RSL10 Getting Started Guide

AND9697/D November 2019, Rev. P7



ON Semiconductor®

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

Table of Contents

	Page
1. Introduction	. 4
1.1 Overview	. 4
1.2 Intended Audience	
1.3 Conventions	. 4
2. Setting Up the Hardware	. 5
2.1 Prerequisite Hardware	. 5
2.2 Connecting the Hardware	. 5
2.3 Preloaded Sample	. 5
3. Getting Started with the Eclipse-Based ON Semiconductor IDE	
3.1 Software to Download	
3.2 ON Semiconductor IDE and RSL10 CMSIS-Pack Installation Procedures	
3.3 Building Your First Sample Application with the ON Semiconductor IDE	
3.3.1 Launching the ON Semiconductor IDE	. 9
3.3.2 Importing the Sample Code	. 9
3.3.3 Build the Sample Code	11
3.4 Debugging the Sample Code	.13
3.4.1 Debugging with the .elf File	
3.4.2 Peripheral Registers View with the ON Semiconductor IDE	15
4. Getting Started with Keil	
4. Oretang Stated with Ken	
4.1 Relequisite Software	
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	
5. Getting Started with IAR	
5.1 Prerequisite Software	
5.2 RSL10 CMSIS-Pack Installation Procedure	
5.3 Building Your First Sample Application with the IAR Embedded Workbench	
5.3.1 Import the Sample Code	
5.3.2 Building the Sample Code.	
5.3.3 Debugging the Sample Code	29
5.3.3.1 Debugging Applications	29
6. Resolving External CMSIS-Pack Dependencies	.32
1 External CMSIS-Pack Dependencies	.32
2. Resolving External Dependencies	
7. Advanced Debugging	.35
7.1 Printf Debug Capabilities	.35
7.1.1 Adding Printf Debug Capabilities	
7.2 Debugging Applications that Do Not Start at the Base Address of Flash	
7.3 Arm Cortex-M3 Core Breakpoints	

7.4 Debugging with Low Power Sleep Mode 3 7.4.1 Downloading Firmware in Sleep Mode 4	
8. More Information	4
8.1 Folder Structure of the RSL10 CMSIS-Pack Installation	4
8.2 Documentation	5
8.2.1 Documentation Included with the CMSIS-Pack	5
8.2.2 Documentation in the RSL10 Documentation Package	9
A. Migrating to CMSIS-Pack	51
A.1 Migrating an Existing Eclipse Project to the CMSIS-Pack Method	51
A.2 Using the Latest RSL10 Firmware in a Previous Version of the Eclipse-Based IDE	52
B. Arm Toolchain Support	;3
B.1 Basic Installation	
B.2 Configuring the Arm Toolchain in the ON Semiconductor IDE	
B.3 Additional Settings	;3

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 51 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. Android and iOS mobile apps are available on their respective app stores to demonstrate and explore RSL10 features.

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, 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.

Setting Up the Hardware

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.

2.3 PRELOADED SAMPLE

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

RSL10 Getting Started Guide

- "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.

Getting Started with the Eclipse-Based ON Semiconductor IDE

3.1 SOFTWARE TO DOWNLOAD

- 1. Download the ON Semiconductor IDE Installer from.<u>www.onsemi.com/RSL10</u>.
- Download the RSL10 Software Package from <u>www.onsemi.com/RSL10</u> and extract the RSL10 CMSIS-Pack (ONSemiconductor:RSL10.<version>.pack) to any temporary folder. (The temporary folder can be on any drive on your computer.)

3.2 ON SEMICONDUCTOR IDE AND RSL10 CMSIS-PACK INSTALLATION PROCEDURES

You can uninstall any previous version of the ON Semiconductor IDE by going to the Windows **Control Panel** and uninstalling **ON Semiconductor IDE** or **RSL10 Development Tools** (depending on how old your previous version is). Optionally, you can skip the uninstallation process and install the new version while keeping the old version(s) available for use.

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_V*<version>.

- You are prompted to install SEGGER J-Link. You need the J-Link software to download and debug applications on the Evaluation and Development Board. The J-Link Installation Check screen will guide you through the process of installing J-Link if no valid J-Link installation is found.
- 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. It is important to create a new workspace for each new version of the IDE to ensure compatibility. Create a new workspace at, for example, *c:\workspace* using either Windows Explorer or the ON Semiconductor Launcher in step 2.
- Open the ON Semiconductor IDE by going to the Windows Start menu and selecting ON Semiconductor > ON Semiconductor IDE. From the ON Semiconductor IDE Launcher screen, browse to your new workspace, select it, and click Launch.
- 3. On the top right corner of the Workbench perspective, click on the Open Perspective icon, select CMSIS Pack Manager, and click Open (see Figure 2).
- NOTE: If you cannot see the **CMSIS-Pack Manager** item, re-install the IDE in your user folder (i.e., C:\Users\<user_name>).





Figure 2. 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 3.1.575, and click **Open** (see Figure 3).

	C Import Packs		×
	← → ▼ ↑ 📕 « Program Files (x86) > RSL10_Software_Package V 🖏 Search RSL10	D_Software_Packa	. P
≝ 🕜 i ½ マ 🖓 マ 🖏 マ ⇔ マ	Organize 🔻 New folder		?
•	Firmware_referer ^ Name ^ Date modified	Туре	5
⊞ □ Ø ♥ ♥ B ≥ ♥ ♥ Ø ▼ □ Import Existing Packs	Getting Started ONSemiconductor.RSL10.3.1.575.pack 2019-09-23 5:25 PM ONSemiconductor.RSL10.3.1.575.pack 2019-09-23 5:25 PM Dropbox	PACK File	
r. Please import your CMSIS-Pack.	OneDrive Kenichi Kenichi Kenichi	_	>
	File name: ONSemiconductor.RSL10.3.1.575.pack Pack Files (*. Open	pack) Cancel	~ I

Figure 3. 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 4 illustrates what the Pack Manager perspective looks like after installation.

tests was see	Project Run Window Help	10 P			
< 0 <	V No Launch Configur	rations 🗸 on:	~ ☆	☆ • 🗑 🕼 🖻 & & 2 🐸 🔤 💕 💁 • 🖨	<i>4</i> 9 −
					Quick Access 🗄 🛃 🛙 🖬 隆
📕 Devices 🖾 🗾 Boards	⊞ ⊟ 🐐 🕐 🗢 🗖	谢 Packs 🕴 📑 Examples		🖽 🖻 🤣 🍣 🐸 🎦 🍞 🔍 🖘 🗖	😑 Pack Properties 🛛 🖓 🗖
Search Device		Search Pack			
Device • * All Devices • • ONSemiconductor		Pack Pack Pevice Specific Pevice Specific Period		Description RSL10 selected ON Semiconductor RSL10 Device Family Pack	type filter text
 ✓ [®] RSL10 Series ■ RSL10 	1 Device ARM Cortex-M3 48 MHz, 24 KI	# 3.1.575 • Generic	Remove	Release notes available at www.onsemi.com/rsl10 Software Packs with generic content not specific to	 > Components > ■ Devices > ■ Examples
<	>	<		>	<
Q Console Progress CMSIS Console		2			⊾∎®∣≝ ¤ + ri + = c

Figure 4. Pack Manager Perspective after RSL10 CMSIS-Pack is Installed

3.3 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.

3.3.1 Launching the ON Semiconductor IDE

Open the ON Semiconductor IDE by going to the Windows Start menu and selecting **ON Semiconductor** > **ON Semiconductor IDE**.

3.3.2 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 5).

RSL10 Getting Started Guide

< O <	No Launch Configurations	✓ on:): 🗝 🖵 🖂 🖂	💷 🕹 🦑 ≷ 🐸 🔐 💕 💁 🕇 🕞 /	0 🗸
 ₩ • ₩ • Φ • • • •	The Earlieft Conlightations				Quick Access
📕 Devices 😂 📕 Boards	▣ □ ▼ □	A Packs Sector State S			Pack Properties
Search Device		Only show exam	ples from insta	lled packs 🖑 ぞ 🐸 🔐 🥑 🤍	œ I
Device	Summary	Search Example			type filter text
✓ ⁴ All Devices	1 Device	Example	Action	Description ^	v 🖶 ONSemicond
 ONSemiconductor 	1 Device	ADC_UART (RSL10 Evaluation Board)		ADC with UART Sample Code	> 🗾 Boards
v ⁴ RSL10 Series	1 Device	aes128 (RSL10 Evaluation Board)	Copy	AES128 - Simple AES-128 Sample	> 🚸 Compone
RSI 10	ARM Cortex-M3 48 MHz, 24 KB RAM, 3	ble android asha (RSL10 Evaluation Board)		Android Audio Streaming Hearing	> E Devices
NDE 10		ble central client bond (RSL10 Evaluation)	100	BLE Central Client Bonding Samp	> 📑 Examples
		ble_central_client_bond (RSE10 Eval)		Central Device with Client Scanne	
		 ble_central_crient_scan (RSL10 Evalu ble central peripheral (RSL10 Evalu 		Central Peripheral Device Sample	
		 ble_peripheral_server_bond (RSL10 		BLE Peripheral Server Bonding Sa	
		 ble_peripheral_server_bond (RSE10 ble_peripheral_server_hrp (RSL10 E) 		Heart Rate Peripheral Device with	
		 ble_peripheral_server_hrp_fota (RSL 		Heart Rate Peripheral Device with	
		 ble_peripheral_server_IIIp_Tota (RSL10 E 		Private resolvable Address with St	
		blinky (RSL10 Evaluation Board)	Copy	Blinky GPIO I/O Sample Code	
		 bootloader (RSL10 Evaluation Board) 		Bootloader Sample Code	
		 central client uart (RSL10 Evaluation Board) 		Central Device with Client UART S	
		 custom protocol trx (RSL10 Evaluate 		Low Latency Audio Sample Appli	
		default MANU INFO INIT (RSL10 E	AND ROOM AND AND	Default System Initialization Func	
		dma driver (RSL10 Evaluation Boar		DMA Driver Sample Code	
		DMIC OD (RSL10 Evaluation Board)		DMIC and OD Sample Code	
		flash copier and crc (RSL10 Evaluation Board)		Flash Copier and CRC Sample Coc	
		freertos_ble_central_client_bond (R		BLE Central Client Bonding Sample	
			1		
		••••••••••••••••••••••••••••••••••••••		and a state of a second se	
¢	>	<		>	<
 Console ⅔ ➡ Progress CMSIS Console 	>	freetos_ble_peripheral_server_bon	1	BLE Peripheral Server Bonding Sal	< 🖬 🖭 🖻 🗉

