My First Altera® SoC FPGA Exercise Manual (Atlas-SoC / DE10-Nano board version)

Ver.20.1

February 2025 Rev. 4

Contents

Before Reading This Manual

The contents of this manual are current as of February 2025.

Some of the software, hardware, and operating procedures described in this manual are common even if they are not the specified versions or devices, but some of them may not be common.

Symbols in Documentation

ⓘ Info	Provides supplementary information.
Ⓟ Point	Important points are included.
🗎 Reference	Reference materials and sites are introduced to deepen your understanding.
⚠ Note	Although not discussed in detail in this document, the information and knowledge required is provided.
⊘ Prohibited	Notes and what not to do are provided.

Notations in sentences

Underline	Click to jump to another chapter in the document or to an external site.
Bold italic	Indicates the characters displayed on menus and windows when operating on the screen.
xxxxxxx↵	Indicates the command string to be entered.
Shaded	Indicates the tool to be used.

1. Overview

In this exercise, you will learn how to develop Cyclone® V SoC hardware and software using the Cyclone® V SoC FPGA evaluation kit DE0-Nano-SoC Kit/Atlas-SoC Kit (Atlas-SoC board below) or DE10-Nano Kit (DE10 Nano board below).

By completing this exercise, you will learn the basic operations of the Quartus® Prime development software, the development environment for Altera® SoC FPGA, the Platform Designer system configuration tool (formerly known as Qsys System Integration Tool), and the SoC FPGA Embedded Development Suite (SoC EDS) software development environment.

This exercise consists of the following 4 parts:

- Lab 1: Hardware Exercise
- Lab 2: Software Exercise (1)
- Lab 3: Software Exercise (2)
- Lab 4: Linux Application Exercise (optional exercise)

Lab 1 uses Quartus® Prime to configure the hardware including the Arm® processor and design a simple SoC system.

Lab 2 uses the SoC EDS tool to generate the 28 nm generation boot loader, Preloader.

Lab 3 uses Arm® Development Studio for Intel® SoC FPGA Edition (Arm® DS below) to develop software and debug bare metal applications.

Lab 4 uses an SD card image to run Linux on the SoC device and then runs and debugs the application using Arm® DS.

ⓘ **Info:**

Lab 4 is an optional exercise that will not be available at our company SoC Startup Trial Seminar due to time constraints.

ⓘ **Info:**

Starting with version 20.1, the SoC EDS standard integrated development environment tool is Arm® Development Studio (Arm® DS). The previous product, Arm® Development Studio 5 (DS-5™), is available for pre-version 19.4 environments.

1-1. Requirements

The following software is used in this exercise.

- Quartus® Prime Standard Edition v20.1 (Lite Edition is also available)

  You will also need to register Cyclone® V as Device Data.
  Download and installation instructions are available at:
  How to Download Intel® Quartus® Prime Development Software and Questa* - Intel® FPGA Edition
  How to Install Intel® Quartus® Prime Development Software and Questa* - Intel® FPGA Edition

- SoC FPGA Embedded Development Suite Standard Edition v20.1 (SoC EDS)

  For installation instructions, refer to the following website:
  How to install SoC FPGA Embedded Development Suite (SoC EDS) ver. 20.1

- Exercise Data (SoC-Trial_Seminer_Lab_data_atlas_de10nano_v20.1_r2.exe)

  When you double-click the exercise data .exe file, it will be expanded to the following location by default:

  ```
  C:¥lab¥soc_lab¥cv_soc_lab
  ```

  This document assumes that the exercise data is expanded to the above location.

- Host PC OS: Windows® 10 Enterprise

  This exercise uses Windows® 10 Enterprise (version 1803) to verify operation.

⚠ **Notes:**

When using SoC EDS v20.1std in a Windows® 10 environment, the tools **bsp-create-settings** and **sopc-create-header-files** must be resolved before running.
After setting up SoC EDS, you should also review the following reference sites:

Reference:

Macnica Altera FPGA Insights "Workaround for bsp-create-settings execution error in SoC EDS environment"

Reference:

Macnica Altera FPGA Insights "Workaround for sopc-create-header-files execution error in SoC EDS environment"

* This exercise uses these two tools, so both of these should be addressed.

2. Board Setup

This section describes the board setup required to complete exercises 1, 2, and 3.

2-1. Board Layout

The following diagram illustrates the layout of the Atlas-SoC board used in this exercise.
The DE10 Nano board is basically the same.

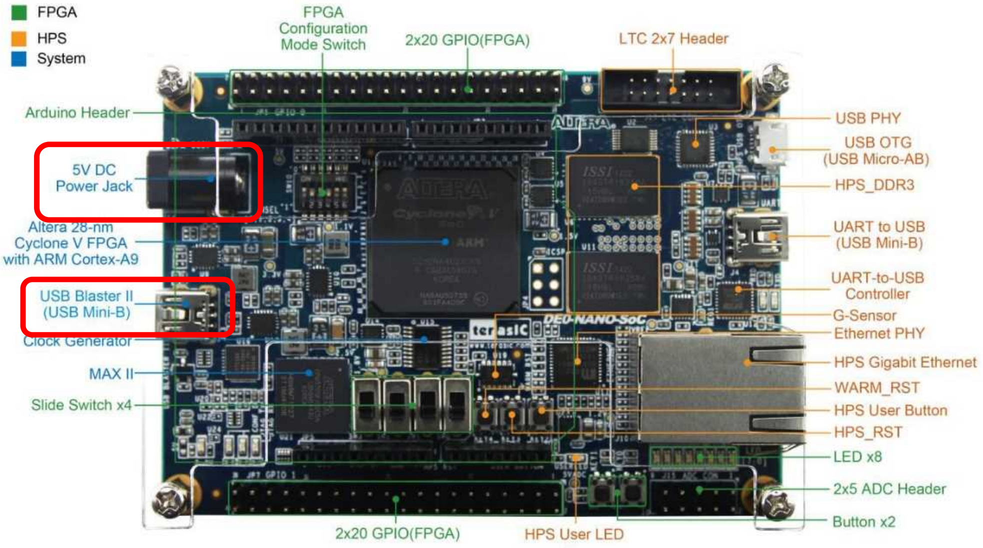


Figure 2-1. Atlas-SoC board layout diagram

2-2. Power and cable connections

Connect the AC adaptor and cables as follows.

- Connect the power (AC adaptor) to the DC input (J14).
- Use the Mini USB cable to connect the work PC to the onboard USB-Blaster™ II connector (J13).

2-3. SW10 settings

Make sure that the SW10 (MSEL setting switch) is set as follows.
This setting puts the FPGA in FPPx32 mode.

Table 2-1. SW10 Settings

Board Reference	Signal Name	Settings
SW10. 1	MSEL0	ON (“0”)
SW10. 2	MSEL1	OFF (“1”)
SW10. 3	MSEL2	ON (“0”)
SW10. 4	MSEL3	OFF (“1”)
SW10. 5	MSEL4	ON (“0”)
SW10. 6	N/A	N/A

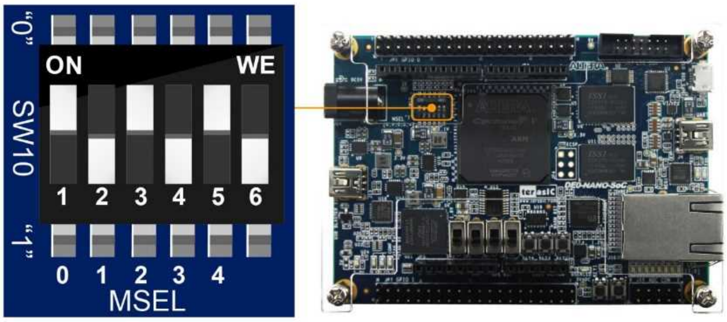


Figure 2-2. Jumper settings

3. Lab 1: Hardware Exercise

In this section, you will use Quartus® Prime and Platform Designer to design the hardware including the following Arm® processors.

Altera® SoC FPGA is not limited to Cyclone® V, but uses a tool called Platform Designer included in Quartus® Prime to configure the system. Platform Designer provides a set of components that can be implemented on the FPGA side, including Hard Processor System (HPS) blocks, and allows you to optimize resources by implementing only the desired components. In addition, since the created system can be easily ported to other devices if the peripherals are supported, you can use the system itself as a design asset.

In this exercise, several components and clock source components are already implemented in the Platform Designer system to shorten the exercise time. To do this, you will add an HPS block (the blue block in the bold box) and connect the existing components. You will perform the following exercises:

Exercise:

- Add an HPS component to an existing Platform Designer system
- Configure the HPS interface and other parameters
- Connect existing components to HPS
- Generate the Platform Designer system

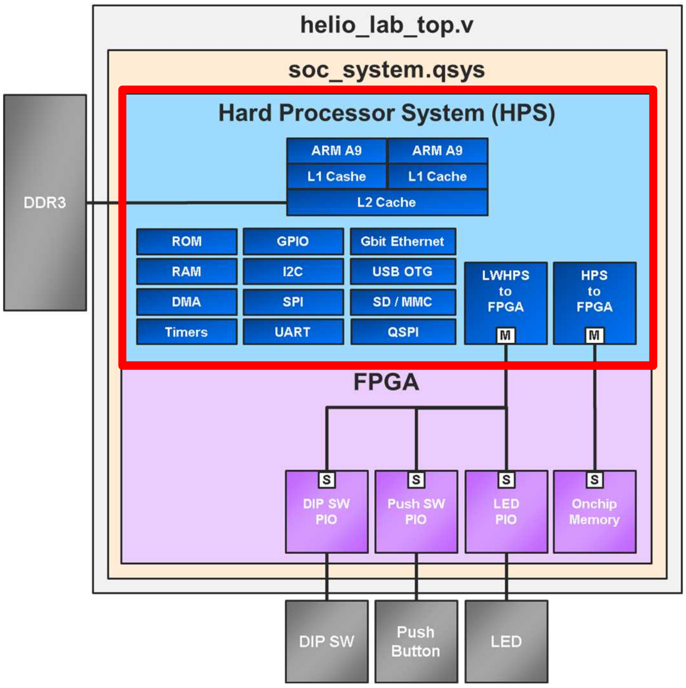


Figure 3-1. Block diagram of the SoC system designed in Lab 1

3-1. Step 1: Hardware Exercise Open the design project

As you proceed with the exercises, carefully read all the instructions in each step of this manual.

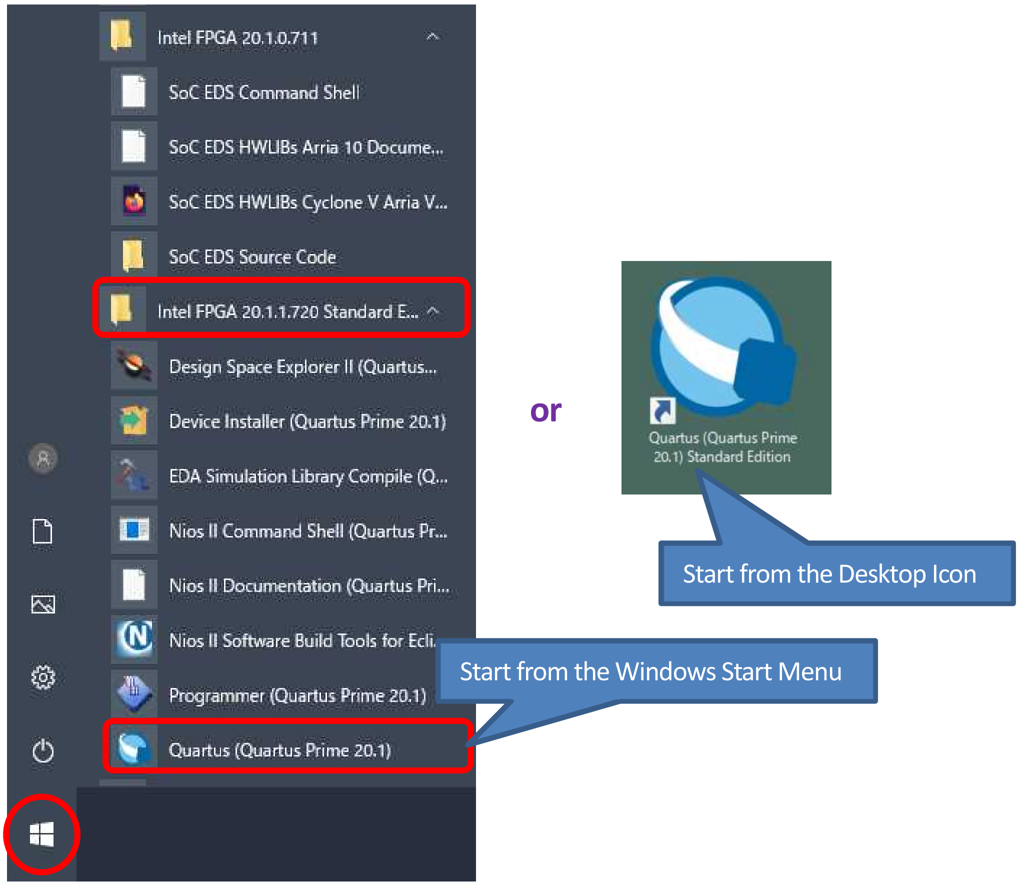
In this manual, the working directory is described as `C:¥lab¥soc_lab` folder. If you have changed the working folder, read it again according to your environment.

Let's get started.

____ 1. Start Quartus® Prime from the installed Quartus® Prime 20.1 Standard Edition (or Lite Edition) development software.

If you leave the default, you can find it below.

Windows Start => ***Intel FPGA 20.1.1.720 Standard Edition/Lite Edition*** => ***Quartus (Quartus Prime 20.1)***


Intel FPGA 20.1.0.711
SoC EDS Command Shell
SoC EDS HWLIBs Arria 10 Docume...
SoC EDS HWLIBs Cyclone V Arria V...
SoC EDS Source Code
Intel FPGA 20.1.1.720 Standard E...
Design Space Explorer II (Quartus...
Device Installer (Quartus Prime 20.1)
EDA Simulation Library Compile (Q...
Nios II Command Shell (Quartus Pr...
Nios II Documentation (Quartus Pri...
Nios II Software Build Tools for Ecli...
Programmer (Quartus Prime 20.1)
Quartus (Quartus Prime 20.1)
or
Quartus (Quartus Prime 20.1) Standard Edition

Start from the Desktop Icon

Start from the Windows Start Menu

Figure 3-2. Start Quartus® Prime

____ 2. From the **Quartus® Prime** menu bar, select ***File*** => ***Open Project*** and select soc_system.qpf located in `C:¥lab¥soc_lab¥cv_soc_lab`.

This qpf file is the project file in Quartus® Prime.

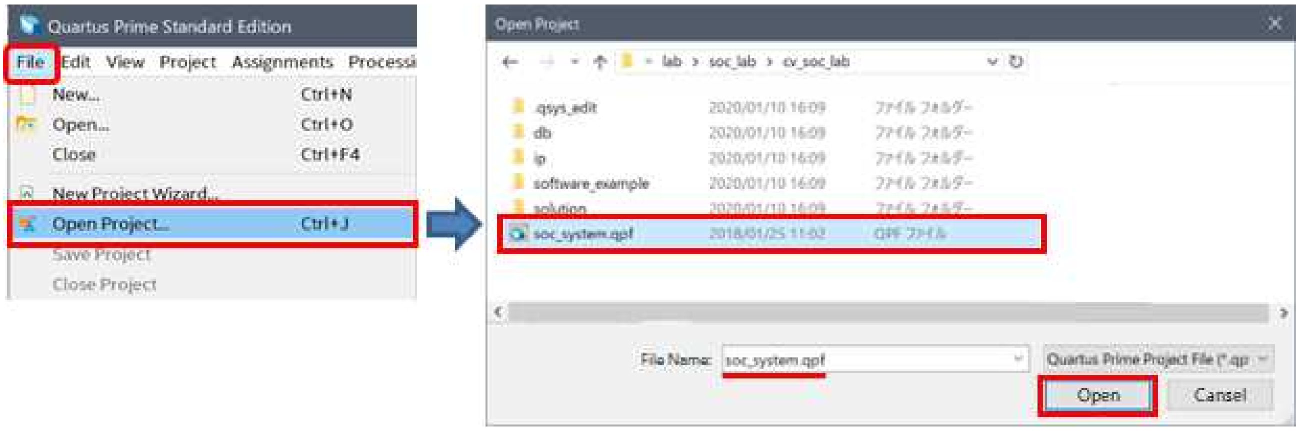

Quartus Prime Standard Edition
File Edit View Project Assignments Processi
New... Ctrl+N
Open... Ctrl+O
Close Ctrl+F4
New Project Wizard...
Open Project... Ctrl+J
Save Project
Close Project
Open Project
lab > soc_lab > cv_soc_lab
.qsys_edit 2020/01/10 16:09 ファイル フォルダー
db 2020/01/10 16:09 ファイル フォルダー
ip 2020/01/10 16:09 ファイル フォルダー
software_example 2020/01/10 16:09 ファイル フォルダー
solution 2020/01/10 16:09 ファイル フォルダー
soc_system.qpf 2018/01/25 11:02 QPF ファイル
File Name: soc_system.qpf
Quartus Prime Project File (*.qp
Open
Cansel

Figure 3-3. Opening the Quartus® Prime Project

____ 3. Select the board. Refer to the figure to set the board to be used.

- For DE0 Nano-SoC/Atlas-SoC boards: select **atlas**
- For DE10 Nano boards: select **DE10 Nano**

By making this setting, you will be able to use the pre-set information such as pin placement and device to be used for the board to be used this time.

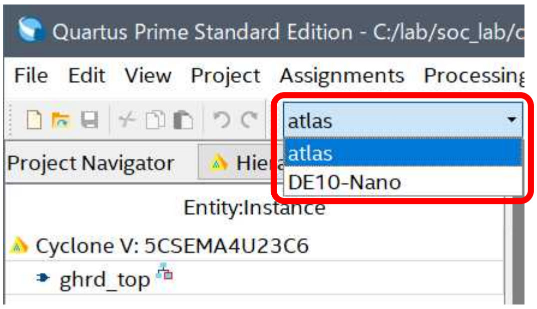

Quartus Prime Standard Edition - C:/lab/soc_lab/c
File Edit View Project Assignments Processing
atlas
atlas
DE10-Nano
Project Navigator
Hie
Entity:Instance
Cyclone V: 5CSEMA4U23C6
ghrd_top

Figure 3-4. Selecting the board to be used

____ 4. Launch ***Platform Designer*** from ***Tools*** in Quartus® Prime. Alternatively, click the Platform Designer icon in the toolbar to launch Platform Designer. Click to launch Platform Designer.

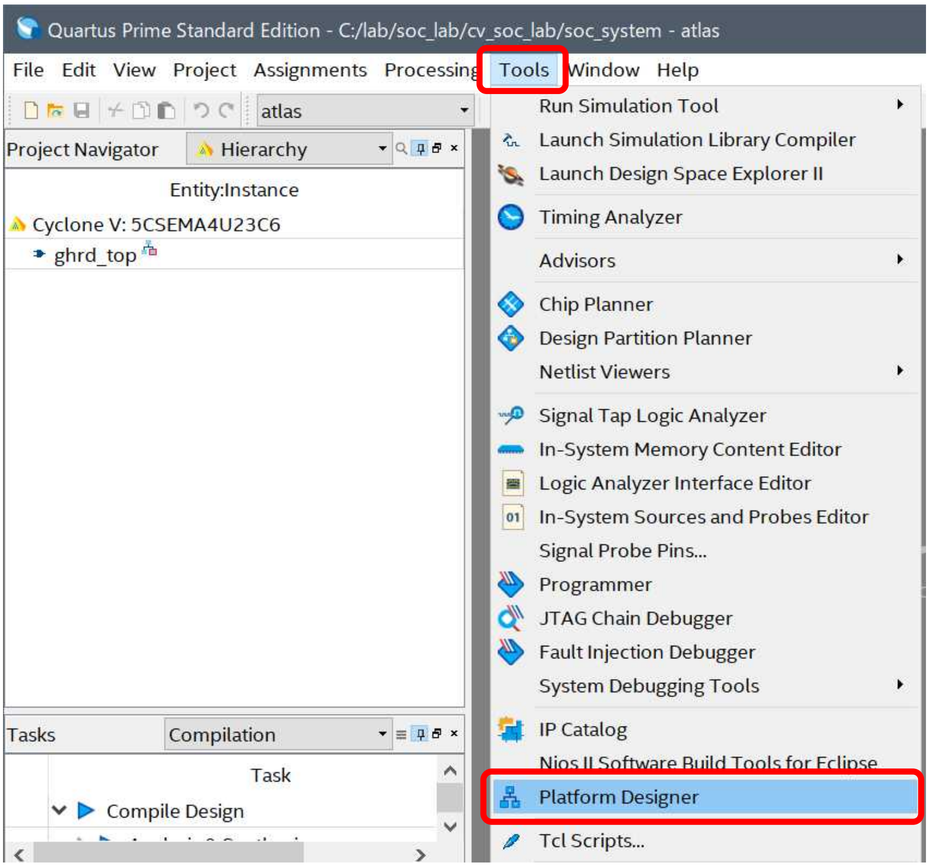
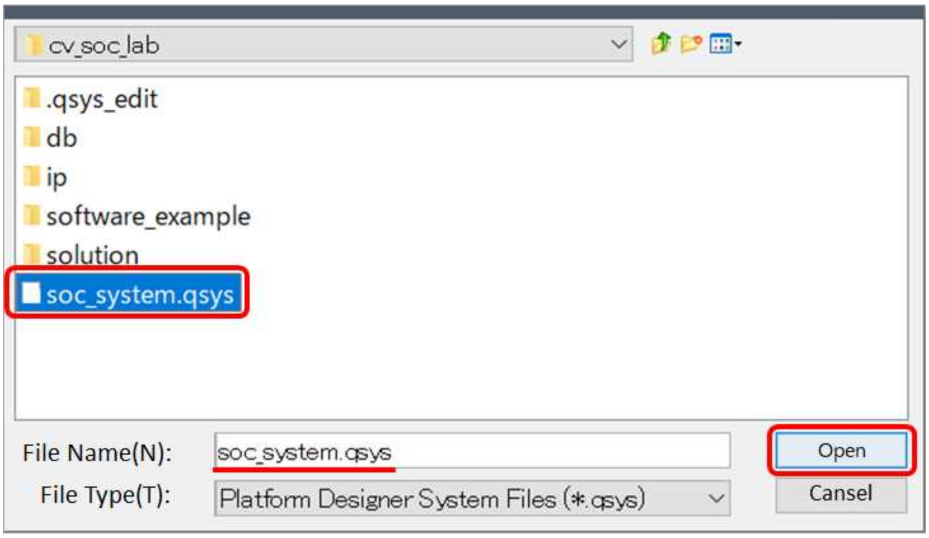

Quartus Prime Standard Edition - C:/lab/soc_lab/cv_soc_lab/soc_system - atlas
File Edit View Project Assignments Processing Tools Window Help
atlas
Project Navigator
Hierarchy
Entity:Instance
Cyclone V: 5CSEMA4U23C6
ghrd_top
Tasks
Compilation
Task
Compile Design
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer II
Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
Signal Tap Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
Signal Probe Pins...
Programmer
JTAG Chain Debugger
Fault Injection Debugger
System Debugging Tools
IP Catalog
Nios II Software Build Tools for Eclipse
Platform Designer
Tcl Scripts...

Figure 3-5. Starting the Platform Designer

____ 5. Open the soc_system.qsys file.



File Name(N): soc_system.qsys

File Type(T): Platform Designer System Files (*.qsys)

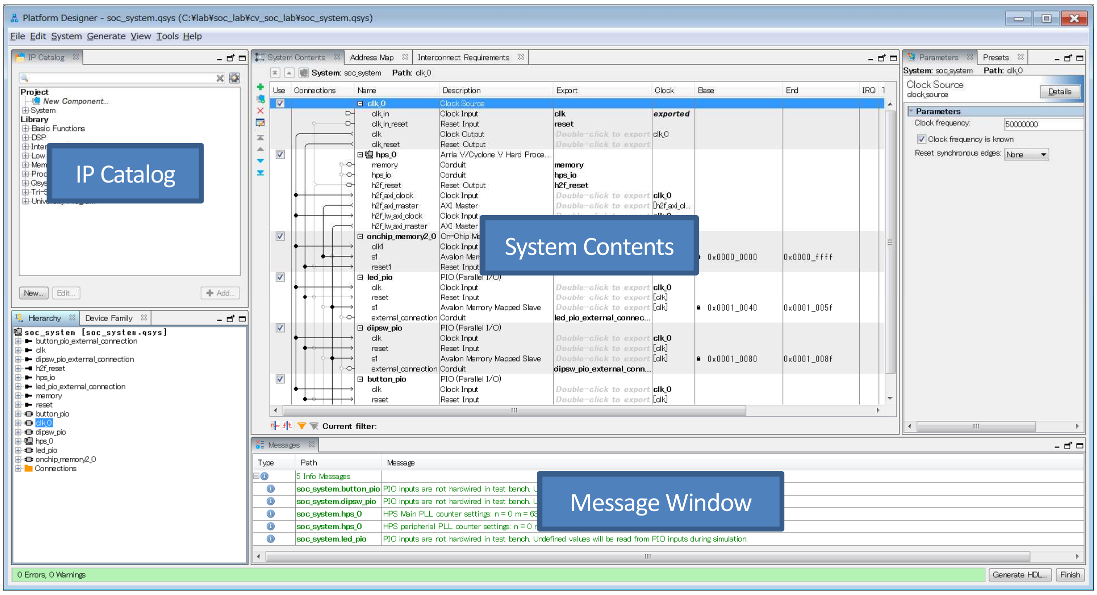
Figure 3-6. Opening the Platform Designer File

Let's first briefly explain how to use Platform Designer.

There are 3 main screens in Platform Designer: IP Catalog, System Contents, and Message Window.

The IP Catalog contains components that can be used by Platform Designer. You add the components you want to implement to System Contents. You then connect the components in System Contents to create a system.

The hard macroized part of the chip called HPS is also available as a software component in IP Catalog. By implementing this component in the Platform Designer system, you can use it on the HPS side of SoC devices.


