

FreeRTOS User's Guide

FreeRTOS User's Guide Version 1.4.0, September 2019 © 2019 Analog Devices, Inc. http://www.analog.com processor.tools.support@analog.com

Contents

1	Intro	oduction		7						
	1.1	Analog	g Devices FreeRTOS	7						
	1.2	What's	s in the user's guide	7						
2	Hard	dware a	nd software set up	9						
	2.1	1 Get the hardware ready								
	2.2	Get th	e source code ready	10						
		2.2.1	Download FreeRTOS source code	10						
		2.2.2	Install the FreeRTOS product from Analog Devices	10						
	2.3	3 Software environment set up for CrossCore Embedded Studio								
	2.4	Softwa	are environment set up for IAR Embedded Workbench	11						
	2.5	Softwa	are environment set up for Keil MDK	12						
3	Run	ning the	Examples on the ADuCM302x EZ-Kit	14						
	3.1	Runni	ng the Basic Example for ADuCM302x EZ-Kit with CrossCore Embedded Studio	14						
		3.1.1	Overview	14						
		3.1.2	Environment Setup	14						
		3.1.3	Build the Example	15						
		3.1.4	Run the Example	17						
		3.1.5	Test Results	18						
	3.2	Runni	ng the Basic Example for ADuCM302x EZ-Kit with IAR Embedded Workbench	19						
		3.2.1	Overview	19						
		3.2.2	Environment Setup	19						
		3.2.3	Build the Example	20						
		3.2.4	Run the Example	21						
		3.2.5	Test Results	22						
	3.3	Runni	ng the Basic Example for ADuCM302x EZ-Kit with Keil MDK	22						
		3.3.1	Overview	22						
		3.3.2	Environment Setup	22						
		3.3.3	Build the Example	23						
		3.3.4	Run the Example	24						
		3.3.5	Test Results	25						
4	Run	ning the	Examples on the ADuCM4x50 EZ-Kit	26						
	4.1	Runni	ng the Basic Example for ADuCM4x50 EZ-Kit with CrossCore Embedded Studio	26						
		4.1.1	Overview	26						
		4.1.2	Environment Setup	26						
		4.1.3	Build the Example	27						
		4.1.4	Run the Example	29						
		4.1.5	Test Results	30						
	4.2	Runni	ng the Basic Example for ADuCM4x50 EZ-Kit with IAR Embedded Workbench	30						
		4.2.1	Overview	30						
		4.2.2	Environment Setup	31						

		4.2.3	Build the Example	31
		4.2.4	Run the Example	33
		4.2.5	Test Results	34
4	1.3	Runnin	g the Basic Example for ADuCM4x50 EZ-Kit with Keil MDK	34
		4.3.1	Overview	34
		4.3.2	Environment Setup	34
		4.3.3	Build the Example	35
		4.3.4	Run the Example	36
		4.3.5	Test Results	37
5 F	Run	ning the	Examples on the ADSP-SC589 EZ-Kit	38
5	5.1	Runnin	g the Basic Example for ARM on ADSP-SC589 EZ-Kit with CrossCore Embedded Studio	38
		5.1.1	Overview	38
		5.1.2	Environment Setup	38
		5.1.3	Build the Example	40
		5.1.4	Run the Example	42
		5.1.5	Test Results	44
5	5.2	Runnin	ig the Basic Example for SHARC+ on ADSP-SC589 EZ-Kit with CrossCore Embedded Studie	0
44	1			
		5.2.1	Overview	44
		5.2.2	Environment Setup	44
		5.2.3	Build the Example	45
		5.2.4	Run the Example	47
		5.2.5	Test Results	49
5	5.3	Runnin	g the LwIP Example for ARM on ADSP-SC589 EZ-Kit with CrossCore Embedded Studio	49
		5.3.1	Overview	49
		5.3.2	Environment Setup	49
		5.3.3	Build the Example	51
		5.3.4	Run the Example	53
		5.3.5	Test Results	55
6 F	Run	ning the	Examples on the ADSP-SC584 EZ-Kit	57
6	5.1	Runnin	g the Basic Example for ARM on ADSP-SC584 EZ-Kit with CrossCore Embedded Studio	57
		6.1.1	Overview	57
		6.1.2	Environment Setup	57
		6.1.3	Build the Example	59
		6.1.4	Run the Example	61
		6.1.5	Test Results	63
6	6.2	Runnin	ig the Basic Example for SHARC+ on ADSP-SC584 EZ-Kit with CrossCore Embedded Studio	0
63	3			
		6.2.1	Overview	63
		6.2.2	Environment Setup	63
		6.2.3	Build the Example	64
		6.2.4	Run the Example	66
		6.2.5	Test Results	67

6.3 Running the LwIP Example for ARM on ADSP-SC584 EZ-Kit with CrossCore Embedded Studio 68

		6.3.1	Overview	68
		6.3.2	Environment Setup	68
		6.3.3	Build the Example	69
		6.3.4	Run the Example	71
		6.3.5	Test Results	74
7	Run	ning the	Examples on the ADSP-SC573 EZ-Kit	75
	7.1	Runnin	g the Basic Example for ARM on ADSP-SC573 EZ-Kit with CrossCore Embedded Studio	75
		7.1.1	Overview	75
		7.1.2	Environment Setup	75
		7.1.3	Build the Example	77
		7.1.4	Run the Example	79
		7.1.5	Test Results	81
	7.2	Runnin	g the Basic Example for SHARC+ on ADSP-SC573 EZ-Kit with CrossCore Embedded Stuc	lio
	81			
		7.2.1	Overview	81
		7.2.2	Environment Setup	81
		7.2.3	Build the Example	82
		7.2.4	Run the Example	84
		7.2.5	Test Results	85
	7.3	Runnin	g the LwIP Example for ARM on ADSP-SC573 EZ-Kit with CrossCore Embedded Studio	86
		7.3.1	Overview	86
		7.3.2	Environment Setup	86
		7.3.3	Build the Example	87
		7.3.4	Run the Example	89
		7.3.5	Test Results	92
8	Run	ning the	Examples on the ADSP-BF7XX EZ-Kit	94
	8.1	Runnin	g the Basic Example for ADSP-BF707 EZ-Kit with CrossCore Embedded Studio	94
		8.1.1	Overview	94
		8.1.2	Environment Setup	94
		8.1.3	Build the Example	95
		8.1.4	Run the Example	97
		8.1.5	Test Results	98
9	Run	ning the	Examples on the ADSP-21569 EZ-Kit	99
	9.1	Runnin	g the Basic Example for ADSP-21569 EZ-Kit with CrossCore Embedded Studio	99
		9.1.1	Overview	99
		9.1.2	Environment Setup	99
		9.1.3	Build the Example	100
		9.1.4	Run the Example	101
		9.1.5	Test Results	103
10) Usii	ng Cross	Core Embedded Studio System Services and Device Drivers with FreeRTOS	104
11	Арр	pendix A:	FreeRTOS Performance	106
	11.1	ADSF	P-21569 (SHARC Core) Benchmark Data	107
	11.2	2 ADSF	P-SC589 (Cortex-A Core) Benchmark Data	109
	11.3	B ADSF	P-SC589 (SHARC+ Core) Benchmark Data	111

11.4	ADuCM3029 Benchmark Data	113
11.5	ADuCM4050 Benchmark Data	115
11.6	ADZS-BF707 Benchmark Data	117

Copyright Information

© 2019 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, A²B, 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 following development environments:

- CrossCore Embedded Studio
- IAR Embedded Workbench
- Keil MDK5

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

Processors Supported	Examples Provided For
ADSP-SC5xx Cortex-A5 Core	ADSP-SC589 EZ-Kit ADSP-SC584 EZ-Kit ADSP-SC573 EZ-Kit
ADSP-SC5xx SHARC+ Core	ADSP-SC589 EZ-Kit ADSP-SC584 EZ-Kit ADSP-SC573 EZ-Kit
ADSP-BF7xx	ADSP-BF707 EZ-Kit
ADuCM302x	ADuCM3029 EZ-Kit

Processors Supported	Examples Provided For
ADuCM405x	ADuCM4050 EZ-Kit
ADSP-2156x	ADSP-21569 EZ-Kit

Detailed description for setting up the hardware and software environment, and how to run the demo exmaples 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 a couple of reference development board from Analog Devices, including ADuCM3029/4050 EZ-Kit board, ADSP-SC589/ADSP-SC584/ADSP-SC573 EZ-Kit board, BF707 EZ-Kit board and ADSP-21569 EZ-Kit board. Depending on which software development tool you are using, different JTAG debug board are required.

Below is a list of the hardware involved.