Figure 5. 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 6).

le Edit Source Refactor Naviga	No Launch Configurations		v on:		v Alte			4 . 8 . 7 . 0	• • • • • • • • • • • • • • • • •			
hour benefat benefat 1	11		on in		* white	3 • 03 00 1 @ •						
FⅢ112・Ⅱ・ΦΦ・Φ・									Quick Access			
Project Explorer 😂 🧧 🗖	♦ blinky.rteconfig 88								🖁 Out., 🕴 🗑 Tes., 🛞 Bui., 🛸 Do.	-		
E 😵 🔛 🗡	Components SR											
 blinky blinkudes blincludes 	Software Components	Sel	. Variant	Vendor ONSemicondu	Version	and the second second	3 48 MHz, 24 K8 RAM, 38	IS KE ROM	There is no active editor that provides outline.	an		
> S RTE	>											
> R app.c	v 🕈 Device											
blinky.rteconfig	> 🎐 Bluetooth Core	Q.			1.0.0	BLE Core imple	mentation					
readme_blinky.md	Bluetooth Profile	e										
	 Libraries 											
							ary (libcalibratelib)					
	Custom Prot						idio Streaming Custom P	rotocol Library (libc				
			source			DMA Driver for						
	Flash					Flash Library (li						
	GPIO		source			GPIO Driver for						
	Math		source			Math Library (li						
							hone Library (libremote	micLib)				
	System						and Library (libsyslib)					
	Startup	M	source	ONSemicondu	: 3.1.5/5	System Startup	for ON Semiconductor F	SL10				
	Validation Output			De	scription							
	Components Device Pack	5										
	🛃 Problems 🕮 🎒 Tasks	Co	nsole 🖾 Prope	erties					∋ v	0 0		
	0 items	~		2.0	10/212	2	1	1				
	Description			Reso	ource	Path	Location	Туре				

Figure 6. RTE Configuration for the Blinky Example Project in the ON Semiconductor IDE

3.3.3 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 Build Project icon, which looks like a hammer, as shown in Figure 7.

s bl	olorer 🛛 🔍 🗘 bl	inkv.rteconfig 🖾											
8		induction in g							➢ Out ∺	🛙 Tas 🧕	9 Bui 🏌	1 Do	-
	New	>	e					0	There is no a	rtive edito	r that nr	wides as	21
6	Go Into		el. Variant	Vendor	Version				outline.	cuve conto	a court pre	macs a	
	Open in New Window			ONSemiconduc		ARM Cortex-M3 48 MHz	, 24 KB RAM, 388	KB ROM					
2	Show in Local Terminal	>											
•	Сору	Ctrl+C			1.0.0	BLE Core implementatio	•						
•	Paste	Ctrl+V			1.0.0	BLE Core implementatio	n						
	Delete	Delete											
		Ctrl+Alt+Shift+Down	source	ONSemiconduc	3.1.575	Calibration Library (libca	libratelib)						
	Source Move	,	source			Low Latency Audio Street		tocol Library (libc					
	Rename	F2	source	ONSemiconduc	3.1.575	DMA Driver for RSL10							
		F4	source			Flash Library (libflashlib)							
	Import Export		source			GPIO Driver for RSL10							
			source			Math Library (libmathlib							
1	CMSIS C/C++ Project	>	source			Remote Microphone Lib		cLib)					
	Build Project Clean Project		source			System Macros and Libra System Startup for ON S		10					
	Refresh	F5	source	UNSemiconduc	3.1.575	_ System Startup for ON S	emiconductor KSL	10					
	Close Project	10		Des	cription								
	Close Unrelated Project												
	Build Targets	>											
	Index	Ś											
	Build Configurations	, ,											
	Profiling Tools Run As	>	onsole 🔲 Pro	nation							-14	1 20 1	
1	Debug As	· · · · · ·	onsole EPro	perues							-11		
	Profile As	>		0		Death	t a set la s	×					
-1	Restore from Local History			Reso	urce	Path	Location	Туре					
nky 3	Run C/C++ Code Analysis												

Figure 7. Starting to Build a Project in the ON Semiconductor IDE

2. When the build is running, the output of the build is shown in the ON Semiconductor IDE C/C++ Development Tooling (CDT) Build Console, as illustrated in Figure 8.

 Flash Li source UNSemicor 	nduc: 3.1.575 J Hash Library (libflashlib)	> ``				
Validation Output	Description					
Components Device Packs						
🗈 Problems 🥝 Tasks 🖾 Console 👯 🗔 Properties			- 0 0 B	🛛 🗊 🐷 🖻 🕅	🖻 🔍 🕶 📑	• • •
CDT Build Console (blinky)						
Invoking: Cross ARM GNU Create Flash Image arm-none-eabi-objcopy -0 ihex "blinky.elf" " Finished building: blinky.hex	blinky.hex"					^
Invoking: Cross ARM GNU Print Size arm-none-eabi-sizeformatberkeley "blinky. text data bss dec hex filen 4204 100 1036 5340 14dc blinky Finished building: blinky.siz	ame					
13:52:29 Build Finished. $\boldsymbol{\theta}$ errors, $\boldsymbol{\theta}$ warnings	. (took 7s.976ms)					v

Figure 8. Example of Build Output

- 3. The key resulting output in Project Explorer, in the *Debug* folder, includes:
 - blinky.hex: HEX file for loading into Flash memory ٠
 - *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 9.

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 9. Output Files from Building a Sample Project

3.4 DEBUGGING THE SAMPLE CODE

3.4.1 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 Configuration**. 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 Name** field. Ensure that **SWD** is selected as the target interface (as shown in Figure 10).

₿ 🏚 🗎 🗶 🗐 🏂 🗸	Name: blinky									
vpe filter text	📄 Main 🕸 Debugg	er 🕨 Sta	rtup 🦻 Sour	e 🗖 Com	mon 🛃 SVD P	ath				
C/C++ Application	J-Link GDB Server S	etup								
C/C++ Attach to Application	Start the J-Link	Start the J-Link GDB server locally								
C/C++ Postmortem Debugge C/C++ Remote Application	Executable path:	\${jlink_path}/\${jlink_gdbserver}					Browse	Variables		
	Actual executable:	C:/Progra	m Files (x86)/	SEGGER/JLi	ink/JLinkGDBS	erverCL.exe				
Cti C/C++ Unit GDB Hardware Debugging		(to change	e it use the glo	bal or wor	kspace prefere	nces pages or th	ne <u>project</u> pro	perties page)		
GDB Hardware Debugging	Device name:	RSL10					Supported of	device names		
GDB SEGGER J-Link Debuggir	Endianness:	Little	OBig							
C blinky	Connection:	O USB	O IP ↓		(USB serial or IP I	name/address)		
Launch Group Launch Group (Deprecated)	Interface:	● SWD	◯ JTAG							
Launch Group (Deprecated)	Initial speed:	O Auto	O Adaptiv (Fixed	1000 kHz					
	GDB port:	2331								
	SWO port:	2332			🔽 Verify	downloads 🗸	Initialize regi	sters on start		
	Telnet port:	2333			🗹 Local	host only	Silent			
	Log file:							Browse		
	Other options:	-singleru	n -strict -time	out 0 -noqu	i					
	Allocate consol					console for semi	hosting and S	WO		
	GDB Client Setup						-			
	abb chem betup									
	Executable name:	Sicross p	refiv)adb\$/cm	acc cuffix)			Browse	Variables		

Figure 10. Setting Up a GDB Launch Configuration, Debugger Tab

- NOTE: To debug an application that does not start at the first address of flash memory, see Chapter 7, "Advanced Debugging" on page 35.
- 4. Once the updates to the configuration are completed, make sure that 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 a compatible J-Link version (see Section 3.2, "ON Semiconductor IDE and RSL10 CMSIS-Pack Installation Procedures" on page 7).

If you are having trouble downloading firmware because an application with Sleep Mode is on the Evaluation and Development Board, see Section 7.4.1, "Downloading Firmware in Sleep Mode" on page 43.

- 5. You are prompted to switch to the debug perspective. Click Switch.
- 6. The Debug perspective opens and the application runs to *main*, as shown in Figure 11. 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).

on-semiconductor-workspace - blinky/app.c - ON Semiconductor IDE	_	o x
<u>F</u> ile <u>E</u> dit <u>S</u> ource Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp		
🔨 漆 🔳 株 Debug 🗸 E blinky 🗸 🔅 🗋 🖷 🗟 🗟 🗐		
🔌 🕪 🗉 🛢 🖉 🌫 👁 🕼 🖬 🗮 沈 🕹 🎸 🗸 🔿 🗣 🎧 🖉 🏈 🖬 🌽 🐲 🌾 🗘 🗸	⇒ -	
Οι	uick Access 🔡 📑	lia 🛞 🔅
🌟 Debug 🕱 隆 Project Ex 🖳 🗋 🚸 blinky.rteconfig 🔀 app.c 💥 🖬 0x0	3 »	
Int main(void)	* <u>*</u>	
✓ C blinky cobs scotck y-tink bebuggin	1	2
v m Thread #1 57005 (Suspended: 110 led toggle status = 1;	\bigtriangledown	
■ main() at app.c:124 0x10040 111	Nam	ie
JLinkGDBServerCL.exe 112 /* Initialize the system */ 113 Initialize();		
arm-none-eabi-gdb		
Semihosting and SWV 115 /* Spin loop */		
116 while (1) 117 {		
118 /* Refresh the <u>watchdog</u> timer */		
<pre>119 Sys_Watchdog_Refresh(); 120</pre>		
121 /* Toggle GPIO 6 (if toggling is enabled) then wait	0.5 s	
122 if (led_toggle_status == 1) 123 {		
123 { \$ 124 Sys_GPIO_Toggle(LED_DIO);		
125 }	<	>
126 else 127 {		
128 Sys_GPIO_Set_Low(LED_DIO);		
129 }		
<pre>130 Sys_Delay_ProgramROM((uint32_t)(0.5 * SystemCoreClose 131 }</pre>	ск));	
132 }	¥	× .
<	>	>
📮 Console 🐹 👫 Registers 🔝 Problems 🜔 Executables 🙀 Debugger Console	e 📋 Memory	
🔳 🗶 🔆 🗟 🚮	2 🖓 🖓 🖓	🖃 🔻 📑 👻
blinky [GDB SEGGER J-Link Debugging] JLinkGDBServerCL.exe		
Breakpoint reached @ address 0x00100400 Reading all registers		^
Read 4 bytes @ address 0x00100400 (Data = 0xF7FF2006)		
		*
		>
Writable Smart Insert 124 : 1		1

Figure 11. Debug Perspective

3.4.2 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 3.4.1, "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\<version>\svd (see Figure 12). Click Debug.

