
# FreeRTOS User's Guide

FreeRTOS User's Guide
Version 1.5.1, August 2021


http://www.analog.com
processor.tools.support@analog.com

# Contents

## Copyright Information



## Disclaimer



## Trademark and Service Mark Notice

# 1 Introduction

## 1.1 Analog Devices FreeRTOS

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

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

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

## 1.2 What's in the user's guide

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

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<br>ADSP-SC584 EZ-Kit<br>ADSP-SC573 EZ-Kit<br>ADSP-SC594 EZ-Kit |
| ADSP-SC5xx SHARC+ Core | ADSP-SC589 EZ-Kit<br>ADSP-SC584 EZ-Kit<br>ADSP-SC573 EZ-Kit<br>ADSP-SC594 EZ-Kit |
| ADSP-BF7xx | ADSP-BF707 EZ-Kit |
| ADSP-2156x | ADSP-21569 EZ-Kit |

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

# 2 Hardware and software set up

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

## 2.1 Get the hardware ready

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

Below is a list of the hardware involved.

ADI reference board:

- ADSP-SC594 SOM and ADSP-SOMCRR-EZKIT: https://www.analog.com/en/products/adsp-sc594.html#product-evaluationkit
- ADSP-SC589 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/EVAL-ADSP-SC589.html
- ADSP-SC584 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/EVAL-ADSP-SC584.html
- ADSP-SC573 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/SC573EZKIT.html
- ADSP-BF707 EZ-kit: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/eval-bf707.html
- ADSP-21569 EZ-kit: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/ADZS-21569-EZKIT.html

Jtag debugger:

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

PC:

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

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

## 2.2 Get the source code ready

This page describes how to get the FreeRTOS source code.

### 2.2.1 Download FreeRTOS source code

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

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

### 2.2.2 Install the FreeRTOS product from Analog Devices

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

For example, unzip them into folder "freertos":

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

You will get the path such as "C:\Analog Devices\freertos\FreeRTOSv10.0.0\FreeRTOS".

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

It will add new files and overwrite some files saved in "C:\Analog Devices\freertos\FreeRTOSv10.0.0 \FreeRTOS".

## 2.3 Software environment set up for CrossCore Embedded Studio

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

**ADSP-SC589/ADSP-SC584/ADSP-SC573/ADSP-SC594**

- Analog Devices CrossCore Embedded Studio version 2.10.0 or later

**ADSP-BF7XX EZ-Kit**

- Analog Devices CrossCore Embedded Studio version 2.10.0 or later

**ADSP-2156X EZ-Kit**

- Analog Devices CrossCore Embedded Studio version 2.10.0 or later

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

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

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

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

The tasks performed in the Analog Devices Basic Demo include:

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

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

### 3.1.1 Overview

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

### 3.1.2 Environment Setup

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

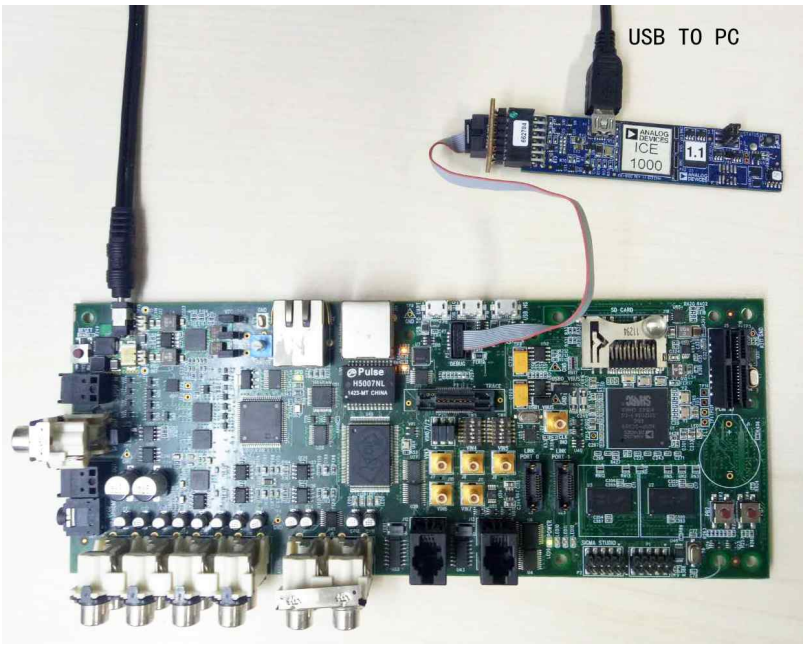
### Software Requirement

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

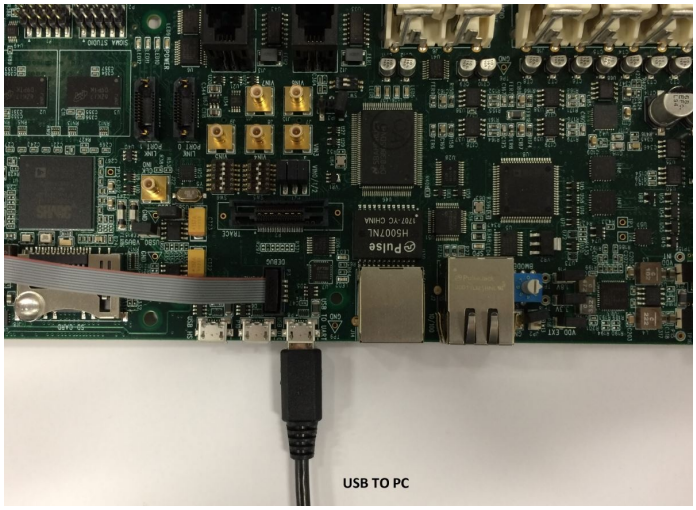
### Hardware Setup

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

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




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

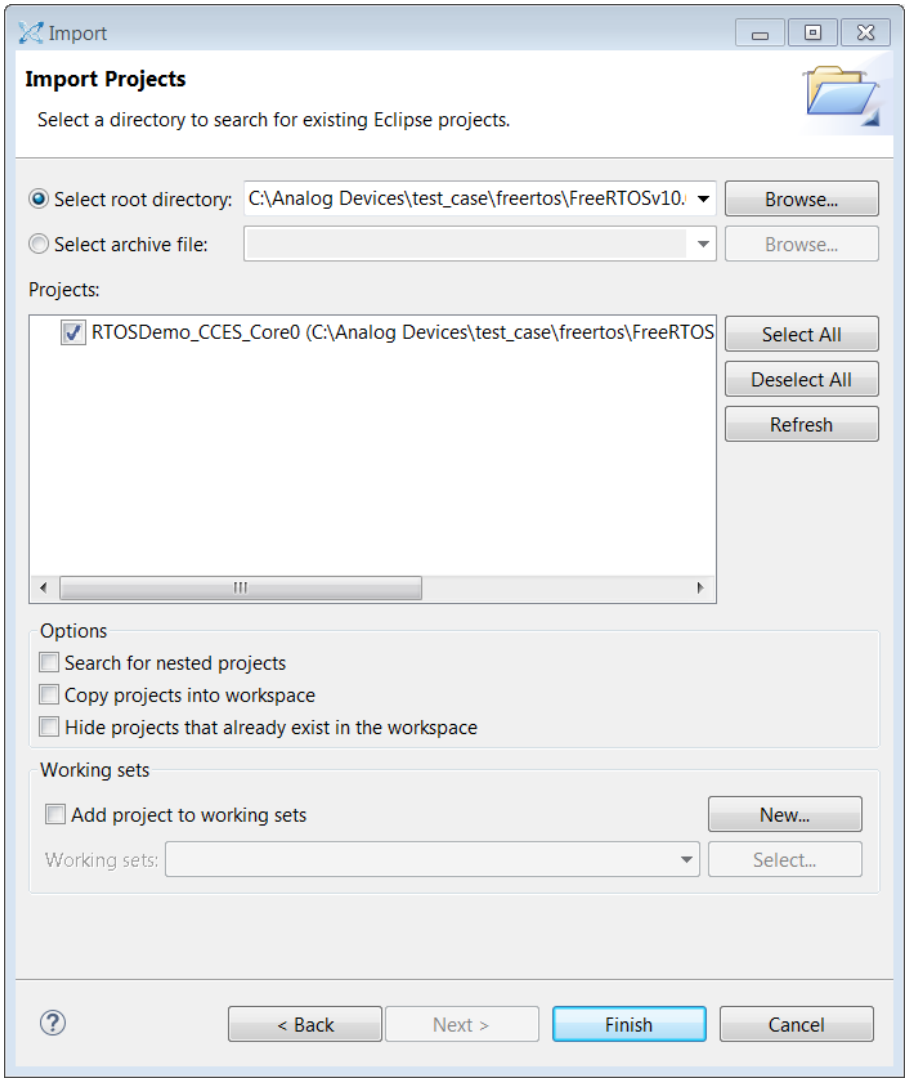



### 3.1.3 Build the Example

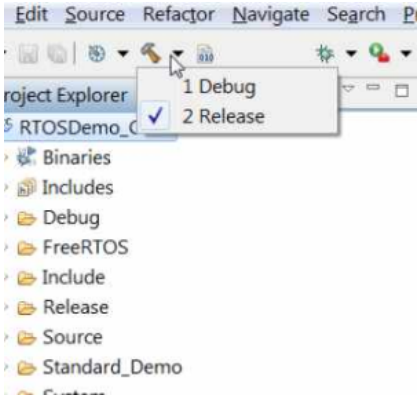
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the **File** menu and then select the **Import** option from the menu
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX_A5_ADSP_SC589_CCES** folder
- Click **Finish** to close the file browser dialog
- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the **projects** pane and click **Import**




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




3. Build the project in CrossCore Embedded Studio:

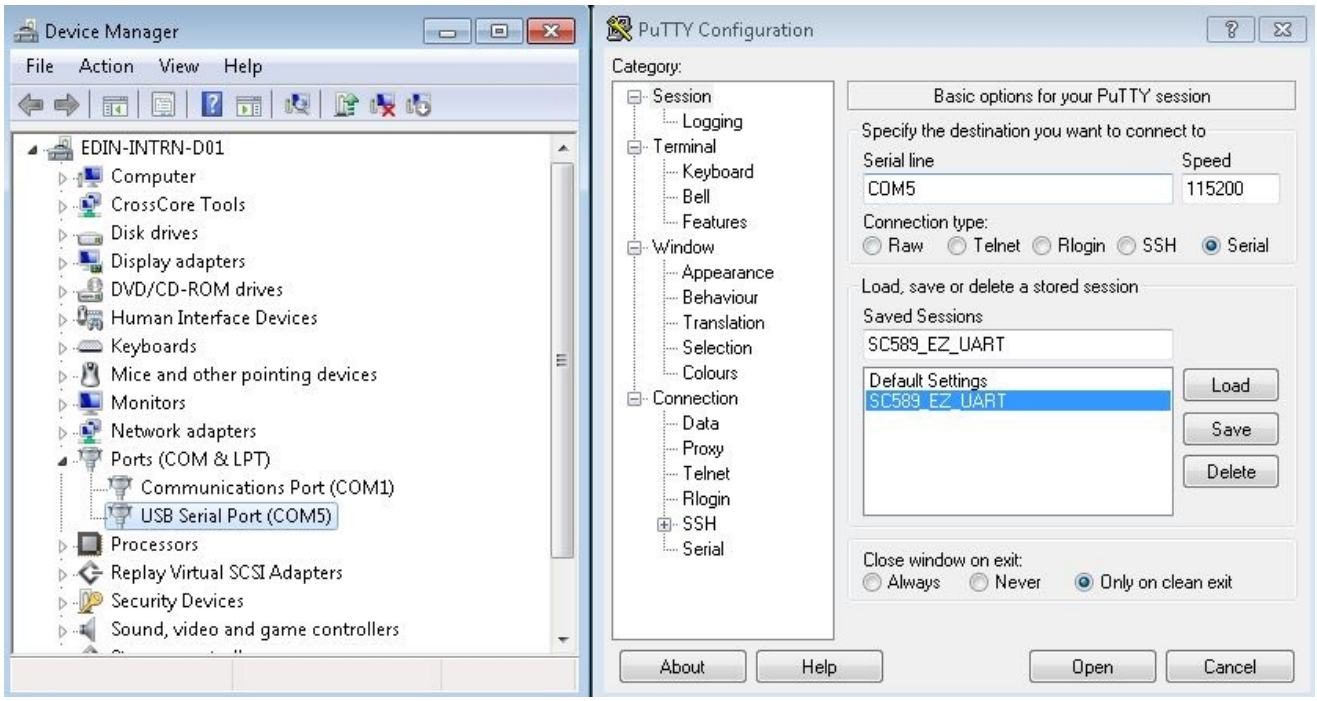
- In the **Project Explorer** right click on the **RTOSDemo_CCES_Core0** project and select the **Build Project** option from the menu

### 3.1.4 Run the Example

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

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

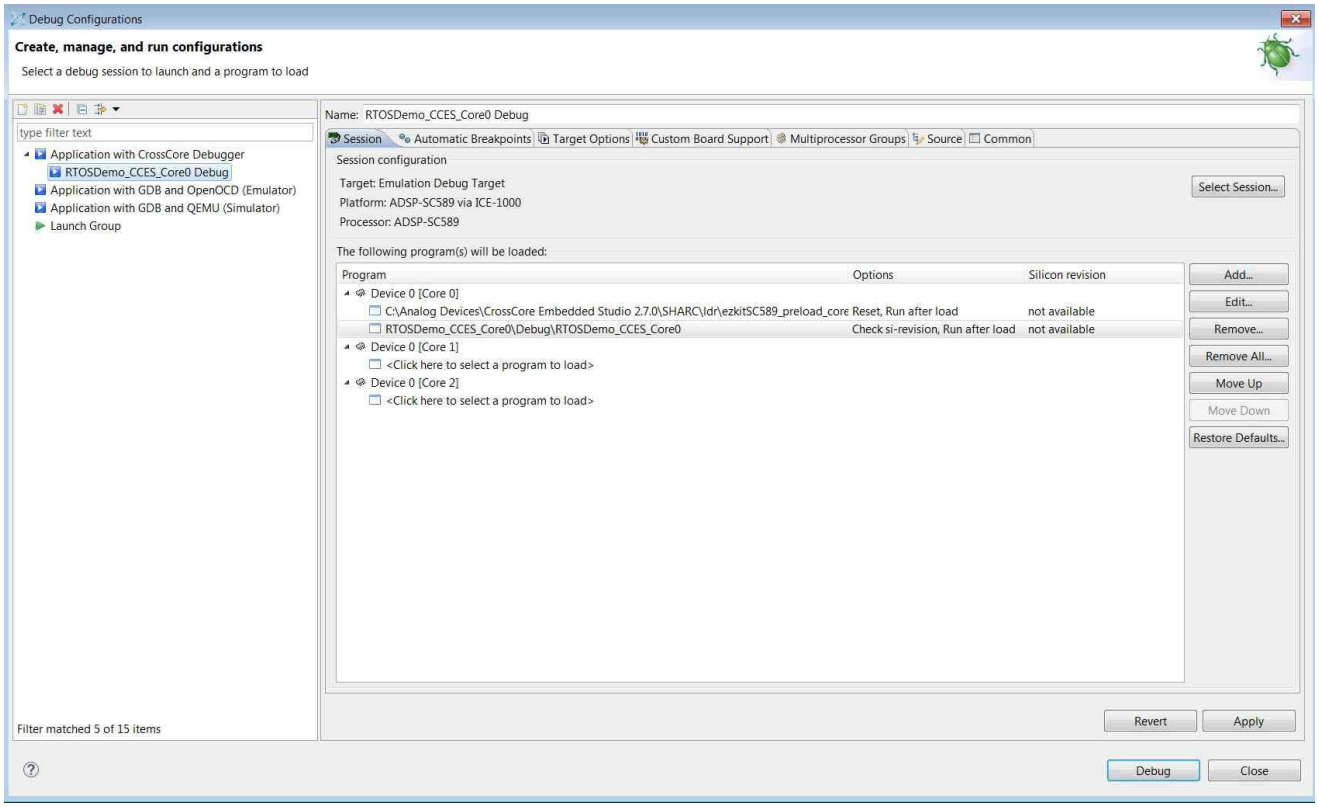
Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the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



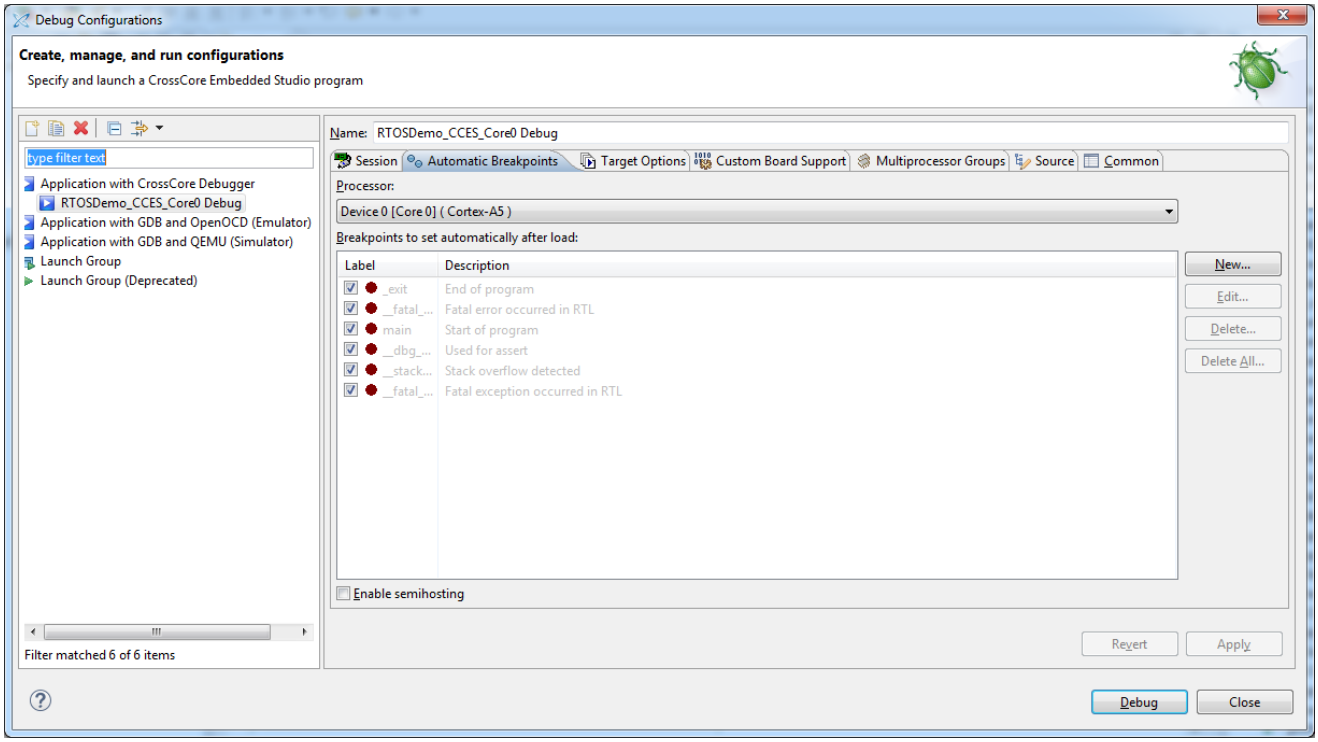

After this, follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_CCES_Core0** project and select the **Debug As** option from the menu