ADI reference board:

- ADuCM3029 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/eval-aducm3029-ezkit.htmL
- ADuCM4050 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/ADZS-U4050LF-EZKIT.html
- 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
- J-Link: https://www.segger.com/products/debug-probes/j-link/models/j-link-lite/j-link-lite-arm/

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	1.4.0	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-1.4.0.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-2156X, ADuCM3029 EZ-Kit, ADuCM4050 EZ-Kit AND ADSP-BF7XX EZ-Kit.

ADSP-SC589/ADSP-SC584/ADSP-SC573

- Analog Devices CrossCore Embedded Studio version 2.8.3 or later
- For Users want to try LwIP, install LwIP for CCES: Lightweight TCP/IP Stack for CrossCore Embedded Studio Rev. 2.6.0 (<u>http://www.analog.com/en/design-center/processors-and-dsp /evaluation-and-development-software/adswp-lwip.html#dsp-relatedsoftware</u>)

ADuCM3029 EZ-Kit

- Analog Devices CrossCore Embedded Studio version 2.8.3 or later
- Device Family Pack (DFP) for CCES: ADuCM302x Software for Keil version 1.0.6 or later (http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boardskits/eval-aducm3029-ezkit.html#eb-relatedsoftware)

ADuCM4050 EZ-Kit

- Analog Devices CrossCore Embedded Studio version 2.8.3 or later
- Device Family Pack (DFP) for CCES: ADuCM4x50 Device Family Pack version 3.1.0 or later (<u>http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/ADZS-U4050LF-EZKIT.html#eb-relatedsoftware</u>)

ADSP-BF7XX EZ-Kit

• Analog Devices CrossCore Embedded Studio version 2.8.3 or later No other software needed to run the examples in CCES on ADSP-BF707 board.

ADSP-2156X EZ-Kit

• Analog Devices CrossCore Embedded Studio version 2.9.1 or later No other software needed to run the examples in CCES on ADSP-2156X board.

2.4 Software environment set up for IAR Embedded Workbench

This part shows the software environment setup for IAR Embedded Workbench on two kinds of boards: ADuCM3029 EZ-Kit and ADuCM4050 EZ-Kit. For ADSP-SC589/SC584/SC573 EZ-Kit boards, users do not need to install any other pack files except the software IAR Embedded Workbench version 7.60 or later.

ADuCM3029 EZ-Kit

- IAR Embedded Workbench version 7.60 or later
- Board Support Package (BSP) for IAR: ADuCM302x EZ-Kit Lite BSP for IAR version 1.0.6 (http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boardskits/eval-aducm3029-ezkit.html#eb-relatedsoftware)
- Edit an environment variable for your account (**ADI_CM302x_BSP_PATH**: This should point to the ADuCM302x_EZ_Kit/ directory of your ADuCM302x BSP installation)

ADuCM4050 EZ-Kit

- IAR Embedded Workbench version 7.60 or later
- ADuCM4050 EZ-Kit: ADuCM4x50 BSP for IAR version 1.1.0 (http://www.analog.com/en /design-center/evaluation-hardware-and-software/evaluation-boards-kits/ADZS-U4050LF-EZKIT.html#eb-relatedsoftware)
- Edit an environment variable for your account (**ADI_CM4x50_BSP_PATH**: This should point to the ADuCM4x50_EZ_Kit/ directory of your ADuCM4x50 BSP installation)

2.5 Software environment set up for Keil MDK

This part shows the software environment setup for Keil MDK on two kinds of boards: ADuCM3029 EZ-Kit and ADuCM4050 EZ-Kit. For ADSP-SC589/SC584/SC573 EZ-Kit boards, users do not need to install any other pack files except the software Keil MDK for ARM processors version 5.21a or later.

ADuCM3029 EZ-Kit

- Keil MDK for ARM processors version 5.21a or later
- Device Family Pack (DFP) for Keil: ADuCM3029 EZ-Kit: ADuCM302x Software for Keil version 1.0.6 (<u>http://www.analog.com/en/design-center/evaluation-hardware-and-software</u> /evaluation-boards-kits/eval-aducm3029-ezkit.html#eb-relatedsoftware)
- Board Support Package (BSP) for Keil: ADuCM302x EZ-Kit Lite BSP for IAR version 1.0.6 (http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boardskits/eval-aducm3029-ezkit.html#eb-documentation)
- Edit an environment variable for your account (ADI_CM302x_BSP_PATH: This should point to the ADuCM302x_EZ_Kit/ directory of your ADuCM302x BSP installation)

ADuCM4050 EZ-Kit

- Keil MDK for ARM processors version 5.21a or later
- Device Family Pack (DFP) for Keil: ADuCM4x50 Device Family Pack version 3.1.0 (_ http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boardskits/ADZS-U4050LF-EZKIT.html#eb-relatedsoftware_)

- Board Support Package (BSP) for Keil: ADuCM4x50 EZ-KIT Board Support Pack version 3.1.0 (<u>http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/ADZS-U4050LF-EZKIT.html#eb-relatedsoftware</u>)
- Edit an environment variable for your account (ADI_CM4x50_BSP_PATH: This should point to the ADuCM4x50_EZ_Kit/ directory of your ADuCM4x50 BSP installation)

3 Running the Examples on the ADuCM302x EZ-Kit

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

	Processor	Toolchain	Example(s)		
	ADuCM3029	IAR Embedded Workbench	Basic Demo		
	ADuCM3029	Keil MDK	Basic Demo		
	ADuCM3029	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 ADuCM302x EZ-Kit with CrossCore Embedded Studio

3.1.1 Overview

This page describes the steps required to build and run basic example on ADuCM3029 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 inormation please refer to Get the source code ready

Hardware Setup

- An ADuCM302x EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P5** 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



3.1.3 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 from the menu and 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 FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX_M3_ADuCM302x_CCES folder
- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the **projects** pane and click **Finish** to close the file browser dialog

🔀 Import	
Import Projects Select a directory to search for existing Eclipse projects.	
 Select root directory: C:\Analog Devices\freertos\FreeRTOSv10.0.0 ▼ Select archive file: ▼ Projects: 	Browse Browse
	Select All Deselect All Refresh
Options Options Search for nested projects Copy projects into workspace Hide projects that already exist in the workspace	
Working sets	New Select
Rext > 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** project and select the **Build Project** option from the menu

3.1.4 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** 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									
Create, manage, and run configurations									
Specify and launch an application with GDB and	OpenOCD	200							
Image: The second s	Name: RTOSDemo_CCES Debug Target Main Debugger Startup Debugger Debugger Startup Debugger	WSE							
Filter matched 5 of 8 items	Revert App	ly							
0	Debug	se							

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

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

 Eile
 Edit
 Source
 Refactor
 Navigate
 Search
 Project
 Run
 Window
 Help

 \square \blacksquare \square \square \square \blacksquare \square \blacksquare \square \blacksquare \square \blacksquare \square \blacksquare \blacksquare \square \blacksquare \blacksquare

3.1.5 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 are blinking. **The test is ok for <num> round** (s) will be printed constantly and **Test Passed** which means the test is successful will be printed after 25 rounds in the console.

🗳 Cor	nsole	ß	ø	Tasks	s) 🔝	Problems	Executa	bles	🗟 Debugg	er Console	🔗 Search	
RTOSE	emo_		S D	ebug	[Ap	plication w	ith GDB and	d Op	enOCD (Em	ulator)] RTC	SDemo_C	CES
The t	test	is	ok	for	17	round(s)					
The t	test	is	ok	for	18	round(s)					
The t	test	is	ok	for	19	round(s)					
The t	test	is	ok	for	20	round(s)					
The t	test	is	ok	for	21	round(s)					
The t	test	is	ok	for	22	round(s)					
The t	est	is	ok	for	23	round(s)					
The t	test	is	ok	for	24	round(s)					
The t	test	is	ok	for	25	round(s)					
Test	pass	ed										

3.2 Running the Basic Example for ADuCM302x EZ-Kit with IAR Embedded Workbench

3.2.1 Overview

This page describes the steps required to build and run basic example on ADuCM3029 EZ-KIT board using IAR Embedded Workbench.

3.2.2 Environment Setup

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

Software Requirement

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

Hardware Setup

- An ADuCM302x EZ-Kit board
- A J-Link Lite

Connect **DEBUG P4** port of the EZ-KIT board to a PC running IAR Embedded Workbench using the J-Link 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 IAR Embedded Workbench, follow below three steps to import and build it.

1. Select the Add Existing Project from the Project Menu and browse to the FreeRTOSv10.0.0 \FreeRTOS\Demo\CORTEX_M3_ADuCM302x_IAR\iar folder within the FreeRTOS product

directory and import the project

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



3. Build the project in the IAR workbench

• From the **Project** menu select the **Make** option and enter the file name



3.2.4 Run the Example

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

- 1. From the Project menu select the Download and Debug option
- 2. The application should load and halt at main. Continue the application to see it run.

