Getting Started with RSL10

AND9697/D June 2019, Rev. P5



© SCILLC, 2019 Previous Edition © 2018 "All Rights Reserved"

Table of Contents

								Page
1. Introduction								
1.1 Overview								
1.2 Intended Audience								
1.3 Conventions	•		•	•		•		. 4
2. Setting Up the Hardware and Software								
2.1 Prerequisite Hardware				•				. 5
2.2 Connecting the Hardware								. 5
2.2.1 Preloaded Sample	•			•				. 6
2.3 Prerequisite Software								. 6
3. Getting Started with the Eclipse-Based ON Semiconductor IDE								. 7
3.1 ON Semiconductor IDE and RSL10 CMSIS-Pack Installation Procedures								
3.2 Building Your First Sample Application with the ON Semiconductor IDE								. 9
3.2.1 Launching the ON Semiconductor IDE								. 9
3.2.2 Importing the Sample Code								. 9
3.2.3 Build the Sample Code.								
3.3 Debugging the Sample Code								
3.3.1 Debugging with the .elf File								
3.3.2 Peripheral Registers View with the ON Semiconductor IDE								
4. Getting Started with Keil								
4.1 Prerequisite Software								
4.2 RSL10 CMSIS-Pack Installation Procedure								
4.3 Building Your First Sample Application with the Keil uVision IDE								
4.3.1 Import the Sample Code								
4.3.2 Build the Sample Code.								
4.3.3 Debugging the Sample Code								
4.3.3.1 Preparing J-Link for Debugging								
4.3.3.2 Debugging Applications	•			•				23
5. Getting Started with IAR								.25
5.1 Prerequisite Software								.25
5.2 RSL10 CMSIS-Pack Installation Procedure								
5.3 Building Your First Sample Application with the IAR Embedded Workber								
5.3.1 Import the Sample Code								26
5.3.2 Building the Sample Code								28
5.3.3 Debugging the Sample Code								29
5.3.3.1 Debugging Applications								29
6. Advanced Debugging								.32
6.1 Printf Debug Capabilities								.32
6.1.1 Adding Printf Debug Capabilities								.32
6.2 Debugging Applications that Do Not Start at the Base Address of Flash								.32
6.3 Arm Cortex-M3 Core Breakpoints 6.4 Debugging with Low Power Sleep Mode 6.4 Debugging with Low Power Sleep Mode	•	• •	•	•	• •	•	•	.33 .33
								.33 40
6.4.1 Downloading Firmware in Sleep Mode	•	• •	·	·	• •	•	·	40

7. More Information
7.1 Folder Structure of the RSL10 CMSIS-Pack Installation
7.2 Documentation
7.2.1 Documentation Included with the CMSIS-Pack
7.2.2 Documentation in the documentation.zip File
A. Migrating to CMSIS-Pack
A.1 Migrating an Existing Eclipse Project to the CMSIS-Pack Method
A.2 Using the Latest RSL10 Firmware in a Previous Version of the Eclipse-Based IDE
B. Arm Toolchain Support
B.1 Basic Installation
B.2 Configuring the Arm Toolchain in the ON Semiconductor IDE
B.3 Additional Settings

CHAPTER 1

Introduction

1.1 OVERVIEW

RSL10 is a multi-protocol, Bluetooth[®] 5 certified, radio System on Chip (SoC), with the lowest power consumption in the industry. It is designed to be used in devices that require high performance and advanced wireless features, with minimal system size and maximized battery life. The RSL10 Software Development Kit (SDK) includes firmware, software, example projects, documentation, and development tools. The Eclipse-based ON Semiconductor Integrated Development Environment (IDE) is offered as a free download with optional support for Arm[®] Keil[®] μ Vision[®] and IAR Embedded Workbench[®].

Software components, device and board support information are delivered using the CMSIS-Pack standard. Standard CMSIS-Drivers for peripheral interfaces and FreeRTOS sample applications are supported. With the CMSIS-Pack standard, you can easily go beyond what is included in our software package and have access to a variety of generic Cortex-M software components. If you have existing RSL10 projects and have not used the RSL10 CMSIS-Pack before, see Appendix A, "Migrating to CMSIS-Pack" on page 48 for more information.

The RSL10 SDK allows for rapid development of ultra-low power Bluetooth Low Energy applications. Convenient abstraction decouples user application code from system code, allowing for simple modular code design. Features such as FOTA (Firmware Over-the-Air) can easily be added to any application. Advanced debugging features such as support for SEGGER® RTT help developers monitor and debug code. Sample applications, from Blinky to ble_peripheral_server_bond and everything in between, help get software development moving quickly. An optional Bluetooth mesh networking CMSIS-Pack quickly enables mesh networking for any application.

This document helps you to get started with the RSL10 SDK. It guides you through the process of connecting your RSL10 Evaluation and Development Board, installing an IDE and the CMSIS-Pack, configuring your environment, and building and debugging your first RSL10 application.

NOTE: RSL10 contains a low power DSP processor core; see *RSL10 LPDSP32 Software Package.zip* for more information.

1.2 INTENDED AUDIENCE

This manual is for people who intend to develop applications for RSL10. It assumes that you are familiar with software development activities.

1.3 CONVENTIONS

The following conventions are used in this manual to signify particular types of information:

monospace	Commands and their options, file and path names, error messages, code samples and code snippets.
mono bold	A placeholder for the specified information. For example, replace filename with the actual name of the file.
bold	Graphical user interface labels, such as those for menus, menu items and buttons.
italics	File names and path names, or any portion of them.

CHAPTER 2

Setting Up the Hardware and Software

2.1 Prerequisite Hardware

The following items are needed before you can make connections:

- RSL10 Evaluation and Development Board and a micro USB cable
- A computer running Windows

2.2 CONNECTING THE HARDWARE

To connect the Evaluation and Development Board to a computer:

1. Check the jumper positions:

Ensure that the jumper CURRENT is connected and POWER OPTIONS is selected for USB. Also, connect the jumpers TMS, TCK and SWD. Finally, connect the headers P7, P8, P9 and P10 to 3.3 V, as highlighted in Figure 1.

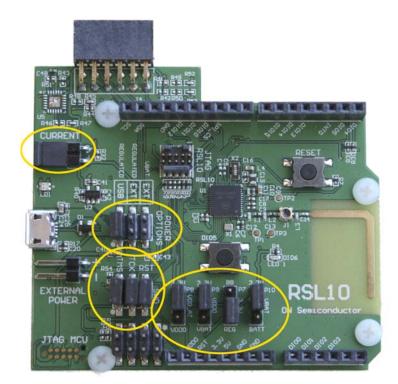


Figure 1. Evaluation and Development Board with Pins and Jumpers for Connection Highlighted

2. Once the jumpers are in the right positions, you can plug the micro USB cable into the socket on the board. The LED close to the USB connector flashes green during the first time plugging in, then turns a steady green once the process is finished.

2.2.1 Preloaded Sample

The Evaluation and Development Boards come with one of the following preloaded sample applications:

- "Peripheral Device with Sleep Mode" is on boards with a serial number lower than 1741xxxxx.
- "Peripheral Device with Server" is on boards with a serial number higher than 1741xxxxx.

For more information about sample applications, refer to the RSL10 Sample Code User's Guide.

2.3 PREREQUISITE SOFTWARE

Install the latest version of J-Link. It is available from the <u>SEGGER website</u>. Make sure to run the J-Link DLL updater, as shown in Figure 2, to update the J-Link DLL inside your IDE and confirm that the version used by the IDE has been updated.

R SEGGER J-Link DLL Updater V6.34h		×
2 applications found that can be updated to V6.34h of the J-Link software:		
✓ SEGGER Embedded Studio ARM 3.20 (x86) (DLL V6.40 in "C\Program Files (x86)\SEGGER\ ✓ Keil MDK-ARM (DLL V6.40 in "C\Keil_v5\ARM\Segger")	SEGGER Em	bedded Studio :
Select All Select None		
Select the ones you would like to replace by this version. The previous version will be renamed and kept in the same folder, allowing manual "undo".		
la construit de la seconte de contrata en la factoria (n. 1977).	Ok	Cancel

Figure 2. J-Link DLL Updater

CHAPTER 3

Getting Started with the Eclipse-Based ON Semiconductor IDE

3.1 ON SEMICONDUCTOR IDE AND RSL10 CMSIS-PACK INSTALLATION PROCEDURES

If you have a previous version of the ON Semiconductor IDE (formerly known as the RSL10 Software Development Kit (SDK)) installed:

- 1. Uninstall RSL10 Development Tools using Windows Control Panel.
- 2. Check if the RSL10 SDK folder is still there; if it is, delete it.

Install your new ON Semiconductor IDE by running *ON_Semiconductor_IDE.msi*. The ON Semiconductor IDE is installed in this location by default: *C:\Program Files (x86)\ON Semiconductor\IDE*.

The release version and build number are stored in the *REVISION* text file at the root of the installed ON Semiconductor IDE.

To install the RSL10 CMSIS-Pack:

- 1. Download the RSL10 CMSIS-Pack from <u>www.onsemi.com/RSL10</u> and save it in any temporary folder.
- 2. Open the ON Semiconductor IDE and choose the desired location for your new workspace for example, *c:\workspace* — and click **OK**.
- 3. On the top right corner, click on the **Open Perspective** icon, select **CMSIS Pack Manager**, and click **OK** (see Figure 3).

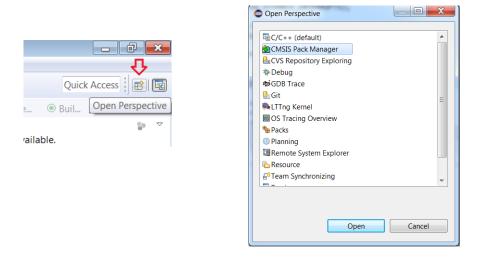


Figure 3. Opening the CMSIS Pack Manager Perspective

4. Click on the **Import existing packs** icon, select your pack file *ONSemiconductor.RSL10.version.pack*, where **version** is a number such as 2.3.27, and click **Open** (see Figure 4).

	Import Packs	Search Park	X
	Computer + S	YSTEM (C:) ► cmsis_packs ►	Search cmsis_packs
	Organize 🔻 New folder		II • 🔟 📀
	E Desktop	Name	Date modified
	Downloads Recent Places	.Download	3/2/2018 10:12 AN
🗘 Quick Access 🛛 😰 🖷 🚳 🕸	~	Web	3/2/2018 9:56 AM
🗏 🔗 🍸 🗖 📮 Pack Properties 🛛 🗖 🗖	4 🧱 Libraries	ONSemiconductor	3/2/2018 10:12 AN
	Documents	ONSemiconductor.RSL10.1.2.0.p	ack 8/29/2017 3:52 PM
mport existing packs	Music		
	Pictures		
	Videos		
	4 🌬 Computer		
	D 😂 SYSTEM (C:)	▼ <	•
	File name: OI	NSemiconductor.RSL10.1.2.0.pack	iles (*.pack) 🔻
		Op	en 👻 Cancel

Figure 4. Installing the RSL10 CMSIS-Pack

- 5. The IDE prompts you to read and accept our license agreement, and then installs the RSL10 CMSIS-Pack in the specified pack root folder.
- 6. The RSL10 CMSIS-Pack now appears in the list of installed packs. In the Devices tab, if you expand All Devices > ONSemiconductor > RSL10 Series you can see RSL10 listed there. You can manage your installed packs in the Packs tab. Expanding ONSemiconductor.RSL10 makes the Pack Properties tab display the details of the RSL10 CMSIS-Pack. Figure 5 on page 9 illustrates what the Pack Manager perspective looks like after installation.

	r Navigate Search Project Run Window H					Quick Access 🔡 🗟 🗱
Devices 22 Boards		A Packs 22 C Examples		⊞ ⊟ ⑧ ≷ ≝ ∻ ‱ ▼ = □	Pack Properties 22	
type filter text		Search Pack			type filter text	
Device	Summary	Pack	Action	Description	A # ONSemiconduct	or.RSI.10.1.2.0
All Devices	1 Device	Device Specific	1 Pack	RSL10 selected	Boards	
 ONSemiconductor 		ONSemiconductor.RSL10		ON Semiconductor RSL10 Device F	A Components	
 RSL10 Series 	1 Device	# 1.2.0		Release 1.2.0	 Pevice 	
RSL10	ARM Cortex-M3 48 MHz, 32 kB RAM, 384 kB	Previous	- nemore	ONSemiconductor.RSL10 - Previour	🔺 🍳 Libraries	5
- 1000	Auto Contex mis 46 minz, 52 kb 10 mi, 50 kb	Generic	_	Software Packs with generic conter	Calib	rate
				sonnore racio intrigenene conten	Custo	om Protocol
					Flash	
					Syste	m
					Math	1
					Kerne	el
					BLE	
					Remo	ote_Mic
					Weal	k_PRF
					Startup	
					Bluetoo	th Profiles
					A B Devices	
					4 🏤 RSL10 Serie	es
					RSL10	
					 Examples 	
					ADC_UART	(RSL10 Evaluation Board)
					Aes128 (RS)	SL10 Evaluation Board)
					 Blinky (RSL 	.10 Evaluation Board)
					E Central_clie	ent (RSL10 Evaluation Board)
		•			< I	1
Console 😫 🤜 Progress						k 🖬 🖻 🛃 🔍 + 😁 + °
MSIS Pack Manager						
	ack ONSemiconductor.RSL10.1.2.0 comp	leted				

Figure 5. Pack Manager Perspective after RSL10 CMSIS-Pack is Installed

3.2 BUILDING YOUR FIRST SAMPLE APPLICATION WITH THE ON SEMICONDUCTOR IDE

This section guides you through importing and building your first sample application, named *blinky*. This application makes the LED (DIO6) blink on the Evaluation and Development Board.

