# **RSL10 Getting Started Guide**

M-20836-010 July 2022

© SCILLC, 2022 Previous Edition © 2021 "All Rights Reserved"



\_

# **Table of Contents**

P	age
1. Introduction	4
1.1 Overview	4
1.2 Intended Audience	4
1.3 Conventions	4
2. Setting Up the Hardware	6
2.1 Prerequisite Hardware	6
2.2 Connecting the Hardware	6
2.3 Preloaded Sample	6
3. Getting Started with the Eclipse-Based onsemi IDE	8
3.1 Software to Download	8
3.2 onsemi IDE and RSL10 CMSIS-Pack Installation Procedures	8
3.3 Building Your First Sample Application with the onsemi IDE	10
3.3.1 Launching the onsemi IDE	10
3.3.2 Importing the Sample Code	10
3.3.3 Build the Sample Code	12
3.4 Debugging the Sample Code	14
3.4.1 Debugging with the .elf File.	14
3.4.2 Peripheral Registers View with the onsemi IDE	16
	20
4. Getting Started with Kell.	20
4.1 Prerequisite Software	20
4.2 KSL10 CMSIS-Fack Installation Flocedure	20 21
4.5 Durining rout rust sample Appreation with the Ken division IDE	21 ))
4.2.2 Duild the Sample Code	 
	23
4.3.3 Debugging the Sample Code	24
4.3.3.1 Preparing J-Link for Debugging	24
4.3.3.2 Debugging Applications	24
5. Getting Started with IAR	26
5.1 Prerequisite Software	26
5.2 RSL10 CMSIS-Pack Installation Procedure	26
5.3 Building Your First Sample Application with the IAR Embedded Workbench	27
5.3.1 Import the Sample Code	27
5.3.2 Building the Sample Code	29
5.3.3 Debugging the Sample Code	30
5.3.3.1 Debugging Applications	30
6 Resolving External CMSIS-Pack Dependencies	22
1 External CMSIS-Pack Dependencies	33
2 Resolving External Dependencies	33
7 Advanced Debugging	26
7. Auvalieu Debuggilig	36
7.1.1. Adding Printf Debug Capabilities	36
7.2. Debugging Applications that Do Not Start at the Dece Address of Elech	27
7.2 Debugging Applications that Do Not Start at the Base Address of Flash	)/ )0
	20

7.4 Debugging with Low Power Sleep Mode		•		38 45
8. More Information.				46
8.1 Folder Structure of the RSL10 CMSIS-Pack Installation				46
8.2 Documentation				47
8.2.1 Documentation Included with the CMSIS-Pack	•			47
8.2.2 Documentation in the RSL10 Documentation Package				51
A. Migrating to CMSIS-Pack				53
A.1 Migrating an Existing Eclipse Project to the CMSIS-Pack Method				53
A.2 Using the Latest RSL10 Firmware in a Previous Version of the Eclipse-Based IDE				54
B. Arm Toolchain Support.				55
B.1 Basic Installation				55
B.2 Configuring the Arm Toolchain in the onsemi IDE				55
B.3 Additional Settings				55

## CHAPTER 1

## Introduction

#### 1.1 OVERVIEW

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

RSL10 is a multi-protocol, Bluetooth<sup>®</sup> 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 onsemi Integrated Development Environment (IDE) is offered as a free download with optional support for Arm<sup>®</sup> Keil<sup>®</sup> µVision<sup>®</sup> and IAR Embedded Workbench<sup>®</sup>.

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 53 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 <b>filename</b> with the actual name of the file.

**bold** Graphical user interface labels, such as those for menus, menu items and buttons.

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

## **CHAPTER 2**

## Setting Up the Hardware

### 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:

- "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.

## CHAPTER 3

## Getting Started with the Eclipse-Based onsemi IDE

## 3.1 SOFTWARE TO DOWNLOAD

- 1. Download the onsemi 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. Make sure your J-Link software is version 7.66b or higher.

## 3.2 ONSEMI IDE AND RSL10 CMSIS-PACK INSTALLATION PROCEDURES

For instructions on installing the onsemi IDE, see the *onsemi Installation Instructions and Release Notes* document.

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 onsemi Launcher in step 2.
- 2. Open the onsemi IDE by going to the Windows Start menu and selecting **onsemi** > **onsemi IDE**. From the onsemi IDE Launcher screen, browse to your new workspace, select it, and click Launch.
- 3. On the top row of the Workbench perspective, click the "Make the CMSIS Packs Manager perspective visible" icon (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).



## Figure 3. Installing the RSL10 CMSIS-Pack

- 5. The IDE installs the RSL10 CMSIS-Pack in the specified pack root folder.
- 6. Read the license agreement, *Software Use Agreement use and accept (ONIPLAW 08142020).pdf*, found in the root directory of the installed CMSIS-Pack.
- 7. The RSL10 CMSIS-Pack now appears in the list of installed packs. In the Devices tab, if you expand All Devices > onsemi > 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.

Workspace_2019-11-18 -	ON Semiconductor IDE				- 🗆 X
File Edit Navigate Search	n Project Run Window Help	)			
« O I	V No Launch Configu	rations 🗸 on:	~ <del>\$</del>		A -
					Quick Access 🗄 🛃 🗟 嫠
📕 Devices 🖾 📕 Boards	∃ □ × □ □	🚵 Packs 🕴 📑 Examples		H E 🔗 💝 🐸 🕍 🍞 🗇 🔻 🗆	😑 Pack Properties 🛛 🗖 🗖
Search Device		Search Pack			E 0 V
Device	Summary 1 Device 1 Device 1 Device <u>ARM Cortex-M3 48 MHz, 24 KI</u>	Pack	Action 1 Pack V Up to da Remove	Description RSL10 selected ON Semiconductor RSL10 Device Family Pack Release notes available at www.onsemi.com/rs10 Software Packs with generic content not specific to	type filter text ✓
<	>	٢		,	< >>
🖾 Console 🕮 🦷 Progress					k 🖬 B≠ 🖻 + 🗗 + 🗉 D
CMSIS Console					
					<u>^</u>
<					>

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

### 3.3 BUILDING YOUR FIRST SAMPLE APPLICATION WITH THE ONSEMI 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 onsemi IDE

Open the onsemi IDE by going to the Windows Start menu and selecting **onsemi** > **onsemi 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).

Workspace_2019-11-18 -	ON Semiconductor IDE				- 🗆
File Edit Navigate Search	Project Run Window Help		A		
<u> </u>	No Launch Configurations	• on: • •			≪ <b>-</b>
					Quick Access
Devices      Boards		Backs      Examples      X		- 0	E Pack Properties
Search Device		Only show example	s from insta	lled packs 🔗 ぞ 🐸 🕍 🧭 🔍	œ e
Device	Summan	Search Example			type filter text
	1 Device			Desister	v 🖶 ONSemicondu
<ul> <li>All Devices</li> <li>MSemiconductor</li> </ul>	1 Device	Example	Action	Description	> Boards
PSI 10 Series	1 Device	ADC_UART (KSLTU Evaluation Board)	Copy	ADC with UARI Sample Code	> 🚸 Componen
PCI 10	APM Cortex M2 49 MHz 24 KP PAM :	esize (RSLIU Evaluation Board)	Copy	AES128 - Simple AES-128 Sample	> Devices
- KSLIU	ARIVI COITEX-IVIS 46 IVITIZ, 24 KB KAIVI, 1	ble_android_asha (RSL10 Evaluation B)	Copy	Android Audio Streaming Hearing	> P Examples
		ble_central_client_bond (RSL10 Evalua)	Copy	BLE Central Client Bonding Sampl	Li Examples
		ble_central_client_scan (RSL10 Evaluation)	Copy	Central Device with Client Scanne	
		ble_central_peripheral (RSL10 Evaluation)	😻 Сору	Central Peripheral Device Sample	
		ble_peripheral_server_bond (RSL10 Evaluation of the server_bond)	😻 Сору	BLE Peripheral Server Bonding Sa	
		ble_peripheral_server_hrp (RSL10 Evaluation)	😻 Сору	Heart Rate Peripheral Device with	
		ble_peripheral_server_hrp_fota (RSL10)	🕸 Сору	Heart Rate Peripheral Device with	
		ble_peripheral_server_PRA (RSL10 Eval	🔅 Сору	Private resolvable Address with Se	
		blinky (RSL10 Evaluation Board)	Copy	Blinky GPIO I/O Sample Code	
		bootloader (RSL10 Evaluation Board)	🕸 Сору	Bootloader Sample Code	
		central_client_uart (RSL10 Evaluation B	🕸 Сору	Central Device with Client UART S	
		custom_protocol_trx (RSL10 Evaluation	😻 Сору	Low Latency Audio Sample Appli	
		default_MANU_INFO_INIT (RSL10 Evaluation)	🔅 Сору	Default System Initialization Func	
		dma_driver (RSL10 Evaluation Board)	🕸 Сору	DMA Driver Sample Code	
		DMIC_OD (RSL10 Evaluation Board)	🚸 Сору	DMIC and OD Sample Code	
		flash_copier_and_crc (RSL10 Evaluation	🔅 Сору	Flash Copier and CRC Sample Coc	
		freertos_ble_central_client_bond (RSL1	🕸 Сору	BLE Central Client Bonding Samp	
		freertos_ble_peripheral_server_bond (F	🕸 Сору	BLE Peripheral Server Bonding Sal	
1			( <b>*</b> -	the second s	1
Console S S Progress					k 🚮 📴 🖻 🔻 📑
Civisis Console					
<					