Debug Configurations				
Create, manage, and run configu	ations			Ś
Image: Second Secon	Name: blinky Main * Debugger Startup SVD file (used by the peripheral registers viewer) File path: C:\Users\\ON_Semiconductor\ <user_id></user_id>	PACK\ONSemiconductor\RSL10\	\svd Browse ↓ sion>	Variables
ilter matched 12 of 12 items	<u> </u>		Revert	Apply
•			Debug	Close

Figure 12. 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. In the Peripherals window, select **DIO**. Open the Memory window to monitor the RSL10 peripheral. Read only registers are highlighted in green. You might want to drag your Memory window and place it side-by-side with your source code view (see Figure 13) to prevent the console from switching focus away from the Memory window.
- 5. 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 (in this case, from 0xF060 to 0xF020). The register turns vellow to indicate that you have activated real-time monitoring for it (see Figure 14 on page 17).

	🗸 💽 blinky Debug 🛛 🗸 🏟 🔝 🐨 🔝 🔞 🔛	1 📾 1 🕷 1 👞 10 🗰 64 🐨 75 12 1 🖬 🗠	🙁 🕹 🕸 * O * 💁 * 🙋 🥖	· · · · · · · · · · · · · · · · · · ·	0 - 0 - 0	Quick Access 🔡 🔯 🔯
g 🕮 🔥 Project 📍 🗖	♦ blinky.rteconfig (app.c 23) 0.0	CO+ Variables	% Breakpoints of Expressions 👪 Module	es 😤 Peripherals 🔅	1	신 🙀 티 🍸 🗎
16 1 T	84 * single interrupt event occurs for each push of		Address Description			
linky Debug (GDB SEGGER J-Link)	85 The <u>debounce</u> circuit always has to be used in co 86 transition mode to deal with the <u>debounce</u> circuit		0x40001500 Baseband Cor	ntroller		
blinky.elf	87 * A debounce filter time of 50 ms is used. */	TE Haltations.		ntroller Interface		
P Thread #1 57005 (Suspended	88 Sys_DIO_Config(LED_DIO, DIO_MODE_GPIO_OUT_0);	25 CU	K 0x40000100 Clock Generat	tion		
main() at app.c:130 0x1008	<pre>39 Sys_DIO_Config(BUTTON_DIO, DIO_MODE_GPIO_IN_@ DIO 90 DIO_UPF_DISABLE);</pre>					
JLinkGDBServerCL.exe	91 Sys DIO IntConfig(0,	7,00		oller		
arm-none-eabi-gdb Semihosting and SWV	92 DIO_EVENT_TRANSITION DIO_SRC(B	UTTON_DIO)				
administring and arrest	93 94 DIO DEBOUNCE ENABLE, 94 DIO DEBOUNCE SLOWCLK DIV1024, 49);		and Digital Pad contr ler Configuration and		
	<pre>95 NVIC_EnableIRQ(DIO0_IRQn);</pre>		A CHECODOLO DIMA CONDOL	ier conriguration and	Control	
	96					
	<pre>97 printf_init(); 98</pre>	Memory	11			• • • • • • BE Outline 😫 • •
	99 /* Unmask all interrupts */	Monitors 🧄	💥 🧏 🔀 DIO: 0x40000700 🔅 🔶 🔶 New	Renderings		> □ 15, N, x ² + W
	<pre>100set_PRIPASK(PRIMASK_ENABLE_INTERRUPTS);</pre>	DIO	Register	Address	Value	app.h
	101) 102	CLK	✓ 5. DIO	0x40000700		1 printf.h
	1038 /*		> ttt DIO_CFG[]	0x40000700		 DIO0_IRQHandler(void) : vo
	184 * Function : int main(void)		V IIII DIO DATA	0x40000740	0.0000F060 - 1111 0000 ([]10 0000	 Initialize(void) : void
	105 * 106 * Description : Initialize the system, then toggle (Not as controlled by	0 DIO	[15:0]	0.4060	 main(void) : int.
	107 * DIOS (press to toggle input/output).		St GPIO	[15:0]	0xF060	
	108 * Inputs : None		> ### DIO_DIR	0x40000744	0x00008040	
	109 " Outputs : None 110 " Assumptions : None		> IIIT DIO_MODE	0x40000748	0x00000060	
	111 *		> # DIO_INT_CFG[] > # DIO_INT_DEBOUNCE	0x4000074C	0x00000111	
	1120 int main(void)		> IIII DIO_INI_DEBOONCE	0x40000750	0x0000131	
	113 { 114 /"Initialize global variables "/		> Mt DIO_SPL_SRC[]	0x40000764	00011111	
	115 led toggle_status = 1;		> IIII DIO UART SRC	0x4000076C	0x00000011	
	116 117 /" Initialize the system "/		> ###_DIO_I2C_SRC	0x40000770	0x00001111	
	<pre>117 /* Initialize the system */ 118 Initialize();</pre>		> ## DIO_AUDIOSINK_SRC	0x40000774	0x00000011	
	119 PRINTF("DEVICE INITIALIZED\n");		> ###_DIO_NMLSRC	0x40000778	0x00000030	
	120		> ### DIO_BB_RX_SRC	0x4000077C	0x00121212	
	121 /* Spin loop */ 122 while (1)		> ### DIO_88_SPI_SRC	0x40000780	0x00000012	
	123 (> IIII DIO_RF_SPI_SRC	0x40000784 0x40000788	0x00121212 0x12121010	
	124 /* Refresh the watchdog timer */		> ### DIO_RF_GPI003_SRC > ### DIO_RF_GPI047_SRC	0x40000785	0x12121010	
	<pre>125 Sys_Watchdog_Refresh(); 126</pre>		> IIII DIO RF GPIOR9 SRC	0x40000790	0x00001010	
	127 /* Toggle GPIO 6 (if toggling is enabled) then	wait 0.5 seconds */	> III DIO DMIC SRC	0x40000794	0x00001111	
	<pre>128 if (led_toggle_status -= 1)</pre>		> ### DIO_LPDSP32_JTAG_SRC	0x40000798	0x00111111	
	129 { Sys GPIO Toggle(LED DIO);		> ### DIO_ITAG_SW_PAD_CFG	0x4000079C	0x000000DD	
	131 PRINTF("LED Xs\n", (DIO->CFG[LED_DIO] & 0x	1 ? "ON" : "OFF"));	> ### DIO_EXTCLK_CFG	0x400007A0	0x00000001	
	132 }		> ### DIO_PAD_CFG	0x400007A4	0x00000000	
	133 else 134 (
	135 Sys_GPIO_Set_Low(LED_DIO);					
	136 }					
	<pre>137 Sys_Delay_ProgramROM((uint32_t)(0.5 * SystemCom 138 }</pre>	reClock));				
	130 }					

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

7. To manually change the register value, click on the Value field of the GPIO register to change the (HIGH/LOW) state of GPIO6. Figure 14 shows the view before making the change, and Figure 15 illustrates the view after making the change. You can observe that the LED (DIO6) on your board changes state.

🗞 🎄 📕 🎋 Debug	V C blinky Debug	3 .e i+ 🗟 🙁 4	🍐 🕸 = O = 🌯 = 🙋 🖨	A • 1 # 2 2 •	· 创 - 作 今 - 今 -
🛚 Debug 😢 🍋 Project 😐 🗖	blinky.rteconfig app.c (3) c 0.0	(*)= Variables 🔍 Bre	akpoints 🕂 Expressions 🛋 Mo	dules 🚼 Peripherals 🖇	2
後 🖬 🔍	34 single interrupt event occurs for each push of the pushbutton.	Peripheral	Address Description	n	
C blinky Debug (GDB SEGGER J-Link I	85 The debounce circuit always has to be used in combination with the	2, 88	0x40001500 Baseband	Controller	
v 🔐 blinky.elf	86 " transition mode to deal with the <u>debounce</u> circuit limitations. 87 " A debounce filter time of 50 ms is used. "/	BBIF	0x40001400 Baseband	Controller Interface	
v 🤌 Thread #1 57005 (Suspended	88 Sys_DIO_Config(LED_DIO, DIO_MODE_GPIO_OUT_0);	S 🖓 CLK	0x40000100 Clock Ger	eration	
main() at app.c:130 0x1006	89 Sys_DIO_Config(BUTTON_DIO, DIO_MODE_GPIO_IN_0 DIO_WEAK_PULL_UP	CRC	0x40000F00 CRC Gene	rator Control	
JLinkGDBServerCL.exe	<pre>90 DIO_LPF_DISABLE);</pre>	🗆 💑 DEBUG	OxE000EDF0 Debug Co	ntroller	
arm-none-eabi-gdb	91 Sys_DIO_IntConfig(0, 92 DIO EVENT TRANSITION DIO SRC(BUTTON DIO)	🗆 🚼 DIG	0x40000200 Reset		
Semihosting and SWV	93 DIO DEBOUNCE_ENABLE,			ace and Digital Pad contr	
	94 DIO_DEBOUNCE_SLOWCLK_DIV1024, 49);	DMA	0x40000600 DMA Con	troller Configuration and	i Control
	<pre>95 NVIC_EnableIRQ(DIO0_IRQn); 96</pre>				
	97 printf init();	Memory 23			ng ger 🗅 🛃 🚳 🖪 🖏 🖷 👻
	98 99 /* Unmask all interrupts */	Monitors 🐥 💥 🍇	🖳 DIO: 0x40000700 🙁 🔪 🔶	New Renderings	
	<pre>100set_PRIMASK(PRIMASK_ENABLE_INTERRUPTS);</pre>	DIO	Register	Address	Value
	101)	CLK	V 5 DIO	0x40000700	
	102 1030 /*		> 10 CFG[]	0x40000700	
	1030/- 104 * Function : int main(void)			0x40000740	0x0000F060
	105 *		S DIO	[15:0]	0xF060
	106 * Description : Initialize the system, then toggle DIO6 as controlled by		gpio	[15:0]	0x0: GPI00 LOW
	107 * DIO5 (press to toggle input/output). 108 * Inputs : None		> IIII DIO DIR	0x40000744	0x0: GPI00_LOW
	100 - Inputs : None		> IIII DIO_MODE	0x40000748	0x0: GPI01_LOW
	110 * Assumptions : None		> ## DIO_INT_CFG[]	0x4000074C	0x0: GPI02_LOW 0x0: GPI03 LOW
	111 *		> ### DIO_INT_DEBOUNCE	0x4000075C	0x0: GPI05_EOW
	1120 int main(void) 113 {		> IIII DIO_PCM_SRC	0x40000760	0x0: GPIOS LOW
	113 (114 /*Initialize global variables */		> M DIO_SPI_SRC[]	0x40000764	0x0: GPI06_LOW
	115 led_toggle_status = 1;		> ### DIO_UART_SRC	0x4000076C	0x0: GPI07_LOW
	116		> ###_DIO_I2C_SRC	0x40000770	0x0: GPIO8_LOW 0x0: GPIO9_LOW
	<pre>117 /* Initialize the system */</pre>		> IIII DIO_AUDIOSINK_SRC	0x40000774	0x0: GPI010_LOW
	<pre>118 Initialize(); 119 PRINTF("DEVICE INITIALIZED\n");</pre>		> IIII DIO_NMI_SRC	0x40000778	0x0: GPI011_LOW
	120		> ### DIO_BB_RX_SRC	0x4000077C	0x0: GPIO12_LOW
	121 /* Spin loop */		> ### DIO_BB_SPI_SRC	0x40000780	0x0: GPI013_LOW 0x0: GPI014_LOW
	122 uhile (1)		> IIII DIO_RF_SPI_SRC	0x40000784	0x0: GPI014_LOW
	123 { 124 /" Refresh the watchdog timer "/		> IIII DIO_RF_GPIO03_SRC	0x40000788	0x1: GPIO0_HIGH
	124 /* Refresh the watchdog timer */ 125 Sys Watchdog Refresh():		> ### DIO_RF_GPIO47_SRC	0x4000078C	0x2: GPI01_HIGH
	126 Sys_watchoog_werresh();		> ### DIO_RF_GPI089_SRC	0x40000790	0x4: GPIO2_HIGH
	127 /* Toggle GPIO 6 (if toggling is enabled) then wait 0.5 seconds */		> IIII DIO_DMIC_SRC	0x40000794	0x8: GPIO3_HIGH 0x10: GPIO4_HIGH
	<pre>128 if (led_toggle_status == 1)</pre>		> ### DIO_LPDSP32_ITAG_		0x20: GPIOS_HIGH
	129 { Sys GPIO Toggle(LED DIO);		> ### DIO_JTAG_SW_PAD_C		0x40: GPI06_HIGH
	130 Sys_GPIO_Toggle(LED_DIO); 131 PRINTF("LED %s\n", (DIO->CFG[LED_DIO] & 0x1 ? "OH" : "OFF"));		> ## DIO_EXTCLK_CFG	0x400007A0	0x80: GPIO7_HIGH
			> III DIO_PAD_CFG	0x400007A4	0x100: GPIO8_HIGH

