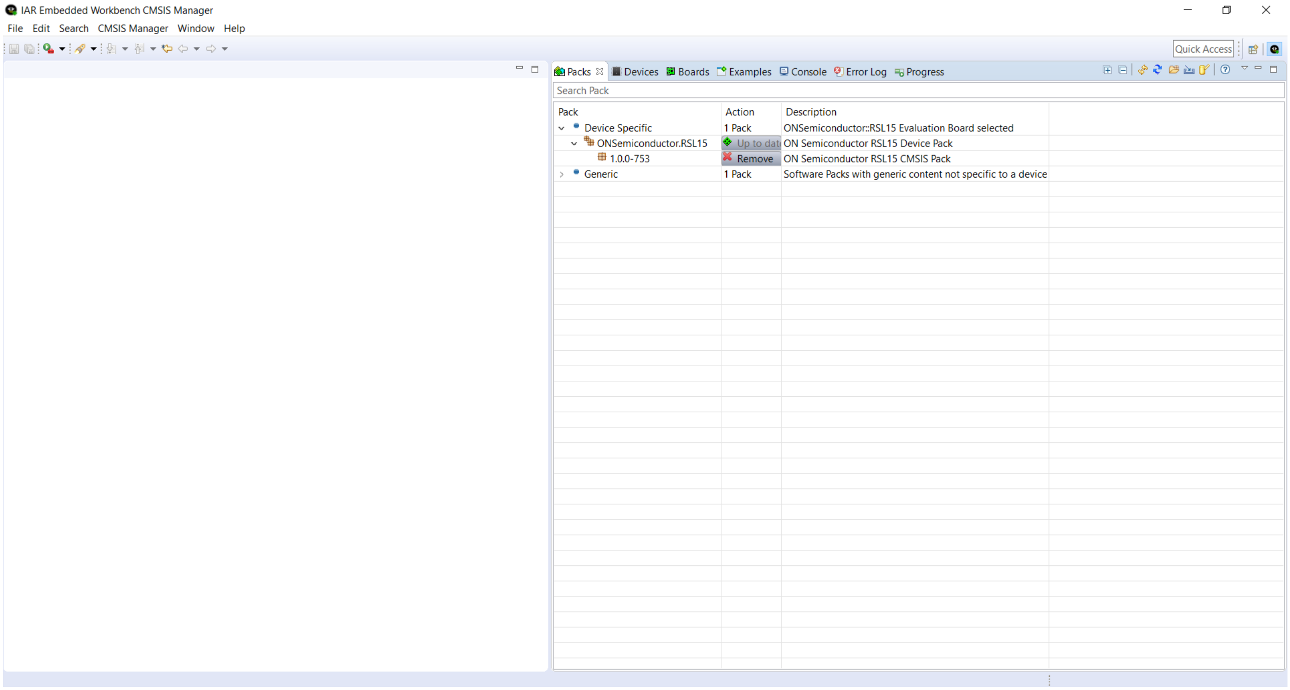


Figure 34. Refresh Pack after Installation

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 35) illustrates what the **Pack Manager** perspective looks like after installation.


IAR Embedded Workbench CMSIS Manager
File Edit Search CMSIS Manager Window Help
Quick Access
Packs
Devices
Boards
Examples
Console
Error Log
Progress
Search Pack

Pack	Action	Description
Device Specific	1 Pack	ONSemiconductor::RSL15 Evaluation Board selected
ONSemiconductor.RSL15	Up to dat	ON Semiconductor RSL15 Device Pack
1.0.0-753	Remove	ON Semiconductor RSL15 CMSIS Pack
Generic	1 Pack	Software Packs with generic content not specific to a device

Figure 35. IAR Embedded Workbench CMSIS Manager after RSL15 CMSIS-Pack is Installed

6.3 BUILDING YOUR FIRST SAMPLE APPLICATION WITH THE IAR EMBEDDED WORKBENCH

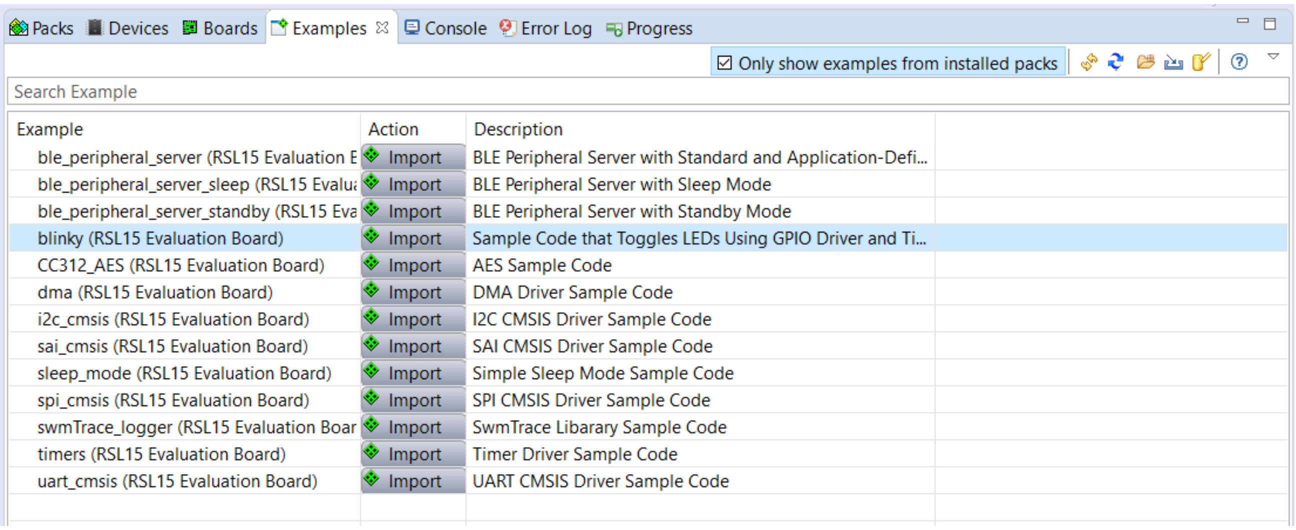
This section guides you through importing and building your first sample application, named *blinky*. This application makes the LED (DIO6) blink on the RSL15 Evaluation and Development Board. The procedure described in this section assumes that you have installed the SDK.

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 36)). Choose a destination folder for a copy of the sample code.



Example	Action	Description
ble_peripheral_server (RSL15 Evaluation E	Import	BLE Peripheral Server with Standard and Application-Defi...
ble_peripheral_server_sleep (RSL15 Evalu	Import	BLE Peripheral Server with Sleep Mode
ble_peripheral_server_standby (RSL15 Eva	Import	BLE Peripheral Server with Standby Mode
blinky (RSL15 Evaluation Board)	Import	Sample Code that Toggles LEDs Using GPIO Driver and Ti...
CC312_AES (RSL15 Evaluation Board)	Import	AES Sample Code
dma (RSL15 Evaluation Board)	Import	DMA Driver Sample Code
i2c_cmsis (RSL15 Evaluation Board)	Import	I2C CMSIS Driver Sample Code
sai_cmsis (RSL15 Evaluation Board)	Import	SAI CMSIS Driver Sample Code
sleep_mode (RSL15 Evaluation Board)	Import	Simple Sleep Mode Sample Code
spi_cmsis (RSL15 Evaluation Board)	Import	SPI CMSIS Driver Sample Code
swmTrace_logger (RSL15 Evaluation Boar	Import	SwmTrace Libarary Sample Code
timers (RSL15 Evaluation Board)	Import	Timer Driver Sample Code
uart_cmsis (RSL15 Evaluation Board)	Import	UART CMSIS Driver Sample Code

Figure 36. 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 38) and the "RTE Configuration for the Blinky Example Project in the IAR Embedded Workbench Window" figure (Figure 37)).