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

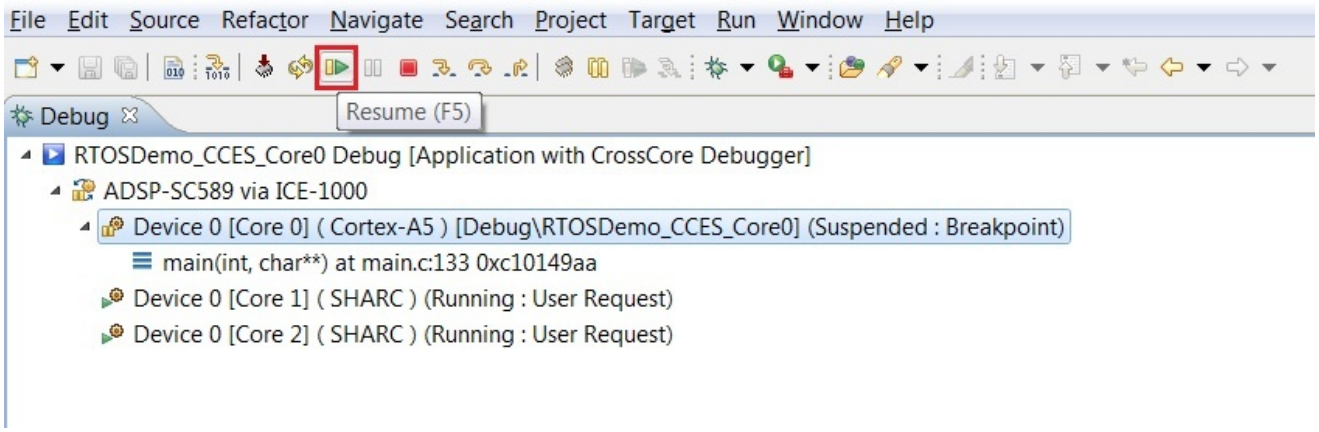



3 . Disable the **semihosting** function in **Automatic Breakpoints**




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

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




### 3.1.5 Test Results

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

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

### 3.2.1 Overview

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

### 3.2.2 Environment Setup

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

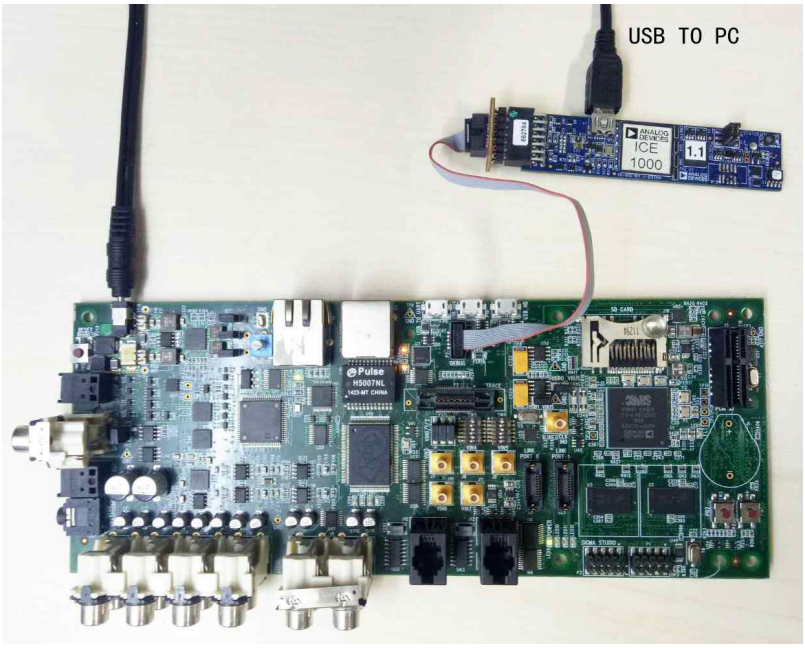
#### Software Requirement

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

#### Hardware Setup

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

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

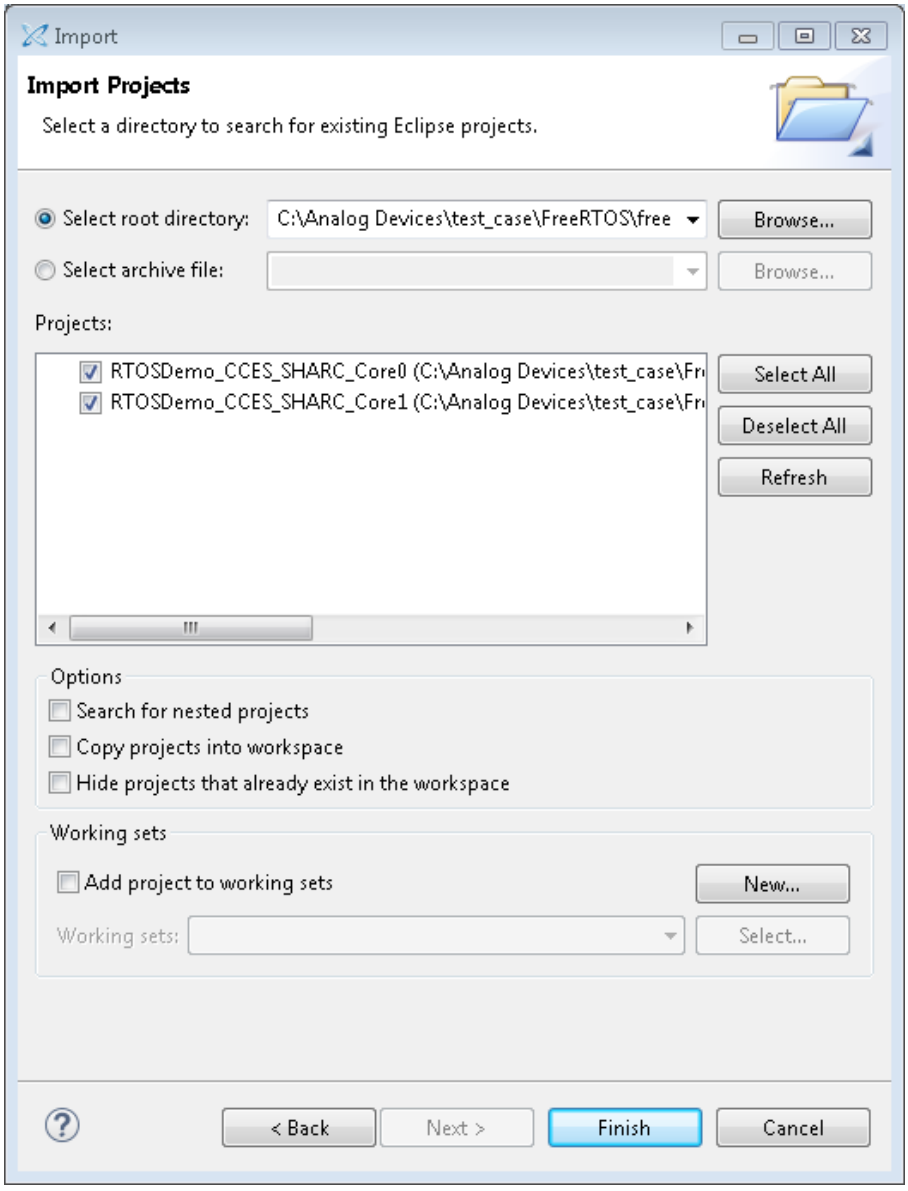



### 3.2.3 Build the Example

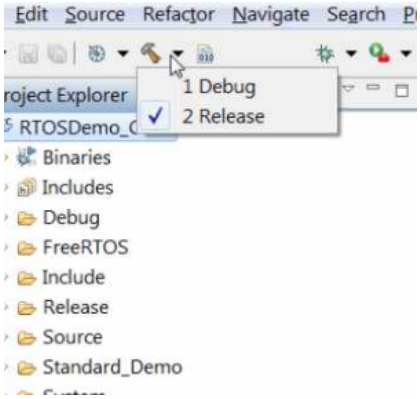
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the **File** menu and then select the **Import** option from the menu
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC_ADSP_SC589_CCES** folder
- Click **Finish** to close the file browser dialog
- Two projects should appear in the **Project Explorer**




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




3. Build the project in CrossCore Embedded Studio:

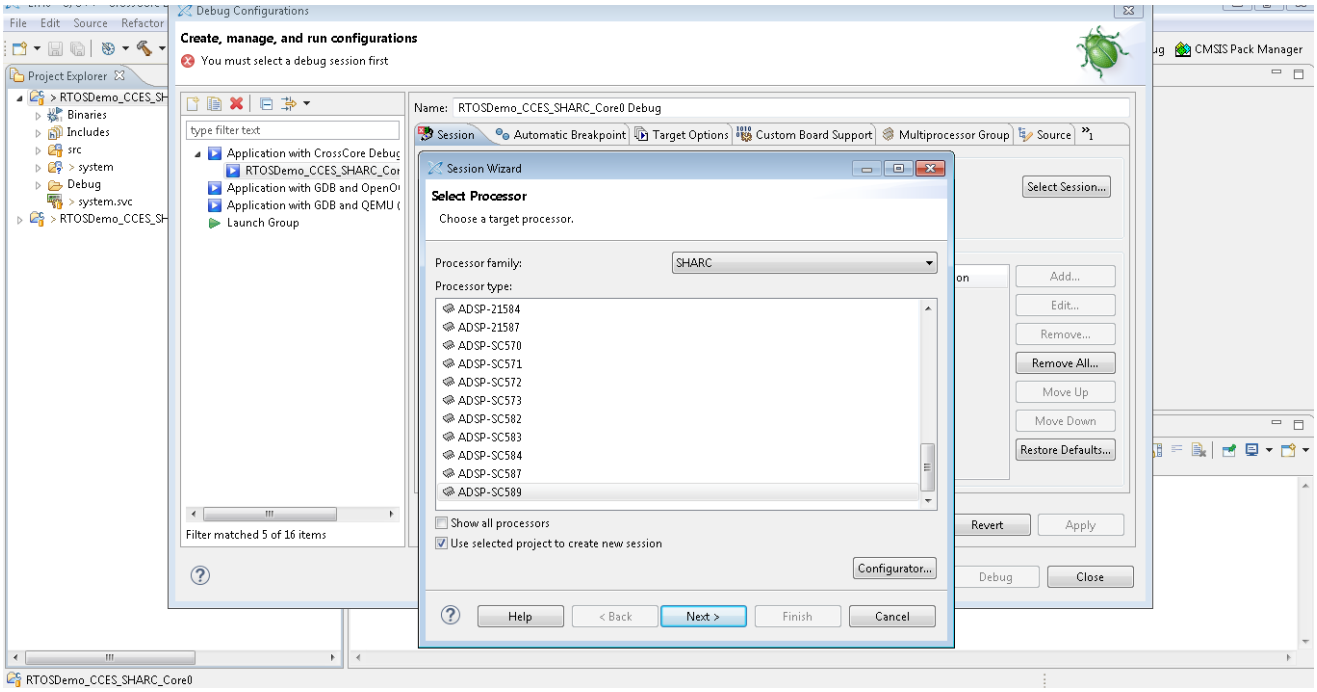
- In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_Core0** and **RTOSDemo_CCES_SHARC_Core1** project, then select the **Build Project** option from the menu

### 3.2.4 Run the Example

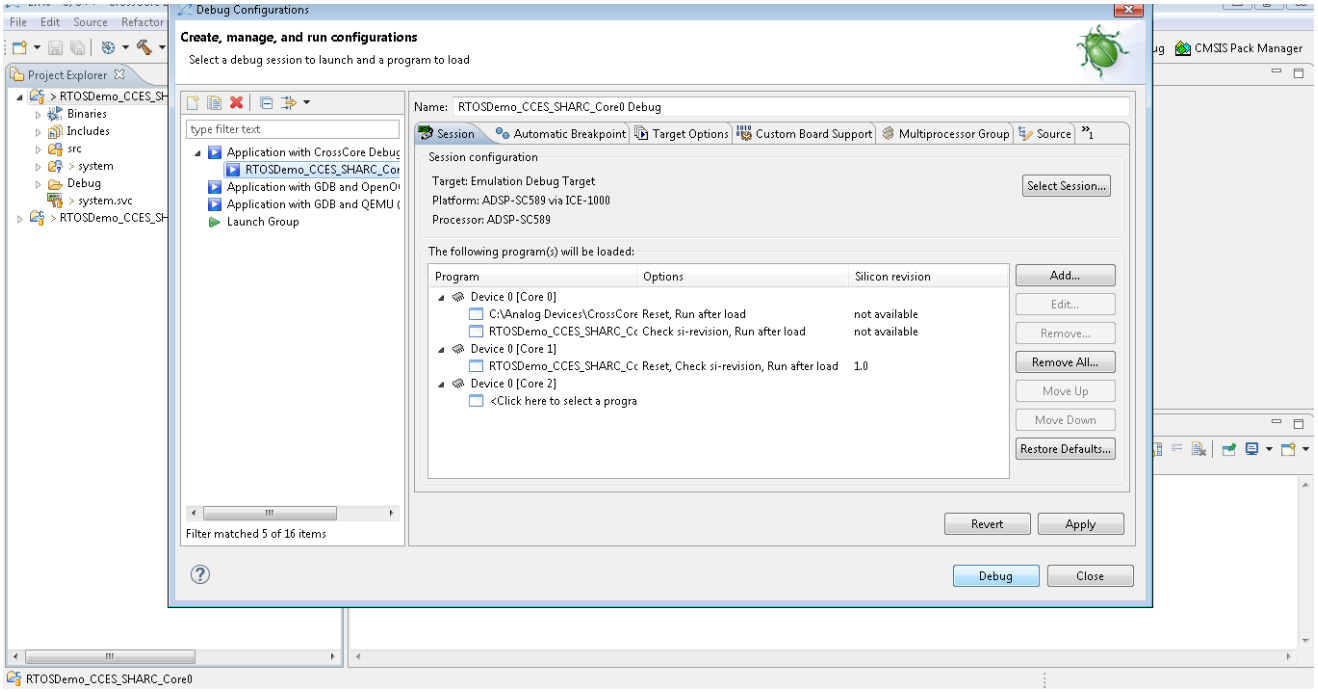
Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_Core0** project and select the **Debug As** option from the menu

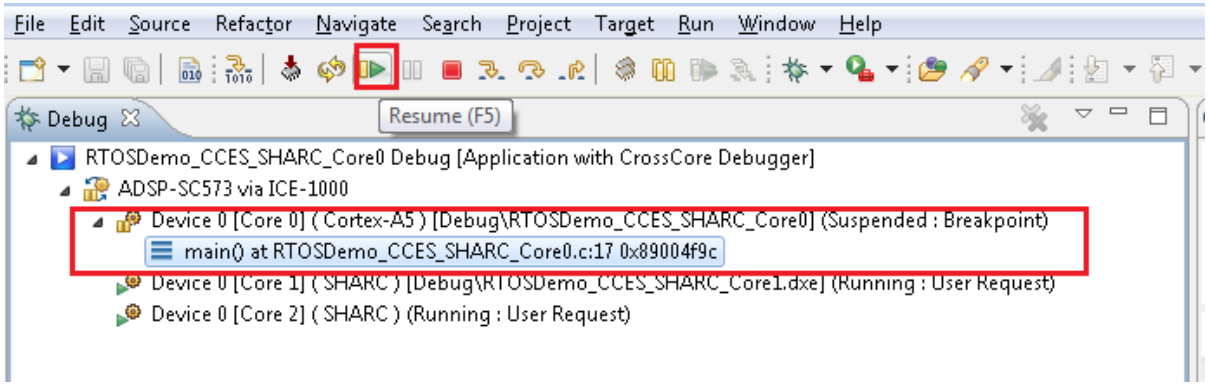
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board



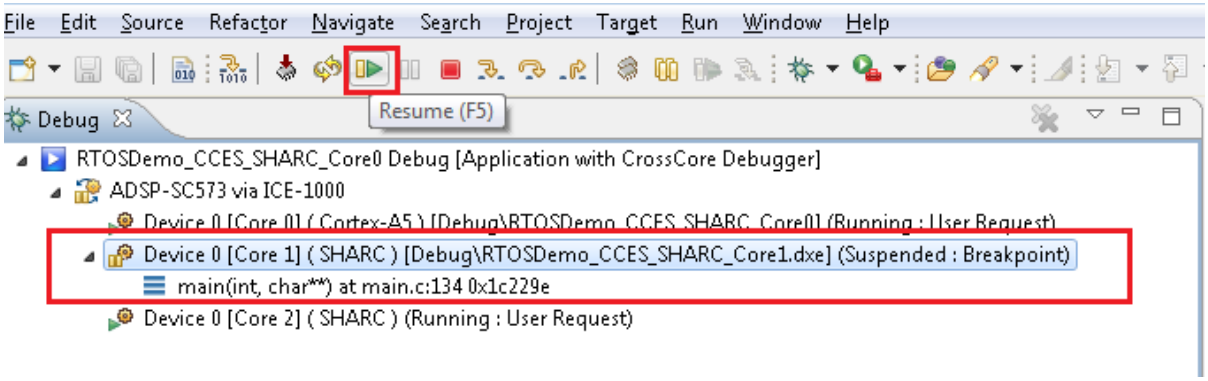

3. Click the **Debug** button to close the **Debug ConfigurationS** window




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

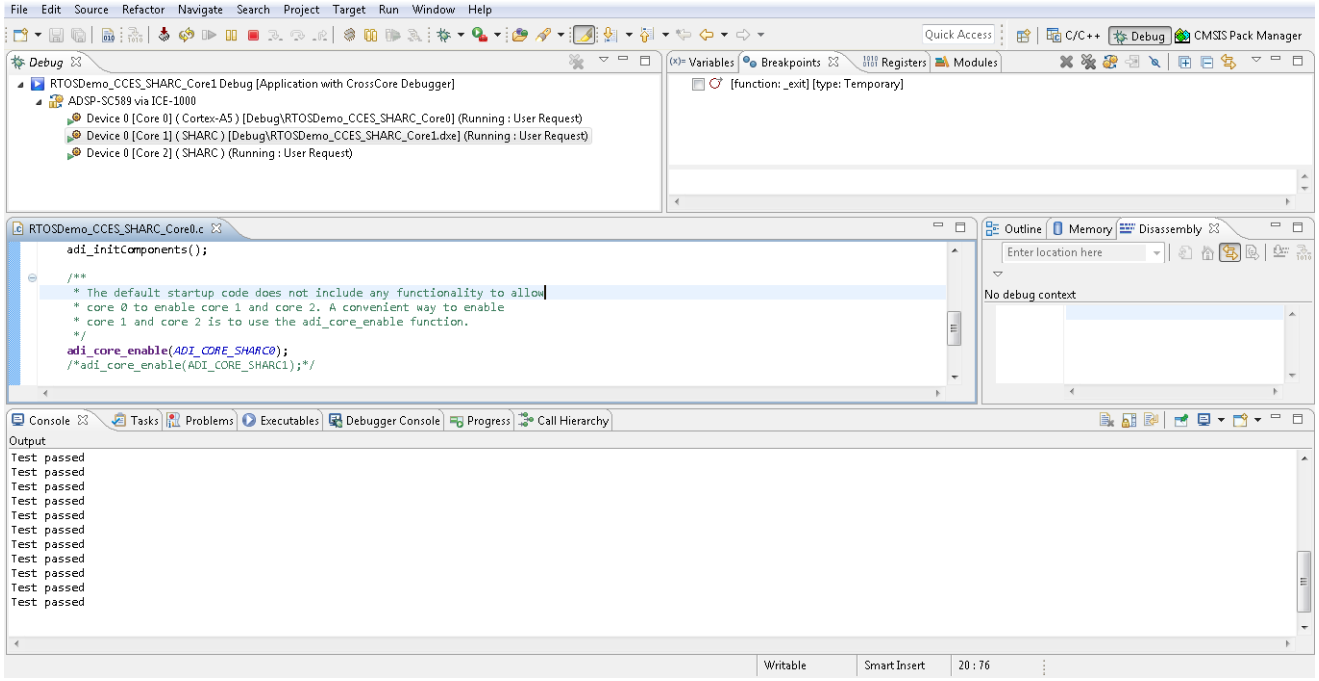



5. Then Choose Core1 and keep to click **Run/Resume** button to start running Core1 application



### 3.2.5 Test Results

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



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

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

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

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

The tasks performed in the Analog Devices Basic Demo include:

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

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

### 4.1.1 Overview

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

### 4.1.2 Environment Setup

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

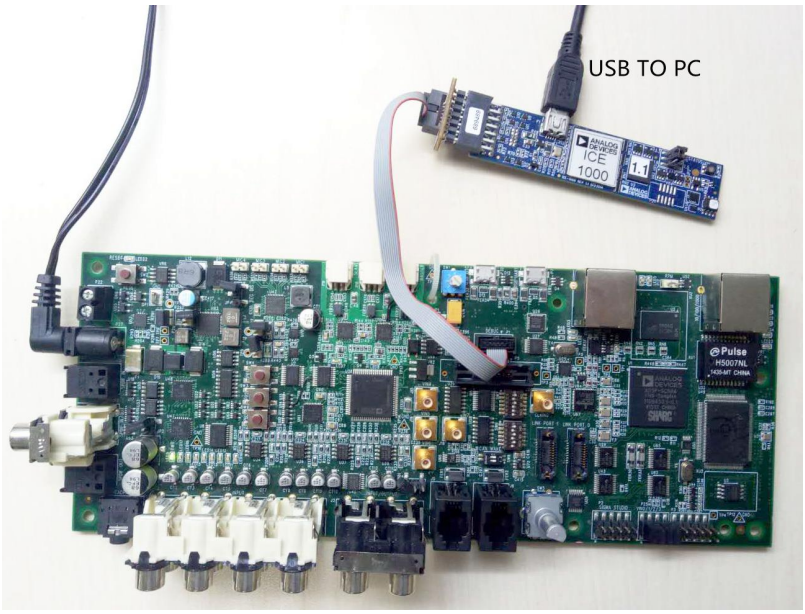
### Software Requirement

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

### Hardware Setup

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

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



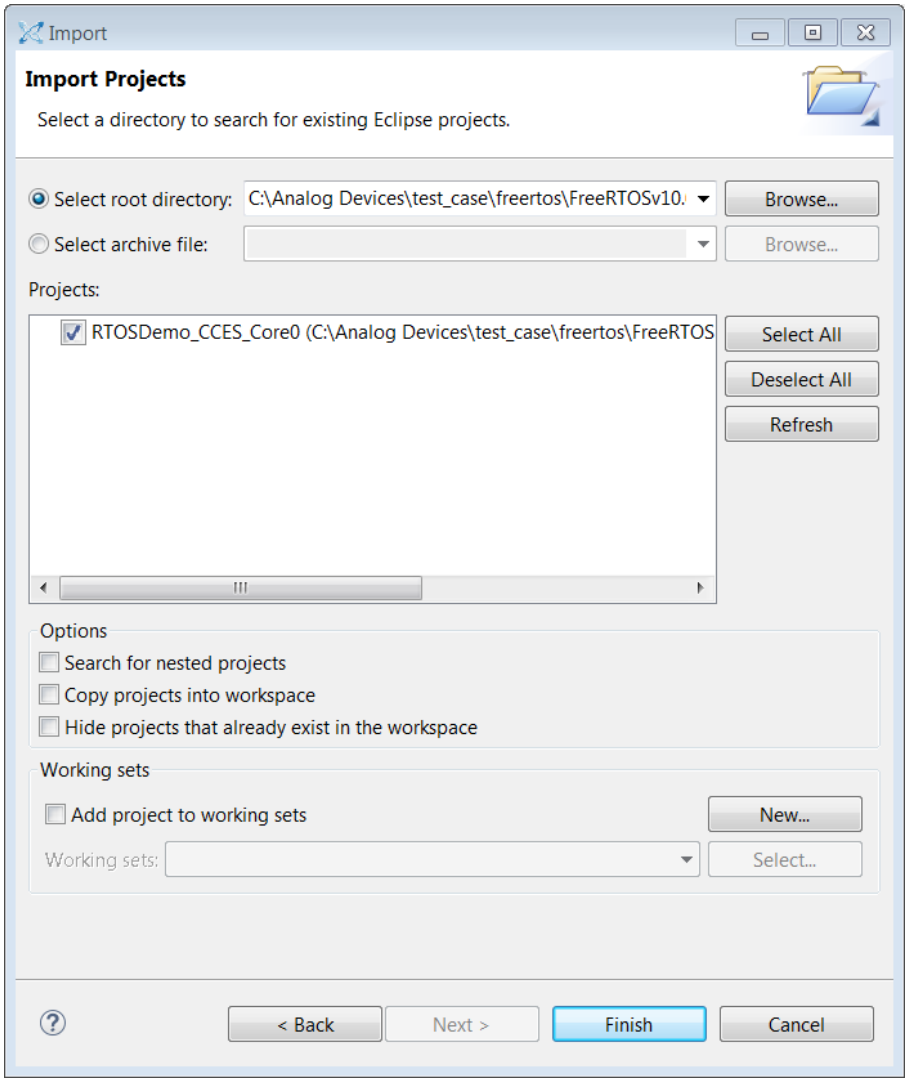

Connect the **USB to UART** port of the EZ-Kit to the host PC with a USB cable as shown below:


### 4.1.3 Build the Example

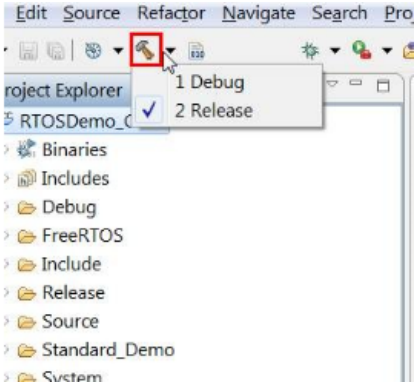
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the **File** menu and then select the **Import** option from the menu
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX_A5_ADSP_SC584_CCES** folder
- Click **Finish** to close the file browser dialog
- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the **projects** pane and click **Import**