Figure 14. Toggling RSL10 DIO Using the Peripheral Registers View: Before

RSL10 Getting Started Guide

🖌 🏘 🔳 🏘 Debug	V C blinky Debug V (2) Image: Second secon	🕫e 🗺 🗮 🙁	🎸 🕸 🕶 🗿 🕶 🍓	• 🙋 😂 🛷 • 🕖 😒 💯	• §] • ⊕ ⇔ • ⇔ •	
Debug 😢 🍋 Project 📍 🗖	🚸 blinky.rteconfig 🛛 app.c 🐹 💽 0x0 💙 🗖	(*)* Variables 💁 Brei	akpoints 🕂 Expressi	ons 🛋 Modules 🔀 Peripherals	22	
· · · · · · · · · · · · · · · · · · ·	84 * single interrupt event occurs for each push of the pushbutton.	Peripheral	Address	Description		
blinky Debug (GDB SEGGER J-Link I	85 The debounce circuit always has to be used in combination with the	□ <mark>2</mark> , 88	0x40001500	Baseband Controller		
🔐 blinky.elf	86 * transition mode to deal with the debounce circuit limitations. 87 * A debounce filter time of 50 ms is used. */	C 2 BBIF	0x40001400	Baseband Controller Interface		
✓ P Thread #1 57005 (Suspended)	88 Sys DIO Config(LED DIO, DIO MODE GPIO OUT 0);	R CLK	0x40000100	Clock Generation		
main() at app.c:130 0x1008	89 Sys_DIO_Config(BUTTON_DIO, DIO_MODE_GPIO_IN_0 DIO_WEAK_PULL_UP	CRC 2	0x40000F00	CRC Generator Control		
JLinkGDBServerCL.exe	90 DIO_LPF_DISABLE); 91 Sys DIO IntConfig(0,	DEBUG	0xE000EDF0	Debug Controller		
arm-none-eabi-gdb Semihosting and SWV	92 DIO EVENT TRANSITION DIO SRC(BUTTON DIO)	DIG	0x40000200	Reset		
Seminosting and SVVV	93 DIO_DEBOUNCE_ENABLE,	DIO	0x40000700 0x40000600	DIO Interface and Digital Pad cor DMA Controller Configuration a		
	<pre>94 DIO_DEBOUNCE_SLOWCLK_DIV1024, 49); 95 NVIC_EnableIRQ(0100_IRQn);</pre>	C 💦 DMA	0x40000600	DMA Controller Configuration a	nd Control	
	<pre>96 97 printf_init();</pre>	Memory SS			ma en 🕒 📑 📧 🖽	🝇 🖏 • ⊽ =
	98 99 /* Unmask all interrupts */	Monitors 🖕 👷 🎇	: 🔀 DIO: 0x40000700	🕄 🔶 New Renderings		
	<pre>100set_PRIMASK(PRIMASK_ENABLE_INTERRUPTS);</pre>	DIO	Register	Address	Value	
	101 }	CLK	✓ 3 DIO	0x40000700		
	1038 /*		> M DIO_CFC	6[] 0x40000700		
	104 * Function : int main(void)		V III DIO_DAT		0x0000F020 1111 0000 001	0 0000
	105 * 106 * Description : Initialize the system, then toggle DIO6 as controlled by		🚆 DIO	[15:0]	0.4F020	
	100 - Description : Initialize the system, then toggle blob as controlled 107 " DIOS (press to toggle input/output).			[15:0]	0xF020	
	108 * Inputs : None		> IIIT DIO_DIR		0x00008040	
	109 * Outputs : None 110 * Assumptions : None		> ## DIO_MO > ## DIO_INT		050000000	
	110 - Assumptions : None		> IIII DIO INT		0x00000131	
	1120 int main(void)		> IIIT DIO PCM		0:00111111	
	113 (> M DIO SPI			
	<pre>114 /*Initialize global variables */ 115 led toggle status = 1;</pre>		> IIII DIO_UART_SRC		0x00000011	
	116		> IIII DIO_U2C	SRC 0x40000770	0x00001111	
	117 /* Initialize the system */		> IIIT DIO_AUE	DIOSINK_SRC 0x40000774	0x00000011	
	<pre>118 Initialize(); 119 PRINTF("DEVICE INITIALIZED\n");</pre>		> IIII DIO_NM	LSRC 0x40000778	0x00000030	
	129		> ## DIO_BB_		0x00121212	
	121 /* Spin loop */		> ## DIO_BB_		0x00000012	
	122 while (1)		> ## DIO_RF_		0x00121212	
	123 { 124 /* Refresh the watchdog timer */		> ## DIO_RF_		0x12121010	
	125 Sys_Watchdog_Refresh();		> ### DIO_RF_		0x10101012	
	126		> IIIT DIO_RF_		0x00001010	
	127 /* Toggle GPIO 6 (if toggling is enabled) then wait 0.5 seconds */		> ## DIO_DM		0x00001111	
	128 if (led_toggle_status 1) 129 {			SP32_JTAG_SRC 0x40000798	0x00111111	
	129 Sys GPIO Toggle(LED DIO);			G_SW_PAD_CFG 0x4000079C	0x000000DD 0x00000001	
			> IIII DIO_EXT	CLK_CFG 0x400007A0		

Figure 15. Toggling RSL10 DIO Using the Peripheral Registers View: After

Getting Started with Keil

4.1 PREREQUISITE SOFTWARE

- 1. Download and install the Keil µVision IDE from the Keil website, using the vendor's instructions.
- Download the RSL10 Software Package from <u>www.onsemi.com/RSL10</u> and extract the RSL10 CMSIS-Pack (ONSemiconductor.RSL10.<version>.pack) to any temporary folder.

4.2 RSL10 CMSIS-PACK INSTALLATION PROCEDURE

To install the RSL10 CMSIS-Pack:

 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
	📬 🗖		X 🖬 🕻	5 9	@	⇒ ¶ ? *	图图		٩Ļ
٢		i 🗼 •		Targ	et 1	- *	📥 🗟	5 🔶 😚	• 💑

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.

	Marks Karal Strategy
	Computer
	Organize 🔹 New folder
	Favorites Name Date modified Type
<i>a</i>	Desktop ONSemiconductor.RSL10.2.2.347 10/31/2018 4:29 PM uVision Downloads
	Recent Places
Refresh Import Import from Folder Manage Local Repositories Exit Summary Frit Summary B	 ☐ Libraries ③ Documents ④ Music ④ Pictures ⑤ Videos
	Scomputer
	📲 SYSTEM (C:) 🗸 K 🔢 🖓
	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 %LOCALAPPDATA%\Arm\Packs folder.
- 4. After installation, use File > Refresh as shown in Figure 18 to update your pack proprieties.

8	Pack Installer - C:\Keil_v5\ARM\PA	СК
File	Packs Window Help	
	Refresh	
	Import	
	Import from Folder	Pack
	Manage Local Repositories	Pack
	Fxit	Summary

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 illustrates what the Pack Installer perspective looks like after installation.