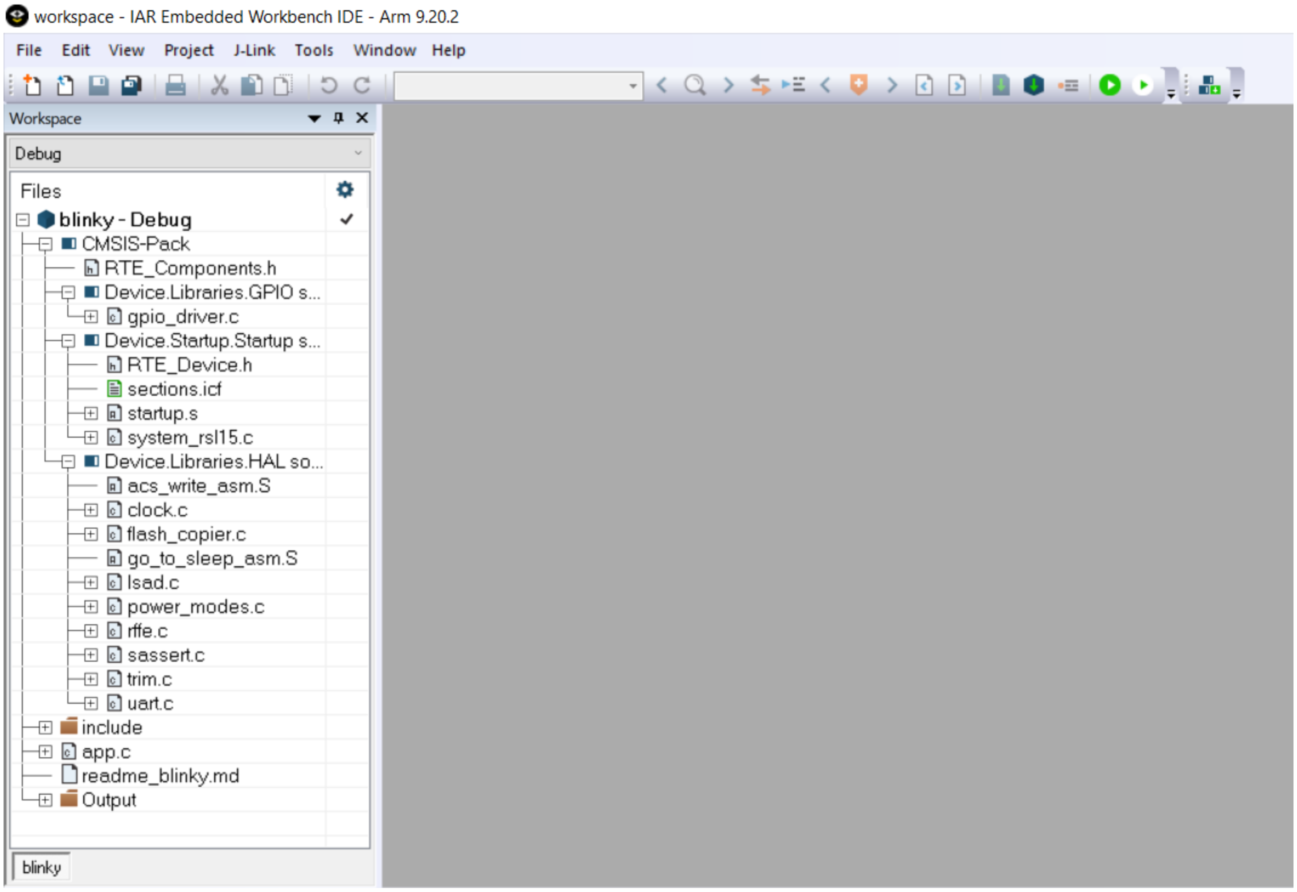
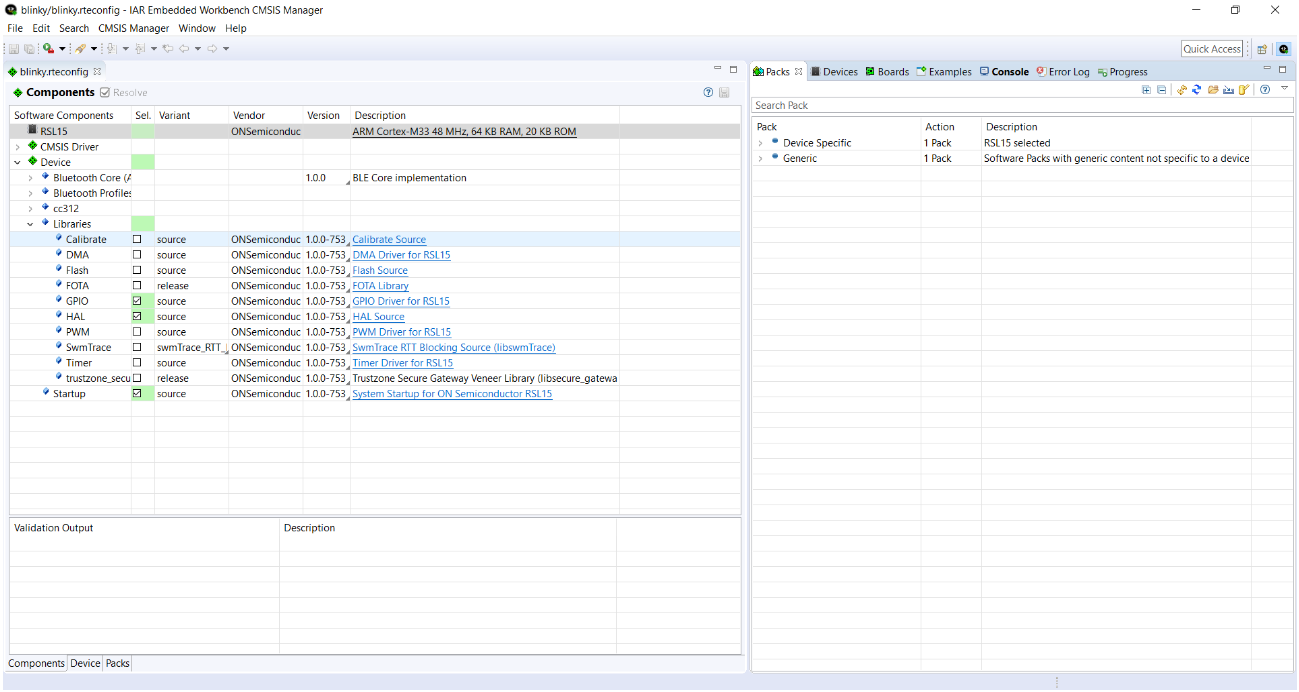


Figure 37. RTE Configuration for the Blinky Example Project in the IAR Embedded Workbench Window



Software Components	Sel.	Variant	Vendor	Version	Description
RSL15			ONSemiconduc		ARM Cortex-M33 48 MHz, 64 KB RAM, 20 KB ROM
CMSIS Driver					
Device					
Bluetooth Core (A				1.0.0	BLE Core implementation
Bluetooth Profiles					
cc312					
Libraries					
Calibrate	☐	source	ONSemiconduc	1.0.0-753	Calibrate Source
DMA	☐	source	ONSemiconduc	1.0.0-753	DMA Driver for RSL15
Flash	☐	source	ONSemiconduc	1.0.0-753	Flash Source
FOTA	☐	release	ONSemiconduc	1.0.0-753	FOTA Library
GPIO	☑	source	ONSemiconduc	1.0.0-753	GPIO Driver for RSL15
HAL	☑	source	ONSemiconduc	1.0.0-753	HAL Source
PWM	☐	source	ONSemiconduc	1.0.0-753	PWM Driver for RSL15
SwmTrace	☐	swmTrace_RTT_	ONSemiconduc	1.0.0-753	SwmTrace RTT Blocking Source (libswmTrace)
Timer	☐	source	ONSemiconduc	1.0.0-753	Timer Driver for RSL15
trustzone_secu	☐	release	ONSemiconduc	1.0.0-753	Trustzone Secure Gateway Veneer Library (libsecure_gatewa
Startup	☑	source	ONSemiconduc	1.0.0-753	System Startup for ON Semiconductor RSL15

Pack	Action	Description
Device Specific	1 Pack	RSL15 selected
Generic	1 Pack	Software Packs with generic content not specific to a device

Figure 38. 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 39).