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




3. Build the project in CrossCore Embedded Studio:

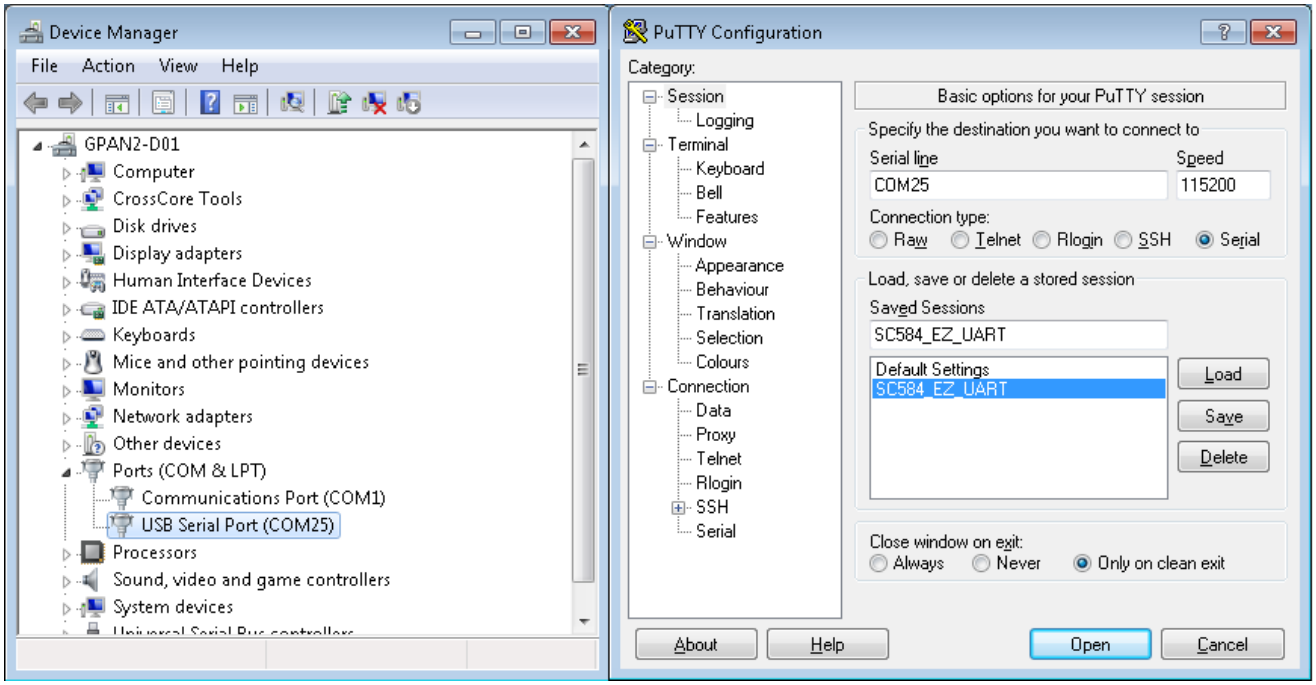
- In the **Project Explorer** right click on the **RTOSDemo_CCES_Core0** project and select the **Build Project** option from the menu

### 4.1.4 Run the Example

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

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

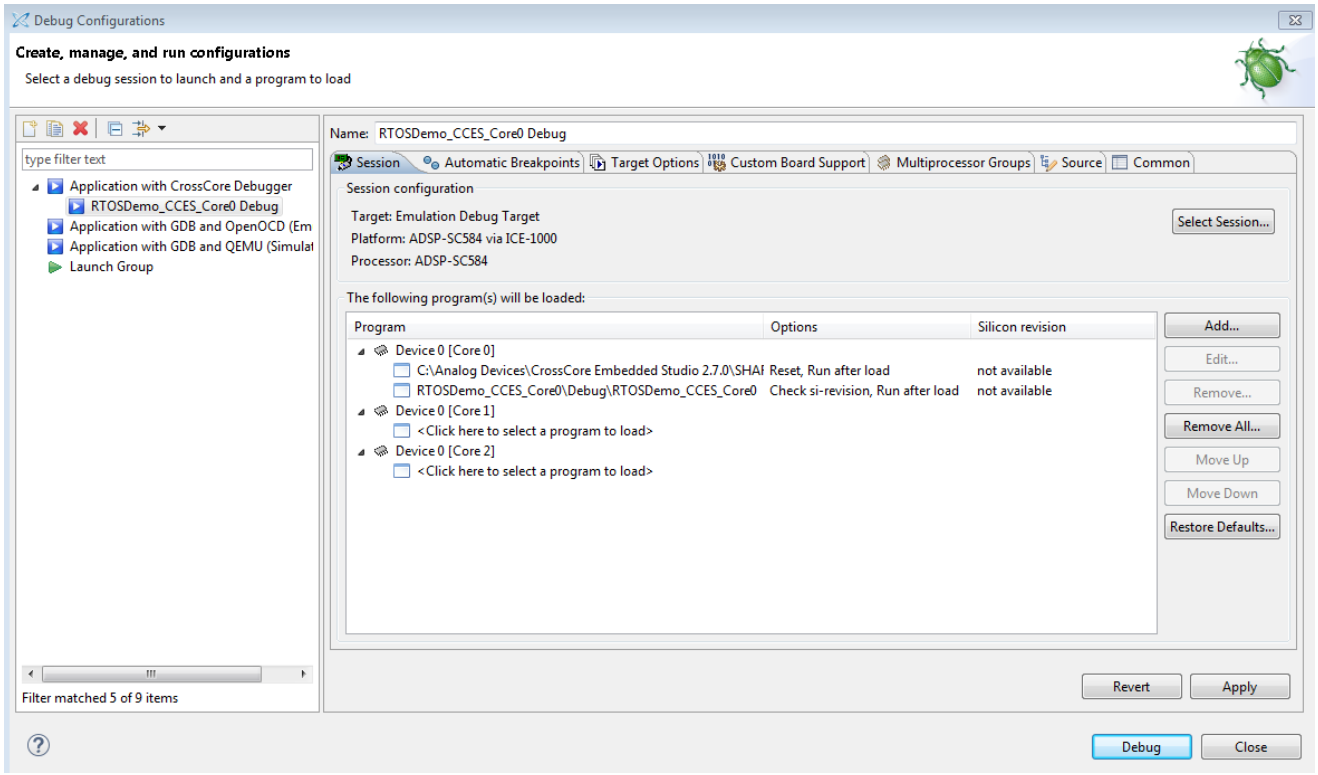
Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the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



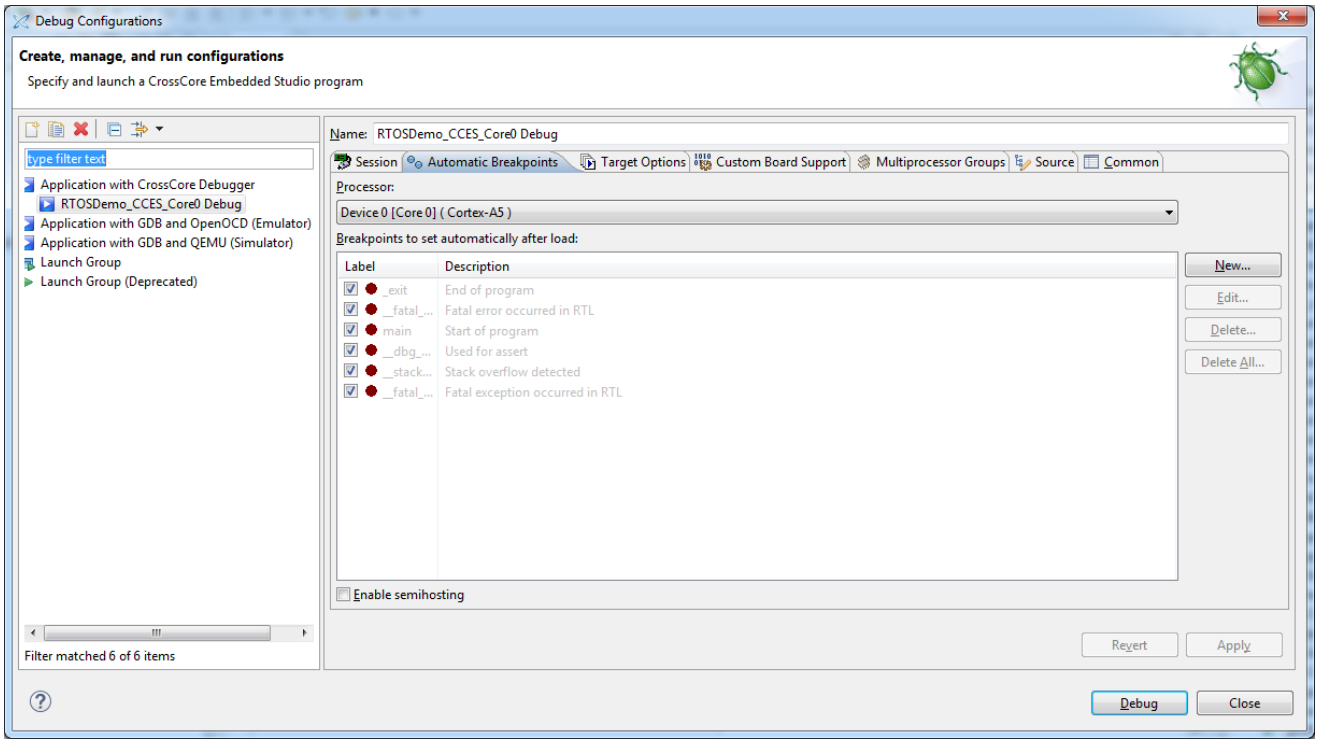

Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_CCES_Core0** project and select the **Debug As** option from the menu

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

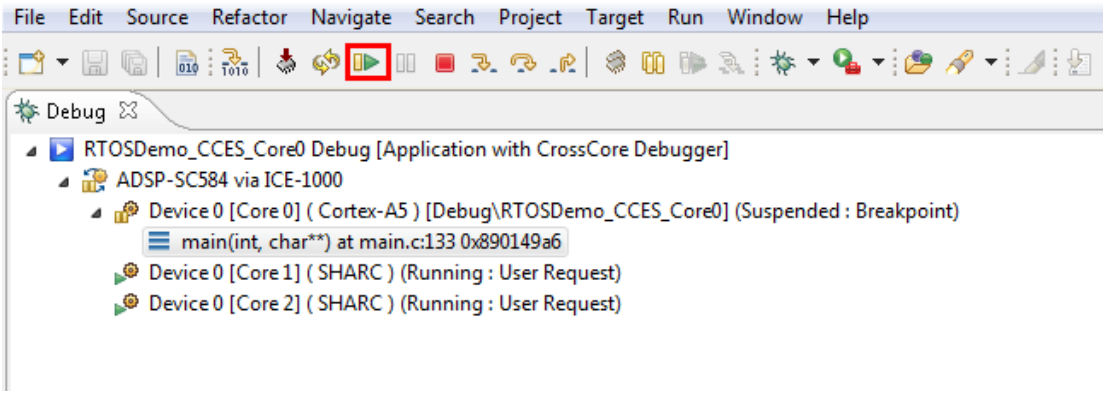



3. Disable the **semihosting** function in **Automatic Breakpoints**




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

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




### 4.1.5 Test Results

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

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

### 4.2.1 Overview

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

### 4.2.2 Environment Setup

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

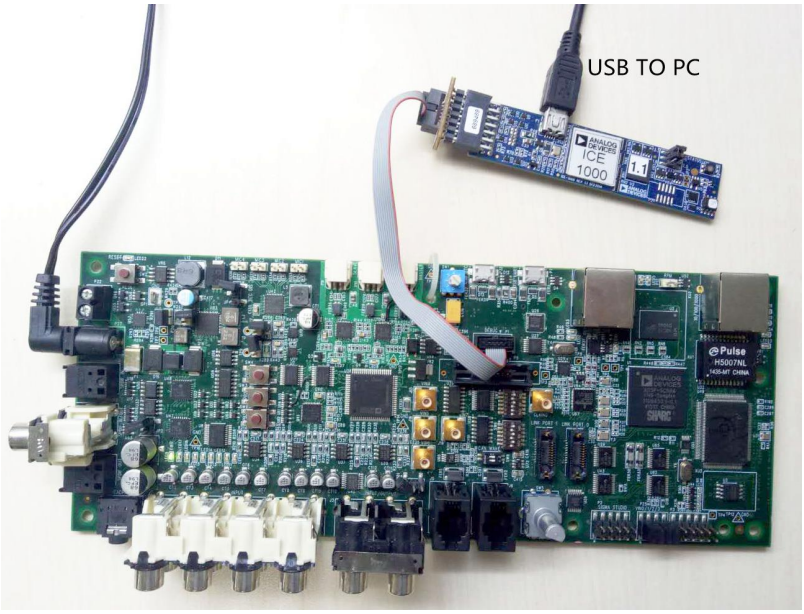
#### Software Requirement

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

#### Hardware Setup

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

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

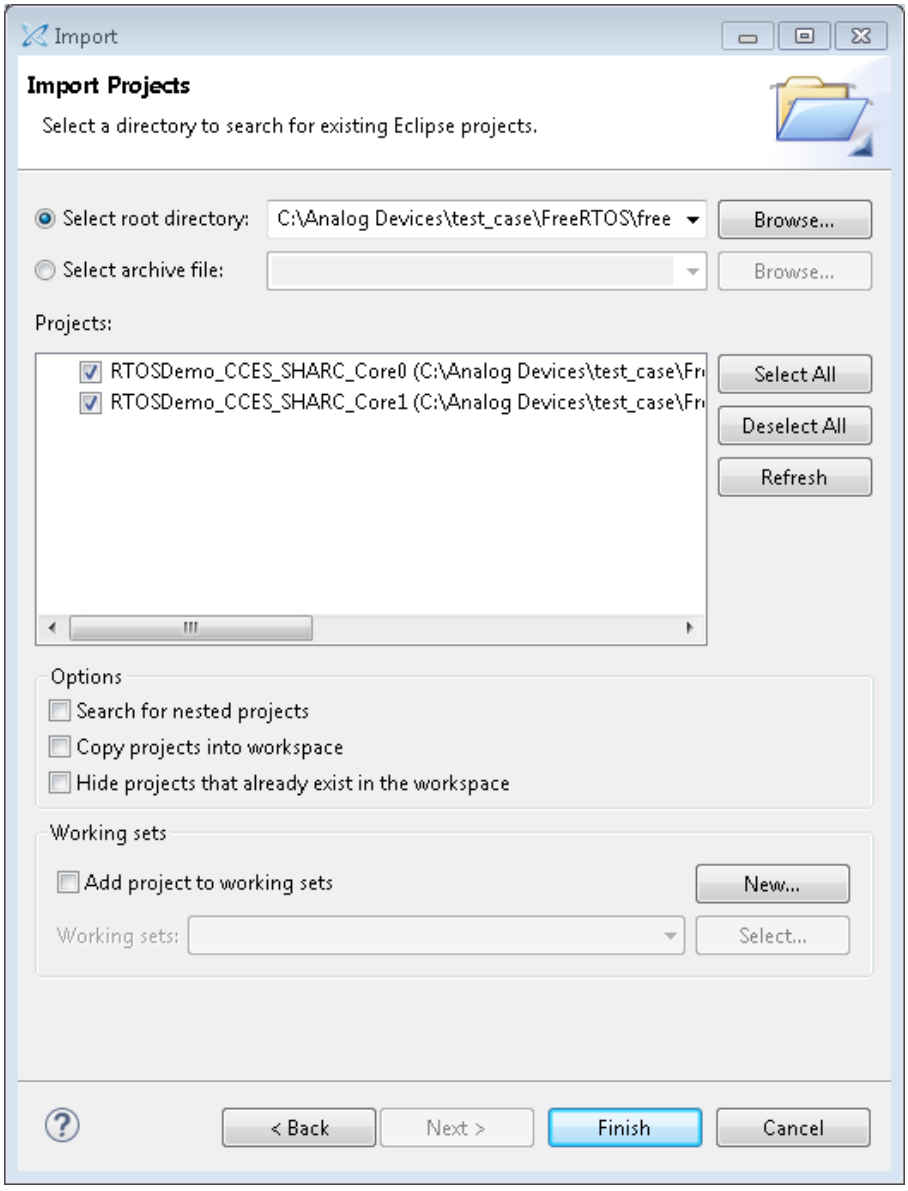



### 4.2.3 Build the Example

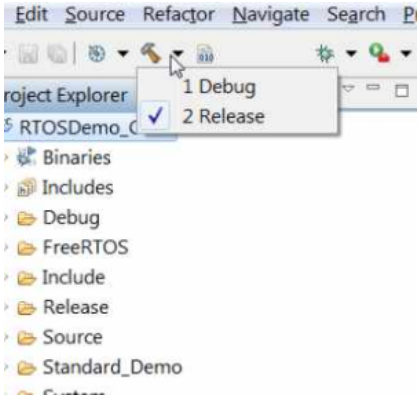
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the **File** menu and then select the **Import** option from the menu
- When the **Import** project window appears:
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC_ADSP_SC584_CCES** folder
- Click **Finish** to close the file browser dialog
- Two projects should appear in the **Project Explorer**




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




3. Build the project in CrossCore Embedded Studio:

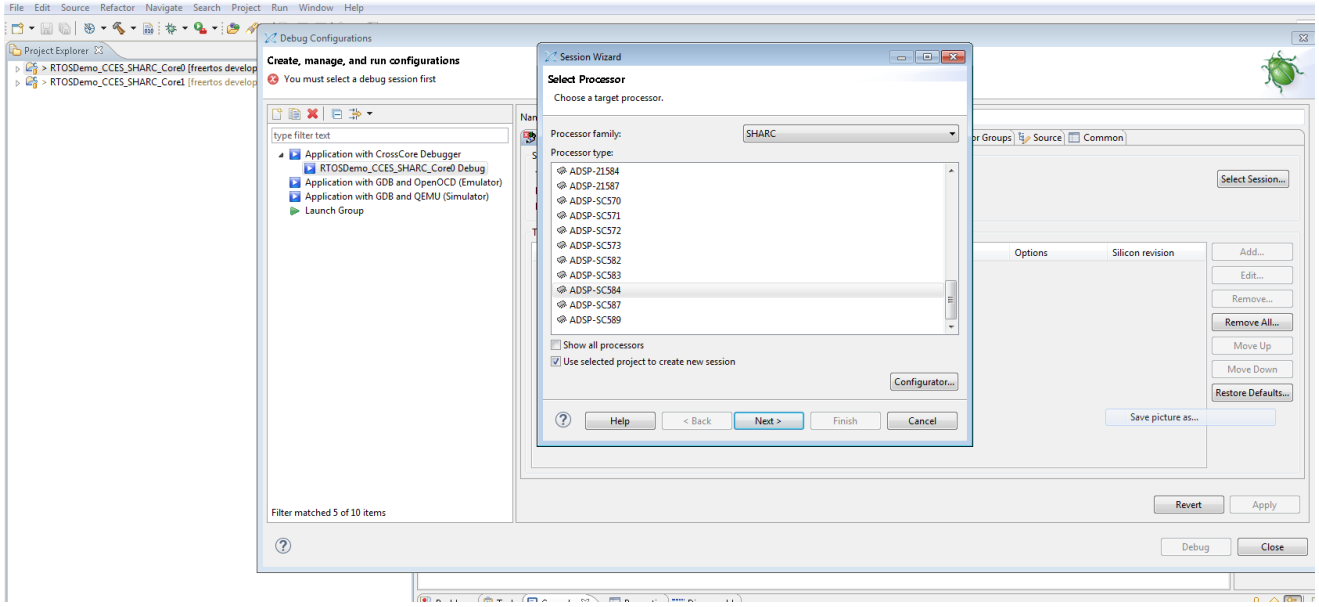
- In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_Core0** and **RTOSDemo_CCES_SHARC_Core1** project, then select the **Build Project** option from the menu

### 4.2.4 Run the Example

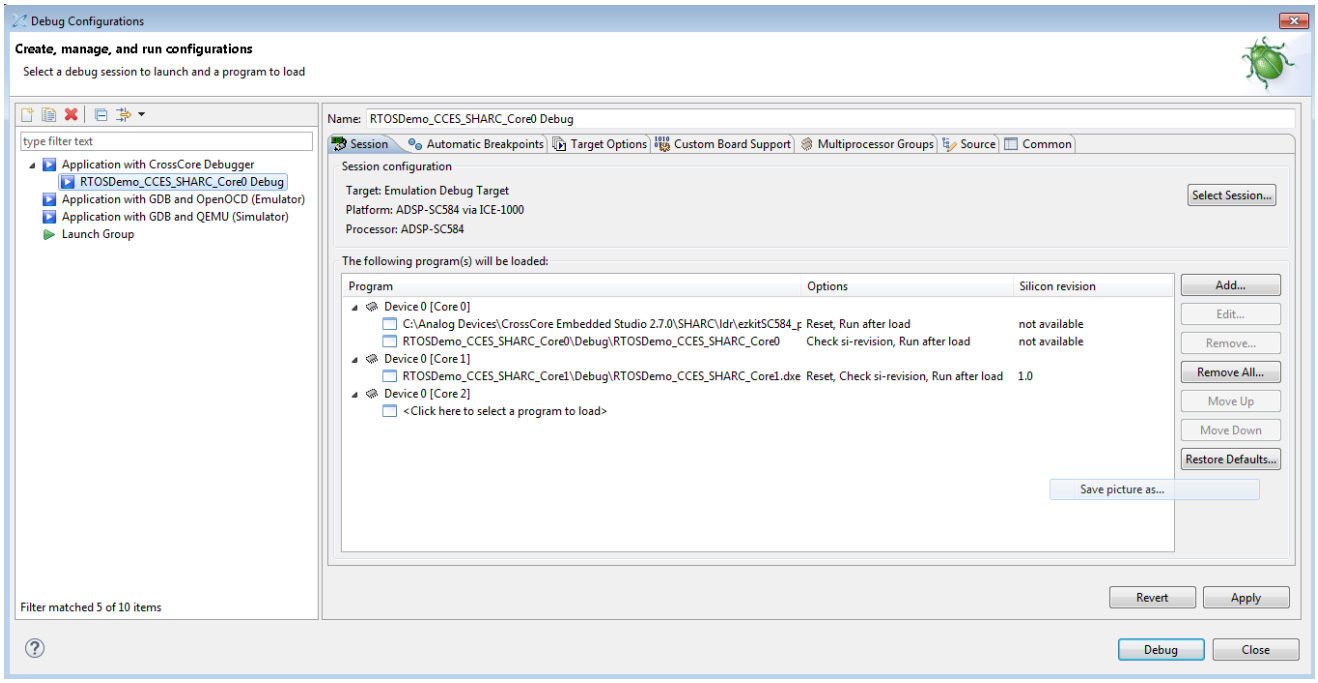
Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_Core0** project and select the **Debug As** option from the menu