Platform Designer - soc_system.qsys (C:¥lab¥soc_lab¥cv_soc_lab¥soc_system.qsys)
File Edit System Generate View Tools Help
IP Catalog
System Contents
Address Map
Interconnect Requirements
Parameters
Presets
Hierarchy
Device Family
Message Window
Messages
Current filter:
Generate HDL...
Finish

System: soc_system Path: clk_0

Use	Name	Description	Export	Clock	Base	End
☑	clk_0	Clock Source				
	clk_in	Clock Input	clk	exported		
	clk_in_reset	Reset Input	reset			
	clk	Clock Output	Double-click to export	clk_0		
	clk_reset	Reset Output	Double-click to export			
☑	hps_0	Arria V/Cyclone V Hard Proce...				
	memory	Conduit	memory			
	hps_io	Conduit	hps_io			
	h2f_reset	Reset Output	h2f_reset			
	h2f_axi_clock	Clock Input	Double-click to export	clk_0		
	h2f_axi_master	AXI Master	Double-click to export	[h2f_axi_cl...		
	h2f_lw_axi_clock	Clock Input				
	h2f_lw_axi_master	AXI Master				
☑	onchip_memory2_0	On-Chip Me				
	clk1	Clock Input				
	s1	Avalon Mem			0x0000_0000	0x0000_ffff
	reset1	Reset Input				
☑	led_pio	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	clk_0		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_0040	0x0001_005f
	external_connection	Conduit	led_pio_external_connec...			
☑	dipsw_pio	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	clk_0		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_0080	0x0001_008f
	external_connection	Conduit	dipsw_pio_external_conn...			
☑	button_pio	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	clk_0		
	reset	Reset Input	Double-click to export	[clk]		

Type	Path	Message
	5 Info Messages	
	soc_system.button_pio	PIO inputs are not hardwired in test bench. U
	soc_system.dipsw_pio	PIO inputs are not hardwired in test bench. U
	soc_system.hps_0	HPS Main PLL counter settings: n = 0 m = 63
	soc_system.hps_0	HPS peripherial PLL counter settings: n = 0 r
	soc_system.led_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

0 Errors, 0 Warnings

Figure 3-7. Platform Designer Screen

The following components (white) have been installed in the opened Platform Designer system. Add and configure the HPS block (blue) for this system, and connect the installed components.

- Implemented component (white):
  - Clock source
  - On-chip memory
  - PIO peripheral for LED/Button control
    - DIP switch PIO
    - Button PIO
    - LED PIO

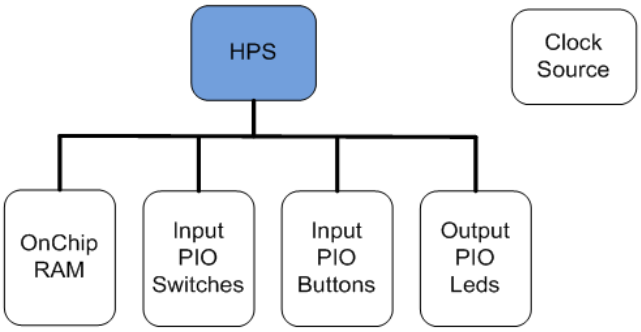


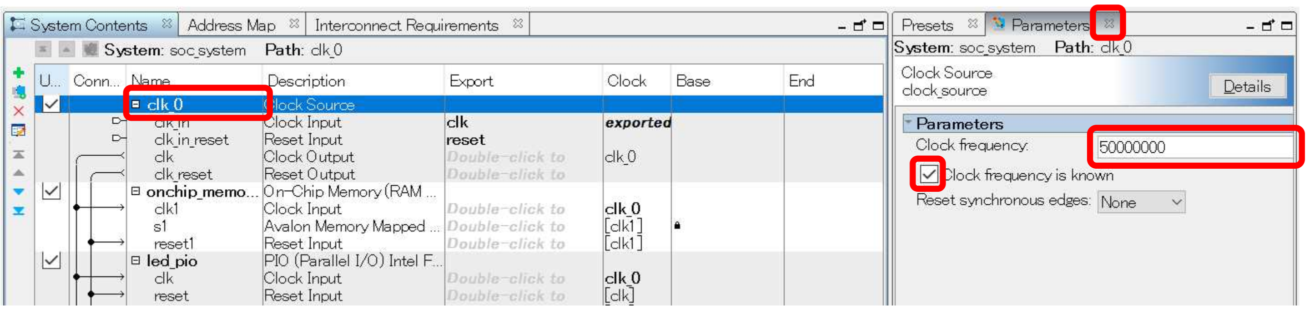
Figure 3-8. Platform Designer system to be designed

- Components to be added in the exercise (blue):
  - HPS

Platform Designer provides a setting screen for each IP. Double-click a component in System Contents to open the setting screen for that component.

____ 6. Double-click the Clock Source component (clk_0) and make sure that the ***Clock Frequency*** is set to 50 MHz to match the oscillator on the development board.

____ 7. Make sure that ***Clock frequency is known*** is checked.



System: soc_system Path: clk_0

U...	Conn...	Name	Description	Export	Clock	Base	End
☑		⊟ clk_0	Clock Source				
		clk_in	Clock Input	clk	*exported*		
		clk_in_reset	Reset Input	reset			
		clk	Clock Output	Double-click to	clk_0		
		clk_reset	Reset Output	Double-click to			
☑		⊟ onchip_memo...	On-Chip Memory (RAM ...				
		clk1	Clock Input	Double-click to	clk_0		
		s1	Avalon Memory Mapped ...	Double-click to	[clk1]	🔒	
		reset1	Reset Input	Double-click to	[clk1]		
☑		⊟ led_pio	PIO (Parallel I/O) Intel F...				
		clk	Clock Input	Double-click to	clk_0		
		reset	Reset Input	Double-click to	[clk]		

System: soc_system Path: clk_0

Clock Source
clock_source

Parameters

Clock frequency: 50000000

☑ Clock frequency is known

Reset synchronous edges: None

Figure 3-9. Checking the Clock Source

____ 8. Click **Close** (**X**) on the Parameters tab to close the ***Parameters*** tab.

The settings of each Platform Designer component are retained when you close the ***Parameters*** tab, unless you close Platform Designer.

3-2. Step 2: Adding HPS Components

The HPS consists of Dual-core Arm® Cortex™-A9 MP Core processors and various peripherals. As shown below, the Altera® SoC FPGA consists of two main blocks: the HPS block and the FPGA block.

In this step, you add and configure the HPS block in the Platform Designer system. You can configure the HPS block in the HPS block in the Platform Designer system.

The GUI used to configure the HPS provides multiple tabs (FPGA interfaces, Peripheral Pins, HPS Clocks, SDRAM), and you can configure settings for each of them.

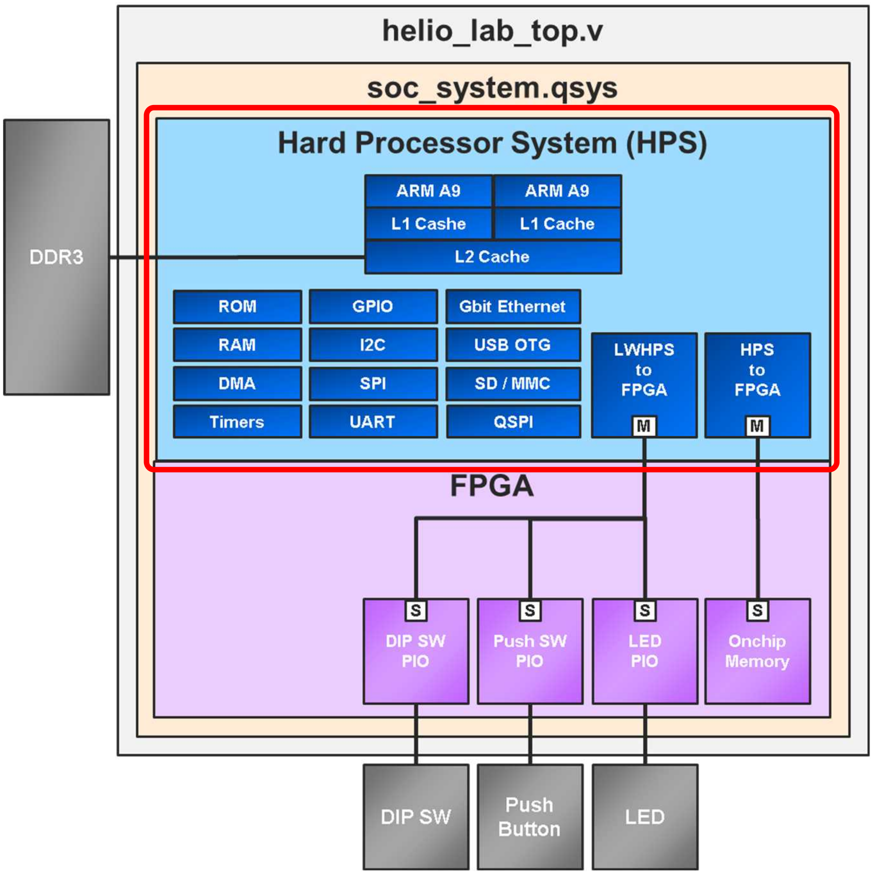


Figure 3-10. HPS Block Added to the Platform Designer System

From the next page, add an HPS block to the Platform Designer system and make various settings.

____ 1. In the search box under the ***IP Catalog*** tab, enter **processor**.

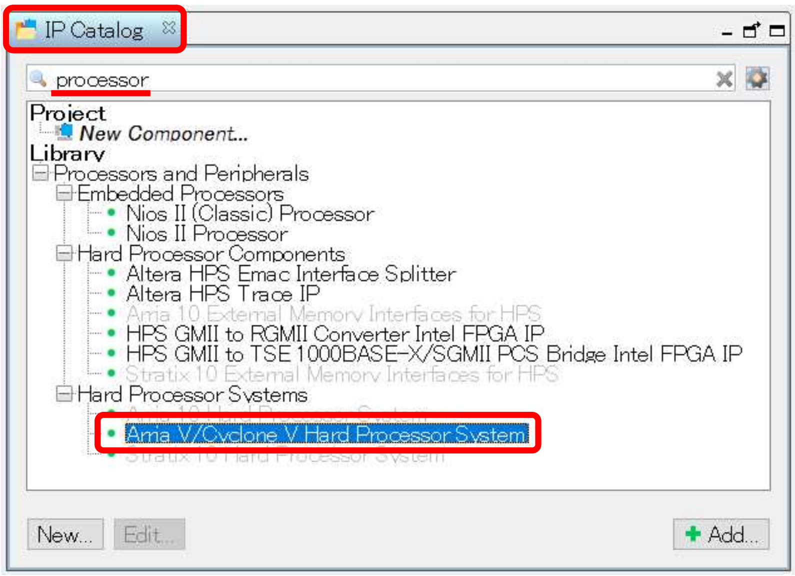

IP Catalog
processor
Project
New Component...
Library
Processors and Peripherals
Embedded Processors
Nios II (Classic) Processor
Nios II Processor
Hard Processor Components
Altera HPS Emac Interface Splitter
Altera HPS Trace IP
Arria 10 External Memory Interfaces for HPS
HPS GMII to RGMII Converter Intel FPGA IP
HPS GMII to TSE 1000BASE-X/SGMII PCS Bridge Intel FPGA IP
Stratix 10 External Memory Interfaces for HPS
Hard Processor Systems
Arria 10 Hard Processor System
Arria V/Cyclone V Hard Processor System
Stratix 10 Hard Processor System
New...
Edit...
Add...

Figure 3-11. IP Catalog Search Box

____ 2. Double-click ***Arria V/Cyclone V Hard Processor System***.

This component is the block for configuring HPS components. The dialog box for the HPS component to be configured appears. This window opens as a separate window only the first time. After clicking the ***Finish*** button, if you want to redisplay it the second time or later, double-click the HPS component from the ***System Contents*** tab.

The ***FPGA Interfaces*** tab allows you to specify whether to use signals between the HPS and the FPGA connected inside the device. Depending on the settings, the status of the HPS side can be notified to the FPGA, or the FPGA side can control the HPS side.

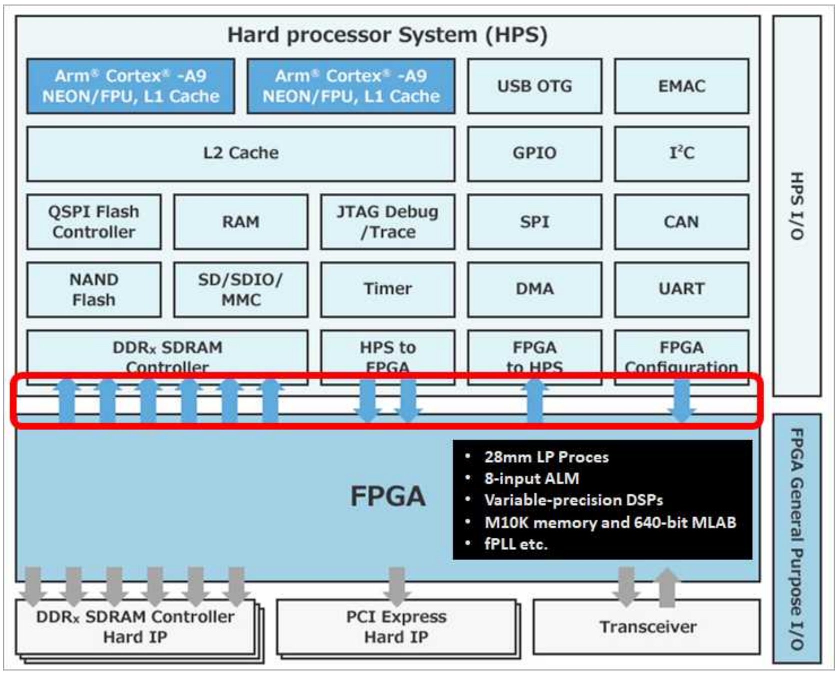
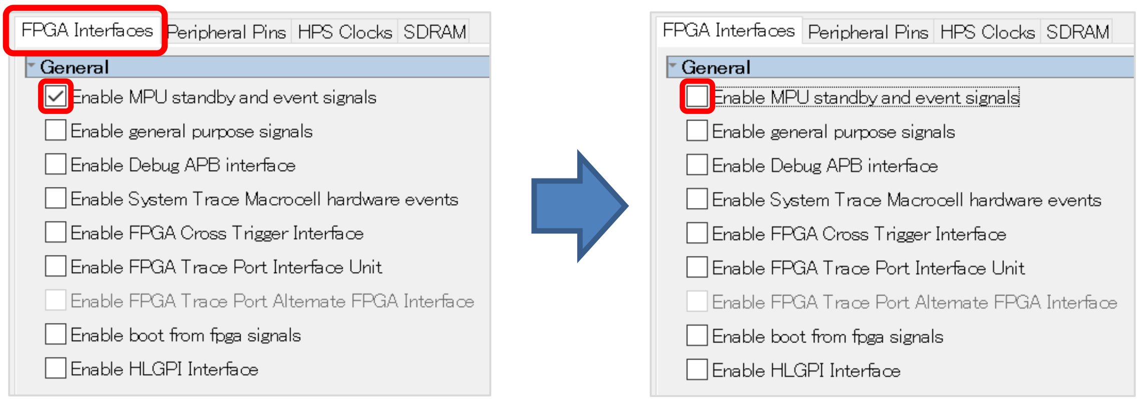


Figure 3-12. Internal bus between the HPS peripheral and the FPGA

____ 3. Click the ***FPGA Interfaces*** tab to disable ***Enable MPU standby and event signals,*** which are enabled by default.



FPGA Interfaces | Peripheral Pins | HPS Clocks | SDRAM

General

- ☑ Enable MPU standby and event signals
- □ Enable general purpose signals
- □ Enable Debug APB interface
- □ Enable System Trace Macrocell hardware events
- □ Enable FPGA Cross Trigger Interface
- □ Enable FPGA Trace Port Interface Unit
- □ Enable FPGA Trace Port Alternate FPGA Interface
- □ Enable boot from fpga signals
- □ Enable HLGPI Interface

FPGA Interfaces | Peripheral Pins | HPS Clocks | SDRAM

General

- □ Enable MPU standby and event signals
- □ Enable general purpose signals
- □ Enable Debug APB interface
- □ Enable System Trace Macrocell hardware events
- □ Enable FPGA Cross Trigger Interface
- □ Enable FPGA Trace Port Interface Unit
- □ Enable FPGA Trace Port Alternate FPGA Interface
- □ Enable boot from fpga signals
- □ Enable HLGPI Interface

Figure 3-13. FPGA Interface tab settings

ⓘ **Info:**

This is an internal signal that indicates whether the microprocessor is in standby mode or the CPU can wake up. You can connect this input signal to logic high for permanent effect, or as a processor event.

____ 4. Make sure ***Enable HLGPI Interface*** is unchecked and disabled (default).

ⓘ **Info:**

This is an option to use an unused pin (14bit) on the SDRAM interface as a general-purpose input only pin. This signal is not required for this exercise.

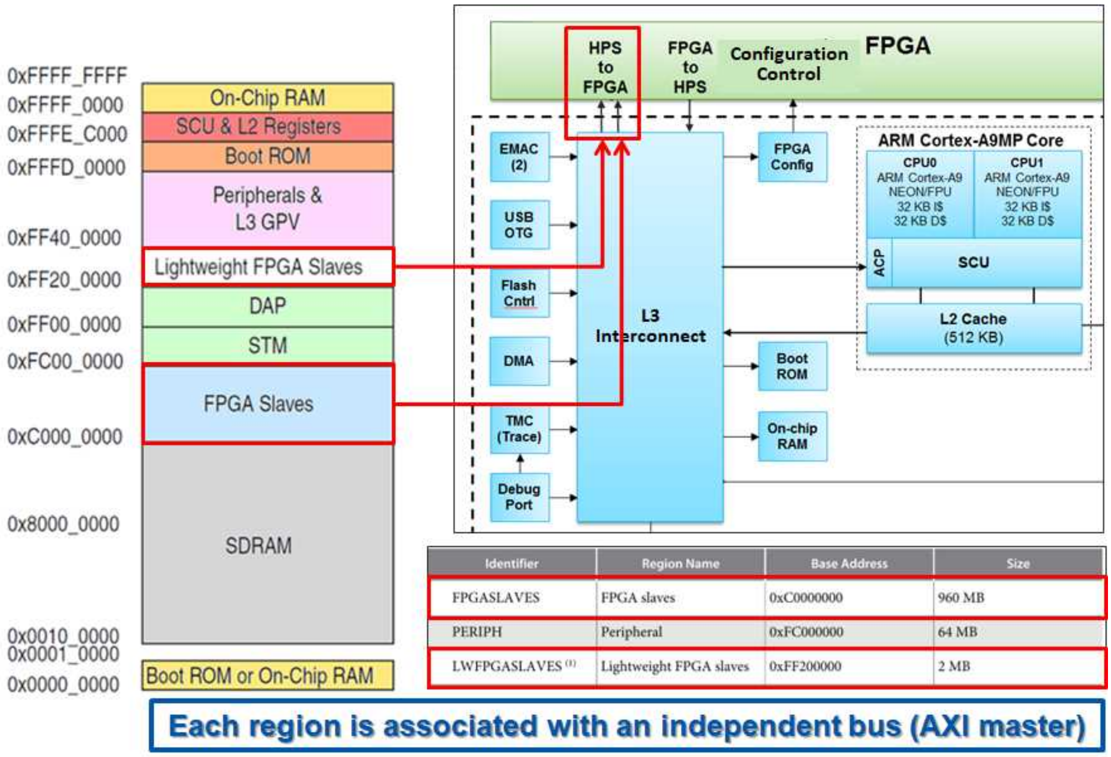
Next, configure the bridge between the HPS and the FPGA.

There are ports between the HPS and the FPGA that are master and slave respectively. There are two ports from the HPS to the FPGA and one port from the FPGA to the HPS. The two ports from the HPS to the FPGA are HPS-to-FPGA interface and lightweight HPS-to-FPGA interface respectively. The one port from the FPGA to the HPS is FPGA-to-HPS. For all ports, you can set the bus width and use or not use the port according to the path to be accessed.

When accessing from the Arm® processor or the Master on the HPS side,
you can access by specifying the address of "Bridge address + FPGA component offset address." The bridge address is as shown in the following figure.

HPS-to-FPGA interface is **`0xC000_0000`**

Lightweight HPS-to-FPGA interface is **`0xFF20_0000`**



Identifier	Region Name	Base Address	Size
FPGASLAVES	FPGA slaves	0xC0000000	960 MB
PERIPH	Peripheral	0xFC000000	64 MB
LWFPGASLAVES (1)	Lightweight FPGA slaves	0xFF200000	2 MB

Figure 3-14. HPS and FPGA internal bus and address map viewed from Arm

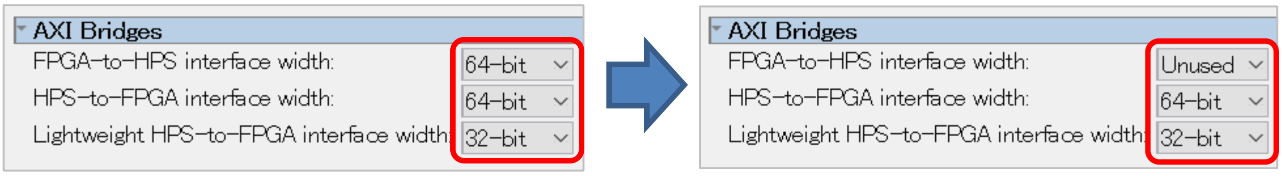
Reference:

For more information on the HPS to FPGA interface, please refer to the Macnica website technical information. See also.

Beginner's Guide to SoC - How to access between HPS-FPGA (Cyclone® V SoC/Arria® V SoC)

Make the settings on the following page.

____ 5. In the ***AXI Bridges*** section, set the ***FPGA-to-HPS interface width*** to ***Unused,*** the ***HPS-to-FPGA interface width*** to ***64 bit, and the Lightweight HPS-to-FPGA interface width*** to ***32 bit***.



AXI Bridges

FPGA-to-HPS interface width: 64-bit

HPS-to-FPGA interface width: 64-bit

Lightweight HPS-to-FPGA interface width: 32-bit

AXI Bridges

FPGA-to-HPS interface width: Unused

HPS-to-FPGA interface width: 64-bit

Lightweight HPS-to-FPGA interface width: 32-bit

Figure 3-15. Configuring AXI Bridges

ⓘ **Info:**

Enabling FPGA-to-HPS interfaces allows the master in the FPGA to access the peripheral of the HPS. This exercise does not use it.

When HPS-to-FPGA interface is enabled, HPS becomes the master and can access the peripheral of the FPGA. HPS-to-FPGA interfaces can be 32/64/128 bit wide, but for this exercise we will use the middle 64bit width.

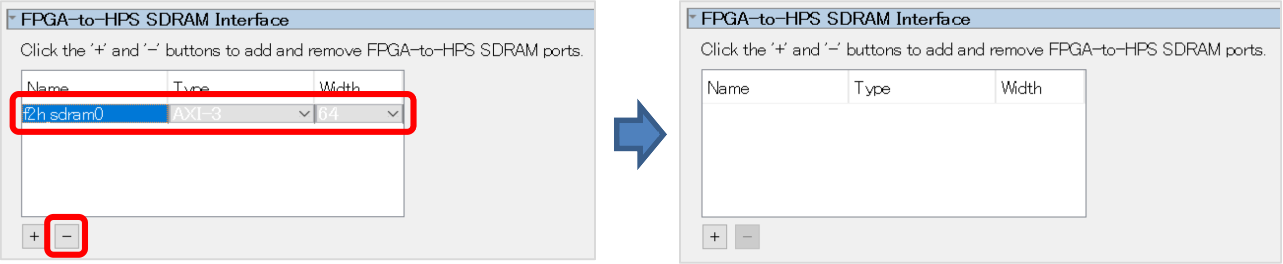
____ 6. Scroll down the ***FPGA interface*** page to find more options, including ***FPGA-to-HPS SDRAM interface, Resets*** and ***DMA Peripheral Request*** sections.

____ 7. Scroll through the ***FPGA interface*** window until you see ***FPGA to HPS SDRAM Interface***.

____ 8. Click on the ***F2h_sdram0*** interface and delete the interface by clicking the − button.

This is a broadband port that allows direct access from the FPGA to the SDRAM on the HPS side. It does not involve an interconnect and ACP (Accelerator Coherency Port) for fast access. On the other hand, data coherency is up to the user.

This is not going to be used, so I will delete the port.



FPGA-to-HPS SDRAM Interface

Click the '+' and '−' buttons to add and remove FPGA-to-HPS SDRAM ports.

Name	Type	Width
f2h sdram0	AXI-3	64

FPGA-to-HPS SDRAM Interface

Click the '+' and '−' buttons to add and remove FPGA-to-HPS SDRAM ports.

Name	Type	Width

Figure 3-16. FPGA-to-HPS SDRAM Interface Configuration

____ 9. Scroll down to the ***Resets*** section.

____ 10. In the ***Resets*** section, make sure all options for HPS reset are **disabled**.

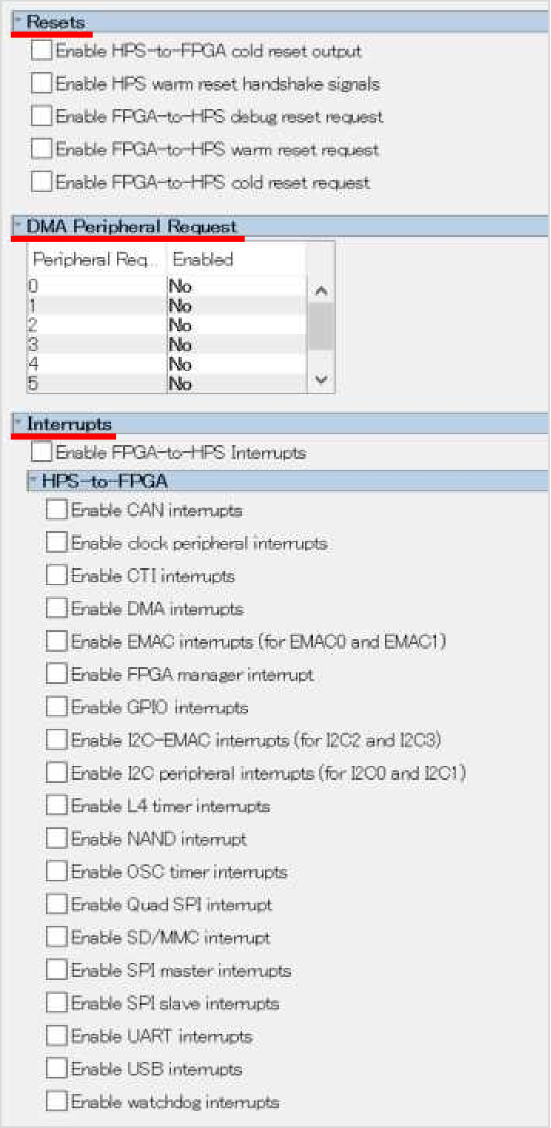
____ 11. In the ***DMA Peripheral Request*** section, make sure all lines under the ***Enabled* column** display ***No***.