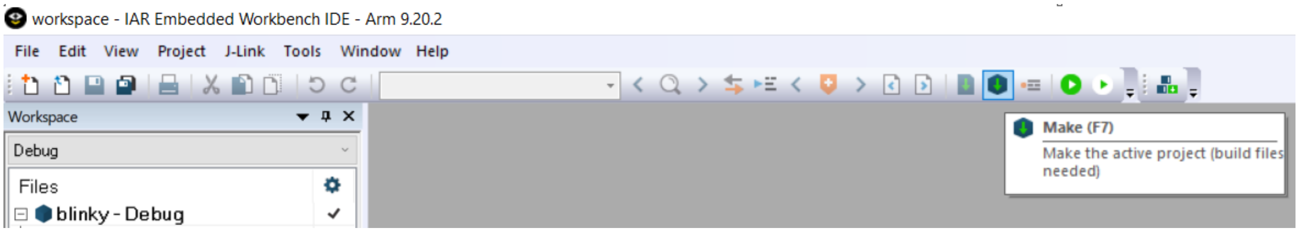
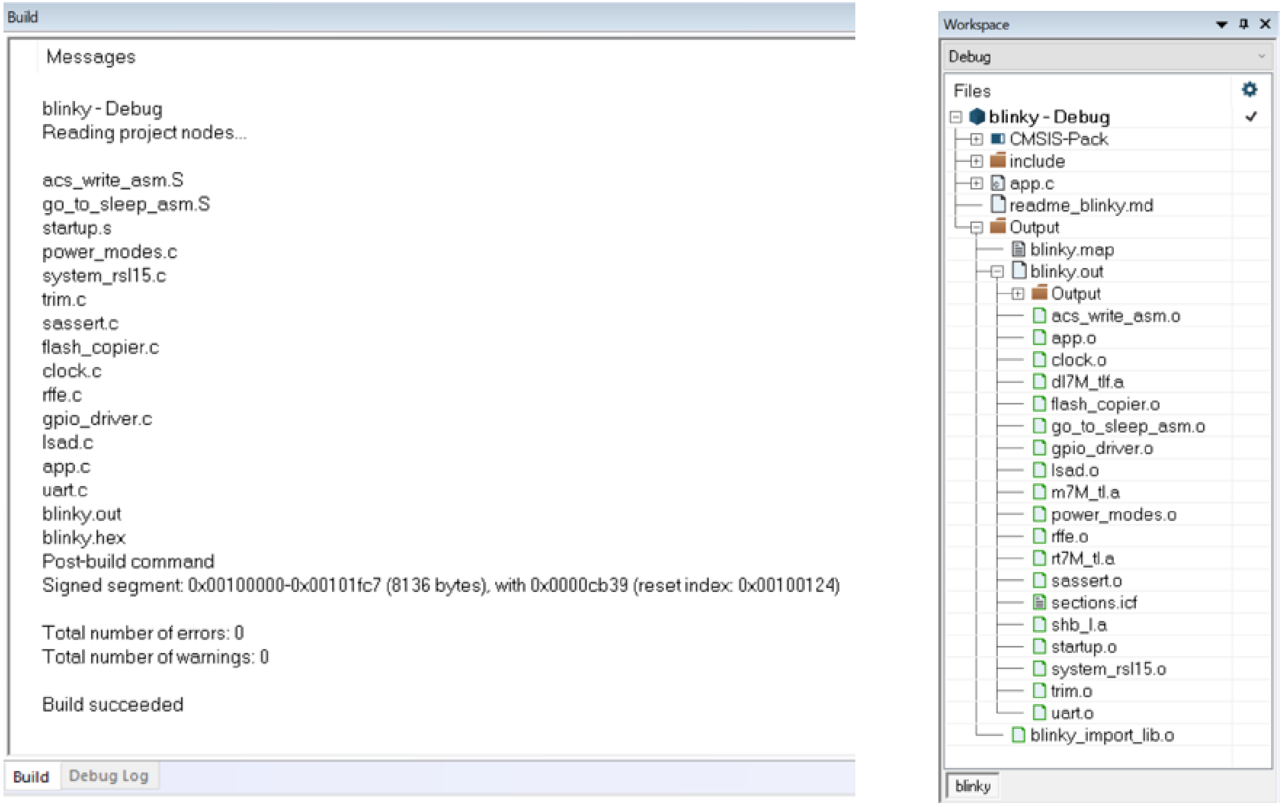

workspace - IAR Embedded Workbench IDE - Arm 9.20.2
File Edit View Project J-Link Tools Window Help
Workspace
Debug
Files
blinky - Debug
Make (F7)
Make the active project (build files
needed)

Figure 39. 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 40).



```
blinky - Debug
Reading project nodes...

acs_write_asm.S
go_to_sleep_asm.S
startup.s
power_modes.c
system_rsl15.c
trim.c
sassert.c
flash_copier.c
clock.c
rffe.c
gpio_driver.c
lsad.c
app.c
uart.c
blinky.out
blinky.hex
Post-build command
Signed segment: 0x00100000-0x00101fc7 (8136 bytes), with 0x0000cb39 (reset index: 0x00100124)

Total number of errors: 0
Total number of warnings: 0

Build succeeded
```

Figure 40. 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 USB-C 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).

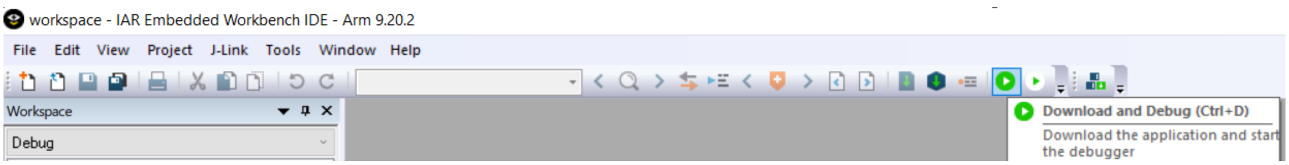
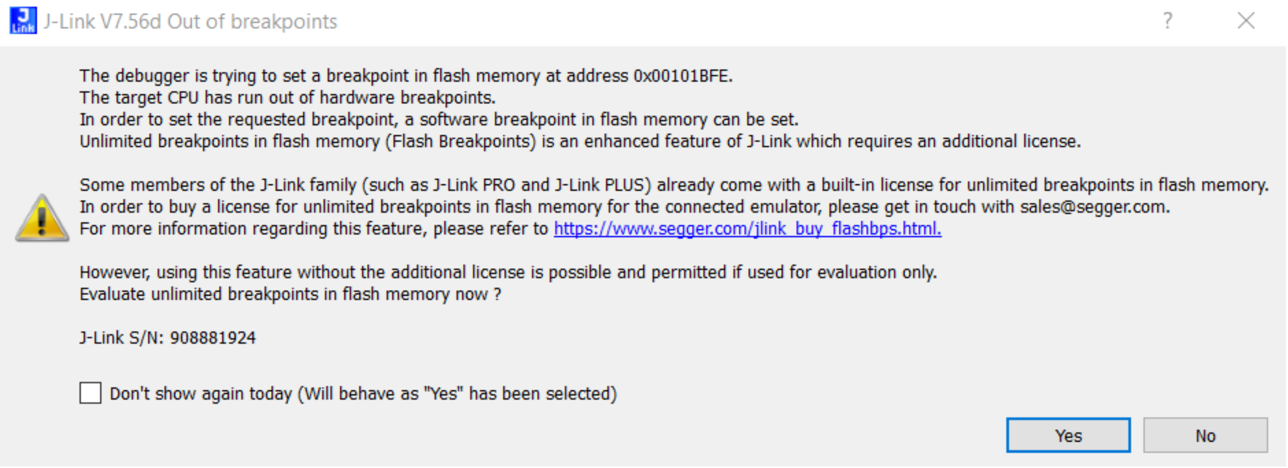

workspace - IAR Embedded Workbench IDE - Arm 9.20.2
File
Edit
View
Project
J-Link
Tools
Window
Help
Workspace
Debug
Download and Debug (Ctrl+D)
Download the application and start
the debugger

Figure 41. Start/Stop Debug Session Icon



The debugger is trying to set a breakpoint in flash memory at address 0x00101BFE.
The target CPU has run out of hardware breakpoints.
In order to set the requested breakpoint, a software breakpoint in flash memory can be set.
Unlimited breakpoints in flash memory (Flash Breakpoints) is an enhanced feature of J-Link which requires an additional license.

