

# **FreeRTOS User's Guide**

FreeRTOS User's Guide Version 1.5.1, August 2021

> © 2021 Analog Devices, Inc. http://www.analog.com processor.tools.support@analog.com

# Contents

1	Intro	duction		6				
	1.1	Analog	Devices FreeRTOS	6				
	1.2	1.2 What's in the user's guide						
2	Hard	dware an	d software set up	8				
	2.1	2.1 Get the hardware ready						
	2.2	Get the	source code ready	9				
		2.2.1	Download FreeRTOS source code	9				
		2.2.2	Install the FreeRTOS product from Analog Devices	9				
	2.3	Softwa	re environment set up for CrossCore Embedded Studio	9				
3	Run	ning the	Examples on the ADSP-SC589 EZ-Kit	11				
	3.1	Runnin	g the Basic Example for ARM on ADSP-SC589 EZ-Kit with CrossCore Embedded Studio	11				
		3.1.1	Overview	11				
		3.1.2	Environment Setup	11				
		3.1.3	Build the Example	13				
		3.1.4	Run the Example	15				
		3.1.5	Test Results	17				
	3.2	Runnin	g the Basic Example for SHARC+ on ADSP-SC589 EZ-Kit with CrossCore Embedded Stud	oic				
	17							
		3.2.1	Overview	17				
		3.2.2	Environment Setup	17				
		3.2.3	Build the Example	18				
		3.2.4	Run the Example	20				
		3.2.5	Test Results	22				
4	Run	ning the	Examples on the ADSP-SC584 EZ-Kit	23				
	4.1	Runnin	g the Basic Example for ARM on ADSP-SC584 EZ-Kit with CrossCore Embedded Studio	23				
		4.1.1	Overview	23				
		4.1.2	Environment Setup	23				
		4.1.3	Build the Example	25				
		4.1.4	Run the Example	27				
		4.1.5	Test Results	29				
	4.2	Runnin	g the Basic Example for SHARC+ on ADSP-SC584 EZ-Kit with CrossCore Embedded Stud	oic				
	29							
		4.2.1	Overview	29				
		4.2.2	Environment Setup	29				
		4.2.3	Build the Example	30				
		4.2.4	Run the Example	32				
		4.2.5	Test Results	33				
5	Run	ning the	Examples on the ADSP-SC573 EZ-Kit	35				
5.1 Running the Basic Example for ARM on ADSP-SC573 EZ-Kit with CrossCore Embedded S				35				
		5.1.1	Overview	35				

		5.1.2	Environment Setup	35
		5.1.3	Build the Example	37
		5.1.4	Run the Example	39
		5.1.5	Test Results	41
	5.2	Runnin	g the Basic Example for SHARC+ on ADSP-SC573 EZ-Kit with CrossCore Embedded Studi	io
4	41			
		5.2.1	Overview	41
		5.2.2	Environment Setup	41
		5.2.3	Build the Example	42
		5.2.4	Run the Example	44
		5.2.5	Test Results	45
6	Run	ning the	Examples on the ADSP-SC594 EZ-Kit	47
	6.1	Runnin	g the Basic Example for ARM on ADSP-SC594 EZ-Kit with CrossCore Embedded Studio	47
		6.1.1	Overview	47
		6.1.2	Environment Setup	47
		6.1.3	Build the Example	48
		6.1.4	Run the Example	50
		6.1.5	Test Results	52
	6.2	Runnin	g the Basic Example for SHARC+ on ADSP-SC594 EZ-Kit with CrossCore Embedded Studi	io
!	52			
		6.2.1	Overview	52
		6.2.2	Environment Setup	52
		6.2.3	Build the Example	53
		6.2.4	Run the Example	55
		6.2.5	Test Results	56
	6.3	Runnin	g the MCAPI Example on ADSP-SC594 EZ-Kit with CrossCore Embedded Studio	56
		6.3.1	Overview	56
		6.3.2	Environment Setup	56
		6.3.3	Build the Example	57
		6.3.4	Run the Example	59
		6.3.5	Test Results	61
7	Run	ning the	Examples on the ADSP-BF7XX EZ-Kit	62
	7.1		g the Basic Example for ADSP-BF707 EZ-Kit with CrossCore Embedded Studio	62
		7.1.1	Overview	62
		7.1.2	Environment Setup	62
		7.1.3	Build the Example	63
		7.1.4	Run the Example	65
		7.1.5	Test Results	66
8	Run	ning the	Examples on the ADSP-21569 EZ-Kit	67
	8.1		g the Basic Example for ADSP-21569 EZ-Kit with CrossCore Embedded Studio	67
		8.1.1	Overview	67
		8.1.2	Environment Setup	67
		8.1.3	Build the Example	68
		8.1.4	Run the Example	69

	8	.1.5 Test Results	71
9	Using	CrossCore Embedded Studio System Services and Device Drivers with FreeRTOS	72
10	Appe	ndix A: FreeRTOS Performance	74
	10.1	ADSP-21569 (SHARC Core) Benchmark Data	75
	10.2	ADSP-SC589 (Cortex-A Core) Benchmark Data	77
	10.3	ADSP-SC589 (SHARC+ Core) Benchmark Data	79
	10.4	ADZS-BF707 Benchmark Data	81

# **Copyright Information**

© 2021 Analog Devices, Inc., ALL RIGHTS RESERVED. This document may not be reproduced in any form without prior, express written consent from Analog Devices, Inc.

# Disclaimer

Analog Devices, Inc. reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent rights of Analog Devices, Inc.

# **Trademark and Service Mark Notice**

The Analog Devices logo, CrossCore, A<sup>2</sup>B, EngineerZone, EZ-Board, EZ-KIT Lite and VisualDSP++ are registered trademarks of Analog Devices, Inc. Blackfin, Blackfin+, SHARC, SHARC+ and SigmaStudio are trademarks of Analog Devices, Inc. All other brand and product names are trademarks or service marks of their respective owners.

# **1** Introduction

# **1.1 Analog Devices FreeRTOS**

The Analog Devices FreeRTOS product is an add-on to the FreeRTOS Real-time operating system that provides additional support for Analog Devices processors. The product is installed on top of the FreeRTOS operating system in order to gain additional platform support.

In order to avoid confusion the FreeRTOS operating system will be referred to as the **FreeRTOS product.** Components from Analog Devices are always referred to as the **Analog Devices FreeRTOS product** or the **FreeRTOS product from Analog Devices**.

The Analog Devices FreeRTOS product contains ports of FreeRTOS specific to Analog Devices processors and FreeRTOS example applications for Analog Devices processors. It is intended to be installed on top of version 10.0.x of the FreeRTOS operating system.

# 1.2 What's in the user's guide

This User's Guide document provides instructions on getting started with FreeRTOS for these boards using the CrossCore Embedded Studio development environment.

Processors Supported	Examples Provided For
ADSP-SC5xx Cortex-A5 Core	ADSP-SC589 EZ-Kit
	ADSP-SC584 EZ-Kit
	ADSP-SC573 EZ-Kit
	ADSP-SC594 EZ-Kit
ADSP-SC5xx SHARC+ Core	ADSP-SC589 EZ-Kit
	ADSP-SC584 EZ-Kit
	ADSP-SC573 EZ-Kit
	ADSP-SC594 EZ-Kit
ADSP-BF7xx	ADSP-BF707 EZ-Kit
ADSP-2156x	ADSP-21569 EZ-Kit

The following processors are supported with examples being provided for the following EZ-Kits:

Detailed description for setting up the hardware and software environment, and how to run the demo examples on Analog Devices processor boards are included. An appendix containing RTOS benchmark data for various platforms is also provided.

# 2 Hardware and software set up

To run the FreeRTOS examples, this section would guide users how to get the hardware and software ready, including get the FreeRTOS source code and set up running environment.

# 2.1 Get the hardware ready

The Analog Devices FreeRTOS product supports several reference development boards from Analog Devices, including the ADSP-SC589/ADSP-SC584/ADSP-SC573 EZ-Kit board, BF707 EZ-Kit board and ADSP-21569 EZ-Kit board.

Below is a list of the hardware involved.

ADI reference board:

- ADSP-SC594 SOM and ADSP-SOMCRR-EZKIT: https://www.analog.com/en/products/adspsc594.html#product-evaluationkit
- ADSP-SC589 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/EVAL-ADSP-SC589.html
- ADSP-SC584 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/EVAL-ADSP-SC584.html
- ADSP-SC573 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/SC573EZKIT.html
- ADSP-BF707 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/eval-bf707.html
- ADSP-21569 EZ-kit: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/ADZS-21569-EZKIT.html

Jtag debugger:

• ICE1000/2000: http://www.analog.com/en/design-center/evaluation-hardware-and-software /evaluation-boards-kits/emulators.html

#### PC:

A mainstream configuration of Windows PC is required. Verify that your PC has these minimum requirements:

- 2 GHz single core processor; 3.3GHz dual core or better recommended
- 4 GB RAM; 8GB or more recommended
- 2 GB available disk space
- One open USB port

# 2.2 Get the source code ready

This page describes how to get the FreeRTOS source code.

## 2.2.1 Download FreeRTOS source code

Source code for both upstream official FreeRTOS release, and the Analog Devices FreeRTOS add on product release are required. As shown in the list:

Name	Version	Download from	
Official FreeRTOS source code	10.0.0	https://sourceforge.net/projects/freertos/files/FreeRTOS/	
Analog Devices FreeRTOS	1.5.0	http://www.analog.com/en/design-center/processors-and-dsp/evaluation-and- development-software/freertos.html#dsp-relatedsoftware	

## 2.2.2 Install the FreeRTOS product from Analog Devices

To install the Analog Devices FreeRTOS product you will need to first unzip the FreeRTOS product and then install the Analog Devices FreeRTOS product on top of it:

For example, unzip them into folder "freertos":

1. Unzip the "FreeRTOSv10.0.0.zip" into C:\Analog Devices\freertos.

You will get the path such as "C:\Analog Devices\freeRTOSv10.0.0\FreeRTOS".

2. Unzip "adi-freertos-1.5.0.zip" into C:\Analog Devices\freertos.

It will add new files and overwrite some files saved in "C:\Analog Devices\freeRTOSv10.0.0 \FreeRTOS".

# 2.3 Software environment set up for CrossCore Embedded Studio

This part shows the software environment setup for CrossCore Embedded Studio on four kinds of boards: ADSP-SC589/ADSP-SC584/ADSP-SC573/ADSP-SC594/ADSP-2156X, AND ADSP-BF7XX EZ-Kit.

#### ADSP-SC589/ADSP-SC584/ADSP-SC573/ADSP-SC594

• Analog Devices CrossCore Embedded Studio version 2.10.0 or later

#### ADSP-BF7XX EZ-Kit

• Analog Devices CrossCore Embedded Studio version 2.10.0 or later

#### ADSP-2156X EZ-Kit

• Analog Devices CrossCore Embedded Studio version 2.10.0 or later

# 3 Running the Examples on the ADSP-SC589 EZ-Kit

The FreeRTOS product for Analog Devices processors contains the following examples:

Processor	Core	Toolchain	Example(s)
ADSP-SC589	ARM A5	CrossCore Embedded Studio	Basic Demo
ADSP-SC589	SHARC+	CrossCore Embedded Studio	Basic Demo

The basic demo example is based on the **Standard Demo Tasks** that FreeRTOS recommend are provided for each port of the FreeRTOS Operating System. For more information on the Standard Demo Tasks please refer to http://www.freertos.org/a00013.html.

The tasks performed in the Analog Devices Basic Demo include:

- LED flash
- Polled queue tasks
- Recursive Mutex tasks
- Blocking Queue tasks
- Statically allocated tasks
- Suicidal tasks

# 3.1 Running the Basic Example for ARM on ADSP-SC589 EZ-Kit with CrossCore Embedded Studio

## 3.1.1 Overview

This page describes the steps to build and run basic example for ARM on ADSP-SC589 EZ-Kit board using CrossCore Embedded Studio.

## 3.1.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

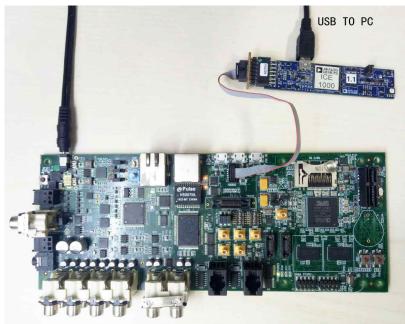
#### **Software Requirement**

- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio
- FreeRTOS product and the Analog Devices FreeRTOS product. For more information please refer to Get the source code ready

#### **Hardware Setup**

- An ADSCP-SC589 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P3** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 5 volts as in the diagram below.