3.2.5 Test Results

Output from the application should be visible within the **Terminal I/O** window which can be found in View menu in the IAR Embedded Workbench IDE. You should see three LEDs on the EZ-Kit are blinking. **The test is ok for <num> round(s)** will be printed constantly and **Test Passed** which means the test is successful will be printed after 25 rounds in the console.

Terminal I/O	×
Output:	Log file: Off
The test is ok for 13 round(s) The test is ok for 14 round(s) The test is ok for 15 round(s) The test is ok for 16 round(s) The test is ok for 17 round(s) The test is ok for 18 round(s) The test is ok for 19 round(s) The test is ok for 20 round(s) The test is ok for 21 round(s) The test is ok for 22 round(s) The test is ok for 23 round(s) The test is ok for 24 round(s) The test is ok for 25 round(s) The test is ok for 25 round(s) The test passed	•
<	۱.

3.3 Running the Basic Example for ADuCM302x EZ-Kit with Keil MDK

3.3.1 Overview

This page describes the steps required to build and run basic example on ADuCM3029 EZ-KIT board using Keil MDK.

3.3.2 Environment Setup

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

Software Requirement

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

Hardware Setup

- An ADuCM302x EZ-Kit board
- A J-Link Lite

Connect **DEBUG P4** port of the EZ-Kit to the host PC using the J-Link connector, connect the **USB** to **UART** port on the EZ-Kit to the host PC using the USB cable provided and simultaneously connect the power supply with 5 volts as in the diagram below



3.3.3 Build the Example

Before you run the FreeRTOS example in Keil MDK, follow below four steps to import and build it.

1. Import the FreeRTOS example into Keil MDK

- Select the **Open Project** option from the **Project** menu
- In the file tree window browse to the FreeRTOSv10.0.0
 \FreeRTOS\Demo\CORTEX_M3_ADuCM302x_KEIL folder in the FreeRTOS product installation and select the RTOSDemo.uvprojx file.
- Click the **Open** button to import the project

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



- 3. Build the project in Keil MDK
 - Select the Build Target/Rebuild All Target Files option from the Project menu

4. Configure a serial console application of your choice to view the output from the **UART to USB connection** on the HOST PC

• The Keil MDK is unable to output and text to the console within the MDK IDE. The output of the application is transmitted via UART to the serial console. 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 57600



3.3.4 Run the Example

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

1. From the **Flash** menu select the **Download** sub-menu and then choose the **Start/Stop Debug Session** option from **Debug** menu.

2. Click the **Run** button to start running the application



3.3.5 Test Results

When the application runs it will blink three LEDs on the EZ-Kit, **The test is ok for <num> round(s)** will be printed constantly and **Test Passed** which means the test is successful will be printed after 25 rounds on the console.

The	test	is	ok	for	6 round(s)
The	test	is	ok	for	7 round(s)
The	test	is	ok	for	8 round(s)
The	test	is	ok	for	9 round(s)
The	test	is	ok	for	10 round(s)
The	test	is	ok	for	11 round(s)
The	test	is	ok	for	12 round(s)
The	test	is	ok	for	13 round(s)
The	test	is	ok	for	14 round(s)
The	test	is	ok	for	15 round(s)
The	test	is	ok	for	16 round(s)
The	test	is	ok	for	17 round(s)
The	test	is	ok	for	18 round(s)
The	test	is	ok	for	19 round(s)
The	test	is	ok	for	20 round(s)
The	test	is	ok	for	21 round(s)
The	test	is	ok	for	22 round(s)
The	test	is	ok	for	23 round(s)
The	test	is	ok	for	24 round(s)
The	test	is	ok	for	25 round(s)
Test	t pass	sed			

4 Running the Examples on the ADuCM4x50 EZ-Kit

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

Processor		Toolchain	Example(s)		
	ADuCM4050	CrossCore Embedded Studio	Basic Demo		
	ADuCM4050	IAR Embedded Workbench	Basic Demo		
	ADuCM4050	Keil MDK	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 ADuCM4x50 EZ-Kit with CrossCore Embedded Studio

4.1.1 Overview

This page describes the steps required to build and run basic example on ADuCM4x50 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 ADuCM4x50 EZ-Kit board
- An ICE1000 or ICE2000 emulator

Connect the ICE1000 or ICE2000 emulator to **DEBUG P5** 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



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. Select the **File** menu and then select the **Import** option from the menu and 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 FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX_M4_ADuCM4x50_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** to Import the project

🔀 Import		
Import Projects		
Select a directory to sea	rch for existing Eclipse projects.	
Select root directory:	C:\Analog Devices\test_case\freeRTOSv10.	Browse
Select archive file:	·	Browse
Projects:		
RTOSDemo_CCES	(C:\Analog Devices\test_case\freertos\FreeRTOSv10.0.0	Select All
		Deselect All
		Refresh
•	4	
Options		
Search for nested pro	jects	
Copy projects into we	orkspace	
Working sets		
	·	New
		New
Working sets:	•	Select
	< Back Next > Finish	Cancel

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

File Edit Source Refactor Navig	gate Search Project Run Window Help
📬 🖬 🕞 😻 🕶 🍝 🔹	• 💁 • 🤌 • 🔛 🗉 🗊 🖢 • 🖓 • 🏷 •
ि Project Explorer 🛛	

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

4.1.4 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** 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 Specify and launch an application with GDB and	OpenOCD	Ť.
Application with CrossCore Debugger Application with CrossCore Debugger Application with GDB and OpenOCOL (Emu TOSDemo_CCES Debug Application with GDB and QEMU (Simulat Launch Group	Name: RTOSDemo_CCES Debug Target Main Startup Source Common Command: openocd Target (processor) Board Analog Devices ADuCM4050 Interface: Analog Devices ICE-1000 Emulator Max speed: I MHz Halt options Halt options Arguments:	Browse
Filter matched 5 of 8 items	Revert	Apply
0	Debug	Close

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

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

4.1.5 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. **The test is ok for <num>** round(s) will be printed constantly and **Test Passed** which means the test is successful will be printed after 25 rounds in the console.

```
🖳 Console 🛛 🖉 Tasks 🔝 Problems 💽 Executables 🗟 Debugger Console 🔗 Search
RTOSDemo_CCES Debug [Application with GDB and OpenOCD (Emulator)] RTOSDemo_CCES
The test is ok for 9 round(s)
The test is ok for 10 round(s)
The test is ok for 11 round(s)
The test is ok for 12 round(s)
The test is ok for 13 round(s)
The test is ok for 14 round(s)
The test is ok for 15 round(s)
The test is ok for 16 round(s)
The test is ok for 17 round(s)
The test is ok for 18 round(s)
The test is ok for 19 round(s)
The test is ok for 20 round(s)
The test is ok for 21 round(s)
The test is ok for 22 round(s)
The test is ok for 23 round(s)
The test is ok for 24 round(s)
The test is ok for 25 round(s)
Test passed
```

```
4.2 Running the Basic Example for ADuCM4x50 EZ-Kit with IAR Embedded Workbench
```

4.2.1 Overview

This page describes the steps required to build and run basic example on ADuCM4x50 EZ-KIT board using IAR Embedded Workbench.

4.2.2 Environment Setup

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

Software Requirement

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

Hardware Setup

- An ADuCM4x50 EZ-Kit board
- A J-Link Lite

Connect **DEBUG P4** port of the EZ-KIT board to a PC running IAR Embedded Workbench using the J-Link and simultaneously connect the power supply with 5 volts as in the diagram below



4.2.3 Build the Example

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

1. Select the **Add Existing Project** from the **Project** Menu and browse to the **FreeRTOSv10.0.0** **FreeRTOS\Demo\CORTEX_M4_ADuCM4x50_IAR\iar** folder within the FreeRTOS product directory and import the project

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



3. Build the project in the IAR workbench



• From the **Project** menu select the Make option and enter the file name

4.2.4 Run the Example

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

1. From the Project menu select the Download and Debug option

You may meet the Device Selection window, please select Coretex-M4 device.

2. The application should load and halt at main. Continue the application to see it run.

4.2.5 Test Results

Output from the application should be visible within the **Terminal I/O** window which can be found in View menu in the IAR Embedded Workbench IDE. You should see three LEDs on the EZ-Kit are blinking. **The test is ok for <num> round(s)** will be printed constantly and **Test Passed** which means the test is successful will be printed after 25 rounds in the console.

Terminal I/O	
Output:	Log file: Off
The test is ok for 13 round(s) The test is ok for 14 round(s) The test is ok for 15 round(s) The test is ok for 16 round(s) The test is ok for 17 round(s) The test is ok for 18 round(s) The test is ok for 19 round(s) The test is ok for 20 round(s) The test is ok for 21 round(s) The test is ok for 22 round(s) The test is ok for 23 round(s) The test is ok for 24 round(s) The test is ok for 25 round(s)	•
	P.