Some members of the J-Link family (such as J-Link PRO and J-Link PLUS) already come with a built-in license for unlimited breakpoints in flash memory.
In order to buy a license for unlimited breakpoints in flash memory for the connected emulator, please get in touch with sales@segger.com.
For more information regarding this feature, please refer to https://www.segger.com/jlink_buy_flashbps.html.

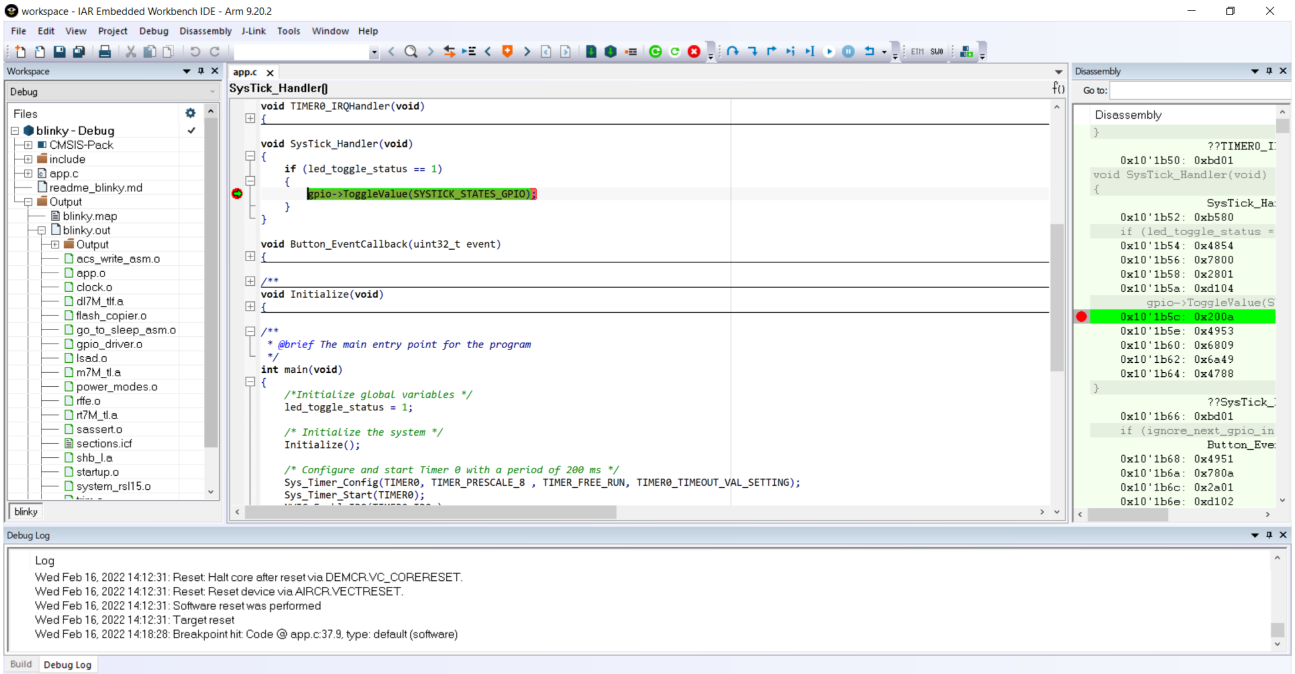
However, using this feature without the additional license is possible and permitted if used for evaluation only.
Evaluate unlimited breakpoints in flash memory now ?

J-Link S/N: 908881924

☐ Don't show again today (Will behave as "Yes" has been selected)

Figure 42. 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.



```
void TIMER0_IRQHandler(void)
{

void SysTick_Handler(void)
{
    if (led_toggle_status == 1)
    {
        gpio->ToggleValue(SYSTICK_STATES_GPIO);
    }
}

void Button_EventCallback(uint32_t event)
{

/**
void Initialize(void)
{

/**
 * @brief The main entry point for the program
 */
int main(void)
{
    /*Initialize global variables */
    led_toggle_status = 1;

    /* Initialize the system */
    Initialize();

    /* Configure and start Timer 0 with a period of 200 ms */
    Sys_Timer_Config(TIMER0, TIMER_PRESCALE_8 , TIMER_FREE_RUN, TIMER0_TIMEOUT_VAL_SETTING);
    Sys_Timer_Start(TIMER0);
```

```
Disassembly
}
                    ??TIMER0_I
    0x10'1b50: 0xbd01
void SysTick_Handler(void)
{
                    SysTick_Ha
    0x10'1b52: 0xb580
    if (led_toggle_status =
    0x10'1b54: 0x4854
    0x10'1b56: 0x7800
    0x10'1b58: 0x2801
    0x10'1b5a: 0xd104
        gpio->ToggleValue(S
    0x10'1b5c: 0x200a
    0x10'1b5e: 0x4953
    0x10'1b60: 0x6809
    0x10'1b62: 0x6a49
    0x10'1b64: 0x4788
}
                    ??SysTick_
    0x10'1b66: 0xbd01
    if (ignore_next_gpio_in
                    Button_Eve
    0x10'1b68: 0x4951
    0x10'1b6a: 0x780a
    0x10'1b6c: 0x2a01
    0x10'1b6e: 0xd102
```

```
Log
Wed Feb 16, 2022 14:12:31: Reset: Halt core after reset via DEMCR.VC_CORERESET.
Wed Feb 16, 2022 14:12:31: Reset: Reset device via AIRCR.VECTRESET.
Wed Feb 16, 2022 14:12:31: Software reset was performed
Wed Feb 16, 2022 14:12:31: Target reset
Wed Feb 16, 2022 14:18:28: Breakpoint hit: Code @ app.c:37.9, type: default (software)
```

Figure 43. Debug Session in the IAR Embedded Workbench

CHAPTER 7

Resolving External CMSIS-Pack Dependencies

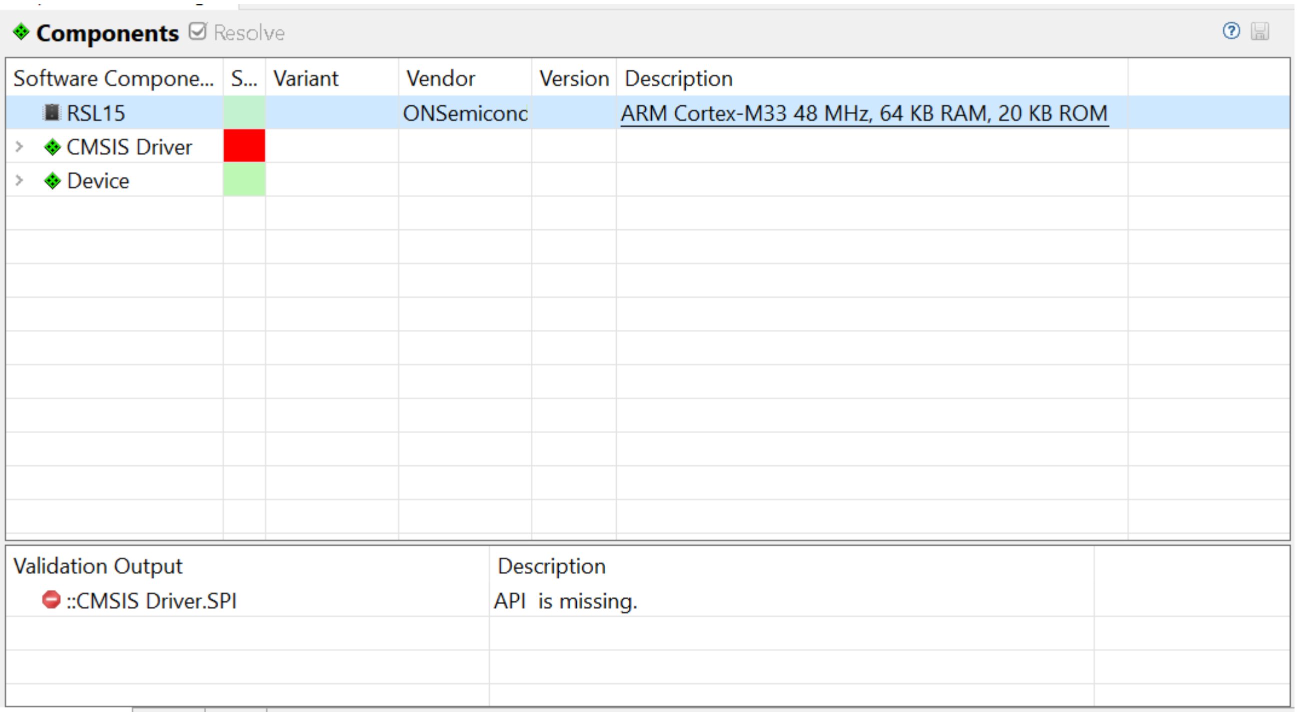
This topic explains CMSIS-Pack dependencies, and how to resolve them so that "component is missing" errors are avoided when importing sample code from the RSL15 CMSIS-Pack into the workspace.