For more information about the sample applications, see the RSL10 Sample Code User's Guide.

3.2.1 Launching the ON Semiconductor IDE

To use the IDE for the first time, follow the steps below:

- 1. To start the IDE, go to the Windows Start menu, and select ON Semiconductor > ON Semiconductor IDE.
- 2. When you open the IDE for the first time, you are prompted to select a workspace for the session. The workspace is the work area for all your IDE projects.

IMPORTANT: Create a new workspace for your version of the ON Semiconductor IDE. Re-using an existing workspace originally created with another Eclipse-based IDE might not be compatible.

3.2.2 Importing the Sample Code

Import the sample code as follows:

- 1. In the Pack Manager perspective, click on the **Examples** tab to list all the example projects included in the RSL10 CMSIS-Pack.
- 2. Choose the example project called *blinky*, and click the **Copy** button to import it into your workspace (see Figure 6 on page 10).

Frampla	Action	Description
Example		
blinky (RSL10 Evaluation Board)	Copy	Blinky GPIO I/O Sample Code
central_client (RSL10 Evaluation Board)	🈻 Сору	Central Device with Client Sample Code
central_client_double (RSL10 Evaluation	в 🕸 Сору	Central Device with Client Sample Code - Double
central_peripheral (RSL10 Evaluation Bo	oar 🧇 Copy	Central Peripheral Device Sample Code
custom_protocol_trx (RSL10 Evaluation	Вс ጳ Сору	Low Latency Audio Sample Application with Custom Prot
default_MANU_INFO_INIT (RSL10 Evaluation	ati 🕸 Copy	Default System Initialization Function
hci_app (RSL10 Evaluation Board)	🚸 Copy	Host Controller Interface Application
pair_bond (RSL10 Evaluation Board)	🚸 Сору	Pairing and Bonding with Peripheral Device Sample Code
pair_bond_master (RSL10 Evaluation Bo	ar ጳ Copy	Pairing and Bonding with Central Device Sample Code
peripheral_server (RSL10 Evaluation Boa	arc ጳ Copy	Peripheral Device with Server Sample Code
peripheral_server_FOTA (RSL10 Evaluati	or ጳ Copy	Peripheral Device with Server for Sending Firmware Over
peripheral_server_hrp (RSL10 Evaluation	n B 🇇 Copy	Heart Rate Peripheral Device with Server Sample Code
peripheral_server_sleep (RSL10 Evaluati	on ጳ Copy	Sleep Mode Sample Code for Peripheral Device with Serv
sleep_ble_advertisements (RSL10 Evalua	ati 🎨 Copy	Sleep and Wakeup with Bluetooth Low Energy Technolog
sleep_RAM_retention (RSL10 Evaluation	В 🏶 Сору	Sleep and Wakeup Sample Code
standby_power_mode (RSL10 Evaluatio	n 🐶 Copy	Standby Power Mode Sample Code
supplemental calibrate (RSL10 Evaluati	on 🄄 Copy	Supplemental Calibration Sample Code

Figure 6. 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. On the right side, the blinky.rteconfig file displays software components. If you expand Device > Libraries, you can see the System library (*libsyslib*) and the Startup (libcmsis) components selected for blinky (see Figure 7 on page 11).
- NOTE: Sample projects are preconfigured with *Release* versions of RSL10 libraries, which are distributed as object files. In the RTE configuration, you can switch to the Source variant to include the source code of the library directly into your project (see Figure 7 on page 11).

		I ▼ 9 ▼ 10 00 A ■ 10 0			1 × ⊳ 00 ■ 14		弓 第 心 動 列 × 句 ◆ ◆ ◆ ×	
Project Explorer 🕸	G Ø 1 6			22				
🛎 blinky		Components Service Report R	solve					0 🖬
 M Includes M RTE 		Software Components	Sel.	Variant	Vendor	Version	Description	
Ric Main.c		RSL10			ONSemicondu		ARM Cortex-M3 48 MHz, 32 kB RAM, 384 kB ROM	
 blinky.rteconfig 		A Device						
readme_blinky.txt		Bluetooth Profile	n -					
sections.ld		Libraries						
		♀ BLE		release	ONSemicondu	1.0.0	Bluetooth Stack (libblelib)	
· 續 - cmais (watvcs) - 節 - measure_r_cosc (watvcs) - 値 remote_mic_pr_aw - 値 remote_mic_b_raw	Calibrate		release	ONSemicondu		Calibration Library (libcalibratelib)		
	Custom Proto	c 🗆	release	ONSemicondu		Low Latency Audio Streaming Custom Protocol Library (libc		
	Flash		release	ONSemicondu		Flash Library (libflashlib)		
	Kernel		release	ONSemicondu	1.0.0	Event Kernel Library (libkelib)		
	Math		release	ONSemicondu	1.0.0	Math Library (libmathlib)		
	Remote_Mic		release	ONSemicondu	1.0.0	Remote Microphone Library (libremote_micLib)		
	System		release	ONSemicondu	1.0.0	System Macros and Library (libsyslib)		
	Weak_PRF		release	ONSemicondu	1.0.0	Weak Profile Library (weak_prf)		
		Startup		release	ONSemicondu	1.0.0	RSL10-CMSIS Startup Library and Include Folders (libcmsis)	
		Validation Output	-		Da	scription		_
		Vandation Output			De	scrption		

Figure 7. RTE Configuration for the Blinky Example Project in the ON Semiconductor IDE

3.2.3 Build 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 hammer icon shown in Figure 8.

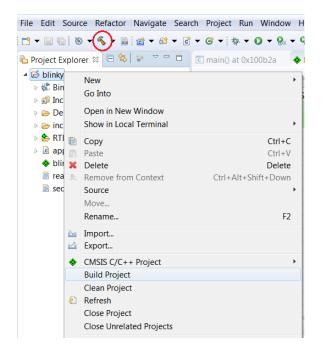


Figure 8. Starting to Build a Project in the ON Semiconductor IDE

When the build is running, the output of the build is shown in the ON Semiconductor IDE C/C++2. Development Tooling (CDT) Build Console, as illustrated in Figure 9.



Figure 9. Example of Build Output

- The key resulting output in Project Explorer includes: 3.
 - 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 Figure 10 on page 13.

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.

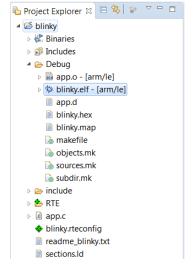


Figure 10. Output Files from Building a Sample Project

NOTE: If the ON Semiconductor IDE has trouble finding the GNU toolchain, it might be caused by having other GNU toolchains installed. See Appendix B, "Arm Toolchain Support" on page 50 for more information.

3.3 DEBUGGING THE SAMPLE CODE

3.3.1 Debugging with the .elf File

Debug the application using the .*elf* file as follows:

- 1. Within the **Project Explore**r, right-click on the *blinky.elf* file and select **Debug As > Debug Configurations...**
- 2. When the **Debug Configurations** dialog appears, right-click on **GDB SEGGER J-Link Debugging** and select **New**. A new configuration for *blinky* appears under the **GDB SEGGER** heading, with new configuration details in the right side panel.
- 3. Change to the **Debugger** tab, and enter RSL10 in the **Device** field. Ensure that **SWD** is selected as the target interface (as shown in Figure 11 on page 14).

1 🗈 🗶 😑 🐡 🕶	Name: blinky			
ype filter text	🗋 Main 🕸 Debu	gger 🔪 🗭 Startup 🧤 Source) 📼 Common		
 E C/C++ Application E C/C++ Attach to Application E C/C++ Postmortem Debugger C C/C++ Remote Application E GDB Hardware Debugging C GDB SEGGER J-Link Debugging C BDB SEGGER J-Link Debugging Launch Group 	J-Link GDB Serv Start the J-L Executable: Device name: Endianness: Connection: Interface: Initial speed: GDB port: SWO port:	nk GDB server locally Connect to running target \$(jlink, path)/\$(jlink,gdbserver}	Supported de	
	Telnet port:		Silent	
	remerpore	2555 Eccal host only	Sheric	
	1			0
	Log file:			Browse
	Other options:	-singlerun -strict -timeout 0		Browse
	Other options:	sole for the GDB server 📝 Allocate console for semihosting	and SWO	Browse
	Other options: Allocate cor GDB Client Setu	p		
	Other options: C Allocate con GDB Client Setu Executable:	sole for the GDB server 📝 Allocate console for semihosting	and SWO	
	Other options: Allocate cor GDB Client Setu	p		
	Other options: Allocate con GDB Client Setu Executable: Other options: Commands: Remote Target	sole for the GDB server Allocate console for semihosting S(cross_prefix)gdb\$(cross_suffix) set mem inaccessible-by-default off		Variables
	Other options: Cher options: CDB Client Setu Executable: Other options: Commands: Remote Target Host name or I	sole for the GDB server Allocate console for semihosting		Variables.
	Other options: Cher options: GDB Client Sett. Executable: Other options: Commands: Remote Target Host name or I Port number:	sole for the GDB server Allocate console for semihosting S(cross_prefix)gdb\$(cross_suffix) set mem inaccessible-by-default off		

Figure 11. Setting Up a GDB Launch Configuration, Debugger Tab

- NOTE: If you want to debug an application that does not start at the first address of flash memory, see Chapter 6, "Advanced Debugging" on page 32.
- 4. Once the updates to the configuration are completed, make sure the Evaluation and Development Board is connected to the PC via a micro USB cable, and click Debug. J-Link automatically downloads the blinky sample code to RSL10's flash memory.
- NOTE: If J-Link does not automatically write your program to RSL10's flash memory, make sure you are using the J-Link version specified in Section 2.3, "Prerequisite Software" on page 6.

If you are having trouble downloading firmware because an application with Sleep Mode is on the Evaluation and Development Board, see Section 6.4.1, "Downloading Firmware in Sleep Mode" on page 40.

- The ON Semiconductor IDE asks if you would like to open the Debug perspective. Answer Yes, and click on 5. Remember my decision so that the question is not asked again.
- 6. The Debug perspective opens and the application runs up to the first breakpoint in main, as shown in Figure 12 on page 15. You can press F6 multiple times to step through the code and observe that the LED changes its state when the application executes the function Sys GPIO Toggle (LED DIO).

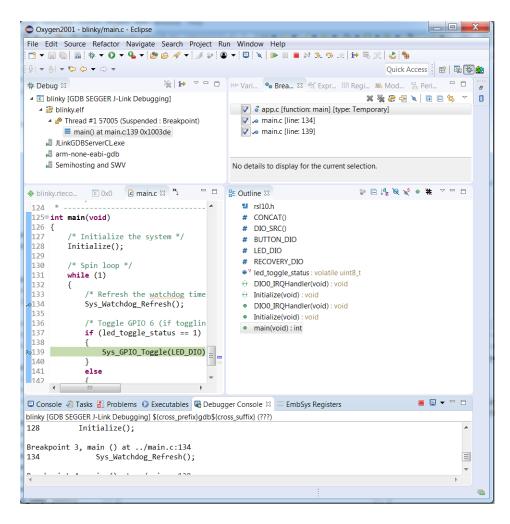


Figure 12. Debug Perspective

3.3.2 Peripheral Registers View with the ON Semiconductor IDE

The ON Semiconductor IDE includes a peripheral register view plugin that enables you to visualize and modify all of the RSL10 registers during a debug session. It can be configured by setting the path to the SVD file in the Debug session.

The following steps demonstrate how to configure and use the Peripheral Registers View with the *Blinky* application:

- 1. Right click on the *blinky.elf* file, select **Debug As** > **Debug Configurations**, and open your configuration details set, as described in Section 3.3.1, "Debugging with the .elf File" on page 13.
- Change to the SVD Path tab, and set the path to the *rsl10.svd* file as C:\Users\<user_id>\ON_Semiconductor\PACK\ONSemiconductor\RSL10\<pack_version>svd (see Figure 13). Click Debug.