Connect the **USB to UART** port of the EZ-Kit to the host PC with a USB cable as shown below:



# 3.1.3 Build the Example

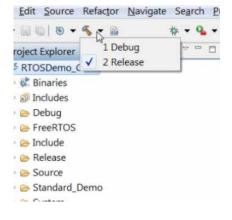
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the File menu and then select the Import option from the menu
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the Select root directory radio button and then click the Browse button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX\_A5\_ADSP\_SC589\_CCES folder
- Click **Finish** to close the file browser dialog
- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the **projects** pane and click **Import**

K Import		
Import Projects		
Select a directory to sea	rch for existing Eclipse projects.	
Select root directory:	C:\Analog Devices\test_case\freertos\FreeRTOSv10	Browse
Select archive file:	<b></b>	Browse
Projects:		
RTOSDemo_CCES	_Core0 (C:\Analog Devices\test_case\freertos\FreeRTOS	Select All
		Deselect All
		Refresh
•	4	
Options		
Search for nested pro	ojects	
Copy projects into w		
	ready exist in the workspace	
Working sets	_	
Add project to work	ing sets	New
Working sets:	•	Select
2	< Back Next > Finish	Cancel
J		Cancer

2. Choose Debug/Release mode to build the project.



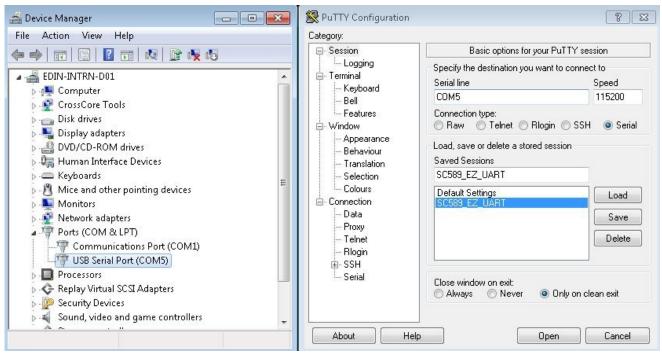
- 3. Build the project in CrossCore Embedded Studio:
  - In the **Project Explorer** right click on the **RTOSDemo\_CCES\_Core0** project and select the **Build Project** option from the menu

## 3.1.4 Run the Example

The semihosting I/O mechanism, which writes to the CCES console during debug sessions, uses SWI interrupts. This is incompatible with default GCC-compiled I/O code which also uses SWI interrupts. For this reason, stdio function calls initiated on the ARM core are routed out over UART instead and shall be read with a serial terminal external to CCES. Importantly, note that:

- 1. This is currently only supported *within* FreeRTOS threads, any stdio function call performed outwith a thread will crash the application.
- 2. If you need to use other peripherals, you should take care not to change the power service clock rate (which is set in the UART I/O device).

Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the COM port. Configure your serial console application to connect to the port with a baud rate of 115,200.



After this, follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_CCES\_Core0** project and select the **Debug As** option from the menu

2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board

2 Debug Configurations					
Create, manage, and run configurations					
Select a debug session to launch and a program to load				200	
	Name: RTOSDemo_CCES_Core0 Debug				
type filter text   A pplication with CrossCore Debugger	Session Automatic Breakpoints Target Options W Custom Board Support Multiproce	ssor Groups 💱 Source 🔲 Commo	m		
RTOSDemo_CCES_Core0 Debug	Session configuration Target: Emulation Debug Target			Select Session	
Application with GDB and QEMU (Simulator) Launch Group	Platform: ADSP-SC589 via ICE-1000 Processor: ADSP-SC589				
	The following program(s) will be loaded:				
	Program A @ Device 0 [Core 0]	Options	Silicon revision	Add	
	C:\Analog Devices\CrossCore Embedded Studio 2.7.0\SHARC\Idr\ezkitSC589_preload_cor		not available	Edit	
	RTOSDemo_CCES_Core0\Debug\RTOSDemo_CCES_Core0     Ø Device 0 [Core 1]	Check si-revision, Run after load	not available	Remove	
	<click a="" here="" load="" program="" select="" to=""> A @ Device 0 [Core 2]</click>			Remove All Move Up	
	Click here to select a program to load>			Move Down	
				Restore Defaults	
				(manufacture of the second sec	
Filter matched 5 of 15 items			Revert	Apply	
0			Debug	Close	

#### 3 . Disable the **semihosting** function in **Automatic Breakpoints**

2 Debug Configurations					
Create, manage, and run configurations Specify and launch a CrossCore Embedded Studio p	rogram		Ť.		
🗋 🗎 🗶 📄 🎲 🗸	Name: RTOSDe	mo_CCES_Core0 Debug			
type filter text	👦 Session 💿	Automatic Breakpoints 🕞 Target Options) 🎇 Custom Board Support 🦃 Multiprocessor Groups 🖏 Source 🔲 Common			
Application with CrossCore Debugger	Processor:				
RTOSDemo_CCES_Core0 Debug Application with GDB and OpenOCD (Emulator)	Device 0 [Core	0] ( Cortex-A5 ) 🗸	1		
Application with GDB and QEMU (Simulator)	Breakpoints to s	et automatically after load:	í		
🔋 Launch Group	Label	Description	<u>N</u> ew		
▶ Launch Group (Deprecated)	→	End of program Fatal error occurred in RTL Start of program Used for assert Stack overflow detected Fatal exception occurred in RTL	Edit Delete Delete All		
	Enable semil	nosting			
Im     Filter matched 6 of 6 items		Revert	Apply		
?		Debug	Close		

- 4. Click the **Debug** button to close the **Debug Configurations** window
- 5. Click the Run/Resume button to start running your application

<u>File Edit Source Refactor N</u> avigate Se <u>a</u> rch <u>P</u> roject Target <u>R</u> un <u>W</u> indow <u>H</u> elp
😁 🕶 📾 📾 📾 📾 🖕 🍫 📭 💷 🗷 🗷 🖉 🚳 🛍 🐃 🔍 🤅 ヤ 🎴 🕶 🏉 🛩 🗇 🗸 🖓 🕶 🖓 🕶 🖓 🕶 🖓
Ar Debug 🛛 Resume (F5)
A INTOSDemo_CCES_Core0 Debug [Application with CrossCore Debugger]
ADSP-SC589 via ICE-1000
Device 0 [Core 0] ( Cortex-A5 ) [Debug\RTOSDemo_CCES_Core0] (Suspended : Breakpoint)
main(int, char**) at main.c:133 0xc10149aa
Device 0 [Core 1] ( SHARC ) (Running : User Request)
Device 0 [Core 2] (SHARC) (Running : User Request)

# 3.1.5 Test Results

Output from the application should be visible within the TTY terminal (e.g. PuTTY/TeraTerm). You should see the LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if all the tests passed.

# 3.2 Running the Basic Example for SHARC+ on ADSP-SC589 EZ-Kit with CrossCore Embedded Studio

## 3.2.1 Overview

This page describes the steps to build and run basic example for SHARC+ on ADSP-SC589 EZ-Kit board using CrossCore Embedded Studio.

## 3.2.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

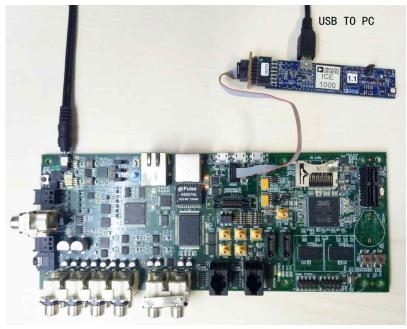
#### **Software Requirement**

- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio
- FreeRTOS product and the Analog Devices FreeRTOS product. For more information please refer to Get the source code ready

#### Hardware Setup

- An ADSCP-SC589 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P3** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 5 volts as in the diagram below.



## 3.2.3 Build the Example

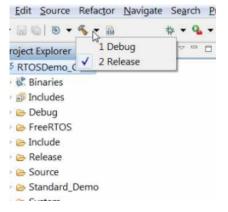
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the File menu and then select the Import option from the menu
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0. FreeRTOS\Demo\SHARC\_ADSP\_SC589\_CCES** folder
- Click **Finish** to close the file browser dialog
- Two projects should appear in the **Project Explorer**

X Import		
Import Projects Select a directory to searc	ch for existing Eclipse projects.	
<ul> <li>Select root directory:</li> <li>Select archive file:</li> <li>Projects:</li> </ul>	C:\Analog Devices\test_case\FreeRTOS\free 💌	Browse Browse
RTOSDemo_CCES	S_SHARC_Core0 (C:\Analog Devices\test_case\Fn S_SHARC_Core1 (C:\Analog Devices\test_case\Fn	Select All Deselect All Refresh
Options     Search for nested pro     Copy projects into we     Hide projects that alr	-	
Working sets	ing sets	New Select
?	< Back Next > Finish	Cancel

2. Choose Debug/Release mode to build the project.



3. Build the project in CrossCore Embedded Studio:

• In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_Core0** and **RTOSDemo\_CCES\_SHARC\_Core1** project, then select the **Build Project** option from the menu

#### 3.2.4 Run the Example

Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_Core0** project and select the **Debug As** option from the menu

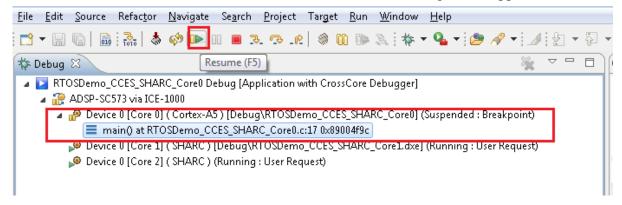
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board

File Edit Source Refactor	🛛 🔀 Debug Configurations			23	
Project Explorer ☎	Create, manage, and run configuration:	S		TO TO	ug 🊵 CMSIS Pack Manager
	type filter text  Type filter text  RTOSDemo_CCES_SHARC_Cor  Application with GDB and OpenO'  Application with GDB and OpenO'  Application with GDB and OpenO'	Name: RTOSDemo_CCES_SHARC_Core0 Debug Session Automatic Breakpoint T Session Wizard Select Processor Choose a target processor.	arget Options) 🌃 Custom Board Support 🔇 Multipro		
		Processor family:           Processor type:	SHARC -	on Add Edit Remove Remove All Move Up Move Down Restore Defaults	
< [ m	Filter matched 5 of 16 items	Show all processors Use selected project to create new session Help <back< td=""><td>Configurator Next &gt; Finish Cancel</td><td>Revert Apply Debug Close</td><td></td></back<>	Configurator Next > Finish Cancel	Revert Apply Debug Close	
RTOSDemo_CCES_SHARC_	Core0				

#### 3. Click the Debug button to close the Debug ConfigurationS window

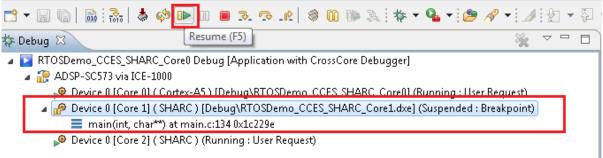
ile Edit Source Refactor	C Debug Configurations				
🖆 🕶 🔚 🐘 👻 🔨 🗸				- A	ug   🏟 CMSIS Pack Manager
Project Explorer 🛛	Select a debug session to launch and a prog				
▲ Carlos RTOSDemo_CCES_SH > ∰ Binaries		Name: RTOSDemo_CCES_SHARC_Core0 Debug	<u>_</u>		
b 👔 Includes b 🍪 section & a D Annication with CrossCore Debug					
<ul> <li>▷ @ &gt; system</li> <li>▷ Debug</li> <li>♥ &gt; system.svc</li> <li>♥ &gt; RTOSDemo_CCES_SH</li> </ul>	<ul> <li>Application with CrossCore Debug</li> <li>RTOSDemo_CCES_SHARC_Cor</li> <li>Application with GDB and OpenO<sup>1</sup></li> <li>Application with GDB and QEMU (</li> <li>Launch Group</li> </ul>	Session configuration Target: Emulation Debug Target Platform: ADSP-SC589 via ICE-1000 Processor: ADSP-SC589		Select Session	
		The following program(s) will be loaded:			
		Program Options     @ Device 0 [Core 0]     C:\Analog Devices\CrossCore Reset, Run after load     RTOSDemo_CCES_SHARC_Cc Check si-revision, Run after load     @ Device 0 [Core 1]     RTOSDemo_CCES_SHARC_Cc Reset, Check si-revision, Run after loa     @ Device 0 [Core 2]	Silicon revision not available not available ad 1.0	Add Edit Remove Remove All Move Up Move Down	
	< )			Restore Defaults	
	Filter matched 5 of 16 items		Re	evert Apply	
	2			Debug Close	
RTOSDemo CCES SHARC C	Corell				

#### 4. Choose Core0 and click the **Run/Resume** button to start running Core0 application



5. Then Choose Core1 and keep to click Run/Resume button to start running Core1 application

<u>File 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 Tar<u>g</u>et <u>R</u>un <u>W</u>indow <u>H</u>elp



## 3.2.5 Test Results

Output from the application should be visible within the **Console** window in the CrossCore Embedded Studio IDE. You should see three LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if the all tests passed.