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.



Software Compone...	S...	Variant	Vendor	Version	Description
RSL15	🟩		ONSemiconc		ARM Cortex-M33 48 MHz, 64 KB RAM, 20 KB ROM
CMSIS Driver	🟥				
Device	🟩				

Validation Output	Description
::CMSIS Driver.SPI	API is missing.

Figure 44. RTE Configuration Perspective Before Resolving Pack Dependencies

The "RTE Configuration Perspective Before Resolving Pack Dependencies" figure (Figure 44) 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 45)).

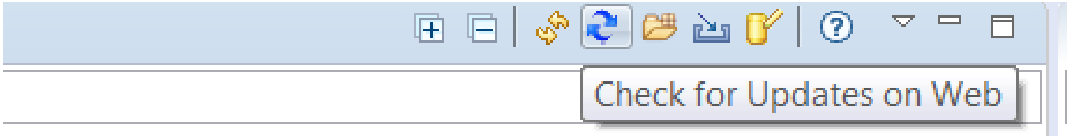
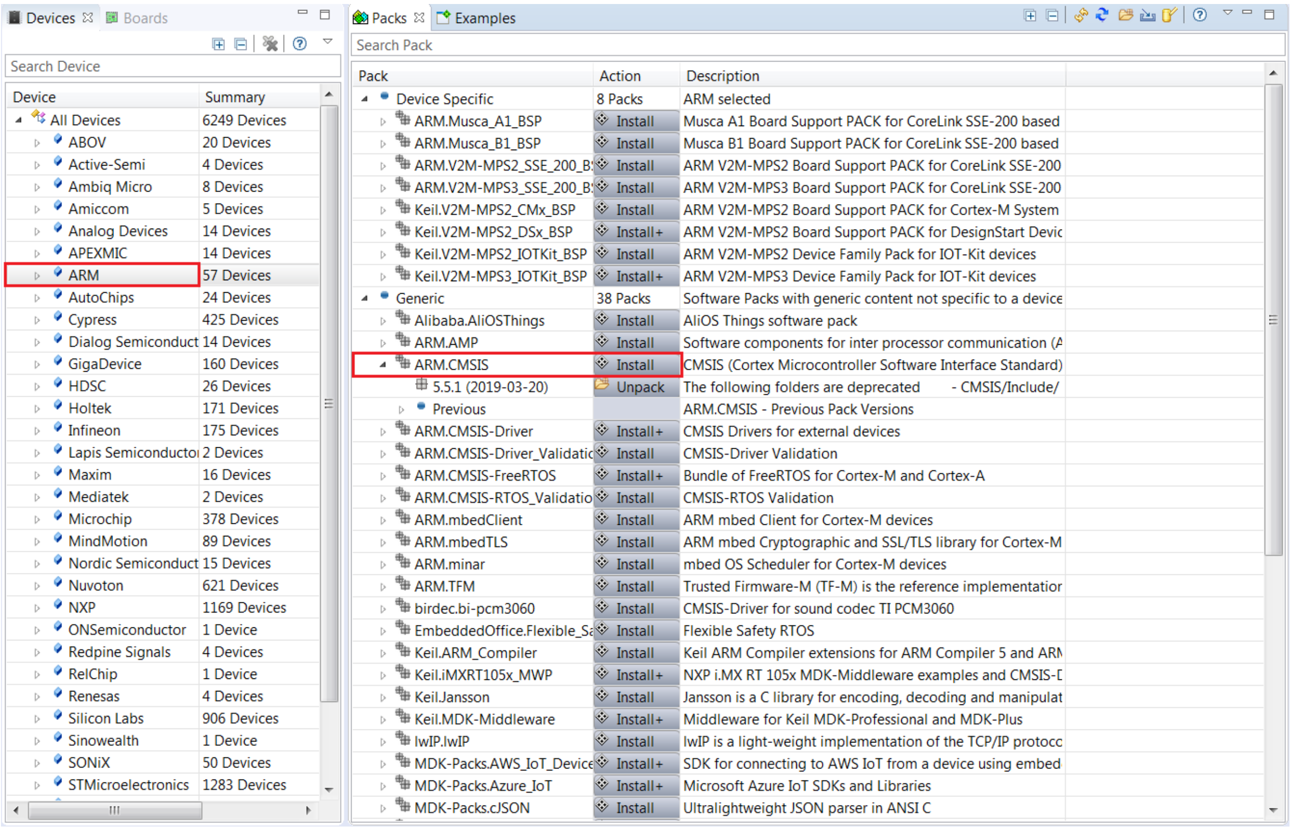


Figure 45. Check for Updates on Web Button

The "Installing the Arm CMSIS-Pack" figure (Figure 46) shows an example of the Packs tab after checking for updates.

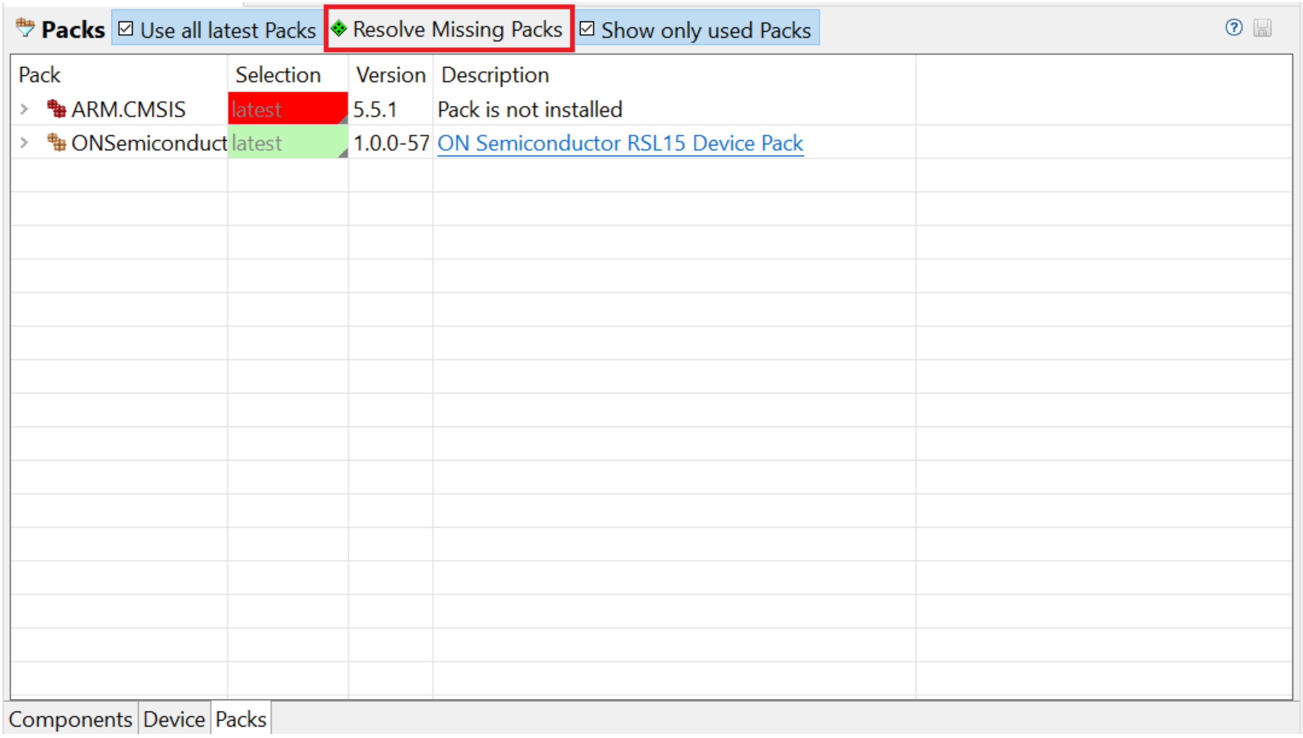


Device	Summary
All Devices	6249 Devices
ABOV	20 Devices
Active-Semi	4 Devices
Ambiq Micro	8 Devices
Amiccom	5 Devices
Analog Devices	14 Devices
APEXMIC	14 Devices
ARM	57 Devices
AutoChips	24 Devices
Cypress	425 Devices
Dialog Semiconduct	14 Devices
GigaDevice	160 Devices
HDSC	26 Devices
Holtek	171 Devices
Infineon	175 Devices
Lapis Semiconducto	2 Devices
Maxim	16 Devices
Mediatek	2 Devices
Microchip	378 Devices
MindMotion	89 Devices
Nordic Semiconduct	15 Devices
Nuvoton	621 Devices
NXP	1169 Devices
ONSemiconductor	1 Device
Redpine Signals	4 Devices
RelChip	1 Device
Renesas	4 Devices
Silicon Labs	906 Devices
Sinowealth	1 Device
SONiX	50 Devices
STMicroelectronics	1283 Devices