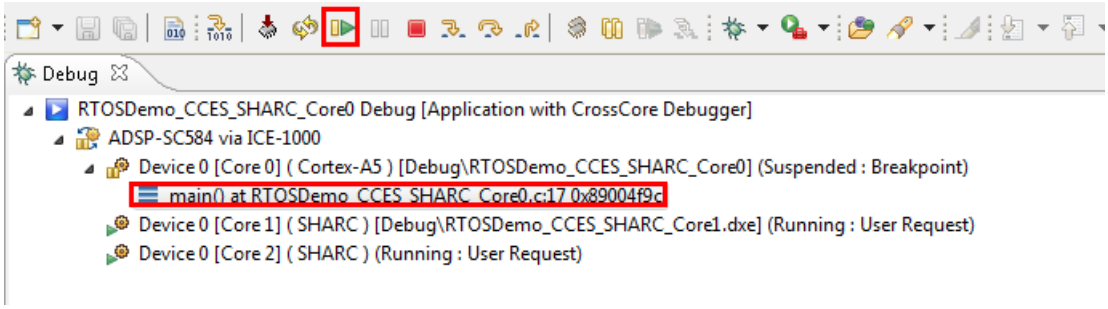
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board



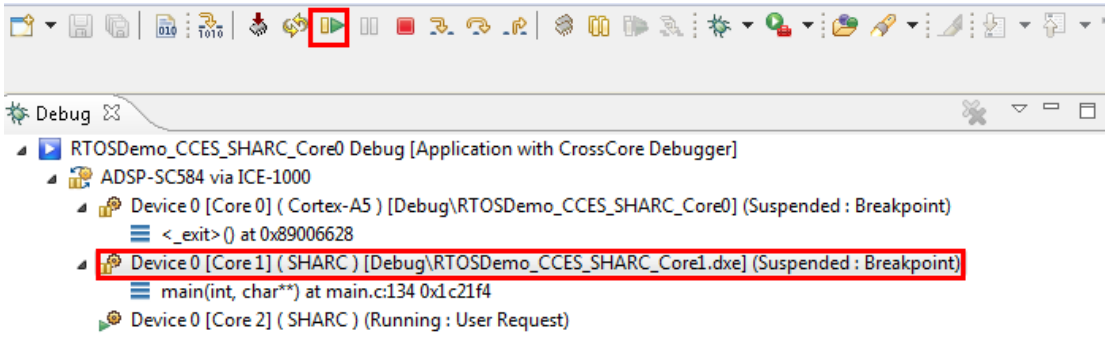

3. Click the **Debug** button to close the **Debug Configurations** window




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

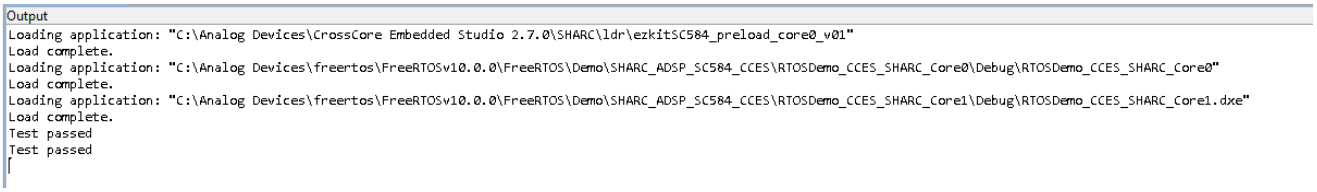



5. Then choose Core1 and keep to click **Run/Resume** button to start running Core1 application




### 4.2.5 Test Results

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


Output
Loading application: "C:\Analog Devices\CrossCore Embedded Studio 2.7.0\SHARC\ldr\ezkitSC584_preload_core0_v01"
Load complete.
Loading application: "C:\Analog Devices\freertos\FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC_ADSP_SC584_CCES\RTOSDemo_CCES_SHARC_Core0\Debug\RTOSDemo_CCES_SHARC_Core0"
Load complete.
Loading application: "C:\Analog Devices\freertos\FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC_ADSP_SC584_CCES\RTOSDemo_CCES_SHARC_Core1\Debug\RTOSDemo_CCES_SHARC_Core1.dxe"
Load complete.
Test passed
Test passed

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

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

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

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

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

The tasks performed in the Analog Devices Basic Demo include:

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

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

### 5.1.1 Overview

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

### 5.1.2 Environment Setup

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

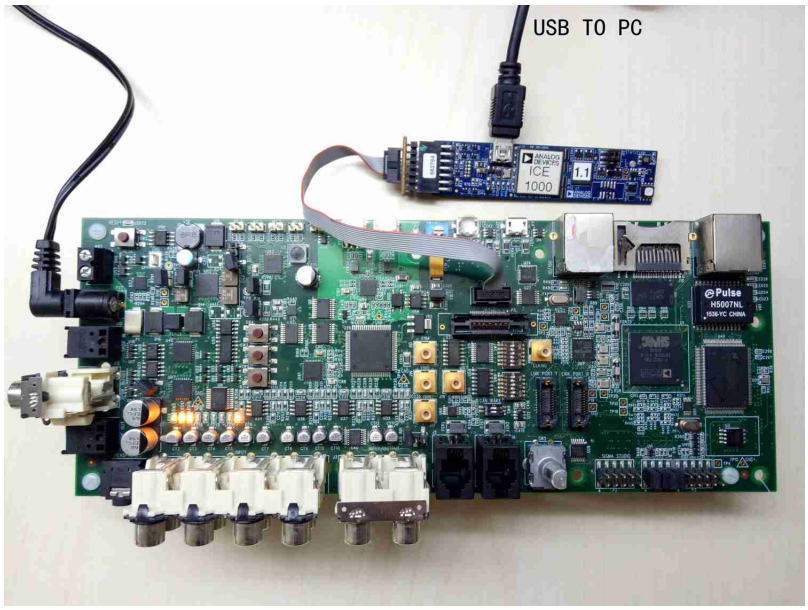
### Software Requirement

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

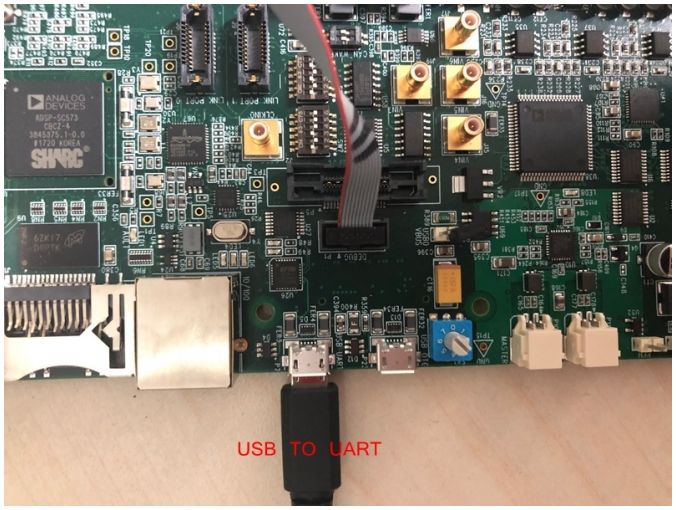
### Hardware Setup

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

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




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

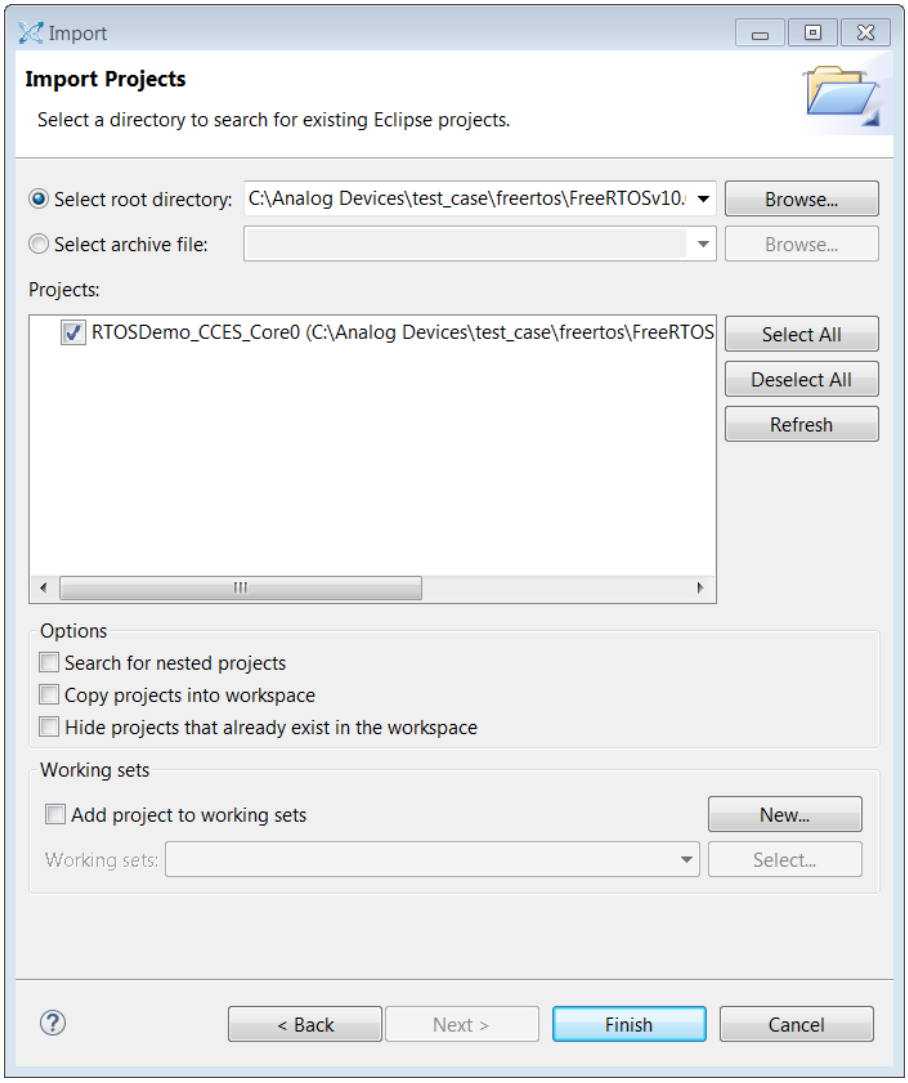



### 5.1.3 Build the Example

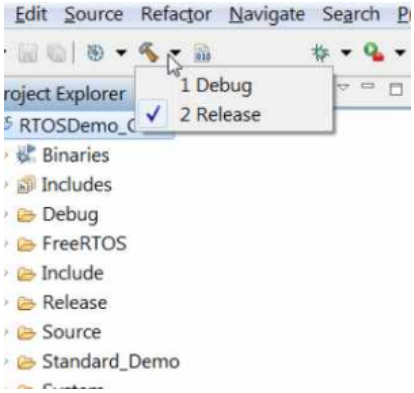
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the **File** menu and then select the **Import** option from the menu
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX_A5_ADSP_SC573_CCES** folder
- Click **Finish** to close the file browser dialog
- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the **projects** pane and click **Import**




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




3. Build the project in CrossCore Embedded Studio:

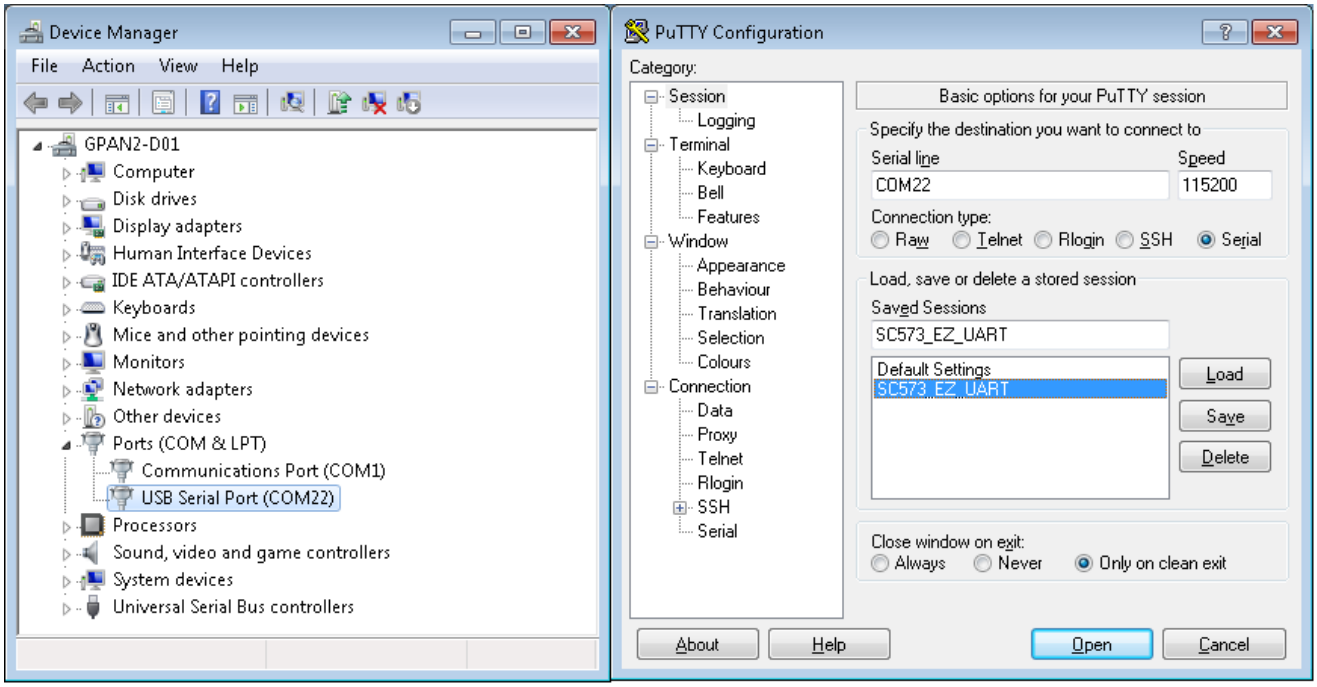
- In the **Project Explorer** right click on the **RTOSDemo_CCES_Core0** project and select the **Build Project** option from the menu

### 5.1.4 Run the Example

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

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

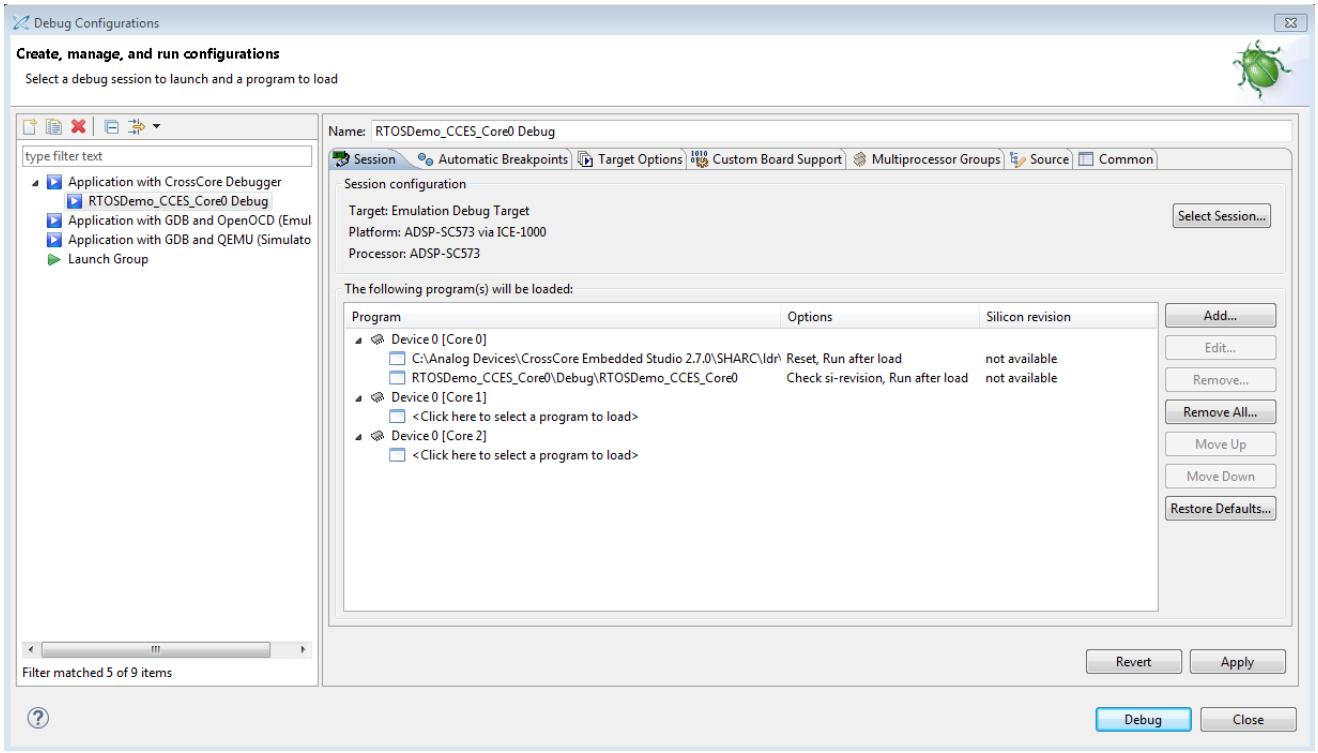
Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the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



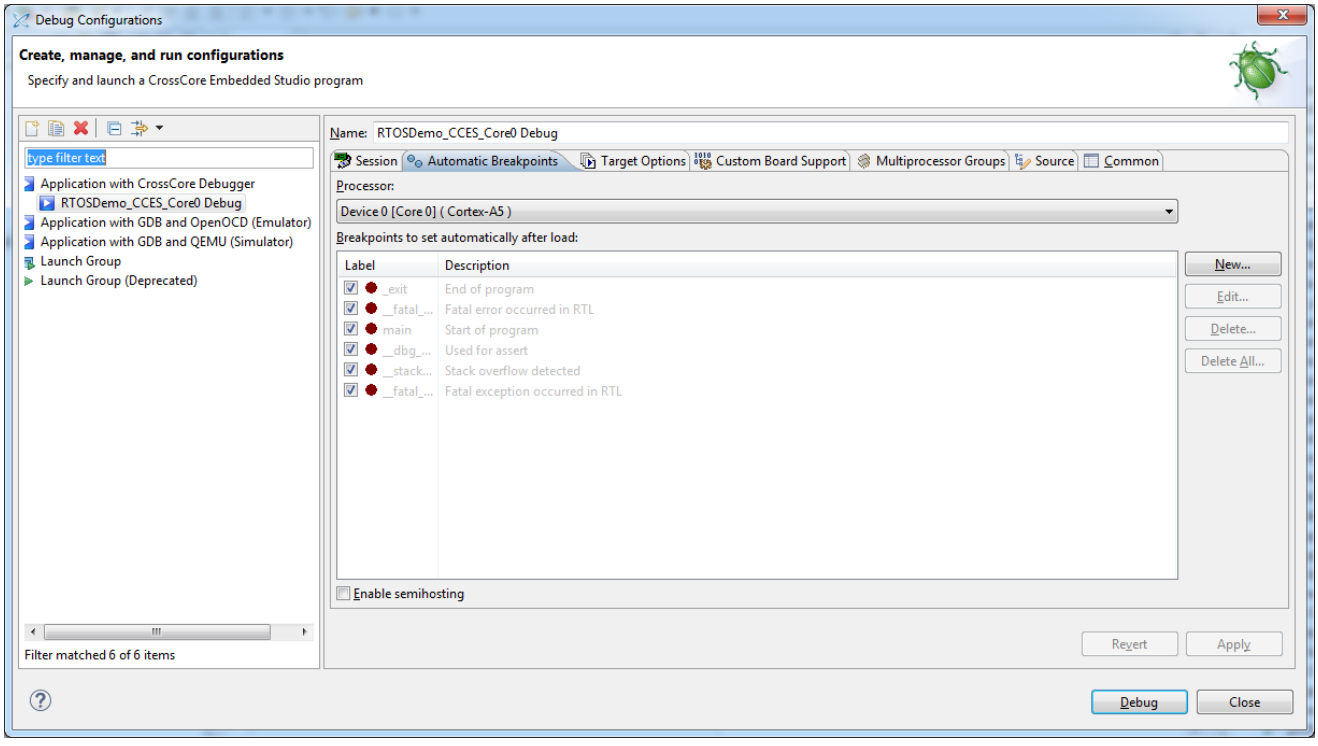

After this, follow the steps below to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_CCES_Core0** project and select the **Debug As** option from the menu