Image: State Stat	File Edit Source Refactor Navigate Search Project Target Run Window Help				
Trossency CCES_SHARC_Corel Debug [Application with CrossCore Debugger]     ADSP-SCSB9 is LCE 1000     Porce 0 [Core 1] (Cottex-ds ) [Debug XRTOSDerno_CCES_SHARC_Corel.dxd [Running: User Request)     Dexice 0 [Core 1] (Cottex-ds ) [Debug XRTOSDerno_CCES_SHARC_Corel.dxd [Running: User Request)     Dexice 0 [Core 1] (SHARC) (Debug XRTOSDerno_CCES_SHARC_Corel.dxd [Running: User Request)     Dexice 0 [Core 1] (SHARC) (Debug XRTOSDerno_CCES_SHARC_Corel.dxd [Running: User Request)     Dexice 0 [Core 1] (SHARC) (Debug XRTOSDerno_CCES_SHARC_Corel.dxd [Running: User Request)     Dexice 0 [Core 1] (SHARC) (Debug XRTOSDerno_CCES_SHARC_Corel.dxd [Running: User Request)     det	i 🗂 🕶 🔚 🔞   💩   🎄 🤣 🕪 💵 🔳 🎿 🗇 Le   🦃 🛍 🕪 ష   🎋 🗣 🗣 🖌 🤔 🖋 🕶 ル 🖗	• 🖘 🛟 • 🖒 •	Quick Access	😭   🗟 C/C++ 🎼	Debug 酸 CMSIS Pack Manager
ADSP-SCSBB visite-1008     ADSP-SCSBB visite-1008     ADSP-SCSBB visite-1008     ADSP-SCSBB visite-1008     ADSP-SCSBB visite-1008     Device 0 [Core 2] (SHARC) [Debug]RTOSDemo_CCES_SHARC_Corel.dxc [Running: User Request)     Bevice 0 [Core 2] (SHARC) [Debug]RTOSDemo_CCES_SHARC_Corel.dxc [Running: User Request)     adf_IntiComponents();     adf_IntiComponents();     adf_IntiComponents();     core 1 and core 2 is to use the adj_core_enable function.	🎋 Debug 🕱 🦹 👻 🗖 🗖	🗱 🗘 🔍 (x)= Variables 💁 Breakpoints 🛛 👬 Registr	ers 🛋 Module	s) 🛛 💥 💥 🖓 🔄	) 🔌 🕀 🖻 🕵 🔻 🗖
RTOSDemo_CCES_SHARC_Core&c B     add_initComponents();     /**     The default startup code does not include any functionality to allow     core 0 to enable core 1 and core 2. A convenient way to enable     core 0 to enable(ADI_CORE_SHARC(D);     /*adi_core_enable(ADI_CORE_SHARC(D);     /*adi_core_enable(ADI_CORE_SHARC(D);*/	ADSP-SCS89 via ICE-1000 ADSP-SCS89 via ICE-1000 Device 0 [Core 1] (Cortex-A5) [Debug]RTOSDemo_CCES_SHARC_Core10 [Running : User Request) Device 0 [Core 1] (SHARC) [Debug]RTOSDemo_CCES_SHARC_Core1.dxe] (Running : User Request)	🔲 🗘 [function: _exit] [type: Temporary]			
<pre>adi_initComponents();  /**  * The default startup code does not include any functionality to allow * core 0 to enable core 1 and core 2. A convenient way to enable * core 1 and core 2. A convenient way to enable * core anable(ADI_CORE_SHARC0); /*adi_core_enable(ADI_CORE_SHARC0); /*adi_core_enable(ADI_CORE_SHARC0); /*adi_core_enable(ADI_CORE_SHARC0); * Console © Tasks @ Problems © Executables @ Debugger Console @ Progress @ Call Hierarchy Output Test passed Test pas</pre>		4			÷
<pre>     /**     * The default startup code does not include any functionality to allow     * core 0 to enable core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. A convenient way to enable     * core 1 and core 2. Based     * core 1 and core 2. Based     * core 1 and core 2. A convenient way to enable     *</pre>	RTOSDemo_CCES_SHARC_Core0.c 🛛			🗉 Outline 🚺 Memory 🕮	Disassembly 🛛 🗖 🗖
* core 1 and core 2 is to use the adi_core_enable function. */ adi_core_enable(ADI_CORE_SHARCO); /*adi_core_enable(ADI_	<pre>@ /** * The default startup code does not include any functionality to allow</pre>			~	- L & S & L &
Console X Tasks Problems Executables Debugger Console Progress Call Hierarchy Cutyut Test passed Test	<pre>* core 1 and core 2 is to use the adi_core_enable function. */ adi_core_enable(ADI_CORE_SHARC0);</pre>		н		A
Output Test passed	4			*	
Output Test passed	😇 Console 🖄 🧔 Tasks 🐘 Problems 📭 Executables 🖶 Debugger Console 🚍 Progress 🎥 Call Hierarchy				=    =
Test passed Test passed					
Test passed Test passed Test passed Test passed Test passed	Test passed Test passed Test passed Test passed Test passed				^
Writable Smart Insert 20:76	Test passed Test passed Test passed				E
		Writable Smart Inse	rt 20:76		

# 4 Running the Examples on the ADSP-SC584 EZ-Kit

The FreeRTOS product for Analog Devices processors contains the following examples:

Processor	Core	Toolchain	Example(s)
ADSP-SC584	ARM A5	CrossCore Embedded Studio	Basic Demo
ADSP-SC584	SHARC+	CrossCore Embedded Studio	Basic Demo

The basic demo example is based on the **Standard Demo Tasks** that FreeRTOS recommend are provided for each port of the FreeRTOS Operating System. For more information on the Standard Demo Tasks please refer to http://www.freertos.org/a00013.html.

The tasks performed in the Analog Devices Basic Demo include:

- LED flash
- Polled queue tasks
- Recursive Mutex tasks
- Blocking Queue tasks
- Statically allocated tasks
- Suicidal tasks

# 4.1 Running the Basic Example for ARM on ADSP-SC584 EZ-Kit with CrossCore Embedded Studio

#### 4.1.1 Overview

This page describes the steps to build and run basic example for ARM on ADSP-SC584 EZ-Kit board using CrossCore Embedded Studio.

#### 4.1.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

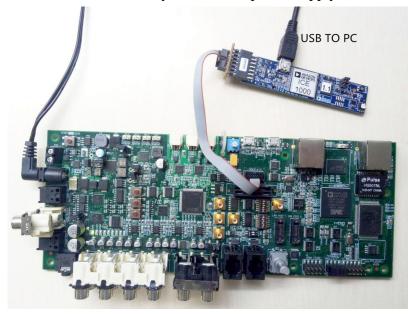
#### **Software Requirement**

- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio
- FreeRTOS product and the Analog Devices FreeRTOS product. For more inormation please refer to Get the source code ready

#### **Hardware Setup**

- An ADSCP-SC584 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P1** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below.



Connect the **USB to UART** port of the EZ-Kit to the host PC with a USB cable as shown below:



## 4.1.3 Build the Example

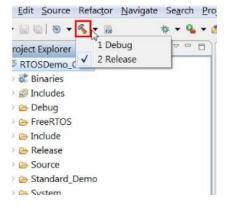
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the File menu and then select the Import option from the menu
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the Select root directory radio button and then click the Browse button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX\_A5\_ADSP\_SC584\_CCES folder
- Click **Finish** to close the file browser dialog
- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the projects pane and click Import

C Import		
Import Projects		
Select a directory to sea	rch for existing Eclipse projects.	
Select root directory:	C:\Analog Devices\test_case\freertos\FreeRTOSv10	Browse
Select archive file:		Browse
Projects:		
RTOSDemo_CCES	_Core0 (C:\Analog Devices\test_case\freertos\FreeRTOS	Select All
		Deselect All
		Refresh
•	4	
Options		
Search for nested pro		
Copy projects into w		
Working sets	ready exist in the workspace	
Add project to work	ing sets	New
Working sets:	• • • • • • • • • • • • • • • • • • •	Select
?	< Back Next > Finish	Cancel

2. Choose Debug/Release mode to build the project.



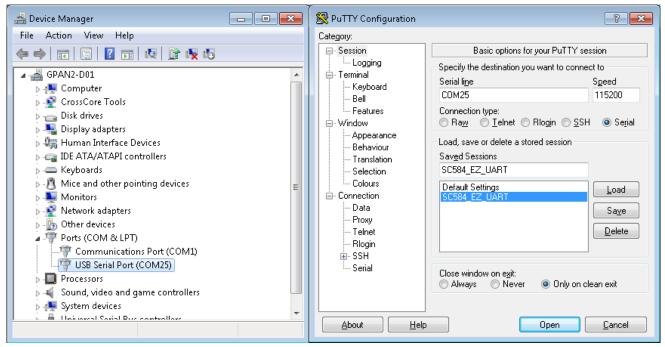
- 3. Build the project in CrossCore Embedded Studio:
  - In the **Project Explorer** right click on the **RTOSDemo\_CCES\_Core0** project and select the **Build Project** option from the menu

## 4.1.4 Run the Example

The semihosting I/O mechanism, which writes to the CCES console during debug sessions, uses SWI interrupts. This is incompatible with default GCC-compiled I/O code which also uses SWI interrupts. For this reason, stdio function calls initiated on the ARM core are routed out over UART instead and shall be read with a serial terminal external to CCES. Importantly, note that:

- 1. This is currently only supported *within* FreeRTOS threads, any stdio function call performed outwith a thread will crash the application.
- 2. If you need to use other peripherals, you should take care not to change the power service clock rate (which is set in the UART I/O device.)

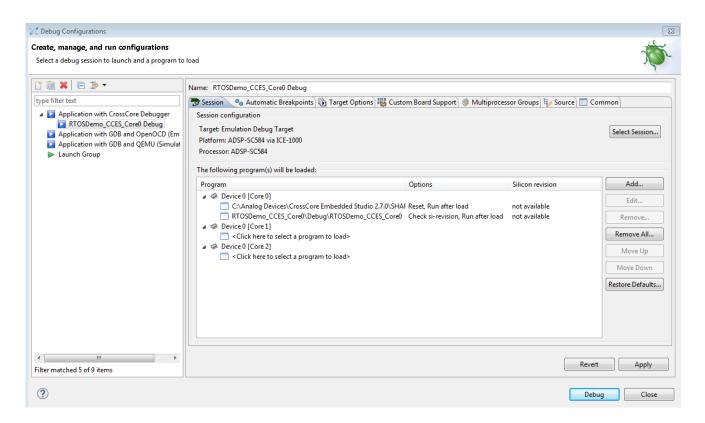
Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the COM port. Configure your serial console application to connect to the port with a baud rate of 115,200.



Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_CCES\_Core0** project and select the **Debug As** option from the menu

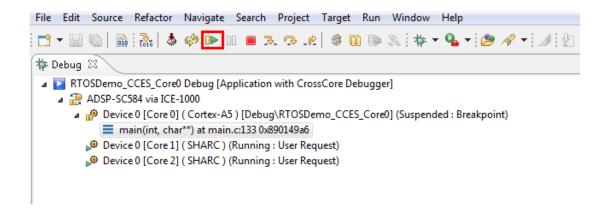
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board



#### 3. Disable the semihosting function in Automatic Breakpoints

Debug Configurations			×
Create, manage, and run configurations Specify and launch a CrossCore Embedded Studio p	rogram		Ť
	Name: RTOSDe	mo_CCES_Core0 Debug	
type filter text		Automatic Breakpoints 🕞 Target Options) 🏭 Custom Board Support 🛞 Multiprocessor Groups 💱 Source 🔲 Common	
Application with CrossCore Debugger	Processor: Device 0 [Core 0] (Cortex-A5)		
RTOSDemo_CCES_Core0 Debug Application with GDB and OpenOCD (Emulator)			
Application with GDB and QEMU (Simulator)	Breakpoints to	set automatically after load:	
↓ Launch Group ▶ Launch Group (Deprecated)	Label	Description	<u>N</u> ew
<ul> <li>Launch Group (Deprecated)</li> </ul>	🔽 🔶 _exit	End of program	Edit
		Fatal error occurred in RTL Start of program	Delete
	🔽 🔶 _dbg_	Used for assert	Delete All
		Stack overflow detected	Delete <u>A</u> ll
	M 🖤fatal_	Fatal exception occurred in RTL	
	Enable semi	hosting	·
Filter matched 6 of 6 items		Reyert	Apply
?		Debug	Close

- 4. Click the Debug button to close the Debug Configurations window
- 5. Click the Run/Resume button to start running your application



## 4.1.5 Test Results

Output from the application should be visible within the TTY terminal (e.g. PuTTY/TeraTerm). You should see the LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if all the tests passed.