Pack	Action	Description
Device Specific	8 Packs	ARM selected
ARM.Musca_A1_BSP	Install	Musca A1 Board Support PACK for CoreLink SSE-200 based
ARM.Musca_B1_BSP	Install	Musca B1 Board Support PACK for CoreLink SSE-200 based
ARM.V2M-MPS2_SSE_200_B	Install	ARM V2M-MPS2 Board Support PACK for CoreLink SSE-200
ARM.V2M-MPS3_SSE_200_B	Install	ARM V2M-MPS3 Board Support PACK for CoreLink SSE-200
Keil.V2M-MPS2_CMx_BSP	Install	ARM V2M-MPS2 Board Support PACK for Cortex-M System
Keil.V2M-MPS2_DSx_BSP	Install+	ARM V2M-MPS2 Board Support PACK for DesignStart Devic
Keil.V2M-MPS2_IOTKit_BSP	Install	ARM V2M-MPS2 Device Family Pack for IOT-Kit devices
Keil.V2M-MPS3_IOTKit_BSP	Install+	ARM V2M-MPS3 Device Family Pack for IOT-Kit devices
Generic	38 Packs	Software Packs with generic content not specific to a device
Alibaba.AliOSThings	Install	AliOS Things software pack
ARM.AMP	Install	Software components for inter processor communication (A
ARM.CMSIS	Install	CMSIS (Cortex Microcontroller Software Interface Standard)
5.5.1 (2019-03-20)	Unpack	The following folders are deprecated - CMSIS/Include/
Previous		ARM.CMSIS - Previous Pack Versions
ARM.CMSIS-Driver	Install+	CMSIS Drivers for external devices
ARM.CMSIS-Driver_Validatic	Install	CMSIS-Driver Validation
ARM.CMSIS-FreeRTOS	Install+	Bundle of FreeRTOS for Cortex-M and Cortex-A
ARM.CMSIS-RTOS_Validatio	Install	CMSIS-RTOS Validation
ARM.mbedClient	Install	ARM mbed Client for Cortex-M devices
ARM.mbedTLS	Install	ARM mbed Cryptographic and SSL/TLS library for Cortex-M
ARM.minar	Install	mbed OS Scheduler for Cortex-M devices
ARM.TFM	Install	Trusted Firmware-M (TF-M) is the reference implementatior
birdec.bi-pcm3060	Install	CMSIS-Driver for sound codec TI PCM3060
EmbeddedOffice.Flexible_Sa	Install	Flexible Safety RTOS
Keil.ARM_Compiler	Install	Keil ARM Compiler extensions for ARM Compiler 5 and ARN
Keil.iMXRT105x_MWP	Install+	NXP i.MX RT 105x MDK-Middleware examples and CMSIS-D
Keil.Jansson	Install	Jansson is a C library for encoding, decoding and manipulat
Keil.MDK-Middleware	Install+	Middleware for Keil MDK-Professional and MDK-Plus
lwIP.lwIP	Install	lwIP is a light-weight implementation of the TCP/IP protocc
MDK-Packs.AWS_IoT_Device	Install+	SDK for connecting to AWS IoT from a device using embed
MDK-Packs.Azure_IoT	Install+	Microsoft Azure IoT SDKs and Libraries
MDK-Packs.cJSON	Install	Ultralightweight JSON parser in ANSI C

Figure 46. Installing the Arm CMSIS-Pack

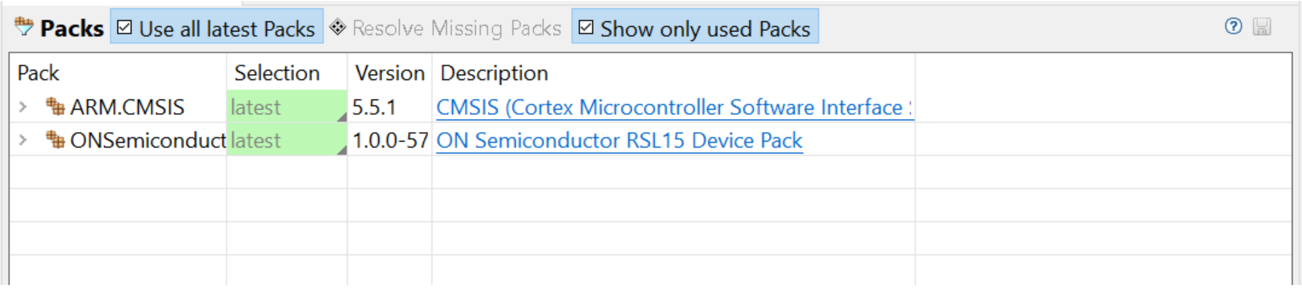
2. To manually install a CMSIS-Pack, select the **Packs** tab and search for the required CMSIS-Pack (in this example, we installed the **ARM.CMSIS** pack); click the **Install** button (shown in the "Installing the Arm CMSIS-Pack" figure (Figure 46)). Alternatively, follow the next steps to automatically 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 47)).



Pack	Selection	Version	Description
ARM.CMSIS	latest	5.5.1	Pack is not installed
ONSemiconduct	latest	1.0.0-57	ON Semiconductor RSL15 Device Pack

Figure 47. 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 48) illustrates the RTE configuration after resolving missing Packs.



Pack	Selection	Version	Description
ARM.CMSIS	latest	5.5.1	CMSIS (Cortex Microcontroller Software Interface
ONSemiconduct	latest	1.0.0-57	ON Semiconductor RSL15 Device Pack

Figure 48. 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 49)).