### 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).

Workspace_2019-11-18 - blinky/blin	ky.rteconfig - ON Semicondu	ictor	IDE		<u> </u>				- 🗆 X
File Edit Source Refactor Navigate	Search Project Run Wi	ndov	r Help						
5 0 <b>•</b> ·	No Launch Configurations		~ on:		* # C	•	5 - B D X 2	1 · 61 · 2 · 0	
₩									Quick Access 🗄 😰   🔜 🎰
🖕 Project Explorer 😂 👘 🗖	♦ blinky.rteconfig 😫							- 0	2: Out. 21 @ Tes. @ Bui. 12 Do. " 0
8 2 2 7	♦ Components [2] Res							0	p v
✓ i blinky		Carlos	144,00,010	100000	144.0000	Les Monte and			There is no active editor that provides an
> 🗊 Includes	Software Components	Sel.	Variant	ONEcomicondu	version	Description	40 MILLS 24 PO DAMA 20	0 20 0000	outline.
> 😂 include	CMSIS Driver			Orssemicondu	c .	Anim Cortex-M3	40 MH2, 24 KB IDAM, 30	O KO KUMI	
> E RIE	v      Device								
> is app.c	> * Bluetooth Core (	,			1.0.0	BLE Core implem	nentation		
Blinky.rtecontig	> * Bluetooth Profile					1			
E readme_blinky.md	<ul> <li>Libraries</li> </ul>								
	Calibrate		source	ONSemicondu	c 3.1.575	Calibration Libra	ry (libcalibratelib)		
	Custom Proto		source	ONSemicondu	c 3.1.575	Low Latency Aud	tio Streaming Custom P	rotocol Library (libc	
	DMA		source	ONSemicondu	c 3.1.575	DMA Driver for R	RSL10		
	Flash		source	ONSemicondu	c 3.1.575	Flash Library (lib)	flashlib)		
	GPIO		source	ONSemicondu	c 3.1.575	GPIO Driver for R	RSL10		
	Math		source	ONSemicondu	c 3.1.575	Math Library (libr	mathlib)		
	Remote_Mic		source ONSemiconduc 3.1.575 Remote Microphone Library (libremote_micLib)						
	System	Ø	source	ONSemicondu	c 3.1.575	System Macros a	nd Library (libsyslib)		
	✓ Startup	Ø	source	ONSemicondu	c 3.1.575	System Startup for	or ON Semiconductor R	SL10	
	Validation Output Description								
	Components Device Packs	5							
	🗄 Problems 🕮 🕘 Tasks 🛛	Co	nsole 🔟 Prop	erties					> ≥ ≥ ≥ ○ □
	0 items	~		1.255	0.000	S. 1963	109000-5-80	157202	
	Description			Res	ource	Path	Location	Туре	
( )									

## Figure 6. RTE Configuration for the Blinky Example Project in the onsemi 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.

File Edit	s	ource Refactor Navigate	Search Project Run Wind	iow He	elp						
(3)0	1	• ··· · ·	No Launch Configurations	Ý	on:		~ 🌣 i c	• 🔛 🐚   🕲 + 🐐 +	a   🛛 i 🗙 i 🕹 i 🖉	• • 63 • 🖻 • G	• ] \$\$ • O • 9; • 9 <b>.</b> • ] 🕭 😂 🛷 •
Ε- (i) π	: 2										Quick Access 🕴 😨 🔯 🌚
Project	t Ex	plorer 🛱 🗖 🗖	blinky.rteconfig							- 0	문 Out 23 팀 Tas ⑧ Bui 范Do 『 다
		New	)	E.						0 🔛	Şı ⊽
~ 😂 bl		Go Into		el. Var	iant	Vendor	Version	Description			There is no active editor that provides an
> 🔊		Open in New Window				ONSemiconduc		ARM Cortex-M3 48 MHz	2 24 KB RAM, 388 I	KB ROM	outline.
> 📩		Show in Local Terminal	>								
> 🖻	D	Сору	Ctrl+C				100	RLE Core implementatio	20		
\$	10	Paste	Ctrl+V				1.0.0	Dec core implementatio	<u>///</u>		
	2	Remove from Context	Ctrl+Alt+Shift+Down								
		Source	>	sou	rce	ONSemiconduc	3.1.575	Calibration Library (libca	alibratelib)		
		Move		sou	rce	ONSemiconduc	3.1.575	Low Latency Audio Stree	aming Custom Prot	tocol Library (libc	
		Rename	F2	sou	rce	ONSemiconduc	3.1.575	Elash Library (libflashlib)	\ \		
	ы	Import		sou	rce	ONSemiconduc	3.1.575	GPIO Driver for RSL10	<u>.</u>		
	4	Export		sou	rce	ONSemiconduc	3.1.575	Math Library (libmathlib	<u>)</u>		
	٠	CMSIS C/C++ Project	>	sou	rce	ONSemiconduc	3.1.575	Remote Microphone Lib	arary (libremote_mi	icLib)	
		Build Project		sou	rce	ONSemiconduc	3.1.575	System Macros and Libr	ary (libsyslib)	14	
	ຄ	Clean Project	55	sou	rce	ONSemiconduc	3.1.575	System Startup for ON S	emiconductor RSL	10	
	•	Close Project				Des	cription				
		Close Unrelated Project		L							
		Build Targets	>	-							
		Index	>	-							
		Build Configurations	>								
		Profiling Tools	>								1
	0	Run As	>	onsole	Proper	rties					\$ <u></u>
	#	Debug As	,	-							
		Restore from Local History	,			Reso	urce	Path	Location	Туре	
C hlat	*	Run C/C++ Code Analysis									
C Dlinky		-									

#### Figure 7. Starting to Build a Project in the onsemi IDE

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

Flash LI source UNSemic	onduc: 5.1.575 J Hash Library (libflashlib)	*	
Validation Output	Description		
Components Device Packs			
🗈 Problems 🐵 Tasks 🖾 Console 👯 🗔 Properties			0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
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-sizeformat-berkeley "blinky text data bss dec hex file 4204 100 1036 5340 14dc blin Finished building: blinky.siz	.elf" name ky.elf		
13:52:29 Build Finished. 0 errors, 0 warning	s. (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
  - *blinky.elf*: Arm<sup>®</sup> 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:

- Within the **Project Explore**, right-click on the *blinky.elf* file and select **Debug As > Debug** 1. 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							
type filter text	📄 Main 🏇 Debugge	er 🕨 Sta	rtup 🦆 Sourc	e 🗖 Comme	on 🚼 SVD Pat	h		
C/C++ Application	J-Link GDB Server S	etup						
C/C++ Attach to Application	Start the J-Link	GDB server	locally		Connect	t to running targ	get	
C/C++ Postmortem Debugge	Executable path:	\${jlink_pa	th}/\${jlink_gdt	server}			Browse	Variables
C/C++ Remote Application	Actual executable:	erCL.exe	æ					
Cti C/C++ Unit © GDB Hardware Debugging © GDB OpenOCD Debugging ~ © GDB SEGGER J-Link Debuggir		(to change	e it use the glo	oal or works	pace preferenc	es pages or the	project pro	perties page)
	Device name:	RSL10					Supported of	<u>device names</u>
	Endianness:	Little	OBig					
C blinky	Connection:	● USB	O IP		(US	B serial or IP na	ame/address	)
Launch Group	Interface:	● SWD	◯ JTAG					
Launch Group (Deprecated)	Initial speed:	⊖ Auto	🔿 Adaptiv 🖉	Fixed 1	000 kHz			
	GDB port:	2331						
	SWO port:	2332	]		🗹 Verify d	ownloads 🗹 Ir	nitialize <mark>reg</mark> i:	sters on start
	Telnet port:	2333			🗹 Local he	ost only 🗌 S	ilent	
	Log file:							Browse
	Other options:	-singleru	n -strict -timeo	ut 0 -noqui				
	Allocate consol	e for the G	DB server	ato nogu	Allocate co	nsole for semiho	osting and S	wo
	CDR Client Setur		DD Server	L		isole for seriilik	osting and s	
	GDB Client Setup	flanar -	na fini a alla fi /	an auffini			Description	Madablas
			the second se					

### 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 36.
- 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, "onsemi IDE and RSL10 CMSIS-Pack Installation Procedures" on page 8).

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 45.

- 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/a	pp.c - ON Semiconductor IDE -	. 🗆	×
<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		
🔦 🔯 🔳 🔅 Debug	✓ C blinky ✓ 🔅 🖄 ✓ 🔚 👘 📄 🗮		
🔪   📭 💷 🖧 🖓 🔳 💷 🔺   🖌	5 ∞   &   ☆ • O • • • • / ● ⊝ <i>&gt;</i> •   ⊿ ⇒ ! ⊴ • 전 • や ⇔ • → •		
	Quick Access	😰   🔤	🔅 🗞
🏇 Debug 🛛 🏠 Project Ex 🖳 🗖	🚸 blinky.rteconfig 💽 app.c 🐹 💽 0x0 🖓 🗖	3 ¥4 1	
<ul> <li>kit </li> <l< th=""><th><pre>106 * 107@ int main(void) 108 { 109</pre></th><th>Kan oti i Cin oti i ✓ Name</th><th>►  </th></l<></ul>	<pre>106 * 107@ int main(void) 108 { 109</pre>	Kan oti i Cin oti i ✓ Name	►
	🖳 Console 🕱 👯 Registers 🖹 Problems 🕥 Executables 🙀 Debugger Console 🔋 Memory		- 8
	= × 🔆 🗟 🖉	🛃 📮 🔻	<b></b> 🛨
	blinky[GDBSEGGERJ-LinkDebugging]JLinkGDBServerCL.exe Breakpoint reached @ address 0x00100400 Reading all registers Read 4 bytes @ address 0x00100400 (Data = 0xF7FF2006)		^ ~
< >>	<		>
	Writable Smart Insert 124 : 1		100

### Figure 11. Debug Perspective

## 3.4.2 Peripheral Registers View with the onsemi IDE

The onsemi 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 14.
- Change to the SVD Path tab, and set the path to the *rsl10.svd* file as C:\Users\<username>\AppData\Local\Arm\Packs\ONSemiconductor\RSL10\<version>\svd\rsl10.svd (see Figure 12). Click Debug.