4.3 Running the Basic Example for ADuCM4x50 EZ-Kit with Keil MDK

4.3.1 Overview

This page describes the steps required to build and run basic example on ADuCM4x50 EZ-KIT board using Keil MDK.

4.3.2 Environment Setup

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

Software Requirement

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

Hardware Setup

- An ADuCM4x50 EZ-Kit board
- A J-Link Lite

Connect **DEBUG P4** port of the EZ-Kit to the host PC using the J-Link connector, connect the **USB** to **UART** port on the EZ-Kit to the host PC using the USB cable provided and simultaneously connect the power supply with 5 volts as in the diagram below



4.3.3 Build the Example

Before you run the FreeRTOS example in Keil MDK, follow below four steps to import and build it.

1. Import the FreeRTOS example into Keil MDK

- Select the **Open Project** option from the **Project** menu
- In the file tree window browse to the **FreeRTOSv10.0.0** **FreeRTOS\Demo\CORTEX_M4_ADuCM4x50_KEIL** folder in the **FreeRTOS product installation** and select the **RTOSDemo.uvprojx** file.
- Click the **Open** button to import the project
- 2. Choose Debug/Release mode to build the project.

	File	Edi	t V	/iew	Proj	ect	Flash	D	ebug	Per	riphe	erals
*****	ľ	2	H	Ø	*	Đ		5	6	-	⇒	P
*****	٨			6		LOAD	Rele	ease				-

3. Build the project in Keil MDK

• Select the Build Target/Rebuild All Target Files option from the Project menu

4. Configure a serial console application of your choice to view the output from the **UART to USB connection** on the HOST PC

• The Keil MDK is unable to output and text to the console within the MDK IDE. The output of the application is transmitted via UART to the serial console. 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 9600

🚔 Device Manager		RuTTY Configuration]	23
Pevice Manager File Action View Help Image: Proceeding of the second		Category: 	Basic options for your PuTTY session Specify the destination you want to connect to Serial line Speed CDM6 9600 Connection type: Raw Raw Telnet Rlogin Saved Sessions Load Default Settings Load Save Delete Close window on exit: Always Never Only on clean exit 	3
> 🚑 System devices	•	About	Open Cancel	

4.3.4 Run the Example

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

1. From the **Flash** menu select the **Download** sub-menu and then the **Start/Stop Debug Session** option from **Debug** menu.

2. Click the **Run** button to start running the application

File Edit Vie	w Project Flash	Debug	Peripherals	Tools SVCS
n 📴 🖬 🕻	1 * * *	9 6	~~ ~	RRR
🏦 🖪 📀	()	🔶 💽	10 🖬 🗐) 🗛 🗸 🚺
Registers		д 🖂 (Disassembly	
Register	Value		\$ <mark>0×000003</mark>	4E E7FE
Core			0x000003	50 E7FE
BO BO	0x000059C0		0x000003	52 E7FE
B1	0x20004008		0x000003	54 E7FE
B2	0x00002690		0x000003	56 E7FE
B3	0v000000000		0x000003	58 E7FE
П	0-000000000		0x000003	5A E7FE
DE	000000000		0x000003	SC 4804
na ina	0x00000000			
4.3.5 Test Results

When the application runs it will blink three LEDs on the EZ-Kit, **The test is ok for <num> round(s)** will be printed constantly and **Test Passed** which means the test is successful will be printed after 25 rounds on the console.

The	test	is	ok	for	6 rou	nd(s)			
The	test	is	ok	for	7 rou	nd(s)			
The	test	is	ok	for	8 rou	nd(s)			
The	test	is	ok	for	9 rou	nd(s)			
The	test	is	ok	for	10 ro	und(s)			
The	test	is	ok	for	11 ro	und(s)			
The	test	is	ok	for	12 ro	und(s)			
The	test	is	ok	for	13 ro	und(s)			
The	test	is	ok	for	14 ro	und(s)			
The	test	is	ok	for	15 ro	und(s)			
The	test	is	ok	for	16 ro	und(s)			
The	test	is	ok	for	17 ro	und(s)			
The	test	is	ok	for	18 ro	und(s)			
The	test	is	ok	for	19 ro	und(s)			
The	test	is	ok	for	20 ro	und(s)			
The	test	is	ok	for	21 ro	und(s)			
The	test	is	ok	for	22 ro	und(s)			
The	test	is	ok	for	23 ro	und(s)			
The	test	is	ok	for	24 ro	und(s)			
The	test	is	ok	for	25 ro	und(s)			
Test	t pass	sed							

5 Running the Examples on the ADSP-SC589 EZ-Kit

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

5.1 Running the Basic Example for ARM on ADSP-SC589 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-SC589 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 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:



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_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:		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	orkspace	
Hide projects that al	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

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 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

Z Debug Configurations				×
Create, manage, and run configurations				The
Select a debug session to launch and a program to load				2
□ @ × 8 » -	Name: RTOSDemo_CCES_Core0 Debug			
type filter text	🔊 Session 💊 Automatic Breakpoints 🔞 Target Options 👑 Custom Board Support 🔅 Multiproce	ssor Groups 💱 Source 🔳 Commo	n	
 Application with CrossCore Debugger RTOSDemo_CCES_Core0 Debug Application with GDB and OpenOCD (Emulator) Application with GDB and QEMU (Simulator) Launch Group 	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	Silicon revision	Add
	 Device 0 [Core 0] C:\Analog Devices\CrossCore Embedded Studio 2.7.0\SHARC\ldr\ezkitSC589_preload_core 	e Reset, Run after load	not available	Edit
	RTOSDemo_CCES_Core0\Debug\RTOSDemo_CCES_Core0	Check si-revision, Run after load	not available	Remove
	 Ø Device 0 [Core 1] Output: Context of the second sec			Remove All
	A @ Device 0 [Core 2]			Move Up
	Click here to select a program to load>			Move Down
				Restore Defaults
Filter matched 5 of 15 items			Revert	Apply
0			Debug	Close

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	👦 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) 🔹			
Application with GDB and QEMU (Simulator)	Breakpoints to set automatically after load:				
Taunch Group	Label	Description	<u>N</u> ew		
▶ Launch Group (Deprecated)	 ✓ ● _cxit ✓ ● _fatal ✓ ● main ✓ ●dbg ✓ ●stack ✓ ●fatal_ 	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		
Filter matched 6 of 6 items	Enable semil	hosting Reyert Debug	Apply		

- 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
😁 🕶 📾 📾 📾 💩 💩 📭 💷 🗷 🧟 . 🖉 🐘 🔍 🐄 🕶 🂁 🛩 🖉 🕶 🖉 🕶 🖓 🕶
☆ Debug ⊠ Resume (F5)
RTOSDemo_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)

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-SC589 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-SC589 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 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.



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
- 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**

Import Projects Select a directory to searc	ch for existing Eclipse projects.	
 Select root directory: Select archive file: Projects: 	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	jects orkspace eady exist in the workspace	
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

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

File Falls Courses Defender	🛛 🔀 Debug Configurations			23	
Project Explorer ≥	Create, manage, and run configuration:	s		T.	ug 🊵 CMSIS Pack Manager
	Ype filter text ✓ Application with CrossCore Debug ✓ Application with CrossCore Debug ✓ Application with CossCore Debug ✓ Application with GDB and OpenOr ✓ Application with GDB and QEMU (✓ Launch Group	Name: RTOSDemo_CCES_SHARC_Core0 Debug Session Automatic Breakpoint T Session Wizard Select Processor Choose a target processor.	arget Options) 👑 Custom Board Support 🦃 Multipro	cessor Group 🖶 Source 🎢	
		Processor family: Processor type:	SHARC -	on Add Edit Remove Remove All Move Up Move Down Restore Defaults	
۲. m	Filter matched 5 of 16 items	Show all processors Use selected project to create new session Help <back< td=""><td>Configurator Next > 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

File Edit Course Defector	Configurations			
	Create, manage, and run configuratio Select a debug session to launch and a pro	ons ogram to load	1 Contraction of the second se	ig 🌰 CMSIS Pack Manager
Image: Second	Create, manage, and run configuratio Select a debug session to launch and a pro Uppe filter text Application with CrossCore Debug Application with GDB and QEMU (Launch Group	orgam to load Name: RTOSDemo_CCES_SHARC_CoreD Debug Session Onfiguration Target: Emulation Debug Target Platform: ADSP-SC589 valid: 1000 Processor: ADSP-SC589 The following program(s) will be loaded: The following program(s) will be loaded: The following Device 0 [Core 0] C:VANalog Devices/CoresCore Reset, Run after load RTOSDemo_CCES_SHARC_Cc Check si-revision, Run after load RTOSDemo_CCES_SHARC_Cc Check si-revision, Run after load RTOSDemo_CCES_SHARC_Cc Check si-revision, Run after load Cick here to select a progra Cick here to select a progra	upport Multiprocessor Group & Source "1 Select Session Silicon revision Add Edit Remove 1.0 Move Up Move Down Restore Defaults	ıg ∰ CMSIS Pack Manager
	[©]		Debug Close	
< III	• •			
	2			

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



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.