Device: ONSemiconductor - RSL10							
Devices Boards		4	Packs Examples				
Search: • 🗙 🕒	12 d		Show examples from installed Packs or				
Device	/ Summary		Example	Action	Description		
🗉 🤗 Maxim	9 Devices	<u> </u>	ADC_UART (RSL10 Evaluation Board)	🔶 Сору	ADC with UART		
🗉 🔗 MediaTek	2 Devices		ble_central_client_bond (RSL10 Eval	< Сору	BLE Central Clier		
🗉 🤗 Microchip	345 Devices		ble_central_client_scan (RSL10 Eval	🔶 Сору	Pairing and Bon		
Microsemi	6 Devices		ble_peripheral_server_bond (RSL10	🔶 Сору	BLE Peripheral S		
MindMotion	2 Devices		ble_peripheral_server_hrp (RSL10 Ev	🔶 Сору	Pairing and Bon		
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		
	1223 Devices		-hci_app (RSL10 Evaluation Board)	🚸 Сору	Host Controller		
ONSemiconductor	1 Device			🔶 Сору	I2C CMSIS-Drive		
🖻 🍄 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 Devic		
🗉 🔗 Renesas	4 Devices		spi_cmsis_driver (RSL10 Evaluation	🔶 Сору	SPI CMSIS-Drive		
🗉 🔗 Silicon Labs	783 Devices		supplemental_calibrate (RSL10 Eval	🚸 Сору	Default System I		
Sinowealth	1 Device		uart_cmsis_driver (RSL10 Evaluation	🚸 Сору	UART CMSIS-Dr		
🗉 🔗 SONIX	50 Devices						
STMicroelectronics	1061 Devices						
🗉 🔗 Texas Instruments	350 Devices	-	•				
Dutput			(F				
pdate available for Keil::ARM_Compiler (inst	alled: 1.3.3 available: 1.4.0)						



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

🖨 Workspace_2019-11-18 -	ON Semiconductor IDE	
File Edit Navigate Search	Project Run Window Help	
 Image: Second sec	V No Launch Configurations	✓ on:
Devices Boards	⊞ ⊟ 💥 💿 ⊽ 🗖 🗖	Macks ■ Examples X
Search Device		Only show examples from i
Device	Summary	Search Example
✓ [№] All Devices	1 Device	Example Action
 ONSemiconductor 	1 Device	ADC UART (RSL10 Evaluation Board)
🗸 🏄 RSL10 Series	1 Device	aes128 (RSL10 Evaluation Board) Cop
RSL10	ARM Cortex-M3 48 MHz, 24 KB RAM, 3	ble_android_asha (RSL10 Evaluation B) Solution
		🗢 ble_central_client_bond (RSL10 Evalua 🗇 Cop
		🗢 ble_central_client_scan (RSL10 Evaluati 🗇 Cop
		🖨 ble_central_peripheral (RSL10 Evaluati 🕸 Cop
		🖨 ble_peripheral_server_bond (RSL10 Ev: 🕸 Cop
		🖨 ble_peripheral_server_hrp (RSL10 Evalı 🕸 Cop
		🖨 ble_peripheral_server_hrp_fota (RSL10 🕸 Cop
		🖨 ble_peripheral_server_PRA (RSL10 Eval 🕸 Cop
		blinky (RSL10 Evaluation Board)

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).

kt	📮 🔝 Manage Run-Time Environme	nt			
Project: blinky	Software Component	Sel.	Variant	Version	Description
Source	 Device Startup 	V	source	1.0.0	Startup, System Setup System Startup for ON Semiconductor RSL10
e → include app.h	Bluetooth Profiles beta Libraries				
 Device Trsl10_protocol.c (Libraries:System) 	Weak_PRF System	v	release source	▼ 1.0.0 1.0.0	Weak Profile Library (weak prf) System Macros and Library (libsyslib)
sill_romvect.c (Libraries:System)	Remote_Mic Math		source	1.0.0	Remote Microphone Library (libremote micLib) Math Library (libmathlib)
sline statute sline statute rsline statute <p< td=""><td></td><td></td><td>release</td><td> 1.0.0 1.0.0 </td><td>Event Kernel Library (libkelib) Flash Library (libflashlib)</td></p<>			release	 1.0.0 1.0.0 	Event Kernel Library (libkelib) Flash Library (libflashlib)
 Isl10_sys_clocks.c (Libraries:System) Isl10_sys_crc.c (Libraries:System) 	Custom Protocol		source	1.0.0	Low Latency Audio Streaming Custom Protocol Library (libcustom protocol
 ssi10_sys_dma.c (Libraries:System) rs110_sys_flash.c (Libraries:System) 	Calibrate BLE		release	1.0.0	Calibration Library (libcalibratelib) Bluetooth Stack (libblelib)
rsl10_sys_power.c (Libraries:System)	File System Graphics		MDK-Plus MDK-Plus	 6.10.1 5.46.5 	File Access on various storage devices User Interface on graphical LCD displays
 Tsl10_sys_power_modes.c (Libraries:Syste Tsl10_sys_rffe.c (Libraries:System) 	m)		MDK-Plus MDK-Plus	▼ 7.9.0	IPv4 Networking using Ethernet or Serial protocols
 Isl10_sys_timers.c (Libraries:System) Isl10_sys_uart.c (Libraries:System) 	W- OSB		MDK-Plus	▼ 6.12.8	USB Communication with various device classes
 startup_rsl10.s (Startup) 	Validation Output		Descrip	tion	
system_rsl10.c (Startup)					

Figure 21. RTE Configuration for the Blinky Example Project in the Keil $\mu\text{Vision IDE}$