Create, manage, and run configurat	ions 🔅
Image: Second Secon	Name: blinky Debug Main * Debugger * Startup * Source Common * SVD Path SVD file (used by the peripheral registers viewer) File path: C:\Users_\ON_Semiconductor\PACK\ONSemiconductor\RSL10\3.0.503\svd\rs10.svd File path: C:\User_id>
Filter matched 18 of 104 items	Revert Apply Debug Close

Figure 13. SVD Path Tab Perspective

- 3. 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 RSL10 peripherals displayed.
- 4. Select the peripheral that you need to monitor, and open the Memory window to show the RSL10 peripheral registers. The read only registers are highlighted in green. If you wish, you can drag your Memory window and place it side-by-side with your source code view (see Figure 15 on page 18).
- 5. Select DIO and CLK in the peripherals window. Now you can monitor the selected peripherals from the Monitors tab and switch between them. To see or change the **DIO** register status, choose **DIO** and expand the **DIO** > **DIO DATA** register in the Memory window.
- Press F6 to step through the code. You can observe that this register's bit 6 toggles its state when 6. Sys_GPIO_Toggle (LED_DIO) is executed. The register turns yellow to indicate that you have activated real-time monitoring for it. (see Figure 14).

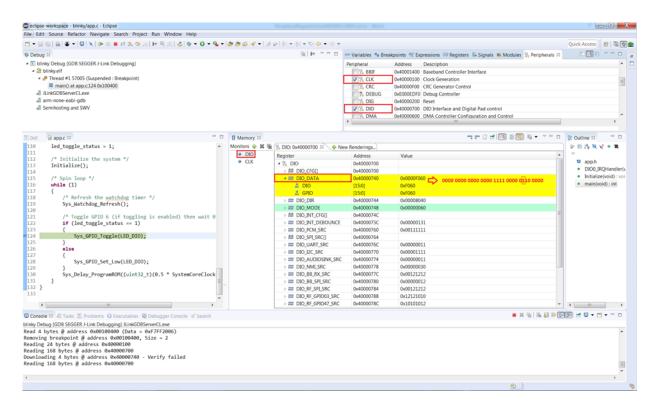


Figure 14. Peripheral Registers View Perspective in Debug Session After Setting SVD Path

7. Click on the **Value** tab of the **GPIO** register to change the (**HIGH/LOW**) state of GPIO6, as shown in Figure 15. You can observe that the LED (DIO6) on your board changes its state.

	🌋 🛃 DIO: 0x40000700 🛛 🗍 🕂 Nev	v Renderings		
DIO	Register	Address	Value	
CLK	A 🚡 DIO	0x40000700		
	▷ # DIO_CFG[]	0x40000700		
	▲ IIII DIO_DATA	0x40000740	0x0000F060	
	🔐 DIO	[15:0]	0xF060	
	SPIO .	[15:0]	0x0: GPIO0_LOW	Ψ.
	▷ 👯 DIO_DIR	0x40000744	0x0: GPIO12_LOW	•
	▷ 🚻 DIO_MODE	0x40000748	0x0: GPIO13_LOW	
	▷ INT_CFG[]	0x4000074C	0x0: GPIO14_LOW	
	▷ IIII DIO_INT_DEBOUNCE	0x4000075C	0x0: GPIO15_LOW	
	▷ 🔐 DIO_PCM_SRC	0x40000760	0x1: GPIO0_HIGH 0x2: GPIO1_HIGH	
	▷ # DIO_SPI_SRC[]	0x40000764	0x4: GPIO2 HIGH	
	▷ 🗤 DIO_UART_SRC	0x4000076C	0x8: GPIO3_HIGH	
	▷ IIII DIO_I2C_SRC	0x40000770	0x10: GPIO4_HIGH	
	▷ ## DIO_AUDIOSINK_SRC	0x40000774	0x20: GPIO5_HIGH	
	▷ 🗤 DIO_NMI_SRC	0x40000778	0x40: GPIO6_HIGH	
	▷ ## DIO BB RX SRC	0x4000077C	0x80: GPIO7_HIGH	
	▷ ## DIO BB SPI SRC	0x40000780	0x100: GPIO8_HIGH 0x200: GPIO9 HIGH	=
	▷ ## DIO_RF_SPI_SRC	0x40000784	0x400: GPIO10 HIGH	
	▷ ## DIO_RF_GPIO03_SRC	0x40000788	0x800: GPIO11 HIGH	
	▷ ## DIO RF GPIO47 SRC	0x4000078C	0x1000: GPIO12_HIGH	
	▷ ### DIO RF GPIO89 SRC	0x40000790	0x2000: GPIO13_HIGH	

Figure 15. Toggling RSL10 DIO Using the Peripheral Registers View

CHAPTER 4

Getting Started with Keil

4.1 PREREQUISITE SOFTWARE

Download and install the Keil µVision IDE from the Keil website, using the vendor's instructions.

4.2 RSL10 CMSIS-PACK INSTALLATION PROCEDURE

To install the RSL10 CMSIS-Pack:

1. Open the Keil µVision IDE and navigate to **Project > Manage > Pack** installer or click on the icon shown in Figure 16.

File	Edit	View	Project	Flash	Debug	Peripherals	Tools	SVCS	Wine
	🞽 🖬		3 h 🕻	9	• ا	⇒ 🧖 🖄	内内	*	۰ ₽
٢) 🗳 🗸		Targ	et 1	- 🔊	📥 🗟	6 🔶 👌	› 💑

Figure 16. Pack Installer Icon

2. Click on File > Import, select your pack file ONSemiconductor.RSL10.version.pack, and click Open (see Figure 17). version is the RSL10 version, such as 2.2.347.

	🛃 Import Packs
	Good and the second
	Organize 🔻 New folder 🔠 💌 🗍 😧
	Favorites Ame Date modified Type
	Desktop ONSemiconductor.RSL10.2.2.347 10/31/2018 4:29 PM uVis
Pack Installer - C:\Keil_v5\ARM\PACK File Packs Window Help Refresh Import Import from Folder Mapage Local Dependitories Pack	Downloads Recent Places Libraries Documents Music
Manage Local Repositories Summary Pack Exit Summary Devices T = 2 All Devices 5734 Devices #	S Pictures Videos Computer
	📲 SYSTEM (C:) 🗸 🖌 🔢
	File name: ONSemiconductor.RSL10.2.2.347 Software Pack - PACK (*.zip; •) Open • Cancel

Figure 17. Installing the RSL10 CMSIS-Pack for the Keil $\mu\text{Vision IDE}$

- 3. The IDE prompts you to read and accept our license agreement, then installs the RSL10 CMSIS-Pack in the C:\Keil_v5 folder.
- 4. After installation, use File > Refresh as shown in Figure 18 to update your pack proprieties.

🛞 Pa	ack Installer - C:\Keil_v5\ARM\PACK	
File	Packs Window Help	
	Refresh	
I	Import Import from Folder Manage Local Repositories	Pack
	Fxit	Summary De

Figure 18. Refresh Pack after installation

5. The RSL10 CMSIS-Pack now appears in the list of installed packs. In the Devices tab, if you expand All Devices > ONSemiconductor > RSL10 Series, you can see RSL10 listed there. You can manage your installed packs in the Packs tab. Expanding ONSemiconductor.RSL10 makes the Pack Properties tab display the details of the RSL10 CMSIS-Pack. Figure 19 on page 20 illustrates what the Pack Installer perspective looks like after installation.

File Packs Window Help Device: ONSemiconductor - RSL10					
Devices Boards	·	⊳	4 Packs Examples		
Search: • X 🖻]		Show examples from installed Packs or	nly	
Device	/ Summary		Example	Action	Description
🗄 🔗 Maxim	9 Devices		ADC_UART (RSL10 Evaluation Board)	🔶 Сору	ADC with UART
🗉 🔗 MediaTek	2 Devices		ble_central_client_bond (RSL10 Eval	🔶 Сору	BLE Central Clie
🗉 🔗 Microchip	345 Devices		ble_central_client_scan (RSL10 Eval	🔶 Сору	Pairing and Bor
🗉 📍 Microsemi	6 Devices		ble_peripheral_server_bond (RSL10	🔶 Сору	BLE Peripheral S
MindMotion	2 Devices		ble_peripheral_server_hrp (RSL10 Ev	🔶 Сору	Pairing and Bor
Nordic Semiconductor	13 Devices		blinky (RSL10 Evaluation Board)	🔶 Сору	Blinky GPIO I/O
🗉 🔗 Nuvoton	487 Devices		default_MANU_INFO_INIT (RSL10 E	🔶 Сору	Default System
I VXP	1223 Devices		hci_app (RSL10 Evaluation Board)	🔶 Сору	Host Controller
ONSemiconductor	1 Device			🔶 Сору	I2C CMSIS-Driv
🖻 🍄 RSL10 Series	1 Device		kernel_timer (RSL10 Evaluation Boa	🔶 Сору	Kernel Timer Sa
RSL10	ARM Cortex-M3, 48 MHz		measure_rc_osc (RSL10 Evaluation	🔶 Сору	Measure 32 kHz
Redpine Signals	2 Devices		peripheral_server_standby (RSL10 E	🔶 Сору	Peripheral Devi
🗉 📍 Renesas	4 Devices		spi_cmsis_driver (RSL10 Evaluation	🔶 Сору	SPI CMSIS-Drive
Silicon Labs	783 Devices		supplemental_calibrate (RSL10 Eval	🔶 Сору	Default System
Sinowealth	1 Device		uart_cmsis_driver (RSL10 Evaluation	🔶 Сору	UART CMSIS-D
🗉 🔮 SONIX	50 Devices				
STMicroelectronics	1061 Devices				
🗉 💡 Texas Instruments	350 Devices	_	•		
Dutput			, With		
pdate available for Keil::ARM_Compiler (in:	stalled: 1.3.3, available: 1.4.0)				
pdate available for Keil::MDK-Middleware	(installed: 7.5.0, available: 7.6.0)				

Figure 19. Pack Installer after RSL10 CMSIS-Pack is Installed in the Keil $\mu\text{V}\textsc{ision}$ IDE

4.3 BUILDING YOUR FIRST SAMPLE APPLICATION WITH THE KEIL UVISION IDE

This section guides you through importing and building your first sample application, named *blinky*. This application makes the LED (DIO6) blink on the Evaluation and Development Board.

For more information about the sample applications, see the RSL10 Sample Code User's Guide.

4.3.1 Import 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 RSL10 CMSIS-Pack.
- 2. Choose the example project called *blinky*, and click the **Copy** button to import it into your workspace (see Figure 20 on page 21). Choose a destination folder for a copy of the sample code.

Search Example		
Search Example		
Example	Action	Description
blinky (RSL10 Evaluation Board)	🕸 Copy	Blinky GPIO I/O Sample Code
central_client (RSL10 Evaluation Board)	🚸 Сору	Central Device with Client Sample Code
central_client_double (RSL10 Evaluation	n B 🇇 Copy	Central Device with Client Sample Code - Double
central_peripheral (RSL10 Evaluation B	oar ጳ Copy	Central Peripheral Device Sample Code
custom_protocol_trx (RSL10 Evaluation	Вс 🇇 Сору	Low Latency Audio Sample Application with Custom Prot
default_MANU_INFO_INIT (RSL10 Evalu	iati ጳ Copy	Default System Initialization Function
hci_app (RSL10 Evaluation Board)	🚸 Сору	Host Controller Interface Application
pair_bond (RSL10 Evaluation Board)	🚸 Сору	Pairing and Bonding with Peripheral Device Sample Code
pair_bond_master (RSL10 Evaluation Bo	oar 🄄 Copy	Pairing and Bonding with Central Device Sample Code
peripheral_server (RSL10 Evaluation Bo	arc ጳ Copy	Peripheral Device with Server Sample Code
peripheral_server_FOTA (RSL10 Evaluat	ior ጳ Copy	Peripheral Device with Server for Sending Firmware Over .
peripheral_server_hrp (RSL10 Evaluatio	n B 🧇 Copy	Heart Rate Peripheral Device with Server Sample Code
peripheral_server_sleep (RSL10 Evaluat	ion 🇇 Copy	Sleep Mode Sample Code for Peripheral Device with Serv.
sleep_ble_advertisements (RSL10 Evalu	ati 🧇 Copy	Sleep and Wakeup with Bluetooth Low Energy Technolog.
sleep_RAM_retention (RSL10 Evaluation	n B 🧇 Copy	Sleep and Wakeup Sample Code
standby_power_mode (RSL10 Evaluation	on I 🇇 Copy	Standby Power Mode Sample Code
supplemental calibrate (RSL10 Evaluat	ion 🏵 Copy	Supplemental Calibration Sample Code

Figure 20. Pack Manager Perspective: Examples Tab

Sample projects are preconfigured with *Release* versions of RSL10 libraries, which are distributed as object files. For Keil, **System library** (*libsyslib*) and **Startup** (*libcmsis*) are preconfigured with the Source variant, so the source code of those libraries is included directly (see Figure 21 on page 22).