Debug Configurations							×
Create, manage, and run configu	rations					Ŕ	Š
Image: Second Secon	Name: blinky Main * Debugger SVD file (used by the per File path: C:\Users\ <ul> <li><use< li=""> </use<></li></ul>	• Startup ( Source C Cor ripheral registers viewer) \ON_Semiconductor 	nmon 🕞 SVD Path	<version></version>	Browse	Variables	
Filter matched 12 of 12 items				Reve	rt	Apply	
?				Deb	ug	Close	

### Figure 12. SVD Path Tab Debug 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.
- 6. Press F6 to step through the code. You can observe that this register's bit 6 toggles its state when Sys\_GPIO\_Toggle(LED\_DIO) is executed (in this case, from 0xF060 to 0xF020). The register turns yellow to indicate that you have activated real-time monitoring for it (see Figure 14 on page 18).

<pre>PD Deg 10 _ project (</pre>	🐔 🏘 🔳 🎄 Debug	🗸 💽 blinky Debug 🚽 🔅 📑 🐨 🗟 🕲 🗽 🕨 🔳 🕅 🖎 🔅	s 18   🖬 🗮 🕱   4	ار 😅 🔹 ۹۰ ۲ 🕐 ۲ 🗴	P • [ 1 ] = [ 2] •	δ] • ♥= Φ • Φ • Φ •		Quick Access 🔡 😨	<b>8</b> 9
<pre>% [1 + %]</pre> <pre>% [1</pre>	to Debug 22 ha Project	🗄 blinky.rteconfig 💽 app.c 🐹 💽 0.0	(+)+ Variables Bre	akpoints 😽 Expressions 🛋 Modul	es 😤 Peripherals 🕽	1		80 🔩 B 🗵	- 0
#       #	i+     >       ibinity beta (0500H).i-ink1       >ibinity beta (0500H).i-ink1    <	<pre>64 * single interrupt event accurs for each push of the pushbatton. * * The debonce circuit always has to be used in combination with the 6 * transition made to deal with the debonce circuit limitations. * A debonce filter time in the debonce circuit limitations. * A debonce filter time in the debonce circuit limitation. * Sys_DOD_config(BVTON_DDD, DD) push (DD) push (</pre>	Peripheral	Address Description Gw40001500 Baseband Co Gw40001400 Baseband Co Gw40001900 Clock General Gw40000100 CPC General Gw40000100 DP0 Debug Contr Gw40000000 DP0 Dio Interface Gw40000000 DMA Control	ntroller introller Interface tion or Control oller and Digital Pad contr ller Configuration and	rol I Control			
**       *       **       *       *		96 97 printf_init();	Memory 32					Se Outline 12	
100       100       100       Patter       Mark       Water         100       100       Patter       Mark       Water       Water       Water         100       Patter       Mark       Mark       Mark       Water       Water       Water         100       Patter       Mark       Mark       Mark       Mark       Mark       Water       Wate		98 99 /* Uneask all interrupts */	Monitors 💩 👷 🎉	🕞 DIO: 0x40000700 🖄 🍐 Nev	v Renderings			D D R N V + I	* *
111       1111       111       111		100set_PRIPASK(PRIPASK_ENABLE_INTERRUPTS);	DIO	Peninter	Address	Value		app.h	
138       * Function       1 ist main(void)       0000000       000000000       000000000       00000000       00000000       00000000       000000000       00000000       00000000       00000000       000000000       000000000       000000000       00000000       000000000 <td></td> <td>101 )</td> <td>CLK</td> <td>- DO</td> <td>0-40000700</td> <td>1000</td> <td></td> <td>printf.h</td> <td></td>		101 )	CLK	- DO	0-40000700	1000		printf.h	
101       * Faction       1 is tail(vid)       0000 (press to taggle low (secontrolled by 0000 (press to taggle low (secontrolled by 00000 (press to taggle low (secontrolled by 0000 (press to taggle low (secontrolled by 00000 (press to taggle low (secontrolled by 000000 (press to taggle low (secontrolled by 00000000 (press to taggle low		1038 /*		DIO CEGI	0x40000700			<ul> <li>DIO0_IRQHandler(void) ::</li> </ul>	roid
<pre></pre>		184 * Function : int main(void)		V IIII DIO DATA	0x40000740	0x0000F060		Initialize(void) : void	
100       100       100       100         100       100       0.000       0.0000         100       0.000       0.0000       0.0000         100       0.000       0.0000       0.0000         100       0.000       0.0000       0.0000         100       0.000       0.0000       0.0000         100       0.000       0.0000       0.0000         101       100       0.000       0.0000       0.0000         101       100       0.000       0.0000       0.00000         101       100       0.000       0.00000       0.000000         101       100       0.0000       0.000000       0.000000         101       100       0.00000       0.0000000       0.0000000         101       100       0.0000000       0.0000000       0.00000000         101       100       0.0000000       0.00000000       0.00000000         101       100       0.0000000       0.00000000       0.00000000         101       0.00000000       0.00000000       0.00000000       0.00000000         101       0.00000000       0.00000000       0.00000000       0.00000000       0.0000000000		105 *			[15:0]	0.4060	10 0000	<ul> <li>main(void) : int</li> </ul>	
<pre>100 * I puts : ince 100 * Outputs : in</pre>		107 * DIOS (press to toggle input/output).		e GPIO	[15:0]	0xF060			
100 * 0.04pts       1100         111 * Antipetitions       100         112 * Interministic methods       100         113 * Interministic methods       100         114 * Interministic methods       100         115       100       100         116       100       100         117       100       100         118       100       100         119       100       100         110       100       100         111       100       100         111       100       100         111       100       100         111       100       100         111       100       100         111       100       100         111       100       100         111       100       100         111       100       100          111       100       100         111       100       100         111       100       100         111       100       100         111       100       100         111       100       100       100         1110 </td <td></td> <td>108 * Inputs : None</td> <td></td> <td>&gt; ### DIO_DIR</td> <td>0x40000744</td> <td>0x00008040</td> <td></td> <td></td> <td></td>		108 * Inputs : None		> ### DIO_DIR	0x40000744	0x00008040			
<pre>111 * All total _ look 112 * All total _ look 113 * All total _ look 114 * All total _ look 115 * All total _ look 115 * All total _ look 116 * All total _ look 117 * Initialize to system // 118 * All total _ look 119 * All total _ look 110 * All total _ look 111 * All total _ look 111 * All total _ look 112 * All total _ look 113 * All total _ look 114 * All total _ look 115 * All total _ look 115 * All total _ look 116 * All total _ look 117 * Initialize to system // 118 * All total _ look 119 * All total _ look 110 * All total _ look 111 * All total _ look 111 * All total _ look 112 * All total _ look 113 * All total _ look 114 * All total _ look 115 * All total _ look 115 * All total _ look 116 * All total _ look 117 * Total total _ look 118 * All total _ look 119 * All total _ look 110 * All total _ look 111 * All total _ look 111 * All total _ look 111 * All total _ look 112 * All total _ look 113 * All total _ look 114 * All total _ look 115 * All total _ look 115 * All total _ look 115 * All total _ look 116 * All total _ look 117 * Total total _ look 118 * All total _ look 119 * All total _ look 110 * All total _ look 111 * All total _ look 112 * All total _ look 113 * All total _ look 114 * All total _ look 115 * A</pre>		109 " Outputs : None		> IIII DIO_MODE	0x40000748	0x00000060			
<pre>110 int main(veid) 113 { 113 { 114 { 115 int main(veid) 115 { 115 int main(veid) 115 { 115 int main(veid) 115 { 116 int main(veid) 115 int ma</pre>		110 Assumptions : None		> M DIO_INT_CFG[]	0x4000074C				
133 {       /*initialize global variables */       >		112@ int main(void)		> ## DIO_INT_DEBOUNCE	0x4000075C	0x00000131			
<pre>114 //initialize global veriables */ 114 det_gegige_status + 1; 115 // initialize the system // 114 det_gegige_status + 2; 115 // initialize the system // 114 det_gegige_status + 2; 115 // initialize the system // 115 // initialize the system // 115 // initialize the system // 116 // initialize the system // 117 // initialize the system // 118 // initialize the system // 118</pre>		113 {		> IIII DIO_PCM_SRC	0x40000760	0x00111111		-	
113       114       115       1		114 /*Initialize global variables */		> M DIO_SPI_SRC[]	0x40000764				
<pre>11</pre>		<pre>115 led_toggle_status = 1;</pre>		> IIII DIO_UART_SRC	0x4000076C	0x00000011			
113       1111       111       111		117 /* Initialize the system */		> ###_DIO_I2C_SRC	0x40000770	0x00001111			
<pre>119 PHINT('DNGCE DATIALTEDN'); 120 PHINT('DNGCE DATIALTEDN'); 121 DD_MC_MCE DATIALTEDN'); 122 PHINE DD_MCE DATIALTEDN'); 123 PHINE DD_MCE DATIALTEDN'); 124 PHINE DD_MCE DATIALTEDN'); 125 PHINE DD_MCE DATIALTEDN'; 126 PHINE DD_MCE DATIALTEDN'; 127 PHINE DD_MCE DATIALTEDN'; 128 PHINE PHINE DD_MCE DATIALTEDN'; 129 PHINE PH</pre>		<pre>118 Initialize();</pre>		> IIII DIO_AUDIOSINK_SRC	0x40000774	0x00000011		-	
133       //* Spin-loop */       300 0000000000000000000000000000000000		<pre>119 PRINTF("DEVICE INITIALIZED\n");</pre>		> IIII DIO_NMI_SKC	0x40000778	0x00000030			
122       wklif(1)       3       0/01/95/95/96       00000118       00000110         123       (**sfreshift(1)       3       0/01/95/95/96       00000118       00000100         123       (**sfreshift(1)       3       0/01/95/95/96       00000118       00000100         124       (**sfreshift(1)       3       0/01/95/95/96       00000118       00000100         125       (**sfreshift(1)       3       0/01/95/95/96       0000010       00000100         125       (**sfreshift(1)       3       0/01/95/95/96       0000010       00000100         126       (**sfreshift(1)       3       0/01/95/95/96       00000010       00000100         126       (**sfreshift(1)       3       0/01/95/95/96       00000010       00000100         127       (**sfreshift(1)       0/01/95/95/96/96       00000010       00000000       00000000         128       0/01/96/95/96/96/96       00000010       00000000       00000000       00000000         128       0/01/96/96/96/96/96       00000010       00000000       00000000       00000000         128       0/01/96/96/96/96/96       00000000       00000000       00000000       00000000         128       0/01/96/96/96/96		128 121 /# Sola loop #/		5 III 00,88,00,580	0.10000776	0.000121212			
123     ( ************************************		122 while (1)		> ## 00.86.5PLSKC	0-10000784	0.00000012			
124       /* #efresh the matchding timer -/       >		123 (		> 111 DIO RE GRIGOD ERC	0.40000784	0.00121212			
10       5%_juktobagkfreek();         11       00_jktobag_juktobag_kfreek();         12       5%_juktobag_kfreek();         13       5%_juktobag_kfreek();		124 /" Refresh the watchdog timer "/		> ### DIO_RF_GPIOUS_SRC	0-40000786	0x12121010			
127     ** Toggle GFD 0 (ff foggling is mabled) then wait 0.5 seconds */     > 110 Op/MaC/SEC     \$00000010       128     \$0000010     \$00000111       129     \$100 Op/MaC/SEC     \$00000010       129     \$100 Op/MaC/SEC     \$00000010       129     \$100 Op/MaC/SEC     \$00000010       129     \$100 Op/MaC/SEC     \$00000010       129     \$110 Op/MaC/SEC     \$00000010       129     \$110 Op/MaC/SEC     \$00000010       129     \$110 Op/MaC/SEC     \$00000010       120     \$110 Op/MaC/SEC     \$00000010       121     \$110 Op/MaC/SEC     \$00000010       122     \$110 Op/MaC/SEC     \$00000010       123     \$110 Op/MaC/SEC     \$00000111       124     \$110 Op/MaC/SEC     \$00000014       125     \$110 Op/		125 Sys_Watchdog_Refresh();		> 111 DIO_RF_GPIO47_SRC	0.40000780	0,00001010			
123     {f (led_teggit_states = 1)     > (model() > (tool set() >		125 127 /* Togela (PIO 6 (if togeling is apphled) then wait 0.5 seconds */		5 111 DIO_RP_GPI089_SKC	0.10000790	0.00001010			
123 {		128 if (led toggle status == 1)		> III DIO DOME SHE	0-10000794	0.000011111			
0         5y		129 (		5 III 010_0P05P32_1146_5KC	0.10000798	0.000111111			
111         )         PREMTY(*L05 % to ', '(05CFG(LtD_DD0) & Act )* '00* '; '00*''));         )         )         )         )         00000000           101		Sys_GPI0_Toggle(LED_DIO);		S III DIO SIAG SNI PAD CPG	0.40000790	0.0000000			
132     else     > III 00_90_010     0000000       135     { 5ys_0010_5et_Lew(k0_0000);     0000000     0000000       137     5ys_0010_90((uint32,t)(0.5 * SystemCoreClock));     0     0000000		131 PRINTF("LED Xs\n", (DIO->CF6[LED_DIO] & 0x1 ? "ON" : "OFF"));		> III DIO EXICLE CFG	0x40000740	0x0000001		-	
134         (same           135         (sys_0P10_5et_Low(L00_D10);           136         )           137         Sys_Delay_ProgramBOH((uint32_t)(0.5 * SystemCoreClock));           138         )		132 }		> IIII DIO_WAD_CFG	0540000744	0x0000000			
105         \$ys_0P40_54t_Lew(U6_000);           107         \$ys_0P40_yProgramBOR((uint32_t)(0.5 * SystemCoreClock));           108         }		134 (							
136 } 137 Sys_Delay_ProgramBOH((uint32_t)(0.5 * SystemCoreClock)); 138 }		135 Sys_GPI0_Set_Low(LED_DI0);							
137     Sys_DeLay_ProgramBOH(uint32_t)(0.5 * SystemCoreClock));       138     }		136 }							
		<pre>137 Sys_Delay_ProgramROW((uint32_t)(0.5 * SystemCoreClock));</pre>							
		199 3							
140		140						1	