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

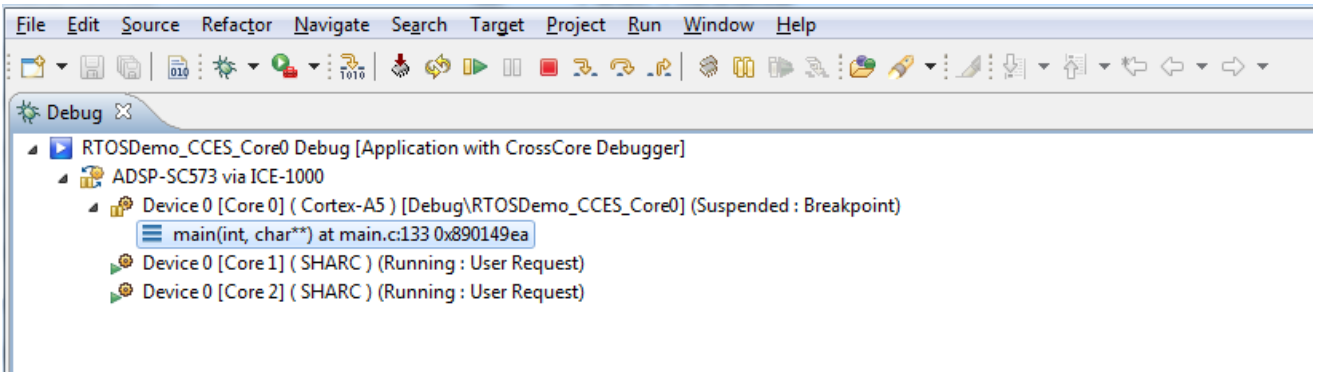



3. Disable the **semihosting** function in **Automatic Breakpoints**




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

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




### 5.1.5 Test Results

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

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

### 5.2.1 Overview

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

### 5.2.2 Environment Setup

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

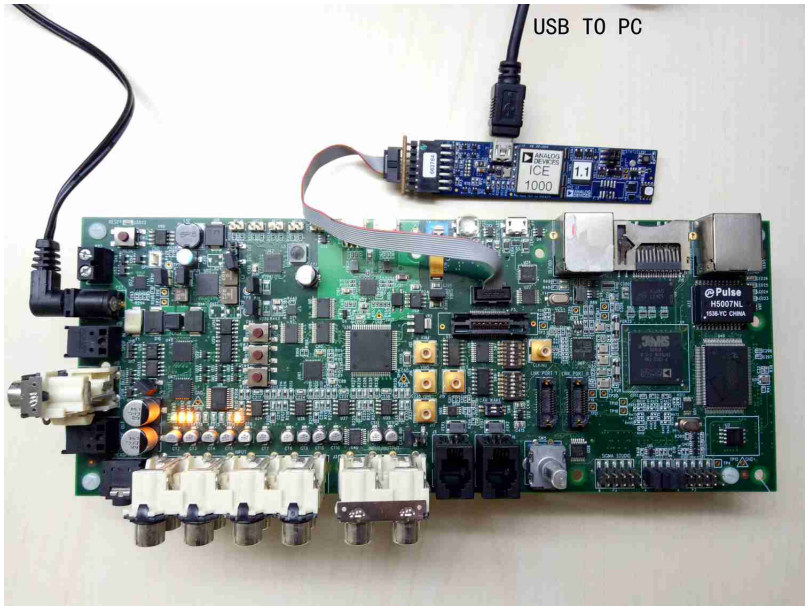
#### Software Requirement

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

#### Hardware Setup

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

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

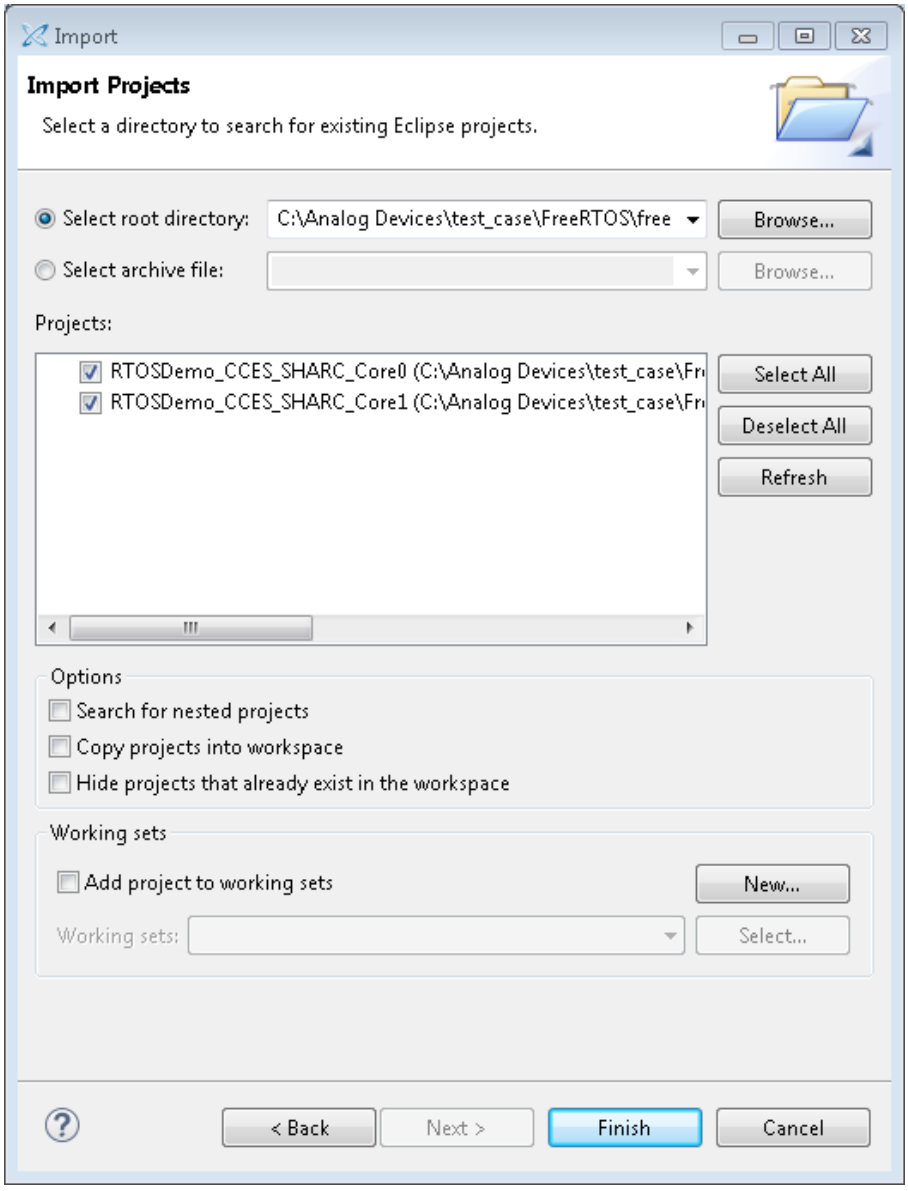



### 5.2.3 Build the Example

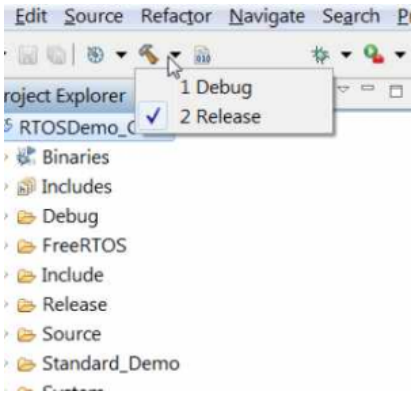
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the **File** menu and then select the **Import** option from the menu
- When the **Import** project window appears:
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC_ADSP_SC573_CCES** folder
- Click **Finish** to close the file browser dialog
- Two projects should appear in the **Project Explorer**




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




3. Build the project in CrossCore Embedded Studio:

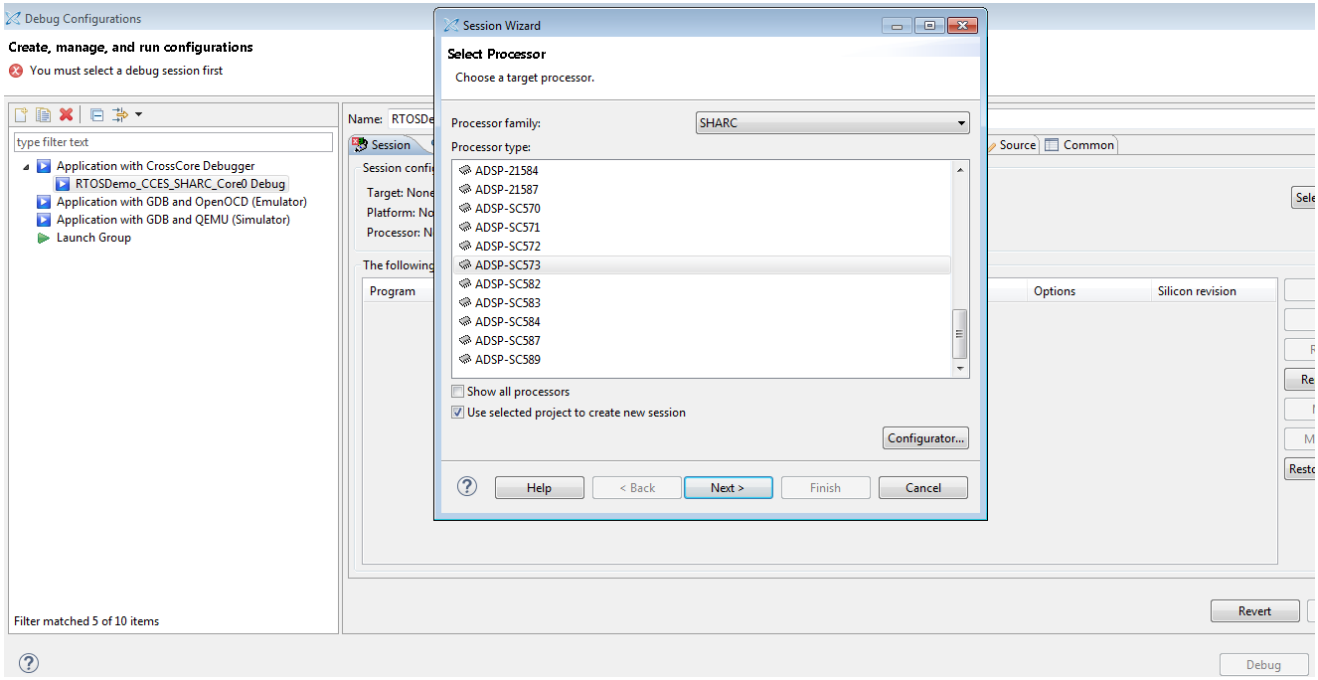
- In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_Core0** and **RTOSDemo_CCES_SHARC_Core1** project, then select the **Build Project** option from the menu

### 5.2.4 Run the Example

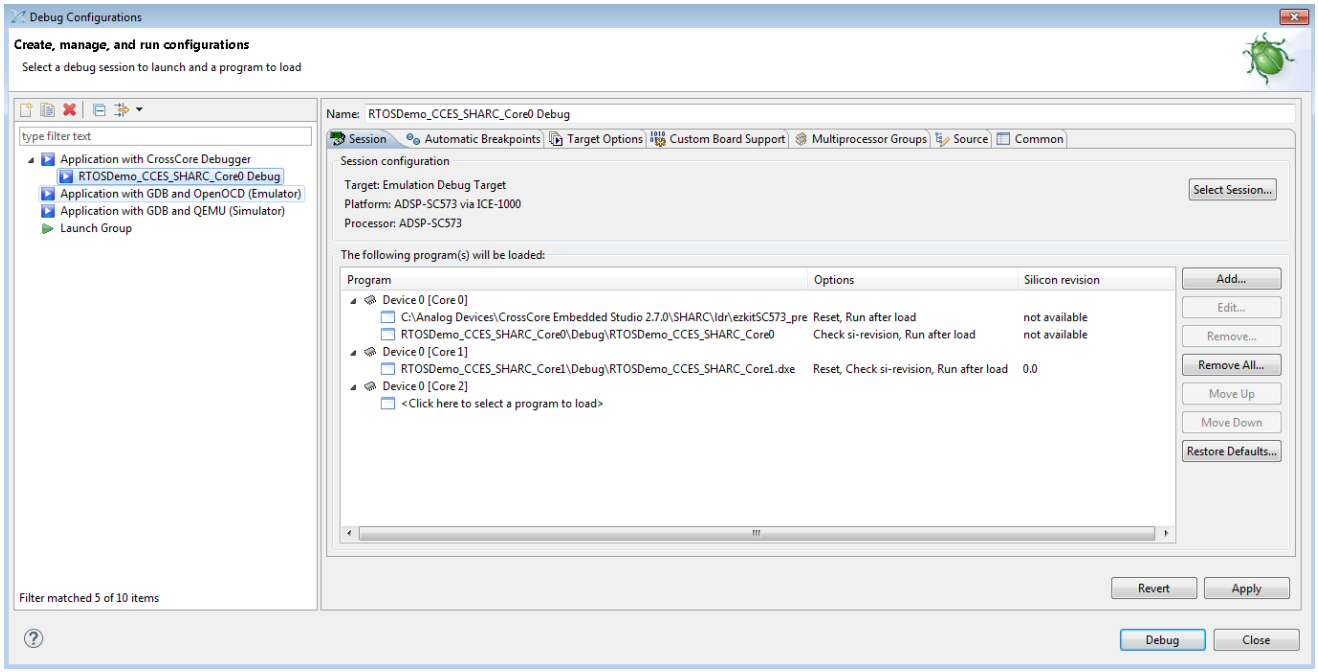
Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_Core0** project and select the **Debug As** option from the menu

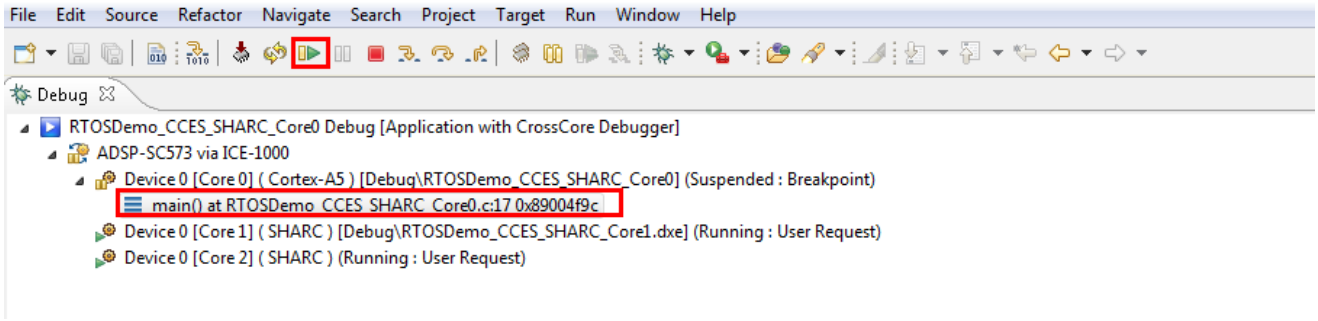
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board



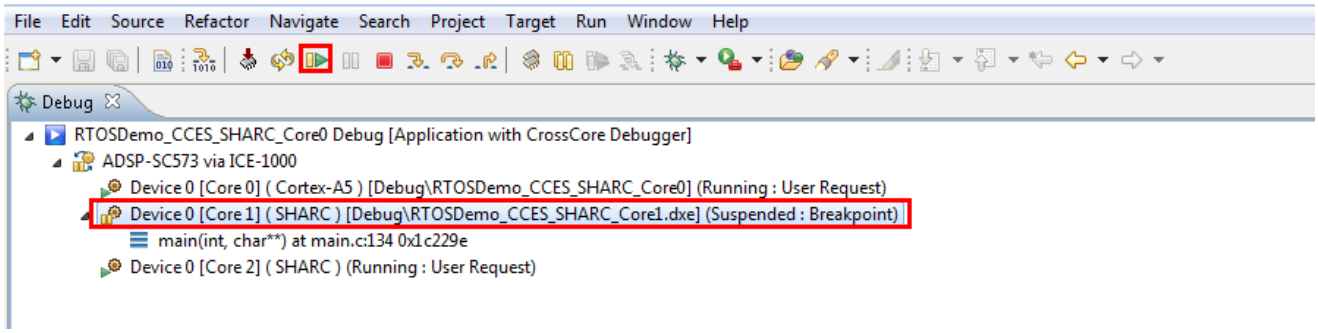

3. Click the **Debug** button to close the **Debug Configurations** window




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

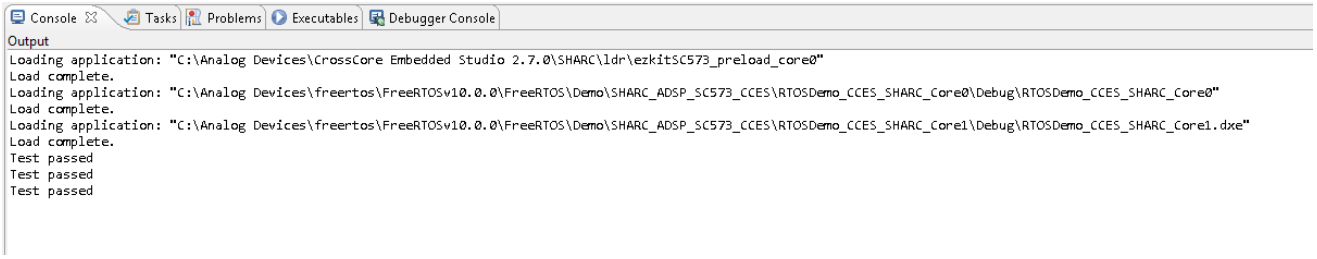



5. Then choose Core1 and keep to click **Run/Resume** button to start running Core1 application




### 5.2.5 Test Results

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


Console
Tasks
Problems
Executables
Debugger Console
Output
Loading application: "C:\Analog Devices\CrossCore Embedded Studio 2.7.0\SHARC\ldr\ezkitSC573_preload_core0"
Load complete.
Loading application: "C:\Analog Devices\freertos\FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC_ADSP_SC573_CCES\RTOSDemo_CCES_SHARC_Core0\Debug\RTOSDemo_CCES_SHARC_Core0"
Load complete.
Loading application: "C:\Analog Devices\freertos\FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC_ADSP_SC573_CCES\RTOSDemo_CCES_SHARC_Core1\Debug\RTOSDemo_CCES_SHARC_Core1.dxe"
Load complete.
Test passed
Test passed
Test passed

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

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

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

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

The tasks performed in the Analog Devices Basic Demo include:

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

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

### 6.1.1 Overview

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

### 6.1.2 Environment Setup

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

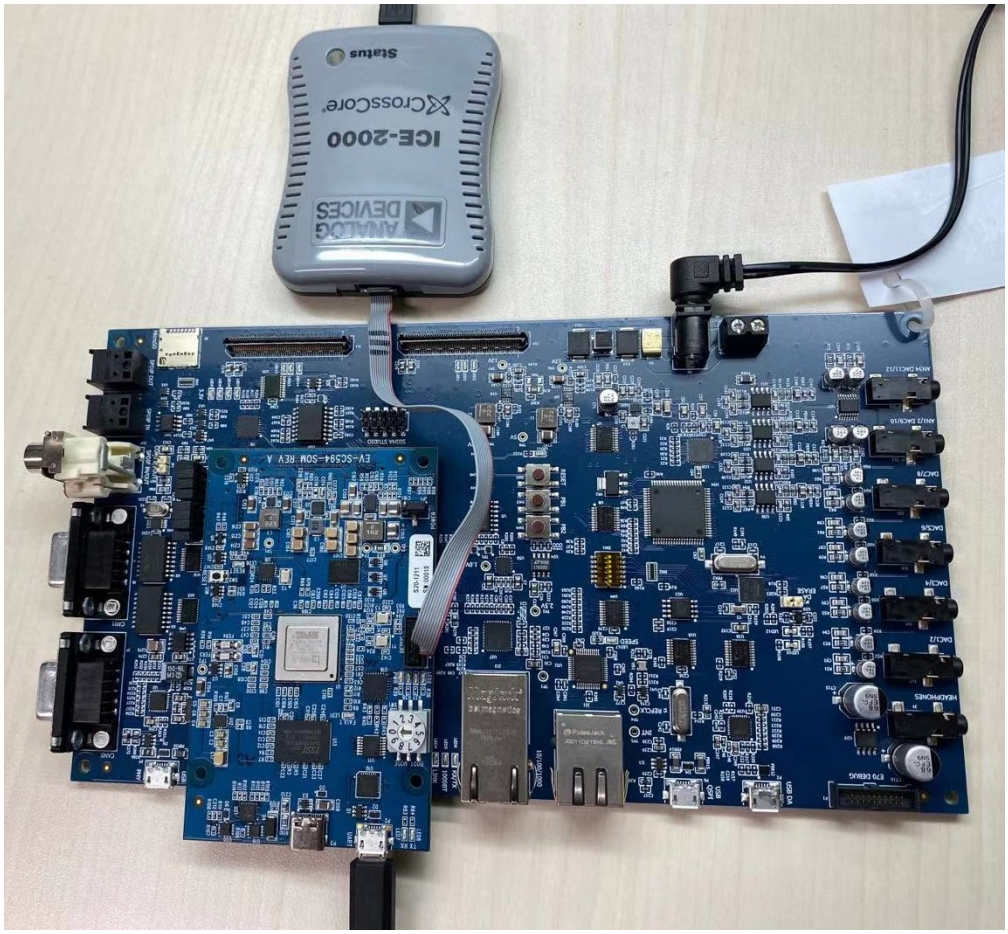
### Software Requirement

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

### Hardware Setup

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

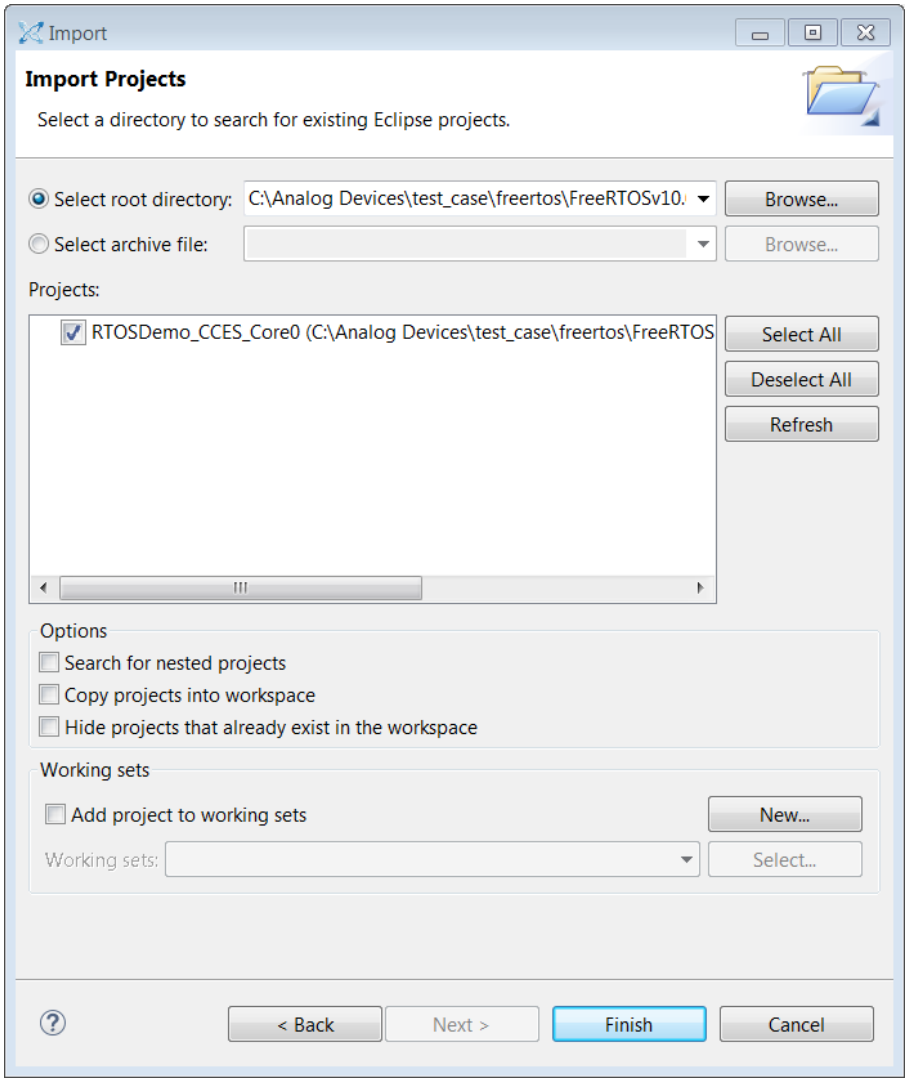
Connect the ICE1000 or ICE2000 emulator to **DEBUG** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below. A nd connect the **USB to UART** port of the EZ-Kit to the host PC with a USB cable as shown below:



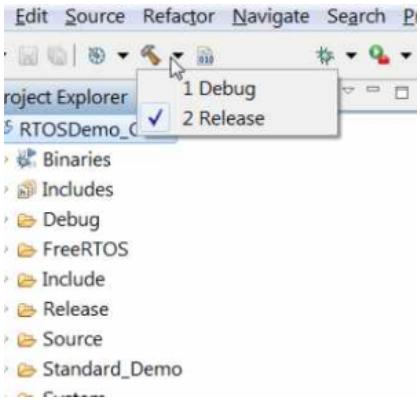

## 6.1.3 Build the Example

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

1. Import the FreeRTOS example into CrossCore Embedded Studio:
    - Select the **File** menu and then select the **Import** option from the menu
    - Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
    - Click the **Select root directory** radio button and then click the **Browse** button
    - Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0\FreeRTOS\Demo\CORTEX_A5_ADSP_SC594_CCES** folder
    - Click **Finish** to close the file browser dialog
    - A single project should appear in the **projects** pane of the **Import** window
    - Check the entry in the **projects** pane and click **Import**




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




3. Build the project in CrossCore Embedded Studio:

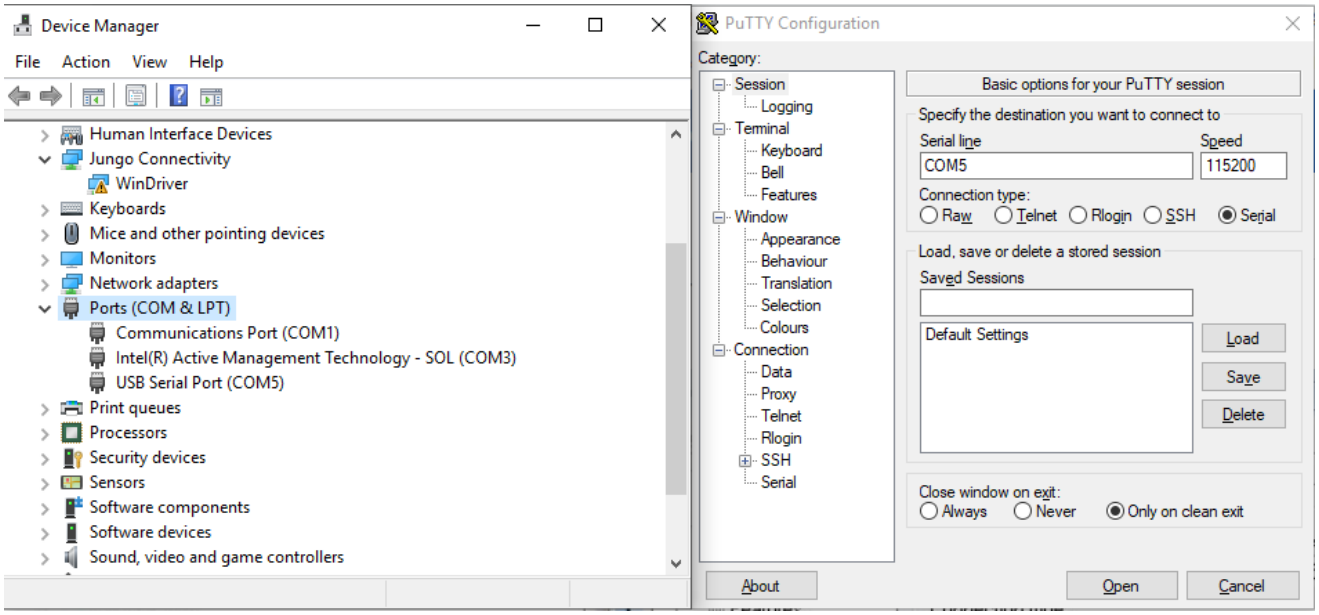
- In the **Project Explorer** right click on the **RTOSDemo_CCES_Core0** project and select the **Build Project** option from the menu

### 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 studio function call performed out with a thread will crash the application.
2. If you need to use other peripherals, you should take care not to change the power service clock rate (which is set in the UART I/O device).