ct 🗣 🔝	Manage Run-Time Environment				-
Project: blinky	Software Component	Sel.	Variant	Version	Description
Source app.c include	Device Startup B- Bluetooth Profiles Libraries	V	source	1.0.0	Startup, System Setup System Startup for ON Semiconductor RSL10
Device Device Trillo_protocolc (LibrariesSystem) Trillo_romvectc (LibrariesSystem) Trillo_sys_asrcc (LibrariesSystem) Trillo_sys_audioc (LibrariesSystem) Trillo_sys_colockc (LibrariesSystem) Trillo_sys_crockc (LibrariesSystem) Trillo_sys_fact(LibrariesSystem) Trillo_sys_fact(LibrariesSystem) Trillo_sys_fact(LibrariesSystem) Trillo_sys_power (LibrariesSystem) Trillo_sys_fact(LibrariesSystem) Trillo_sys_fact(LibrariesSystem) Trillo_sys_fact(LibrariesSystem) Trillo_sys_fact(LibrariesSystem) Trillo_sys_fact(LibrariesSystem) Trillo_sys_tfmec (LibrariesSystem) Trillo_sys_tfmesc (LibrariesSystem) Trillo_sys_tfmesc (LibrariesSystem)			release source source release source source source release MDK-Plus MDK-Plus MDK-Plus	▼ 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 5.0.1 ▼ 7.9.0 ▼ 6.12.8	Weak Profile Library (likeak, prf) System Macros and Library (likesystiib) Remote Microphone Library (libremote micLib) Math Library (library) (libremote micLib) Event Kernel Library (librelib) Flash Library (libralibratelib) Cow Latency Audio Streaming Custom Protocol Library (libcustom protoc Calibration Library (libralibratelib) Biuetocoth Stack (libbelib) File Access on various storage devices User Interface on graphical LCD displays IPv4 Networking using Ethernet or Serial protocols USB Communication with various device classes
T rsl10_sys_version.c (LibrariesSystem) startup_rsl10.s (Startup) system_rsl10.c (Startup)	Validation Output		Descrip	tion	

Figure 21. RTE Configuration for the Blinky Example Project in the Keil μ Vision IDE

4.3.2 Build the Sample Code

Build the sample code as follows:

1. Right click on **Target 1** and choose **Rebuild all target files**. Alternatively, you can use the icon shown in the Figure 22.

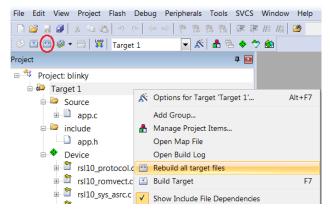


Figure 22. Starting to Build a Project in the Keil $\mu\text{Vision IDE}$

2. When the build is running, the output of the build is shown in the **Build Output** view in the IDE, as illustrated in Figure 23.

```
Build Output

*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'

Build target 'Target 1'

compiling app.c...

linking...

Program Size: Code=1508 RO-data=32 RW-data=4 ZI-data=3076

FromELF: creating hex file...

".\Objects\blinky.axf" - 0 Error(s), 0 Warning(s).

Build Time Elapsed: 00:00:02
```

Figure 23. 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

4.3.3 Debugging the Sample Code

4.3.3.1 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 the latest version of J-Link and have run the J-Link DLL Updater, as shown in Section 2.3, "Prerequisite Software" on page 6.

4.3.3.2 Debugging Applications

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

- 1. Make sure the Evaluation and Development Board is connected to the PC via a micro USB cable.
- 2. Select Debug > Start/Stop Debug Session or click the icon shown in Figure 24.

File Edit View Project Flash (Debug Peripherals Tools SVCS Window He	lp
🗋 🗃 😹 🎒 👗 🛍 🛍 🔊	Start/Stop Debug Session Ctrl+F5	const union gapc_d 🔻 🔜 🥐 🔍 🔿
🔗 🕮 🕮 🧼 🕶 🤐 🗱 🛛 Targe	Energy Measurement without Debug	\sim
	Reset CPU	

Figure 24. Start/Stop Debug Session Icon

If you are having trouble downloading firmware because an application with Sleep Mode is on the Evaluation and Development Board, see Section 6.4.1, "Downloading Firmware in Sleep Mode" on page 40.

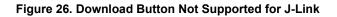
3. The application runs up to the first breakpoint in *main*, as shown in Figure 25. You can press F11 multiple times to step through the code and observe that the LED changes its state when the application executes the function Sys_GPIO_Toggle (LED_DIO).

File Edit View Project Flash Debug Peripherals Tools			anc d 💌 🗟 🥐 🔘 ·		2	
		-				
roject 🗸 🖉	Disassembly					ņ
 Project: blinky Target 1 Source p.c include app.h Device 	80:	801 CM 0116 BN Sys } else { 0006 MO if((uint8_	P r0, #0x01 E 0x00100574 _GPIO_Toggle (LED_D) VS r0, #0x06 t) DIO->CFG[gpio_pir	-	10_Mask) == 0)	
👜 🎬 rsl10_protocol.c (Libraries:System)	0∞00100548_0	INR1 TS	TS 91 90 #2			4
e→ [©] rsl10_romvect.c (Libraries:System) e→ [©] rsl10_sys_asrc.c (Libraries:System)	app.c	startup_r	sl10.s			-
 rsl10_sys_audio.c (Libraries:System) rsl10_sys_clocks.c (Libraries:System) rsl10_sys_crc.c (Libraries:System) rsl10_sys_dma.c (Libraries:System) rsl10_sys_power.c (Libraries:System) rsl10_sys_power.c (Libraries:System) rsl10_sys_timers.c (Libraries:System) startup_rsl10.s (Startup) system_rsl10.c (Startup) 	108 = { 109 = 1 109 = 1 109 = 1 110 = 1 111 = 1 112 = 1 113 = 1 114 = 1 115 = 1 116 = 1 117 = 1 118 = 1 120 = 1 121 = 1 122 = 1 123 = 1 124 = 1 125 = 1 126 = 1 127 = 1 4	<pre>led_tog /* Init Initial /* Spin while ({ /* Sys /* if { els { // //</pre>	<pre>clop */ l) Refresh the wat _Watchdog_Refre Toggle GPI0 6 (led_toggle_sta Sys_GPI0_Toggl e</pre>	<pre>cchdog timer */ sch(); if toggling is end tus == 1)</pre>	abled) then wa	Þ
ommand			Call Stack + Locals			Ļ
 evice: RSL10 Target = 3.300V tate of Pins: TCK: 0, TDI: 1, TDO: 1, TMS: 1, TRES ardware-Breakpoints: 2 oftware-Breakpoints: 8192	: 1, TRST: 1	•	Name — 🌳 main	Location/Value 0x00000000	Type int f()	
atchpoints: 1 IAG speed: 2667 kHz ad "C:\\Users\\zbhn3b\\Desktop\\cmsisp\\Files\\so JLink Info: J-Link: Flash download: Bank 0 @ 0x00 JLink Info: Executing RSL10 reset type: 0x0000000 S \\blinky\app.cl24 III	100000: Skipped.					
IAG speed: 2667 kHz Dad "C:\\Users\\zbhn3b\\Desktop\\cmsisp\\Files\\so Jink Info: J-Link: Flash download: Bank 0 @ 0x00 Jink Info: Executing RSL10 reset type: 0x0000000 S \\blinky\app.c\124	100000: Skipped. 0	Content E	- Call Stack + Locals	Memory 1		

Figure 25. Debug Session in the Keil $\mu\text{Vision IDE}$

NOTE: Debug configurations are preconfigured for the sample applications in the IDE's CMSIS-Pack. Flash downloading through the Download icon (Figure 26) or F8 is not supported for J-Link in the IDE at this point. The IDE may add support for this feature in future releases.

File Edit	View	Project	Flash	Debug	Peripherals
🗋 💕 (3 🥥	X 16 (B	19	€ ←	⇒ p p
0 🖾 🖉	🖆 🗳 🕶		Targ	et 1	- *
Project					



CHAPTER 5

Getting Started with IAR

5.1 PREREQUISITE SOFTWARE

Download and install the IAR Embedded Workbench from the IAR Website, using the vendor's instructions.

5.2 RSL10 CMSIS-PACK INSTALLATION PROCEDURE

To install the RSL10 CMSIS-Pack:

- 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 Figure 27.



Figure 27. Pack Installer Icon

 Click on CMSIS Manager > Import Existing Packs, select your pack file ONSemiconductor.RSL10.version.pack, and click Open (see Figure 28). version is the RSL10 version, such as 2.3.27.

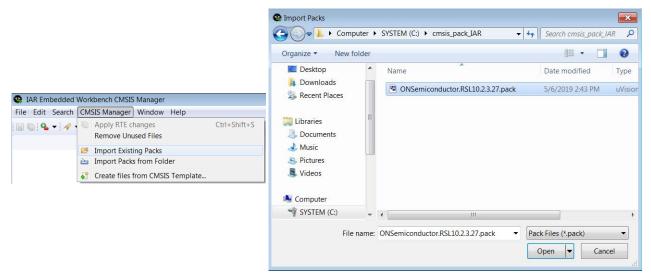


Figure 28. Installing the RSL10 CMSIS-Pack for the IAR Embedded Workbench IDE

- 4. The IDE prompts you to read and accept the license agreement, then installs the RSL10 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 Figure 29), to update your pack proprieties.

🖄 Packs 🛛	Devices	Boards	📑 Examples	E Console 😑 Pack Properties 👘 🗖
				⊞ ⊟ �� ≥ 🗠 Ø 🔻
Search Pack				
Pack			Action	Description

Figure 29. Refresh Pack after installation

6. In the **Devices** tab, expand **All Devices** > **ONSemiconductor** > **RSL10** Series, and select **RSL10** from the list. The RSL10 CMSIS-Pack now appears in the list of installed packs in the Packs tab. Expanding ONSemiconductor.RSL10 makes the Pack Properties tab display the details of the RSL10 CMSIS-Pack. Figure 30 on page 26 illustrates what the Pack Manager perspective looks like after installation.

IAR Embedded Workbench CMSIS	Manager			
File Edit Search CMSIS Manager	Window Help			
i 🛄 🔞 i 💁 🕶 i 🔗 🕶 i 🖗 💌 🖏 🕶 😂	⇔ • ⇔ •			Quick Access 🗄 📑 💽
	🚵 Packs 🛛 🔳 Devices 📓 Boards	E Pack Prope	rties 📑 Examples 📮 Console	🗄 🖂 🤣 🍣 🐸 🗎 🔍 🔻 🗖
	Search Pack			
	Pack	Action	Description	
	Device Specific	1 Pack	RSL10 selected	
	ONSemiconductor.RSL10	😻 Up to dat	ON Semiconductor RSL10 Device Family Pack	
	# 2.4.450	🔀 Remove	www.onsemi.com	
	 Generic 		Software Packs with generic content not specific to a devi	

Figure 30. The IAR Embedded Workbench CMSIS Manager after RSL10 CMSIS-Pack is Installed

5.3 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 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 RSL10 Sample Code User's Guide.

5.3.1 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 RSL10 CMSIS-Pack.

Choose the example project called *blinky*, and click the Copy button to import it into your workspace (see 2. Figure 31 on page 27). Choose a destination folder for a copy of the sample code.