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 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: 00:002
```

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 a compatible version of J-Link.

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	p
🗋 🖬 🖉 🐰 🖉 🕺 🛍 🖉	Start/Stop Debug Session Ctrl+F5	const union gapc_d 🔻 🗟 🥐 🙋 🜖
🔗 🕮 🕮 🧼 📲 🙀 Targe	Energy Measurement without Debug	\checkmark
Project	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 7.4.1, "Downloading Firmware in Sleep Mode" on page 43.

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).

) 💀 • 🔜 • 🔜 • 🙀 •	** -			
roject 🛛 🗣 🔯	Disassembly				ą
** Project: blinky ** Target 1 ** Source ** app.c ** include ** app.h	125: } 126: else 127: { ♦0x00100546 2006 79: if(((uin		c);	PIO_Mask) == 0)	
Device	80: {	T ST S \$1 \$0 #2			4 111
 Institution of the second state of the second state					
 rsl10_sys_asrc.c (Libraries:System) 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_flash.c (Libraries:System) rsl10_sys_power.c (Libraries:System) rsl10_sys_power.c (Libraries:System) rsl10_sys_power_modes.c (Libraries:System) rsl10_sys_flash.c (Libraries:System) rsl10_sys_tflec.c (Libraries:System) rsl10_sys_tflec.c (Libraries:System) rsl10_sys_tflec.c (Libraries:System) 	108 ⊟ { 109 /*In 110 led_1 111 led_1 112 /*In 113 Init. 114 li5 /* S 116 while 117 日 {	<pre>up_rsl0.s itialize global va: toggle_status = 1; nitialize the syste ialize(); pin loop */ e (1) /* Refresh the wat;</pre>	em */		
⊕ 🗳 rsl10_sys_uart.c (Libraries:System)	119 120	Sys_Watchdog_Refre			
 rsl10_sys_uart.c (Libraries:System) rsl10_sys_version.c (Libraries:System) startup_rsl10.s (Startup) system_rsl10.c (Startup) 	120 121 122 123 ⊟ 123 ⊟ 124 125 - 126	<pre>Sys_Watchdog_Refre: /* Toggle GPIO 6 (: if (led_toggle_state) { Sys_GPIO_Toggle } else</pre>	<pre>sh(); if toggling is ena tus == 1)</pre>	abled) then wai	t 0.5 secor
 rsl10_sys_uart.c (Libraries:System) rsl10_sys_version.c (Libraries:System) startup_rsl10.s (Startup) system_rsl10.c (Startup) 	120 121 122 123 ⊟ 124 125 -	/* Toggle GPIO 6 (if (led_toggle_sta { Sys_GPIO_Toggle }	<pre>sh(); if toggling is ena tus == 1)</pre>	abled) then wai	ſ
 rsl10_sys_uart.c (Libraries:System) rsl10_sys_version.c (Libraries:System) startup_rsl10.s (Startup) system_rsl10.c (Startup) Project Registers 	120 121 122 123 □ 124 125 - 126 127 □ *	/* Toggle GPIO 6 (if (led_toggle_sta { Sys_GPIO_Toggle }	<pre>sh(); if toggling is ena tus == 1)</pre>	abled) then wai	ſ
 rsl10_sys_uart.c (Libraries:System) rsl10_sys_version.c (Libraries:System) startup_rsl10.s (Startup) system_rsl10.c (Startup) 	120 121 122 123 ⊟ 124 125 - 126 127 ⊟ Ⅲ ↓	/* Toggle GPIO 6 (: if (led_toggle_sta: { Sys_GPIO_Toggle } else {	<pre>sh(); if toggling is ena tus == 1)</pre>	abled) then wai	4

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

NOTE: Debug configurations are preconfigured for the sample applications in the CMSIS-Pack. Flash downloading through the Download icon (Figure 26) or F8 is not supported for J-Link.

File	Edit	View	Project	Flash	Debug	Periphe	rals
			8 h C			⇒ ¶ª	毘
		i 🗳 •		Targ	et 1	-	×
Proje							

Figure 26. Download Button Not Supported for J-Link

Getting Started with IAR

5.1 PREREQUISITE SOFTWARE

- 1. Download and install the IAR Embedded Workbench from the IAR Website, using the vendor's instructions.
- Download the RSL10 Software Package from <u>www.onsemi.com/RSL10</u> and extract the RSL10 CMSIS-Pack (ONSemiconductor.RSL10.<version>.pack) to any temporary folder.

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.

File	Edit	View	Project	J-Link	Tools	Window														_	
8 t t	2 🖻			101	DC		-	<	Q	>	\$►=	<	0	> [¢ >	0	•==	0	Ð	÷	₽

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	🕒 Console 😑 Pack Properties 👘 🗖
				⊞⊟ ⊗?≥≥ы ⊘ ⊽
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.

Search CMSIS N	/lanager			
File Edit Search CMSIS Manager	Window Help			
🔲 🔞 🍕 🕶 🚀 🕶 🖗 🖛 🏷	⇔ - ⇔ -			Quick Access
	🚵 Packs 🛛 📕 Devices 📓 Boards	E Pack Proper	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	X 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.

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

Search Example			
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 Evalua	🔶 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 Evaluation	🔶 Сору	Peripheral Device with Server and Standby Power Mode S	
spi_cmsis_driver (RSL10 Evaluation Board)	🔶 Сору	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 **IAR Embedded Workbench IDE** windows (see Figure 32 on page 27 and Figure 33 on page 28).

blinky/blinky.rteconfig - IAR Embedded Workbench CM File Edit Source Refactor Navigate Search Project		lindow blalo								
										Quick Access
0 • 2 6 0 • 2 • 2 • 2 • 6 • 6 • 6			5.6	• • •						and the second s
Project Explorer 11	88.4.0	♦ blinky.rteconfig II							- 0	월 Ou_ 11 @ 8u_ * 0
 B blinky 		♦ Components 🕑 🔤	olve						0	
Debug		Software Components	Gel	Variant	Vendor	Version	Description			An outline is not available.
> 😂 include		RSL10			ONSemicond			48 MHz, 24 kB RAM, 3	84 kB ROM	
* 🏝 RTE		A Device			Crtiserine Grins		- automation into		01.80110111	
A Device		Bluetooth Profile								
* 😂 RSL10		 Libraries 								
rsl10_protocol.c [ONSemiconductor:Dev		@ BLE		release	ONSemicond	ur 24450	Bluetooth Stack	(Dhblath)		
R rsl10_romwect.c (ONSemiconductor:Devi		Calibrate		source				ary (libcalibratelib)		
a rsl10_sys_asrc.c (ONSemiconductor:Devi		 Custom Proto 		source				dio Streaming Custom	Protocol Library (libr	
R rsl10_sys_audio.c (ONSemiconductor:De		 Flash 		source			Flash Library (lib		Protocol crowing (noc	
Ref rs110_sys_clocks.c [ONSemiconductor:De		· Fota		release			Fota Library (iib			
Rental Sys_crc.c (ONSemiconductor:Device		Kernel		release			Event Kernel Lib			
R rsl10_sys_dma.c (ONSemiconductor:Dev		Math		source			Math Library (III			
R rsl10_sys_flash.c (ONSemiconductor:Dev		Remote Mic		source				hone Library (libremote	micLib)	
rsl10_sys_power_modes.c [ONSemicondu		System		source				and Library (libsyslib)	(Junearie)	
R rsl10_sys_power.c [ONSemiconductor:De		Weak PRF		release			Weak Profile Lik			
R rs110_sys_rffe.c [ONSemiconductor:Devic		Startup	E	source				for ON Semiconductor	RS110	
In rsl10_sys_timers.c (ONSemiconductor:Device and the rsl10_sys_uart.c (ONSemiconductor:Device and the rsl10 sys_uart.c (ONSemicondu		4								
									14	
Instantup_rs10_sys_version.c (ON5emiconductor:Device) Startup_rs10.s (ON5emiconductor:Device)		Validation Output			D	escription				
system_rs10.c [ONSemiconductor:Devic										
	e Startup source)									
RTE_Components.h										
b is settings										
app.c		Components Device Pack								
blinky.ewp		Components Device Paci	9							
blinky.ewp		Problems 😫 🍭 Tasks	00	insole 🖾 Pro	perties					3
 blinky.ewt blinky.rteconfig 		0 items								
blinky.rteconfig iii readme_blinky.txt		Description	*		Re	source	Path	Location	Type	
sections.icf		ADDRESS OF THE								
iiii sections.ici										
<										

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

	1.1					0.	
orkspace	▼ 1	1 X					
lebug		-					
Files	\$	•					
🗣 blinky - Debug	~						
📮 🔳 CMSIS-Pack							
- 🗗 🗖 Device.Startup source							
│		•					
│ └─⊞ 🗟 system_rsl10.c		•					
🗕 🖵 🗖 Device.Libraries.System source							
–⊞ 🗟 rsl10_protocol.c		•					
⊞ 💽 rsl10_rom∨ect.c		•					
-⊞ 🖬 rsl10_sys_asrc.c		•					
–⊞ 🗟 rsl10_sys_audio.c		•					
–⊞ 🗟 rsl10_sys_clocks.c		•					
–⊞ 🗟 rsl10_sys_crc.c		•					
—⊞ 🗟 rsl10_sys_dma.c		•					
—⊞ 🗟 rsl10_sys_flash.c		•					
– 🕀 🖻 rsl10_sys_power.c		•					
⊢⊞ 🗟 rsl10_sys_power_modes.c		•					
⊢⊞ 🗟 rsl10_sys_rffe.c		•					
–⊞ 🗟 rsl10_sys_timers.c		•					
–⊞ 🗟 rsl10_sys_uart.c		•					
└─⊞ 🖻 rsl10_sys_version.c		•					
🕀 🛑 include							
—⊞ 🗟 app.c		•					
— 🗎 readme_blinky.txt							
— 🗉 🛋 Output							



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.

Build	Files	•
	🗆 🗉 🛢 blinky - Debug	~
Messages	Here include	
Building configuration: blinky -	Debug H 🖬 🖬 app.c	
Updating build tree	- Breadme_blinky.txt	
startup rsl10.s	He CMSIS-Pack	
rsl10 protocol.c	🖵 📮 📠 Output	
rsl10 romvect.c	🗕 🗎 blinky.map	
rsl10_sys_asrc.c	http://www.out	
app.c	- 🖓 🛋 Output	
rsl10_sys_audio.c	blinky.hex	
rsl10_sys_clocks.c	🔄 🖾 🗎 blinky.map	
rsl10_sys_crc.c	🗋 app.o	
rsi10_sys_crc.c rsi10_sys_dma.c	— 🗋 dl7M_tin.a	
rsi10_sys_uma.c rsi10_sys_flash.c	🗋 m7M_tl.a	
rsi10_sys_nasri.c rsi10_sys_power.c	🗋 rsl10_protocol.o	
	🗋 rsl10_romvect.o	
rsl10_sys_rffe.c	rsl10_sys_asrc.o	
rsl10_sys_power_modes.c	rsl10_sys_audio.o	
rsl10_sys_timers.c	rsl10_sys_clocks.o	
rsl10_sys_uart.c	rsl10_sys_crc.o	
rsl10_sys_version.c	🗋 rsl10_sys_dma.o	
system_rsl10.c	rsl10_sys_flash.o	
Linking	rsl10_sys_power.o	
blinky.out		
Converting	rsl10_sys_rffe.o	
	limers.o	
Total number of errors: 0	rsl10_sys_uart.o	
Total number of warnings: 0	rsl10_sys_version.o	
	rt7M_tl.a	
Build Debug Log	ections.icf	
	D 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	nk V6.34h Out of breakpoints 🛛 🕅
<u>^</u>	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 7.4.1, "Downloading Firmware in Sleep Mode" on page 43.

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

Resolving External CMSIS-Pack Dependencies

1. EXTERNAL CMSIS-PACK DEPENDENCIES

Some of the RSL10 sample applications depend on software components from external vendors. For example, applications that make use of CMSIS-Drivers or FreeRTOS depend on CMSIS-Packs provided by Arm[®]. The dependencies are displayed in the RTE Configuration (see Figure 39 for an example).

2. RESOLVING EXTERNAL DEPENDENCIES

The following instructions show how to easily identify and resolve external dependencies in RSL10 sample applications using the CMSIS-Pack manager.

Software Components	Sel.	Variant	Vendor		Version	Description			
RSL10			ONSemico	nduc		ARM Cortex-M3 48 MHz, 24 KB RAM, 388 KB ROM			
CMSIS									
CMSIS Driver									
Device									
RTOS		FreeRTOS	ARM						
Validation Output				Des	cription				
ARM::CMSIS.RTOS2	2.FreeR	TOS		Com	ponent is	missing. Pack is not installed: ARM.CMSIS-FreeRTOS			
ARM.FreeRTOS::RT	OS.Cor	fig.CMSIS RT	OS2	Com	ponent is	missing. Pack is not installed: ARM.CMSIS-FreeRTOS			
ARM.FreeRTOS::RT	OS.Cor	e.Cortex-M		Com	ponent is	missing. Pack is not installed: ARM.CMSIS-FreeRTOS			
ARM.FreeRTOS::RT	OS.Eve	nt Groups		Component is missing. Pack is not installed: ARM.CMSIS-FreeRTOS					
ARM.FreeRTOS::RT	OS.Hea	ap.Heap_4		Component is missing. Pack is not installed: ARM.CMSIS-FreeRTOS					
ARM.FreeRTOS::RTOS.Timers				Component is missing. Pack is not installed: ARM.CMSIS-FreeRTOS					

Figure 39. RTE Configuration Perspective Before Resolving Pack Dependencies

Figure 39, above, shows the RTE Configuration view when Pack dependencies are unresolved. To resolve Pack dependencies, follow these steps:

1. In the CMSIS-Pack Manager perspective, click on the Check for Updates on Web button (see Figure 40).



Figure 40. Check for Updates on Web Button

Figure 41, below, shows an example of the Packs tab after checking for updates.

	🕀 🖻 💥 🕐 🔻	Search Pack		
Search Device		Pack	Action	Description
Device	Summary	Pack Pack Pack Pack Pack	8 Packs	Description ARM selected
All Devices	6249 Devices		♦ Install	Musca A1 Board Support PACK for CoreLink SSE-200 based
All Devices	20 Devices	ARM.Musca_A1_BSP	Install	Musca B1 Board Support PACK for CoreLink SSE-200 based
Active-Semi	4 Devices	ARM.V2M-MPS2_SSE_200_B		ARM V2M-MPS2 Board Support PACK for CoreLink SSE-200 Dased
Active-Seriii	8 Devices			
	5 Devices	ARM.V2M-MPS3_SSE_200_B		ARM V2M-MPS3 Board Support PACK for CoreLink SSE-200
Annecom		Keil.V2M-MPS2_CMx_BSP	✤ Install	ARM V2M-MPS2 Board Support PACK for Cortex-M System
Analog Devices	14 Devices	Keil.V2M-MPS2_DSx_BSP	✤ Install+	ARM V2M-MPS2 Board Support PACK for DesignStart Devic
APEXMIC	14 Devices		Install	ARM V2M-MPS2 Device Family Pack for IOT-Kit devices
ARM	57 Devices		Install+	ARM V2M-MPS3 Device Family Pack for IOT-Kit devices
AutoChips	24 Devices	 Generic 	38 Packs	Software Packs with generic content not specific to a device
Cypress	425 Devices	Alibaba.AliOSThings	🅸 Install	AliOS Things software pack
Dialog Semiconduc		ARM.AMP	🕸 Install	Software components for inter processor communication (A
GigaDevice	160 Devices	A # ARM.CMSIS	🕸 Install	CMSIS (Cortex Microcontroller Software Interface Standard)
HDSC	26 Devices	₱ 5.5.1 (2019-03-20)	🐸 Unpack	The following folders are deprecated - CMSIS/Include/
Holtek	171 Devices	Previous		ARM.CMSIS - Previous Pack Versions
🖻 🍳 Infineon	175 Devices	ARM.CMSIS-Driver	Install+	CMSIS Drivers for external devices
Lapis Semiconducto	2 Devices	ARM.CMSIS-Driver_Validation	🔅 Install	CMSIS-Driver Validation
Maxim	16 Devices	ARM.CMSIS-FreeRTOS	Install+	Bundle of FreeRTOS for Cortex-M and Cortex-A
Mediatek	2 Devices	ARM.CMSIS-RTOS_Validation	Install	CMSIS-RTOS Validation
Microchip	378 Devices	ARM.mbedClient	Install	ARM mbed Client for Cortex-M devices
MindMotion	89 Devices	ARM.mbedTLS	Install	ARM mbed Cryptographic and SSL/TLS library for Cortex-M
Nordic Semiconduc	t 15 Devices	ARM.minar	Install	mbed OS Scheduler for Cortex-M devices
Nuvoton	621 Devices	ARM.TFM	Install	Trusted Firmware-M (TF-M) is the reference implementation
NXP	1169 Devices	birdec.bi-pcm3060	♦ Install	CMSIS-Driver for sound codec TI PCM3060
ONSemiconductor	1 Device	Bindedid participation Bindedid participation		Flexible Safety RTOS

Figure 41. Installing the Arm CMSIS-Pack

- 2. To manually install a CMSIS-Pack, select the **Packs** tab and search for the required CMSIS-Pack (in this example, we installed the *ARM.CMSIS* pack); click the **Install** button (shown in Figure 41). Alternatively, follow the next steps to automatically resolve any Pack dependencies that are missing.
- 3. Open the *. rteconfig file; in the Packs tab, select the Resolve Missing Packs button (see Figure 42).

ack	Selection	Version	Description
ARM.CMSIS	latest	5.5.1	Pack is not installed
ARM.CMSIS-FreeRTOS	latest	3.0.534	Pack is not installed
ONSemiconductor.RSL10	latest	3.0.534	ON Semiconductor RSL10 Device Family Pack

Figure 42. Resolve Missing Packs Icon

The IDE prompts you to read and accept the license agreement, then installs the missing Packs. Figure 43 4. illustrates the RTE configuration after resolving missing Packs.

Pack	Selection	Version	Description
ARM.CMSIS	latest	5.5.1	CMSIS (Cortex Microcontroller Software Interface Standard
ARM.CMSIS-FreeRTOS	latest	10.2.0	Bundle of FreeRTOS for Cortex-M and Cortex-A
ONSemiconductor.RSL10	latest	3.0.534	ON Semiconductor RSL10 Device Family Pack

Figure 43. RTE Configuration Perspective After Resolving Pack Dependencies

Advanced Debugging

7.1 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.

7.1.1 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.

7.2 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 3.4.1, "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 44:

RSL10 Getting Started Guide