> ⓘ **Info:**
>
> Enabling DMA peripheral request allows the HPS DMA controller's Peripheral Request signal to be connected to the FPGA fabric.
>
> Normally, this should be set to No, unless you are using the Peripheral Request signal for DMA transfer.

____ 12. In the ***Interrupts*** section, make sure the ***Enable FPGA-to-HPS interrupts*** option is **disabled**.

In this case, the component implemented in the FPGA will not interrupt the Arm® processor.

The Resets/DMA/Interrupts settings are as follows (no changes from the default):



Resets
- □ Enable HPS-to-FPGA cold reset output
- □ Enable HPS warm reset handshake signals
- □ Enable FPGA-to-HPS debug reset request
- □ Enable FPGA-to-HPS warm reset request
- □ Enable FPGA-to-HPS cold reset request

DMA Peripheral Request

Peripheral Req...	Enabled
0	No
1	No
2	No
3	No
4	No
5	No

Interrupts
- □ Enable FPGA-to-HPS Interrupts

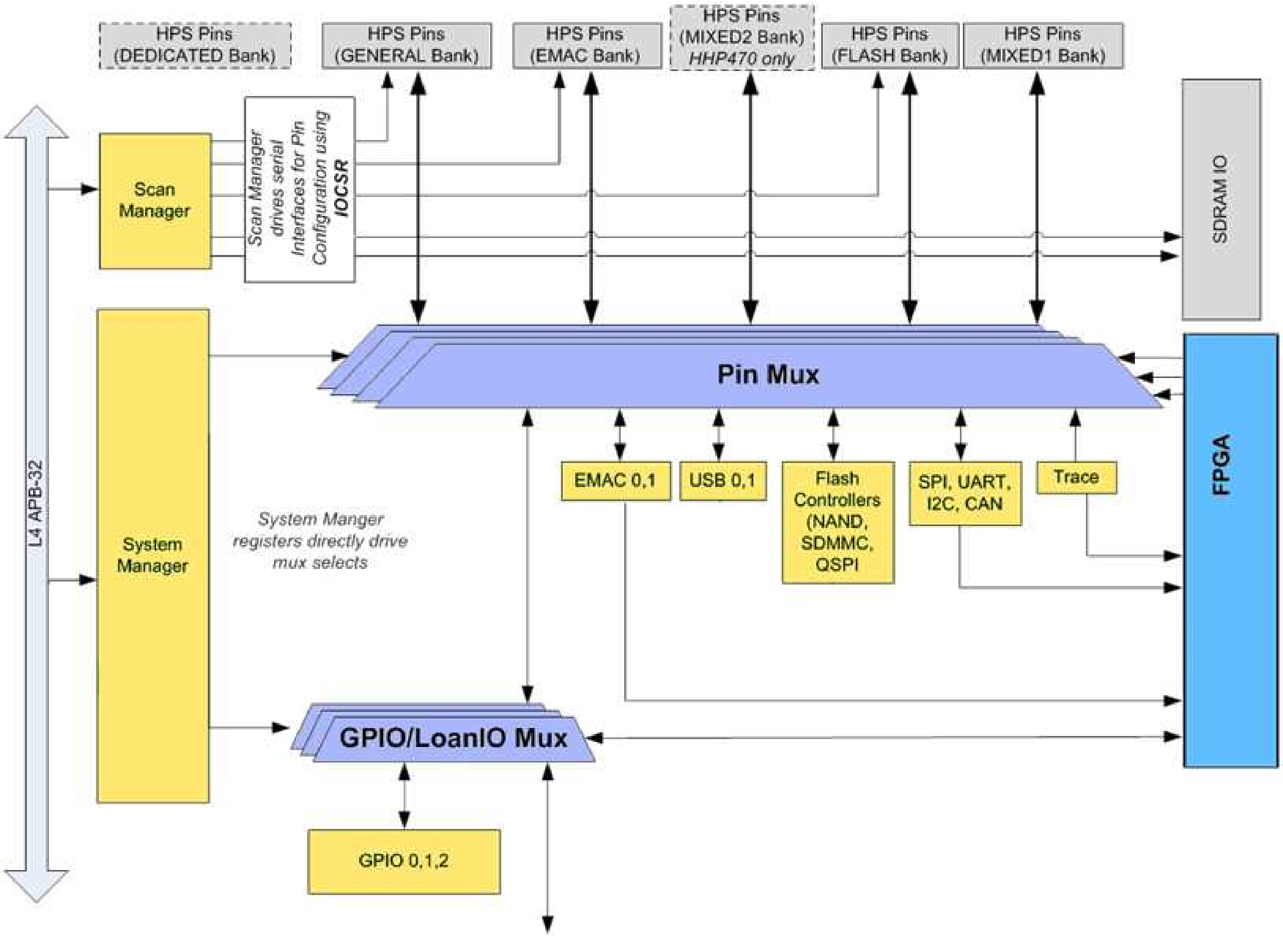
HPS-to-FPGA
- □ Enable CAN interrupts
- □ Enable clock peripheral interrupts
- □ Enable CTI interrupts
- □ Enable DMA interrupts
- □ Enable EMAC interrupts (for EMAC0 and EMAC1)
- □ Enable FPGA manager interrupt
- □ Enable GPIO interrupts
- □ Enable I2C-EMAC interrupts (for I2C2 and I2C3)
- □ Enable I2C peripheral interrupts (for I2C0 and I2C1)
- □ Enable L4 timer interrupts
- □ Enable NAND interrupt
- □ Enable OSC timer interrupts
- □ Enable Quad SPI interrupt
- □ Enable SD/MMC interrupt
- □ Enable SPI master interrupts
- □ Enable SPI slave interrupts
- □ Enable UART interrupts
- □ Enable USB interrupts
- □ Enable watchdog interrupts

Figure 3-17. Resets/DMA/Interrupts Settings

3-3. Step 3: HPS Peripheral Settings (MAC, UART, I2C, SDIO, USB)

The ***Peripheral Pins*** tab enables HPS peripherals that are hardcoded inside HPS.

Many HPS pins are shared by up to 4 peripherals. However, only 1 peripheral can be used. Therefore, pin assignments must be specified so that the peripherals to be enabled do not conflict. Pin assignments can be selected from up to 3 different parameters (HPS I/O Set 0 ~ 3).



HPS Pins (DEDICATED Bank)
HPS Pins (GENERAL Bank)
HPS Pins (EMAC Bank)
HPS Pins (MIXED2 Bank) HHP470 only
HPS Pins (FLASH Bank)
HPS Pins (MIXED1 Bank)
Scan Manager
Scan Manager drives serial Interfaces for Pin Configuration using IOCSR
SDRAM IO
L4 APB-32
System Manager
Pin Mux
FPGA
EMAC 0,1
USB 0,1
Flash Controllers (NAND, SDMMC, QSPI
SPI, UART, I2C, CAN
Trace
System Manger registers directly drive mux selects
GPIO/LoanIO Mux
GPIO 0,1,2

Figure 3-18. HPS I/O Pin Multiplexer

____ 1. Select the ***Peripheral Pins*** **tab**.

____ 2. Set the ***EMAC1 pin*** of the ***Ethernet Media Access Controller*** to ***HPS I/O Set 0***.

____ 3. Set the ***EMAC1 mode*** of the ***Ethernet Media Access Controller*** to ***RGMII***.

____ 4. Set the ***SDIO pin*** of the ***SD/MMC Controller*** to ***HPS I/O Set 0***.

____ 5. Set the ***SDIO mode*** of the ***SD/MMC Controller*** to ***4-bit Data***.

____ 6. Set the ***USB1 pin*** of the ***USB Controllers*** to ***HPS I/O Set 0***.

____ 7. Set the ***USB1 PHY interface mode*** of the ***USB Controllers*** to ***SDR with PHY clock output mode***.

____ 8. Set the ***SPIM1 pin*** of the ***SPI Controllers*** to ***HPS I/O Set 0***.

____ 9. Set ***SPIM1 mode*** of ***SPI controllers*** to ***Single Slave Select***.

____ 10. Set ***UART0 pin*** of ***UART controllers*** to ***HPS I/O Set 0***.

____ 11. Set ***UART0 mode*** of ***UART controllers*** to ***No Flow Control***.

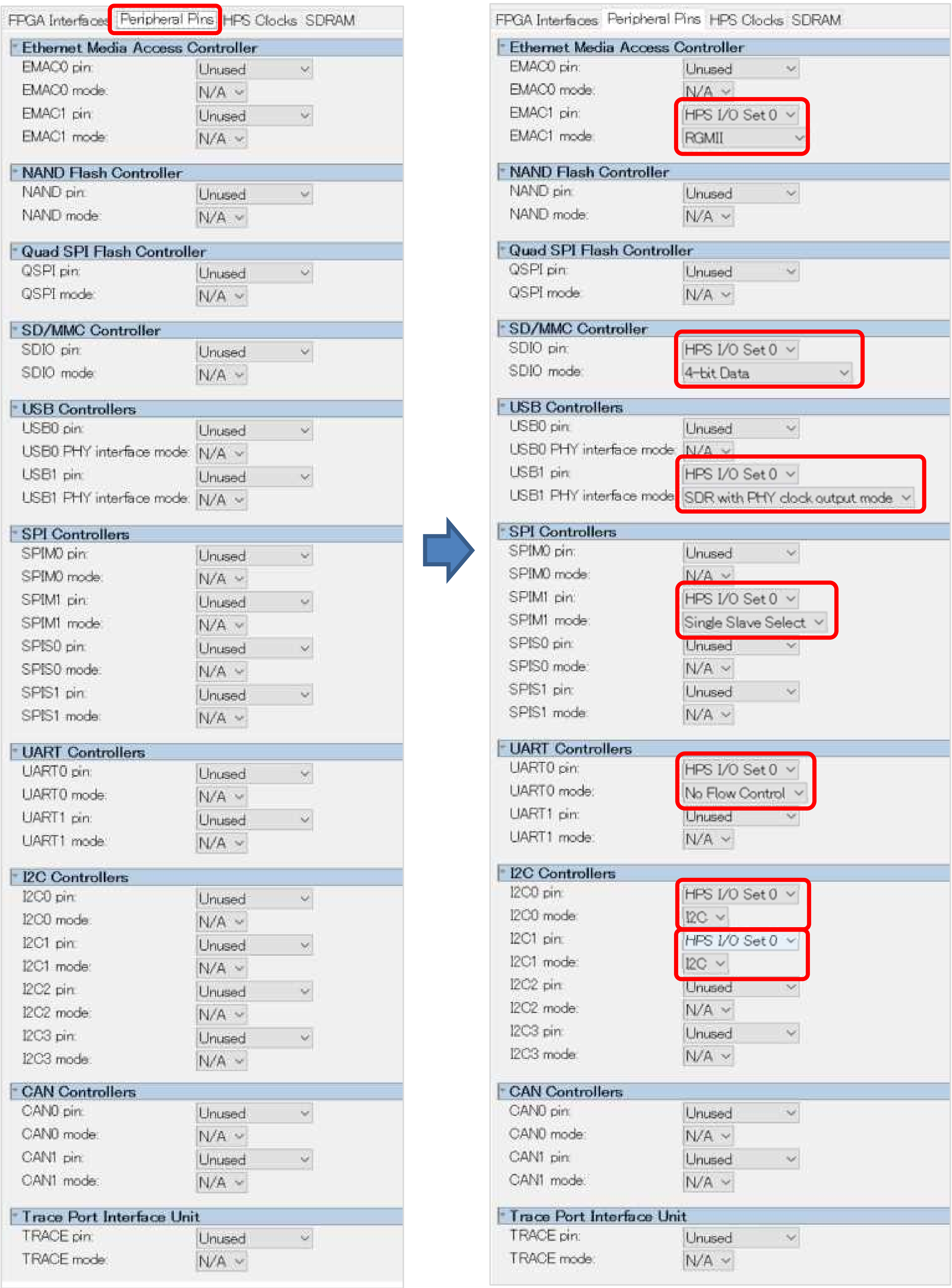
____ 12. Set ***I2C pin*** of ***I2C controllers*** to ***HPS I/O Set 0***.

____ 13. Set ***I2C0 mode of I2C controllers*** to ***I2C***.

____ 14. Set ***I2C1 pin*** of ***I2C controllers*** to ***HPS I/O Set 0***.

____ 15. Set ***I2C1 mode of I2C controllers*** to ***I2C***.

Refer to the next page for the parameters after setting.



FPGA Interfaces | Peripheral Pins | HPS Clocks | SDRAM

Ethernet Media Access Controller
EMAC0 pin: Unused
EMAC0 mode: N/A
EMAC1 pin: Unused
EMAC1 mode: N/A

NAND Flash Controller
NAND pin: Unused
NAND mode: N/A

Quad SPI Flash Controller
QSPI pin: Unused
QSPI mode: N/A

SD/MMC Controller
SDIO pin: Unused
SDIO mode: N/A

USB Controllers
USB0 pin: Unused
USB0 PHY interface mode: N/A
USB1 pin: Unused
USB1 PHY interface mode: N/A

SPI Controllers
SPIM0 pin: Unused
SPIM0 mode: N/A
SPIM1 pin: Unused
SPIM1 mode: N/A
SPIS0 pin: Unused
SPIS0 mode: N/A
SPIS1 pin: Unused
SPIS1 mode: N/A

UART Controllers
UART0 pin: Unused
UART0 mode: N/A
UART1 pin: Unused
UART1 mode: N/A

I2C Controllers
I2C0 pin: Unused
I2C0 mode: N/A
I2C1 pin: Unused
I2C1 mode: N/A
I2C2 pin: Unused
I2C2 mode: N/A
I2C3 pin: Unused
I2C3 mode: N/A

CAN Controllers
CAN0 pin: Unused
CAN0 mode: N/A
CAN1 pin: Unused
CAN1 mode: N/A

Trace Port Interface Unit
TRACE pin: Unused
TRACE mode: N/A

FPGA Interfaces | Peripheral Pins | HPS Clocks | SDRAM

Ethernet Media Access Controller
EMAC0 pin: Unused
EMAC0 mode: N/A
EMAC1 pin: HPS I/O Set 0
EMAC1 mode: RGMII

NAND Flash Controller
NAND pin: Unused
NAND mode: N/A

Quad SPI Flash Controller
QSPI pin: Unused
QSPI mode: N/A

SD/MMC Controller
SDIO pin: HPS I/O Set 0
SDIO mode: 4-bit Data

USB Controllers
USB0 pin: Unused
USB0 PHY interface mode: N/A
USB1 pin: HPS I/O Set 0
USB1 PHY interface mode: SDR with PHY clock output mode

SPI Controllers
SPIM0 pin: Unused
SPIM0 mode: N/A
SPIM1 pin: HPS I/O Set 0
SPIM1 mode: Single Slave Select
SPIS0 pin: Unused
SPIS0 mode: N/A
SPIS1 pin: Unused
SPIS1 mode: N/A

UART Controllers
UART0 pin: HPS I/O Set 0
UART0 mode: No Flow Control
UART1 pin: Unused
UART1 mode: N/A

I2C Controllers
I2C0 pin: HPS I/O Set 0
I2C0 mode: I2C
I2C1 pin: HPS I/O Set 0
I2C1 mode: I2C
I2C2 pin: Unused
I2C2 mode: N/A
I2C3 pin: Unused
I2C3 mode: N/A

CAN Controllers
CAN0 pin: Unused
CAN0 mode: N/A
CAN1 pin: Unused
CAN1 mode: N/A

Trace Port Interface Unit
TRACE pin: Unused
TRACE mode: N/A

Figure 3-19. Configuring HPS Peripherals

In the ***Peripherals Mux Table*** section, you can see the placement of the pins you set up.

A pin can only have one role. Therefore, multiple HPS peripherals cannot use the same pin, and the same pin cannot have the role of HPS Peripheral and GPIO. Therefore, use this ***Peripherals Mux Table*** section to see what each pin is used for.

The left column shows the pin name, and if the pin is used, it is bolded. Pins that are not used as peripheral pins can be used as HPS GPIO pins. In this case, you can activate them by pressing the respective GPIO button for each pin.

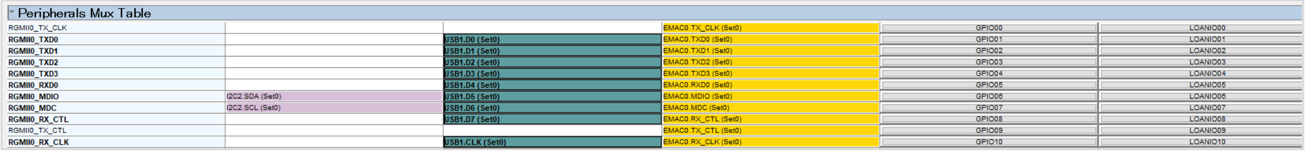
If there is a conflict for a pin, an **Error** is displayed in the **Message Window** and the pin field is highlighted in red so that you can see in real time which pin is causing the conflict.

Let's set the pins that are not used as GPIO pins.

____ 16. Enable GPIO09 by clicking ***GPIO09*** in the ***Peripherals Mux Table*** section.

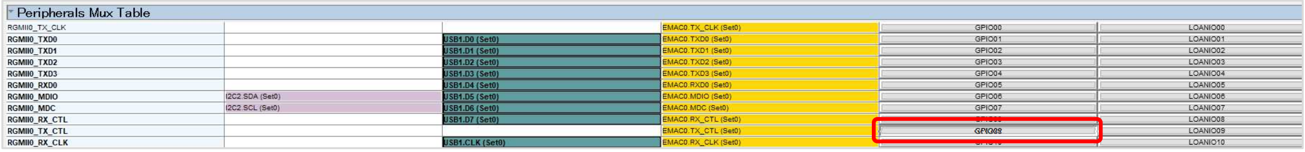
Ⓟ Point:

The response may take a while, so be careful not to press it too many times.



Peripherals Mux Table

RGMII0_TX_CLK			EMAC0.TX_CLK (Set0)	GPIO00	LOANIO00
RGMII0_TXD0		USB1.D0 (Set0)	EMAC0.TXD0 (Set0)	GPIO01	LOANIO01
RGMII0_TXD1		USB1.D1 (Set0)	EMAC0.TXD1 (Set0)	GPIO02	LOANIO02
RGMII0_TXD2		USB1.D2 (Set0)	EMAC0.TXD2 (Set0)	GPIO03	LOANIO03
RGMII0_TXD3		USB1.D3 (Set0)	EMAC0.TXD3 (Set0)	GPIO04	LOANIO04
RGMII0_RXD0		USB1.D4 (Set0)	EMAC0.RXD0 (Set0)	GPIO05	LOANIO05
RGMII0_MDIO	I2C2.SDA (Set0)	USB1.D5 (Set0)	EMAC0.MDIO (Set0)	GPIO06	LOANIO06
RGMII0_MDC	I2C2.SCL (Set0)	USB1.D6 (Set0)	EMAC0.MDC (Set0)	GPIO07	LOANIO07
RGMII0_RX_CTL		USB1.D7 (Set0)	EMAC0.RX_CTL (Set0)	GPIO08	LOANIO08
RGMII0_TX_CTL			EMAC0.TX_CTL (Set0)	GPIO09	LOANIO09
RGMII0_RX_CLK		USB1.CLK (Set0)	EMAC0.RX_CLK (Set0)	GPIO10	LOANIO10



Peripherals Mux Table

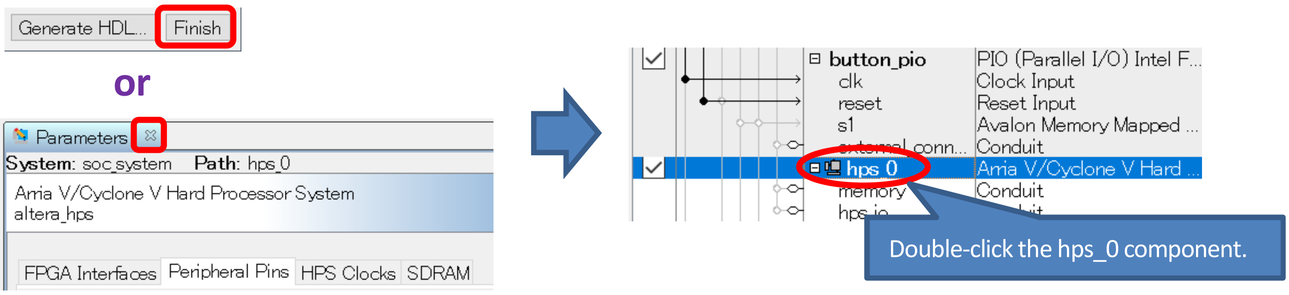
RGMII0_TX_CLK			EMAC0.TX_CLK (Set0)	GPIO00	LOANIO00
RGMII0_TXD0		USB1.D0 (Set0)	EMAC0.TXD0 (Set0)	GPIO01	LOANIO01
RGMII0_TXD1		USB1.D1 (Set0)	EMAC0.TXD1 (Set0)	GPIO02	LOANIO02
RGMII0_TXD2		USB1.D2 (Set0)	EMAC0.TXD2 (Set0)	GPIO03	LOANIO03
RGMII0_TXD3		USB1.D3 (Set0)	EMAC0.TXD3 (Set0)	GPIO04	LOANIO04
RGMII0_RXD0		USB1.D4 (Set0)	EMAC0.RXD0 (Set0)	GPIO05	LOANIO05
RGMII0_MDIO	I2C2.SDA (Set0)	USB1.D5 (Set0)	EMAC0.MDIO (Set0)	GPIO06	LOANIO06
RGMII0_MDC	I2C2.SCL (Set0)	USB1.D6 (Set0)	EMAC0.MDC (Set0)	GPIO07	LOANIO07
RGMII0_RX_CTL		USB1.D7 (Set0)	EMAC0.RX_CTL (Set0)	GPIO08	LOANIO08
RGMII0_TX_CTL			EMAC0.TX_CTL (Set0)	GPIO09	LOANIO09
RGMII0_RX_CLK		USB1.CLK (Set0)	EMAC0.RX_CLK (Set0)	GPIO10	LOANIO10

Figure 3-20. HPS GPIO09 settings

Ⓟ Point:

If you cannot click it, select [***Finish***] in the lower right corner of the HPS component dialog box, or click "**x**" in the Parameters tab to close the HPS component dialog box.

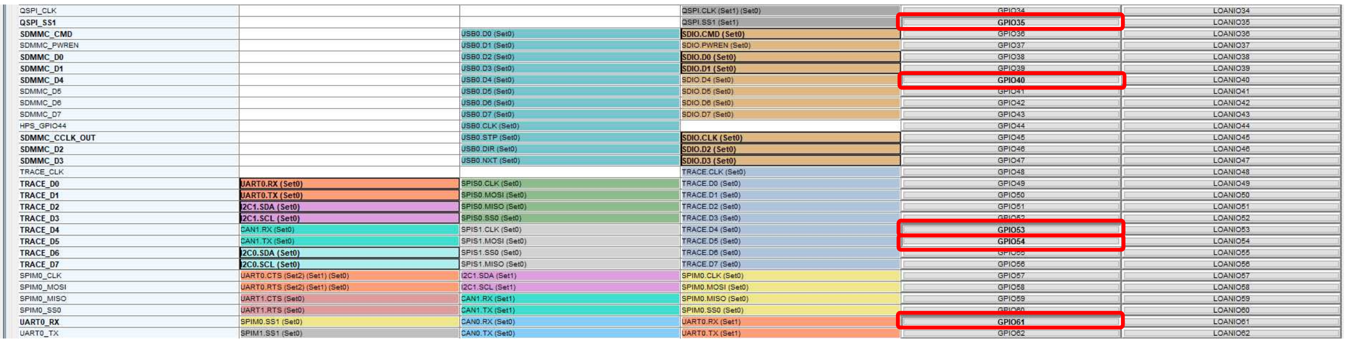
Double-click the hps_0 component again to open the parameters window and continue working.



Double-click the hps_0 component.

Figure 3-21. What to do if you cannot click during GPIO settings

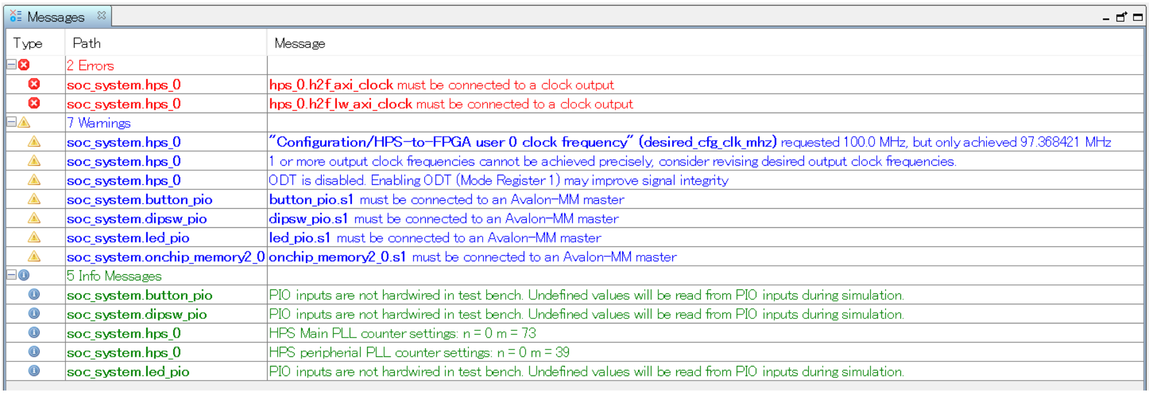
____ 17. Similarly, enable ***GPIO35, GPIO40, GPIO53, GPIO54, and GPIO61***.