🕸 Packs 🔳 Devices 📓 Boards 📑 Example		sole	🗆 Only show examples from installed packs 🛛 🖑 🏞 🐸 🔯 💿 🚿
Search Example			
Example	Action	Description	
ADC_UART (RSL10 Evaluation Board)	🗇 Сору	ADC with UART Sample Code	
ble_central_client_bond (RSL10 Evaluation	🗇 Сору	BLE Central Client Bonding Sample Code	
ble_central_client_scan (RSL10 Evaluation	🔶 Сору	Central Device with Client Scanner Sample Code	
ble_peripheral_server_bond (RSL10 Evaluation	🔶 Сору	BLE Peripheral Server Bonding Sample Code	
ble_peripheral_server_hrp (RSL10 Evaluati	🔶 Сору 👘	Heart Rate Peripheral Device with Server Sample Code	
blinky (RSL10 Evaluation Board)	💠 Сору	Blinky GPIO I/O Sample Code	
default_MANU_INFO_INIT (RSL10 Evaluati	💠 Сору	Default System Initialization Function	
hci_app (RSL10 Evaluation Board)	🔶 Сору	Host Controller Interface Application	
i2c_cmsis_driver (RSL10 Evaluation Board	🔶 Сору	I2C CMSIS-Driver Sample Code	
kernel_timer (RSL10 Evaluation Board)	💠 Сору	Kernel Timer Sample Code	
measure_rc_osc (RSL10 Evaluation Board)	💠 Сору	Measure 32 kHz RC Oscillator	
peripheral_server_sleep (RSL10 Evaluation	🔶 Сору	Sleep Mode Sample Code for Peripheral Device with Serv	
peripheral_server_standby (RSL10 Evaluat	і� Сору	Peripheral Device with Server and Standby Power Mode S	
spi_cmsis_driver (RSL10 Evaluation Board	🔶 Copy	SPI CMSIS-Driver Sample Code	
supplemental_calibrate (RSL10 Evaluation	🔶 Сору	Default System Initialization Function	
uart_cmsis_driver (RSL10 Evaluation Board	💠 Сору	UART CMSIS-Driver Sample Code	

Figure 31. IAR Embedded Workbench CMSIS Manager: Examples Tab

Sample projects are preconfigured with Release versions of RSL10 libraries, which are distributed as object files. For the IDE, System library (libsyslib) and Startup (libcmsis) are preconfigured with the Source variant, so the source code of those libraries is included directly in both CMSIS Manager and IDE windows (see Figure 32 on page 27 and Figure 33 on page 28).

ile Edit Source Refactor Navigate Search Projec	t CMSIS Manager Run W	indow Help					
	* • 0 • 💁 • 🥭 🛷 •	· [] [] [] [] [] · [] · [] · []	0 ¢	• 🗘 •			
È Project Explorer ⊠		♦ blinky.rteconfig 🛙					
🔺 🗁 blinky		The Components I Re	solve				
🖻 🗁 Debug							
🕨 🗁 include		Software Components	Sel	. Variant	Vendor	Version	Description
🔺 🏂 RTE		RSL10			ONSemiconduc		ARM Cortex-M3 48 MHz, 24 kB
🔺 🗁 Device		Device					
🔺 🗁 RSL10		Bluetooth Profile	95				
R rsl10_protocol.c [ONSemiconductor::Dev	ice.Libraries.System.source]	Libraries					
ronvect.c [ONSemiconductor::Device.Libraries.System.source] ronvect.c [ONSemiconductor::Device.Libraries.System.source]	BLE	Ц	release	ONSemiconduc	2.4.450	Bluetooth Stack (libblelib)	
	ce.Libraries.System.source]	Calibrate		source	ONSemiconduc	2.4.450	Calibration Library (libcalibrate
R rsl10_sys_audio.c [ONSemiconductor::De		Custom Proto	cΩ	source	ONSemiconduc	2.4.450	Low Latency Audio Streaming
R rsl10_sys_clocks.c [ONSemiconductor::De		Flash		source	ONSemiconduc	2.4.450	Flash Library (libflashlib)
R rsl10_sys_crc.c [ONSemiconductor::Devic		📍 Fota		release	ONSemiconduc	2.4.450	Fota Library (libfota)
R rsl10_sys_dma.c [ONSemiconductor::Dev		Kernel		release	ONSemiconduc	2.4.450	Event Kernel Library (libkelib)
R rsl10_sys_flash.c [ONSemiconductor::Dev	and the second	Math		source	ONSemiconduc	2.4.450	Math Library (libmathlib)
R rsl10 sys power modes.c [ONSemicondu		Remote_Mic		source	ONSemiconduc	2.4.450	Remote Microphone Library (li
rsl10_sys_power_modes.c [ONSemiconductor::De	he are the set of the	System		source	ONSemiconduc	2.4.450	System Macros and Library (lib
rsl10 sys rffe.c [ONSemiconductor::Device [ONSemiconductor::Device]		Weak_PRF		release	ONSemiconduc	2.4.450	Weak Profile Library (weak_prf)
rsl10 sys timers.c [ONSemiconductor::Device [ONSemiconductor::Device]		Startup		source	ONSemiconduc	2.4.450	System Startup for ON Semicor
R rsl10 sys uart.c [ONSemiconductor::Devi		•	-			111	
R rsl10 sys version.c [ONSemiconductor::D		Validation Output			Dec	cription	
startup_rsl10.s [ONSemiconductor::Devic		valuation Output			Des	cripuon	
system_rsl10.c [ONSemiconductor::Devic							
RTE_Components.h	ere cal colore er e e l						

Figure 32. RTE Configuration for the Blinky Example Project in the IAR Embedded Workbench CMSIS Manager window

bug es Device Startup source Device Startup source Device Startup source Device Startup source Device Libraries System source Device Startup System Startup System source Device Startup System Startup Star	• 4 ×						
es blinky - Debug CMSIS-Pack CMSI	•						
blinky - Debug CMSIS-Pack RTE_Components.h Device Startup source system_rsi10.c Device Libraries System source rsi10_protocol.c rsi10_protocol.c rsi10_sys_audio.c rsi10_sys_power.c rsi10_sys_power.c rsi10_sys_power.d rsi10_sys_power.c rsi10_sys_power.c rsi10_sys_power.c rsi10_sys_power.c 	•						
DCMSIS-Pack Device.Startup source Device.Startup source Device.Startup source Device.Startup resilos Device.Ubranes.System source Device							
RTE_Components h Device.Startup source Device.Startup_rsl10.s Device.Ubranes.System source Device.Ubranes.System source Device.Ubranes.System source Device.Ubranes.System.source Device.ubranes.System.source Device.ubranes.System.co Device.ubranes.co Dev							
Device Statup source Device Statup source Device Libraries System sources							
Image: Big startup_rsillos Image: Big system_rsilloc							
B system_still c Device_Libraries_System source Device_Libraries_System sources_Libraries_System source Device_Libraries_System sources_Libraries_System source Device_Libraries_System sources_Libraries_System sources_Libraries_System sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sources_Libraries_Sourc							
Device Libraries System source Device Libr							
→ ⊞ □ rsl10_sys_sarcc → ⊞ □ rsl10_sys_soudic.c → ⊞ □ rsl10_sys_clocks.c → ⊞ □ rsl10_sys_clocks.c → ⊞ □ rsl10_sys_clocks.c → ⊞ □ rsl10_sys_clocks.c → ⊞ □ rsl10_sys_power.c → ⊞ □ rsl10_sys_power.c → ⊞ □ rsl10_sys_power.c → ⊞ □ rsl10_sys_power.c							
-00 rs110_sys_asr.cc -00 rs110_sys_audio.c -00 rs110_sys_clocks.c -00 rs110_sys_drac. -00 rs110_sys_plash.c -00 rs110_sys_power.c -00 rs110_sys_power.c -00 rs110_sys_power.c -00 rs110_sys_power.c							
HB rs110_sys_clocks.c HB rs110_sys_crc.c HB rs110_sys_dma.c HB rs110_sys_dtash.c HB rs110_sys_power.c HB rs110_sys_power.c HB rs110_sys_power.c HB rs110_sys_power.c HB rs110_sys_filec.c							
	:						
	:						
	:						
──⊞ lo rsi10_sys_power.c ──⊞ lo rsi10_sys_power_modes.c ──⊞ lo rsi10_sys_rffe.c							
Here Institution							
He isi10_sys_rffe.c	-						
	•						
	٠						
–⊞ 🗟 rsl10_sys_timers.c							
⊣⊞ 🗟 rsl10_sys_uart.c							
└─⊞ 🖻 rsl10_sys_version.c							
🗉 🛋 include							
🖽 🗟 app.c							
– 🗎 readme_blinky.txt							
🗉 🛋 Output							

Figure 33. RTE Configuration for the Blinky Example Project in the IAR Embedded Workbench window

5.3.2 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 Figure 34.

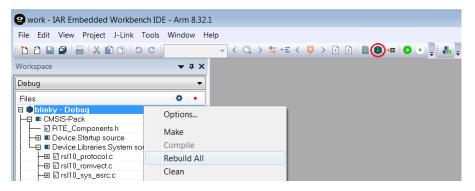


Figure 34. Starting to Build a Project in the IAR Embedded Workbench

2. When the build is running, the output of the build is displayed in the **Build Output** view in the IDE, as illustrated in Figure 35.

	Files	•
	🗆 🌒 blinky - Debug	~
uild	— ⊞ 🛋 include	
ana	🖃 🖬 app.c	
Messages	- 🗎 readme_blinky.txt	
5	- 🖽 🔳 CMSIS-Pack	
Building configuration: blinky - Debug	🖵 📮 🛑 Output	
Updating build tree	- B blinky.map	
startup_rsl10.s	🖵 🖸 blinky.out	
rsl10_protocol.c	- 🛱 💼 Output	
rsl10_romvect.c	blinky.hex	
rsl10_sys_asrc.c	blinky.map	
app.c	- D app.o	
rsl10_sys_audio.c	- dl7M tln.a.	
rsl10_sys_clocks.c	- 1 m7M tl.a	
rsl10_sys_crc.c		
rsl10_sys_dma.c	- I rsl10 romvect.o	
rsl10_sys_flash.c rsl10_sys_power.c rsl10_sys_ffe.c rsl10_sys_power_modes.c rsl10_sys_timers.c	- I rsl10_sys_asrc.o	
	- Instructions	
	- Institution sys clocks.o	
	rsi10_sys_crc.o	
	isino_sys_cic.o	
rsl10_sys_uart.c	rsi10_sys_dila.0	
rsl10_sys_version.c	rsi10 sys power.o	
system_rsl10.c	rsi10_sys_power.o	
Linking	rsi10 sys_power_modes.o	
blinky.out	rsi10_sys_timers.o	
Converting	rsi10_sys_unters.0	
	rsi10_sys_uarto	
Total number of errors: 0		
Total number of warnings: 0	C rt7M_tl.a.	
-		
Build Debug Log	shb_l.a	
	Startup_rsl10.o	
	🖵 🗋 system_rsl10.o	

Figure 35. 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

5.3.3 Debugging the Sample Code

5.3.3.1 Debugging Applications

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

- 1. Make sure the Evaluation and Development Board is connected to the PC via a micro USB cable.
- 2. Select **Project > Download and Debug,** or click the icon shown in Figure 36, then accept the J-Link pop-up dialog in order to use the flash breakpoints (as shown in Figure 37).



Figure 36. Start/Stop Debug Session Icon

🔝 J-Li	ink V6.34h Out of breakpoints
	The debugger is trying to set a breakpoint in flash memory at address 0x001003E8. 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 http://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: 483035975
	🖾 Do not show this message again fi
	Yes No Install existing license

Figure 37. J-link "Out of breakpoints" pop-up dialog

If you are having trouble downloading firmware because an application with Sleep Mode is on the Evaluation and Development Board, see Section 6.4.1, "Downloading Firmware in Sleep Mode" on page 40.

3. The application runs up to the first breakpoint in *main*. You can press F5 or 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 function Sys GPIO Toggle (LED DIO). To stop the debug session, press the Stop icon.

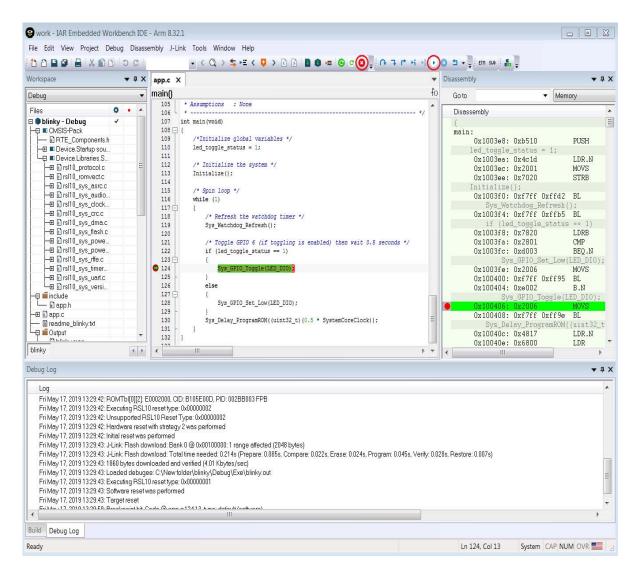


Figure 38. Debug Session in the IAR Embedded Workbench

CHAPTER 6

Advanced Debugging

6.1 PRINTF DEBUG CAPABILITIES

The PRINTF() macro is used to provide printf() debug capability in RSL10 applications. The implementation of the PRINTF () macro is user selectable to allow for different types of debug interfaces. The functionality is accessed via the tracing API.

The tracing API supports two debug interfaces: UART and RTT. The implementation of the tracing functions can be found in the *app_trace.c* file. The developer can select the debug interface during the compilation process by setting the RSL10 DEBUG macro in the *app_trace.h* file. If the macro is set to DBG NO, tracing is disabled. This is the default behavior in all sample applications.

NOTE: The files *app_trace.c* and *app_trace.h* need to be present in your sample application, and initialized using TRACE INIT(), in order to for you use the PRINTF() feature. You can find these two required files in most Bluetooth Low Energy sample applications, such as ble_peripheral_server_bond.

To debug time critical applications, we recommend setting the tracing option to DBG RTT option. With SEGGER RTT (Real Time Transfer), you can output information from the target MCU to the RTT Viewer application at a very high speed without compromising the target's real time behavior. More information about SEGGER RTT can be found in JLINK user manual, at www.segger.com.

6.1.1 Adding Printf Debug Capabilities

To add printf debug capabilities over UART, change the define in the app trace.h file to #define RSL10 DEBUG DBG UART, and set the RSL10 DEBUG macro to DBG UART. A standard terminal program on a PC can be used to view the debug output.