File Edit Source Refactor Navigate Search Project Target Run Window Help					
: 🗂 ▾ 🔜 🕼 🐘 ! Ѯ₀ 🌢 🏟 🕪 💷 🔳 🌫 🕫 ⊥ℓ 🏶 🛍 🕪 🂫 ! ‡ ▾ 🂁 🖋 ▾ 🌆 ▾ ! 🌽 🖋 ▾ 🚺	• 🖘 🔶 • 🔿 •	Quick A	ccess 🗄	😤 🔚 C/C++ 🛛 🔆 Debug 🚳 CMSIS Pack Ma	anager
🎋 Debug 🛛 🦌 🗸 🖓 🖉 🗖	(×)= Variables 🤷 Breakpoints 🖄	🛛 👫 Registers 🛋 N	odules	💥 💥 🔐 🕾 🔍 🖽 🖻 🛸 💎	
 RTOSDerno, CCES_SHARC_Core1 Debug [Application with CrossCore Debugger] 	C (function: _exit) [type:]	Temporary]			
	4				
C RIOSDemo_CCES_SHARC_CoreU.c X			Duti	line U Memory I Disassembly 23	
<pre>adf_init(omponents(); /** * 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 a ind core 2 is to use the adj_core_enable function. */ adj_core_enable(ADI_CORE_SHARCO); /*adj_core_enable(ADI_CORE_SHARCI);*/ </pre>		H	⊽ No debu	er location here ♥ € ि ि 🥸 😣 ag context	
📮 Console 🕱 🧔 Tasks 🖹 Problems 🚺 Executables 🗟 Debugger Console 🖷 Progress 🍃 Call Hierarchy				🖹 📑 🖻 📑 🚍 🛨 📬 🛨	- 0)
Output					
Test passed Test passed Test passed Test passed Test passed Test passed Test passed Test passed					*
Test passed Test passed Test passed Test passed					E
					-
					•
	Writable	Smart Insert	U:/6	1	

5.3 Running the LwIP Example for ARM on ADSP-SC589 EZ-Kit with CrossCore Embedded Studio

5.3.1 Overview

This page describes the steps to build and run the LwIP Example for ARM on ADSP-SC589 EZ-Kit with CrossCore Embedded Studio.

5.3.2 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, connect the target board to the same LAN as PC using standard network 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:



5.3.3 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_ADSP_SC589_CCES_LwIP folder
- Click **Finish** to close the file browser dialog
- A single project should appear in the **Project Explorer**

🔀 Import	
Import Projects	
Select a directory to search for existing Eclipse projects.	
	-
Select root directory: C:\Analog Devices\freertos\FreeRTOSv10.0.0 ▼	Browse
Select archive file:	Browse
Projects:	
DnsClient_FreeRTOS_A5 (C:\Analog Devices\freertos\FreeRTOSv1(Select All
	Deselect All
	Refresh
۰ III ا	
Options	
Search for nested projects	
Copy projects into workspace	
Hide projects that aiready exist in the workspace	
Working sets	
Add project to working sets	New
Working sets:	Select
Over the second seco	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 **DnsClient_FreeRTOS_A5** project and select the **Build Project** option from the menu

5.3.4 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 stdio 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.



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 **DnsClient_FreeRTOS_A5** 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					23
Create, manage, and run configurations	(a	1
8 You must select a debug session first		Select Brosses			
Image: Second Secon	Name: D Session Target Platfo Proce: The fol Progr	Choose a target processor. Processor family: Processor type: ADSP-21584 ADSP-21587 ADSP-3C571 ADSP-3C571 ADSP-3C572 ADSP-3C572 ADSP-3C582 ADSP-3C582 ADSP-3C582 ADSP-3C584 ADSP-3C584 ADSP-3C584 ADSP-3C589 Show all processors Use selected project to create new session Help <back next=""> Fri</back>	Configurator	ups) to Source Common	Select Session Add Edit Remove Remove All Move Up Move Down Restore Defaults
← Ⅲ → Filter matched 5 of 11 items				Reve	rt Apply
0				Det	oug Close

3. Disable the semihosting function in Automatic Breakpoints

🔀 Debug Configurations			
Create, manage, and run configurations			
Specify and launch a CrossCore Embedded Studio program		-Q	
	Name: DnsClient FreeRTOS A5 Debug		
type filter text	🔭 Session 💁 Automatic Breakpoints 🛛 🚯 Target Options) 🎇 Custom Board Support) 🎯 Multiprocessor Groups) 🦉 Source) 🔲 Common		
Application with CrossCore Debugger	Processor:		
DnsClient_FreeRTOS_A5 Debug Application with GDB and OpenOCD (Emulator)	Device 0 [Core 0] (Cortex-A5)		
Application with GDB and QEMU (Simulator)	Breakpoints to set automatically after load:		
🚙 Launch Group	Label Description	<u>N</u> ew	
Launch Group (Deprecated)	I I I I I I I I I I I I I I I I I I I	Edit	
		Delete	
	☑ ● _dbg Used for assert	Delete All	
	■ ● _stack Stack overflow detected	Delece Alt	
	■ Tata Fatal exception occurred in KIL		
	Enable semihosting		
Filter matched 6 of 8 items	Reyert	Apply	
(Y)	<u>D</u> ebug	Close	

4. 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.
Image: Control of the second secon	Name: DnsClient_FreeRTOS_AS Debug Session Configuration Target: Emulation Debug Target Platform: ADSP-SC589 The following program(s) will be loaded: Program	t) Multiprocessor Groups S S Options Reset, Run after load Check si-revision, Run after load	ource) Common)	Select Session, Add Edit Remove Remove All Move Up Move Dowm Restore Defaults
Filter matched 5 of 11 items				Revert Apply Debug Close

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



5.3.5 Test Results

Output from the application should be visible within the TTY terminal (e.g. PuTTY/TeraTerm). You should see link is established, and **IP ADDRESS** assigned from DHCP server will be printed after you install the LwIP.

B COM3 - Putty	- • •
Clock Configuration	A
CCLK = 375000000, SCLK = 187500000, SCLK0 = 93750000, SCLK1 = 9375000	O, DCLK = 3
75000000, OCLK = 125000000, EMACO CLK = 125000000	
Configuring switches for the ethernet operation	
User need to set the MAC address in system.svc as MAC address is not :	stored on b
oard	
Incorrect MAC address in system.svc Using temporary MAC: 0x0012345678	91
Waiting for the link to be established	
Link established	
IP ADDRESS: 10.99.24.117	
IP Address of analog.com is : 137.71.25.128	
	~

6 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	ARM A5	CrossCore Embedded Studio	Lwip 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

6.1 Running the Basic Example for ARM on ADSP-SC584 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-SC584 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 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:



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_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	ojects	
Copy projects into w	orkspace	
Working sets	ready exist in the workspace	
Add project to work	ing sets	New
Add project to work		INEW
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

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 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

Z 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)		
Application with GDB and QEMU (Simulator)	Breakpoints to	set automatically after load:	
Taunch Group	Label	Description	<u>N</u> ew
 Launch Group (Deprecated) 	V exit	End of program	Edit
	III III ●fatal_ III III ● main	Fatal error occurred in RTL Start of program	Delete
		Used for assert	Delete All
	Stack	Stack overflow detected	Delete All
	Tatal_	Fatal exception occurred in RTL	
	Enable semi	hosting	·
	L		
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



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-SC584 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-SC584 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 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.