# 4.2 Running the Basic Example for SHARC+ on ADSP-SC584 EZ-Kit with CrossCore Embedded Studio

## 4.2.1 Overview

This page describes the steps to build and run basic example for SHARC+ on ADSP-SC584 EZ-Kit board using CrossCore Embedded Studio.

## 4.2.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

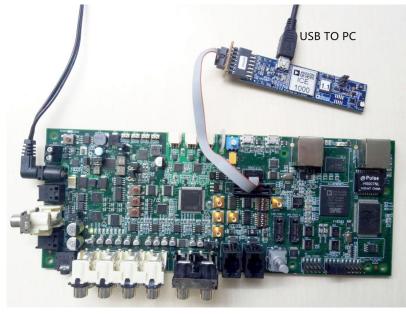
#### Software Requirement

- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio
- FreeRTOS product and the Analog Devices FreeRTOS product. For more inormation please refer to Get the source code ready

#### Hardware Setup

- An ADSCP-SC584 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P1** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below.



## 4.2.3 Build the Example

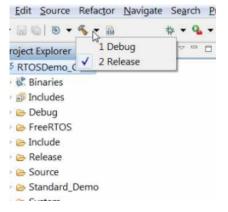
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the **File** menu and then select the **Import** option from the menu
- When the **Import** project window appears:
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0. FreeRTOS\Demo\SHARC\_ADSP\_SC584\_CCES** folder
- Click **Finish** to close the file browser dialog
- Two projects should appear in the Project Explorer

Import Projects Select a directory to searc	ch for existing Eclipse projects.	
<ul> <li>Select root directory:</li> <li>Select archive file:</li> <li>Projects:</li> </ul>	C:\Analog Devices\test_case\FreeRTOS\free 👻	Browse Browse
RTOSDemo_CCES	S_SHARC_Core0 (C:\Analog Devices\test_case\Fn S_SHARC_Core1 (C:\Analog Devices\test_case\Fn	Select All Deselect All Refresh
Working sets Add project to work Working sets:	ing sets	New Select
?	< Back Next > Finish	Cancel

2. Choose Debug/Release mode to build the project.



3. Build the project in CrossCore Embedded Studio:

• In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_Core0** and **RTOSDemo\_CCES\_SHARC\_Core1** project, then select the **Build Project** option from the menu

#### 4.2.4 Run the Example

Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_Core0** project and select the **Debug As** option from the menu

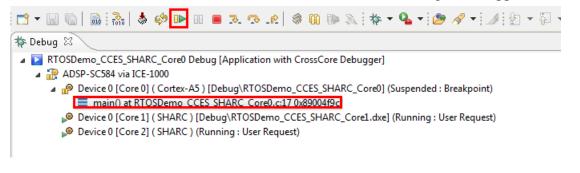
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board

File Edit Source Refactor Navigate Search Project	t Run Window Help		
📑 • 🔛 🐚   🛞 • 🗞 • 📾 💠 • 🂁 🔗	Z Debug Configurations		Ĩ
	22 Debug Configurations Create, manage, and run configurations	Processor family:         SHARC           Processor type:         •           Ø A059-21584         •           Ø A059-21587         •           Ø A059-21587         •           Ø A059-21587         •           Ø A059-21587         •           Ø A059-52571         •           Ø A059-52573         •           Ø A059-52582         •	r Groups & Source Common Select Session Options Silicon revision Edit
	Filter matched 5 of 10 items	A ADSP-SCS4     A ADSP-SCS57     A ADSP-SCS57     A ADSP-SCS57     A DSP-SCS57     Use selected project to create new session     Configurator.     Back Next> Finish Cancel	Remove       Remove All       Move Up       Move Up       Move Down       Restore Defaults       Save picture as       Revet     Apply       Debug     Close
1	100 m + ( m m + ( m	te i en en la companya de la compa	0 🛆 💽

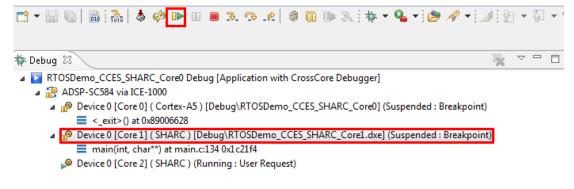
3. Click the **Debug** button to close the **Debug Configurations** window

Z Debug Configurations				×	
Create, manage, and run configurations Select a debug session to launch and a program to load				Ť.	
	Name: RTOSDemo_CCES_SHARC_Core0 Debug				
type filter text	🐡 Session 🔹 Automatic Breakpoints 🚯 Target Options 👯	Custom Board Support) 🏶 Multiprocessor Groups) 🧤 Source 🛽	Common		
Application with CrossCore Debugger     Application with GDB and OpenOCD (Emulator)     Application with GDB and OpenOCD (Emulator)     Application with GDB and QEMU (Simulator)     Launch Group	Session configuration Target: Emulation Debug Target Platform: ADSP-SCS84 via ICE-1000 Processor: ADSP-SCS84				
	The following program(s) will be loaded:				
	Program	Options	Silicon revision	Add	
			not available not available	Edit Remove	
		_CCES_SHARC_Core1.dxe Reset, Check si-revision, Run after load	1.0	Remove All Move Up Move Down	
			Save picture as	Restore Defaults	
			Save picture as		
Filter matched 5 of 10 items			Revert	Apply	
?			Debu	g Close	

4. Choose Core0 and click the **Run/Resume** button to start running Core0 application



5. Then choose Core1 and keep to click Run/Resume button to start running Core1 application



## 4.2.5 Test Results

Output from the application should be visible within the **Console** window in the CrossCore Embedded Studio IDE. You should see three LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if the all tests passed.

Output Loading application: "C:\Analog Devices\CrossCore Embedded Studio 2.7.0\SHARC\ldr\ezkitSC584\_preload\_core0\_v01" Load complete. Loading application: "C:\Analog Devices\freertos\FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC\_ADSP\_SC584\_CCES\RTOSDemo\_CCES\_SHARC\_Core0\Debug\RTOSDemo\_CCES\_SHARC\_Core0" Loading application: "C:\Analog Devices\freertos\FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC\_ADSP\_SC584\_CCES\RTOSDemo\_CCES\_SHARC\_Core1\Debug\RTOSDemo\_CCES\_SHARC\_Core1.dxe" Loading application: "C:\Analog Devices\freertos\FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC\_ADSP\_SC584\_CCES\RTOSDemo\_CCES\_SHARC\_Core1\Debug\RTOSDemo\_CCES\_SHARC\_Core1.dxe" Load complete. Lest passed Test passed [

# 5 Running the Examples on the ADSP-SC573 EZ-Kit

The FreeRTOS product for Analog Devices processors contains the following examples:

Processor	Core	Toolchain	Example(s)
ADSP-SC573	ARM A5	CrossCore Embedded Studio	Basic Demo
ADSP-SC573	SHARC+	CrossCore Embedded Studio	Basic Demo

The basic demo example is based on the **Standard Demo Tasks** that FreeRTOS recommend are provided for each port of the FreeRTOS Operating System.

For more information on the Standard Demo Tasks please refer to http://www.freertos.org/a00013. html.

The tasks performed in the Analog Devices Basic Demo include:

- LED flash
- Polled queue tasks
- Recursive Mutex tasks
- Blocking Queue tasks
- Statically allocated tasks
- Suicidal tasks

# 5.1 Running the Basic Example for ARM on ADSP-SC573 EZ-Kit with CrossCore Embedded Studio

## 5.1.1 Overview

This page describes the steps to build and run basic example for ARM on ADSP-SC573 EZ-Kit board using CrossCore Embedded Studio.

## 5.1.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

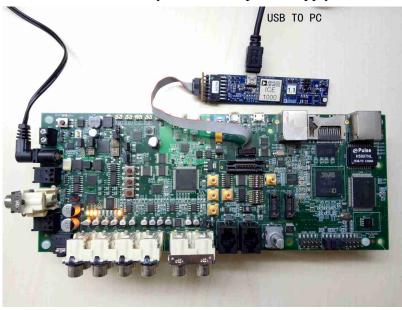
#### **Software Requirement**

- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio
- FreeRTOS product and the Analog Devices FreeRTOS product. For more inormation please refer to Get the source code ready

#### **Hardware Setup**

- An ADSCP-SC573 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P1** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below.



Connect the **USB to UART** port of the EZ-Kit to the host PC with a USB cable as shown below:



# 5.1.3 Build the Example

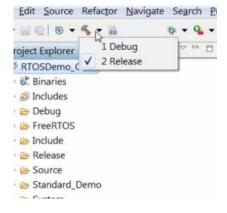
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the File menu and then select the Import option from the menu
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the Select root directory radio button and then click the Browse button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX\_A5\_ADSP\_SC573\_CCES folder
- Click **Finish** to close the file browser dialog
- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the **projects** pane and click **Import**

K Import		
Import Projects		
Select a directory to sea	rch for existing Eclipse projects.	
Select root directory:	C:\Analog Devices\test_case\freertos\FreeRTOSv10	Browse
Select archive file:		Browse
Projects:		
RTOSDemo_CCES	_Core0 (C:\Analog Devices\test_case\freertos\FreeRTOS	Select All
		Deselect All
		Refresh
•	4	
Options		
<ul> <li>Search for nested pro</li> <li>Copy projects into we</li> </ul>	-	
	ready exist in the workspace	
Working sets		
Add project to work	ing sets	New
Working sets:	•	Select
0	< Back Next > Finish	Cancel
•	Next > Finish	Cancer

2. Choose Debug/Release mode to build the project.



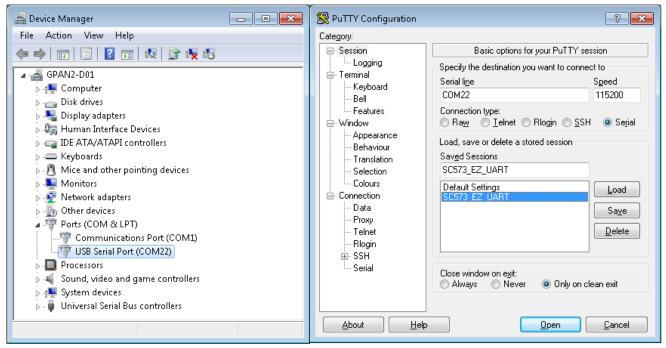
- 3. Build the project in CrossCore Embedded Studio:
  - In the **Project Explorer** right click on the **RTOSDemo\_CCES\_Core0** project and select the **Build Project** option from the menu

## 5.1.4 Run the Example

The semihosting I/O mechanism, which writes to the CCES console during debug sessions, uses SWI interrupts. This is incompatible with default GCC-compiled I/O code which also uses SWI interrupts. For this reason, stdio function calls initiated on the ARM core are routed out over UART instead and shall be read with a serial terminal external to CCES. Importantly, note that:

- 1. This is currently only supported *within* FreeRTOS threads, any stdio function call performed outwith a thread will crash the application.
- 2. If you need to use other peripherals, you should take care not to change the power service clock rate (which is set in the UART I/O device).

Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the COM port. Configure your serial console application to connect to the port with a baud rate of 115,200.



After this, follow the steps below to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_CCES\_Core0** project and select the **Debug As** option from the menu

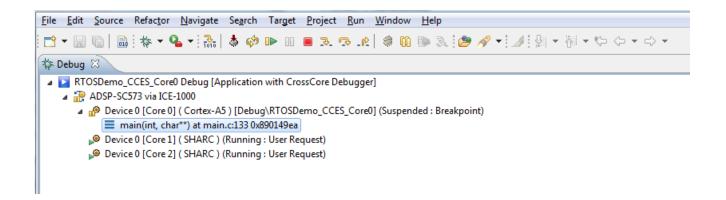
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board

🔀 Debug Configurations				×
Create, manage, and run configurations Select a debug session to launch and a program to lo	oad			Ť.
	Name: RTOSDemo_CCES_Core0 Debug Session @ Automatic Breakpoints Target Options W Cust Session configuration Target: Emulation Debug Target Platform: ADSP-SC573 via ICE-1000 Processor: ADSP-SC573 The following program(s) will be loaded: Program	om Board Support ) 🏶 Multiprocessor Gr Options	oups] 🦭 Source] 🔲 Com	Mon Select Session)
			not available not available	Edit Remove Remove All Move Up Move Down Restore Defaults
III     Filter matched 5 of 9 items			Re	evert Apply
?				Debug Close

#### 3. Disable the semihosting function in Automatic Breakpoints

Configurations			×
Create, manage, and run configurations Specify and launch a CrossCore Embedded Studio p	rogram		Ť.
Image: Second secon		no_CCES_Core0 Debug Automatic Breakpoints 🔨 🚯 Target Options) 🎼 Custom Board Support ) 🎯 Multiprocessor Groups) 🤯 Source) 🔲 Common )	
Application with CrossCore Debugger	Processor:		
RTOSDemo_CCES_Core0 Debug	Device 0 [Core 0	)] ( Cortex-A5 )	
Application with GDB and OpenOCD (Emulator) Application with GDB and QEMU (Simulator)		et automatically after load:	/
🔋 Launch Group	Label	Description	<u>N</u> ew
► Launch Group (Deprecated)	<ul> <li>✓ ● main</li> <li>✓ ●dbg_,</li> <li>✓ ●stack</li> </ul>	End of program - Fatal error occurred in RTL Start of program Used for assert - Stack overflow detected - Fatal exception occurred in RTL	Edit Delete All
	Enable semih	osting	
<► Filter matched 6 of 6 items		Reyert	Apply
0		Debug	Close

- 4. Click the **Debug** button to close the **Debug Configurations** window
- 5. Click the **Run/Resume** button to start running your application



## 5.1.5 Test Results

Output from the application should be visible within the TTY terminal (e.g. PuTTY/TeraTerm). You should see the LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if all the tests passed.