```
set {int} &__VTOR = ISR_Vector_Table
set $sp = *((int *) &ISR_Vector_Table)
```

🖹 Main 🏇 Debugger 🕟 Startup 🛛 🦆 Source 🔲 Common	
Initialization Commands	
☑ Initial Reset and Halt Type: Low speed: 1	000 kHz
JTAG/SWD Speed: Auto Adaptive Fixed kH	z
Enable flash breakpoints	
▼ Enable semihosting Console routed to: ▼ Telnet □ GDB c	ient
Enable SWO CPU freq: 10000000 Hz. SWO freq: 0	Hz. Port mask: 0x1
	_
Load Symbols and Executable	
Coad symbols	
Use project binary: blinky.elf	
O Use file:	Workspace File System
Symbols offset (hex):	
Vertical executable	
Use project binary: blinky.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: (alway)	ys executed at Restart)
set {int} &VTOR = ISR_Vector_Table	•
<pre>set \$sp = *((int *) &ISR_Vector_Table)</pre>	_
Set program counter at (hov)	
Set program counter at (hex):	
Set breakpoint at: main	
Continue	

Figure 44. Setting Up a GDB Launch Configuration, Startup Tab

7.3 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.

7.4 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 45). Save and compile your application.



Figure 45. 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 46, and Figure 47 on page 39).
- 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 46). 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 7.4.1, "Downloading Firmware in Sleep Mode" on page 43.

	ns	S
🗎 🗶 📄 🎲 🗸	Name: peripheral_server_sleep Debug	
pe filter text	🗎 Main 🕸 Debugger 🕨 Startup 🛛 🧐 Source 🗖 Common 🔜 SVD Path	
C/C++ Application	Initialization Commands	
C/C++ Attach to Application	✓ Initial Reset and Halt Type: 1 Low speed: 1000 kHz	
 C/C++ Postmortem Debugger C/C++ Remote Application 	JTAG/SWD Speed: Auto Adaptive Fixed kHz	
Eclipse Application	Enable flash breakpoints	
COB Hardware Debugging	Enable semihosting Console routed to: V Telnet GDB client	
GDB OpenOCD Debugging	Enable SWO CPU freq: 0 Hz. SWO freq: 0 Hz. Port mask 0x1	
GDB SEGGER J-Link Debugging		
peripheral_server_sleep Debug		
peripheral_server_sleep_swd_att Java Applet		
Java Application	Load Symbols and Executable	
Ju JUnit	✓ Load symbols	
🚏 JUnit Plug-in Test	Use project binary: peripheral_server_sleep.elf	
🖶 Launch Group	Use file: Workspace File System	
Launch Group (Deprecated) Mwe2 Launch	Symbols offset (hex):	
OSGi Framework	V Load executable	
Remote Java Application	Use project binary: peripheral_server_sleep.elf	
	Executable offset (hex):	
		Ĺ
er matched 19 of 100 items	Revert Apply	
	Debug	

Figure 46. Setting Reset Type in the Debug Configuration Session

Debug Configurations	- 0
eate, manage, and run configuratio	ns 🕅
P 🖗 🗎 🗶 🖻 🏞 🕶	Name: peripheral_server_sleep Debug
pe filter text	📔 Main 🕸 Debugger 🍺 Startup 👍 Source 🔲 Common 🛃 SVD Path
C /C++ Application C /C++ Attach to Application C /C++ Characher Launcher C /C++ Postmortem Debugger C /C++ Postmortem Debugger C /C++ Unit Debugging C /C++ Unit Debugging C OB SEGGER J-Link Debugging C peripheral_sever_steep Debug L Launch Group	Initial action Commands Initial Reset and Halt Type Low speed: 1000 kHz ITAG/SWD Speed: Auto Adaptive Fixed kHz Finable fash breakpoints Finable semihosting Console routed to: Telnet GDB client Finable SWO CPU freq: 0 Hz. SWO freq: 0 Hz. Port mask: 0x1
 Launch Group (Deprecated) 	· · · · · · · · · · · · · · · · · · ·
	Load symbols Use project binary: peripheral_server_sleep.eff Use file: Symbols offste (here): Load executable @ Use project binary: peripheral_server_sleep.eff
	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:
er matched 12 of 49 items	Revert Apply
)	
)	Debug Close

Figure 47. Startup Tab: Debug Session that Initiates Restart

- b. Debug session that connects to the running target:
 - i. Create another new debug configuration under the **GDB SEGGER** heading, with new configuration details in the right panel.
 - ii. Adjust the displayed values for your configuration then click on **Apply** (see Figure 48, and Figure 49 on page 41).

RSL10 Getting Started Guide

Debug Configurations								
eate, manage, and run configurations				Ŕ				
) 🕼 🗊 🗶 🖃 🎲 🕶	Name: peripheral_ser	er_sleep Debug_swd_att						
rpe filter text	📄 Main 🟇 Debugg	r 🕟 Startup 🧤 Source 🔲 Common 🔚 SVD Path						
C/C++ Application C/C++ Attach to Application	J-Link GDB Server Se		et					
C/C++ Container Launcher C/C++ Postmortem Debugger	Executable path:	\${jlink_path}/\${jlink_gdbserver}	Browse \	/ariables				
C/C++ Remote Application	Actual executable:	C:/Program Files (x86)/SEGGER/JLink//JLinkGDBServerCL.exe						
Ct C/C++ Unit C GDB Hardware Debugging	(to change it use the <u>global</u> or <u>workspace</u> preferences pages or the <u>project</u> properties page)							
GDB OpenOCD Debugging	Device name:	RSL10	Supported de	vice names				
GDB SEGGER J-Link Debugging	Endianness:	Little Big						
 peripheral_server_sleep Debug peripheral_server_sleep Debug (1) 	Connection:	USB O IP (USB serial or IP	name/address)					
🖡 Launch Group	Interface:	● SWD ○ JTAG						
Launch Group (Deprecated)	Initial speed:	Auto Adaptive Fixed 1000 kHz						
	GDB port:	2331						
	SWO port:		Initialize registe	rs on start				
	Telnet port:	2333 🗹 Local host only	Silent					
	Log file:			Browse				
	Other options:	-singlerun -strict -timeout 0 -nogui						
	Allocate console	for the GDB server Allocate console for semihor	sting and SWO					
	GDB Client Setup							
	Executable name:	\${cross_prefix}gdb\${cross_suffix}	Browse \	/ariables				
	Actual executable:	arm-none-eabi-gdb						
	Other options:							
	Commands:	set mem inaccessible-by-default off		< _ >				
	Remote Target							
	Host name or IP ad	Iress: localhost						
	Port number:	2331						
	Force thread list up	date on suspend						
er matched 13 of 50 items			Revert	Apply				
)			Debug	Clos				

Figure 48. Debugger Tab: Debug Session that Connects to the Running Target

ype filter text C/C++ Application C/C++ Atach to Application C/C++ Container Launcher C/C++ Container Launcher C/C++ Container Launcher TAG/SWD	set and Halt Type: Low speed: 1000 kHz
e filter text C/C++ Application C/C++ Ataphication C/C++ Atach to Application C/C++ Container Launcher C/C++ Container Leuncher C/C++ Container Debugger	Debugger 🕟 Startup 🗄 Source 🔲 Common 😨 SVD Path Commands set and Halt Type Low speed: 1000 kHz
C C/C++ Application C /C++ Attach to Application C /C++ Container Launcher C /C++ Postmortem Debugger JTAG/SWD	Commands set and Halt Type: Low speed: 1000 kHz
C C/C++ Attach to Application C C/C++ Container Launcher C C/C++ Postmortem Debugger JTAG/SWD	set and Halt Type: Low speed: 1000 kHz
Cti C/C++ Unit	Speed: Ø.Auto O.Adaptive Fixed kHz ash breakpoints meminstring Console routed to: Telnet GDB client WO CPU freq; 0 Hz. SWO freq; 0 Hz. Port mask: 0x1
C GDB SEGGER J-Link Debugging peripheral_server_sleep Debug peripheral_server_sleep Debug (1)	^
◯ Use file Symbols o □ Load ext ○ Use pro Use file	ject binary: peripheral_server_sleep.elf Workspace File System fiset (hex): peripheral_server_sleep.elf Workspace File System offiset (hex):
Run/Restart	Commands Restart reset Type: (always executed at Restart)
Set prog	· · · · · · · · · · · · · · · · · · ·
	Restore defa
er matched 14 of 44 items	Revert Apply

Figure 49. Startup Tab: Debug Session that Connects to the Running Target

5. Start the first debug session (which initiates target restart). Once the target is halted at main, resume the execution (see Figure 50).

on-semiconductor-workspace - peripheral_server_sleep/app.c - ON Semiconductor IDI	E
File Edit Source Refactor Navigate Search Project Run Window Help	
🔨 🔅 🔳 🎋 Debug 🗸 🕑 peripheral_server_sleep Debug	✓♣ ↓ 🗂 ▼ 📓 🔞 ↓ 🖶 ↓ ♥ ↓ ● № 3. 3. 3. (2) ↔ ↓
4a ▼ 7a ▼ *Þ	
🗱 Debug 🔀 🎦 Project Explorer 🦓 🖬 👻 🗖 🗖	i app_process.c i app.c ⊠
 c peripheral_server_sleep Debug [GDB SEGGER J-Link Debugging] peripheral_server_sleep.elf Thread #1 57005 (Suspended : Breakpoint) main() at app.c:21 0x1030f4 J.LinkGDBServerCL.exe arm-none-eabi-gdb Semihosting and SWV 	<pre>9 * 10 * app.c 11 * - Main application file 12 * 13 * \$Revision: 1.73 \$ 14 * \$Date: 2018/10/09 15:03:34 \$ 15 * 16 17 #include "app.h" 18 19@ int main() 20 { 21 22 22 23 24 /* Whit for 2 scends to place as flaching directly </pre>

Figure 50. First Debug Session Perspective Before Starting Execution

6. Wait until the target enters Deep Sleep Mode. At this point the debug connection is lost; and even when the target is awake, it cannot establish a connection with JTAG. The following output is generated on the console (see Figure 51).

						Executables 🖷 Debugger Console 🔋 Memory 🔗 Search
	-					ER J-Link Debugging] JLinkGDBServerCL.exe
				0		(R2) while CPU is running
				0		(R3) while CPU is running
				0		(R4) while CPU is running
				0		(R5) while CPU is running
				0		(R6) while CPU is running
				0		(R7) while CPU is running
						(R8) while CPU is running
						(R9) while CPU is running
ERROR:	Can	not	read	register	10	(R10) while CPU is running