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
- 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		
Import Projects Select a directory to searc	ch for existing Eclipse projects.	
 Select root directory: Select archive file: Projects: 	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	jects orkspace eady exist in the workspace	
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

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_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

📑 • 🔛 🕼 🛞 • 🗞 • 📾 🚸 • 💁 🖉	Z Debug Configurations		Ĩ
Image: The second s	✓ Debug Configurations Create, manage, and run configurations ♥ You must select a debug session first ♥ The set of the	✓ Session Witard	r Groups 15/ Source Common
		(a) AD3>-SC/3 (b) AD3>-SC/3 (a) AD3>-SC/33 (b) AD3>-SC/33 (a) AD3>-SC/33 (b) AD3>-SC/33 (a) AD3>-SC/33 (b) AD3>-SC/33 (a) AD3>-SC/33 (b) AD3>-SC/33 (c) AD3>-SC/33 (c) AD3>-SC/33 (c) AD3>-SC/33	Options Silicon revision Add Edt Remove Move Up Move Down Restore Default Save picture ss
	Filter matched 5 of 10 items		Revert Apply Debug Close
		la and an all and a second and a	n ^ (F)

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 CBB 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-SC584 via ICE-1000 Processor: ADSP-SC584			Select Session
	The following program(s) will be loaded:			
	Program	Options	Silicon revision	Add
		\SHARC\\dr\ezkitSC584_F Reset, Run after load _CCES_SHARC_Core0 Check si-revision, Run after load	not available not available	Edit
		_CCES_SHARC_Core1.dxe Reset, Check si-revision, Run after load	1.0	Remove All Move Up
				Move Down 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



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.

```
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\freeRTOSV10.0.0\FreeRTOS\Debug\RTOSDemo\SHARC_ADSP_SC584_CCES\RTOSDemo_CCES_SHARC_Core1\Debug\RTOSDemo_CCES_SHARC_Core1.dxe"

Loading application: "C:\Analog Devices\freeRTOSV10.0.0\FreeRTOS\Debug\RTOSDemo\SHARC_ADSP_SC584_CCES\RTOSDemo_CCES_SHARC_COre1\Debug\RTOSDEmo_CCES_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHARC_SHA
```

6.3 Running the LwIP Example for ARM on ADSP-SC584 EZ-Kit with CrossCore Embedded Studio

6.3.1 Overview

This page describes the steps to build and run the LwIP Example for ARM on ADSP-SC584 EZ-Kit with CrossCore Embedded Studio.

6.3.2 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, connect the target board to the same LAN as PC using standard network cable and simultaneously connect the power supply with 12 volts as in the diagram below.

If connect SC584 to the same LAN as PC with 1000M switch, the SC584 EZ-Kit board in **silicon 0.1 can't work with LwIP Example**.



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_ADSP_SC584_CCES_LwIP 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 C:\Analog Devices\freertos\FreeRTOS\10.0.0.	Browere
Select archive file:	Browsen
	DIOWSE
DnsClient FreeRTOS A5 (C:\Analog Devices\freertos\FreeRTOSv1(Select All
	Refresh
	Keresii
4	
Options	
Search for nested projects	
Copy projects into workspace Hide projects that already exist in the workspace	
Working sets	
Add project to working sets	New
Working sets:	Select
Rext > 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 **DnsClient_FreeRTOS_A5** 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 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 **DnsClient_FreeRTOS_A5** 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

24 Debug Configurations		
Create, manage, and run configurations vou must select a debug session first	Z Session Wizard	1
	Select Processor	7
	Na Choose a target processor.	
type filter text ■ Application with CrossCore Debugger ■ DnsClent_FreeRTOS_AS Debug ■ Application with GDB and OpenOCD (Emulator) ■ Application with GDB and QEMU (Simulator) ■ Launch Group	Processor family: SHARC Processor type: ADSP-21584 ADSP-21587 ADSP-2570 ADSP-SC570 ADSP-SC571 ADSP-SC571 ADSP-SC572 ADSP-SC573 ADSP-SC584 ADSP-SC584 ADSP-SC584 ADSP-SC587 ADSP-SC587 ADSP-SC587 Show all processors V Use selected project to create new session Configurator	or Groups) & Source Common Select Session Options Silicon revision Add Edit Remove Remove All Move Up Move Down Restore Defaults Restore Defaults
Filter matched 5 of 11 items	Performance Image: Provide the sector of the sec	Revert Apply
?		Debug

3. Disable the semihosting function in Automatic Breakpoints
Z Debug Configurations			X
Create, manage, and run configurations			1
Specify and launch a CrossCore Embedded Studio program			
	Name: DnsClient	FreeRTOS AS Debug]
type filter text	Session 💁 A	utomatic Breaknoint 🔪 🕼 Target Options) 🎆 Custom Board Support) 🍘 Multiprocessor Groups) 📴 Source) 🥅 Common	
Application with CrossCore Debugger	Processor:		
DnsClient_FreeRTOS_A5 Debug Application with GDB and OpenOCD (Emulator)	Device 0 [Core 0]	(Cortex-A5) 🗸]
Application with GDB and QEMU (Simulator)	Breakpoints to se	t automatically after load:	
Launch Group	Label	Description	<u>N</u> ew
·	I I I I I I I I I I I I I I I I I I I	End of program Estal error occurred in RTI	<u>E</u> dit
	Main Main	Start of program	Delete
	₩ • _dbg	Used for assert	Delete <u>A</u> ll
	Stack	Stack overnow detected Fatal exception occurred in RTL	
	Enable semiho	sting	
Filter matched 6 of 8 items		Reyert	Apply
?		Debug	Close

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

🔀 Debug Configurations				×
Create, manage, and run configurations Select a debug session to launch and a program to load				TO-
	Name: DnsClient_FreeRTOS_A5 Debug			
type filter text	🐡 Session 🔹 Automatic Breakpoints 🕞 Target Options 👯 Custo	m Board Support) 🎲 Multiprocessor Groups) 🦆 Source	e 🔲 Common	
Application with CrossCore Debugger DrsClient_FreeRTOS_AS Debug Application with GDB and OpenOCD (Emulator) Application with GDB and QEMU (Simulator) Launch Group	Session configuration Target: Emulation Debug Target Platform: ADSP-SC584 via ICE-1000 Processor: ADSP-SC584			Select Session
	The following program(s) will be loaded:			
	Program	Options	Silicon revision	Add
		C\ldr∖ezkitSCS84_prek Reset, Run after load Check si-revision, Run after load	not available not available	Edit Remove Remove All Move Up Move Down
		m		Restore Defaults
Filter matched 5 of 11 items			Rever	t Apply
?			Deb	ug Close

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

File	Edit	Source	Refacto	or N	avigate	Search	Project	Target	Run	Window	/ He	lp				
1	- 8		1010	\$ 6	ð 📭 (10 🔲 🛛	8. Q. I	2 \$ (0 🕪	*	- 9	• • 2	1	• <i>1</i>	<u>k</u> -	- 72
*	Debug	x												×		
4	🔁 Dn	sClient_Fr	eeRTOS_	A5 De	bug [Ap	plication	with Cros	sCore De	bugge	r]						
	⊿ 🔐	ADSP-SC	584 via I	CE-10	00											
	4	n Devic	e 0 [Core	e 0] (0	Cortex-A	5) [Debu	g\DnsClie	nt_FreeR1	ros_as	5] (Suspen	ded : l	Breakpoi	nt)			
		🔳 m	ain() at a	app_m	ain.c:79	0x894031	6e									
		Devic 🧶	e 0 [Core	e1](S	HARC)	(Running	: User Re	quest)								
		🔎 Devic	e 0 [Core	e 2] (S	HARC)	(Running	: User Re	quest)								

6.3.5 Test Results

Output from the application should be visible within the TTY terminal (e.g. PuTTY/TeraTerm). You should see link is established, and **IP ADDRESS** assigned from DHCP server will be printed after you install the LwIP.



7 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	ARM A5	CrossCore Embedded Studio	Lwip 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

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

7.1.1 Overview

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

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 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:



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. 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		
Search for nested pro	ojects	
Copy projects into w	orkspace	
Hide projects that all	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

7.1.4 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

Configurations				×
Create, manage, and run configurations Select a debug session to launch and a program to lo	oad			- A
Image: Second Secon	Name: RTOSDemo_CCE5_Core0 Debug Session Automatic Breakpoints Target Option Target: Emulation Debug Target Platform: ADSP-SC573 via ICE-1000 Processor: ADSP-SC573 The following program(s) will be loaded: Program C:\Analog Devices\CrossCore Embedded Studie RTOSDemo_CCES_Core0\Debug\RTOSDemo_CC Device 0 [Core 1] < Click here to select a program to load> Click here to select a program to load> 	Custom Board Support Multiprocessor Gr Options 2.7.0\SHARC\\dr\ Reset, Run after load CES_Core0 Check si-revision, Run after load	Silicon revision not available	on Select Session Edit Remove Remove All Move Up Move Down Restore Defaults
< ► Filter matched 5 of 9 items			Reve	ert Apply
?			De	bug Close

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.
Image: Second secon	Name: RTOSDer	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 ()] (Cortex-A5) 🗸 🗸	
Application with GDB and OpenOCD (Emulator) Application with GDB and OEMU (Simulator)	Breakpoints to s	et automatically after load:	
🔋 Launch Group	Label	Description	<u>N</u> ew
■ Launch Group Launch Group (Deprecated)	 ✓ •_exit ✓ •_fatal_ ✓ • main ✓ •_dbg ✓ •_stack. ✓ •_stack. ✓ •_fatal 	End of program - Fatal error occurred in RTL Start of program Used for assert - Stack overflow detected - Fatal exception occurred in RTL	Edit
	Enable semih	osting	
<		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



7.1.5 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.

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

7.2.1 Overview

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

7.2.2 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.