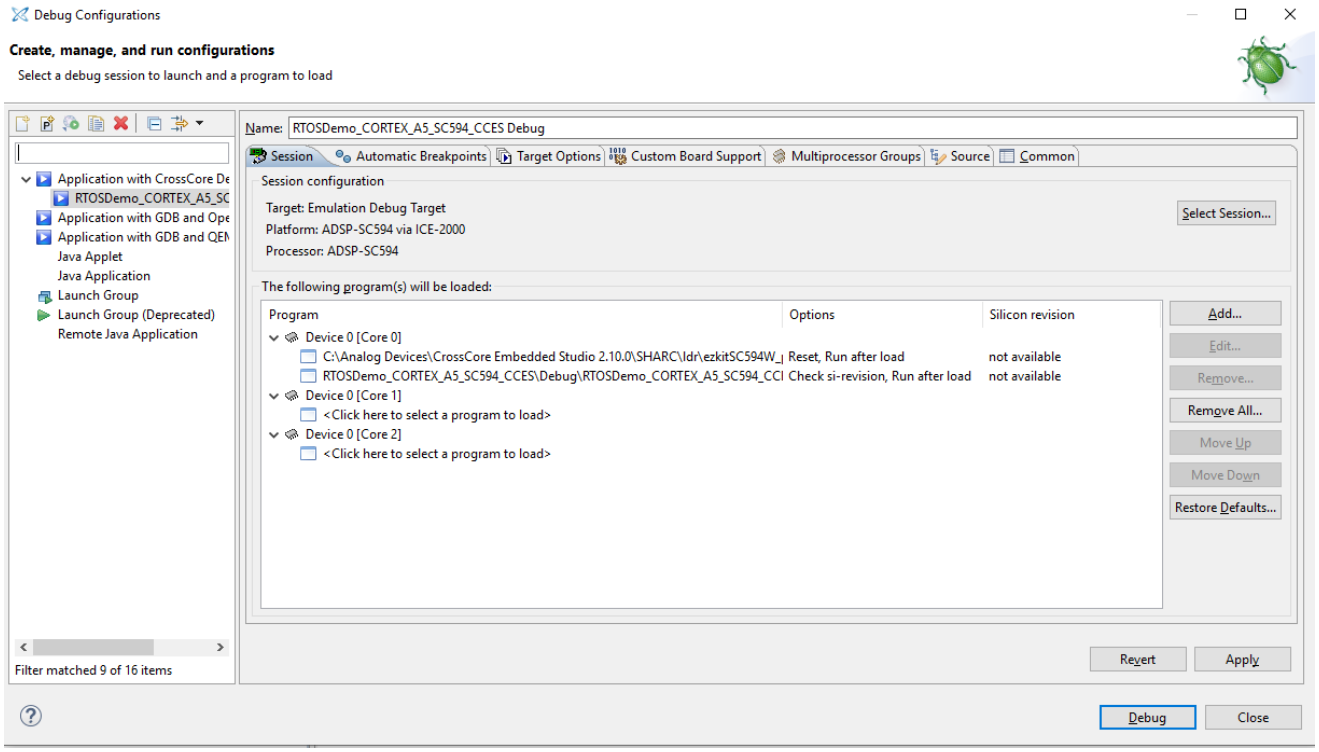
Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the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



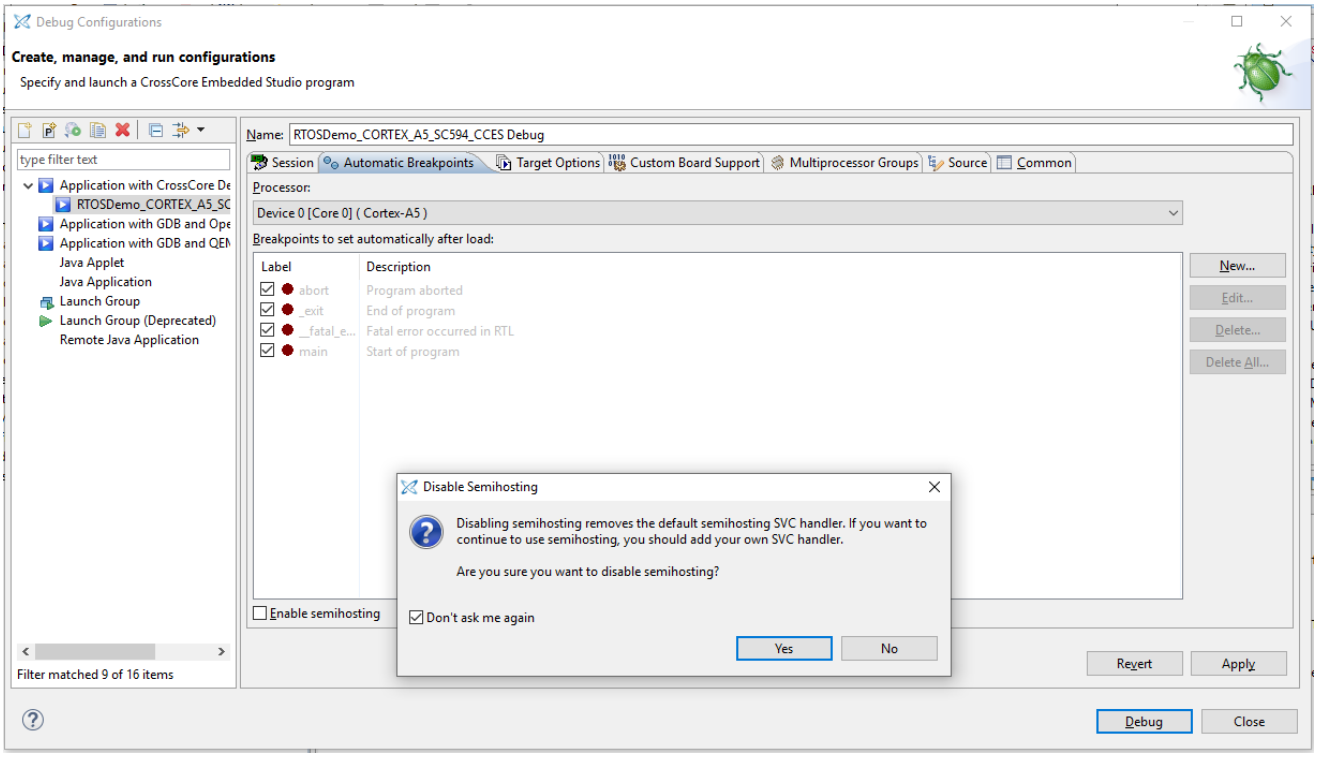

After this, follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_CCES_Core0** project and select the **Debug As** option from the menu

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




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




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

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

### 6.1.5 Test Results

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

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

### 6.2.1 Overview

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

### 6.2.2 Environment Setup

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

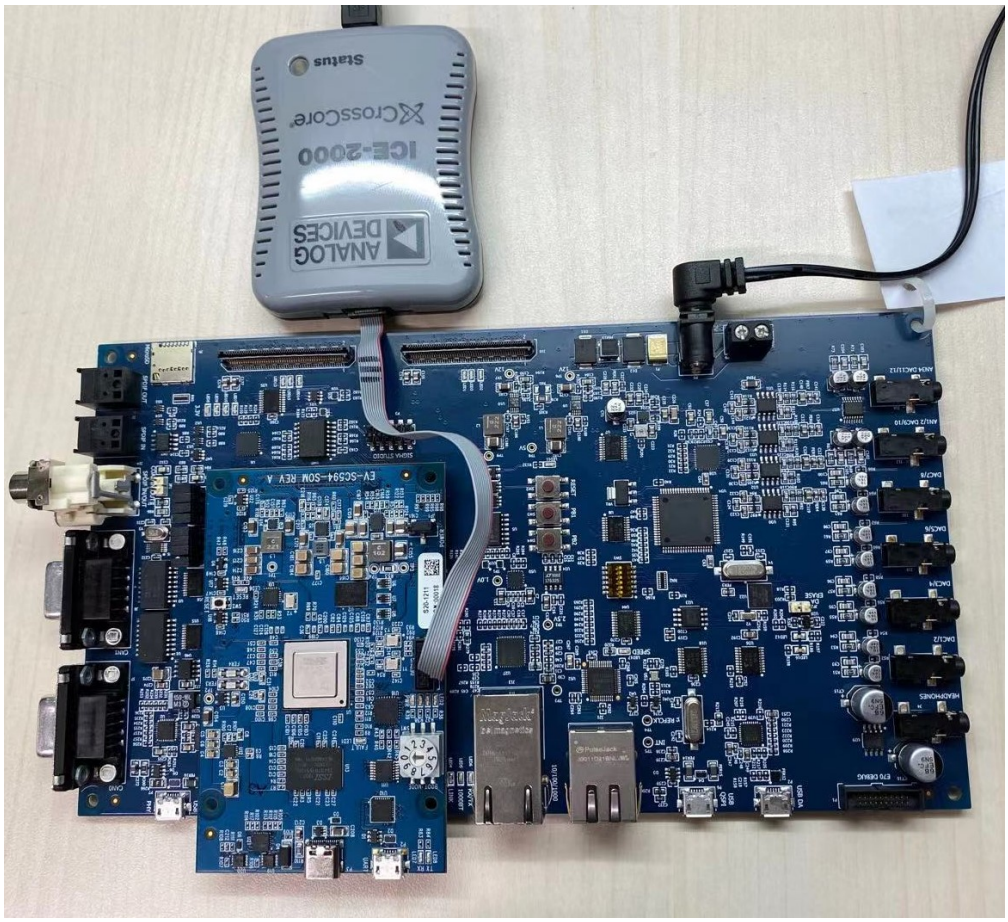
### Software Requirement

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

### Hardware Setup

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

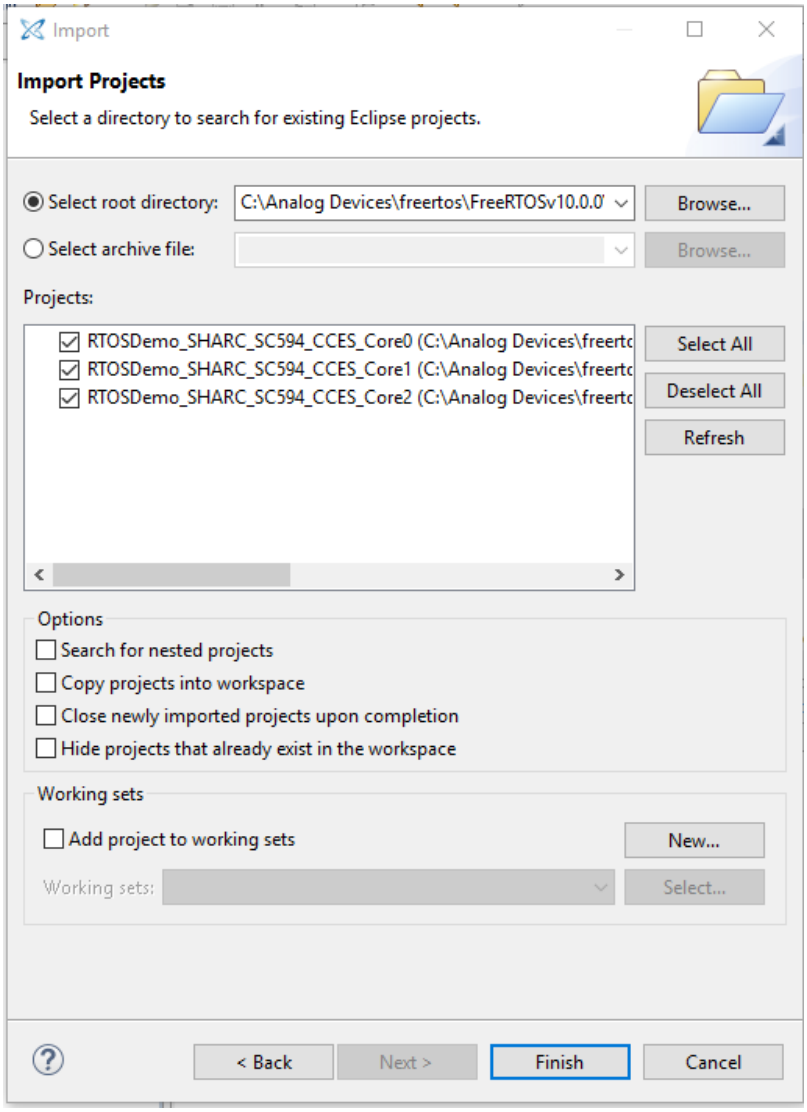
Connect the ICE1000 or ICE2000 emulator to **DEBUG** port of EZ-Kit and the host PC using USB cable and simultaneously connect the power supply with 12 volts as in the diagram below.



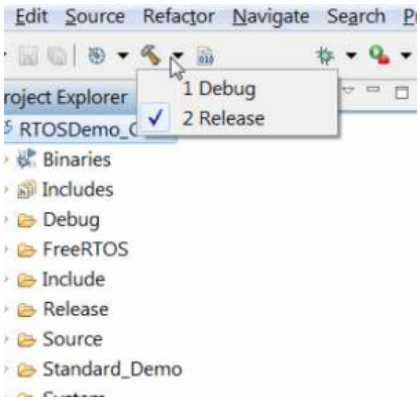

## 6.2.3 Build the Example

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

1. Import the FreeRTOS example into CrossCore Embedded Studio:
    - Select the **File** menu and then select the **Import** option from the menu
    - Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
    - Click the **Select root directory** radio button and then click the **Browse** button
    - Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0\FreeRTOS\Demo\SHARC_ADSP_SC594_CCES** folder
    - Click **Finish** to close the file browser dialog
    - Two projects should appear in the **Project Explorer**




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




3. Build the project in CrossCore Embedded Studio:

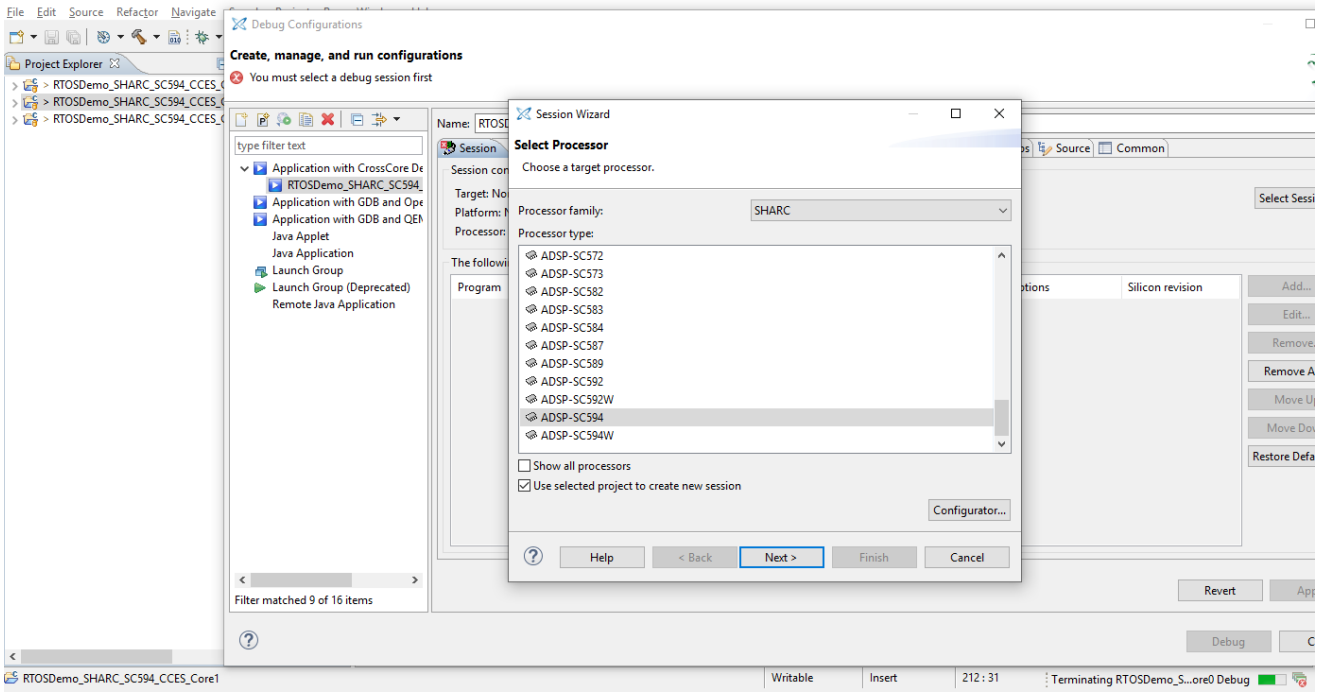
- In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_Core0** and **RTOSDemo_CCES_SHARC_Core1** and **RTOSDemo_CCES_SHARC_Core2** project, then select the **Build Project** option from the menu

### 6.2.4 Run the Example

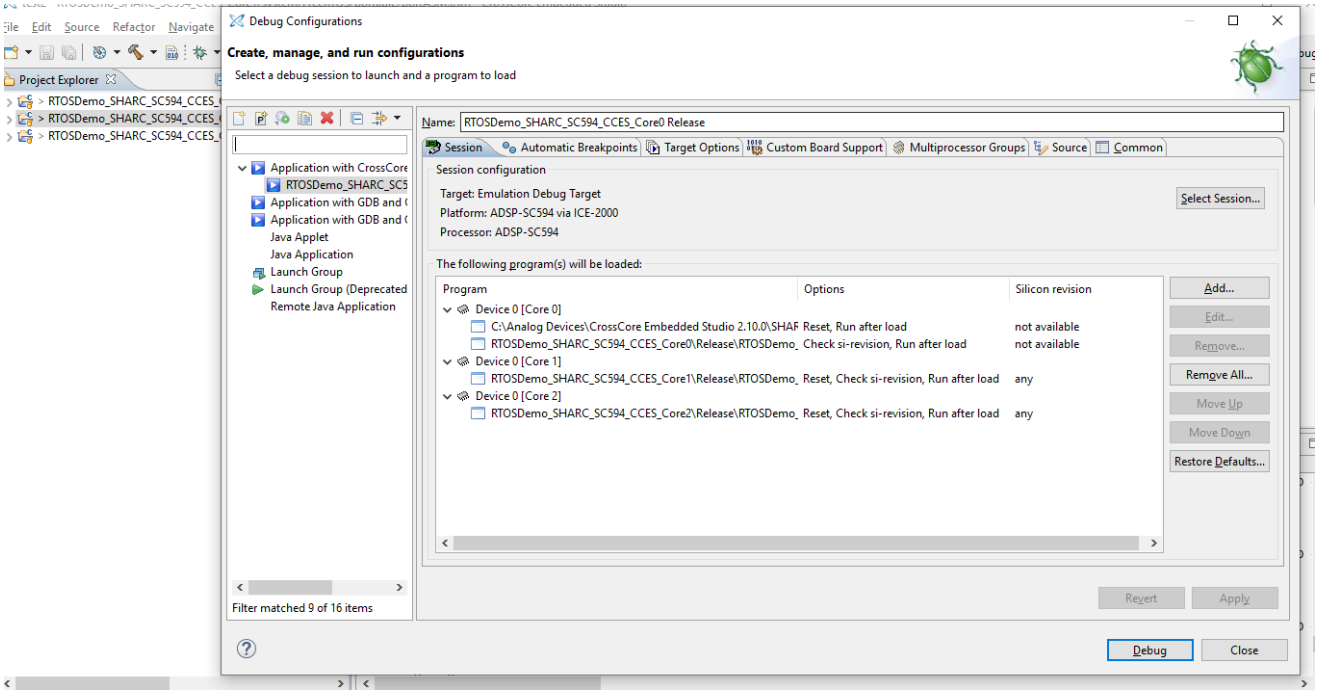
Follow below five steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_Core1** project and select the **Debug As** option from the menu