# 5.2 Running the Basic Example for SHARC+ on ADSP-SC573 EZ-Kit with CrossCore Embedded Studio

## 5.2.1 Overview

This page describes the steps to build and run basic example for SHARC+ on ADSP-SC573 EZ-Kit board using CrossCore Embedded Studio.

## 5.2.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

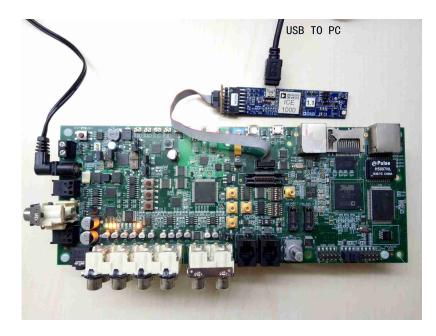
#### Software Requirement

- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio
- FreeRTOS product and the Analog Devices FreeRTOS product. For more inormation please refer to Get the source code ready

#### Hardware Setup

- An ADSCP-SC573 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P1** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below.



## 5.2.3 Build the Example

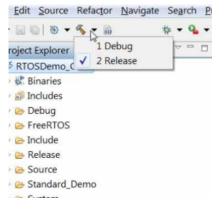
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the File menu and then select the Import option from the menu
- When the **Import** project window appears:
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the FreeRTOSv10.0.\FreeRTOS\Demo\SHARC\_ADSP\_SC573\_CCES folder
- Click **Finish** to close the file browser dialog
- Two projects should appear in the **Project Explorer**

🔀 Import		
Import Projects Select a directory to searc	ch for existing Eclipse projects.	
<ul> <li>Select root directory:</li> <li>Select archive file:</li> <li>Projects:</li> </ul>	C:\Analog Devices\test_case\FreeRTOS\free 👻	Browse Browse
	S_SHARC_Core0 (C:\Analog Devices\test_case\Fn S_SHARC_Core1 (C:\Analog Devices\test_case\Fn	Select All Deselect All Refresh
Options     Search for nested pro     Copy projects into we     Hide projects that alree	-	
Working sets Add project to worki Working sets:	ing sets	New Select
?	< Back Next > Finish	Cancel

2. Choose Debug/Release mode to build the project.



3. Build the project in CrossCore Embedded Studio:

• In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_Core0** and **RTOSDemo\_CCES\_SHARC\_Core1** project, then select the **Build Project** option from the menu

#### 5.2.4 Run the Example

Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_Core0** project and select the **Debug As** option from the menu

2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board

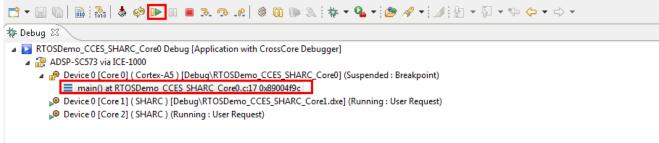
🔀 Debug Configurations		Session Wizard					
Create, manage, and run configurations Ýou must select a debug session first		Select Processor Choose a target processor.					
Image: Contract of the second seco	Name: RTOSDe Session Confi Target: None Platform: Ne Processor: N The following Program	Processor type:           @ ADSP-21584           @ ADSP-21587           @ ADSP-5C570           @ ADSP-5C571           @ ADSP-5C572	SHARC Next > Finish	Configurator Cancel	Source Common Options	Silicon revision	Sele
Filter matched 5 of 10 items						Revert	
?						Debug	3

#### 3. Click the **Debug** button to close the **Debug Configurations** window

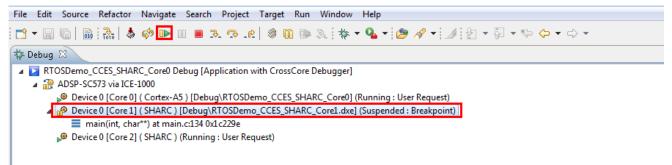
Z Debug Configurations				×
Create, manage, and run configurations Select a debug session to launch and a program to load				to.
Image: Second Secon	Name: RTOSDemo_CCES_SHARC_Core0 Debug  Session @ Automatic Breakpoints 1 Target Options 1 S Custom Board Support Session configuration Target: Emulation Debug Target Platform: ADSP-SC573 via ICE-1000 Processor: ADSP-SC573 The following program(s) will be loaded: Program	Options Reset, Run after load Check si-revision, Run after load	Silicon revision not available not available	Select Session Add Edit Remove Remove All Move Up Move Down Restore Defaults
Filter matched 5 of 10 items			Revert	g Close

#### 4. Choose Core0 and click the **Run/Resume** button to start running Core0 application

File Edit Source Refactor Navigate Search Project Target Run Window Help



#### 5. Then choose Core1 and keep to click Run/Resume button to start running Core1 application



## 5.2.5 Test Results

Output from the application should be visible within the **Console** window in the CrossCore Embedded Studio IDE. You should see three LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if the all tests passed.

🗐 Console 🛛 🤕 Tasks 🖹 Problems 💽 Executables 🗟 Debugger Console

Console 
 Tasks Problems Devices\CrossCore Embedded Studio 2.7.0\SHARC\ldr\ezkitSC573\_preload\_core0"
Load complete.
Lo

# 6 Running the Examples on the ADSP-SC594 EZ-Kit

Processor	Core	Toolchain	Example(s)
ADSP-SC594	ARM A5	CrossCore Embedded Studio	Basic Demo
ADSP-SC594	SHARC+	CrossCore Embedded Studio	Basic Demo
ADSP-SC594	ARM and SHARC+	CrossCore Embedded Studio	MCAPI Demo

The FreeRTOS product for Analog Devices processors contains the following examples:

The basic demo example is based on the **Standard Demo Tasks** that FreeRTOS recommend are provided for each port of the FreeRTOS Operating System. For more information on the Standard Demo Tasks please refer to http://www.freertos.org/a00013.html.

The tasks performed in the Analog Devices Basic Demo include:

- LED flash
- Polled queue tasks
- Recursive Mutex tasks
- Blocking Queue tasks
- Statically allocated tasks
- Suicidal tasks

# 6.1 Running the Basic Example for ARM on ADSP-SC594 EZ-Kit with CrossCore Embedded Studio

#### 6.1.1 Overview

This page describes the steps to build and run basic example for ARM on ADSP-SC594 EZ-Kit board using CrossCore Embedded Studio.

## 6.1.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

#### **Software Requirement**

- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio
- FreeRTOS product and the Analog Devices FreeRTOS product. For more information please refer to Get the source code ready

#### Hardware Setup

- An ADSCP-SC594 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below. A nd connect the **USB to UART** port of the EZ-Kit to the host PC with a USB cable as shown below:



# 6.1.3 Build the Example

Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

- 1. Import the FreeRTOS example into CrossCore Embedded Studio:
  - Select the **File** menu and then select the **Import** option from the menu
  - Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
  - Click the Select root directory radio button and then click the Browse button
  - Browse the root folder where you previously installed the FreeRTOS product and then browse down into the FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX\_A5\_ADSP\_SC594\_CCES folder
  - Click **Finish** to close the file browser dialog
  - A single project should appear in the projects pane of the Import window
  - Check the entry in the **projects** pane and click **Import**

🔀 Import		
Import Projects Select a directory to sea	rch for existing Eclipse projects.	
<ul> <li>Select root directory:</li> <li>Select archive file:</li> <li>Projects:</li> </ul>	C:\Analog Devices\test_case\freertos\FreeRTOSv10	
	_Core0 (C:\Analog Devices\test_case\freertos\FreeRTO:	S Select All Deselect All Refresh
Options     Search for nested pro     Copy projects into we	jects	
<ul> <li>Hide projects that all</li> <li>Working sets</li> <li>Add project to work</li> <li>Working sets:</li> </ul>	eady exist in the workspace ing sets	New Select
?	< Back Next > Finish	Cancel

2. Choose Debug/Release mode to build the project.

Edit Source	Refactor	Navigate	Search P
	5 7 00		* • • •
roject Explorer		ebug	▽ □ □
S RTOSDemo_C	✓ 2 R	elease	
🖗 Binaries			
Includes			
🖻 🗁 Debug			
🗁 FreeRTOS			
😁 Include			
😕 Release			
Source			
😕 Standard_D	)emo		
Curtan			

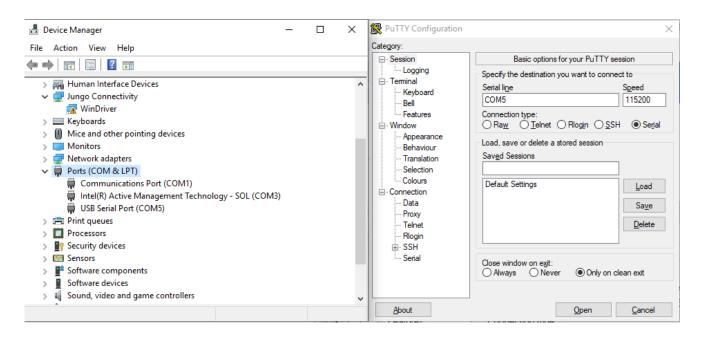
- 3. Build the project in CrossCore Embedded Studio:
  - In the **Project Explorer** right click on the **RTOSDemo\_CCES\_Core0** project and select the **Build Project** option from the menu

## 6.1.4 Run the Example

The semihosting I/O mechanism, which writes to the CCES console during debug sessions, uses SWI interrupts. This is incompatible with default GCC-compiled I/O code which also uses SWI interrupts. For this reason, stdio function calls initiated on the ARM core are routed out over UART instead and shall be read with a serial terminal external to CCES. Importantly, note that:

- 1. This is currently only supported *within* FreeRTOS threads, any studio function call performed out with a thread will crash the application.
- 2. If you need to use other peripherals, you should take care not to change the power service clock rate (which is set in the UART I/O device).

Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the COM port. Configure your serial console application to connect to the port with a baud rate of 115,200.



After this, follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_CCES\_Core0** project and select the **Debug As** option from the menu

2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board

🔀 Debug Configurations				— 🗆 X
Create, manage, and run configura Select a debug session to launch and a				Ť
Application with CrossCore De     TOSDemo_CORTEX_A5_SC     Application with GDB and QEB     Java Application     Launch Group     Launch Group     Launch Group	Name:       RTOSDemo_CORTEX_A5_SC594_CCES Debug         Session       Automatic Breakpoints       Target Options         Session configuration       Target Debug Target         Platform: ADSP-SC594 via ICE-2000       Processor: ADSP-SC594         The following grogram(s) will be loaded:       Program <ul> <li>C:\Analog Devices\CrossCore Embedded Studio 2.10.0\SHAR(</li> <li>RTOSDemo_CORTEX_A5_SC594_CCES\Debug\RTOSDemo_CORT</li> <li></li> <li>Cick here to select a program to load&gt;</li> <li></li> <li></li> <li>Click here to select a program to load&gt;</li> </ul>	Options C\ldr\ezkitSC594W_1 Reset, Run after load	Silicon revision	Select Session Add Edit Remove Remove All Move Up Move Down Restore Defaults
Filter matched 9 of 16 items			Reye	nt Apply bug Close

#### 3. Disable the semihosting function in Automatic Breakpoints

🔀 Debug Configurations		- 🗆 X
Create, manage, and run configura Specify and launch a CrossCore Embed		TO.
Image: Second secon	Name: RTOSDemo_CORTEX_A5_SC594_CCES Debug	
<ul> <li>✓ ► Application with CrossCore De</li> </ul>		
RTOSDemo_CORTEX_A5_SC Application with GDB and Ope		~
Application with GDB and QEN		
Java Applet Java Application	Label Description	<u>N</u> ew
🖏 Launch Group	Image: Program aborted       Image: Program aborted       Image: Program aborted       Image: Program aborted	<u>E</u> dit
Launch Group (Deprecated) Remote Java Application	✓ ●fatal_e       Fatal error occurred in RTL         ✓ ● main       Start of program	<u>D</u> elete
	✓ ● main Start of program	Delete <u>A</u> ll
	Disable Semihosting X	
	Disabling semihosting removes the default semihosting SVC handler. If you want to	
	continue to use semihosting, you should add your own SVC handler.	
	Are you sure you want to disable semihosting?	
	□ Enable semihosting □ Don't ask me again	
< >	Yes No	
Filter matched 9 of 16 items	Re	/ert Apply
?		ebug Close

- 4. Click the **Debug** button to close the **Debug Configurations** window
- 5. Click the Run/Resume button to start running your application

## 6.1.5 Test Results

Output from the application should be visible within the TTY terminal (e.g. PuTTY/TeraTerm). You should see the LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if all the tests passed.