QSPI_CLK			QSPI.CLK (Set1) (Set0)	GPIO34	LOANIO34
QSPI_SS1			QSPI.SS1 (Set1)	GPIO35	LOANIO35
SDMMC_CMD		USB0.D0 (Set0)	SDIO.CMD (Set0)	GPIO36	LOANIO36
SDMMC_PWREN		USB0.D1 (Set0)	SDIO.PWREN (Set0)	GPIO37	LOANIO37
SDMMC_D0		USB0.D2 (Set0)	SDIO.D0 (Set0)	GPIO38	LOANIO38
SDMMC_D1		USB0.D3 (Set0)	SDIO.D1 (Set0)	GPIO39	LOANIO39
SDMMC_D4		USB0.D4 (Set0)	SDIO.D4 (Set0)	GPIO40	LOANIO40
SDMMC_D5		USB0.D5 (Set0)	SDIO.D5 (Set0)	GPIO41	LOANIO41
SDMMC_D6		USB0.D6 (Set0)	SDIO.D6 (Set0)	GPIO42	LOANIO42
SDMMC_D7		USB0.D7 (Set0)	SDIO.D7 (Set0)	GPIO43	LOANIO43
HPS_GPIO44		USB0.CLK (Set0)		GPIO44	LOANIO44
SDMMC_CCLK_OUT		USB0.STP (Set0)	SDIO.CLK (Set0)	GPIO45	LOANIO45
SDMMC_D2		USB0.DIR (Set0)	SDIO.D2 (Set0)	GPIO46	LOANIO46
SDMMC_D3		USB0.NXT (Set0)	SDIO.D3 (Set0)	GPIO47	LOANIO47
TRACE_CLK			TRACE.CLK (Set0)	GPIO48	LOANIO48
TRACE_D0	UART0.RX (Set0)	SPIS0.CLK (Set0)	TRACE.D0 (Set0)	GPIO49	LOANIO49
TRACE_D1	UART0.TX (Set0)	SPIS0.MOSI (Set0)	TRACE.D1 (Set0)	GPIO50	LOANIO50
TRACE_D2	I2C1.SDA (Set0)	SPIS0.MISO (Set0)	TRACE.D2 (Set0)	GPIO51	LOANIO51
TRACE_D3	I2C1.SCL (Set0)	SPIS0.SS0 (Set0)	TRACE.D3 (Set0)	GPIO52	LOANIO52
TRACE_D4	CAN1.RX (Set0)	SPIS1.CLK (Set0)	TRACE.D4 (Set0)	GPIO53	LOANIO53
TRACE_D5	CAN1.TX (Set0)	SPIS1.MOSI (Set0)	TRACE.D5 (Set0)	GPIO54	LOANIO54
TRACE_D6	I2C0.SDA (Set0)	SPIS1.SS0 (Set0)	TRACE.D6 (Set0)	GPIO55	LOANIO55
TRACE_D7	I2C0.SCL (Set0)	SPIS1.MISO (Set0)	TRACE.D7 (Set0)	GPIO56	LOANIO56
SPIM0_CLK	UART0.CTS (Set2) (Set1) (Set0)	I2C1.SDA (Set1)	SPIM0.CLK (Set0)	GPIO57	LOANIO57
SPIM0_MOSI	UART0.RTS (Set2) (Set1) (Set0)	I2C1.SCL (Set1)	SPIM0.MOSI (Set0)	GPIO58	LOANIO58
SPIM0_MISO	UART1.CTS (Set0)	CAN1.RX (Set1)	SPIM0.MISO (Set0)	GPIO59	LOANIO59
SPIM0_SS0	UART1.RTS (Set0)	CAN1.TX (Set1)	SPIM0.SS0 (Set0)	GPIO60	LOANIO60
UART0_RX	SPIM0.SS1 (Set0)	CAN0.RX (Set0)	UART0.RX (Set1)	GPIO61	LOANIO61
UART0_TX	SPIM1.SS1 (Set0)	CAN0.TX (Set0)	UART0.TX (Set1)	GPIO62	LOANIO62

Figure 3-22. HPS GPIO settings

____ 18. After configuring, make sure that there are no errors other than the two errors shown below (these two errors will be resolved later).

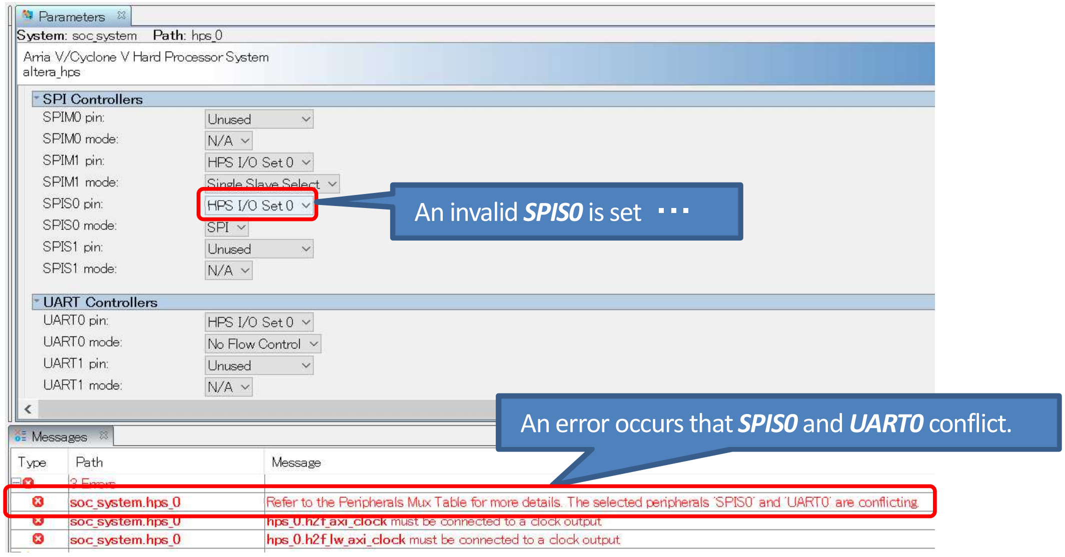


Type	Path	Message
	2 Errors	
	soc_system.hps_0	hps_0.h2f_axi_clock must be connected to a clock output
	soc_system.hps_0	hps_0.h2f_lw_axi_clock must be connected to a clock output
	7 Warnings	
	soc_system.hps_0	"Configuration/HPS-to-FPGA user 0 clock frequency" (desired_cfg_clk_mhz) requested 100.0 MHz, but only achieved 97.368421 MHz
	soc_system.hps_0	1 or more output clock frequencies cannot be achieved precisely, consider revising desired output clock frequencies.
	soc_system.hps_0	ODT is disabled. Enabling ODT (Mode Register 1) may improve signal integrity
	soc_system.button_pio	button_pio.s1 must be connected to an Avalon-MM master
	soc_system.dipsw_pio	dipsw_pio.s1 must be connected to an Avalon-MM master
	soc_system.led_pio	led_pio.s1 must be connected to an Avalon-MM master
	soc_system.onchip_memory2_0	onchip_memory2_0.s1 must be connected to an Avalon-MM master
	5 Info Messages	
	soc_system.button_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
	soc_system.dipsw_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
	soc_system.hps_0	HPS Main PLL counter settings: n = 0 m = 73
	soc_system.hps_0	HPS peripherial PLL counter settings: n = 0 m = 39
	soc_system.led_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

Figure 3-23. Example of display when there is no pin conflict error

For example, if the error shown in Figure 3-24. appears, there is a conflict between the pins of ***SPIS0*** and ***UART0***. Check whether there is an error in the setting and correct it.

In this example, an error occurs because ***SPIS0,*** which should not be used, is to be used. If the setting is ***Unused***, the error disappears.



System: soc_system Path: hps_0

Arria V/Cyclone V Hard Processor System
altera_hps

SPI Controllers

SPIM0 pin:	Unused
SPIM0 mode:	N/A
SPIM1 pin:	HPS I/O Set 0
SPIM1 mode:	Single Slave Select
SPIS0 pin:	HPS I/O Set 0
SPIS0 mode:	SPI
SPIS1 pin:	Unused
SPIS1 mode:	N/A

An invalid ***SPIS0*** is set ･･･

UART Controllers

UART0 pin:	HPS I/O Set 0
UART0 mode:	No Flow Control
UART1 pin:	Unused
UART1 mode:	N/A

An error occurs that ***SPIS0*** and ***UART0*** conflict.

Type	Path	Message
	3 Errors	
	soc_system.hps_0	Refer to the Peripherals Mux Table for more details. The selected peripherals 'SPIS0' and 'UART0' are conflicting.
	soc_system.hps_0	hps_0.h2f_axi_clock must be connected to a clock output
	soc_system.hps_0	hps_0.h2f_lw_axi_clock must be connected to a clock output

Figure 3-24. Example of display with a pin conflict error

3-4. Step 4: Set the HPS Clock

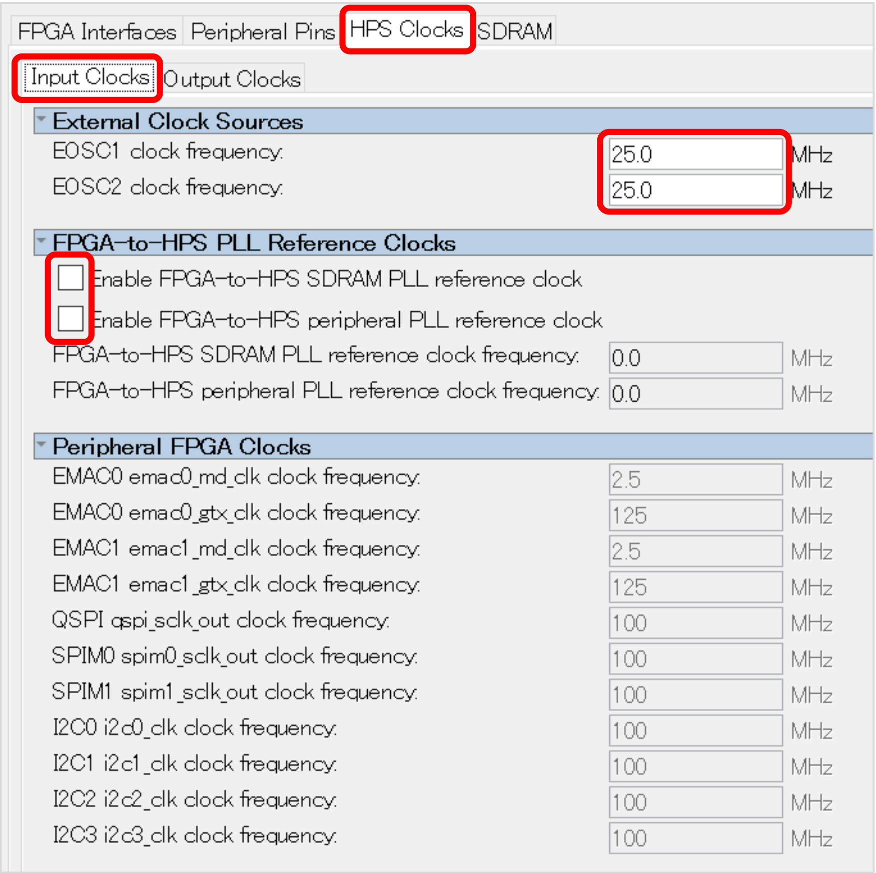
The ***HPS Clocks*** tab sets the Clock source and frequency. All of these parameters are managed by the Clock Manager Component.

The parameters set in this tab are used when the boot loader (Preloader software) is generated. The Preloader is generated by "4. Lab 2 - Software Exercise (1) Generate Preloader."

____ 1. Select the ***HPS Clocks*** tab.

____ 2. Select the ***Input Clocks*** tab.

____ 3. Ensure that the ***EOSC1/EOSC2 clock frequency*** is set to ***25 MHz***. Also ensure that all ***FPGA-to-HPS PLL Reference clocks*** are **disabled**. The EOSC1 is a dedicated pin on the HPS side and is the clock source required to generate the clock for the HPS MPU. The Atlas-SoC board and the DE10 Nano board used in this article are set to 25 MHz.



External Clock Sources

EOSC1 clock frequency: 25.0 MHz

EOSC2 clock frequency: 25.0 MHz

FPGA-to-HPS PLL Reference Clocks

□ Enable FPGA-to-HPS SDRAM PLL reference clock

□ Enable FPGA-to-HPS peripheral PLL reference clock

FPGA-to-HPS SDRAM PLL reference clock frequency: 0.0 MHz

FPGA-to-HPS peripheral PLL reference clock frequency: 0.0 MHz

Peripheral FPGA Clocks

EMAC0 emac0_md_clk clock frequency: 2.5 MHz

EMAC0 emac0_gtx_clk clock frequency: 125 MHz

EMAC1 emac1_md_clk clock frequency: 2.5 MHz

EMAC1 emac1_gtx_clk clock frequency: 125 MHz

QSPI qspi_sclk_out clock frequency: 100 MHz

SPIM0 spim0_sclk_out clock frequency: 100 MHz

SPIM1 spim1_sclk_out clock frequency: 100 MHz

I2C0 i2c0_clk clock frequency: 100 MHz

I2C1 i2c1_clk clock frequency: 100 MHz

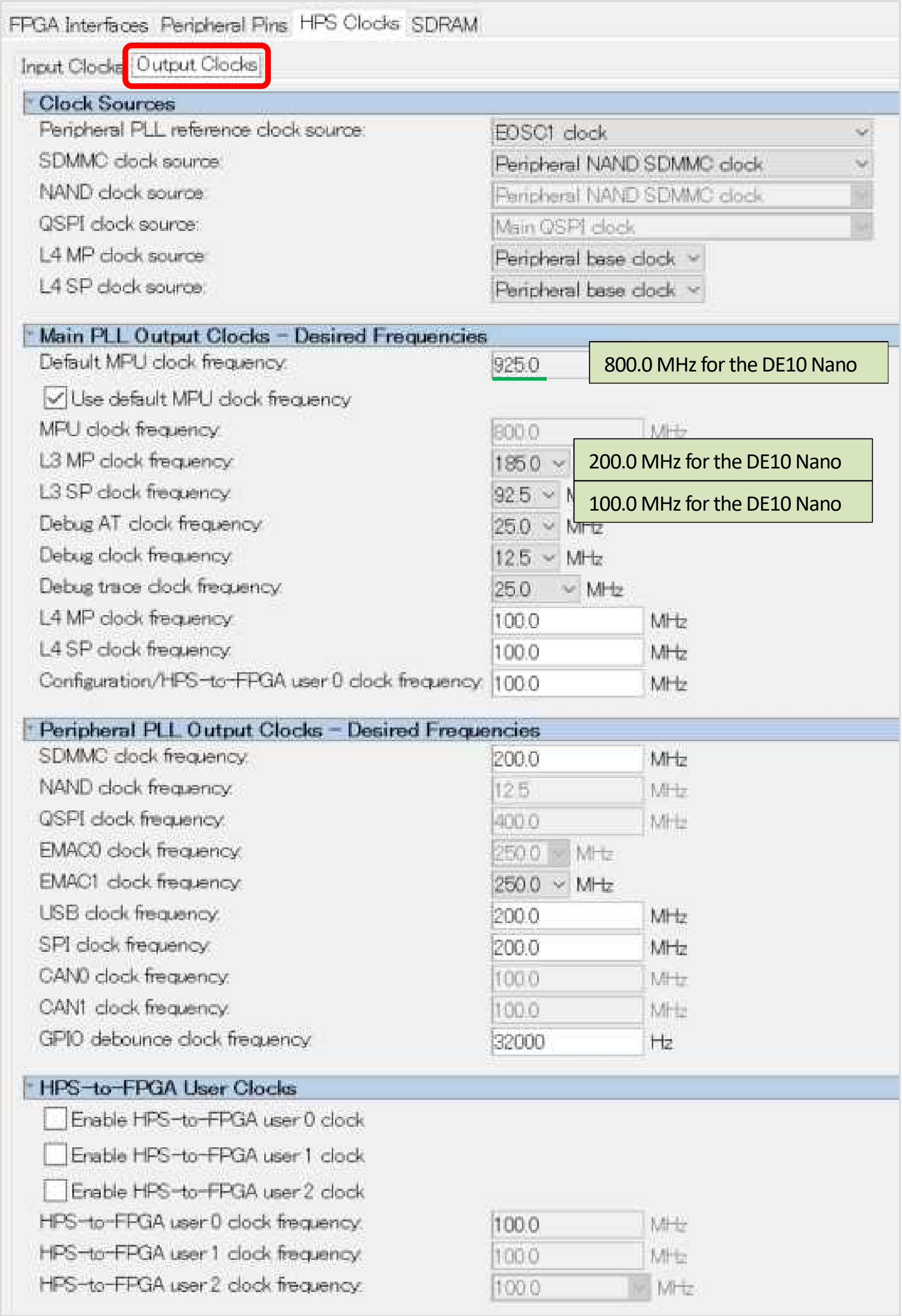
I2C2 i2c2_clk clock frequency: 100 MHz

I2C3 i2c3_clk clock frequency: 100 MHz

Figure 3-25. HPS to FPGA Clock Settings

____ 4. Select the ***Output Clocks*** tab.

____ 5. Make sure that the settings are as shown below (no change from the default). This tab allows you to set the operating frequency of each HPS peripheral. The PLL setting is automatically calculated according to the set value.



Clock Sources

Peripheral PLL reference clock source:	EOSC1 clock
SDMMC clock source:	Peripheral NAND SDMMC clock
NAND clock source:	Peripheral NAND SDMMC clock
QSPI clock source:	Main QSPI clock
L4 MP clock source:	Peripheral base clock
L4 SP clock source:	Peripheral base clock

Main PLL Output Clocks – Desired Frequencies

Default MPU clock frequency:	925.0
☑ Use default MPU clock frequency	
MPU clock frequency:	800.0 MHz
L3 MP clock frequency:	185.0 MHz
L3 SP clock frequency:	92.5 MHz
Debug AT clock frequency:	25.0 MHz
Debug clock frequency:	12.5 MHz
Debug trace clock frequency:	25.0 MHz
L4 MP clock frequency:	100.0 MHz
L4 SP clock frequency:	100.0 MHz
Configuration/HPS-to-FPGA user 0 clock frequency:	100.0 MHz

800.0 MHz for the DE10 Nano

200.0 MHz for the DE10 Nano

100.0 MHz for the DE10 Nano

Peripheral PLL Output Clocks – Desired Frequencies

SDMMC clock frequency:	200.0 MHz
NAND clock frequency:	12.5 MHz
QSPI clock frequency:	400.0 MHz
EMAC0 clock frequency:	250.0 MHz
EMAC1 clock frequency:	250.0 MHz
USB clock frequency:	200.0 MHz
SPI clock frequency:	200.0 MHz
CAN0 clock frequency:	100.0 MHz
CAN1 clock frequency:	100.0 MHz
GPIO debounce clock frequency:	32000 Hz

HPS-to-FPGA User Clocks

☐ Enable HPS-to-FPGA user 0 clock

☐ Enable HPS-to-FPGA user 1 clock

☐ Enable HPS-to-FPGA user 2 clock

HPS-to-FPGA user 0 clock frequency:	100.0 MHz
HPS-to-FPGA user 1 clock frequency:	100.0 MHz
HPS-to-FPGA user 2 clock frequency:	100.0 MHz

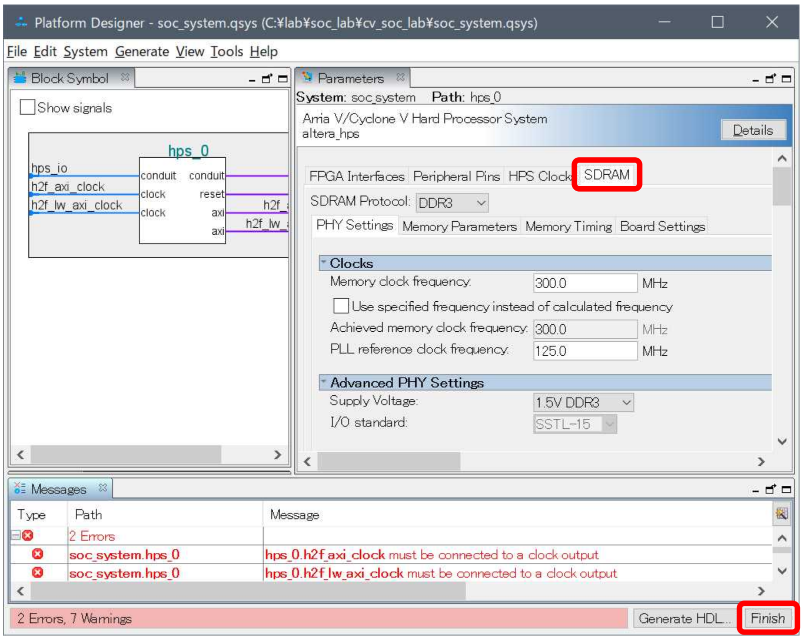
Figure 3-26. Configuring HPS to FPGA Clock

3-5. Step 5: Configuring SDRAM

The ***SDRAM*** tab provides options for configuring parameters for the SDRAM controller on the HPS side and the connected DDR. Within the ***SDRAM*** tab, there are four additional tabs (***PHY Settings, Memory Parameters, Memory Timing, Board Settings***) for SDRAM configuration.

____ 1. Click at the bottom of the ***Arria V/Cyclone V Hard Processor System*** window.

This action adds the HPS component to the Platform Designer system (required to display the Presets window in the next step).



□ Show signals

System: soc_system Path: hps_0
Arria V/Cyclone V Hard Processor System
altera_hps

FPGA Interfaces | Peripheral Pins | HPS Clocks | SDRAM

SDRAM Protocol: DDR3

PHY Settings | Memory Parameters | Memory Timing | Board Settings

Clocks
Memory clock frequency: 300.0 MHz
□ Use specified frequency instead of calculated frequency
Achieved memory clock frequency: 300.0 MHz
PLL reference clock frequency: 125.0 MHz

Advanced PHY Settings
Supply Voltage: 1.5V DDR3
I/O standard: SSTL-15

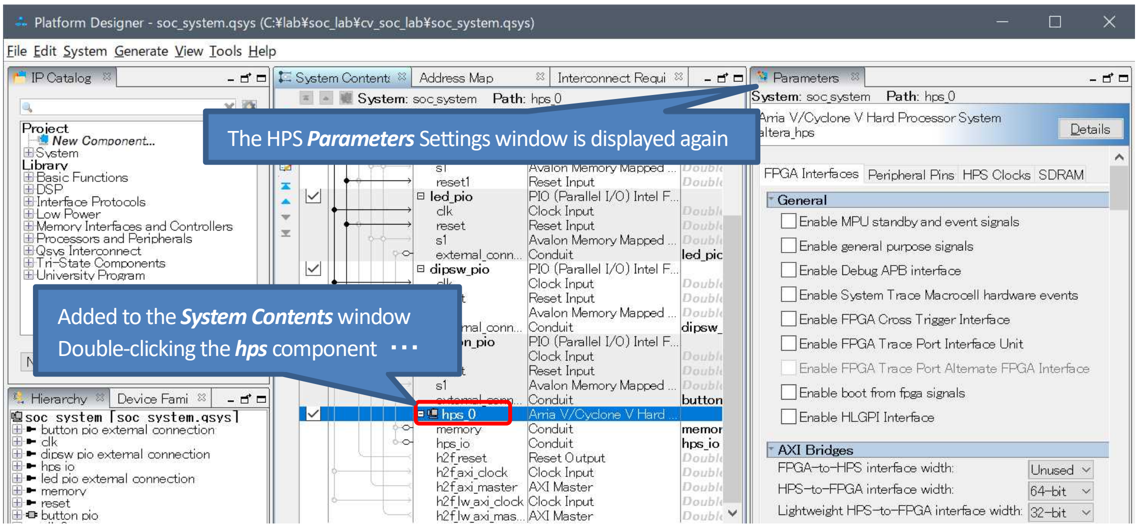
Type	Path	Message
⊗	2 Errors	
⊗	soc_system.hps_0	hps_0.h2f_axi_clock must be connected to a clock output
⊗	soc_system.hps_0	hps_0.h2f_lw_axi_clock must be connected to a clock output

2 Errors, 7 Warnings

Figure 3-27. Preparing to display the Parameters window

____ 2. Double-click the HPS component in the ***System Contents*** window to display the HPS option settings in the ***Parameters*** window again.

This is necessary to display the ***Preset*** window in the next step.



The HPS ***Parameters*** Settings window is displayed again

Added to the ***System Contents*** window
Double-clicking the ***hps*** component ・・・

System: soc_system Path: hps_0
Arria V/Cyclone V Hard Processor System
altera_hps

FPGA Interfaces | Peripheral Pins | HPS Clocks | SDRAM

General
□ Enable MPU standby and event signals
□ Enable general purpose signals
□ Enable Debug APB interface
□ Enable System Trace Macrocell hardware events
□ Enable FPGA Cross Trigger Interface
□ Enable FPGA Trace Port Interface Unit
□ Enable FPGA Trace Port Alternate FPGA Interface
□ Enable boot from fpga signals
□ Enable HLGPI Interface

AXI Bridges
FPGA-to-HPS interface width: Unused
HPS-to-FPGA interface width: 64-bit
Lightweight HPS-to-FPGA interface width: 32-bit

Figure 3-28. Redisplay the HPS parameter setting window

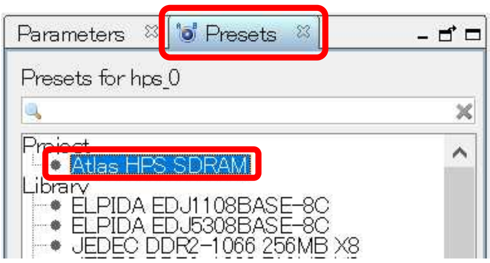
____ 3. Click the ***SDRAM*** tab in the ***Parameters*** window.

This time, we will use the preset SDRAM on the Atlas-SoC board.

Make sure that the ***Presets*** window is displayed.

> ⓘ **Info:**
>
> If the ***Presets*** window is not displayed, select Platform Designer ***View*** menu => ***Presets*** to display it.
>
> If it is not displayed, select Platform Designer ***View*** menu => ***Reset to System Layout*** and select ***Preset*** again.

____ 4. Select the ***Atlas_HPS_SDRAM preset in the Presets*** window.



Parameters | Presets

Presets for hps_0

Project
- Atlas_HPS_SDRAM

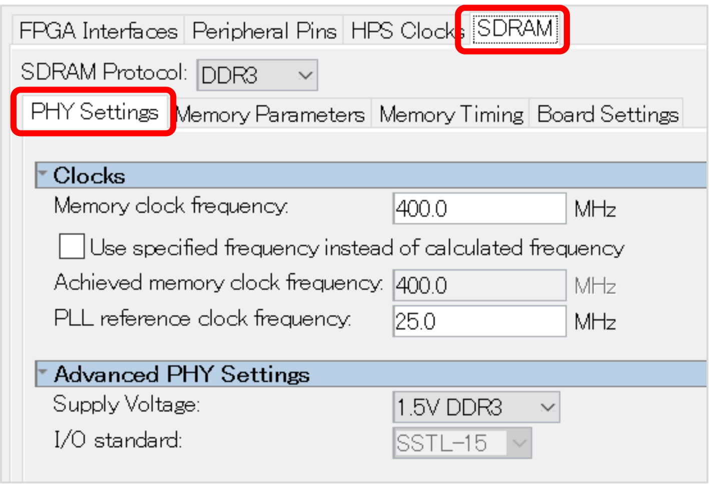
Library
- ELPIDA EDJ1108BASE-8C
- ELPIDA EDJ5308BASE-8C
- JEDEC DDR2-1066 256MB X8

Figure 3-29. Select Preset

____ 5. When you click Apply, ***Atlas_HPS_SDRAM*** should be highlighted in bold. If it is, the settings are applied correctly.