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.

on-semiconductor-workspace - blinky/a	app.c - ON Semiconductor IDE				
File Edit Source Refactor Navigate	Search Project Run Window Help				
🐔 🎄 🔳 🎋 Debug	E blinky Debug	s.e   🗰 🗟 👷 🕹	b 🕸 • O • 💁 • 🧶 🖉	-110-8	- (+ <b>(+ (+</b> ) +
🍁 Debug 😫 🏊 Project 📟 🗖	💠 blinky.rteconfig 🛛 👔 app.c 😫 💽 0x0 📟 🗖	(*)* Variables 🔍 Brea	skpoints 🕂 Expressions 🛋 Module	🛚 👷 Peripherals 😒	
	4 * single interrupt event occurs for each push of the pushbutton. 5 * The debungs circuit always has to be used in combination with the 8 * transition and to deal with the debungs circuit limitations. 7 * A debungs filter time of 50 mg is used. */ 5 \$ys_002 config(ent00, D0, D00 y00 gr00_TM, 0   D00_MBAK_PULL_UP   100_UP (D15AHL); 5 \$ys_002_Inticonfig(0, 100_UP (D15AHL); 5 \$ys_002_Inticonfig(0, 1	Peripheral	Address Description Dx40001500 Baseband Con Dx40001400 Baseband Con Dx40000100 Clock Generat Dx40000F00 CRC Generato Dx6000EPD Debug Contro Dx40000200 Reset	and stroller introller Interface sion r Control iller	
Semihosting and SWV	93 DIO_DEBOUNCE_ENABLE,	2 NO	0x40000700 DIO Interface a	and Digital Pad control	
	94 DIO_DEBOUNCE_SLOWCLK_DIV1024, 49);	DMA	0x40000600 DMA Controll	er Configuration and Cor	ntrol
	96 WVIC_EnableINQ(0100_INQN);				
	<pre>97 printf_init();</pre>	1 Memory 23			my um 📑 🛃 🚳 🗄 🐐 🔻 🖓 🗖
	90 99 /* Unmask all interrupts */	Monitors 🖕 💥 💥	😪 DIO: 0x40000700 😢 🔶 New	Renderings	
	<pre>100set_PRIMASK(PRIMASK_ENABLE_INTERRUPTS);</pre>	DIO	Register	Address	Value
	101 }	CLK	✓ 5 DIO	0x40000700	
	1030 /*		> 111 DIO_CFG[]	0x40000700	
	104 " Function : int main(void)		V III DIO_DATA	0x40000740	0x0000F060
	105 *		S DIO	[15:0]	0xF060
	107 * DIOS (press to toggle input/output).		S GPIO	[15:0]	0x0: GPI00 LOW V
	108 * Inputs : None		> IIII DIO_DIR	0x40000744	Dec GPIOLOW
	109 ° Outputs : None		> INF DIO INT CEGI	0.40000748	0x0: GPIO2_LOW
	110 Assumptions : None		S HIT DIO_INT_COUNCE	0x4000074C	0x0: GPIO3_LOW
	1120 int main(void)		> IIII DIO PCM SRC	0x40000760	0:0: GPIO4_LOW
	113 {		> M DIO SPI SRCI1	0x40000764	0x0: GP106 LOW
	114 /-Initialize global variables -/ 115 led topple status = 1:		> ## DIO_UART_SRC	0x4000076C	0x0: GPI07_LOW
	116		> ### DIO_I2C_SRC	0x40000770	0.0: GPIO8_LOW
	117 /* Initialize the system */		> IIII DIO_AUDIOSINK_SRC	0x40000774	0x0: GPI010_LOW
	118 Initialize(); 110 DETWIE("DEVICE INITIALIZED\=");		> ### DIO_NMI_SRC	0x40000778	0x0: GPI011_LOW
	120		> ### DIO_BB_RX_SRC	0x4000077C	0x0: GPI012_LOW
	121 /* Spin loop */		> ### DIO_BB_SPI_SRC	0x40000780	Dx0: GPI013_LOW
	122 uhile (1)		> IIII DIO_RF_SPI_SRC	0x40000784	0x0: GPI015_LOW
	123 1 124 /* Refresh the watchdog timer */		> DIO_RF_GPIOO3_SRC	0x40000788	0x1: GPI00_HIGH
	125 Sys_Watchdog_Refresh();		> IIII DIO_RF_GPIO47_SRC	0x4000078C	0/2: GPI01_HIGH
	126		> 100 DMIC SPC	0-40000790	0x8: GPIO3 HIGH
	127 /" Toggle GPIO 6 (if toggling is enabled) then wait 0.5 seconds "/		> HIT DIO I PDSP32 ITAG SPC	0-40000798	0x10: GPIO4_HIGH
	129 {		> IIII DIO JTAG SW PAD CFG	0x4000079C	0x20: GPIO5_HIGH
	3130 Sys_GPIO_Toggle(LED_DIO);		> IIII DIO EXTCLK_CFG	0x400007A0	0x80: GP107, HIGH
	131 PRINTF("LED %s\n", (DIO->CFG[LED_DIO] & 0x1 ? "ON" : "OFF"));		> III DIO_PAD_CFG	0x400007A4	0x100: GPI08_HIGH ~
	T T				