To add RTT printf debug capabilities, change the define in the *app trace*.h file to #define RSL10 DEBUG DBG RTT and add the SEGGER RTT files to the application. The Segger RTT Viewer application on a PC can be used to view the debug output.

Samples for RTT are under C:\Program Files (x86)\SEGGER\JLink V640b\Samples\RTT.

More information about the RTT API can be found in the JLINK manual, under C:\Program Files (x86)\SEGGER\JLink V640b\Doc\Manuals.

NOTE: Note that these RTT sample and information files are for SEGGER JLink version 640b.

6.2 DEBUGGING APPLICATIONS THAT DO NOT START AT THE BASE ADDRESS OF FLASH

If you want to debug an application that does not start at the first address of the flash memory (0x00100000), read on. For example, you might be debugging an application in RAM, or a flash memory application that has been placed in a different address.

This procedure assumes you have performed the steps in Section 3.3.1, "Debugging with the .elf File" on page 13, and you are using the ON Semiconductor IDE:

- 1. In your Debug configuration, change to the **Startup** tab
- 2. Enter the following in the **Run/Restart Commands** field as illustrated in Figure 39:

Main 🕸 Debugger 🕨	Startup 🛛 🦆 Sc	ource 🗖 Comn	non				
Initialization Commands							
Initial Reset and Halt	Type:	Low s	peed: 1000	kHz			
JTAG/SWD Speed: 💿 A	uto 🔘 Adaptive	e 🔘 Fixed	kHz				
🔽 Enable flash breakpo	ints						
Enable semihosting		to: V Telnet 🛽	GDB client				
Enable SWO CPU free	q: 10000000	Hz. SWO free	ą: O	Hz. Port mask	: 0x1		
						^	
						-	
and Sumbols and Sussi	tabla						
Load Symbols and Execu Load symbols	lable						
Use project binary:	blinky.elf						
O Use file:				Works	pace	File System	
Symbols offset (hex):							
Load executable							
Our State of Use project binary:	blinky.elf						
O Use file:				Works	pace	File System	
Executable offset (hex):							
Runtime Options	oad after each n	eset/restart)					
		use y restarty					
Run/Restart Commands	Turner		(a)				
Pre-run/Restart reset			(always ex	ecuted at Rest	art)		
<pre>set {int} &VTOR = ISF set \$sp = *((int *) &ISR</pre>						^	
						-	
Set program counter	at (hex):						
Set program counter Set breakpoint at:	at (hex):						

Figure 39. Setting Up a GDB Launch Configuration, Startup Tab

6.3 Arm Cortex-M3 Core Breakpoints

A maximum of two hardware breakpoints can be set at a given time. If you need more than two breakpoints, you can use the Unlimited Flash Breakpoints feature available through J-Link.

IMPORTANT: You can use hardware breakpoints when using the debugger with the Arm Cortex-M3 core, but software breakpoints cannot be used with the flash overlay. Writing to flash memory does not place breakpoints within the overlay, so any attempt to use software breakpoints would be ineffective.

6.4 DEBUGGING WITH LOW POWER SLEEP MODE

Debugging applications that use sleep mode is a challenging task because the hardware debug logic and system clocks are powered down when the device goes to sleep. Therefore, the debug session cannot be kept alive between sleep cycles.

Besides using GPIOs, UART, and other peripherals as tools to help debug your application, you can reattach the debugger after the device wakes up from sleep. To do so, you need to make sure that the device stays awake, and start a new debug session to connect to the running target, making sure a reset is not performed. The following instructions show an example of how to perform this on the *peripheral_server_sleep* sample application in the ON Semiconductor IDE, but you can also adapt it for other applications that use sleep mode, and for other IDEs.

- 1. Copy the *peripheral_server_sleep* application into your workspace and navigate to the *app_process.c* source file under the *code* folder.
- 2. Modify the function void Continue_Application (void) by adding a while loop before the Main_Loop(); call, to make sure that the device stays awake in the infinite loop after waking up (see Figure 40). Save and compile your application.

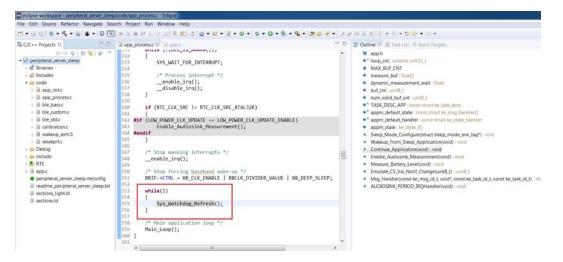


Figure 40. Continue_Application Function Perspective After Adding While Loop

- 3. Within the Project Explorer, right-click on the *.elf* file and select **Debug As > Debug Configurations**.
- 4. When the **Debug Configurations** dialog appears, create two debug sessions:
 - a. Debug session that initiates restart and halts the target:
 - i. Right-click on **GDB SEGGER J-Link Debugging** and select **New**. A new configuration appears under the **GDB SEGGER** heading, with new configuration details in the right panel.
 - ii. Adjust the displayed values for your configuration and click on **Apply** (see Figure 41 and Figure 42 on page 35).
- NOTE: If you are having trouble downloading firmware to the device, in addition to using DIO12, you can also perform the software recovery by setting the **Reset Type** to 1 in the **Debug** session configuration (see Figure 41). The default Reset Type is 0, which only resets the Arm Cortex-M3 core while leaving the device/peripherals in a state where J-Link can't reconnect. Setting the Reset Type to 1 ensures that not only is the Arm Cortex-M3 core reset, but so are all the peripherals. If this does not work, see Section 6.4.1, "Downloading Firmware in Sleep Mode" on page 40.

eate, manage, and run configurati	ions 🕺
🔞 🗶 🖶 🕈 🕈	Name: peripheral_server_sleep Debug
ype filter text C/C++ Application C/C++ Attach to Application C/C++ Attach to Application C/C++ Remote Application C/C++ Remote Application C/DE Hardware Debugging C/DE Application C/DE SEGGER J-Link Debugging C/DE peripheral_server_sleep Debug C/DE peripheral_server_sleep.pud, att	Main & Debugger Startup & Source Common & SVD Path Initialization Commands Source Common & SVD Path Initialization Commands Initialization Initialization Commands Initialization Init
Isva Applet Java Applet Java Applet Java Applet Java Javia Javia Java Sourd Sourd Sourd Sourd Remote Java Application	Load Symbols and Executable C Load Symbols Use project binary: peripheral_server_sleep.elf Use file: Use project binary: peripheral_server_sleep.elf Use file: Executable offset (hex): Executable offset (hex): Executable o
ter matched 19 of 100 items	Revert Apply

Figure 41. Setting Reset Type in the Debug Configuration Session

reate, manage, and run configurations	3						
🗎 🗶 🖶 🖶 💌	Name: peripheral_server_sleep Debug						
ype filter text	Main P Debugger Startup V Source Common S SVD Path						
C/C++ Application	Initialization Commands						
C/C++ Attach to Application C/C++ Postmortem Debugger	☑ Initial Reset and Halt Type: Low speed: 1000 kHz						
C/C++ Postmortem Debugger	JTAG/SWD Speed:						
Eclipse Application							
C GDB Hardware Debugging							
GDB OpenOCD Debugging	Enable SWO CPU freq: 0 Hz. SWO freq: 0 Hz. Port mask: 0x1						
 C GDB SEGGER J-Link Debugging c peripheral_server_sleep Debug 							
I Java Applet							
Java Application	•						
Ju JUnit jt JUnit Plug-in Test & Launch Group	Load Symbols and Executable						
	✓ Load symbols						
 Launch Group (Deprecated) 	Use project binary: peripheral_server_sleep.elf						
Mwe2 Launch	O Use file: Workspace File System						
OSGi Framework Remote Java Application	Symbols offset (hex):						
	Coad executable						
	Use project binary: peripheral_server_sleep.elf						
	O Use file: Workspace File System						
	Executable offset (hex):						
	Runtime Options						
	RAM application (reload after each reset/restart)						
	Run/Restart Commands						
	Pre-run/Restart reset Type: (always executed at Restart)						
	·						
	Set program counter at (hex):						
	Set breakpoint at: main						
	Continue						
Iter matched 18 of 107 items	Revert Apply						

Figure 42. Startup Tab: Debug Session that Initiates Restart

- b. Debug session that connects to the running target:
 - Create another new debug configuration under the GDB SEGGER heading, with new configuration i. details in the right panel.
 - ii. Adjust the displayed values for your configuration then click on Apply (see Figure 43 and Figure 44 on page 37).

reate, manage, and run configurations) C								
1 R X 8 3 ·	Name: peripheral_server_sleep Debug_swd_att								
type filter text	Main Debugger Startup Source Common 5 SVD Path								
yge hiter text © (C++ Application © (C++ Attach to Application © (C++ Attach to Application © (C++ Remote Application © GBB Verbacks Debugging © GBB Verbacks I-Hink Debugging © DeB SEGER3-Hink Debugging © peripheral_server_siese Debug © peripheral_server_siese Debug © Java Application Jr. Nuht © Java Application Jr. Nuht © Java Application Jr. Muht © Java Application © GGB Framework © Remote Java Application	Main © Debuger & Startue] & Source C Common & SVD Path - Luk COB Server Statu Startue + Luk COB server locally Connection: Splink.gbb/sEGGER/Luk V634h/Luk CDBServerCLess to change tus we global or workspace preferences pages of the proceed properties page) Device name: SSL0 Endiamesis & Uittle Big Connection: USB [] USB servia of P name/address) Interface: USB [] USB servia of P name/address I tele topt: 2333 I Uscal host only] Silert I Log file: USB server I Recorder for semihosting and SWO COB Cleft Setup Executable: arm-one-rabigdb Command: eff memory in arm-one-rabigdb Command: eff memory in accessible-by-default off I Remote Target Host name of P address Iocalhost Port number: 233 I Cord memory II address Iocalhost Port number: 233 I Cord device de								
ilter matched 19 of 107 items	Revert Apply								

Figure 43. Debugger Tab: Debug Session that Connects to the Running Target

Image: Sever_sleep Debug Type filter text Image: Debugger Image	~~
Eclipse Application IV Enable flash breakpoints GDB Hardware Debugging IV Enable semihosting Console routed to: IV Telnet □ GDB client GDB OpenOCD Debugging V Enable SWO CPU freq: 0 Hz. SWO freq: 0 Hz. Por	t mask 0x1
C GDB SEGGER J-Link Debugging peripheral_server_sleep_Debug Java Applet Java Applet Java Application Jr JUnit Julit Julit Load symbols and Executable Julit Load symbols Use project binary: peripheral_server_sleep.elf Luce file:	Workspace File System
Launch Group (Deprecated) Mwe2 Launch OSGi Framework OSGi Framework Cload executable Use project binary: peripheral_server_sleep.elf Use file: Executable offset (hex):	Workspace File System
ilter matched 19 of 100 items	Revert Apply

Figure 44. Startup Tab: Debug Session that Connects to the Running Target

- NOTE: If you are having trouble connecting to the running target, you can perform a software recovery by setting the Reset Type to 1 in the Debug session configuration as shown in Figure 44. The default Reset type is set to 0, which only resets the Arm Cortex-M3 core and leaves the device/ peripherals in a state where the J-Link cannot reconnect. By setting it to 1, you ensure that not only the Arm Cortex-M3 core is reset, but also all the peripherals. If this does not work, see Section 6.4.1, "Downloading Firmware in Sleep Mode" on page 40 for more ideas.
- 5. Start the first debug session (which initiates target restart). Once the target is halted at main, resume the execution (see Figure 45).

Getting Started with RSL10

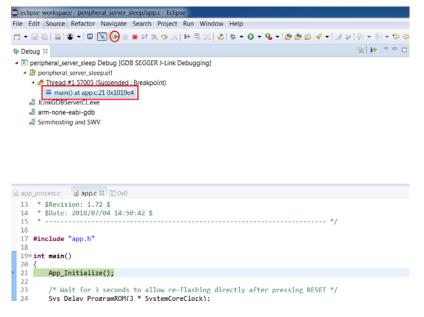


Figure 45. First Debug Session Perspective Before Starting Execution

Wait until the target enters Deep Sleep Mode. At this point the debug connection is lost; and even when the 6. target is awake, it cannot establish a connection with JTAG. The following output is generated on the console (see Figure 46).