____ 6. If the ***SDRAM*** tab is not visible, click the ***SDRAM*** tab.

____ 7. Click the ***PHY Settings*** tab and verify that the settings are as shown below.



FPGA Interfaces | Peripheral Pins | HPS Clocks | SDRAM

SDRAM Protocol: DDR3

PHY Settings | Memory Parameters | Memory Timing | Board Settings

Clocks

Memory clock frequency: 400.0 MHz

□ Use specified frequency instead of calculated frequency

Achieved memory clock frequency: 400.0 MHz

PLL reference clock frequency: 25.0 MHz

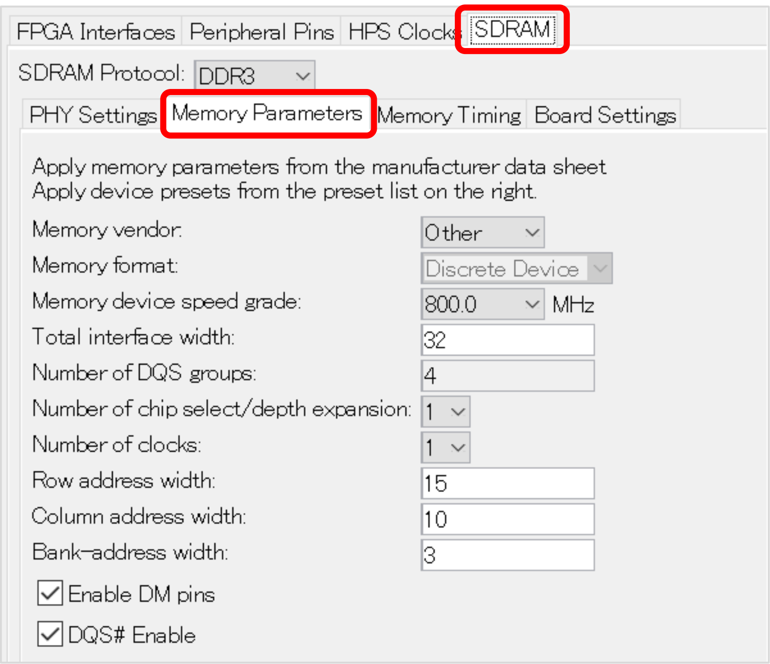
Advanced PHY Settings

Supply Voltage: 1.5V DDR3

I/O standard: SSTL-15

Figure 3-30. Verify PHY Settings

____ 8. Click the ***Memory Parameters*** tab and verify that the settings are as shown below.



FPGA Interfaces | Peripheral Pins | HPS Clocks | SDRAM

SDRAM Protocol: DDR3

PHY Settings | Memory Parameters | Memory Timing | Board Settings

Apply memory parameters from the manufacturer data sheet
Apply device presets from the preset list on the right.

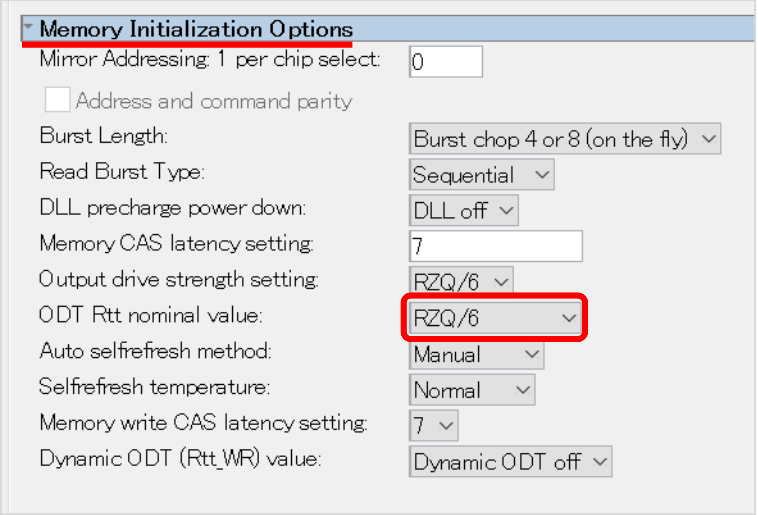
Memory vendor:	Other
Memory format:	Discrete Device
Memory device speed grade:	800.0 MHz
Total interface width:	32
Number of DQS groups:	4
Number of chip select/depth expansion:	1
Number of clocks:	1
Row address width:	15
Column address width:	10
Bank-address width:	3

☑ Enable DM pins

☑ DQS# Enable

Figure 3-31. Memory Parameters

____ 9. Scroll down to the ***Memory Initialization Options*** section and verify that the ***ODT Rtt nominal value*** is set to ***RZQ/6***.

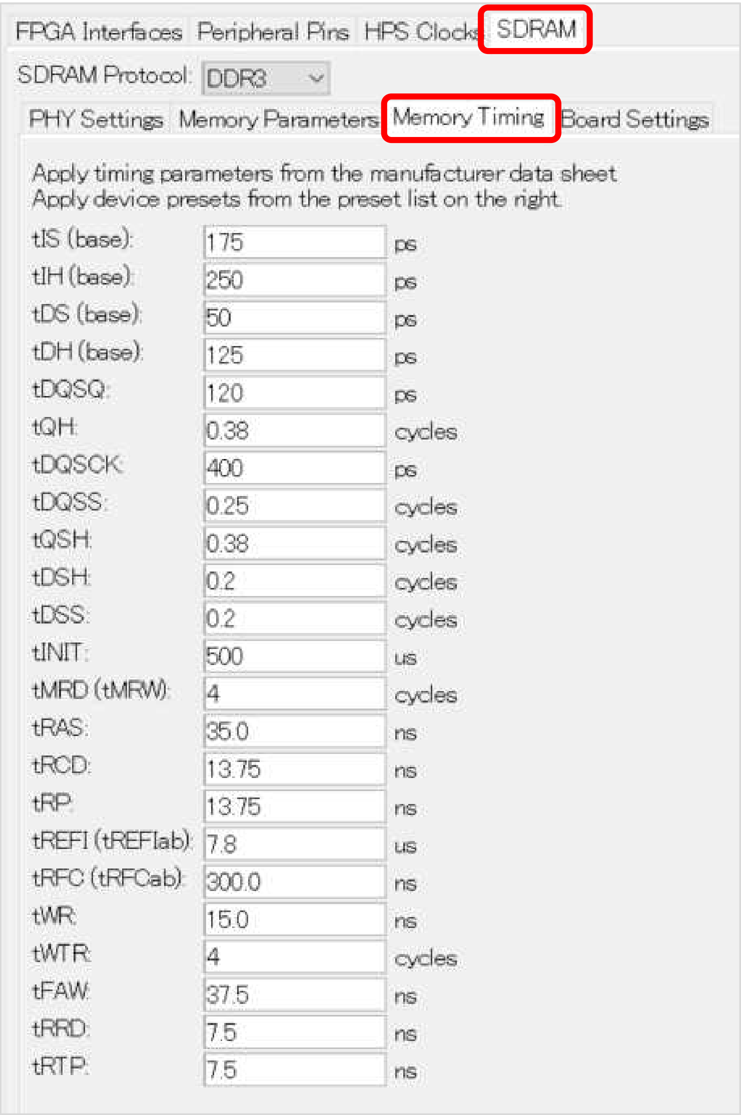


Memory Initialization Options

Mirror Addressing: 1 per chip select:	0
□ Address and command parity	
Burst Length:	Burst chop 4 or 8 (on the fly)
Read Burst Type:	Sequential
DLL precharge power down:	DLL off
Memory CAS latency setting:	7
Output drive strength setting:	RZQ/6
ODT Rtt nominal value:	RZQ/6
Auto selfrefresh method:	Manual
Selfrefresh temperature:	Normal
Memory write CAS latency setting:	7
Dynamic ODT (Rtt_WR) value:	Dynamic ODT off

Figure 3-32. Memory Initialization Options

____ 10. Click on the ***Memory Timing*** tab and verify that the settings are as shown below.



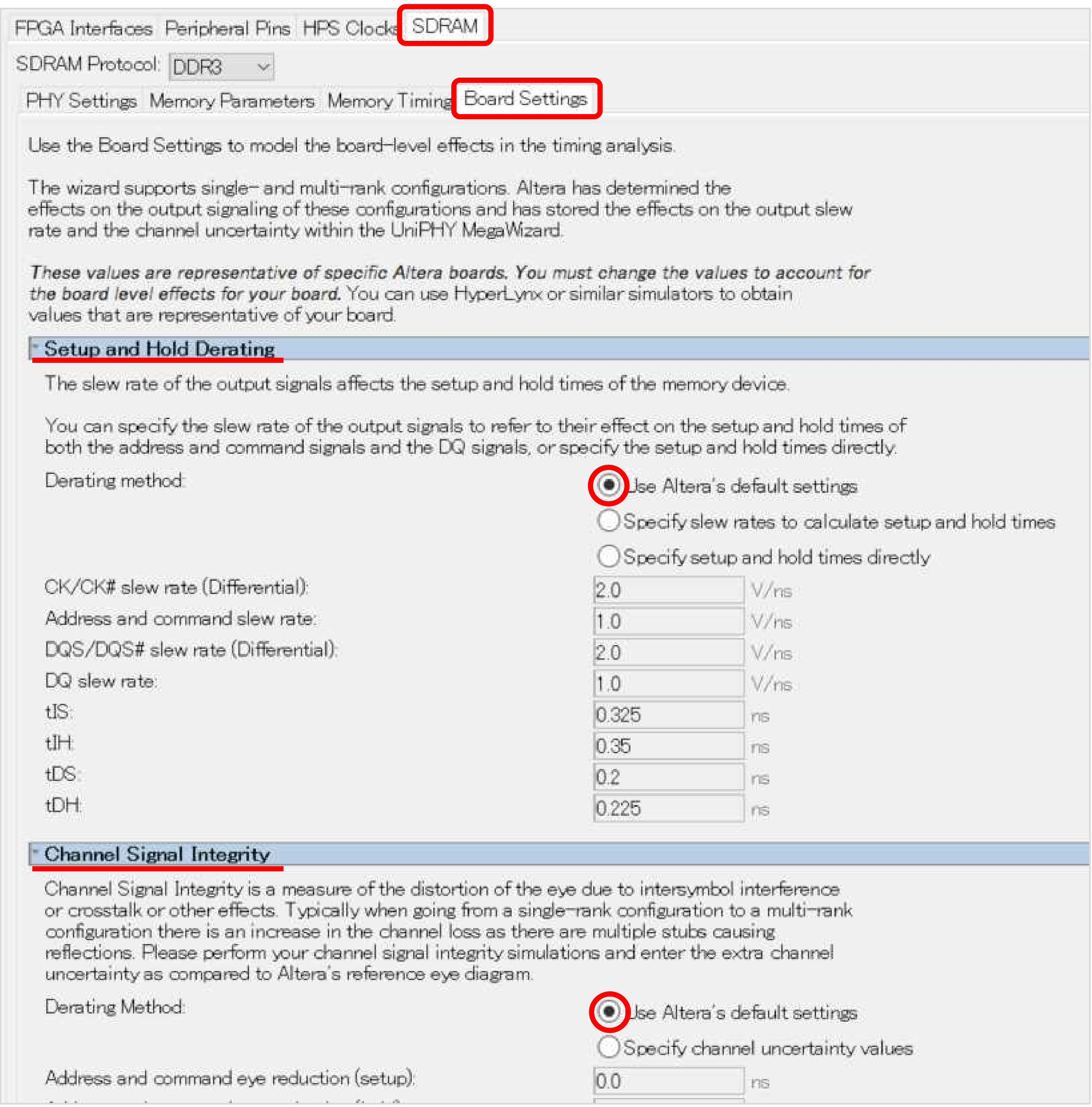
SDRAM Protocol: DDR3

Apply timing parameters from the manufacturer data sheet
Apply device presets from the preset list on the right

tIS (base):	175	ps
tIH (base):	250	ps
tDS (base):	50	ps
tDH (base):	125	ps
tDQSQ:	120	ps
tQH:	0.38	cycles
tDQSCK:	400	ps
tDQSS:	0.25	cycles
tQSH:	0.38	cycles
tDSH:	0.2	cycles
tDSS:	0.2	cycles
tINIT:	500	us
tMRD (tMRW):	4	cycles
tRAS:	35.0	ns
tRCD:	13.75	ns
tRP:	13.75	ns
tREFI (tREFIab):	7.8	us
tRFC (tRFCab):	300.0	ns
tWR:	15.0	ns
tWTR:	4	cycles
tFAW:	37.5	ns
tRRD:	7.5	ns
tRTP:	7.5	ns

Figure 3-33. Memory Timing

____ 11. Click on the ***Board Settings*** tab and verify that ***Use Altera's default settings*** is selected in the ***Setup and Hold Derating*** and ***Channel Signal Integrity*** sections.



SDRAM Protocol: DDR3

Use the Board Settings to model the board-level effects in the timing analysis.

The wizard supports single- and multi-rank configurations. Altera has determined the effects on the output signaling of these configurations and has stored the effects on the output slew rate and the channel uncertainty within the UniPHY MegaWizard.

These values are representative of specific Altera boards. You must change the values to account for the board level effects for your board. You can use HyperLynx or similar simulators to obtain values that are representative of your board.

Setup and Hold Derating

The slew rate of the output signals affects the setup and hold times of the memory device.

You can specify the slew rate of the output signals to refer to their effect on the setup and hold times of both the address and command signals and the DQ signals, or specify the setup and hold times directly.

Derating method: ● Use Altera's default settings ○ Specify slew rates to calculate setup and hold times ○ Specify setup and hold times directly

CK/CK# slew rate (Differential):	2.0	V/ns
Address and command slew rate:	1.0	V/ns
DQS/DQS# slew rate (Differential):	2.0	V/ns
DQ slew rate:	1.0	V/ns
tIS:	0.325	ns
tIH:	0.35	ns
tDS:	0.2	ns
tDH:	0.225	ns

Channel Signal Integrity

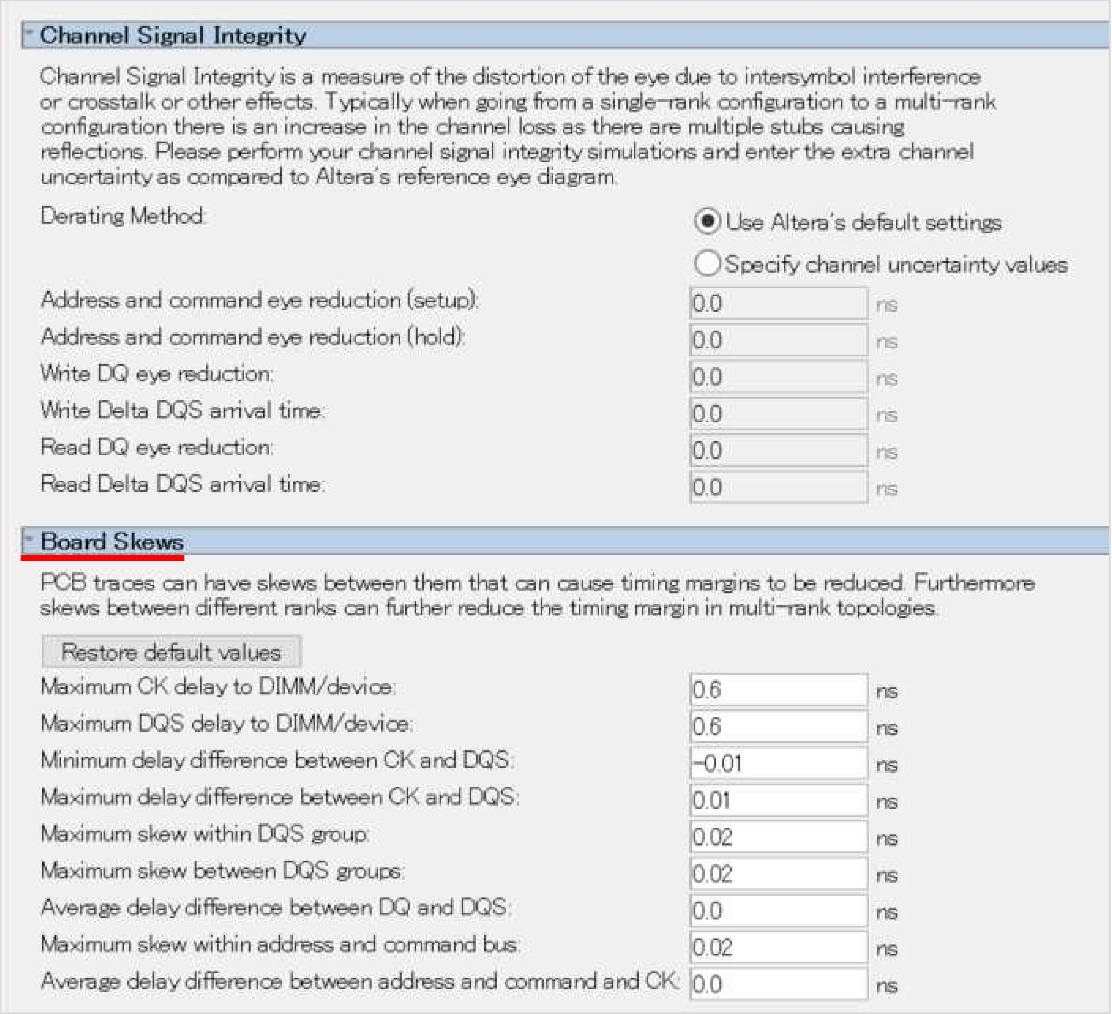
Channel Signal Integrity is a measure of the distortion of the eye due to intersymbol interference or crosstalk or other effects. Typically when going from a single-rank configuration to a multi-rank configuration there is an increase in the channel loss as there are multiple stubs causing reflections. Please perform your channel signal integrity simulations and enter the extra channel uncertainty as compared to Altera's reference eye diagram.

Derating Method: ● Use Altera's default settings ○ Specify channel uncertainty values

Address and command eye reduction (setup): 0.0 ns

Figure 3-34. Board Settings (1)

____ 12. Scroll down to the ***Board Skew*** section and verify that the board skew is as shown below.



Channel Signal Integrity

Channel Signal Integrity is a measure of the distortion of the eye due to intersymbol interference or crosstalk or other effects. Typically when going from a single-rank configuration to a multi-rank configuration there is an increase in the channel loss as there are multiple stubs causing reflections. Please perform your channel signal integrity simulations and enter the extra channel uncertainty as compared to Altera's reference eye diagram.

Derating Method: ● Use Altera's default settings ○ Specify channel uncertainty values

Address and command eye reduction (setup):	0.0	ns
Address and command eye reduction (hold):	0.0	ns
Write DQ eye reduction:	0.0	ns
Write Delta DQS arrival time:	0.0	ns
Read DQ eye reduction:	0.0	ns
Read Delta DQS arrival time:	0.0	ns

Board Skews

PCB traces can have skews between them that can cause timing margins to be reduced. Furthermore skews between different ranks can further reduce the timing margin in multi-rank topologies.

Maximum CK delay to DIMM/device:	0.6	ns
Maximum DQS delay to DIMM/device:	0.6	ns
Minimum delay difference between CK and DQS:	-0.01	ns
Maximum delay difference between CK and DQS:	0.01	ns
Maximum skew within DQS group:	0.02	ns
Maximum skew between DQS groups:	0.02	ns
Average delay difference between DQ and DQS:	0.0	ns
Maximum skew within address and command bus:	0.02	ns
Average delay difference between address and command and CK:	0.0	ns

Figure 3-35. Board Settings (2)

____ 13. Select ***File*** menu => ***Save*** in Platform Designer to save the HPS parameter settings specified in the previous steps.

3-6. Step 6: Setting the HPS Clock and Export Signal

In this step, you will set the HPS H2F bridge clock and the LWH2F bridge clock.

The clocks set here are the clocks on the FPGA side of each bridge (h2f_axi_clk and h2f_lw_axi_clk shown below). The clocks on the HPS side are l3_main_clk and l4_mp_clk, which are set in "3-4. Step 4: Set the HPS Clock," and are different from the clocks set here. The difference in clocks is absorbed within the bridge.

The HPS export signal is also set. This export signal is used for communication outside Platform Designer. For example, it is used to connect Platform Designer to other logic on the FPGA and to place it on pins.

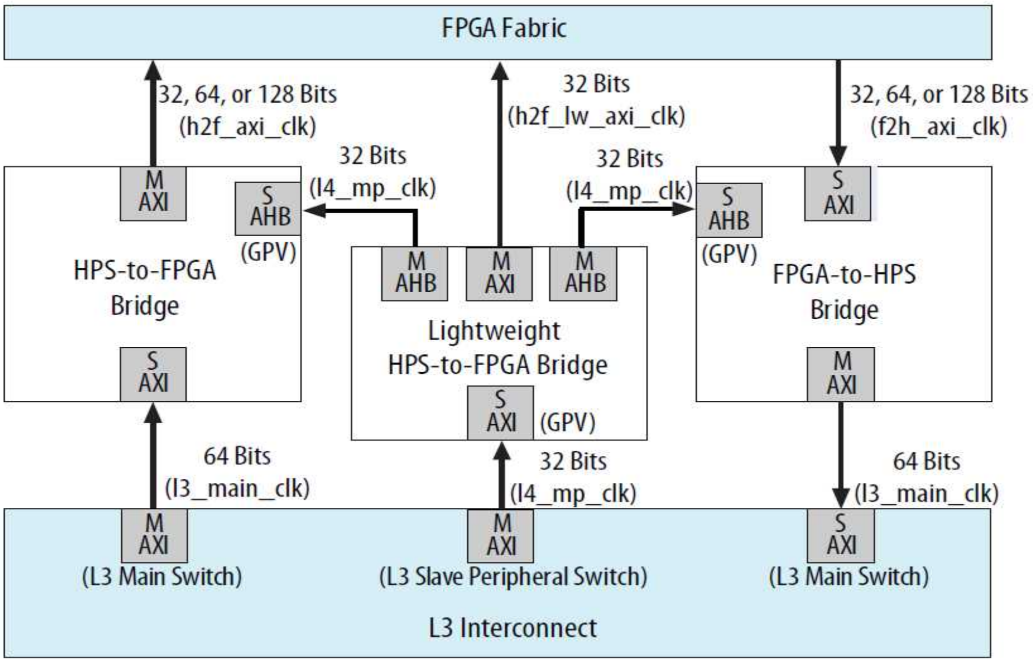


Figure 3-36. Clock between HPS and FPGA

____ 1. Go to the ***System Contents*** tab.

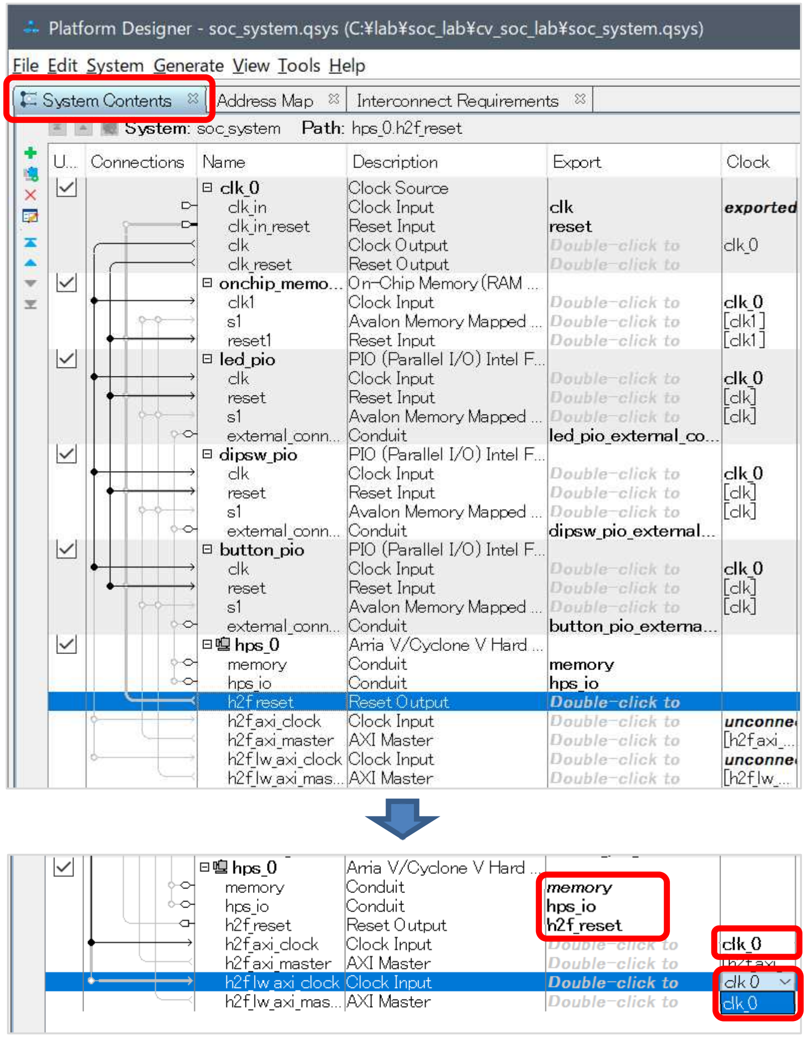
____ 2. By specifying the signal name in the ***Export*** column, you can route the signal to the outside of the Platform Designer system. Make sure that the ***hps_io*** port of the HPS component you just added is exported with the signal name ***hps_io***.

____ 3. Similarly, make sure that the ***memory*** port of the HPS component is exported with the signal name ***memory***. This is the IO of the SDRAM on the HPS side that you just configured.

____ 4. Export the ***h2f_ rest*** of the HPS component. Double-click the ***Export*** column of ***H2f_reset***.
Rename it to "***h2f_ reset***" and press Enter to export.

____ 5. Configure the Clock Input interface ***h2f_axi_clock*** on the HPS. Select ***clk_0*** from the pull-down menu in the ***Clock*** column next to H2f_axi_clock and connect clk_0 to ***h2f_axi_clock***.

____ 6. Configure the Clock Input interface ***h2f_lw_axi_clock*** on the HPS in the same way. Select ***clk_0*** from the pull-down menu in the ***Clock*** column next to H2f_lw_axi_clock and connect ***clk_0*** to ***h2f_lw_axi_clock***.