# 6.2 Running the Basic Example for SHARC+ on ADSP-SC594 EZ-Kit with

# CrossCore Embedded Studio

#### 6.2.1 Overview

This page describes the steps to build and run basic example for SHARC+ on ADSP-SC594 EZ-Kit board using CrossCore Embedded Studio.

#### 6.2.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

#### **Software Requirement**

- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio
- FreeRTOS product and the Analog Devices FreeRTOS product. For more information please refer to Get the source code ready

#### **Hardware Setup**

- An ADSCP-SC594 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below.



# 6.2.3 Build the Example

Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

- 1. Import the FreeRTOS example into CrossCore Embedded Studio:
  - Select the **File** menu and then select the **Import** option from the menu
  - Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
  - Click the **Select root directory** radio button and then click the **Browse** button
  - Browse the root folder where you previously installed the FreeRTOS product and then browse down into the FreeRTOSv10.0.\FreeRTOS\Demo\SHARC\_ADSP\_SC594\_CCES folder
  - Click **Finish** to close the file browser dialog
  - Two projects should appear in the **Project Explorer**

🔀 Import				×
Import Projects Select a directory to sear	ch for existing Eclipse projects.			7
• Select root directory:	C:\Analog Devices\freertos\FreeRTOSv1	0.0.0' ~	Browse	
Select archive file:		~	Browse	
Projects:				
	C_SC594_CCES_Core0 (C:\Analog Device		Select Al	II
	C_SC594_CCES_Core1 (C:\Analog Device C_SC594_CCES_Core2 (C:\Analog Device		Deselect A	All
			Refresh	
		>		
Working sets				
Add project to work	ing sets		New	
Working sets:		~	Select	
?	< Back Next > Finis	sh	Cancel	

2. Choose Debug/Release mode to build the project.

Edit Source	Ref	actor	Navigate	Searc	h E
· 🗟 🕼   🕲 🕶	5	- 00		* • •	2 -
roject Explorer		1 De			. 🗆
S RTOSDemo_0	~	2 Re	lease		
🗟 🗟 Binaries					
Includes					
🖻 📴 Debug					
EreeRTOS					
🕑 🗁 Include					
😕 Release					
Source					
😕 Standard_I	Dem	0			
Contant					

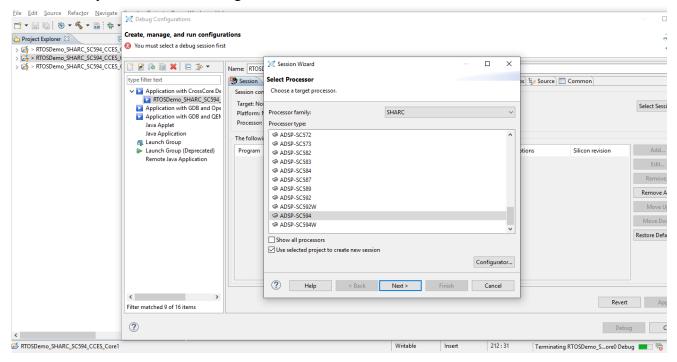
- 3. Build the project in CrossCore Embedded Studio:
  - In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_Core0** and **RTOSDemo\_CCES\_SHARC\_Core1** and **RTOSDemo\_CCES\_SHARC\_Core2** project, then select the **Build Project** option from the menu

#### 6.2.4 Run the Example

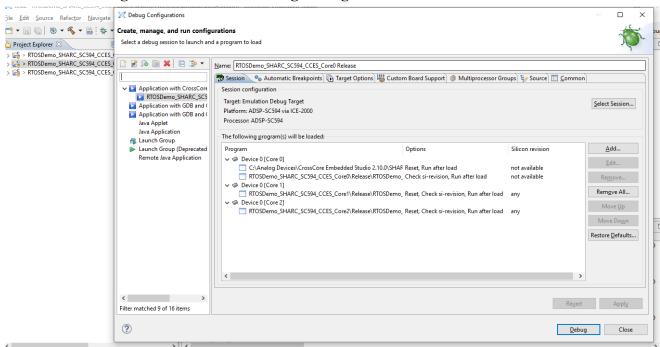
Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_Core1** project and select the **Debug As** option from the menu

2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board



#### 3. Click the **Debug** button to close the **Debug Configurations** window



4. Choose Core0 and click the Run/Resume button to start running Core0 application

5. Then Choose Core1 and Core2 and keep to click **Run/Resume** button to start running Core1 application

## 6.2.5 Test Results

Output from the application should be visible within the **Console** window in the CrossCore Embedded Studio IDE. You should see three LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if the all tests passed.

# 6.3 Running the MCAPI Example on ADSP-SC594 EZ-Kit with CrossCore Embedded Studio

## 6.3.1 Overview

This page describes the steps to build and run MCAPI example for ARM and SHARC+ on ADSP-SC594 EZ-Kit board using CrossCore Embedded Studio.

## 6.3.2 Environment Setup

Before running the MCAPI example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

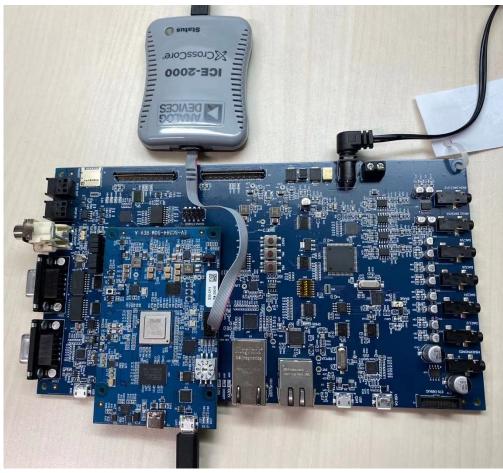
#### **Software Requirement**

- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio
- FreeRTOS product and the Analog Devices FreeRTOS product. For more information please refer to Get the source code ready

#### **Hardware Setup**

- An ADSCP-SC594 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below. A nd connect the **USB to UART** port of the EZ-Kit to the host PC with a USB cable as shown below:



# 6.3.3 Build the Example

Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

- 1. Import the FreeRTOS example into CrossCore Embedded Studio:
  - Select the **File** menu and then select the **Import** option from the menu
  - Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
  - Click the **Select root directory** radio button and then click the **Browse** button
  - Browse the root folder where you previously installed the FreeRTOS product and then browse down into the FreeRTOSv10.0.0
     \FreeRTOS\Demo\CORTEX\_A5\_SHARC\_ADSP\_SC5xx\_CCES\_MCAPI folder
  - Click **Finish** to close the file browser dialog
  - A single project should appear in the **projects** pane of the **Import** window
  - Check the entry in the projects pane and click Import

Import —	
Import Projects Select a directory to search for existing Eclipse projects.	
● Select root directory: RTEX_A5_SHARC_ADSP_SC5xx_CCES_MCAP	B <u>r</u> owse
○ Select <u>a</u> rchive file:	B <u>r</u> owse
Projects:	
RTOSDemo_SC594_MCAPI_Msg_Core0 (C:\Analog Devices\freertos	Select All
RTOSDemo_SC594_MCAPI_Msg_Core1 (C:\Analog Devices\freertos	Deselect All
RTOSDemo_SC594_MCAPI_Msg_Core2 (C:\Analog Devices\freertos	
	R <u>e</u> fresh
Options	
Search for nested projects	
<u>C</u> opy projects into workspace <u>Close newly imported projects upon completion</u>	
Hide projects that already exist in the workspace	
Working sets	
Add projec <u>t</u> to working sets	Ne <u>w</u>
Working sets:	S <u>e</u> lect
(?) < <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel

2. Choose Debug/Release mode to build the project.

Edit Source	Refactor	Navigate	Search P
	5 7 00		* • • •
roject Explorer		ebug	▽ □ □
S RTOSDemo_C	✓ 2 R	elease	
🖗 Binaries			
Includes			
🖻 🗁 Debug			
🗁 FreeRTOS			
😁 Include			
😕 Release			
Source			
😕 Standard_D	)emo		
Curtan			

- 3. Build the project in CrossCore Embedded Studio:
  - In the **Project Explorer** right click on the RTOSDemo\_CCES\_Core0 project and select the **Build Project** option from the menu

#### 6.3.4 Run the Example

The semihosting I/O mechanism, which writes to the CCES console during debug sessions, uses SWI interrupts. This is incompatible with default GCC-compiled I/O code which also uses SWI interrupts. For this reason, stdio function calls initiated on the ARM core are routed out over UART instead and shall be read with a serial terminal external to CCES. Importantly, note that:

- 1. This is currently only supported *within* FreeRTOS threads, any studio function call performed out with a thread will crash the application.
- 2. If you need to use other peripherals, you should take care not to change the power service clock rate (which is set in the UART I/O device).

Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the COM port. Configure your serial console application to connect to the port with a baud rate of 115,200.

🗄 Device Manager 🦳 —	$\times$	🕵 PuTTY Configuration	×
<ul> <li>Device Manager</li> <li>File Action View Help</li> <li>Action View Help</li> <li>Human Interface Devices</li> <li>Jungo Connectivity</li> <li>Keyboards</li> <li>Mice and other pointing devices</li> <li>Monitors</li> <li>Metwork adapters</li> <li>Network adapters</li> <li>Ports (COM &amp; LPT)</li> <li>Communications Port (COM1)</li> <li>Intel(R) Active Management Technology - SOL (COM3)</li> <li>USB Serial Port (COM5)</li> <li>Processors</li> <li>Processors</li> <li>Security devices</li> <li>Sensors</li> <li>Software components</li> <li>Software devices</li> </ul>	×	PuTTY Configuration Category:	Basic options for your PuTTY session Specify the destination you want to connect to Serial line COM5 Connection type: Raw O_Ielnet O Rlogin O_SSH  Serial Load, save or delete a stored session Saved Sessions Default Settings Load Save Delete Cose window on exit: Always O_Never  O Only on clean exit
> 💐 Sound, video and game controllers	¥	About	Open Cancel

After this, follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the RTOSDemo\_CCES\_Core0 project and select the **Debug** As option from the menu

2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board

🔀 Debug Configurations				– 🗆 X	
Create, manage, and run configurations Select a debug session to launch and a program to load				Ť	
Image: Second	Name [RTOSDemo_SC594_MCAPL_Msg_Core1 Debug Session configuration Session configuration Target Emulation Debug Target Platform: ADSP-SC594 The following program(s) will be loaded:				
▲ Launch Group (Deprecated) Eaunch Group (Deprecated) Remote Java Application	Program	Options io 2.10.0/SHARC/Id/iezki35C594W_ Reset, Run after Ioad .RTOSDemo_SC594_MCAPI_Msg_C Check si-revision, Run after Ioad .RTOSDemo_SC594_MCAPI_Msg_C Reset, Check si-revision, Run after Ioad .RTOSDemo_SC594_MCAPI_Msg_C Reset, Check si-revision, Run after Ioad		Add Edit Remove Remove All Move Up Move Up Move Down Restore Defaults	
Filter matched 9 of 16 items			Reve	rt Apply	
0			Deb	Close	

#### 3. Disable the semihosting function in Automatic Breakpoints

Debug Configurations		- 🗆 X			
Create, manage, and run configurations Specify and launch a CrossCore Embedded Studio program		Ť.			
🖺 🖻 🐌 🗎 🗶 🖻 🔅 🗸	Name: RTOSDemo_SC594_MCAPI_Msg_Core1 Debug				
type filter test 🛞 Session 🗞 Automatic Breakpoints 🕞 Target Options 🚻 Custom Board Support 🛞 Multiprocessor Groups 💱 Source 🗔 Common					
<ul> <li>Application with CrossCore Debugger</li> </ul>	Processor:				
<ul> <li>RTOSDemo_SC594_MCAPI_Msg_Core1 Debug</li> <li>Application with GDB and OpenOCD (Emulator)</li> </ul>	Device 0 [Core 0] ( Cortex-A5 ) V				
Application with GDB and QEMU (Simulator)	Breakpoints to set automatically after load:				
Java Applet Java Application	Label Description	New			
🚙 Launch Group	✓     ● abort     Program aborted       ✓     ● _exit     End of program	Edit			
Launch Group (Deprecated) Remote Java Application	☑ ●fatal_e Fatal error occurred in RTL	Delete			
Renote Java Application	☑ ● main Start of program	Delete All			
	Enable semihosting				
Filter matched 9 of 16 items		Revert Apply			
?		Debug Close			

- 4. Click the **Debug** button to close the **Debug Configurations** window
- 5. Click the **Run/Resume** button to start running your application for three cores.