🗳 Console 🛱 🔄 Tasks 🖹 Problems 🕥 Executables 📓 Debugger Console 🔋 Memory 🛷 Search
peripheral_server_sleep Debug [GDB SEGGER J-Link Debugging] JLinkGDBServerCL.exe
ERROR: Can not read register 2 (R2) while CPU is running
ERROR: Can not read register 3 (R3) while CPU is running
ERROR: Can not read register 4 (R4) while CPU is running
ERROR: Can not read register 5 (R5) while CPU is running
ERROR: Can not read register 6 (R6) while CPU is running
ERROR: Can not read register 7 (R7) while CPU is running
ERROR: Can not read register 8 (R8) while CPU is running
ERROR: Can not read register 9 (R9) while CPU is running
ERROR: Can not read register 10 (R10) while CPU is running
ERROR: Can not read register 11 (R11) while CPU is running
ERROR: Can not read register 12 (R12) while CPU is running
ERROR: Can not read register 13 (R13) while CPU is running
ERROR: Can not read register 14 (R14) while CPU is running
ERROR: Can not read register 15 (R15) while CPU is running
ERROR: Can not read register 16 (XPSR) while CPU is running
ERROR: Can not read register 17 (MSP) while CPU is running
ERROR: Can not read register 18 (PSP) while CPU is running
ERROR: Can not read register 24 (PRIMASK) while CPU is running
ERROR: Can not read register 25 (BASEPRI) while CPU is running
ERROR: Can not read register 26 (FAULTMASK) while CPU is running
ERROR: Can not read register 27 (CONTROL) while CPU is running
WARNING: Failed to read memory @ address 0xDEADBEEE
Starting target CPU
ERROR: CPU is not halted
ERROR: Can not read register 15 (R15) while CPU is running
Reading all registers

Figure 46. Debug Session Perspective when Debug Connection is Lost

7. Stop the debug session and click on the **Terminate** icon to remove all terminated targets (see Figure 47).

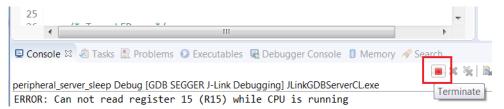


Figure 47. Terminate Targets Icon

8. After the target exits Deep Sleep Mode, it is running in the infinite loop (step 1), and we can connect to the running target by starting the second debug session (see Figure 48). Note that the debugger is able to reattach to the running target and halt the processor after waking up from sleep.

e Edit Source Refactor Navigate Search Project	Run Window Help						
• © © K • 0 💌 • 10 • 1 • 1 × 2 • 1	14 K + K & C + Q + Q + B & K + I + V - A + B & K + K + V + V + V + V + V + V + V + V +	N+N+00+0+			Quick Access		
Debug II	2 4 T T D	🗠 Variables 🍬 Breakpoints 🐔 Expre	ssions 12 III Registers 🗄 Signals 🛤 Modules 5	Peripherals	의 🔟 원 🖉 위 🖄 (1) 년 🔻		
Deripheral server_sleep_swd_att (GDB SEGGER J-Link Debugging) #B peripheral_server_sleep_eff # Thread # 157005 (Suppended : Signal: SIGINT/Interrupt)		Expression Type			Value		
		№ _stack	<text debug="" info="" no="" variable,=""></text>		{ <text debug="" info="" no="" variable,="">} 0x20005fe8</text>		
		Add new expression					
Continue_Application() at app_process.c354 (Wakeup From Sleep Application() at app_pro							
waxeup_rom_seep_Application() at app_pri Ovtfffffe	ocess.cz81 042000008						
a vomme √II JinkGD8serverCLexe ↓I arm-none-eabi-gdb							
J Semihosting and SWV							
						-	
		4			y		
app_process.c 22 @ app.c Ovffffffe 18 rs10_s	vs watchdog.h		- 0	BE Outline 11	PB588.**		
49				app.h			
59 /* Stop forcing baseband wake-up */ 51 BBIF-SCTRL = BB_CLK_DIVIDER_VALUE BB_DEEP_SLEEP;				Voop_ont:volatile uint32_t MAX_BUF_CNT			
			# MAX B				
2 while(1)					re_buf : float[]		
Sys_Watchdog_Refresh();					ic_measurement_wait : float		
55 }				 buf_cnt 		-	
0* Main application loop */ 7 Main_Loop(); 558 559				num_valid_buf_ont : uint8_t * TASK_DESC_APP : const struct ke_task_desc			
			• appm_default_state : const struct ke_msg_handler[]				
				default_handler : const struct ke_state_handler			
60 61#/*				· appm_s	state : ke_state_t[]		
52 * Function : void Enable Audiosin					Mode_Configure(struct sleep_mode_erv_tag*) ::	void	
					p_From_Sleep_Application(void) : void		
	tely RC_OSC_MEASUREMENT_INTERVAL seconds				ue_Application(void) : void		
365 * before enabling audiosink interrupt. *				_Audiosink_Measurement(void) : void	-		

Figure 48. Second Debug Session Perspective After Connecting to the Running Target

6.4.1 Downloading Firmware in Sleep Mode

If an application with Sleep Mode is currently on your board, and changing the Reset Type to 1 as described in Section 6.4, "Debugging with Low Power Sleep Mode" is not working, try the following:

- 1. Connect DIO12 to ground.
- 2. Press the RESET button (this restarts the application, which pauses at the start of its initialization routine).
- 3. Repeat step 2 above. After successfully downloading *blinky* to flash memory, disconnect DIO12 from ground, and press the RESET button so that the application works properly.

Alternatively, use the Stand-Alone Flash Loader (available with its own manual in the *RSL10_Utility_Apps.zip* file) to erase the application with Sleep Mode from the board's flash memory.

CHAPTER 7

More Information

7.1 FOLDER STRUCTURE OF THE RSL10 CMSIS-PACK INSTALLATION

By default, your files are installed in the following location:

- If you are using the Eclipse-based ON Semiconductor IDE: C:\Users\<user_id>\ON_Semiconductor\PACK.
- If you are using the Keil IDE: C:\Keil_v5\ARM\PACK
- If you are using the IAR IDE: C:\Users\<user_name>\IAR-CMSIS-Packs

Subfolders are described in Table 1, below, and Table 2 on page 41.

Table 1. Installed Folders - CMSIS-Pack

Folder	Contents		
configuration	J-Link flash loader files.		
documentation	Hardware, firmware and software documentation in PDF format. Also 3rd-party documentation from other companies besides ON Semiconductor. Available from the books tab in the IDE.		
images	Contains evaluation board pictures.		
include	Include files for the firmware components and libraries. Projects can point to this directory and sub-directories when including firmware header files.		
lib	Pre-built libraries which can be linked to by sample code or other source code. Project linker settings must point to this directory when linking with firmware libraries.		
source	firmware	The source of the provided support libraries.	
	samples/rslx (for ON Semiconductor IDE) samples/uv (for Keil IDE) samples/iar (for IAR IDE)	Sample code sources as ready-to-build projects.	
svd	Contains the System View Description file used in the registers view during debugging.		

Table 2. Installed Folders - ON Semiconductor IDE

Folder	Contents
arm_tools	The Arm Toolchain is installed here.
eclipse	Pre-built libraries which can be linked to by sample code or other source code. Project linker settings must point to this directory when linking with firmware libraries.
jre*	The included JAVA runtime environment.
ON Semiconductor IDE	ON Semiconductor license agreement, revision file and pack description file.

7.2 DOCUMENTATION

7.2.1 Documentation Included with the CMSIS-Pack

A set of documents is included with the CMSIS-Pack installation in C:\Users\<user_id>\ON_Semiconductor\PACK\ONSemiconductor\RSL10\<pack_version>\documentation (where <user_id is your profile name, and <pack_version> is the version number, e.g., 3.0.521).

These documents are also accessible via any of the three IDEs:

Getting Started with RSL10

- ON Semiconductor IDE: documentation is accessible through the C/C++ perspective by opening any RTE • configuration file, such as *blinky.rteconfig*, and selecting the tab **Device** (see Figure 49 on page 42).
- Keil µVision IDE: documentation is available in the **Books** tab, as shown in Figure 50 on page 43. ٠
- IAR Embedded Workbench: documentation is accessible through the IAR Embedded Workbench CMSIS • Manager window, as shown in Figure 51 on page 44.

- 🔜 🐘 🥹 - 🧞 - 📾 🕍 - 63 - 67 - 67 -	\$ • 0 • 9, • 9, • 12 @ @ # • 10 0 0 0 0 • 10 1 × 1 = 11 × 1 = 11 × 1 = 11 × 1	御 * 御 * 🌣 🗢 * 🔶 *
Project Explorer 🛛 📄 😘	Image: Start.c ◆ blinky.rteconfig Image: Ox0 @ main.c	
🛎 blinky	I Device	0
Binaries	Device: RSL10	
2 Includes		hange
Debug	Family: RSL10 Series	CPU: ARM Cortex-M3
blinky.elf - [arm/le]	SubFamily:	Max. Clock: 48 MHz
 Immanuo - (armyle) blinky,hex 	Vendor: ONSemiconductor	Memory: 32 kB RAM, 384 kB ROM
iii blinky.map	Pack ONSemiconductor.RSL10.1.2.0	FPU: none
main.d	URL: http://www.keil.com/dd2/onsemiconductor/rsl10	Endian: Little-endian
a makefile	Device data books:	Description:
lo objects.mk	ARM and Thumb-2 Instruction Set Quick Reference Card	RSL10 is an ultra-low-power, multi-protocol 2.4
la sources.mk	Sap Interface Specification	GHz radio designed for use in wireless devices
li subdir.mk	SATT Interface Specification	that demand low power consumption and a
s RTE	L2C Interface Specification	restricted size.
i main.c	RSL10 Firmware Reference	
blinky.rteconfig	RSL10 Hardware Reference	
readme_blinky.txt	RSL10 Sample Code User's Guide	
sections.ld	RW BLE Alert Notification Profile Interface Specification	
	RW BLE Battery Service Interface Specification	
	RW BLE Blood Pressure Profile (BLP) Interface Specification	
	RW BLE Cycling Power Profile Interface Specification	
	RW BLE Cycling Speed and Cadence Profile Interface Specification	
	RW BLE Device Information Service Interface Specification	
	RW BLE Find Me Profile Interface Specification	
	RW BLE Glucose Profile (GLP) Interface Specification	
	RW BLE Health Thermometer Profile Interface Specification	
	RW BLE Heart Rate Profile (HRP) Interface Specification	
	RW BLE HID Over GATT Profile Interface Specification	
	RW BLE Host Error Code Interface Specification	
	RW BLE Location and Navigation Profile Interface Specification	
	RW BLE Phone Alert Status Profile Interface Specification Interface Specification	
	RW BLE Proximity Profile Interface Specification	
	RW BLE Running Speed and Cadence Profile Interface Specification	
	RW BLE Scan Parameters Profile Interface Specification	

Figure 49. Accessing RSL10 Documentation from the ON Semiconductor IDE

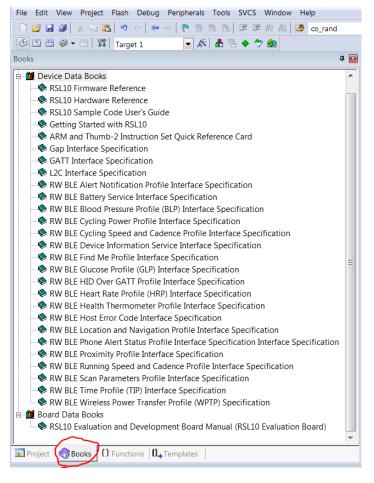


Figure 50. Accessing RSL10 Documentation from the Keil $\mu\text{Vision IDE}$

					concerning all sound	
© 1 • 1 ≠ • 1 9) • 81 • © • • • •					Quick Access	
blinky.rteconfig 🖽	- C	🙆 Packs 🕮 🔳 Devices 🛢 Boards	C* Examples	Console	@ @] 🖗 V 😅 🖮 🛞 🔻 🕾	
Device 0		Search Pack				
blinky.rteconfig 🗉	0 🖬	En actives in province	Action 1 Pack	Console Description RSLD selected ON Semiconductor RSL10 Device Family Pack Software Packs with generic content not specific to a device		
WW BLE Running Speed and Cadence Profile Interface Specification WW BLE Scan Parameters Profile Interface Specification WW BLE Miles Proven Transfer Profile (WPTI) Specification WW BLE Wireless Power Transfer Profile (WPTI) Specification						
ompatible boards:						

Figure 51. Accessing RSL10 documentation from the IAR Embedded Workbench

For more information, see the following:

Arm and Thumb®-2 Instruction Set Quick Reference Card

From the Arm company, this quick reference card provides a short-hand list of instructions for the Arm Cortex-M3 processor.

RSL10 Evaluation and Development Board Manual

This document actually contains a link to the manual that is stored elsewhere on the website. It is a reference manual that provides detailed information on the configuration and use of the RSL10 Evaluation and Development Board. When you use this board with the software development tools, you can test and measure the performance and capabilities of the RSL10 radio SoC.

RSL10 Firmware Reference

The system firmware provides functionality that isolates you from the hardware, and implements complex but common tasks, making it easier to support and maintain your code. The Bluetooth firmware provides an implementation of the Bluetooth host, controller, and profiles, supporting the standards-compliant use of these components within your application. This manual provides a reference to both sets of firmware features, and explains how they can assist with the development of your applications.

RSL10 Hardware Reference

Describes all the functional features provided by the RSL10 SoC, including how these features are configured and how they can be used. This manual is a good place to start when you are designing real-time implementations of your algorithms, or planning a product based on the RSL10 SoC.