Platform Designer - soc_system.qsys (C:¥lab¥soc_lab¥cv_soc_lab¥soc_system.qsys)

File Edit System Generate View Tools Help
System Contents
Address Map
Interconnect Requirements

System: soc_system Path: hps_0.h2f_reset

U...	Connections	Name	Description	Export	Clock
☑		⊟ clk_0	Clock Source		
		clk_in	Clock Input	clk	exported
		clk_in_reset	Reset Input	reset	
		clk	Clock Output	Double-click to	clk_0
		clk_reset	Reset Output	Double-click to	
☑		⊟ onchip_memo...	On-Chip Memory (RAM ...		
		clk1	Clock Input	Double-click to	clk_0
		s1	Avalon Memory Mapped ...	Double-click to	[clk1]
		reset1	Reset Input	Double-click to	[clk1]
☑		⊟ led_pio	PIO (Parallel I/O) Intel F...		
		clk	Clock Input	Double-click to	clk_0
		reset	Reset Input	Double-click to	[clk]
		s1	Avalon Memory Mapped ...	Double-click to	[clk]
		external_conn...	Conduit	led_pio_external_co...	
☑		⊟ dipsw_pio	PIO (Parallel I/O) Intel F...		
		clk	Clock Input	Double-click to	clk_0
		reset	Reset Input	Double-click to	[clk]
		s1	Avalon Memory Mapped ...	Double-click to	[clk]
		external_conn...	Conduit	dipsw_pio_external...	
☑		⊟ button_pio	PIO (Parallel I/O) Intel F...		
		clk	Clock Input	Double-click to	clk_0
		reset	Reset Input	Double-click to	[clk]
		s1	Avalon Memory Mapped ...	Double-click to	[clk]
		external_conn...	Conduit	button_pio_externa...	
☑		⊟ hps_0	Arria V/Cyclone V Hard ...		
		memory	Conduit	memory	
		hps_io	Conduit	hps_io	
		h2f_reset	Reset Output	Double-click to	
		h2f_axi_clock	Clock Input	Double-click to	unconne
		h2f_axi_master	AXI Master	Double-click to	[h2f_axi_...
		h2f_lw_axi_clock	Clock Input	Double-click to	unconne
		h2f_lw_axi_mas...	AXI Master	Double-click to	[h2f_lw_...

☑		⊟ hps_0	Arria V/Cyclone V Hard ...		
		memory	Conduit	memory	
		hps_io	Conduit	hps_io	
		h2f_reset	Reset Output	h2f_reset	
		h2f_axi_clock	Clock Input	Double-click to	clk_0
		h2f_axi_master	AXI Master	Double-click to	[h2f_axi...
		h2f_lw_axi_clock	Clock Input	Double-click to	clk_0 clk_0
		h2f_lw_axi_mas...	AXI Master	Double-click to	

Figure 3-37. Setting the clock and export signal

Now that ***clk0*** is connected, the two error messages in the ***Message Window*** in Figure 3-23. should disappear.

3-7. Step 7: Connecting HPS Components to Other Components

In this step, you will connect the HPS components added to the Platform Designer system and the components already implemented in the Platform Designer system. Since the FPGA will operate at clk_0 (50 MHz), clk_0 is already connected to each component.

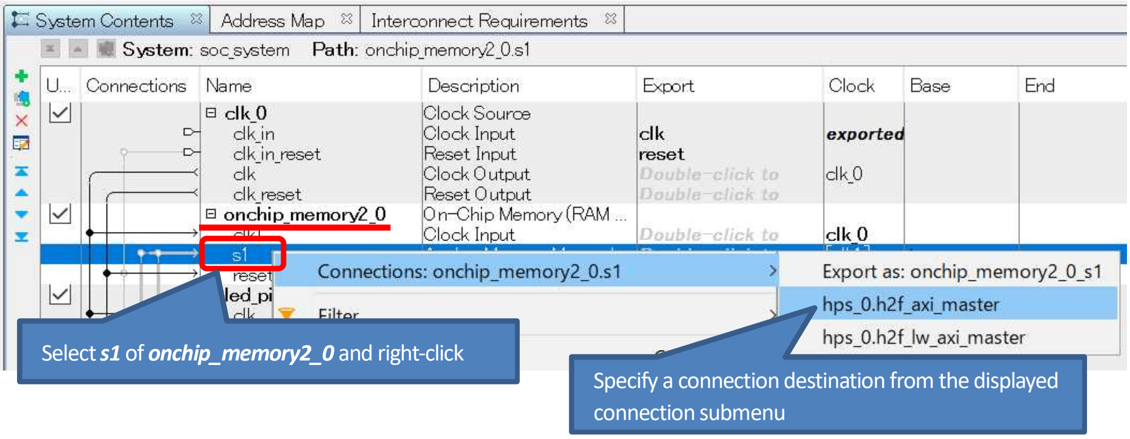
____ 1. Verify that the Clock Input interface of the ***Onchip_memory2_0*** component is connected to ***clk_0***.

____ 2. Verify that the Clock Input interface of the ***Led_pio*** component is connected to ***clk_0***.

____ 3. Verify that the Clock Input interface of the ***Dipsw_pio*** component is connected to ***clk_0***.

____ 4. Verify that the Clock Input interface of the ***Button_pio*** component is connected to ***clk_0***.

____ 5. Select ***s1*** of ***onchip_memory2_0***, then right-click and select ***hps_0.h2f_axi_master*** from the connection submenu. This connects the ***s1*** interface of the ***onchip_memory2_0*** component to the **HPS h2f_axi_master**. This configuration allows the Arm® processor to access onchip_memory on the FPGA side.


System Contents
Address Map
Interconnect Requirements
Connections: onchip_memory2_0.s1
Filter
Export as: onchip_memory2_0_s1
hps_0.h2f_axi_master
hps_0.h2f_lw_axi_master

System: soc_system   Path: onchip_memory2_0.s1

U...	Connections	Name	Description	Export	Clock	Base	End
☑		⊟ clk_0	Clock Source				
		clk_in	Clock Input	clk	exported		
		clk_in_reset	Reset Input	reset			
		clk	Clock Output	Double-click to	clk_0		
		clk_reset	Reset Output	Double-click to			
☑		⊟ onchip_memory2_0	On-Chip Memory (RAM ...				
		clk1	Clock Input	Double-click to	clk_0		
		s1					
		reset					
☑		led_pi					
		clk					

Select ***s1*** of ***onchip_memory2_0*** and right-click

Specify a connection destination from the displayed connection submenu

Figure 3-38. Connecting components

____ 6. Similarly, right-click ***s1*** of ***button_pio*** and select ***hps_0.h2f_lw_axi_master*** from the connection submenu. This connects the ***s1*** interface of the ***button_pio*** component to the **HPS h2f_lw_axi_master**.

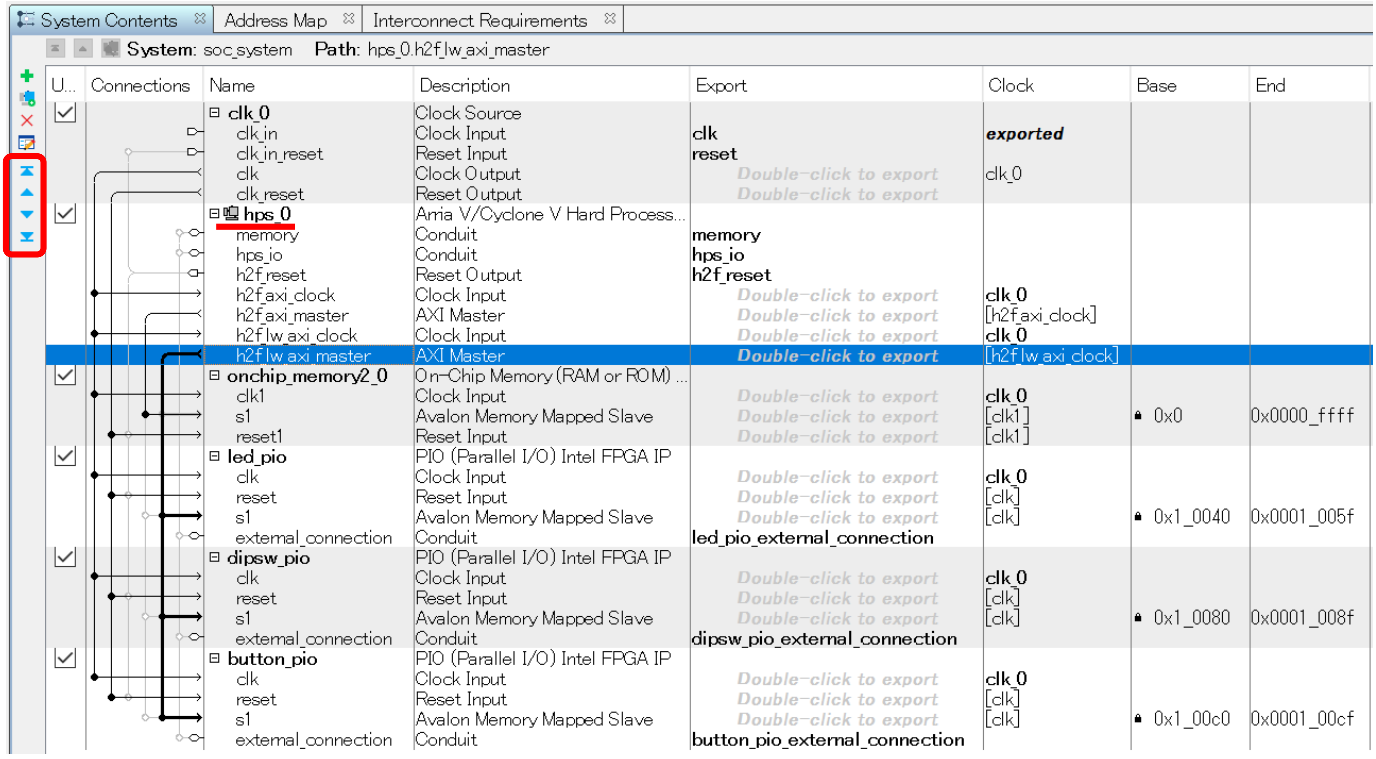
Notice that the connection destination is ***h2f_lw_axi_master***. The same applies to each subsequent PIO component.

____ 7. Similarly, right-click ***s1*** of ***dipsw_pio*** and select ***hps_0.h2f_lw_axi_master*** from the Connect submenu. This connects the ***s1*** interface of the ***dipsw_pio*** component to the **HPS h2f_lw_axi_master**.

____ 8. Similarly, right-click ***s1*** of ***led_pio*** and select ***hps_0.h2f_lw_axi_master*** from the Connect submenu. This connects the ***s1*** interface of the ***led_pio*** component to the **HPS h2f_lw_axi_master**.

____ 9. Select the HPS component and use the **up and down** ⤓▼▲⤒ buttons in the Platform Designer toolbar on the left side of the ***System Contents*** window to move the HPS component below clk_0. Use to move the HPS component below ***clk_0***.

The Platform Designer system after configuration is complete is shown below.



System Contents ⊠ | Address Map ⊠ | Interconnect Requirements ⊠

System: soc_system **Path:** hps_0.h2f_lw_axi_master

U...	Connections	Name	Description	Export	Clock	Base	End
☑		⊟ clk_0	Clock Source				
		clk_in	Clock Input	clk	*exported*		
		clk_in_reset	Reset Input	reset			
		clk	Clock Output	Double-click to export	clk_0		
		clk_reset	Reset Output	Double-click to export			
☑		⊟ hps_0	Arria V/Cyclone V Hard Process...				
		memory	Conduit	memory			
		hps_io	Conduit	hps_io			
		h2f_reset	Reset Output	h2f_reset			
		h2f_axi_clock	Clock Input	Double-click to export	clk_0		
		h2f_axi_master	AXI Master	Double-click to export	[h2f_axi_clock]		
		h2f_lw_axi_clock	Clock Input	Double-click to export	clk_0		
		h2f_lw_axi_master	AXI Master	Double-click to export	[h2f_lw_axi_clock]		
☑		⊟ onchip_memory2_0	On-Chip Memory (RAM or ROM) ...				
		clk1	Clock Input	Double-click to export	clk_0		
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	🔒 0x0	0x0000_ffff
		reset1	Reset Input	Double-click to export	[clk1]		
☑		⊟ led_pio	PIO (Parallel I/O) Intel FPGA IP				
		clk	Clock Input	Double-click to export	clk_0		
		reset	Reset Input	Double-click to export	[clk]		
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	🔒 0x1_0040	0x0001_005f
		external_connection	Conduit	led_pio_external_connection			
☑		⊟ dipsw_pio	PIO (Parallel I/O) Intel FPGA IP				
		clk	Clock Input	Double-click to export	clk_0		
		reset	Reset Input	Double-click to export	[clk]		
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	🔒 0x1_0080	0x0001_008f
		external_connection	Conduit	dipsw_pio_external_connection			
☑		⊟ button_pio	PIO (Parallel I/O) Intel FPGA IP				
		clk	Clock Input	Double-click to export	clk_0		
		reset	Reset Input	Double-click to export	[clk]		
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	🔒 0x1_00c0	0x0001_00cf
		external_connection	Conduit	button_pio_external_connection			

Figure 3-39. The Platform Designer system after configuration is complete

Consider accessing led_pio.

If you look at the second ***Base*** column from the right of led _pio, it is set to 0x0001_0040. This is the offset address of led_pio in Platform Designer. Earlier ____ **8** The master to access led_pio is **HPS h2f_lw_ axis _master**. Since the base address of the Lightweight HPS-to-FPGA bridge was 0xFF20_0000, accessing this led_pio would be:

Bridge base address (0xFF20_0000) + Platform Designer offset address (0x0001_0040) = **0xFF21_0040**

Other components can be considered in the same way, and dipsw_pio would be **0xFF21_0080**.

Next, consider accessing onchip_memory.

The base address of the **HPS h2f_axi_master** bridge, which is another path from the HPS to the FPGA, is 0xC000_0000. In this case, the Platform Designer offset address of onchip_memory connected to the **HPS h2f_axi_master** is 0x0, so in this case, the base address of the bridge (0xC000_0000) is the base address to access onchip_memory.

3-8. Step 8: Connect Resets and Assign Base Addresses

This step provides a bulk reset connection and automatic base address assignment.

____ 1. Select Platform Designer ***System*** menu => ***Create Global Reset Network*** to connect all reset interfaces in the design in **bulk**.

____ 2. Automatically assign base addresses for all components so that there are no duplicate addresses. Select ***System*** menu => ***Assign Base Addresses***.

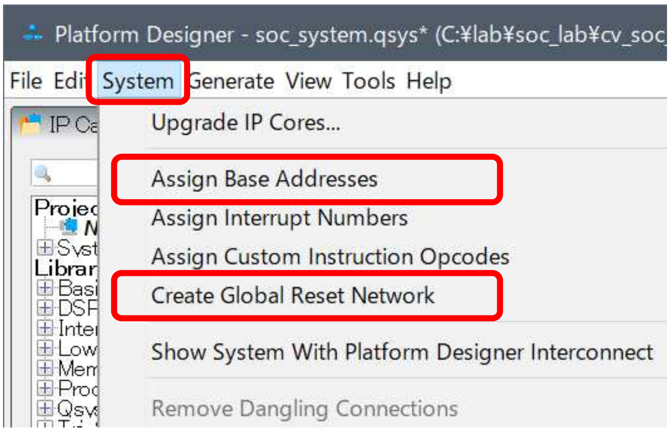
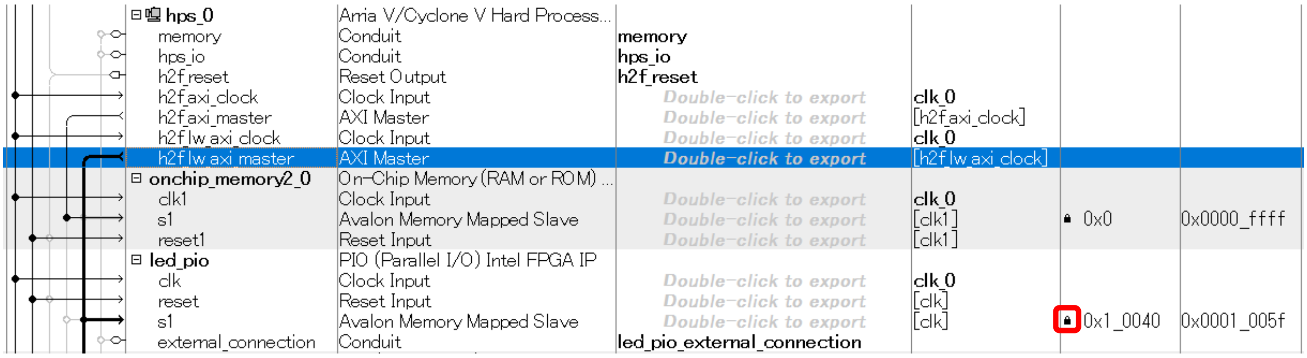

Platform Designer - soc_system.qsys* (C:¥lab¥soc_lab¥cv_soc
File
Edi
System
Generate
View
Tools
Help
IP Ca
Projec
N
Syst
Librar
Basi
DSP
Inte
Low
Mem
Proc
Qsys
Upgrade IP Cores...
Assign Base Addresses
Assign Interrupt Numbers
Assign Custom Instruction Opcodes
Create Global Reset Network
Show System With Platform Designer Interconnect
Remove Dangling Connections

Figure 3-40. Reset Bulk Connect and Assign Base Addresses Automatically

If you did ***Assign Base Addresses***, nothing happened.

In this exercise, the base addresses for each peripheral were fixed beforehand, so they were not automatically assigned.

You can lock the address setting by using the lock symbol 🔒 next to the base address, as shown in Figure 3-41.. Each click toggles whether the address is locked or not. If you want to lock the address, lock it with the lock symbol after setting the address. You can also lock it by selecting Platform Designer's ***Edit*** menu => ***Lock Base Address***.



⊟ hps_0	Arria V/Cyclone V Hard Process...				
memory	Conduit	memory			
hps_io	Conduit	hps_io			
h2f_reset	Reset Output	h2f_reset			
h2f_axi_clock	Clock Input	Double-click to export	clk_0		
h2f_axi_master	AXI Master	Double-click to export	[h2f_axi_clock]		
h2f_lw_axi_clock	Clock Input	Double-click to export	clk_0		
h2f_lw_axi_master	AXI Master	Double-click to export	[h2f_lw_axi_clock]		
⊟ onchip_memory2_0	On-Chip Memory (RAM or ROM) ...				
clk1	Clock Input	Double-click to export	clk_0		
s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	🔒 0x0	0x0000_ffff
reset1	Reset Input	Double-click to export	[clk1]		
⊟ led_pio	PIO (Parallel I/O) Intel FPGA IP				
clk	Clock Input	Double-click to export	clk_0		
reset	Reset Input	Double-click to export	[clk]		
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	🔒 0x1_0040	0x0001_005f
external_connection	Conduit	led_pio_external_connection			

Figure 3-41. Fixing the Base Address

3-9. Step 9: Checking the Platform Designer System

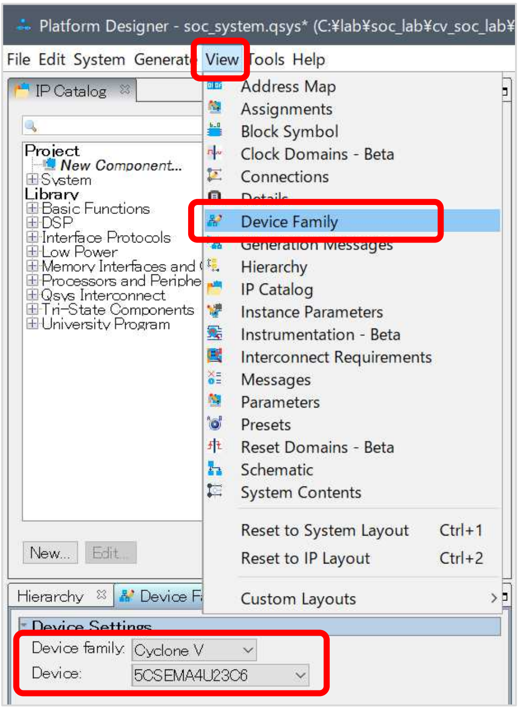
____ 1. Verify that the Platform Designer system you designed is as shown in "Table 3-1. Connectivity of the Platform Designer system after design" below. Also refer to "Figure 3-39. The Platform Designer system after **configuration is complete**."

To be consistent with the Quartus® Prime project for the exercise, make sure that the export signals are properly exported and named correctly. You can use any signal name in the actual design. There is no regulation on the order of components.

Table 3-1. Connectivity of the Platform Designer system after design

Component	Port name	Connection
clk_0	clk_in	Export as **clk**
	clk_in_reset	Export as **reset**
	clk	Connect to all components
	clk_reset	Connect to all components except hps_0
led_pio	external_connection	Export as led_pio_external_connection
dipsw_pio	external_connection	Export as **dipsw_pio_external_connection**
button_pio	external_connection	Export as **button_pio_external_connection**
hps_0	h2f_axi_master	Connect to onchip_memory2_0.s1
	h2f_lw_axi_master	Connect to led_pio.s1
		Connect to dipsw_pio.s1
		Connect to button_pio.s1

____ 2. Select Platform Designer ***View*** menu => ***Device Family*** and confirm that the ***Device Family*** is ***Cyclone V***. The Device is 5CSEMA4U23C6 for the **Atlas-SoC board** and 5CSEBA6U23I7DK for the **DE10 Nano board**.

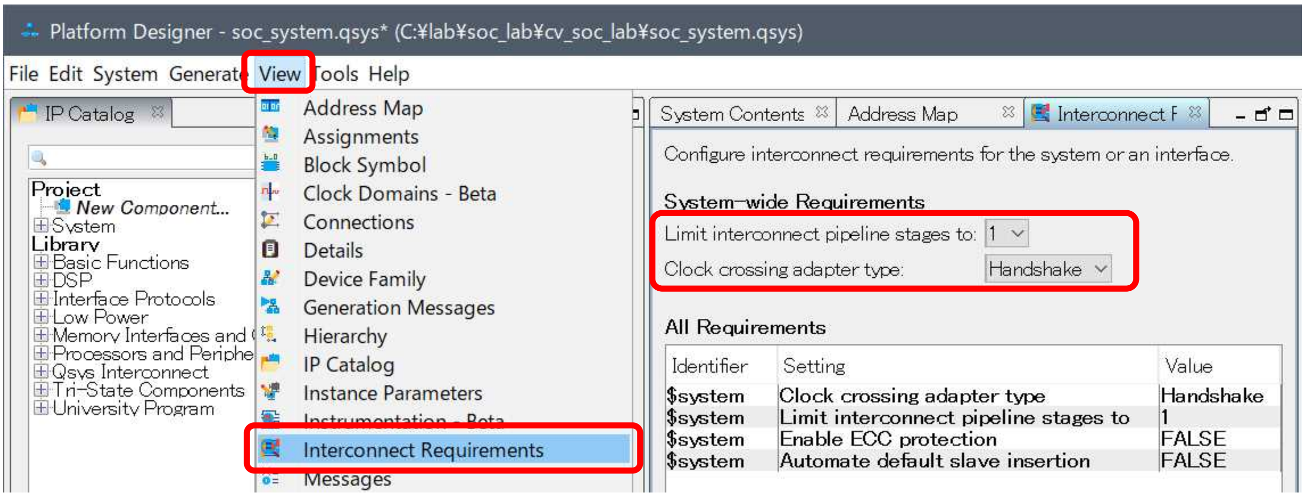


Device family: Cyclone V
Device: 5CSEMA4U23C6

Figure 3-42. Device Family tab

____ 3. Select Platform Designer ***View*** menu Interconnect Requirements and set ***Limit interconnect pipeline stages*** to 1. Increasing the number of stages allows more timing, but also increases the logic of the FPGA. Make sure the ***Clock crossing adapter*** type is set to ***Handshake***. Increasing the number of stages allows more timing, but also increases the logic of the FPGA.

Make sure that the ***Clock crossing adapter type*** is set to ***Handshake***.



Configure interconnect requirements for the system or an interface.

System-wide Requirements

Limit interconnect pipeline stages to: 1

Clock crossing adapter type: Handshake

All Requirements

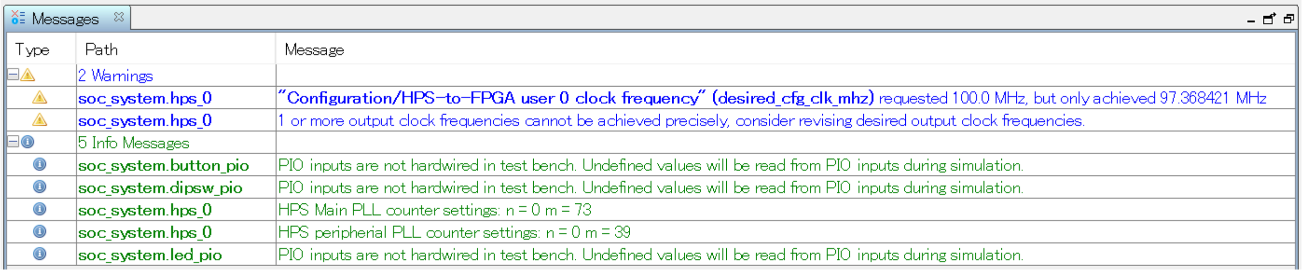
Identifier	Setting	Value
$system	Clock crossing adapter type	Handshake
$system	Limit interconnect pipeline stages to	1
$system	Enable ECC protection	FALSE
$system	Automate default slave insertion	FALSE

Figure 3-43. Set project parameters

3-10. Step 10: Generate the Platform Designer system

Generate the completed Platform Designer system.

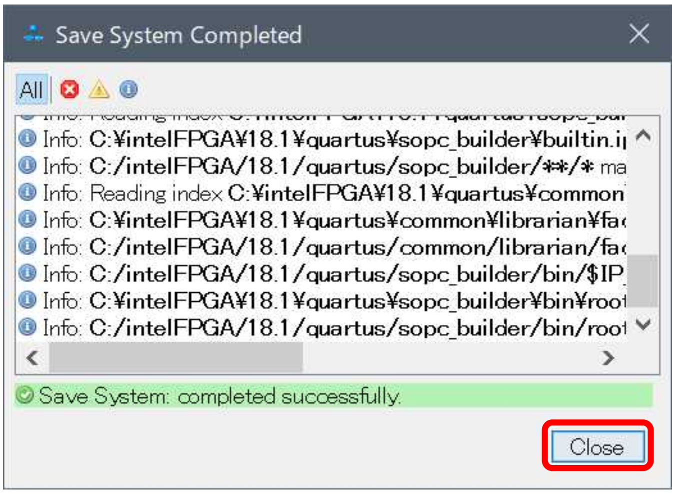
____ 1. Check the *Message* box on the ***System Contents*** tab for any remaining errors. If there are any errors, you must fix them before continuing. Ignore the blue warning at this time.