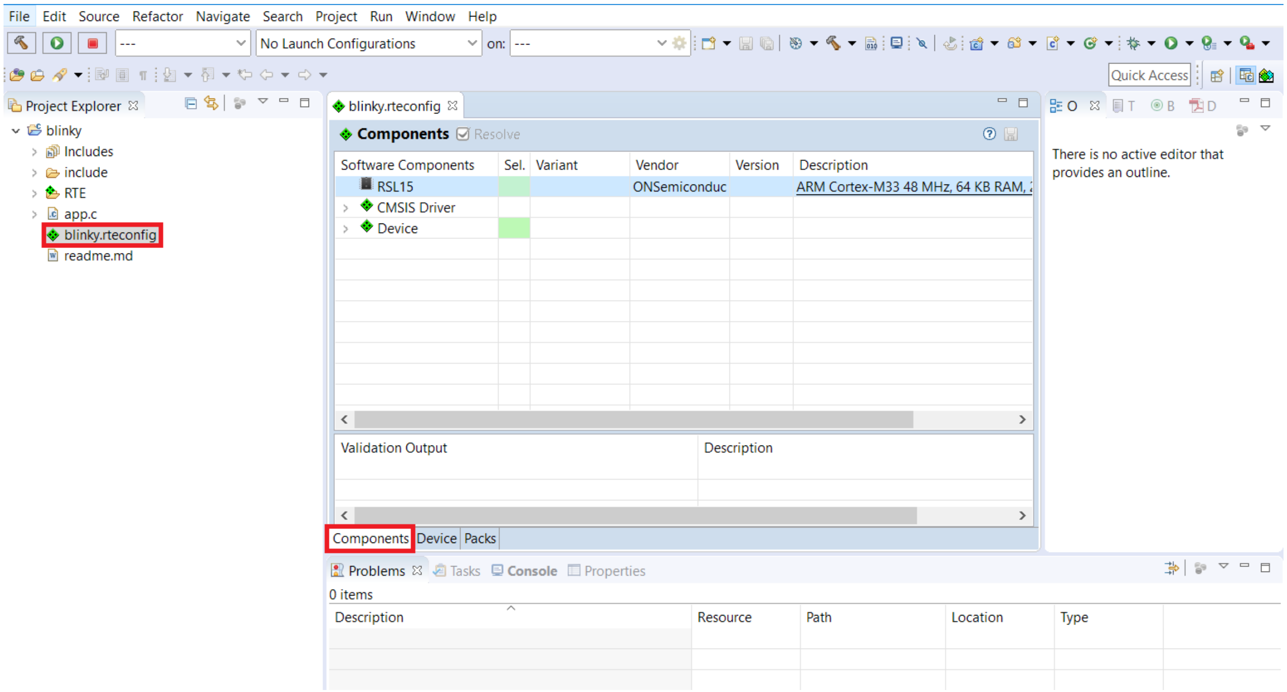
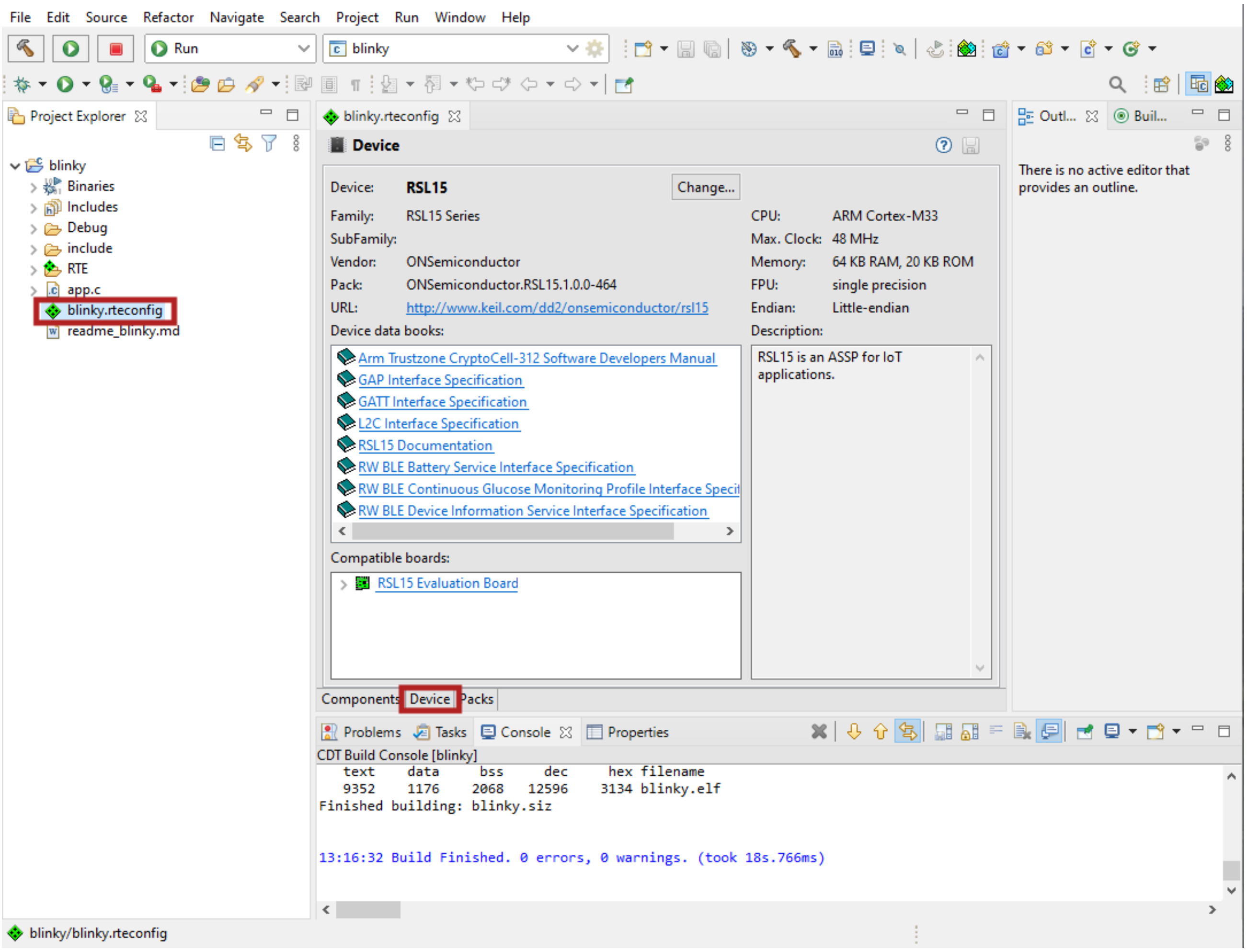


Figure 49. Opening the .rteconfig file in the onsemi IDE

2. Select the **Device** tab (see the "Accessing Documentation with the Device Tab" figure (Figure 50)).



Device: RSL15

Family: RSL15 Series

SubFamily:

Vendor: ONSemiconductor

Pack: ONSemiconductor.RSL15.1.0.0-464

URL: http://www.keil.com/dd2/onsemiconductor/rsl15

CPU: ARM Cortex-M33

Max. Clock: 48 MHz

Memory: 64 KB RAM, 20 KB ROM

FPU: single precision

Endian: Little-endian

Device data books:

- Arm Trustzone CryptoCell-312 Software Developers Manual
- GAP Interface Specification
- GATT Interface Specification
- L2C Interface Specification
- RSL15 Documentation
- RW BLE Battery Service Interface Specification
- RW BLE Continuous Glucose Monitoring Profile Interface Specit
- RW BLE Device Information Service Interface Specification

Description:

RSL15 is an ASSP for IoT applications.

Compatible boards:

- RSL15 Evaluation Board

There is no active editor that provides an outline.

CDT Build Console [blinky]

```
   text    data     bss     dec     hex filename
   9352    1176    2068   12596    3134 blinky.elf
Finished building: blinky.siz

13:16:32 Build Finished. 0 errors, 0 warnings. (took 18s.766ms)
```

Figure 50. 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 51)).