7.2.3 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.	
 Select root directory: Select archive file: Projects: 	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 alree	jects prkspace eady exist in the workspace	
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

7.2.4 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 family: Processor type: ADSP-21584 ADSP-21587 ADSP-2570 ADSP-5C571 ADSP-5C572 ADSP-5C573 ADSP-5C573 ADSP-5C582 ADSP-5C583 ADSP-5C584 ADSP-5C589 Show all processors V Use selected project to create new session	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				Ť.
Image: Second Secon	Name: RTOSDemo_CCES_SHARC_Core0 Debug Session @ Automatic Breakpoints 1 Target Options 1 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	Multiprocessor Groups is Source in Options Reset, Run after load Check si-revision, Run after load Reset, Check si-revision, Run after load	Common Silicon revision not available 0.0	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



7.2.5 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 
Co
```

7.3 Running the LwIP Example for ARM on ADSP-SC573 EZ-Kit with CrossCore Embedded Studio

7.3.1 Overview

This page describes the steps to build and run the LwIP Example for ARM on ADSP-SC573 EZ-Kit with CrossCore Embedded Studio.

7.3.2 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, connect the target board to the same LAN as PC using standard network 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:



7.3.3 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
- 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 FreeRTOSv10.0.0
 \FreeRTOS\Demo\CORTEX_A5_ADSP_SC573_CCES_LwIP folder
- Click **Finish** to close the file browser dialog
- A single project should appear in the **Project Explorer**

🔀 Import		
Import Projects Select a directory to sear	ch for existing Eclipse projects.	
 Select root directory: Select archive file: Projects: 	C:\Analog Devices\freertos\FreeRTOSv10.0.0 -	Browse Browse
☑ DnsClient_FreeRT	OS_A5 (C:\Analog Devices\freertos\FreeRTOSv1(Select All Deselect All Refresh
Options Options Search for nested pro Copy projects into w Hide projects that alr Working sets Add project to work Working sets:	jects orkspace eady exist in the workspace ing sets	New Select
?	< Back Next > Finish	Cancel

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

Edit Source	Ret	actor	Navig	ate S	Search	E
	• %	- 00		*	- 9	-
roject Explore		1 De	bug		₽	
⁵ RTOSDemo	< <u> </u>	2 Re	lease		J	
🖗 Binaries						
Includes						
🖻 🗁 Debug						
EreeRTOS	5					
🗁 Include						
😕 Release						
Bource						
😕 Standard	Dem	0				
Curtan.						

- 3. Build the project in CrossCore Embedded Studio:
 - In the **Project Explorer** right click on the **DnsClient_FreeRTOS_A5** project and select the **Build Project** option from the menu

7.3.4 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 stdio 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.



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 **DnsClient_FreeRTOS_A5** 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				23
Debug Configurations Create, manage, and run configurations You must select a debug session first You must select a debug session fir	Name: Sessi Targ	X Session Wizard Select Processor Choose a target processor. Processor family: Processor type:	roups) 🖅 Source) 🛄 Common	2n
 Application with GDB and QEMU (Simulator) Launch Group 	Pro	ADSP-21587 ADSP-5C570 ADSP-5C571 ADSP-5C572 ADSP-5C582 ADSP-5C582 ADSP-5C583 ADSP-5C584 ADSP-5C584 ADSP-5C584 ADSP-5C585 Use selected project to create new session Configurator P Help < Back Next > Finish Cancel	Silicon revision Add Edit Remove Move Up Move Dow Restore Defau	
III Filter matched 5 of 11 items			Revert	ly
•			Debug	ose

3. Disable the semihosting function in Automatic Breakpoints

C Debug Configurations		
Create, manage, and run configurations Specify and launch a CrossCore Embedded Studio program		Ť
C 🗎 🗶 🖻 🍄 🗸	Name: DnsClient_FreeRTOS_A5 Debug	
type filter text	🕽 Session 💁 Automatic Breakpoints 🕠 Target Options) 🎳 Custom Board Support 🚳 Multiprocessor Groups) 🥪 Source) 🗔 Common	
Application with CrossCore Debugger	Processor:	
DnsClient_FreeRTOS_A5 Debug Application with GDB and OpenOCD (Emulator)	Device 0 [Core 0] (Cortex-A5)	-
Application with GDB and QEMU (Simulator)	Breakpoints to set automatically after load:	
📮 Launch Group	Label Description	<u>N</u> ew
Eaunch Group (Deprecated)	P ← exit End of program	Edit
	III W ●_fstal Fstal error occurred in KTL	Delete
	I I I I I I I I I I I I I I I I I I I	
	🗾 💆 🖕 _stack Stack overflow detected	Delete All
	☑ ● _fatal Fatal exception occurred in RTL	
	Enable comboting	
Filter matched 6 of 8 items	Revert	Apply
?	Debug	Close

4. 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 loa	d			
Image: Second Secon	Name: DnsClient_FreeRTOS_A5 Debug	ÿ Custom Board Support] 👙 Multiprocessor Groups	5 Source 🗍 Common	Select Session
Application with GDB and QEMU (Simulator) Launch Group	Platform: ADSP-SC573 via ICE-1000 Processor: ADSP-SC573			
	The following program(s) will be loaded:			
	Program	Options :0\SHARC\Idr\ezkit Reset, Run after Ioad S_AS Check si-revision, Run after Ioad	Silicon revision not available not available	Add Edit Remove Remove All Move Up Move Down Restore Defaults
Filter matched 5 of 11 items			Re	vert Apply Debug Close

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



7.3.5 Test Results

Output from the application should be visible within the TTY terminal (e.g. PuTTY/TeraTerm). You should see link is established, and **IP ADDRESS** assigned from DHCP server will be printed after you install the LwIP.

Clock Configuration CCLK = 375000000, SCLK = 187500000, SCLK0 = 93750000, SCLK1 = 93750000, DCLK = 3 75000000, OCLK = 125000000, EMACO_CLK = 125000000 Configuring switches for the ethernet operation User need to set the MAC address in system.svc as MAC address is not stored on b oard Incorrect MAC address in system.svc Using temporary MAC: 0x00123456789A Waiting for the link to be established Link established IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	B COM3 - PuTTY
CCLK = 375000000, SCLK = 187500000, SCLK0 = 93750000, SCLK1 = 93750000, DCLK = 3 75000000, OCLK = 125000000, EMACO_CLK = 125000000 Configuring switches for the ethernet operation User need to set the MAC address in system.svc as MAC address is not stored on b oard Incorrect MAC address in system.svc Using temporary MAC: 0x00123456789A Waiting for the link to be established Link established IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	Clock Configuration
75000000, OCLK = 125000000, EMACO_CLK = 125000000 Configuring switches for the ethernet operation User need to set the MAC address in system.svc as MAC address is not stored on b oard Incorrect MAC address in system.svc Using temporary MAC: 0x00123456789A Waiting for the link to be established Link established IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	CCLK = 375000000, SCLK = 187500000, SCLK0 = 93750000, SCLK1 = 93750000, DCLK = 3
Configuring switches for the ethernet operation User need to set the MAC address in system.svc as MAC address is not stored on b oard Incorrect MAC address in system.svc Using temporary MAC: 0x00123456789A Waiting for the link to be established Link established IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	75000000, OCLK = 125000000, EMACO CLK = 125000000
User need to set the MAC address in system.svc as MAC address is not stored on b oard Incorrect MAC address in system.svc Using temporary MAC: 0x00123456789A Waiting for the link to be established Link established IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	Configuring switches for the ethernet operation
oard Incorrect MAC address in system.svc Using temporary MAC: 0x00123456789A Waiting for the link to be established Link established IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	User need to set the MAC address in system.svc as MAC address is not stored on b
Incorrect MAC address in system.svc Using temporary MAC: 0x00123456789A Waiting for the link to be established Link established IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	oard
Waiting for the link to be established Link established IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	Incorrect MAC address in system.svc Using temporary MAC: 0x00123456789A
Link established IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	Waiting for the link to be established
IP ADDRESS: 10.99.24.117 IP Address of analog.com is : 137.71.25.128	Link established
IP Address of analog.com is : 137.71.25.128	IP ADDRESS: 10.99.24.117
	IP Address of analog.com is : 137.71.25.128
	· · · · · · · · · · · · · · · · · · ·

8 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

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

8.1.1 Overview

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

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-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



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.\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.	
 Select root directory: reeRTOS\Demo\Blackfin_ADSP_BF707_CCE\$ ▼ Select archive file: ▼ 	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 Refacto	r Navigate Sea	irch Project Run	Window Help
**	≧ † • •) / - R 🗉 1] 월 ▼ 월 ▼ ♥ � ▼ ↔ ▼
ြဲ 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

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_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 type: ADSP-BF607 ADSP-BF608 ADSP-BF700 ADSP-BF701 ADSP-BF701 ADSP-BF702 ADSP-BF703 ADSP-BF704 ADSP-BF705 ADSP-BF707 Show all processors Use selected project to create new session Configur Help < Back Next > Finish Cance Configur	Tator	: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

C Debug Configurations				
Create, manage, and run configurations Select a debug session to launch and a program to load				Ť.
Application with CrossCore Debugger Application with CrossCore Debugger Application with GDB and OpenOCD (Emulator) Application with GDB and QEMU (Simulator) Launch Group	Name: RTOSDemo_BF707 Debug	tom Board Support] 🏐 Multiprocessor Groups] 🦆 Source 🛛	Common	Select Session
	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



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 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
```