Type	Path	Message
⊟⚠	2 Warnings	
⚠	soc_system.hps_0	"Configuration/HPS-to-FPGA user 0 clock frequency" (desired_cfg_clk_mhz) requested 100.0 MHz, but only achieved 97.368421 MHz
⚠	soc_system.hps_0	1 or more output clock frequencies cannot be achieved precisely, consider revising desired output clock frequencies.
⊟ⓘ	5 Info Messages	
ⓘ	soc_system.button_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
ⓘ	soc_system.dipsw_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
ⓘ	soc_system.hps_0	HPS Main PLL counter settings: n = 0 m = 73
ⓘ	soc_system.hps_0	HPS peripherial PLL counter settings: n = 0 m = 39
ⓘ	soc_system.led_pio	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

Figure 3-44. Message window display

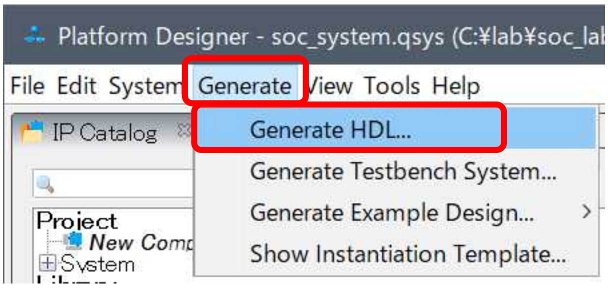
____ 2. Select ***File*** menu => ***Save*** to save the Platform Designer system. [***Close***] when ***Save System Completed*** pops up.



```
Info: C:¥intelFPGA¥18.1¥quartus¥sopc_builder¥builtin.i
Info: C:/intelFPGA/18.1/quartus/sopc_builder/**/* ma
Info: Reading index C:¥intelFPGA¥18.1¥quartus¥common
Info: C:¥intelFPGA¥18.1¥quartus¥common¥librarian¥fa
Info: C:/intelFPGA/18.1/quartus/common/librarian/fa
Info: C:/intelFPGA/18.1/quartus/sopc_builder/bin/$IP
Info: C:¥intelFPGA¥18.1¥quartus¥sopc_builder¥bin¥roo
Info: C:/intelFPGA/18.1/quartus/sopc_builder/bin/roo
Save System: completed successfully.
```

Figure 3-45. Saving the Platform Designer system

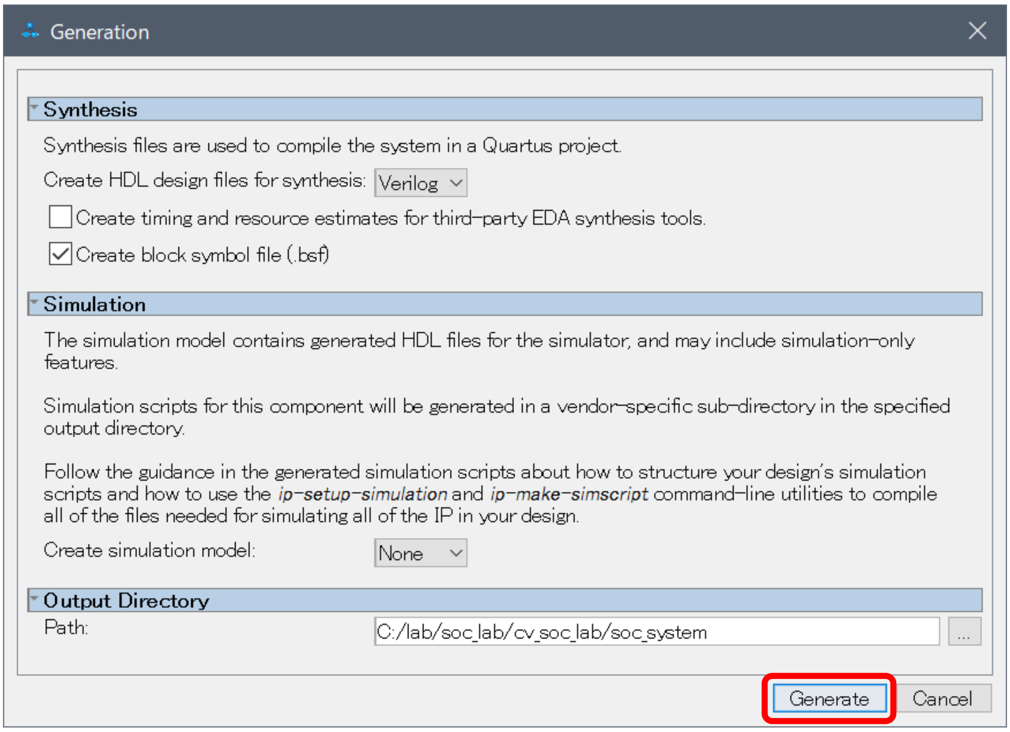
____ 3. Select ***Generate*** menu => ***Generate HDL***.



Platform Designer - soc_system.qsys (C:¥lab¥soc_la
File Edit System Generate View Tools Help
IP Catalog
Project
New Comp
System
Generate HDL...
Generate Testbench System...
Generate Example Design...
Show Instantiation Template...

Figure 3-46. Generating the Platform Designer system

____ 4. Check the settings in the ***Generation*** window and execute [***Generate***].



Synthesis

Synthesis files are used to compile the system in a Quartus project.

Create HDL design files for synthesis: Verilog

□ Create timing and resource estimates for third-party EDA synthesis tools.

☑ Create block symbol file (.bsf)

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: None

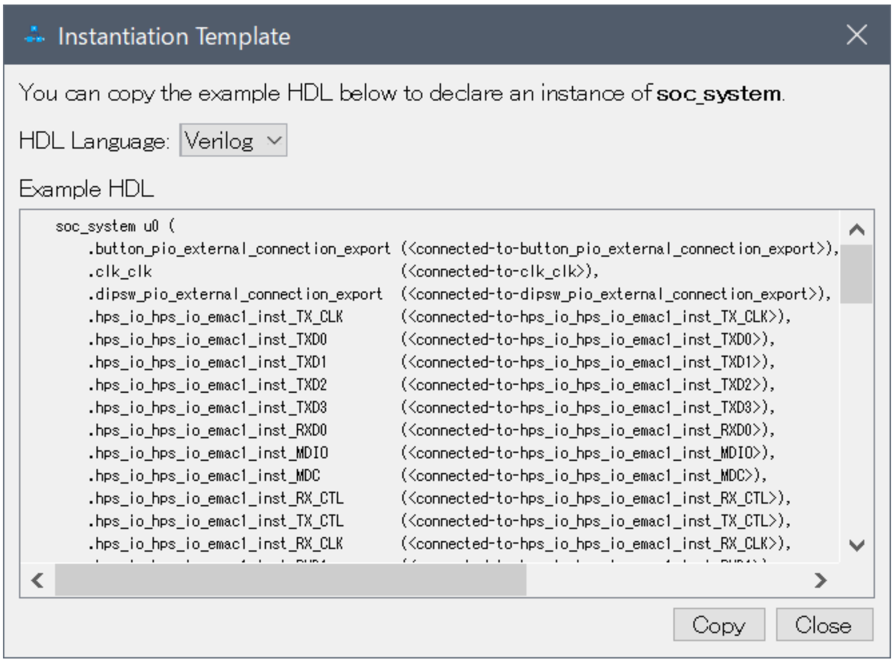
Output Directory

Path: C:/lab/soc_lab/cv_soc_lab/soc_system

Figure 3-47. Platform Designer System Generate Execution Window

____ 5. Platform Designer's ***Generate*** menu => ***Show Instantiation Template*** shows an example of an instance that can be used when instantiating a top design.

This time, it has already been instantiated, so there is no particular work to be done, but it is very useful when actually using it.



You can copy the example HDL below to declare an instance of **soc_system**.

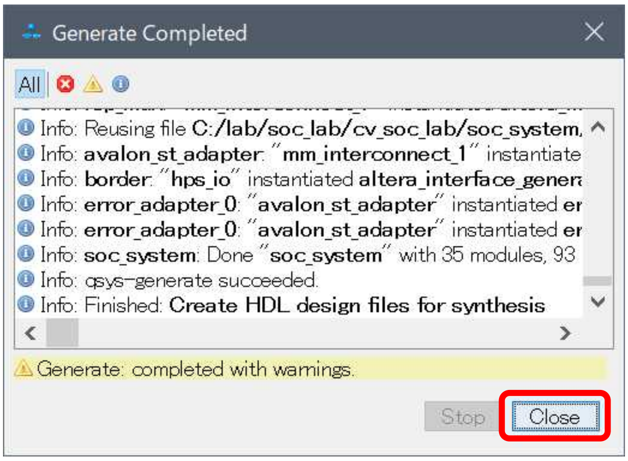
HDL Language: Verilog

Example HDL

```
    soc_system u0 (
        .button_pio_external_connection_export (<connected-to-button_pio_external_connection_export>),
        .clk_clk                               (<connected-to-clk_clk>),
        .dipsw_pio_external_connection_export  (<connected-to-dipsw_pio_external_connection_export>),
        .hps_io_hps_io_emac1_inst_TX_CLK       (<connected-to-hps_io_hps_io_emac1_inst_TX_CLK>),
        .hps_io_hps_io_emac1_inst_TXD0         (<connected-to-hps_io_hps_io_emac1_inst_TXD0>),
        .hps_io_hps_io_emac1_inst_TXD1         (<connected-to-hps_io_hps_io_emac1_inst_TXD1>),
        .hps_io_hps_io_emac1_inst_TXD2         (<connected-to-hps_io_hps_io_emac1_inst_TXD2>),
        .hps_io_hps_io_emac1_inst_TXD3         (<connected-to-hps_io_hps_io_emac1_inst_TXD3>),
        .hps_io_hps_io_emac1_inst_RXD0         (<connected-to-hps_io_hps_io_emac1_inst_RXD0>),
        .hps_io_hps_io_emac1_inst_MDIO         (<connected-to-hps_io_hps_io_emac1_inst_MDIO>),
        .hps_io_hps_io_emac1_inst_MDC          (<connected-to-hps_io_hps_io_emac1_inst_MDC>),
        .hps_io_hps_io_emac1_inst_RX_CTL       (<connected-to-hps_io_hps_io_emac1_inst_RX_CTL>),
        .hps_io_hps_io_emac1_inst_TX_CTL       (<connected-to-hps_io_hps_io_emac1_inst_TX_CTL>),
        .hps_io_hps_io_emac1_inst_RX_CLK       (<connected-to-hps_io_hps_io_emac1_inst_RX_CLK>),
```

Figure 3-48. Platform Designer system instance example

____ 6. After Platform Designer generation is complete, click the [***Close***] button to close the Platform Designer system generation dialog box and return to Quartus® Prime.



```
Info: Reusing file C:/lab/soc_lab/cv_soc_lab/soc_system,
Info: avalon_st_adapter: "mm_interconnect_1" instantiate
Info: border: "hps_io" instantiated altera_interface_genera
Info: error_adapter_0: "avalon_st_adapter" instantiated er
Info: error_adapter_0: "avalon_st_adapter" instantiated er
Info: soc_system: Done "soc_system" with 35 modules, 93
Info: qsys-generate succeeded.
Info: Finished: Create HDL design files for synthesis
```

Generate: completed with warnings.

Figure 3-49. Platform Designer system generation complete

____ 7. Select Quartus® Prime's ***Project*** menu => ***Add/Remove Files in Project*** (The Settings dialog box appears with the Files category selected).

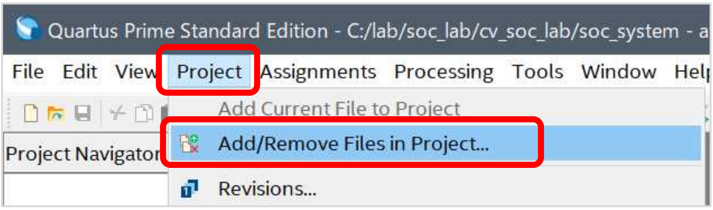
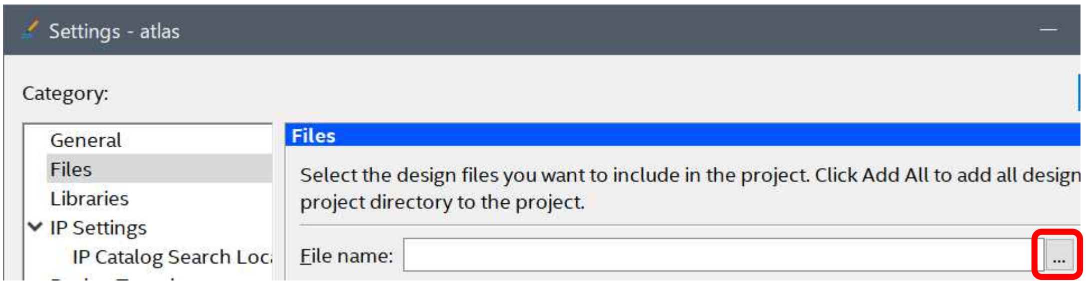

Quartus Prime Standard Edition - C:/lab/soc_lab/cv_soc_lab/soc_system - a
File
Edit
View
Project
Assignments
Processing
Tools
Window
Hel
Project Navigator
Add Current File to Project
Add/Remove Files in Project...
Revisions...

Figure 3-50. Select Add/Remove Files in Project

____ 8. Press the [...] button next to the ***File name*** field in the Settings dialog box and browse to the ***soc_system/synthesis*** folder from the ***Select File*** window.

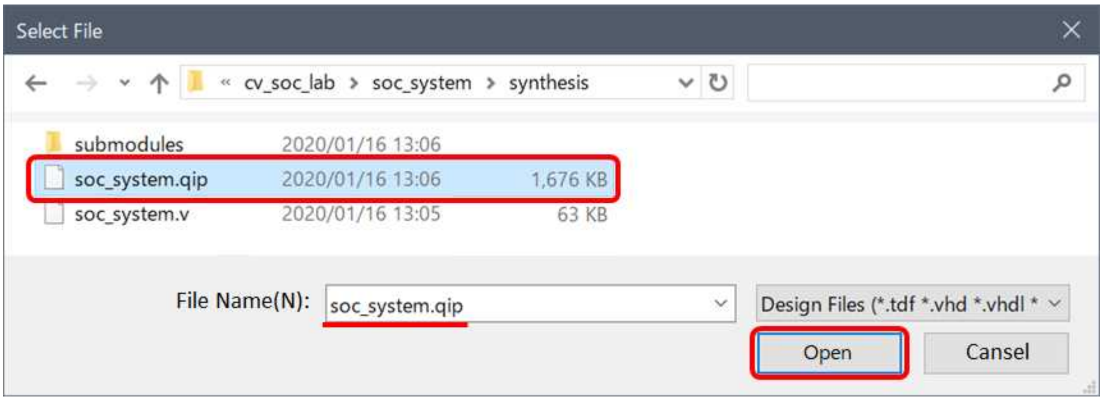


Select the design files you want to include in the project. Click Add All to add all design project directory to the project.

File name:

Figure 3-51. Settings dialog box

____ 9. Select the file ***Soc_system.qip*** and click [***Open***]. This qip file links the generated components in Platform Designer. Instead of registering each generated file individually, you can add a Platform Designer system to a project by registering this qip file.


Select File

cv_soc_lab › soc_system › synthesis

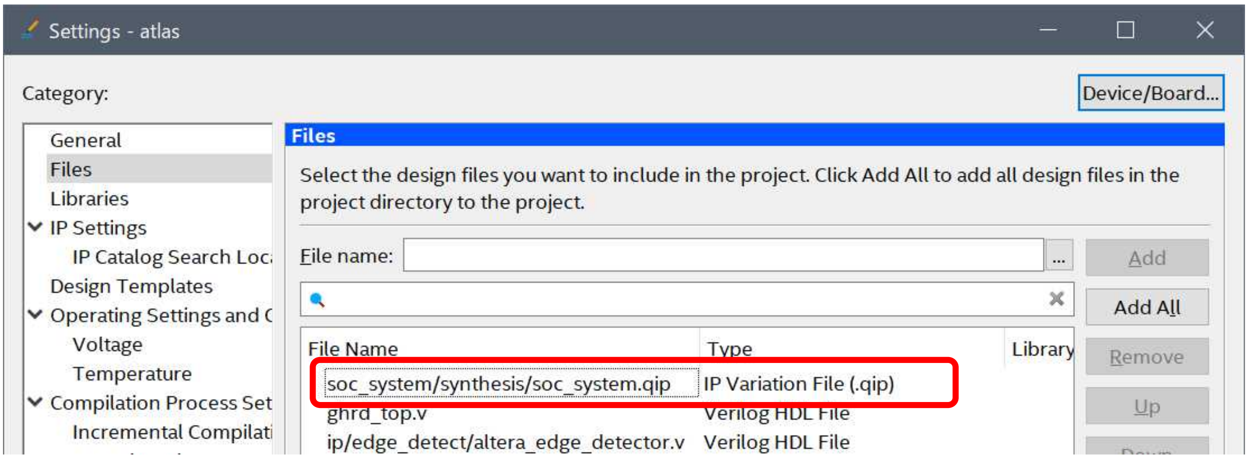
submodules	2020/01/16 13:06	
soc_system.qip	2020/01/16 13:06	1,676 KB
soc_system.v	2020/01/16 13:05	63 KB

File Name(N): soc_system.qip

Design Files (*.tdf *.vhd *.vhdl *

Figure 3-52. Specifying the Qip File

____ 10. Verify that the file has been added.


Settings - atlas

Category:

Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

File name:

File Name	Type	Library
soc_system/synthesis/soc_system.qip	IP Variation File (.qip)	
ghrd_top.v	Verilog HDL File	
ip/edge_detect/altera_edge_detector.v	Verilog HDL File	

Figure 3-53. Register Qip file

____ 11. Close the ***Settings*** dialog box with the [***OK***] button.

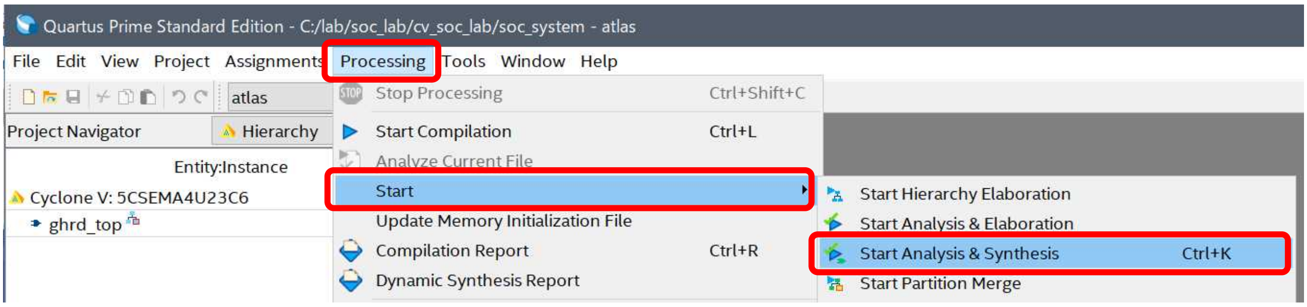
3-11. Step 11: Set pin assignments and compile Quartus® Prime project

For HPS-only IO, pin assignments are basically done automatically by the tool because pin placement is fixed. The exception is that SDRAM interfaces require a tool-generated script to be executed by the designer. To execute the script, first generate the netlist and then execute the script.

Therefore, first run Analysis & Synthesis to create the netlist, then execute the script and compile the FPGA again.

____ 1. In Quartus® Prime, select ***Processing*** menu => ***Start*** => ***Start Analysis & Synthesis***.

(Alternatively, click the Start Analysis & Synthesis button on the GUI.)


Quartus Prime Standard Edition - C:/lab/soc_lab/cv_soc_lab/soc_system - atlas
File Edit View Project Assignments Processing Tools Window Help
atlas
Project Navigator
Hierarchy
Entity:Instance
Cyclone V: 5CSEMA4U23C6
ghrd_top
Stop Processing Ctrl+Shift+C
Start Compilation Ctrl+L
Analyze Current File
Start
Update Memory Initialization File
Compilation Report Ctrl+R
Dynamic Synthesis Report
Start Hierarchy Elaboration
Start Analysis & Elaboration
Start Analysis & Synthesis Ctrl+K
Start Partition Merge

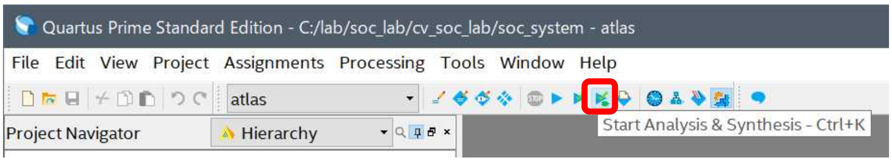

Quartus Prime Standard Edition - C:/lab/soc_lab/cv_soc_lab/soc_system - atlas
File Edit View Project Assignments Processing Tools Window Help
atlas
Project Navigator
Hierarchy
Start Analysis & Synthesis - Ctrl+K

Figure 3-54. Start Analysis & Synthesis

____ 2. When finished, verify that there are no errors. ✓ , there are no errors. The netlist is now created.

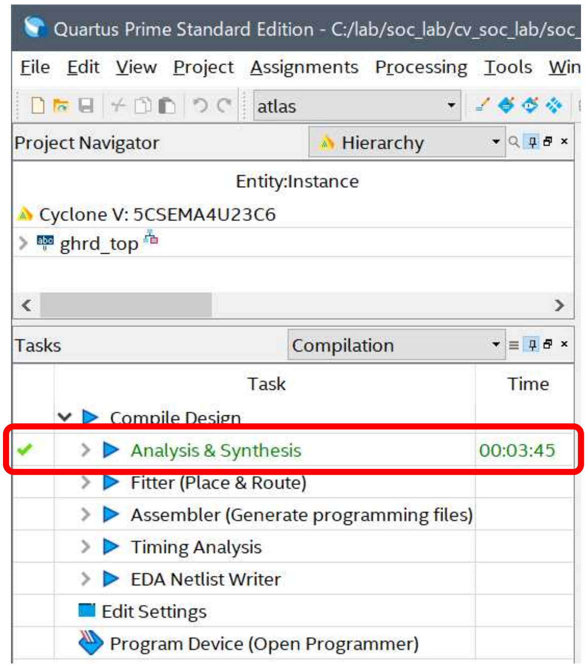


Figure 3-55. Verify successful completion of Start Analysis & Synthesis

____ 3. Select Quartus® Prime ***Tools*** menu => ***TCL scripts***.

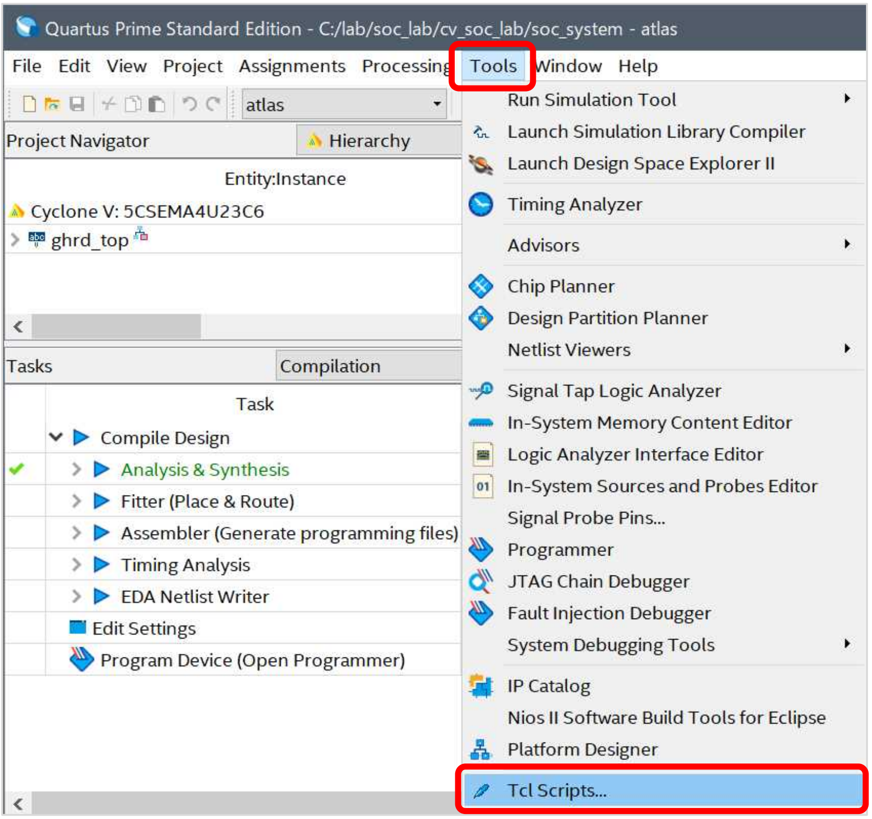
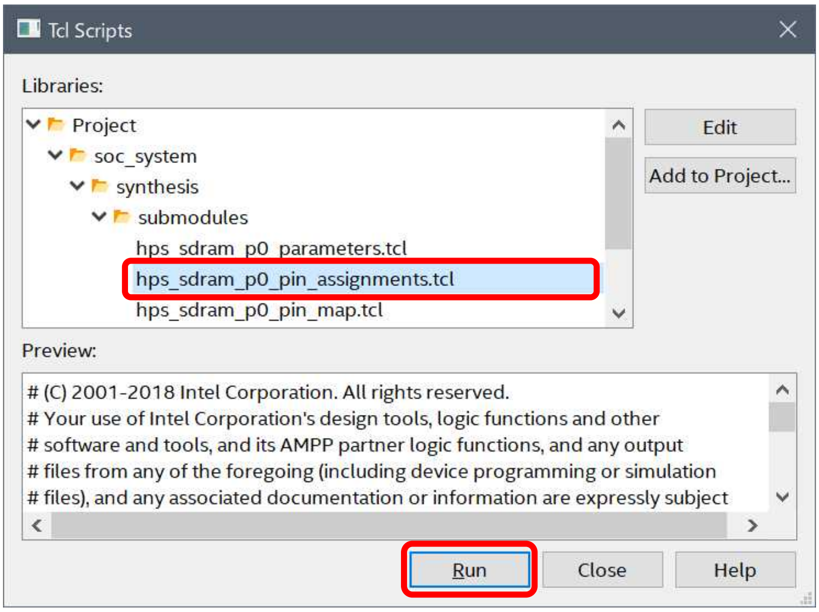

Quartus Prime Standard Edition - C:/lab/soc_lab/cv_soc_lab/soc_system - atlas
File Edit View Project Assignments Processing Tools Window Help
atlas
Project Navigator
Hierarchy
Entity:Instance
Cyclone V: 5CSEMA4U23C6
ghrd_top
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer II
Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
Signal Tap Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
Signal Probe Pins...
Programmer
JTAG Chain Debugger
Fault Injection Debugger
System Debugging Tools
IP Catalog
Nios II Software Build Tools for Eclipse
Platform Designer
Tcl Scripts...