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

p.c - ON Semiconductor IDE				
vice crimer and the construction of the constr	🐢e.] 🗰 🗟 👷 💊	5 🕸 • O • 💁 • 🙋 🖨 🖉	P • ]] > ] 2 •	§ < ♥ ♥ ♥ ♥ ♥ ♥
blinky.rteconfig 🛛 app.c 💱 💽 0x0 😁 🖸	(*)= Variables 💁 Brea	akpoints 🕂 Expressions 🛋 Modu	es 📆 Peripherals 😒	
64     * single intervat event occurs for each push of the pushbutton.       75     The debugge circuit always has to be used in combination with the * transition mode to deal with the dobugget circuit listations.       85     \$y_000_config(100_000_000_000_000_000_000_000_000_000	Peripheral	Address         Description           0x40001500         Baseband Cc           0x40001600         Baseband Cc           0x400001600         Baseband Cc           0x400001600         Clock Generat           0x40000000         Clock Generat           0x40000000         Reset           0x40000000         Reset           0x40000000         DIO Interface           0x40000000         DIMA Control	ntroller ntroller interface tion or Centrol oller and Digital Pad contro ler Configuration and t	i de control
97 printf_init(); 98	Memory 23	(7)		
<pre>99 /* Umsak all interrupts */ 10 _set_PRIMSK(PRIMSK_DNALL_INITERRUPTS); 101 } 103 /* 104 /* 105 ** 105 ** 105 ** 105 ** 105 ** 105 ** 105 ** 105 ** 106 ** 106 ** 107 ** 107 ** 108 ** 108 ** 108 ** 109</pre>	ooneng ⊕ X %	Initia beloacente         ▲ Net           Pegister         ■         ■         ●         Net           > Mb Dio, Crédi         ■         ●         Net         Dio           > Mb Dio, Crédi         ■         ●         Net         Dio           > Mb Dio, Crédi         ■         ●         Net         Dio           > Mb Dio, Mode         ■         Dio         Mode         Net           > Mb Dio, Mode         ■         Dio         Net         Net           > Mb Dio, Mode         ■         Dio         Net         Net           > Mb Dio, Mede         ■         ■         Dio, Mede         Net           > Mb Dio, Mede         ■         ■         Dio, Mede         Net           > Mb Dio, Mede         ■         ■         Dio, Mede         Net           > Mb Dio, Mede         ■         ■         ■	V Renderings     Address     Addres     Address     Address     Address     Addres	Value           0x0000000 ortoo         1111 0000 d[te 0000           0x0000000         0x000000           0x0000011         0x000011           0x0000011         0x000011           0x0000011         0x000011           0x0000010         0x000011           0x0000011         0x0000011           0x0000011         0x0000010           0x0000012         0x0000012           0x000010         0x000010           0x000010         0x000010           0x000010         0x000000           0x000000         0x000000
	<pre>c-ONSemiconductorUS arc Number Netp 'C DiskyperSeduy 'S Select Second Seco</pre>	<pre>c - ON Semiconductor UE and Project Num Window Help 'C SubmyOrks * Single Interrupt cvmt occurs for each path of the publicitum. * single Interrupt cvmt occurs for each path of the publicitum. * a debunce filter the of He gits have. * y_BOIC Config(ENTON_DID. DOD_ONG_GROUPL_) [DID_MEAK_PULL_UP ] DOD_EVENT THANSTICM   DID_SAC(ENTION_DID)   DID_EVENT THANSTICM   DID_SAC(ENTION_DID)   * griant_init. * for the optimum of the state of the gits have. * for the optimum of the state of the gits have. * DIDE (PRESENCE, DUBLE_INTERSUPTS); * DIDE (PRESE to toggle input/output). * DIDE (PRESE to toggle input/output). * for matching is home * click * for the state of the stat</pre>	<pre>c - ON Semiconductor US and Project Num Vindow Help</pre>	<pre>c - ON Semiconductor US and Project Num Vindow Help  V C Entrop Usag  * A deface interrupt rent occurs for Each pub of the publicition</pre>

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

## **CHAPTER 4**

## **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.
- 2. 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.
- 3. Make sure your J-Link software is version 7.66b or higher.

### 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.

F	ile	Edit	View	Project	Flash	Debug	Peripherals	Tools	SVCS	Win
		<u>i</u>		8 h 🕻	3 9	• ا	->   🏞 🕅	内内		۰ <b>₽</b>
	٩		🗄 🧼 •	- 🚉 🕌	Targ	et 1	- 8	रे। 🚠 🗟	• 🔶 🔶	> 💑

### 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
	Computer
	Organize • New folder
	Favorites Name Date modified Type
	Desktop ONSemiconductor.RSL10.2.2.347 10/31/2018 4:29 PM uVisio
Pack Installer - C:\Keil_v5\ARM\PACK	Downloads     Secent Places
Refresh Import Manage Local Repositories Exit Exit Exit Exit Exit Exit Exit Exit	☐ Libraries ③ Documents ④ Music ⑤ Pictures ⑤ Videos
	New Computer
	📲 SYSTEM (C:) 🗸 📢 👘
	File name: ONSemiconductor.RSL10.2.2.347    Software Pack - PACK (*.zip;   Open  Cancel

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

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

🛞 I	Pack Installer - C:\Keil_v5\ARM\PACK		
File	Packs Window Help	_	
	Refresh		
	Import	Þ	4
	Import from Folder		F-
	Manage Local Repositories		Pack
	Evit	Summary	De

### Figure 18. Refresh Pack after installation

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

Devices Deards		4	Packs Examples		
Search: • X 🖻			Show examples from installed Packs of	nly	
Device /	Summary		Example	Action	Description
🗉 🍳 Maxim	9 Devices	<b></b>	ADC_UART (RSL10 Evaluation Board)	🔶 Сору	ADC with UART S
🗉 🌳 MediaTek	2 Devices		ble_central_client_bond (RSL10 Eval	🔶 Сору	BLE Central Clien
Microchip	345 Devices		ble_central_client_scan (RSL10 Eval	🔶 Сору	Pairing and Bond
🗉 🌳 Microsemi	6 Devices		ble_peripheral_server_bond (RSL10	🔶 Сору	BLE Peripheral Se
MindMotion	2 Devices		ble_peripheral_server_hrp (RSL10 Ev	. 🗇 Сору	Pairing and Bond
Nordic Semiconductor	13 Devices		blinky (RSL10 Evaluation Board)	🚸 Сору	Blinky GPIO I/O S
Nuvoton	487 Devices		default_MANU_INFO_INIT (RSL10 E	🔶 Сору	Default System I
	1223 Devices		hci_app (RSL10 Evaluation Board)	🔶 Сору	Host Controller I
ONSemiconductor	1 Device		i2c_cmsis_driver (RSL10 Evaluation	🔶 Сору	I2C CMSIS-Driver
RSL10 Series	1 Device		kernel_timer (RSL10 Evaluation Boa	🚸 Сору	Kernel Timer San
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 Device
🗉 🔗 Renesas	4 Devices		spi_cmsis_driver (RSL10 Evaluation	🔶 Сору	SPI CMSIS-Driver
🗉 🔗 Silicon Labs	783 Devices		supplemental_calibrate (RSL10 Eval	🔶 Сору	Default System I
Sinowealth	1 Device		uart_cmsis_driver (RSL10 Evaluation	🔶 Сору	UART CMSIS-Dri
🗉 🔮 SONIX	50 Devices				
CTMicroelectropics	1061 Devices				
STRUCTORIECTIONICS					

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

### 4.3 BUILDING YOUR FIRST SAMPLE APPLICATION WITH THE KEIL UVISION IDE

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

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

### 4.3.1 Import the Sample Code

To import the sample code:

- 1. In the Pack installer, click on the **Examples** tab to list all the example projects included in the RSL10 CMSIS-Pack.
- 2. Choose the example project called *blinky*, and click the **Copy** button to import it into your workspace (see Figure 20). 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									
<ol> <li>Image: Second sec</li></ol>	V No Launch Configurations	🗸 on: 🗸 🏘 🕅	<b>*</b> 🛛							
৶ৼয়ৼড়৾৾ড়ৼ৾৾৽ৼ										
E Devices 🛛 🖪 Boards	▣ □   💥   🕐 ▽ □ 🗆	Packs ■ Examples ※								
Search Device		Only show examples from								
Device	Summary	Search Example								
🗸 🍄 All Devices	1 Device	Example								
<ul> <li>ONSemiconductor</li> </ul>	1 Device	ADC UART (RSL10 Evaluation Board)	😻 Cop							
🗸 🏄 RSL10 Series	1 Device	aes128 (RSL10 Evaluation Board)	Cop							
RSL10	ARM Cortex-M3 48 MHz, 24 KB RAM, 3	ble_android_asha (RSL10 Evaluation Between the second s	🔶 Cop							
		ble_central_client_bond (RSL10 Evaluation)	🔶 Cop							
		ble_central_client_scan (RSL10 Evaluation)	😻 Cop							
		ble_central_peripheral (RSL10 Evaluation)	🔅 Cop							
		ble_peripheral_server_bond (RSL10 Evaluation of the server_bond)	🗢 Cop							
		ble_peripheral_server_hrp (RSL10 Evalu	🔅 Сор							
		ble_peripheral_server_hrp_fota (RSL10	🕸 Cop							
		ble_peripheral_server_PRA (RSL10 Eval	🕸 Сор							
		blinky (RSL10 Evaluation Board)	Cop							

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).

ne col ver roject nan Deug represa ross 3703 S S S S S S S S S S S S S S S S S S S	If	*   <b>@</b>	l •   • · Ø	<b>e</b>  (		×
<ul> <li>Project Diministry</li> <li>Target 1</li> <li>Source</li> <li>app.c</li> <li>include</li> <li>app.h</li> <li>Device</li> <li>rs10_protocol.c (LibrariesSystem)</li> <li>rs10_sys_asrcc (LibrariesSystem)</li> <li>rs10_sys_crcc (LibrariesSystem)</li> <li>rs10_sys_power_(LibrariesSystem)</li> <li>rs10_sys_timersc (LibrariesSystem)</li> <li>rs10_sys_timersc (LibrariesSystem)</li> <li>rs10_sys_timersc (LibrariesSystem)</li> <li>rs10_sys_timersc (LibrariesSystem)</li> <li>rs10_sys_timersc (LibrariesSystem)</li> <li>rs10_sys_timersc (LibrariesSystem)</li> </ul>	Software Component Device Startup Startup Startup Subtraries System Remote_Mic Math Kernel Fiash Custom Protocol Calibrate BLE Startup Fiash Startup Startup Startup Startup Startup Startup Startup System Startup System Startup System Startup System Startup System Startup System Startup System Startup System Startup	Sel.	Variant source s		Version 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0 5.10.1 5.46.5 7.9.0 5.12.8	Description Startup, System Sctup System Startup for ON Semiconductor RSL10 System Startup for ON Semiconductor RSL10 System Macros and Library (libsyslib) Remote Microphone Library (libsyslib) Remote Microphone Library (libremote micLib) Math Library (librathib) Each Li
Irsl10_sys_version.c (Libraries:System)     startup_rsl10.s (Startup)     system_rsl10.c (Startup)	Validation Output           Resolve         Select Packs         Deta	ils	Descript	ion	ОК	Cancel Help

Figure 21. RTE Configuration for the Blinky Example Project in the Keil  $\mu$ 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<sup>®</sup> 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	Del	oug Peripherals Tools SVCS Window He	lp 🗧
🗋 🖬 🖉 📓 🕷 🖄 🛍 👘	Q	Start/Stop Debug Session Ctrl+F5	const union gapc_d 🔻 🗟 🥐 🙋 🜖
🛛 🕙 🕮 🧼 🕶 🔛 🔤 🖬 Targe	Q	Energy Measurement without Debug	$\sim$
Project	RST	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 45.

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).



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	Peripherals
	📬 😡		X -6 6	5 9	€   ←	-> P P
٨		) 🧼 🗸		Targ	et 1	- 🕅
Proje	ct		Ŭ			

Figure 26. Download Button Not Supported for J-Link

## **CHAPTER 5**

## **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.
- 2. 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.
- 3. Make sure your J-Link software is version 7.66b or higher.

### 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	Help													_	
i 🕇	🎦 🔛		- X -	0 0   ;	DC		•	<	Q	>	\$ ►Ξ	<	Q	>	< >	0	•==	0	٠	, t	5

#### 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   □
				⊞⊟ ��≎ ≌ ≌  ? ▽
Search Pack	1			-
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 27 illustrates what the Pack Manager perspective looks like after installation.

IAR Embedded Workbench CMSIS N	/anager			_ • •
File Edit Search CMSIS Manager	Window Help			
i 🛄 🐚 i 🗣 ▼ i 🖗 ▼ i 🖗 = ∛i ▼ t⊃	\$ ▼ \$ ▼			Quick Access
	🚵 Packs 🛛 🔳 Devices 📓 Boards	E Pack Proper	rties 📑 Examples 📮 Console	0 🗉 🖓 🕹 🐸 🔟 🕐 🗸 🗖
	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	
	<ul> <li>Generic</li> </ul>		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 28). Choose a destination folder for a copy of the sample code.

🕸 Packs 📕 Devices 📕 Boards 📑 Example	s 🛿 📮 Co	nsole	🗆 🗆 Only show examples from installed packs 🛛 🦑 🍣 🐸 🔯 🗇 🌣 🗖
Search Example			
Example	Action	Description	
ADC_UART (RSL10 Evaluation Board)	🔶 Сору	ADC with UART Sample Code	
ble_central_client_bond (RSL10 Evaluation	🔶 Сору	BLE Central Client Bonding Sample Code	
ble_central_client_scan (RSL10 Evaluation	🔶 Сору	Central Device with Client Scanner Sample Code	
ble_peripheral_server_bond (RSL10 Evalua	🔶 Сору	BLE Peripheral Server Bonding Sample Code	
ble_peripheral_server_hrp (RSL10 Evaluation	🔶 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	🔶 Copy	Sleep Mode Sample Code for Peripheral Device with Serv	
peripheral_server_standby (RSL10 Evaluati	🔶 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 28 and Figure 33 on page 29).

File Edit Source Refactor Navigate Search Project	CMSIS Manager Run W	/indow Help								
	0-0-4-04	-100 m (0) (0) - 30 - 1	50							Quick Access
Project Explorer 11	88 0	blinky teconfin II							BE OLL IN CO BL. " D	
<ul> <li></li></ul>		Components      Re							(7)	
<ul> <li>Debug</li> <li>Debug</li> <li>Sindude</li> <li>RTE</li> <li>Device</li> <li>Rs110</li> <li>Rs110, protocolc (ONSemiconductor:Device</li> <li>Rs110, protocolc (ONSemiconductor:Device</li> <li>Rs110, pry, sactic (ONSemiconductor:Device</li> <li>Rs110, pry, sactic (ONSemiconductor:Device</li> <li>Rs110, pry, cordit (ONSemiconductor:Device</li> <li>Rs110, pry, cordit (ONSemiconductor:Device</li> <li>Rs110, pry, cordit (ONSemiconductor:Device</li> <li>Rs110, pry, cordit (ONSemiconductor:Device</li> <li>Rs110, pry, fasher (ONSemiconductor:Device)</li> <li>Rs110, pry, fasher (DNSemiconductor:Device)</li> <li>Rs110, pry, fasher (DNSemiconductor:Device)</li> <li>Rs110, pry, fasher (DNSemiconductor:Device)</li> </ul>	ce.Libraries.System.source) eLibraries.System.source) ice.Libraries.System.source ice.Libraries.System.source Libraries.System.source) eLibraries.System.source) eLibraries.System.source) to.Libraries.System.source)	Software Components # RSL10 Device Bluetooth Profil Ubraries Custom Prote Custom Prote Custom Prote Custom Prote Plash Fota Kernel Math Remote Mic Suitere		Variant release source source source release release source source	Vendor ONSemicondu ONSemicondu ONSemicondu ONSemicondu ONSemicondu ONSemicondu ONSemicondu	Version c c 2.4,450 c 2.4,450 c 2.4,450 c 2.4,450 c 2.4,450 c 2.4,450 c 2.4,450 c 2.4,450 c 2.4,450 c 2.4,450	Description ARM Cortex-M3 Biuetooth Stack Calibration Ubre Low Latency Au Flash Library (III) Fota Library (III) Event Kernel Ub Remote Micropic Scature Micropic	(ibblefib) (ibblefib) ary (ibcalbratelib) dio Streaming Custom i frashlib) fota) fota) bone Library (libremoto bone Library (libremoto	Protocol Library (Ploc	An outline is not available.
Istl0_sys_power.c [ONSemiconductor:Device Istl0_sys_ffle.c [ONSemiconductor:Device Istl0_sys_timers.c [ONSemiconductor:Device	vice.Libraries.System.souro :Libraries.System.souroe] vice.Libraries.System.souro	Weak_PRF     Startup		release	ONSemicondu ONSemicondu	c 2.4.450 c 2.4.450 c 2.4.450	Weak Profile Lib System Startup	for ON Semiconductor	RSL10	
ile rstill sys untr. (DNSemiconductor:Device Ibtraries System source) ile rstill sys versione. (DNSemiconductor:Device.Ibtraries.System source) ile statup. ytillos (ONSemiconductor:Device.Startup.source) ile system, ytillos (ONSemiconductor:Device.Startup.source) ile RTE Componentsh.	Validation Output	Validation Output Description								
<ul> <li>settings</li> <li>app.c</li> <li>blinky.ewd</li> <li>blinky.ewp</li> </ul>		Components Device Pack	5						_	
iii blinky.ewt		Problems II @ Tasks	C0	nsole El Pro	perties					
blinky.rteconfig     readme_blinky.tot     sections.icf		Description	*		Res	ource	Path	Location	Туре	
e	· · ·									