9 Running the Examples on the ADSP-21569 EZ-Kit

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

Processor	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

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

9.1.1 Overview

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

9.1.2 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



9.1.3 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**

27 Import	- • ×
Import Projects	
Select a directory to search for existing Eclipse projects.	
● Select root directory: cces.select.cc	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	
Search for nested projects	
Copy projects into workspace	
Working sets	
Add project to working sets	New
Working sets	Select
	<u>Sciectiii</u>
? Image: Second seco	Cancel

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

File Edit Source Refactor Navigate	Search Project Run Window Help
📬 = 😨 🗞 + 🔦 - 💁	▼ 🤔 🖋 ▼ 📴 🗐 👔 🖢 ▼ 🖓 ▼ 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

9.1.4 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 📃 📼 💌	
You must select a debug session first	Select Processor Choose a target processor.	- Ver
Image: Image	Processor family: SHARC	JICE "1
 RTOSDemo_CCES_SHARC_21569 Application with GDB and OpenOC Application with GDB and OEMU (5) 	@ ADSP-21569 @ ADSP-21571	t Session
Application with GDB and QLWO (5 Launch Group Launch Group (Deprecated)	 ADSP-21573 ADSP-21583 ADSP-21584 	=
		<u>A</u> dd <u>E</u> dit
	Show all processors ✓ Use selected project to create new session	
Filter matched 6 of 8 items	Configurator	Apply
0	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		1 Average State St	
	Name: RTOSDemo_CCES_SHARC_21569 Debug			
type filter text	Session 💊 Automatic Breakpoints 🚇 Target Options 👑 Custom Board Support 🕸 N	Iultiprocessor Groups	^t ∕ Source □ <u>C</u> ommon	
 Application with CrossCore Debugg RTOSDemo_CCES_SHARC_21569 Application with GDB and OpenOCE Application with GDB and QEMU (Sii Launch Group 	Session configuration Target: Emulation Debug Target Platform: ADSP-21569 via ICE-1000 Processor: ADSP-21569		Select Session	
Launch Group (Deprecated)	The following program(s) will be loaded:			
	Program Options	Silicon revision	<u>A</u> dd	
	 C:\Analog Devices\CrossCore Embedded Studio Reset, Run after load RTOSDemo_CCES_SHARC_21569\Debug\RTOSE Check si-revision, Run after load 	any any	<u></u> dit ≡	
			Re <u>m</u> ove	
			Remove All	
			Move Down	
			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
```

9.1.5 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
```

10 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

11 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
- ADuCM3029 EZ-Kit
- ADuCM4050 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

11.1 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

		cycles
	xQueueSend(task waiting, context switch to pending task)	1434
	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
		Data	Code	Total
------------	--------------------------------------	-------	-------	-------
Mutexes	Basic project using 1 static object	59395	32114	91509
	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

11.2 ADSP-SC589 (Cortex-A Core) Benchmark Data

ADSP-SC589 Cortex-A	Core Performance Metrics
---------------------	---------------------------------

		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

		cycles
	xQueueSend(task waiting, context switch to pending task)	1309
	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

		Data	Code	Total
Mutexes	Basic project using 1 static object	97812	18704	116516
	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

11.3 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

		cycles
	xQueueSend(no task pending, no context switch)	476
	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

		Data	Code	Total
	Basic project using 2 static objects	59966	31692	91658
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

11.4 ADuCM3029 Benchmark Data

ADuCM3029 Performance Metrics

		cycles
FreeRTOS_FLGISR	xEventGroupWaitBits (flag available)	258
	xEventGroupWaitBits (flag unavailable, context switch to new task)	604
	xEventGroupSetBits (no task pending, no context switch)	199
	xEventGroupSetBits (task waiting, context switch to pending task)	529
	xEventGroupSetBits (from an ISR, switching to a pending task)	942
FreeRTOS_ISR	Interrupt service time (FreeRTOS)	128
	Time to return from an ISR (FreeRTOS, no task switch)	45
FreeRTOS_MSGISR	xQueueReceive (message available)	213

		cycles
	xQueueReceive (message unavailable, context switch to new task)	1043
	xQueueSend (no task pending, no context switch)	243
	xQueueSend (task waiting, context switch to pending task)	647
	xQueueSend (from an ISR, switching to a pending task)	309
FreeRTOS_MUTISR	xSemaphoreTake (mutex available)	184
	xSemaphoreTake (mutex unavailable, context switch to new task	1252
	xSemaphoreGive (no task pending, no context switch)	232
	xSemaphoreGive (task waiting, context switch to pending task)	754
FreeRTOS_SEMISR	xSemaphoreTake (semaphore available)	175
	xSemaphoreTake (semaphore unavailable, context switch to new task)	1068
	xSemaphoreGive (no task pending, no context switch)	207
	xSemaphoreGive (task waiting, context switch to pending task)	578
	xSemaphoreGive (from an ISR, switching to a pending task)	266

ADuCM3029 Sizing Metrics

		Data	Code	Total
NONE	Basic project	4168	7632	11800
Message Queues	Basic project using 1 static object	4172	7692	11864
	Basic project using 2 static objects	4244	7692	11936

		Data	Code	Total
Flags	Basic project using 1 static object	4124	8136	12260
	Basic project using 2 static objects	4152	8136	12288
Mutexes	Basic project using 1 static object	4168	7728	11896
	Basic project using 2 static objects	4240	7728	11968
Semaphores	Basic project using 1 static object	4168	7680	11848
	Basic project using 2 static objects	4240	7680	11920
ALL	Basic project using 1 static object	4272	8320	12592
	Basic project using 2 static objects	4444	8320	12764

11.5 ADuCM4050 Benchmark Data

ADuCM4050 Performance Metrics

		cycles
FreeRTOS_FLGISR	xEventGroupWaitBits (flag available)	259
	xEventGroupWaitBits (flag unavailable, context switch to new task)	594
	xEventGroupSetBits (no task pending, no context switch)	208
	xEventGroupSetBits (task waiting, context switch to pending task)	532
	xEventGroupSetBits (from an ISR, switching to a pending task)	993
FreeRTOS_ISR	Interrupt service time (FreeRTOS)	386
	Time to return from an ISR (FreeRTOS, no task switch)	157
FreeRTOS_MSGISR	xQueueReceive (message available)	217

		cycles
	xQueueReceive (message unavailable, context switch to new task)	1039
	xQueueSend (no task pending, no context switch)	236
	xQueueSend (task waiting, context switch to pending task)	668
	xQueueSend (from an ISR, switching to a pending task)	330
FreeRTOS_MUTISR	xSemaphoreTake (mutex available)	193
	xSemaphoreTake (mutex unavailable, context switch to new task	1217
	xSemaphoreGive (no task pending, no context switch)	234
	xSemaphoreGive (task waiting, context switch to pending task)	791
FreeRTOS_SEMISR	xSemaphoreTake (semaphore available)	165
	xSemaphoreTake (semaphore unavailable, context switch to new task)	1034
	xSemaphoreGive (no task pending, no context switch)	216
	xSemaphoreGive (task waiting, context switch to pending task)	607
	xSemaphoreGive (from an ISR, switching to a pending task)	273

ADuCM4050 Sizing Metrics

		Data	Code	Total
NONE	Basic project	4168	8084	12252
Message Queues	Basic project using 1 static object	4172	8144	12316
	Basic project using 2 static objects	4244	8144	12388

		Data	Code	Total
Flags	Basic project using 1 static object	4124	8588	12712
	Basic project using 2 static objects	4152	8588	12740
Mutexes	Basic project using 1 static object	4168	8180	12348
	Basic project using 2 static objects	4240	8180	12420
Semaphores	Basic project using 1 static object	4168	8132	12300
	Basic project using 2 static objects	4240	8132	12372
ALL	Basic project using 1 static object	4272	8772	13044
	Basic project using 2 static objects	4444	8772	13216

11.6 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

		cycles
FreeRTOS_MSGISR	xQueueReceive(message available)	447
	xQueueReceive(message unavailable, context switch to new task)	2937
	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

		Data	Code	Total
	Basic project using 2 static objects	6707	13762	20469
Flags	Basic project using 1 static object	6567	14402	20969
	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