#### 6.3.5 Test Results

Output from the application should be visible within the TTY terminal (e.g. PuTTY/TeraTerm) and **Console** window in the CrossCore Embedded Studio IDE . You should see the LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if all the tests passed.

# 7 Running the Examples on the ADSP-BF7XX EZ-Kit

The FreeRTOS product for Analog Devices processors contains the following examples:

Processor Toolchain		Example(s)
ADSP-BF707	CrossCore Embedded Studio	Basic Demo

The basic demo example is based on the **Standard Demo Tasks** that FreeRTOS recommend are provided for each port of the FreeRTOS Operating System.

For more information on the Standard Demo Tasks please refer to http://www.freertos.org/a00013. html.

The tasks performed in the Analog Devices Basic Demo include:

- LED flash
- Polled queue tasks
- Recursive Mutex tasks
- Blocking Queue tasks
- Statically allocated tasks
- Suicidal tasks

# 7.1 Running the Basic Example for ADSP-BF707 EZ-Kit with CrossCore Embedded Studio

## 7.1.1 Overview

This page describes the steps required to build and run basic example on ADSP-BF707 EZ-Kit board using CrossCore Embedded Studio.

## 7.1.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

#### **Software Requirement**

• Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio

• FreeRTOS product and the Analog Devices FreeRTOS product. For more inormation please refer to Get the source code ready

#### Hardware Setup

- An ADSP-BF707 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P3** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 5 volts as in the diagram below



## 7.1.3 Build the Example

Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Select the **File** menu and then select the **Import** option.

- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button

- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the \FreeRTOSv10.0.\FreeRTOS\Demo\Blackfin\_ADSP\_BF707\_CCES folder
- Click **OK** to close the file browser dialog
- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the **projects** pane and click **Finish**

🔀 Import	
Import Projects Select a directory to search for existing Eclipse projects.	
<ul> <li>Select root directory: reeRTOS\Demo\Blackfin_ADSP_BF707_CCE\$ ▼</li> <li>Select archive file: ▼</li> </ul>	Browse
Projects:	
RTOSDemo_BF707 (C:\Analog Devices\freertos\FreeRTOSv10.0.0\F	Select All Deselect All Refresh
۰ III ا	
Options Search for nested projects Copy projects into workspace Hide projects that already exist in the workspace	
Working sets	
Add project to working sets Working sets:	New Select
Reck Next > Finish	Cancel

#### 2. Choose Debug/Release mode to build the project.

File Edit Source Refactor Navigate	e Search Project Run Window Help
📬 🖬 🕼   🛞 🕶 🔦 🖬 İ 🎋 🕶 🍳	<b>▼ : ②                                  </b>
ြာ Project Explorer 🛛	

- 3. Build the project in CrossCore Embedded Studio
  - In the **Project Explorer** right click on the **RTOSDemo\_BF707** project and select the **Build Project** option from the menu

# 7.1.4 Run the Example

Follow below four steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_BF707** project and select the **Debug As** option from the menu

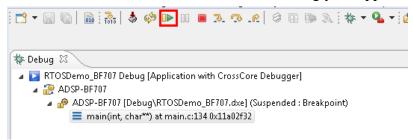
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board

Z Debug Configurations					23
Create, manage, and run configurations ② You must select a debug session first					Ť.
Image: The set of the se	Session Wizard         Select Processor         Choose a target processor.         Processor family:         Processor type:         ADSP-BF607         ADSP-BF708         ADSP-BF709         ADSP-BF701         ADSP-BF703         ADSP-BF704         ADSP-BF705         ADSP-BF706         ADSP-BF707         ADSP-BF708         ADSP-BF709         ADSP-BF701         ADSP-BF703         ADSP-BF704         ADSP-BF705         ADSP-BF706         ADSP-BF707         Show all processors         V Use selected project to create new session         Configure         Prose         Help       < Back	× E v	:essor Groups) 🖅 So Options	urce Common	Select Session Edit Remove Remove All Move Up Move Down Restore Defaults
Filter matched 5 of 12 items				Rever	t Apply
?				Deb	ug Close

#### 3. Click the **Debug** button to close the **Debug Configuration** window

🔀 Debug Configurations 🛛 💽					
Create, manage, and run configurations Select a debug session to launch and a program to load					
Application with CrossCore Debugger     Application with CossCore Debugger     Application with GDB and OpenOCD (Emulator)     Application with GDB and QEMU (Simulator)     Launch Group	Name: RTOSDerno_BF707 Debug Session Onfiguration Target: Blackfin CS Emulator Platform: ADSP-BF707 Target: Blackfin CS Emulator Platform: ADSP-BF707 Platform: ADSP-BF707				
	The following program(s) will be loaded:  Program  RTOSDemo_BF707\Debug\RTOSDemo_BF707.dxe	Options Reset, Check si-revision, Run after load	Silicon revision any	Add Edit Remove Remove All Move Up Move Down Restore Defaults	
m     Filter matched 5 of 12 items			Rever	t Apply	
?			Deb	ug Close	

4. Click the Run/Resume button to start running your application



## 7.1.5 Test Results

Output from the application should be visible within the **Console** window in the CrossCore Embedded Studio IDE. You should see three LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if the all tests passed.

```
Output
Loading application: "C:\Analog Devices\freertos\FreeRTOSv10.0.0\FreeRTOS\Demo\Blackfin_ADSP_BF707_CCES\Debug\RTOSDemo_BF707.dxe"
Load complete.
Test passed
Test passed
Test passed
Test passed
```

# 8 Running the Examples on the ADSP-21569 EZ-Kit

The FreeRTOS product for Analog Devices processors contains the following examples:

Processor Toolchain		Toolchain	Example(s)
	ADSP-21569	CrossCore Embedded Studio	Basic Demo

The basic demo example is based on the **Standard Demo Tasks** that FreeRTOS recommend are provided for each port of the FreeRTOS Operating System.

For more information on the Standard Demo Tasks please refer to http://www.freertos.org/a00013. html.

The tasks performed in the Analog Devices Basic Demo include:

- LED flash
- Polled queue tasks
- Recursive Mutex tasks
- Blocking Queue tasks
- Statically allocated tasks
- Suicidal tasks

# 8.1 Running the Basic Example for ADSP-21569 EZ-Kit with CrossCore Embedded Studio

#### 8.1.1 Overview

This page describes the steps required to build and run basic example on ADSP-21569 EZ-Kit board using CrossCore Embedded Studio.

## 8.1.2 Environment Setup

Before running the basic example with CrossCore Embedded Studio, you should make some preparation for environment setup including software and hardware.

#### **Software Requirement**

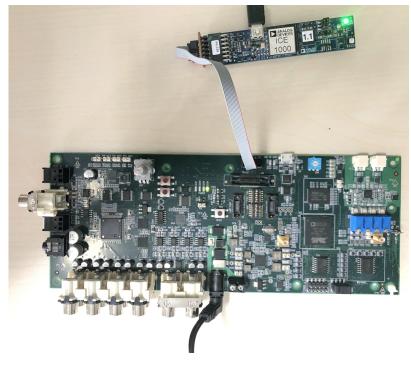
• Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio

• FreeRTOS product and the Analog Devices FreeRTOS product. For more inormation please refer to Get the source code ready

#### Hardware Setup

- An ADSP-21569 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P4** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below



# 8.1.3 Build the Example

Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Select the **File** menu and then select the **Import** option.

- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the FreeRTOSv10.0.0 \FreeRTOS\Demo\SHARC\_ADSP\_21569\_CCES\RTOSDemo\_CCES\_SHARC\_21569 folder
- Click **OK** to close the file browser dialog

- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the **projects** pane and click **Finish**

22 Import	- • ×
Import Projects	
Select a directory to search for existing Eclipse projects.	
● Select root directory: <a href="https://cces.krosbemo_cces_sharc_21569">cces.krosbemo_cces_sharc_21569</a> <a href="https://ces.krosbemo_cces_sharc_21569"></a>	B <u>r</u> owse
Select <u>a</u> rchive file:	B <u>r</u> owse
Projects:	
RTOSDemo_CCES_SHARC_21569 (C:\Analog Devices\test_case\freertos-bitbuc	<u>S</u> elect All
	Deselect All
	R <u>e</u> fresh
4	
Options	
Searc <u>h</u> for nested projects	
<ul> <li>Copy projects into workspace</li> <li>Hide projects that already exist in the workspace</li> </ul>	
Working sets	
Add project to working sets	New
Working sets:	S <u>e</u> lect
working sets:	S <u>e</u> lect
? < Back Next > Einish	Cancel

#### 2. Choose Debug/Release mode to build the project.

File Edit Source Refactor Navigate	Search Project Run Window Help
📬 = 😨   🗞 + 🔦 - 💁	▼   🥙 🖋 ▼   📴 🗐 👔   ½ ▼ 🖓 ▼ ጐ 🔶 ▼
Project Explorer 🛛	

3. Build the project in CrossCore Embedded Studio

• In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_21569** project and select the **Build Project** option from the menu

#### 8.1.4 Run the Example

Follow below four steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo\_CCES\_SHARC\_21569** project and select the **Debug As** option from the menu

2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator(ICE1000 or ICE2000) and target board

🔀 Debug Configurations		×
Create manage and run configuration	🖉 Session Wizard 📃 📼 💌	1
Create, manage, and run configuration	Select Processor Choose a target processor.	
Image: Second secon	Processor family: SHARC   Processor type:	Irce *1
<ul> <li>RTOSDemo_CCES_SHARC_21565</li> <li>Application with GDB and OpenOC</li> <li>Application with GDB and QEMU (S</li> </ul>		t Session
<ul><li>♣ Launch Group</li><li>▶ Launch Group (Deprecated)</li></ul>	<ul> <li>ADSP-21573</li> <li>ADSP-21583</li> <li>ADSP-21584</li> </ul>	=
		<u>A</u> dd <u>E</u> dit
	□ Show all processors         ✓ Use selected project to create new session	
Filter matched 6 of 8 items	<u>Configurator</u>	Apply
(?)	Image: Marking text     Image: Marking text     Einish     Cancel	Close

THE INTERSECTION S. COUSE WITH DATABUT. SIZE COLVER THE LARSE.

#### 3. Click the **Debug** button to close the **Debug Configurations** window

Z Debug Configurations			×		
Create, manage, and run configurations					
Select a debug session to launch and a pr	ogram to load		Ę		
	Name: RTOSDemo_CCES_SHARC_21569 Debug				
type filter text  Application with CrossCore Debugg	Session • Automatic Breakpoints Target Options 🐻 Custom Board Support 🕏 N	Iultiprocessor Groups	Source		
<ul> <li>RTOSDemo_CCES_SHARC_21569</li> <li>Application with GDB and OpenOCE</li> <li>Application with GDB and QEMU (Sii <ul> <li>Launch Group</li> </ul> </li> </ul>	Session configuration Target: Emulation Debug Target Platform: ADSP-21569 via ICE-1000 Processor: ADSP-21569	Select Session			
Launch Group (Deprecated)	The following <u>p</u> rogram(s) will be loaded:				
	Program Options C:\Analog Devices\CrossCore Embedded Studio Reset, Run after load	Silicon revision	Add		
	C. (Analog Devices (Close of embedded studio Keser, Auf arter load     RTOSDemo_CCES_SHARC_21569\Debug\RTOSE Check si-revision, Run after load	any any	Edit		
			Remove All		
			Move Up		
			Move Do <u>w</u> n		
			Restore <u>D</u> efaults		
Filter matched 6 of 8 items		Revert	Apply		
0		<u>D</u> ebug	Close		

#### 4. Click the **Run/Resume** button to start running your application

```
      Eile Edit Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Window Help

      Image: Source Refactor Navigate Search Target Project Run Projec
```

## 8.1.5 Test Results

Output from the application should be visible within the **Console** window in the CrossCore Embedded Studio IDE. You should see the LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if the all tests passed.

```
Console Console Tasks Problems © Executables 

Output
Load complete.
Test passed
Test passe
Test passed
Test passed
Test passed
Test passed
Test passed
T
```

# 9 Using CrossCore Embedded Studio System Services and Device Drivers with FreeRTOS

Note: This section of the document applies to the ADSP-SC5xx (Cortex-A and SHARC+) and ADSPBF7xx processors. It does not apply to the ADuCM\* processor families.

CrossCore Embedded Studio provides support for the on-chip peripherals and EZ-KIT hosted device drivers that are provided for its processors.

In order to use these features with FreeRTOS the source based versions of the drivers must be used rather than the default pre-built libraries that are provided.

Use of the library based version of the System Services and Device Drivers is not compatible with FreeRTOS. Use of the pre-built libraries may result in run-time corruption and execution failure.