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

			- <	$\alpha$	- <b></b>	- <	$\bigcirc$ >	<	>		🕑 🚊 🗄
rkspace	-	пх									
	199										
bug		•									
es	\$	•									
blinky - Debug	Y										
👎 🗖 CMSIS-Pack											
RTE_Components.h											
⊢₽ ■ Device.Startup source											
⊨ ⊞ startup_rsl10.s		•									
L –⊞ 🗟 system_rsl10.c		•									
🖵 🗖 🗖 Device.Libraries.System source											
–⊞ 🔄 rsl10_protocol.c		•									
–⊞ 🗟 rsl10_romvect.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											
-											
		•									
– ⊞ lo] rsl10 svs rffe.c		•									
- ⊞ 💽 rsl10_sγs_timers.c											
—⊞ lo rsl10 svs uart.c											
L-⊞ lo rsl10 sys version.c											
🗉 🛋 include											
E 🗟 app.c											
− ■ readme blinkv.txt											
E 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	⊨⊞ 🗟 app.c	
Lindating build tree	🛏 🖹 readme_blinky.txt	
startun rel10 s	He CMSIS-Pack	
rel10 protocol c	🖵 📮 📫 Output	
rel10_remvectic	🛏 🗎 blinky.map	
ral10 ava eara a	http://www.out	
isito_sys_dsic.c		
app.c	📙 📥 blinky.hex	
rsho_sys_audio.c	🖵 🗎 blinky.map	
rsiiu_sys_clocks.c	- D app.o	
rsiiu_sys_crc.c	🛄 dl7M_tln.a.	
rsiiu_sys_dma.c	🛄 m7M_tl.a.	
rsIIU_sys_flash.c	- Insl10_protocol.o	
rsIIU_sys_power.c	- Insl10_romvect.o	
rsl10_sys_rffe.c	- Isl10_sys_asrc.o	
rsl10_sys_power_modes.c	- Institutional Institution	
rsl10_sys_timers.c	- Insl10_sys_clocks.o	
rsl10_sys_uart.c	rsl10_sys_crc.o	
rsl10_sys_version.c	- Irsl10_sys_dma.o	
system_rsl10.c	- Insl10_sys_flash.o	
Linking	rsl10_sys_power.o	
blinky.out	rsl10_sys_power_modes.o	
Converting	rsl10_sys_rffe.o	
	- irsl10_sys_timers.o	
Total number of errors: 0	rsl10_sys_uart.o	
Total number of warnings: 0	rsl10_sys_version.o	
	- 1 rt7M_tl.a	
Build Debug Log	- Sections.icf	
baild babag tog	- Shb I.a	
	- startup_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
<b>A</b>	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 for the connected emulator, please get in buch 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 45.

3. The application runs up to the first breakpoint in *main*. You can press F5 or the Run icon (as shown in Figure 38) multiple times to step through the code and observe that the LED changes its state when the application executes the function Sys GPIO Toggle (LED DIO). To stop the debug session, press the Stop icon.



Figure 38. Debug Session in the IAR Embedded Workbench

## **CHAPTER 6**

## **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<sup>®</sup>. 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				Desc	ription			
ARM::CMSIS.RTOS	2.FreeR	TOS		Com	ponent is	missing. Pack is not installed: ARM.CMSIS-FreeRTOS		
ARM.FreeRTOS::RT	OS.Cor	fig.CMSIS RT	OS2	Component is missing. Pack is not installed: ARM.CMSIS-FreeRTOS				
ARM.FreeRTOS::RTOS.Core.Cortex-M				Component is missing. Pack is not installed: ARM.CMSIS-FreeRTOS				
ARM.FreeRTOS::RTOS.Event Groups				Component is missing. Pack is not installed: ARM.CMSIS-FreeRTOS				
ARM.FreeRTOS::RTOS.Heap.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.

Devices 🛛 📓 Boards			🏟 Packs 🛛 📑 Examples		. E   €
	H E 💥 🕐	~	Search Pack		
earch Device			Pack	Action	Description
evice	Summary		Device Specific	8 Packs	ARM selected
* All Devices	6249 Devices		ARM.Musca_A1_BSP	Install	Musca A1 Board Support PACK for CoreLink SSE-200 based
ABOV	20 Devices		ARM.Musca_B1_BSP	Install	Musca B1 Board Support PACK for CoreLink SSE-200 based
Active-Semi	4 Devices		ARM.V2M-MPS2_SSE_2	00_B: 🄄 Install	ARM V2M-MPS2 Board Support PACK for CoreLink SSE-200
Ambiq Micro	8 Devices		## ARM.V2M-MPS3_SSE_2	00_B: 🄄 Install	ARM V2M-MPS3 Board Support PACK for CoreLink SSE-200
Amiccom	5 Devices		Keil.V2M-MPS2_CMx_B	SP 🗇 Install	ARM V2M-MPS2 Board Support PACK for Cortex-M System
Analog Devices	14 Devices		Keil.V2M-MPS2_DSx_BS	SP 🗇 Install+	ARM V2M-MPS2 Board Support PACK for DesignStart Devic
APEXMIC	14 Devices		Heil.V2M-MPS2_IOTKit_	BSP 🕸 Install	ARM V2M-MPS2 Device Family Pack for IOT-Kit devices
ARM	57 Devices		Heil.V2M-MPS3_IOTKit	BSP 🄄 Install+	ARM V2M-MPS3 Device Family Pack for IOT-Kit devices
AutoChips	24 Devices		<ul> <li>Generic</li> </ul>	38 Packs	Software Packs with generic content not specific to a device
Cypress	425 Devices		Alibaba.AliOSThings	Install	AliOS Things software pack
Dialog Semiconduc	t 14 Devices		▷ <sup>#</sup> ARM.AMP	Install	Software components for inter processor communication (A
GigaDevice	160 Devices		A HARM.CMSIS	🕸 Install	CMSIS (Cortex Microcontroller Software Interface Standard)
HDSC	26 Devices		·······	🐸 Unpack	The following folders are deprecated - CMSIS/Include/
Holtek	171 Devices	Ξ	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_Val	idatic 🄄 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_Vali	datio 🏵 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 implementatior
NXP	1169 Devices		birdec.bi-pcm3060	Install	CMSIS-Driver for sound codec TI PCM3060
ONSemiconductor	1 Device		EmbeddedOffice.Flexib	ole Sa Install	Elexible 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).

🕆 Packs 🗹 Use all latest Packs 😽 Resolve	e Missing Packs 🗹 Show or	nly used Packs	s
Pack	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
omponents Device Packs			

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

## CHAPTER 7

## 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 <u>www.segger.com</u>.

### 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.

To enable printf, add OUTPUT\_INTERFACE=OUTPUT\_UART or OUTPUT\_INTERFACE=RTT in the preprocessor settings as follows depending on which IDE you are using:

- For Eclipse, right click the sample app name and choose **Properties > Tool Settings > C Compiler > Preprocessor**.
- For Keil, in the menu bar choose **Project > Options for Target > C/C++**.
- For IAR, right click the sample app name and choose **Options** > C/C++ Compiler > Preprocessor.

Samples for RTT are under *C:\Program Files* (*x*86)\*SEGGER\JLink\_V640b\Samples\RTT*.

More information about the RTT API can be found in the JLINK manual, under C:Program Files (x86) $SEGGERJLink_V640bDocManuals$ .

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 14, and you are using the onsemi 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:

```
set {int} &__VTOR = ISR_Vector_Table
set $sp = *((int *) &ISR_Vector_Table)
set $pc = *((int*) (&ISR_Vector_Table+4))
```

reate, manage, and run configu	rations
] 🖻 🍋 🗎 🗶 🖻 🝸 🔹	Name: blinky
type filter text	📄 Main 🕸 Debugger 🝺 Startup 🦆 Source 🔲 Common 🛼 SVD Path
C C/C++ Application C C/C++ Attach to Application C C/C++ Container Launcher C C/C++ Postmortem Debugg C C/C++ Remote Application C C C/C++ Remote Application C C C/C++ Remote Application C C C C C C C C C C C C C C C C C C C	Initial reset and Halt Type Low speed: 1000 kHz JTAG/SWD Speed:  Auto Adaptive Fixed kHz Inable flash breakpoints Enable flash breakpoints Enable Sw0 CPU freq: 10000000 Hz. SW0 freq: 0 Hz. Port mask: 0x1 Load Symbols and Executable Load Symbols and Executable Use project binary: blinky.elf Use file: Workspace File System Symbols offset (hex):
	set {int} &VTOR = ISR_Vector_Table ^ set \$sp = "((int ') &USR_Vector_Table) set \$pc = "((int') (&USR_Vector_Table+4)) / v
	Set program counter at (hex):
	Set breakpoint at: main
	Continue
	Restore default
x > ilter matched 11 of 11 items	Revert Apply

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 onsemi 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 41).
- 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 45.