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



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




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

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

### 6.2.5 Test Results

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

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

### 6.3.1 Overview

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

### 6.3.2 Environment Setup

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

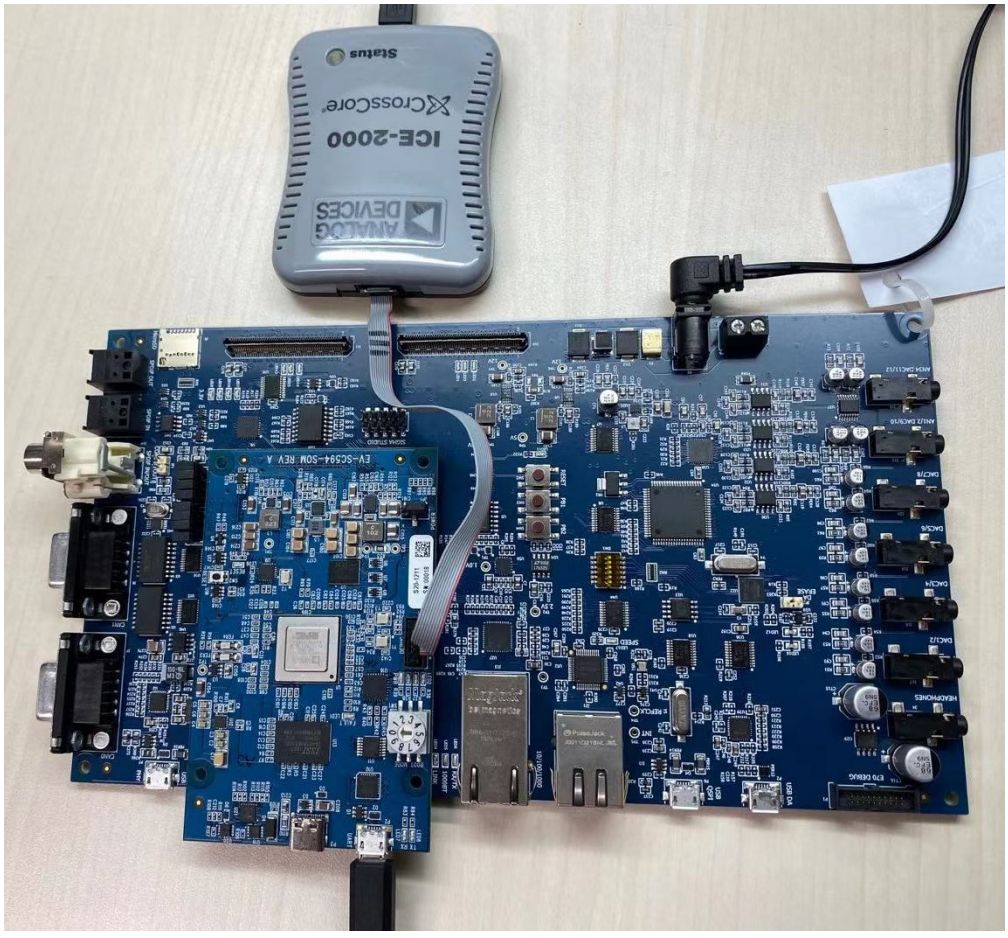
### Software Requirement

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

### Hardware Setup

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

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

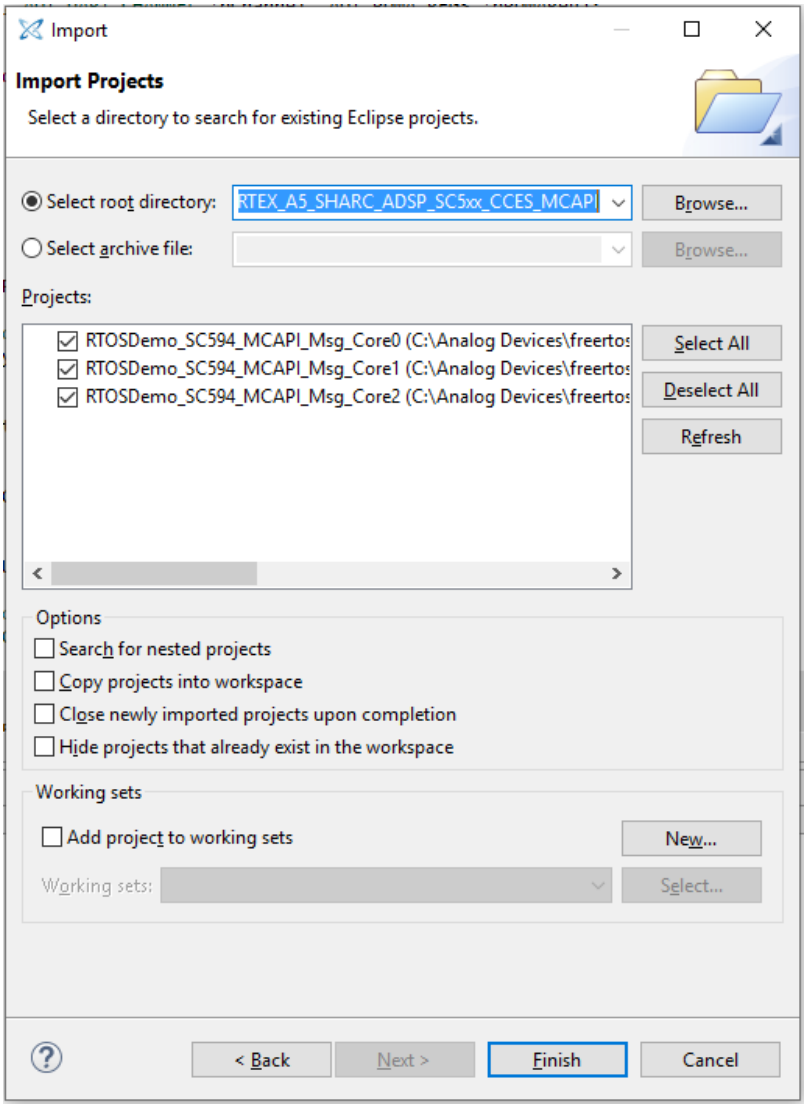



## 6.3.3 Build the Example

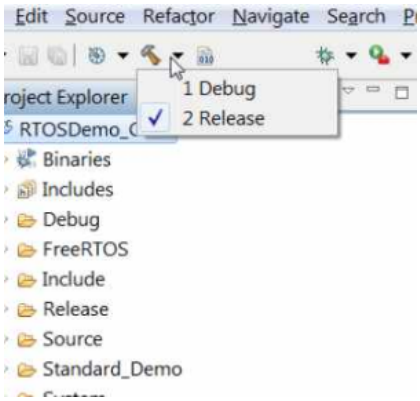
Before you run the FreeRTOS example in CrossCore Embedded Studio, follow below three steps to import and build it.

1. Import the FreeRTOS example into CrossCore Embedded Studio:

- Select the **File** menu and then select the **Import** option from the menu
- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0 \FreeRTOS\Demo\CORTEX_A5_SHARC_ADSP_SC5xx_CCES_MCAPI** folder
- Click **Finish** to close the file browser dialog
- A single project should appear in the **projects** pane of the **Import** window
- Check the entry in the **projects** pane and click **Import**




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




3. Build the project in CrossCore Embedded Studio:

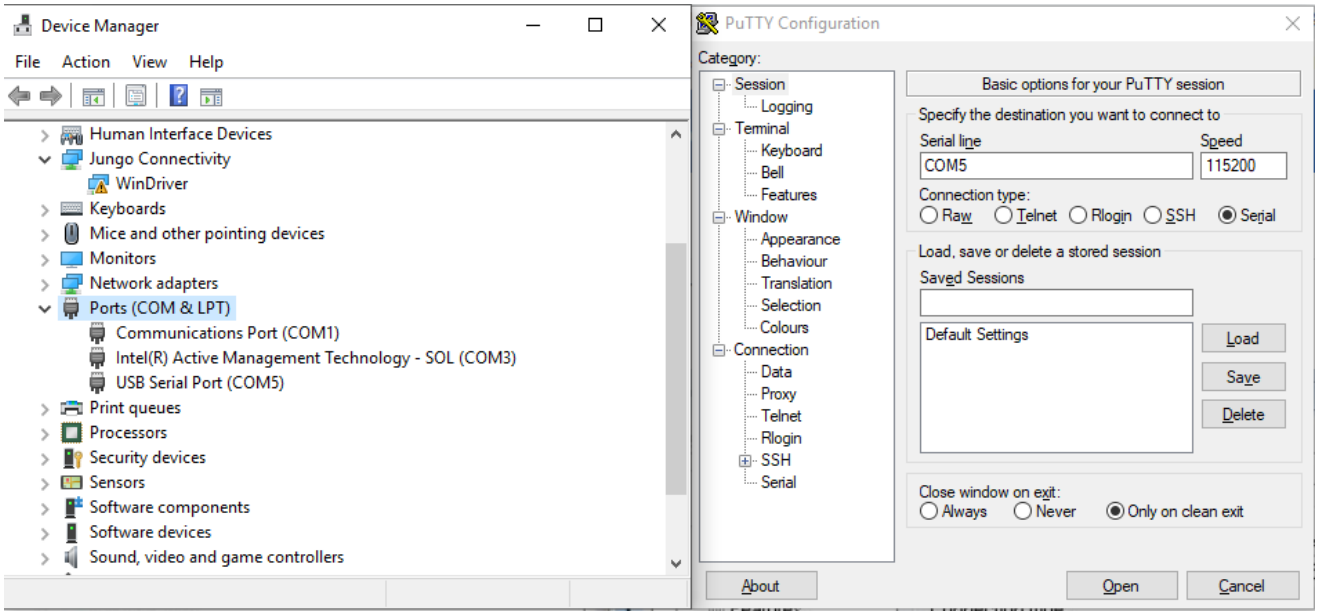
- In the **Project Explorer** right click on the RTOSDemo_CCES_Core0 project and select the **Build Project** option from the menu

### 6.3.4 Run the Example

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

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

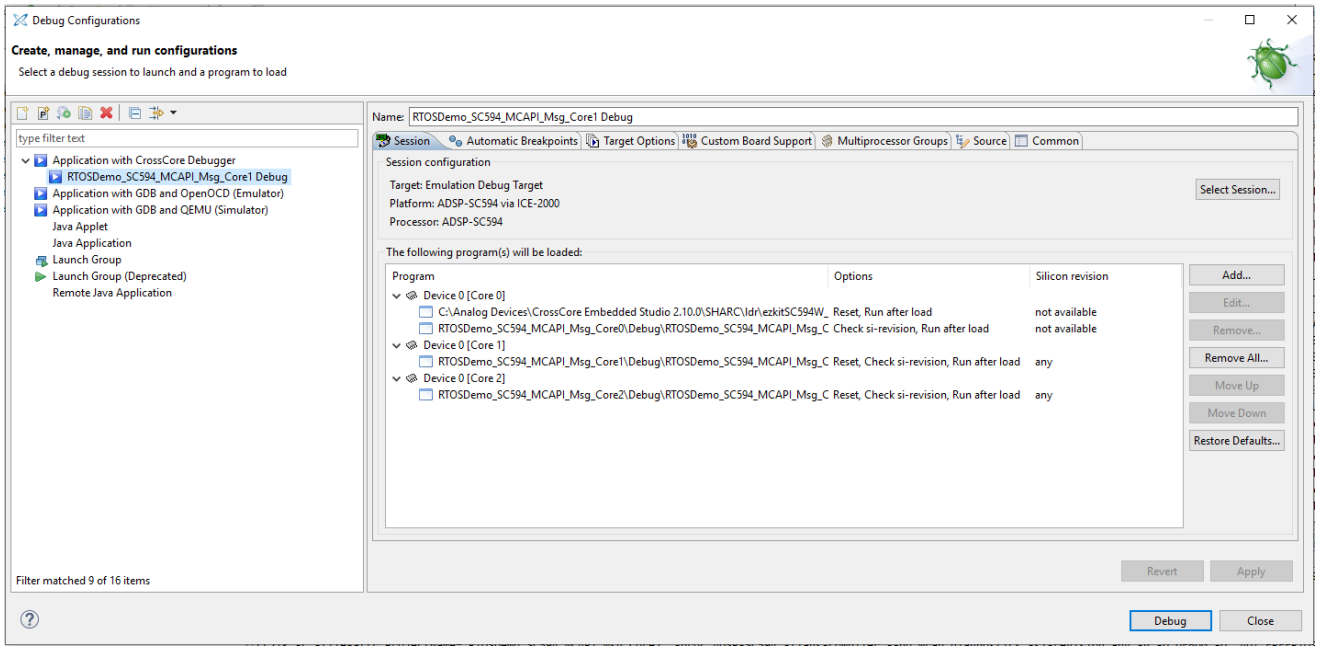
Before running the example, you need to setup the serial terminal of your choice to read the Cortex core output from the **UART to USB connection** on the HOST PC. The easiest way to determine the correct USB device is to view the **Ports** entry in the Windows Device Manager. From here identify the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



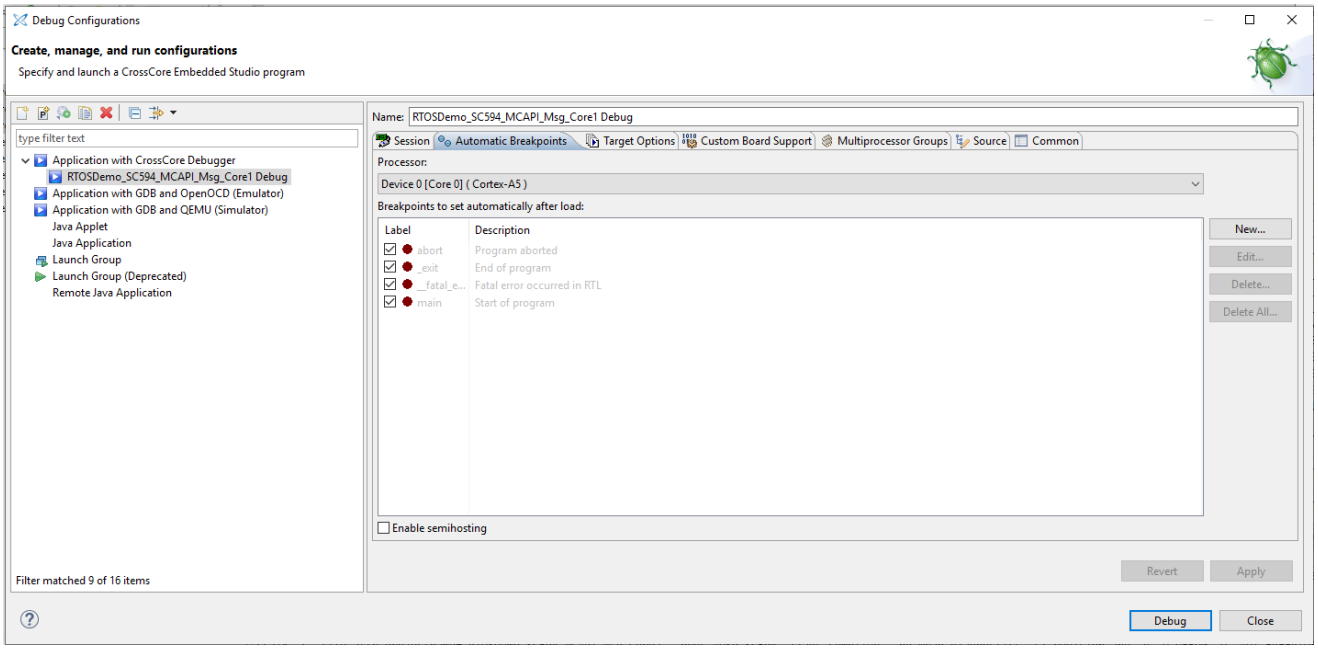

After this, follow below five steps to do debug configuration, download and run the built binary on the target board.

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




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




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

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

### 6.3.5 Test Results

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

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

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

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

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

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

The tasks performed in the Analog Devices Basic Demo include:

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

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

### 7.1.1 Overview

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

### 7.1.2 Environment Setup

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

#### Software Requirement

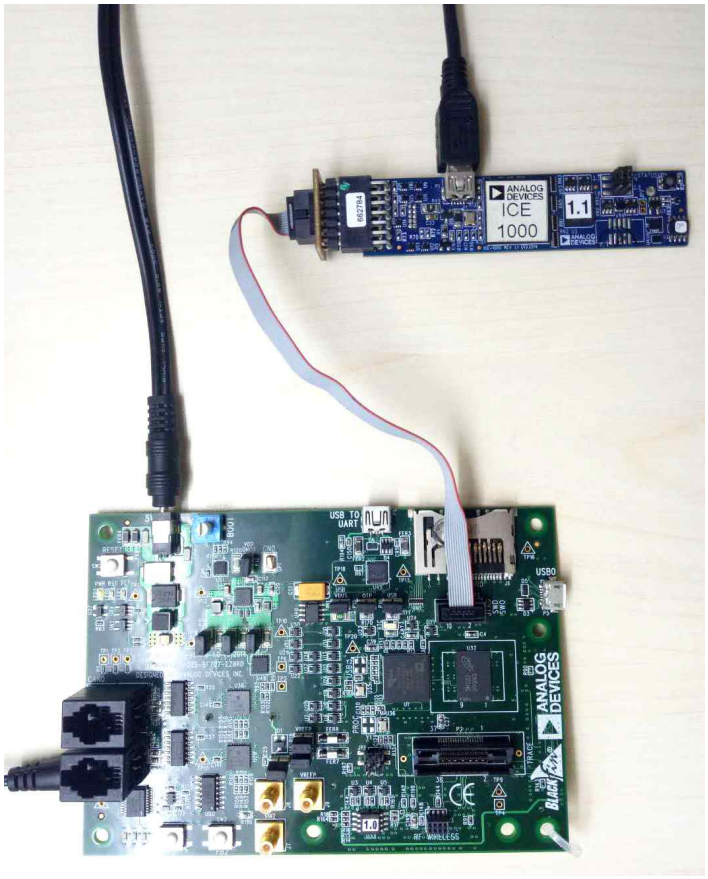
- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio

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

### Hardware Setup

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

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



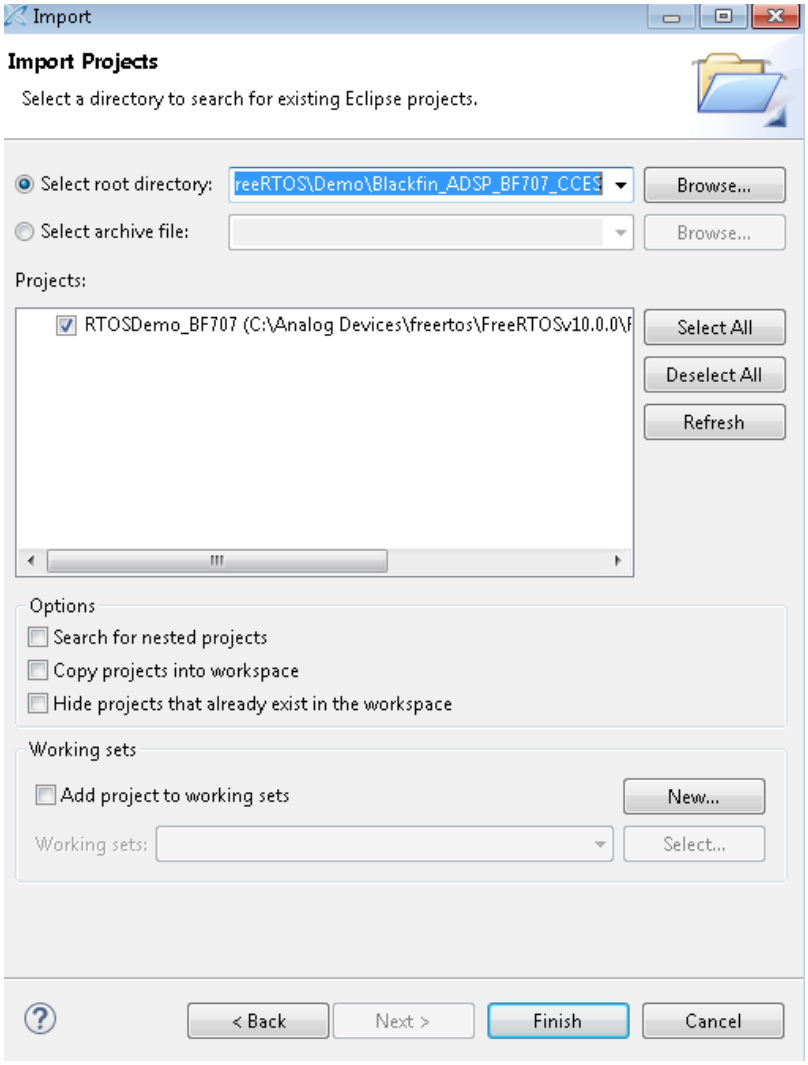

## 7.1.3 Build the Example

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

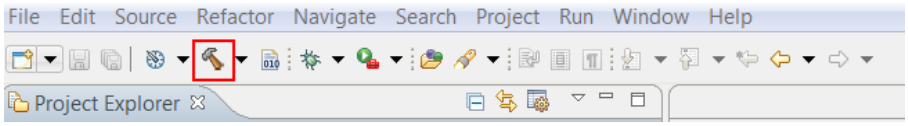
1. Select the **File** menu and then select the **Import** option.