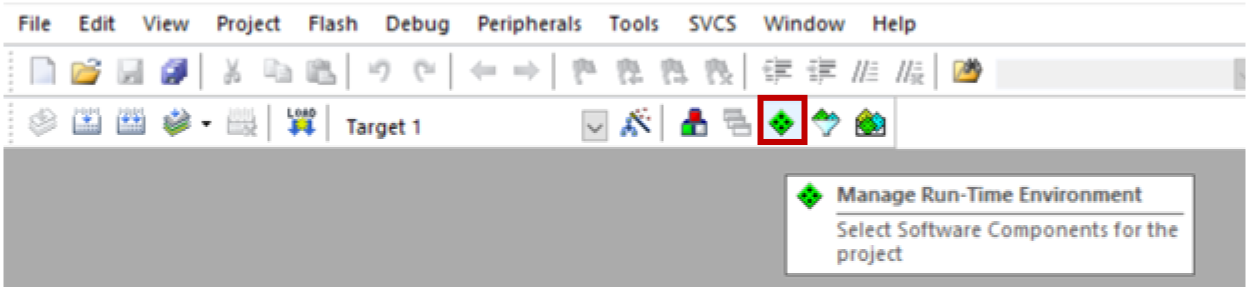
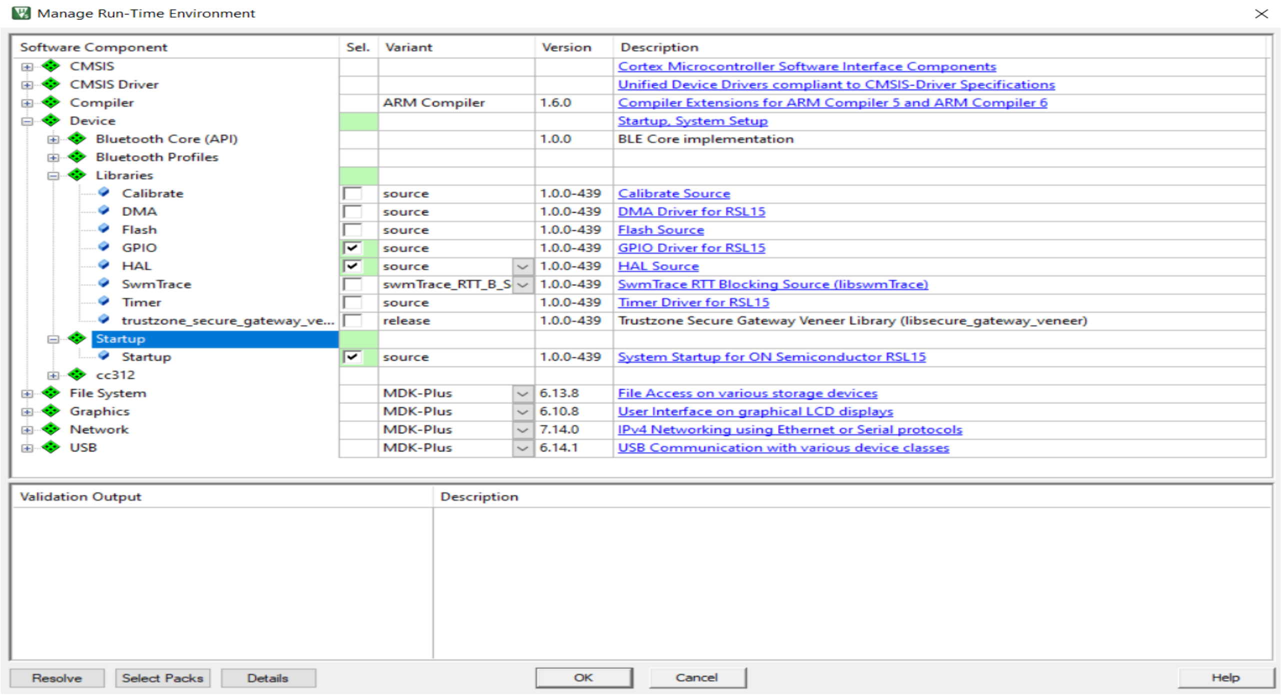

File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
Target 1
Manage Run-Time Environment
Select Software Components for the project

Figure 51. 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 52)).

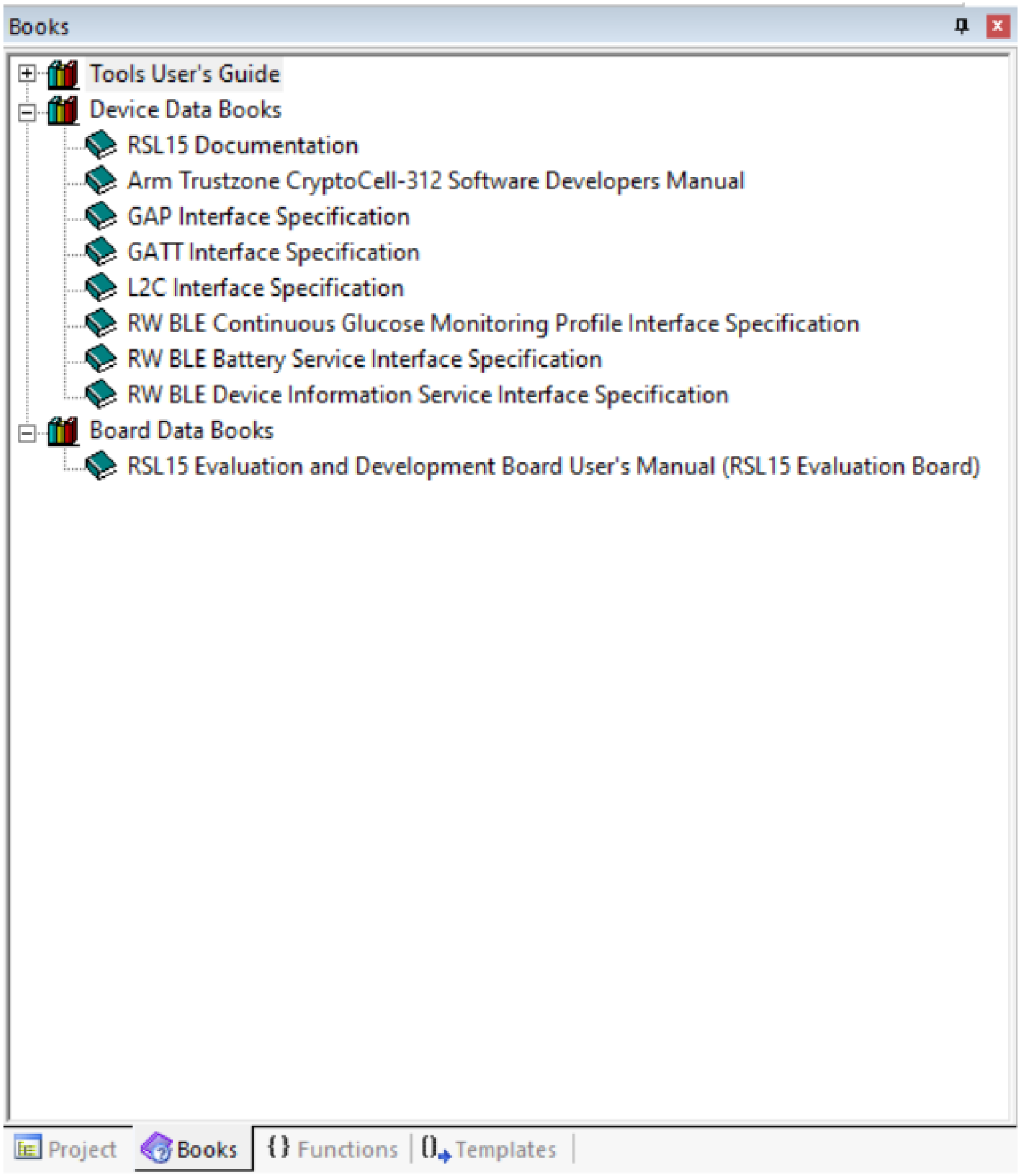


Software Component	Sel.	Variant	Version	Description
CMSIS				Cortex Microcontroller Software Interface Components
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler		ARM Compiler	1.6.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
Bluetooth Core (API)			1.0.0	BLE Core implementation
Bluetooth Profiles				
Libraries				
Calibrate	☐	source	1.0.0-439	Calibrate Source
DMA	☐	source	1.0.0-439	DMA Driver for RSL15
Flash	☐	source	1.0.0-439	Flash Source
GPIO	☑	source	1.0.0-439	GPIO Driver for RSL15
HAL	☑	source	1.0.0-439	HAL Source
SwmTrace	☐	swmTrace_RTT_B_S	1.0.0-439	SwmTrace RTT Blocking Source (libswmTrace)
Timer	☐	source	1.0.0-439	Timer Driver for RSL15
trustzone_secure_gateway_ve...	☐	release	1.0.0-439	Trustzone Secure Gateway Veneer Library (libsecure_gateway_veneer)
Startup				
Startup	☑	source	1.0.0-439	System Startup for ON Semiconductor RSL15
cc312				
File System		MDK-Plus	6.13.8	File Access on various storage devices
Graphics		MDK-Plus	6.10.8	User Interface on graphical LCD displays
Network		MDK-Plus	7.14.0	IPv4 Networking using Ethernet or Serial protocols
USB		MDK-Plus	6.14.1	USB Communication with various device classes

Validation Output	Description

Figure 52. 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 53)).



- Tools User's Guide
- Device Data Books
  - RSL15 Documentation
  - Arm Trustzone CryptoCell-312 Software Developers Manual
  - GAP Interface Specification
  - GATT Interface Specification
  - L2C Interface Specification
  - RW BLE Continuous Glucose Monitoring Profile Interface Specification
  - RW BLE Battery Service Interface Specification
  - RW BLE Device Information Service Interface Specification
- Board Data Books
  - RSL15 Evaluation and Development Board User's Manual (RSL15 Evaluation Board)

Figure 53. Keil Books Window Showing Documentation

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

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