RSL10 Sample Code User's Guide

Explains how to use the sample applications provided with the RSL10 software development tools. You learn about setting up your system, accessing code files, and how the sample applications work, using the Peripheral Device with Server sample code as the prime example.

RivieraWaves Interface Specifications (files in the ceva folder)

Interface Specifications from RivieraWaves provide a description of the API for the specified library:

- GAP Interface Specification
- GATT Interface Specification
- Host Error Code Interface Specification
- L2C Interface Specification
- RW BLE Alert Notification Profile Interface Specification
- RW BLE Battery Service Interface Specification
- RW BLE Blood Pressure Profile (BLP) Interface Specification
- RW BLE Cycling Power Profile Interface Specification
- RW BLE Cycling Speed and Cadence Profile Interface Specification
- RW BLE Device Information Service Interface Specification
- RW BLE Find Me Profile Interface Specification
- RW BLE Glucose Profile (GLP) Interface Specification
- RW BLE HID Over GATT Profile Interface Specification
- RW BLE Heart Rate Profile (HRP) Interface Specification
- RW BLE Health Thermometer Profile Interface Specification
- RW BLE Location and Navigation Profile Interface Specification
- RW BLE Phone Alert Status Profile Interface Specification
- RW BLE Proximity Profile Interface Specification
- RW BLE Running Speed and Cadence Profile Interface Specification
- RW BLE Scan Parameters Profile Interface Specification
- RW BLE Time Profile (TIP) Interface Specification
- RW BLE Wireless Power Transfer System Profile Interface Specification

LPDSP32 Documentation

The following documents are available in the RSL10_LPDSP32_Support.zip file:

- RSL10 Getting Started with the LPDSP32 Processor, which provides an overview of the techniques involved when writing and integrating code for the LPDSP32 processor that is on **RSL10**.
- LPDSP32-V3 Block Diagram, which provides a drawing of all the inputs, outputs, components and process blocks
- LPDSP32-V3 Hardware Reference Manual, which describes the hardware aspects of the LPDSP32-V3 core and its operations to provide an understanding of the core architecture and various kinds of supported operations.
- LPDSP32-V3 Interrupt Support Manual, which describes how interrupts are supported.

• User Guide IP Programmers for LPDSP32-V3, which describes the C application layer, the flow generally followed when any application is ported to LPDSP32, various tips for optimization to make the best use of the processor and compiler resources, and certain things the programmers should be aware of when porting applications. It also provides a few examples to show the usage of LPDSP32 intrinsic functions and to give an idea of how certain DSP functions can be ported to and optimized for LPDSP32.

RSL10 Release Notes

Lists new features in the latest release and known issues. This file is downloaded with the installer in a zip file, and is not in the *documentation* folder.

7.2.2 Documentation in the documentation.zip File

You can access documentation through the *documentation.zip* file available with this release of RSL10. It contains all of the documents included with the CMSIS-Pack as well as the following:

Getting Started with RSL10 Bluetooth Low Energy Mesh

Helps you to get started with the RSL10 mesh package. It guides you through the process of installing the mesh package alongside the RSL10 SDK, configuring your environment, and building and debugging your first RSL10 mesh network.

RSL10 Bluetooth Low Energy Mesh Sample Code User's Guide

Shows you what the mesh sample application (*ble_mesh*) demonstrates, how to configure the project to set up different mesh network scenarios, and how to experiment with them to verify their features and operations.

Files in the mindtree folder (related to Bluetooth Low Energy Mesh networking)

- EtherMind_Mesh_API.chm
- EtherMind_Mesh_Application_Developer's_Guide_Generic.pdf
- *EtherMind_Mesh_CLI_User_Guide.pdf*

RSL10 Bootloader Guide

The RSL10 bootloader provides means of performing firmware updates using the UART interface, and is a required component for Firmware Over the Air (FOTA). The bootloader enables firmware updates without the use of the JTAG interface. Firmware can be loaded from a host microcontroller over UART or over the air from another wireless device using FOTA. The bootloader copies the firmware image to the designated location in flash memory. This document describes the bootloader firmware application and development tools.

RSL10 Firmware Over-The-Air User's Guide

This manual describes Firmware Over-The-Air (FOTA) with RSL10. It provides the prerequisites and instructions necessary to develop FOTA-ready firmware applications and to perform FOTA updates in the field.

RSL10 LPDSP32 Support Manual

Provides an overview of the techniques involved when writing and integrating code for the LPDSP32 processor included with the RSL10 radio System-on-Chip (SoC).

RSL10 Getting Started with the LPDSP32 Processor

Provides an overview of the techniques involved when writing and integrating code for the LPDSP32 processor that is on RSL10.

Manuals in the lpdsp32 folder:

- *LPDSP32-V3 Block Diagram*: provides a drawing of all the inputs, outputs, components and process blocks
- *LPDSP32-V3 Hardware Reference Manual*: Describes the hardware aspects of the LPDSP32-V3 core and its operations to provide an understanding of the core architecture and various kinds of supported operations
- LPDSP32-V3 Interrupt Support Manual: Describes how interrupts are supported
- User Guide IP Programmers for LPDSP32-V3: Describes the C application layer, the flow generally followed when any application is ported to LPDSP32, various tips for optimization to make the best use of the processor and compiler resources, and certain things the programmers should be aware of when porting applications. It also provides a few examples to show the usage of LPDSP32 intrinsic functions and to give an idea of how certain DSP functions can be ported to and optimized for LPDSP32.

RSL10 Stand Alone Flash Loader Manual

Provides the information that you need to use the stand-alone flash loader. It describes the operations that the flash loader can perform, and explains how to configure the flash loader to connect to an RSL10 radio IC. The stand-alone flash loader is used to program, erase and read flash memory in RSL10.

RSL10 Release Notes History

A zip file containing the release notes for previous releases.

APPENDIX A

Migrating to CMSIS-Pack

If you have an existing project and have not used the RSL10 CMSIS-Pack before, this section is for you. Starting from SDK 3.0, the RSL10 firmware is no longer bundled with the Eclipse IDE. The RSL10 Eclipse IDE has been optimized and rebranded as the ON Semiconductor IDE, and the RSL10-specific firmware is now delivered exclusively as a separate CMSIS-Pack that can be imported into the IDE. For future RSL10 releases, you only need to download and import the updated CMSIS-Pack. There is no need to re-install the Eclipse IDE if it has not been updated.

Existing Eclipse project files from previous SDK releases are not compatible with the new ON Semiconductor IDE. Fortunately, migrating your existing project into the new IDE to take advantage of the CMSIS-Pack standard is a straightforward process, as shown in the next section.

A.1 MIGRATING AN EXISTING ECLIPSE PROJECT TO THE CMSIS-PACK METHOD

In order to tell whether your project is managed by CMSIS-Packs, check that a file with the *.rteconfig* extension is present in the project folder. If not, your project is not managed by CMSIS-Packs and needs to be migrated. The easiest way to migrate your existing Eclipse project to the new IDE is to start from one of the CMSIS-Pack RSL10 sample projects, and follow these steps:

- NOTE: This section assumes you know how to import the CMSIS-Pack and a sample application, as shown in Chapter 3, "Getting Started with the Eclipse-Based ON Semiconductor IDE" on page 7.
- 1. Decide on which CMSIS-Pack sample project to import. It is best to import a CMSIS-Pack project that looks similar (in terms of libraries used) to the existing project you would like to migrate. For example, if your existing application uses the Heart Rate Profile, you might want to import the *ble_peripheral_server_hrp* sample application as a reference.
- 2. Right-click the project and rename it as you wish.
- 3. Remove the source code from the sample project.
- 4. Copy over the source and header files from your existing project into the new one.
- 5. Open the RTE Configuration Wizard by double-clicking the *.rteconfig* file, and make sure all the software components (libraries) required for your project are selected.
 - Pay special attention to the Bluetooth components, such as the Bluetooth Low Energy Stack, Kernel, and Profiles. Ensure that these components have the correct variants selected (such as *release_light*, or *release_hci*).
 - Some libraries might have been removed, such as the *weakprf.a*. This library has been replaced by the *stubprf.c* file that is automatically added together with the Bluetooth Low Energy Stack component, so you no longer need to explicitly reference it.
 - You can also remove (deselect) the software components that you do not need in your existing application.
 - If you change the *.rteconfig* file, make sure to save it, so that it can update your project settings automatically (such as the library paths, includes, etc.) to reflect the newly added or removed software components.
- 6. Navigate to your project settings and add or remove the preprocessor *symbol* or *include* folders from your existing project.
- 7. Build your application and make sure it builds correctly.
 - In case of build errors related to missing components, files, or preprocessor symbols, go back to steps 5 and 6 and review your configuration carefully.
 - If you encounter errors related to duplicated code, review the *RTE* folder in your application. Some files that were common to multiple sample applications have been transformed into software components, such as the BLE Abstraction, CMSIS-Drivers, etc.

For errors related to deprecated code or API changes, review the latest RSL10 CMSIS-Pack release notes ٠ and check to see if there are any feature changes that could affect your project.

A.2 USING THE LATEST RSL10 FIRMWARE IN A PREVIOUS VERSION OF THE ECLIPSE-BASED IDE

We recommend always updating your installation to the latest version of the Eclipse-based ON Semiconductor IDE. However, if your circumstances are such that this is impractical, you can manually update the RSL10 firmware files in a previous version of the Eclipse-based IDE. If this is your case, try the following steps:

- 1. Download the RSL10 SDK CMSIS-Pack from www.onsemi.com/RSL10 and save it in any temporary folder.
- 2. Use a compressing tool, such as 7-Zip, and extract the contents of the ONSemiconductor.RSL10.version.pack file.
- 3. Copy and replace the *lib* and *include* folders from the CMSIS-Pack into your existing RSL10 SDK Installation folder.
- 4. Clean and build your application. If the build has been successful, you can see that it now references the updated libraries and include files.

In case of build errors, make sure to review the latest release notes from the CMSIS-Pack and check to see if there are any features or bug fixes that affect your application.

APPENDIX B

Arm Toolchain Support

There are several ways in which the ON Semiconductor IDE determines which Arm GNU toolchain to use when building. Understanding how this works can help prevent confusion and frustration, when the development machine has several versions of GNU toolchains installed.

B.1 BASIC INSTALLATION

The ON Semiconductor IDE supports the Arm toolchain by installing it in the *arm_tools* directory within the installed RSL10 software tools location. The build tools RM and Make are also included with the toolchain, to allow for an easier building experience out of the box.

When the user starts the ON Semiconductor IDE with the *IDE.exe* program (whose shortcut is located in Windows menu items), the *arm_tools\bin* directory is added to the path, to give the ON Semiconductor IDE access to the toolchain installed with the RSL10 software tools.

Conflicts with toolchain versions can occur in the ON Semiconductor IDE, if an Arm-based toolchain has been installed elsewhere or already exists on the path, and the IDE selects that toolchain rather than the one included in *arm_tools*.

B.2 CONFIGURING THE ARM TOOLCHAIN IN THE ON SEMICONDUCTOR IDE

All toolchain location options can be accessed by right clicking on the project in the **Project Explorer** view, selecting **Properties** at the bottom of the pop-up menu, and choosing the **Toolchains** tab. The scope of the toolchain path support is described below.

Global Path:	This is the path used by all workspaces/projects. The global path can be set in the Toolchains tab of the project.
Workspace Path:	This is the path used by all projects in the current workspace.
Project Path:	This is the path used by the current project for its toolchain.

B.3 ADDITIONAL SETTINGS

Additional settings (other than the toolchain paths) are located within the MCU preference. These are:

- The Build Tools path (global, workspace, project-based) for tools such as Make and RM
- The Segger JLink path (global, workspace, project-based) for the location of the Segger JLink executables. This replaces the Run/Debug string substitutions for JLink previously used.

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. IAR and IAR Embedded Workbench are trademarks or registered trademarks of IAR Systems AB. All other brand names and product names appearing in this document are trademarks of their respective holders.

ON Semiconductor and the ON Semiconductor logo are trademarks of Semiconductor Components Industries, LLC dba ON Semiconductor or its subsidiaries in the United States and/or other countries. ON Semiconductor owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of ON Semiconductor's product/patent coverage may be accessed at <u>www.onsemi.com/site/pdf/Patent-Marking.pdf</u>. ON Semiconductor 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. ON Semiconductor 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.

ON Semiconductor 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 ON Semiconductor shall not constitute any representation or warranty by ON Semiconductor, and no additional obligations or liabilities shall arise from ON Semiconductor 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 ON Semiconductor 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 ON Semiconductor 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 ON Semiconductor 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.

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

LIMITATIONS OF LIABILITY: ON Semiconductor 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 ON Semiconductor is advised of the possibility for such damages. In no event shall ON Semiconductor's aggregate liability from any obligation arising out of or in connection with the board, even if ON semiconductor 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