				-		(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	: Fa	iled	t to r	read memor	ry	@ address 0xDEADBEEE
Startin	g ta	rget	CPU.			
ERROR:	CPU	is r	not ha	alted		
ERROR:	Can	not	read	register	15	(R15) while CPU is running
Reading	all	reg	gister	`S		
-	-			• •	^	(DO) 111 ODU 1 1

Figure 51. 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 52).



Figure 52. Terminate Targets Icon

8. After the target exits Deep Sleep Mode, it is running in the infinite loop (step 1), and you can connect to the running target by starting the second debug session (see Figure 53). Note that the debugger is able to reattach to the running target and halt the processor after waking up from sleep.



Figure 53. Second Debug Session Perspective After Connecting to the Running Target

7.4.1 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 7.4, "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 8

More Information

8.1 FOLDER STRUCTURE OF THE RSL10 CMSIS-PACK INSTALLATION

By default, the CMSIS-Pack contents are installed in the following location:

- If you are using the Eclipse-based ON Semiconductor IDE: ٠ C:\Users\<user id>\ON_Semiconductor\PACK\ONSemiconductor\RSL10\<version>.
- If you are using the Keil IDE: %LOCALAPPDATA%\Arm\Packs ٠
- If you are using the IAR IDE: ٠ $C: Users < user_name > IAR-CMSIS-Packs ONSemiconductor RSL10 < version > Variable Statement (RSL10) < version > Variable Statement (R$

Subfolders and files are described in Table 1 and Table 2.

Table 1. Installed Folders and Files - 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.					
ONSemiconductor.RSL10.pdsc	A file that describes the dependencies to devices, processor, toolchains and other software components for the RSL10 CMSIS-Pack.					
PACK_REVISION	Identifies the revision of the RSL10 CMSIS-Pack.					
Software_Use_Agreement.rtf	ON Semiconductor license agreement.					

Table 2. Installed Folders and Files - 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.
ide.exe	Executable that opens the ON Semiconductor IDE.
REVISION	Identifies the revision of the ON Semiconductor IDE.
Software_Use_Agreement.rtf	ON Semiconductor license agreement.
ThirdPartyLicenses.txt	License agreements with third party software included in the IDE.

8.2 DOCUMENTATION

8.2.1 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*\<version>*documentation* (where <user_id is your profile name, and <version> is the version number, e.g., 3.0.521).

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

- 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 54, below).
- Keil µVision IDE: documentation is available in the **Books** tab, as shown in Figure 55.
- IAR Embedded Workbench: documentation is accessible through the IAR Embedded Workbench CMSIS Manager window, as shown in Figure 56 on page 47.



Figure 54. Accessing RSL10 Documentation from the ON Semiconductor IDE



Figure 55. Accessing RSL10 Documentation from the Keil μ Vision IDE

area and a second as a	S P	B Packs 51 B Devices B Boards Earch Pack C Conserved Specific C Conserved Specific Generic C Conserved Specific Generic	Action 1 Pack	Censole Description RSLI0 exterted 0.0 Semiconductor RSLI0 Device Family Pack Software Packs with generic content not specific to a device	Quick Access 28 🕲
Inteconfig II ice	M S	Search Pack Pack Povice Specific ONSemiconductor RSL10	Action 1 Pack	Description RS.10 selected ON Semiconductor RS.10 Device Family Pack	
KSL10 KSL10 KSL10 Change,, KSL10 Change,, KSL10 CPU: ARM Cortex-M3 Max Clock 48 MHz ONSemiconductor ONSemiconductor ONSemiconductor/SSL10.2.4450 FPU: nore http://www.kel.com/dd2/consemiconductor/s110 Endian Description: data books Description: data books CPU: SSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power. Kand Thumb-2 Instruction Set Quick Reference Card	M S	Search Pack Pack Povice Specific ONSemiconductor RSL10	Action 1 Pack	Description RS.10 selected ON Semiconductor RS.10 Device Family Pack	⊕ ⊆] # \$ 8 ia 0 ⊤ ⊤ □
RSL10 Change RSL10 Series CPU: ARM Cortex-M3 Mp: Max Cocice 48 MHz ONSemiconductor Memory: 24 KB RAM, 384 KB R0 OVSemiconductor/SSL10.2.4.450 FPU: none bttp://www.kei.com/dd2/consemiconductor/st10 Enclaint bttp://www.kei.com/dd2/consemiconductor/st10 Enclaint data books Description: and Thumb-2 Instruction Set Quick Reference Cand PSL10 is an Utra-low-power.	P A	Pack Pack Povice Specific Provide Specific Provide Specific Provide Specific Provide Specific Sp	1 Pack	RSL10 selected ON Semiconductor RSL10 Device Family Pack	
RSL10 Series CPU: ARM Cortex-M3 Max Clock: 48 MHz ONSemiconductor ONSemiconductor Memory: 24 48 RAM, 384 8 RO ONSemiconductor/SL10.2.4450 FPU: none http://www.kei.com/dd2/consemiconductor/s110 Endiare Little-endian data books Description: 4 and Thumb: 2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power.	M.	Device Specific Monopole ONSemiconductor.RSL10	1 Pack	RSL10 selected ON Semiconductor RSL10 Device Family Pack	
RSL10 Series CPU: ARM Cortex-M3 Max Clock: 48 MHz ONSemiconductor ONSemiconductor Memory: 24 48 RAM, 384 8 RO ONSemiconductor/SL10.2.4450 FPU: none http://www.kei.com/dd2/consemiconductor/s110 Endiare Little-endian data books Description: 4 and Thumb: 2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power.	M	B ONSemiconductor.RSL10		I ON Semiconductor RSL10 Device Family Pack	
tilly: Max. Clock 48 MHz Memory: 24 K8 RAM, 384 K8 RO ONSemiconductor Memory: 24 K8 RAM, 384 K8 RO ONSemiconductor/RSL10.2.4.450 FPU: none http://www.kei.com/dd2/consemiconductor/rsl10 Enclarc Little-enclan data books Description: 4 and Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power.			 Up to de 		
ONSemiconductor Memory: 24 kB RAM, 384 kB RO ONSemiconductor.RSL10.2.4450 FPU: none Intru/www.keit.com/dd2/onsemiconductor/hs110 Endian data books Description: 4 and Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power.		Generic		Software Packs with generic content not specific to a device	
ONSemiconductor/IS310.2.4.450 FPU: none http://www.kei.com/id32/onsemiconductor/IS100 Endiane Little-endian data books Descriptions: 4 and Thumb-2 Instruction Set Quick Reference Card RS10 is an ultra-low-power.					
http://www.keil.com/dd2/onsemiconductor/still Enclian: Little-enclian data books: Description:					
data books: Description: 4 and Thumb-2 Instruction Set Quick Reference Card RSL10 is an ultra-low-power,	3				
Interface Exection multi-protocol 24 GHz radio					
IT Interface Specification designed for use in wireless devices that demand low power					
ting Started with RSL10 consumption and a restricted size.					
Interface Specification					
10 Bootloader Guide					
10 Firmware Over-The-Air User's Guide 10 Firmware Reference					
10 Hardware Reference					
10 Sample Code Users Guide					
BLE Alert Notification Profile Interface Specification					
BLE Battery Service Interface Specification					
BLE Blood Pressure Profile (BLP) Interface Specification					
BLE Cycling Power Profile Interface Specification					
BLE Cycling Speed and Cadence Profile Interface Specification					
BLE Device Information Service Interface Specification					
BLE Find Me Profile Interface Specification BLE Glucose Profile (GLP) Interface Specification					
BLE Health Thermometer Profile Interface Specification					
BLE Heart Rate Profile (HRP) Interface Specification					
BLE HID Over GATT Profile Interface Specification					
BLE Host Error Code Interface Specification					
BLE Location and Navigation Profile Interface Specification					
BLE Phone Alert Status Profile Interface Specification Interface Specification					
BLE Proximity Profile Interface Specification					
BLE Running Speed and Cadence Profile Interface Specification					
BLE Scan Parameters Profile Interface Specification BLE Time Profile (TIP) Interface Specification					
BLE Wireless Power Transfer Profile (WPTP) Specification					
and the second states a second state of a second state of a second second					
tible boards:					
ents Device Packs					

Figure 56. 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.

8.2.2 Documentation in the RSL10 Documentation Package

You can access documentation through the *RSL10 DOCUMENTATION PACKAGE.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.

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 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 Software Package from www.onsemi.com/RSL10 and extract the RSL10 CMSIS-Pack (ONSemiconductor.RSL10.<version>.pack) to 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 J-Link path (global, workspace, project-based) for the location of the Segger J-Link executables. This replaces the Run/Debug string substitutions for J-Link 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 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 of 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 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