To use the System Services and Device Drivers in your CrossCore Embedded Studio project:

1. Ensure that the pre-processor macro \_\_\_ADI\_FREERTOS is defined for all assembler, C/C++ and linker operations.

This requires the pre-processor macro to defined in three separate locations in the project settings.

- 2. Ensure that the pre-built libdrv library is not linked into the application.
  - a. For Cortex-A projects this is controlled by adding the following option to the Settings >Tool Settings > CrossCore ARM Bare Metal C Linker > Additional Options settings: specs=PATH\_TO\_FREERTOS\FreeRTOS\FreeRTOSv10.0.0
     \FreeRTOSSource\portable\CCES\ARM\_CA5\freertos.specs where

     PATH\_TO\_FREERTOS is replaced with the path to the installation of your FreeRTOS product.
  - b. For SHARC+ and Blackfin projects this is controlled by checking the Settings > Tool Settings > CrossCore Blackfin/SHARC Linker > Libraries > Omit device driver library checkbox
- 3. Enable the source based version of the required services and drivers:
  - a. Double click the system.svc file in the Project Explorer
  - b. Click the Add button in the System Configuration Overview
  - c. Browse the list of Device Drivers and System Services to add new components to the project

#### 4. Build the project

The provided demo examples for the EZ-KITs contain all the appropriate project settings already configured and are an easy way to get started with a new FreeRTOS project

# **10 Appendix A: FreeRTOS Performance**

The following appendix contains code size and performance data for Analog Devices specific ports of FreeRTOS.

#### **Timer Cycles**

The following benchmarks report time and cycle count measurements for post and pending operations using varying methods of communication.

Benchmark data is available for the following EZ-Kits:

- ADSP-SC589 EZ-Kit (Cortex A5 Core)
- ADSP-SC589 EZ-Kit (SHARC+ Core)
- ADSP-SC573 EZ-Kit (Cortex A5 Core)
- ADSP-21569 EZ-Kit (SHARC Core)
- ADSP-BF707 EZ-Kit
- The following projects are executed to gather the benchmark data:
- **ISR:** calculate Interrupt service time and Time to return from an ISR when in FreeRTOS system.
- **FLAG ISR:** calculate FLAG Post/Pend available time,context switch time when unavailable, Interrupt service time and Time to return from an ISR when in FreeRTOS system
- **MSG ISR:** calculate Message queue Post/Pend available time,context switch time when unavailable, Interrupt service time and Time to return from an ISR when in FreeRTOS system
- **SEM ISR:** calculate Semaphore Post/Pend available time,context switch time when unavailable, Interrupt service time and Time to return from an ISR when in FreeRTOS system
- **MUT ISR:** calculate Mutex Post/Pend available time,context switch time when unavailable, Interrupt service time and Time to return from an ISR when in FreeRTOS system

#### Spaces

The following benchmarks report code size for several common RTOS operations within FreeRTOS. The benchmark data is available for the following EZ-Kits:

- ADSP-SC589 EZ-Kit (Cortex A5 Core)
- ADSP-SC589 Ez-Kit (SHARC+ Core)
- ADSP-21569 EZ-Kit (SHARC Core)

#### • ADSP-BF707 EZ-Kit

The following projects are executed to gather the benchmark data:

- NONE: Basic project
- Message Queues: Basic project using 1 static object / Basic project using 2 static objects
- Flags: Basic project using 1 static object / Basic project using 2 static objects
- Mutexes: Basic project using 1 static object / Basic project using 2 static objects
- Semaphores: Basic project using 1 static object / Basic project using 2 static objects
- ALL: Basic project using 1 static object / Basic project using 2 static objects

## 10.1 ADSP-21569 (SHARC Core) Benchmark Data

#### **ADSP-21569 SHARC Core Performance Metrics**

		cycles
FreeRTOS_FLGISR	xEventGroupWaitBits (flag available)	314
	xEventGroupWaitBits (flag unavailable, context switch to new task)	1244
	xEventGroupSetBits (no task pending, no context switch)	255
	xEventGroupSetBits (task waiting, context switch to pending task)	1156
	xEventGroupSetBits (from an ISR, switching to a pending task)	3742
FreeRTOS_ISR	Interrupt service time (FreeRTOS)	227
	Time to return from an ISR (FreeRTOS, no task switch)	174
FreeRTOS_MSGISR	xQueueReceive(message available)	298
	xQueueReceive(message unavailable, context switch to new task)	2108
	xQueueSend(no task pending, no context switch)	363
	xQueueSend(task waiting, context switch to pending task)	1434

		cycles
	xQueueSend(from an ISR, switching to a pending task)	1097
FreeRTOS_MUTISR	xSemaphoreTake(mutex available)	238
	xSemaphoreTake(mutex unavailable, context switch to new task	2411
	xSemaphoreGive(no task pending, no context switch)	329
	xSemaphoreGive(task waiting, context switch to pending task)	1540
FreeRTOS_SEMISR	xSemaphoreTake(semaphore available)	213
	xSemaphoreTake(semaphore unavailable, context switch to new task)	2097
	xSemaphoreGive(no task pending, no context switch)	313
	xSemaphoreGive(task waiting, context switch to pending task)	1293
	xSemaphoreGive (from an ISR, switching to a pending task)	1029

# **ADSP-21569 SHARC Core Sizing Metrics**

		Data	Code	Total
NONE	Basic project	59307	29842	89149
Message Queues	Basic project using 1 static object	59403	29970	89373
	Basic project using 2 static objects	59483	29970	89453
Flags	Basic project using 1 static object	59459	31702	91161
	Basic project using 2 static objects	59491	31702	91193
Mutexes	Basic project using 1 static object	59395	32114	91509

		Data	Code	Total
	Basic project using 2 static objects	59483	32114	91597
Semaphores	Basic project using 1 static object	59395	31982	91377
	Basic project using 2 static objects	59483	31982	91465
ALL	Basic project using 1 static object	59547	34198	93745
	Basic project using 2 static objects	59667	34198	93865

# 10.2 ADSP-SC589 (Cortex-A Core) Benchmark Data

		cycles
FreeRTOS_FLGISR	xEventGroupWaitBits (flag available)	321
	xEventGroupWaitBits (flag unavailable, context switch to new task)	1129
	xEventGroupSetBits (no task pending, no context switch)	324
	xEventGroupSetBits (task waiting, context switch to pending task)	1095
	xEventGroupSetBits (from an ISR, switching to a pending task)	3158
FreeRTOS_ISR	Interrupt service time (FreeRTOS)	109
	Time to return from an ISR (FreeRTOS, no task switch)	24
FreeRTOS_MSGISR	xQueueReceive(message available)	287
	xQueueReceive(message unavailable, context switch to new task)	2290
	xQueueSend(no task pending, no context switch)	309
	xQueueSend(task waiting, context switch to pending task)	1309

		cycles
	xQueueSend(from an ISR, switching to a pending task)	648
FreeRTOS_MUTISR	xSemaphoreTake(mutex available)	239
	xSemaphoreTake(mutex unavailable, context switch to new task	2630
	xSemaphoreGive(no task pending, no context switch)	274
	xSemaphoreGive(task waiting, context switch to pending task)	1438
FreeRTOS_SEMISR	xSemaphoreTake(semaphore available)	212
	xSemaphoreTake(semaphore unavailable, context switch to new task)	2442
	xSemaphoreGive(no task pending, no context switch)	317
	xSemaphoreGive(task waiting, context switch to pending task)	1344
	xSemaphoreGive (from an ISR, switching to a pending task)	638

# ADSP-SC589 Cortex-A Core Sizing Metrics

		Data	Code	Total
NONE	Basic project	97812	18624	116436
Message Queues	Basic project using 1 static object	97812	18680	116492
	Basic project using 2 static objects	97812	18680	116492
Flags	Basic project using 1 static object	97812	19096	116908
	Basic project using 2 static objects	97812	19096	116908
Mutexes	Basic project using 1 static object	97812	18704	116516

		Data	Code	Total
	Basic project using 2 static objects	97812	18704	116516
Semaphores	Basic project using 1 static object	97812	18664	116476
	Basic project using 2 static objects	97812	18664	116476
ALL	Basic project using 1 static object	97812	19264	117076
	Basic project using 2 static objects	97812	19256	117068

# 10.3 ADSP-SC589 (SHARC+ Core) Benchmark Data

## **ADSP-SC589 SHARC+ Core Performance Metrics**

		cycles
FreeRTOS_FLGISR	xEventGroupWaitBits (flag available)	440
	xEventGroupWaitBits (flag unavailable, context switch to new task)	1708
	xEventGroupSetBits (no task pending, no context switch)	340
	xEventGroupSetBits (task waiting, context switch to pending task)	1586
	xEventGroupSetBits (from an ISR, switching to a pending task)	4643
FreeRTOS_ISR	Interrupt service time (FreeRTOS)	236
	Time to return from an ISR (FreeRTOS, no task switch)	182
FreeRTOS_MSGISR	xQueueReceive(message available)	409
	xQueueReceive(message unavailable, context switch to new task)	2710
	xQueueSend(no task pending, no context switch)	476

		cycles
	xQueueSend(task waiting, context switch to pending task)	1877
	xQueueSend(from an ISR, switching to a pending task)	1160
FreeRTOS_MUTISR	xSemaphoreTake(mutex available)	321
	xSemaphoreTake(mutex unavailable, context switch to new task	3199
	xSemaphoreGive(no task pending, no context switch)	484
	xSemaphoreGive(task waiting, context switch to pending task)	2117
FreeRTOS_SEMISR	xSemaphoreTake(semaphore available)	275
	xSemaphoreTake(semaphore unavailable, context switch to new task)	2711
	xSemaphoreGive(no task pending, no context switch)	404
	xSemaphoreGive(task waiting, context switch to pending task)	1718
	xSemaphoreGive (from an ISR, switching to a pending task)	1019

# ADSP-SC589 SHARC+ Core Sizing Metrics

		Data	Code	Total
NONE	Basic project	59838	29764	89602
Message Queues	Basic project using 1 static object	59934	29896	89830
	Basic project using 2 static objects	59934	29896	89830
Flags	Basic project using 1 static object	59966	31692	91658
	Basic project using 2 static objects	59966	31692	91658

		Data	Code	Total
Mutexes	Basic project using 1 static object	59926	32184	92110
	Basic project using 2 static objects	59926	32184	92110
Semaphores	Basic project using 1 static object	59926	32048	91974
	Basic project using 2 static objects	59926	32048	91974
ALL	Basic project using 1 static object	60054	34340	94394
	Basic project using 2 static objects	60054	34340	94394

# 10.4 ADZS-BF707 Benchmark Data

#### **ADZS-BF707** Performance Metrics

		cycles
FreeRTOS_FLGISR	xEventGroupWaitBits (flag available)	420
	xEventGroupWaitBits (flag unavailable, context switch to new task)	1856
	xEventGroupSetBits (no task pending, no context switch)	356
	xEventGroupSetBits (task waiting, context switch to pending task)	1866
	xEventGroupSetBits (from an ISR, switching to a pending task)	3158
FreeRTOS_ISR	Interrupt service time (FreeRTOS)	98
	Time to return from an ISR (FreeRTOS, no task switch)	126
FreeRTOS_MSGISR	xQueueReceive(message available)	447
	xQueueReceive(message unavailable, context switch to new task)	2937

		cycles
	xQueueSend(no task pending, no context switch)	630
	xQueueSend(task waiting, context switch to pending task)	2412
	xQueueSend(from an ISR, switching to a pending task)	1417
FreeRTOS_MUTISR	xSemaphoreTake(mutex available)	337
	xSemaphoreTake(mutex unavailable, context switch to new task	3553
	xSemaphoreGive(no task pending, no context switch)	663
	xSemaphoreGive(task waiting, context switch to pending task)	2624
FreeRTOS_SEMISR	xSemaphoreTake(semaphore available)	271
	xSemaphoreTake(semaphore unavailable, context switch to new task)	2943
	xSemaphoreGive(no task pending, no context switch)	504
	xSemaphoreGive(task waiting, context switch to pending task)	2093
	xSemaphoreGive (from an ISR, switching to a pending task)	1235

#### **ADZS-BF707 Sizing Metrics**

		Data	Code	Total
NONE	Basic project	6535	13698	20233
Message Queues	Basic project using 1 static object	6623	13762	20385
	Basic project using 2 static objects	6707	13762	20469
Flags	Basic project using 1 static object	6567	14402	20969

		Data	Code	Total
	Basic project using 2 static objects	6599	14402	21001
Mutexes	Basic project using 1 static object	6619	14682	21301
	Basic project using 2 static objects	6703	14682	21385
Semaphores	Basic project using 1 static object	6619	14618	21237
	Basic project using 2 static objects	6703	14618	21321
ALL	Basic project using 1 static object	6655	15506	22161
	Basic project using 2 static objects	6771	15506	22277