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

- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **\FreeRTOSv10.0.0\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**




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




3. Build the project in CrossCore Embedded Studio

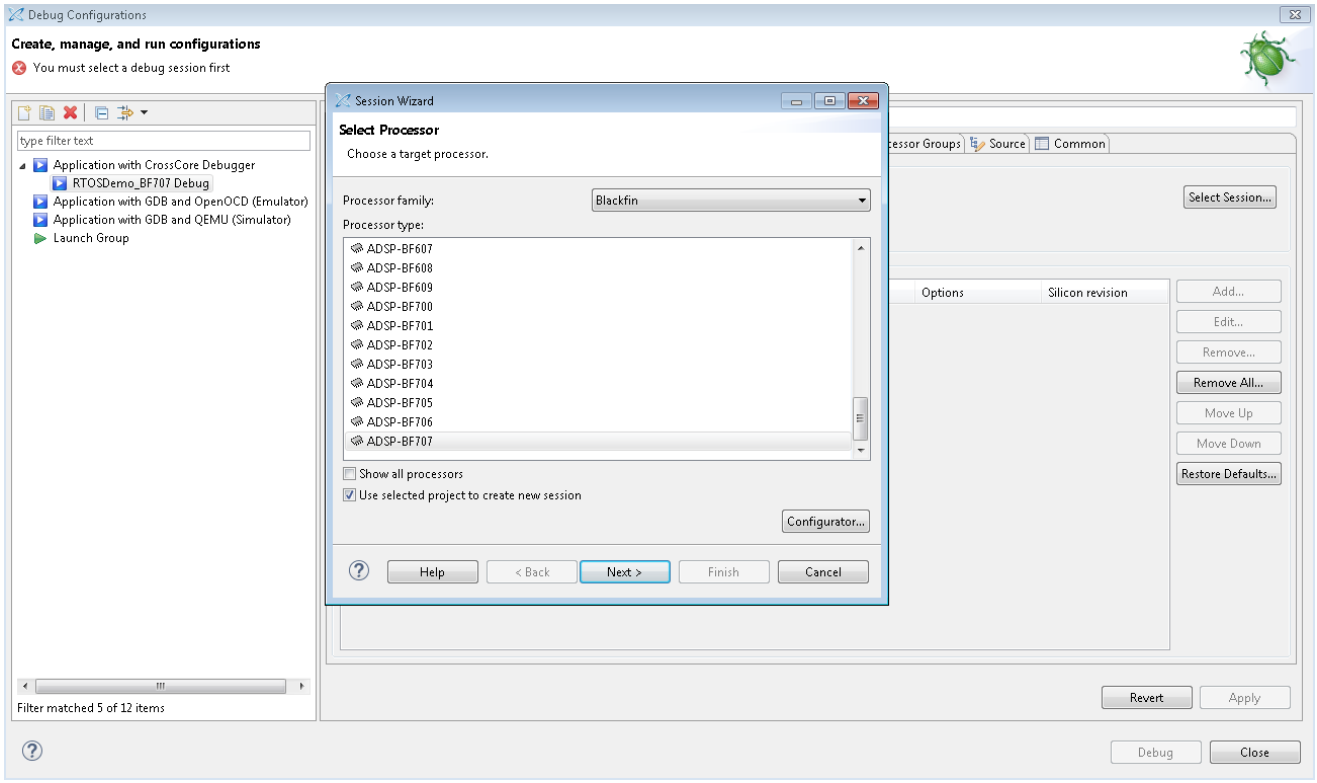
- In the **Project Explorer** right click on the **RTOSDemo_BF707** project and select the **Build Project** option from the menu

### 7.1.4 Run the Example

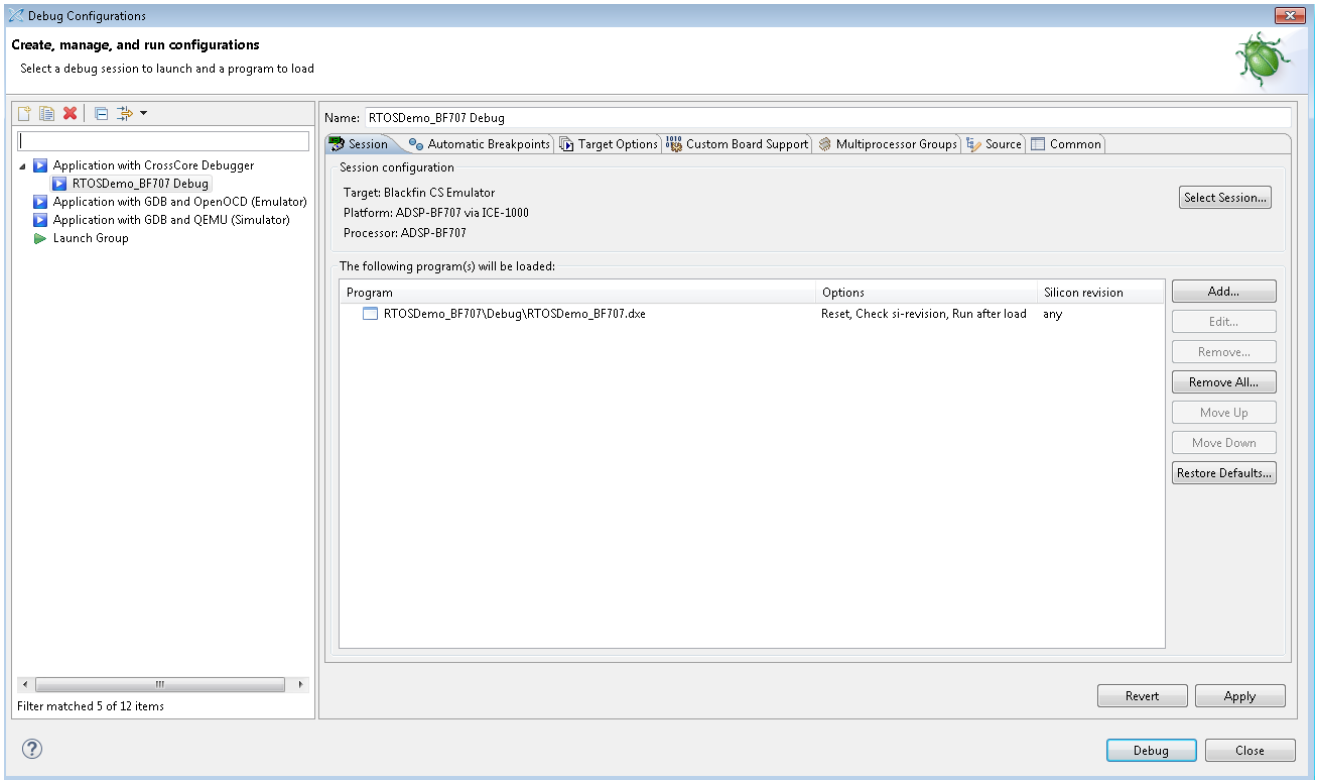
Follow below four steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_BF707** project and select the **Debug As** option from the menu

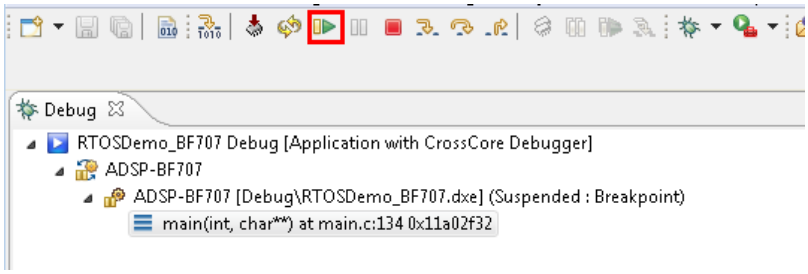
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator and target board




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

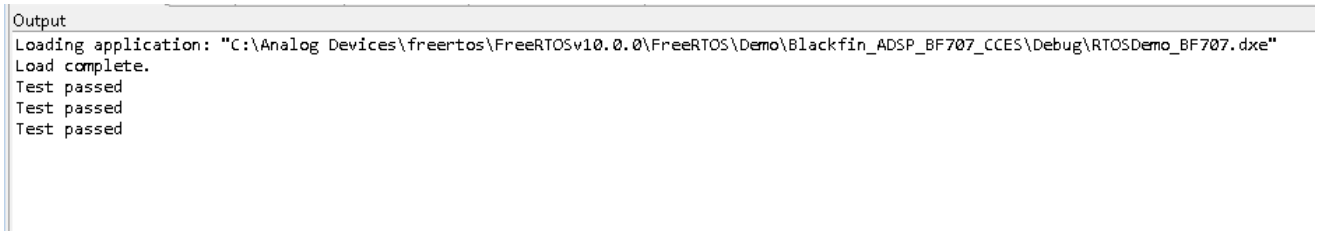



4. Click the **Run/Resume** button to start running your application




## 7.1.5 Test Results

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



# 8 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

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

### 8.1.1 Overview

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

### 8.1.2 Environment Setup

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

#### Software Requirement

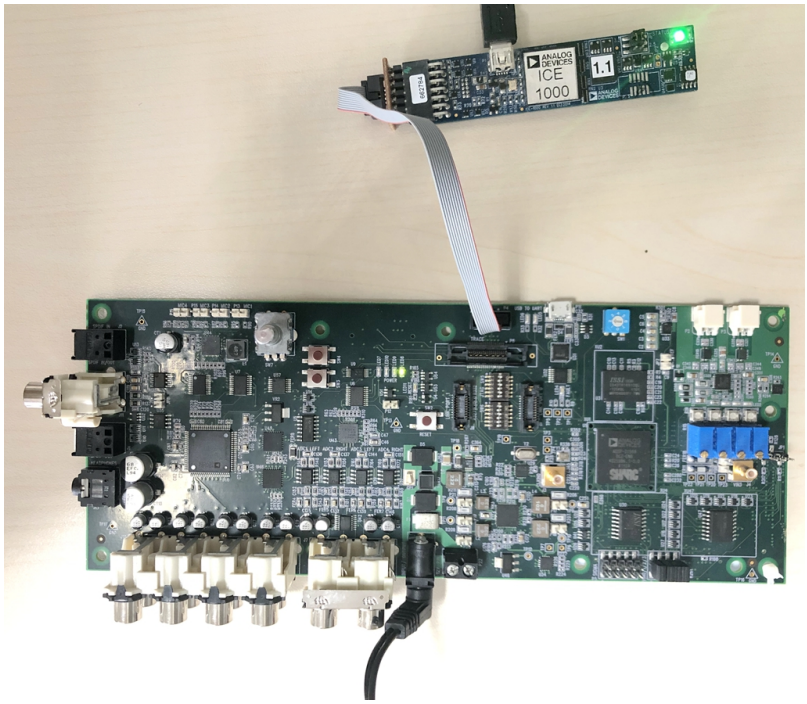
- Analog Devices CrossCore Embedded Studio. For more information please refer to Software environment set up for CrossCore Embedded Studio

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

### Hardware Setup

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

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



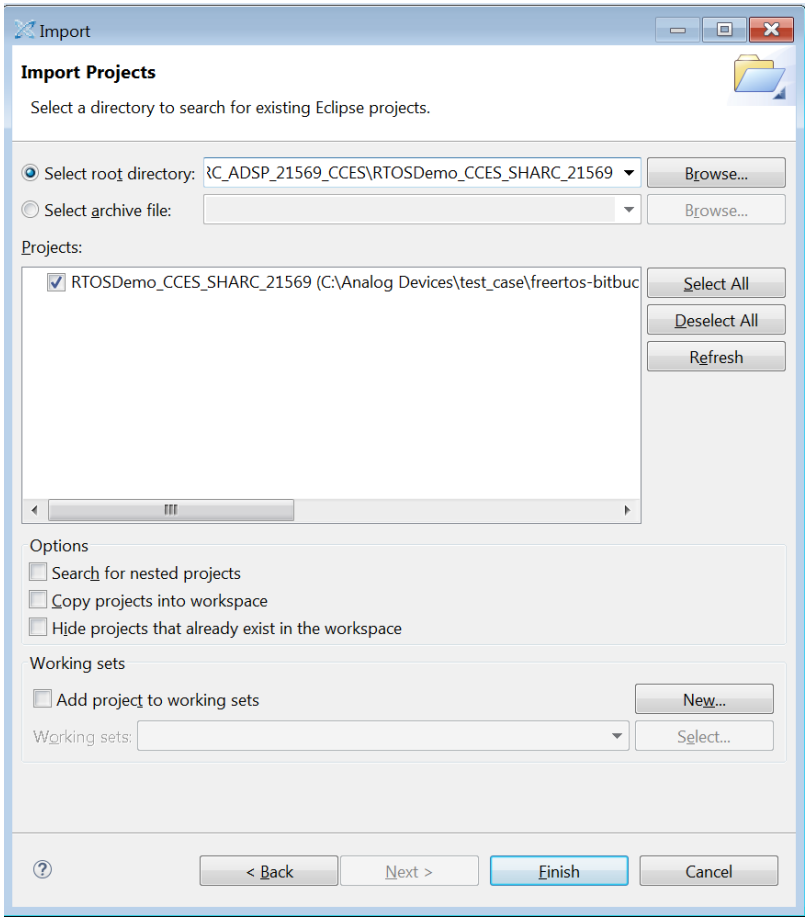

## 8.1.3 Build the Example

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

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

- Click on the **General** folder, then click on the **Existing Projects into Workspace** entry, and click **Next**
- Click the **Select root directory** radio button and then click the **Browse** button
- Browse the root folder where you previously installed the FreeRTOS product and then browse down into the **FreeRTOSv10.0.0 \FreeRTOS\Demo\SHARC_ADSP_21569_CCES\RTOSDemo_CCES_SHARC_21569** folder
- Click **OK** to close the file browser dialog

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




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


3. Build the project in CrossCore Embedded Studio

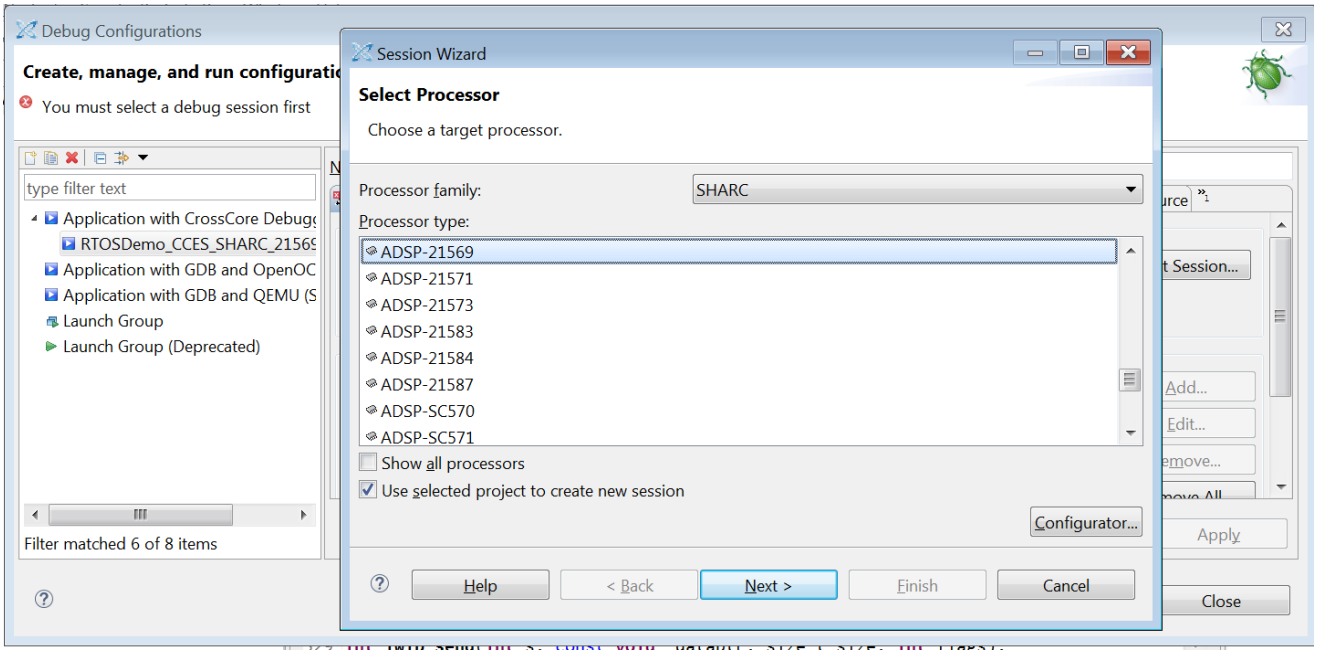
- In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_21569** project and select the **Build Project** option from the menu

### 8.1.4 Run the Example

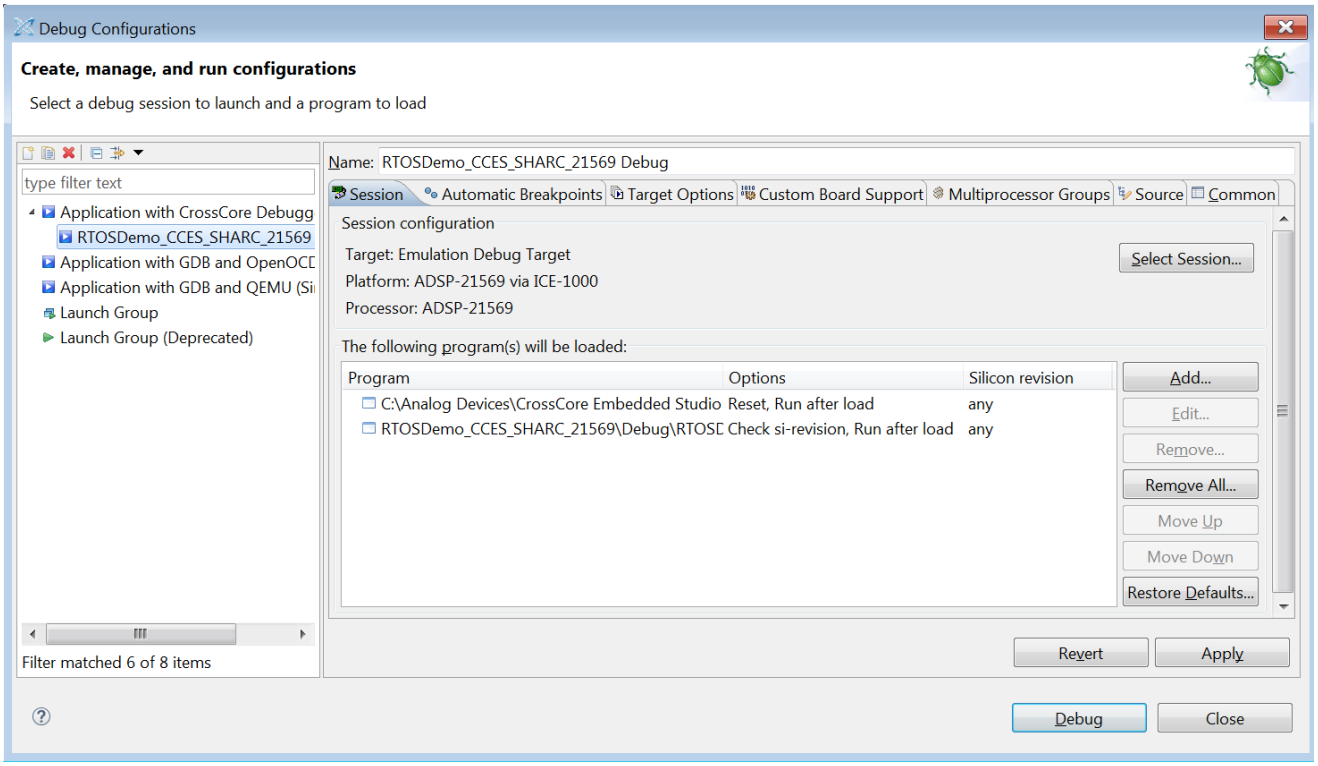
Follow below four steps to do debug configuration, download and run the built binary on the target board.

1. In the **Project Explorer** right click on the **RTOSDemo_CCES_SHARC_21569** project and select the **Debug As** option from the menu

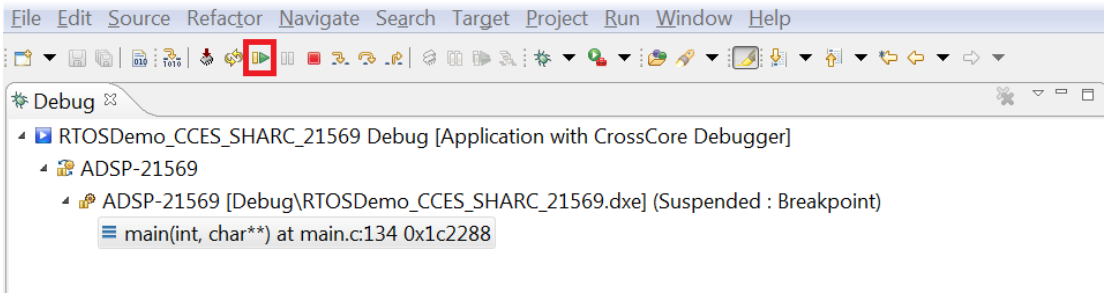
2. From the popup menu select **Debug Configurations** option to create a new debug configuration that matches your emulator(ICE1000 or ICE2000) and target board




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

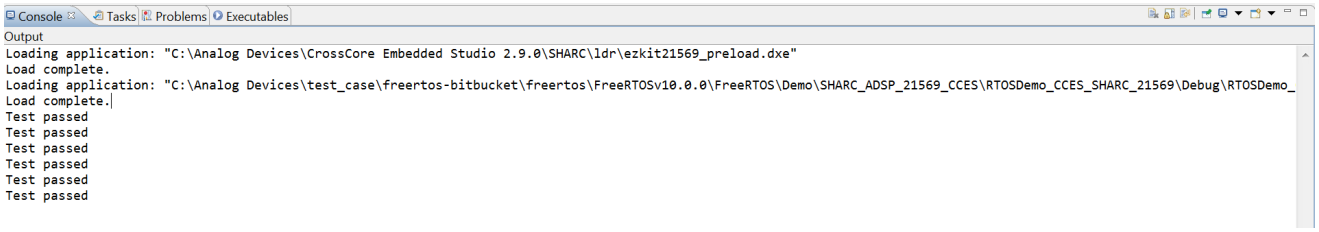



4. Click the **Run/Resume** button to start running your application




### 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 the LEDs on the EZ-Kit begin to flash. **Test Passed** will be printed if the all tests passed.



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

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

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

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

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

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

1. Ensure that the pre-processor macro __ADI_FREERTOS is defined for all assembler,C/C++ and linker operations.
   This requires the pre-processor macro to defined in three separate locations in the project settings.
2. Ensure that the pre-built libdrv library is not linked into the application.
   a. For Cortex-A projects this is controlled by adding the following option to the Settings >Tool Settings > CrossCore ARM Bare Metal C Linker > Additional Options settings: -specs=PATH_TO_FREERTOS\FreeRTOS\FreeRTOSv10.0.0 \FreeRTOSSource\portable\CCES\ARM_CA5\freertos.specs where PATH_TO_FREERTOS is replaced with the path to the installation of your FreeRTOS product.
   b. For SHARC+ and Blackfin projects this is controlled by checking the Settings > Tool Settings > CrossCore Blackfin/SHARC Linker > Libraries > Omit device driver library checkbox
3. Enable the source based version of the required services and drivers:
   a. Double click the system.svc file in the Project Explorer
   b. Click the Add button in the System Configuration Overview
   c. Browse the list of Device Drivers and System Services to add new components to the project

4. Build the project

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

# 10 Appendix A: FreeRTOS Performance

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

### Timer Cycles

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

Benchmark data is available for the following EZ-Kits:

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

### Spaces

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

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

- ADSP-BF707 EZ-Kit

The following projects are executed to gather the benchmark data:

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

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

### ADSP-21569 SHARC Core Performance Metrics

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

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

### ADSP-21569 SHARC Core Sizing Metrics

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

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

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

### 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 |
| | xQueueSend(task waiting, context switch to pending task) | 1309 |

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

### ADSP-SC589 Cortex-A Core Sizing Metrics

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

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

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

### ADSP-SC589 SHARC+ Core Performance Metrics

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

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

### ADSP-SC589 SHARC+ Core Sizing Metrics

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

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

## 10.4 ADZS-BF707 Benchmark Data

### ADZS-BF707 Performance Metrics

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

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

### ADZS-BF707 Sizing Metrics

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

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