° 🗎 🗶 🖪 🐎 🗸	
type filter text	Name: peripheral_server_sleep Debug
C/C++ Attach to Application	Initialization Commands
C/C++ Postmortem Debugger	☑ Initial Reset and Halt Type: 1 Low speed: 1000 kHz
C/C++ Remote Application	JTAG/SWD Speed:  Adaptive Fixed kHz
Eclipse Application	Fnable flash breaknoints
GDB Hardware Debugging	Chable seminating Console routed to:      File Console routed to:
GDB OpenOCD Debugging	
GDB SEGGER J-Link Debugging	Phable Swo CPO freq: 0 Hz. Swo freq: 0 Hz. Port mask: 0x1
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	O Use file: Workspace File System
Mus2 Launch	Symbols offset (hev):
Mwez Laurch	Symbols of set (inc).
Remote Java Application	
	O Use file: Workspace File System
	Executable offset (hex):
	Revert Apply

Figure 46. Setting Reset Type in the Debug Configuration Session

eate, manage, and run configuration	1
) 🖻 🍋 🗎 🗶 🖻 🌩 🗸	Name: peripheral_server_sleep Debug
/pe filter text	Main 🕸 Debugger 🌘 Startup 💱 Source 🔲 Common 👷 SVD Path
C/C++ Application     C/C++ Application     C/C++ Octainer Launcher     C/C++ Oostmontem Debugger     C/C++ Postmontem Debugger     C/C++ Nemote Application     C/C++ Unit     GOB Hardware Debugging     GOB SEGGER J-Link Debugging     G DB SEGGER J-Link Debugging     G peripheral server_sleep Debugg	Initialization Commands Imitialization Commands Imitia
Launch Group Launch Group (Deprecated)	×
	Oldse project binary:       peripheral_server_sleep.elf         Outse file:       Workspace         Symbols offset (hee):
	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):
	Continue

## 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 43).

* 😫 🗐 🗶 🗐 🍫 🕤	Name: peripheral_ser	/er_sleep Debug_swd_att	
pe filter text	Main 🏇 Debugg	er 🕞 Startup 🧤 Source 🔲 Common 🔀 SVD Path	
<ul> <li>C/C++ Application</li> <li>C/C++ Attach to Application</li> <li>C/C++ Attach to Application</li> <li>C/C++ Postmortem Debugger</li> <li>C/C++ Remote Application</li> <li>C/C++ Nemote Application</li> <li>C/C++ Unit</li> <li>GDB Apardware Debugging</li> <li>GDB SEGGER J-Link Debugging</li> <li>peripheral_server_sleep Debug</li> <li>peripheral_server_sleep Debug (1)</li> <li>Launch Group (Deprecated)</li> </ul>	J-Link GDB Server Se Start the J-Link C Executable path: Actual executable: Device name: Endianness: Connection: Interface: Initial speed: GDB port: SWO port: Telnet port:	tup DB server locally Connect S(jiink_path)/S(jiink_gdbserver) C/Program Files (x86)/SEGGER/JLink//JLinkGDBServerCL.ex to change it use the global or workspace preferences pages RSL10 © Little Big USB IP SWD JTAG Auto Adaptive Fixed 1000 kHz 2331 2332 V	to running target Browse Variables e or the project properties page) Supported device name (USB serial or IP name/address) ferify downloads I initialize registers on star ocal host only I Silent
	Log file:		Browse
	Other options:	-singlerun -strict -timeout 0 -nogui for the GDB server	sole for semihosting and SWO
	GDB Client Setup		
	Executable name:	{cross_prefix}gdb{{cross_suffix}	Browse Variables
	Actual executable:	arm-none-eabi-gdb	
	Other options: Commands:	set mem inaccessible-by-default off	~
	Remote Target Host name or IP ad Port number:	dress: localhost 2331	
	Force thread list up	date on suspend	
			Devent Area

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

Debug Configurations	- 0
reate, manage, and run configurations	
Image: Second	Name     peripheral_server_sleep Debug_swd_att       Imain (% Debugger Image: Source)     Imain Common       Imitialization Commands     Imitialization Commands       Imitializet and Halt     Type       Low speed:     1000       kHz
c (/C++ Container Launcher C (/C++ Costmortem Debugger C (/C++ Remote Application Cij C/C++ Unit C 6DB Hardware Debugging C 6DB OnespCCD Debugging	JTAG/SWD Speed:     Adaptive     Fixed     KHz       Inable flash breakpoints     Inable semihosting     Console routed to:     Telnet     GDB client       Inable SWO     CPU freq:     0     Hz.     NOT
Cobs Species Decogging     Cobs Secret Decogging     C peripheral_server_sleep Debug     C peripheral_server_sleep Debug     Launch Group     Launch Group (Deprecated)	Load Symbols and Everytable
	Workspace     File System
	Load executable     O Use project binary: peripheral_server_sleep.elf     Use file:     Executable offset (hed):
	Runtime Options       RAM application (reload after each reset/restart)       Run/Restart Commands       Pre-run/Restart reset       Type:       (always executed at Restart)
	tet program counter at (hes):
	Continue Restore defau
ter matched 14 of 44 items	Revert Apply
?	Debug Close

## 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).



## 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).

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

#### Figure 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).
- Repeat step 2 above. After successfully downloading *blinky* to flash memory, disconnect DIO12 from ground, 3. 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 onsemi IDE or the Keil IDE: %LOCALAPPDATA%\Arm\Packs\ONSemiconductor\RSL10\<version>)
- If you are using the IAR IDE:
   C:\Users\<user\_name>\IAR-CMSIS-Packs\ONSemiconductor\RSL10\<version>

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 onsemi. 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 onsemi 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	onsemi license agreement.		

#### Table 2. Installed Folders and Files - onsemi 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 onsemi IDE.
REVISION	Identifies the revision of the onsemi IDE.
Software_Use_Agreement.rtf	onsemi 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>\AppData\Local\Arm\Packs\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:

- onsemi 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  $\mu$ Vision 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 49.



Figure 54. Accessing RSL10 Documentation from the onsemi IDE



Figure 55. Accessing RSL10 Documentation from the Keil  $\mu$ Vision IDE

ile Edit Search CMSIS Manager Window Help					
361 <b>4 • 14 •</b> 191 • 81 • 12 <b>0 •</b> 0 •					Quick Access
blebutempte 1	~ 0	Darke 11 B Daviese B Roserte	The Examples	Consola	(P. (P.) + 2 (P. (H) (P) = -
Device		Sarch Back			
		Dark	Action	Description	
Device: RSL10 Change		rack	1 Dark	Description RSI 10 calented	
Family: RSL10 Series	CPU: ARM Cortex-M3	ONSemiconductor RSI 10	. Un toid	III ON Semiconductor RSI 10 Device Family Pack	
SubFamily:	Max. Clock: 48 MHz	Generic	and the second	Software Packs with generic content not specific to a device	
Vendor: ONSemiconductor	Memory: 24 kB RAM, 384 kB ROM				
Pack: ONSemiconductor.RSL10.2.4.450	FPU: none				
URL: http://www.keil.com/dd2/orsemiconductor/rsi10	Endian: Little-endian				
Device data books:	Description:				
ARM and Thumb-2 Instruction Set Quick Reference Card	RSL10 is an ultra-low-power,				
Gap Interface Specification	multi-protocol 24 GHz radio				
GATT Interface Specification	designed for use in wireless				
Getting Started with 85L10	consumption and a restricted size.				
L2C Interface Specification					
RSL10 Bootloader Guide					
RSL10 Firmware Over-The-Air User's Guide					
RSL10 Firmware Reference					
RSL10 Hardware Reference					
RSL10 Sample Code User's Guide					
KW BLE Alert Notification Profile Interface Specification					
why but battery service interface specification					
PW BLE Blood Pressure Profile (BUP) Interface Specification     PW BLE Curling Dower Brofile Interface Specification					
PNV BLE Cycling Power Profile Interface Specification     PNV BLE Cycling Enand and Cadance Drofile Interface EnactEration					
RW BLE Cycling Speed and Cadence Profile Interface Specification     BW BLE Davids Information Sanica Interface Specification					
RW BLE End Me Profile Interface Specification					
RW BLE Glucose Profile (GLP) Interface Specification					
RW BLE Health Thermometer Profile Interface Specification					
RW BLE Heart Rate Profile (HRP) Interface Specification					
RW BLE HID Over GATT Profile Interface Specification					
RW BLE Host Error Code Interface Specification					
RW BLE Location and Navigation Profile Interface Specification					
RW BLE Phone Alert Status Profile Interface Specification Interface Specification					
RW BLE Proximity Profile Interface Specification					
RW BLE Running Speed and Cadence Profile Interface Specification					
RW BLE Scan Parameters Profile Interface Specification					
RW BLE Time Profile (TIP) Interface Specification					
RW BLE Wireless Power Transfer Profile (WPTP) Specification					
Lompabble boards:					

### 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 onsemi 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 onsemi 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 onsemi IDE" on page 8.

- 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<sub>-</sub>*, *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 onsemi 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 <u>www.onsemi.com/RSL10</u> 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.*
- 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 onsemi 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 onsemi 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 onsemi 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 onsemi IDE access to the toolchain installed with the RSL10 software tools.

Conflicts with toolchain versions can occur in the onsemi 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 ONSEMI 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 <b>Toolchains</b> 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.

onsemi and the onsemi logo are trademarks of Semiconductor Components Industries, LLC dba onsemi or its subsidiaries in the United States and/or other countries. onsemi owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of onsemi's product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent-Marking.pdf. onsemi reserves the right to make changes without further notice to any products herein. onsemi makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does onsemi assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Buyer is responsible for its products and applications using onsemi products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by onsemi. "Typical" parameters which may be provided in onsemi data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. onsemi does not convey any license under its patent rights nor the rights of others. onsemi products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use onsemi products for any such unintended or unauthorized application, Buyer shall indemnify and hold onsemi and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arisi

#### PUBLICATION ORDERING INFORMATION

#### LITERATURE FULFILLMENT:

Literature Distribution Center for onsemi 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

onsemi Website: www.onsemi.com

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

For additional information, please contact your local Sales Representative