Figure 3-56. Starting the Tcl Scripts Window

____ 4. Select **hps_ sdram _p0_pin_assignments.tcl** in soc_system => synthesis => submodules, and click the [***Run***] button (it takes a while for the application to take effect).

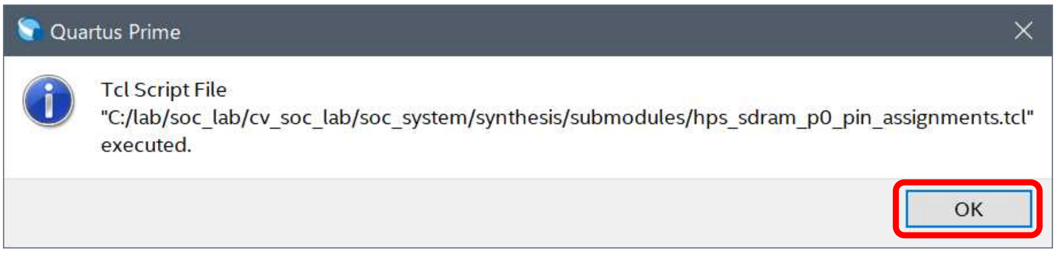
This operation will apply the settings made in the SDRAM Controller tab of HPS, such as SDRAM IO Standard and OCT settings.



```
# (C) 2001-2018 Intel Corporation. All rights reserved.
# Your use of Intel Corporation's design tools, logic functions and other
# software and tools, and its AMPP partner logic functions, and any output
# files from any of the foregoing (including device programming or simulation
# files), and any associated documentation or information are expressly subject
```

Figure 3-57. Executing a Tcl Script

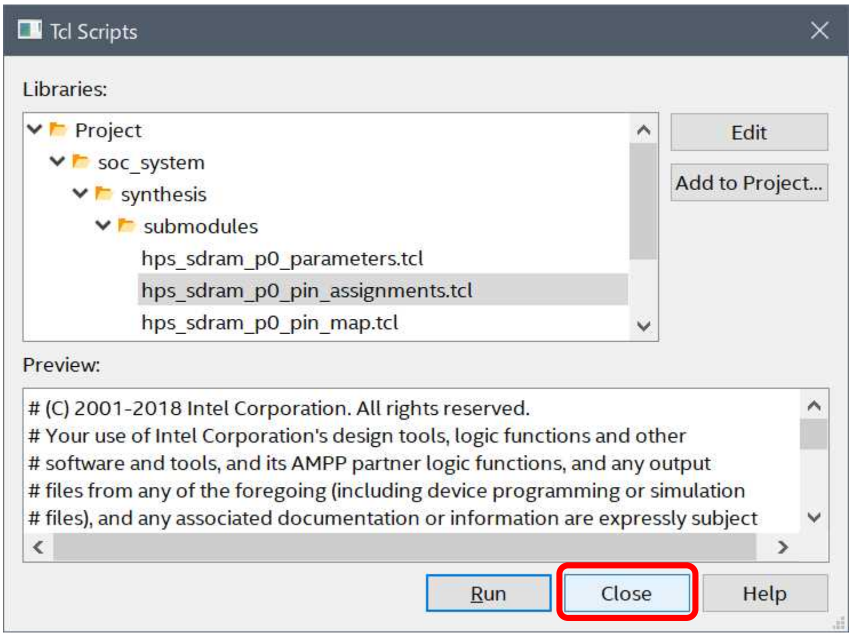
____ 5. When you have finished, click the [***OK***] button.



Tcl Script File "C:/lab/soc_lab/cv_soc_lab/soc_system/synthesis/submodules/hps_sdram_p0_pin_assignments.tcl" executed.

Figure 3-58. Completing a Tcl Script

____ 6. [***Close***] the Tcl Scripts window.



```
# (C) 2001-2018 Intel Corporation. All rights reserved.
# Your use of Intel Corporation's design tools, logic functions and other
# software and tools, and its AMPP partner logic functions, and any output
# files from any of the foregoing (including device programming or simulation
# files), and any associated documentation or information are expressly subject
```

Figure 3-59. Close Tcl Scripts Window

____ 7. Select Quartus® Prime ***Processing*** menu => ***Start Compilation*** (Alternatively, click the Start Compilation button ▶ on the GUI) to compile the FPGA. This compilation creates a .sof file that will be the working image of the HW and a handoff file that will be passed to the next software development.

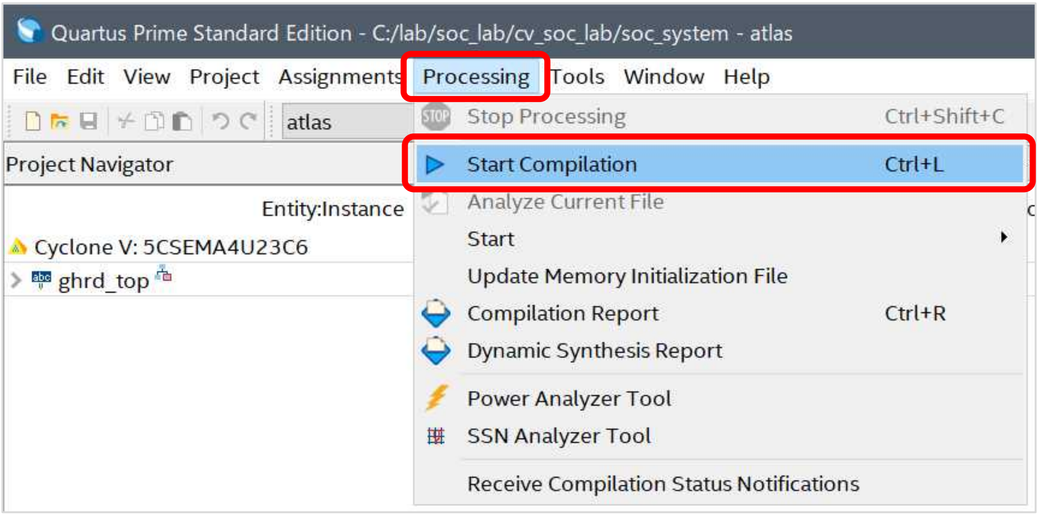
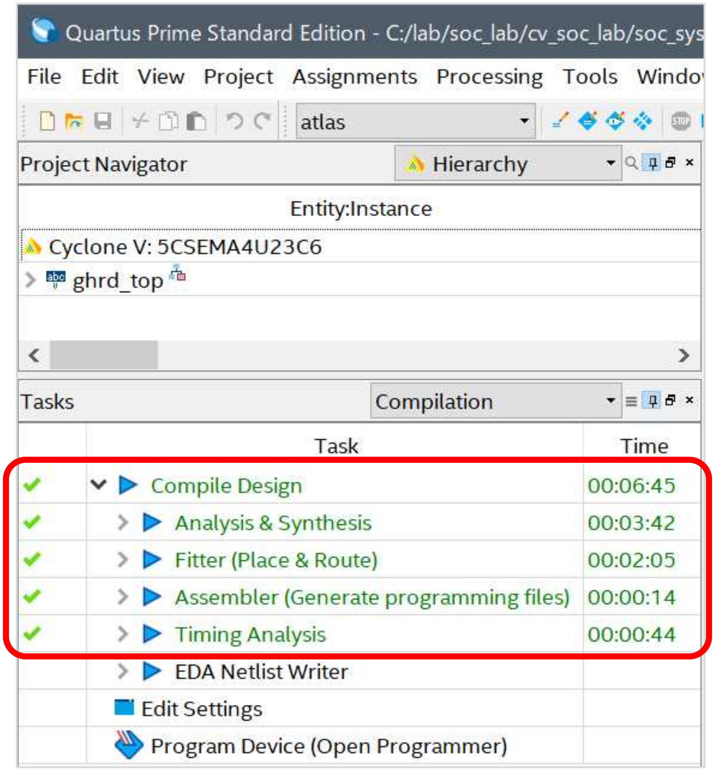


Figure 3-60. Run Start Compilation

____ 8. Verify that the compilation is complete.



	Task	Time
✓	Compile Design	00:06:45
✓	Analysis & Synthesis	00:03:42
✓	Fitter (Place & Route)	00:02:05
✓	Assembler (Generate programming files)	00:00:14
✓	Timing Analysis	00:00:44
	EDA Netlist Writer	
	Edit Settings	
	Program Device (Open Programmer)	

Figure 3-61. Compilation complete

3-12. Step 12: Verify the output file

Verify the output file in Quartus® Prime and Platform Designer.

____ 1. Using Windows® OS Explorer, navigate to the output file folder (below).

```
C:¥lab¥soc_lab¥cv_soc_lab¥output_files
```

____ 2. Verify that there is a .sof file output in the above folder.
atlas.sof for **Atlas-SoC board**
DE10 Nano.sof for **DE10 Nano board**.
As mentioned earlier, this file is the operational image file of the FPGA.
This file will be written to the FPGA on the board using a tool called Programmer in a later exercise.

____ 3. Using Windows® OS Explorer, navigate to the hardware/software handoff directory.

```
C:¥lab¥soc_lab¥cv_soc_lab¥hps_isw_handoff¥soc_system_hps_0
```

Under the above folder are the hardware/software handoff files generated by the tool. These files contain various data set in the HPS component screen of Platform Designer, information about the SDRAM interface of HPS, and other files. These files are used to generate a file called Preloader, which is used to initialize the HPS side.
These files will be used to create Preloader in later exercises.

Lab 1 Hardware Exercise Summary

In this section, you configured the hardware including the Arm® processor by performing the following tasks:

- Adding HPS components to the Platform Designer system
- Configuring HPS components
- Connecting HPS components to other components
- Generating the Platform Designer system
- Checking the files output by Quartus® Prime/Platform Designer

Lab 1 is now complete.

4. Lab 2 - Software Exercise (1) Generate Preloader

In this section, you will generate Preloader using the handoff file created in "3. Lab 1: Hardware Exercise"

Ⓟ **Point:**

In SoC EDS v20.1 (SoC EDS v19.1 and later), you need to build Preloader under the Linux OS environment. In this exercise, you will use the Windows Subsystem for Linux (WSL1) included with Windows® 10 to generate the boot loader.

Preloader is a boot loader based on the U-boot second program loader (Later, u-boot spl) and customized for Altera ® SoC FPGAs. Preloader is responsible for:

- HPS pin multiplex configuration
- HPS IOCSR configuration
- HPS PLL and clock settings
- HPS peripheral reset unreset
- SDRAM initialization (calibration, etc.)
- Deployment of the next stage boot image to SDRAM

As mentioned above, Preloader provides the function of initializing the HPS block and loading the U-boot and OS into SDRAM. Preloader is automatically generated by using a handoff file that is automatically generated when Quartus® Prime/Platform Designer is designed. Therefore, the settings made in Quartus® Prime/Platform Designer can be applied to the HPS block without the user having to build initialization software. The sof file I just checked is the operating image of the FPGA side. On the other hand, the operating image of the HPS side is this file called Preloader. Note that the operating image is executed using different files for the FPGA side and the HPS side.

Now let's create the Preloader.

4-1. Step 1: Launch Embedded Command Shell

____ 1. Launch the Embedded Command Shell included with the SoC EDS.

Launch the Embedded Command Shell by double-clicking the startup script Embedded_Command_Shell.bat located in the Windows® Start menu or in the SoC EDS installation folder.

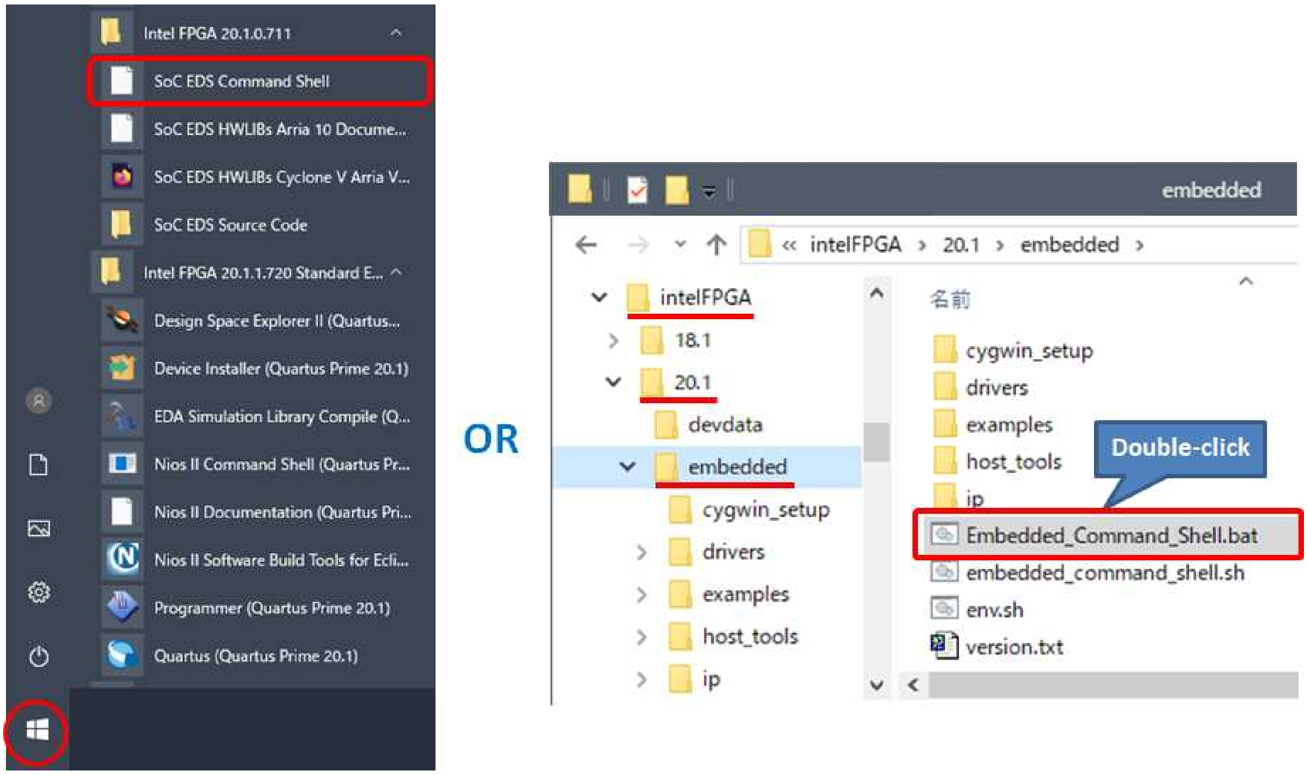

Intel FPGA 20.1.0.711
SoC EDS Command Shell
SoC EDS HWLIBs Arria 10 Docume...
SoC EDS HWLIBs Cyclone V Arria V...
SoC EDS Source Code
Intel FPGA 20.1.1.720 Standard E...
Design Space Explorer II (Quartus...
Device Installer (Quartus Prime 20.1)
EDA Simulation Library Compile (Q...
Nios II Command Shell (Quartus Pr...
Nios II Documentation (Quartus Pri...
Nios II Software Build Tools for Ecli...
Programmer (Quartus Prime 20.1)
Quartus (Quartus Prime 20.1)
OR
embedded
« intelFPGA › 20.1 › embedded ›
intelFPGA
18.1
20.1
devdata
embedded
cygwin_setup
drivers
examples
host_tools
ip
名前
cygwin_setup
drivers
examples
host_tools
ip
Embedded_Command_Shell.bat
embedded_command_shell.sh
env.sh
version.txt
Double-click

Figure 4-1. Starting the Embedded Command Shell

4-2. Step 2: Generate the bsp project

Generate the bsp project required for Preloader generation from the Quartus® Prime design handoff information (hps_isw_handoff¥soc_system_hps_0).

____ 1. Change the current directory of the Embedded Command Shell to the Quartus® Prime project directory.

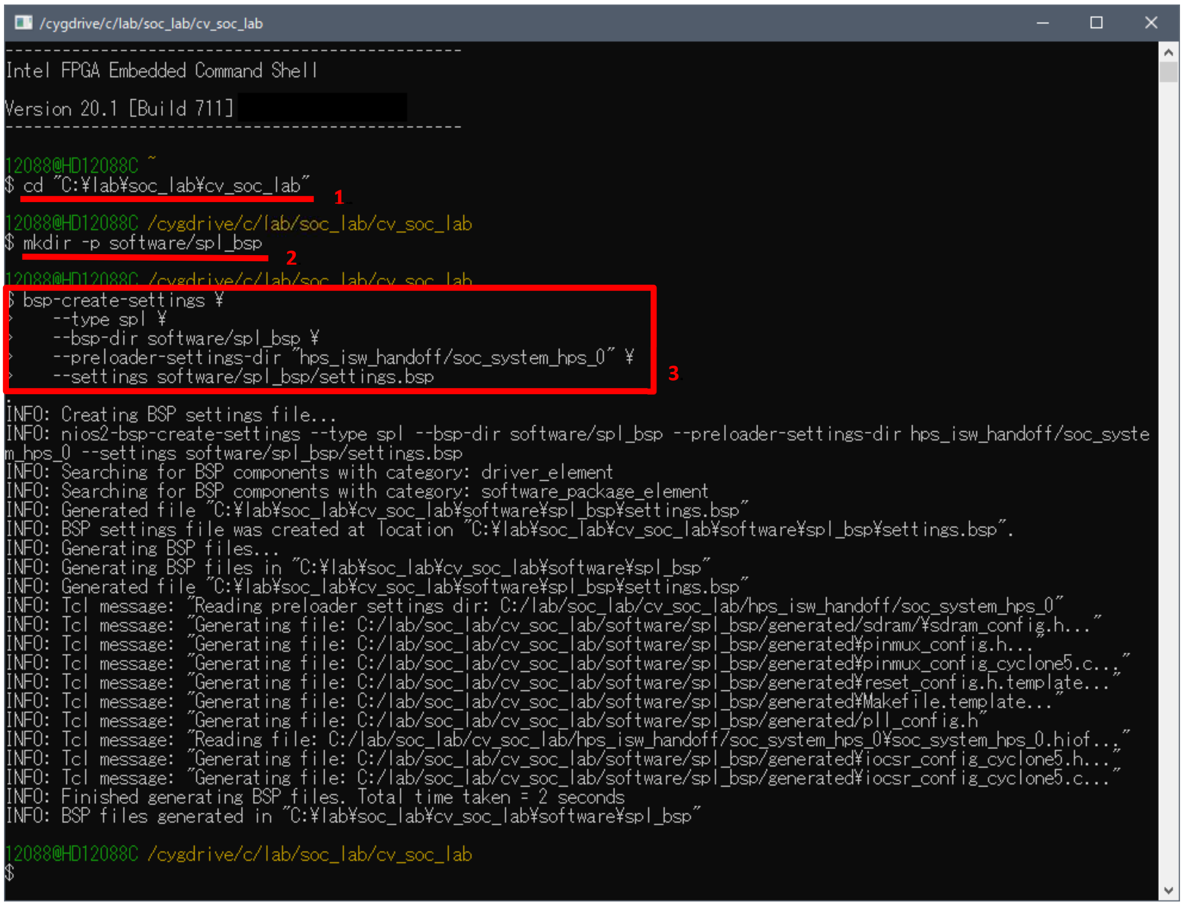
```
$ cd “C:¥lab¥soc_lab¥cv_soc_lab” ↵
```

____ 2. Then run the following command to create the directory where the bsp project will be output:

```
$ mkdir -p software/spl_bsp ↵
```

____ 3. The following command generates the bsp project: The generated directory of the bsp project contains the source code that contains the definitions with the handoff information.

```
$ bsp-create-settings ¥
   --type spl ¥
   --bsp-dir software/spl_bsp ¥
   --preloader-settings-dir “hps_isw_handoff/soc_system_hps_0” ¥
   --settings software/spl_bsp/settings.bsp ↵
```



```
-----------------------------------------------
Intel FPGA Embedded Command Shell

Version 20.1 [Build 711] [redacted]
-----------------------------------------------

12088@HD12088C ~
$ cd "C:¥lab¥soc_lab¥cv_soc_lab"

12088@HD12088C /cygdrive/c/lab/soc_lab/cv_soc_lab
$ mkdir -p software/spl_bsp

12088@HD12088C /cygdrive/c/lab/soc_lab/cv_soc_lab
$ bsp-create-settings ¥
>    --type spl ¥
>    --bsp-dir software/spl_bsp ¥
>    --preloader-settings-dir "hps_isw_handoff/soc_system_hps_0" ¥
>    --settings software/spl_bsp/settings.bsp

INFO: Creating BSP settings file...
INFO: nios2-bsp-create-settings --type spl --bsp-dir software/spl_bsp --preloader-settings-dir hps_isw_handoff/soc_syste
m_hps_0 --settings software/spl_bsp/settings.bsp
INFO: Searching for BSP components with category: driver_element
INFO: Searching for BSP components with category: software_package_element
INFO: Generated file "C:¥lab¥soc_lab¥cv_soc_lab¥software¥spl_bsp¥settings.bsp"
INFO: BSP settings file was created at location "C:¥lab¥soc_lab¥cv_soc_lab¥software¥spl_bsp¥settings.bsp".
INFO: Generating BSP files...
INFO: Generating BSP files in "C:¥lab¥soc_lab¥cv_soc_lab¥software¥spl_bsp"
INFO: Generated file "C:¥lab¥soc_lab¥cv_soc_lab¥software¥spl_bsp¥settings.bsp"
INFO: Tcl message: "Reading preloader settings dir: C:/lab/soc_lab/cv_soc_lab/hps_isw_handoff/soc_system_hps_0"
INFO: Tcl message: "Generating file: C:/lab/soc_lab/cv_soc_lab/software/spl_bsp/generated/sdram/¥sdram_config.h..."
INFO: Tcl message: "Generating file: C:/lab/soc_lab/cv_soc_lab/software/spl_bsp/generated¥pinmux_config.h..."
INFO: Tcl message: "Generating file: C:/lab/soc_lab/cv_soc_lab/software/spl_bsp/generated¥pinmux_config_cyclone5.c..."
INFO: Tcl message: "Generating file: C:/lab/soc_lab/cv_soc_lab/software/spl_bsp/generated¥reset_config.h.template..."
INFO: Tcl message: "Generating file: C:/lab/soc_lab/cv_soc_lab/software/spl_bsp/generated¥Makefile.template..."
INFO: Tcl message: "Generating file: C:/lab/soc_lab/cv_soc_lab/software/spl_bsp/generated/pll_config.h"
INFO: Tcl message: "Reading file: C:/lab/soc_lab/cv_soc_lab/hps_isw_handoff/soc_system_hps_0¥soc_system_hps_0.hiof..."
INFO: Tcl message: "Generating file: C:/lab/soc_lab/cv_soc_lab/software/spl_bsp/generated¥iocsr_config_cyclone5.h..."
INFO: Tcl message: "Generating file: C:/lab/soc_lab/cv_soc_lab/software/spl_bsp/generated¥iocsr_config_cyclone5.c..."
INFO: Finished generating BSP files. Total time taken = 2 seconds
INFO: BSP files generated in "C:¥lab¥soc_lab¥cv_soc_lab¥software¥spl_bsp"

12088@HD12088C /cygdrive/c/lab/soc_lab/cv_soc_lab
$
```

Figure 4-2. Generating a Bsp project

⚠ Note:

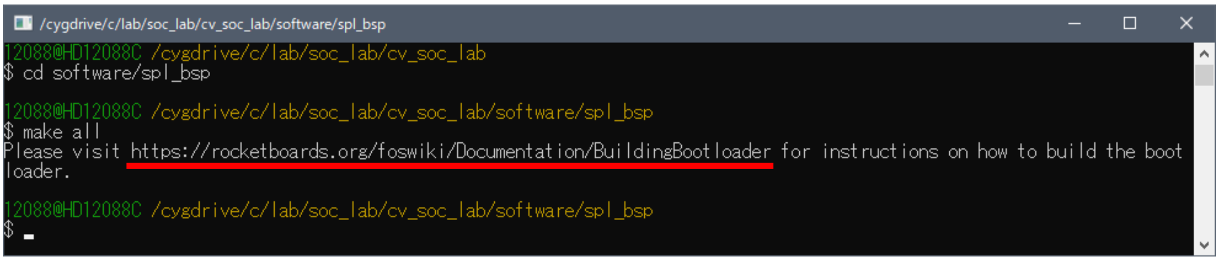
Error protection is required before using bsp-create-settings in SoC EDS v20.1std on Windows® 10. If an error occurs, take the corrective action described in the following reference sites and run bsp-create-settings again.

Reference:

Macnica Altera FPGA Insights "Workaround for bsp-create-settings execution error in SoC EDS environment"

ⓘ **Info:**

The bsp project contains a Makefile but is not used in the SoC EDS v20.1 (SoC EDS v19.1 or later) environment. Running make under the bsp project will take you to the documentation page (URL) on the Rocketboards.org site with instructions for building the bootloader.



/cygdrive/c/lab/soc_lab/cv_soc_lab/software/spl_bsp

```
12088@HD12088C /cygdrive/c/lab/soc_lab/cv_soc_lab
$ cd software/spl_bsp

12088@HD12088C /cygdrive/c/lab/soc_lab/cv_soc_lab/software/spl_bsp
$ make all
Please visit https://rocketboards.org/foswiki/Documentation/BuildingBootloader for instructions on how to build the boot
loader.

12088@HD12088C /cygdrive/c/lab/soc_lab/cv_soc_lab/software/spl_bsp
$ _
```

See: Building Bootloader for Cyclone V and Arria 10

bsp-create-settings created a bsp project under the software/spl_bsp directory. The source files contained in the bsp project will then be used to build Preloader.

4-3. Step 3: Launch Preloader Build Environment

Generate the bsp project needed to generate Preloader from the Quartus® Prime design handoff information (hps_isw_handoff¥soc_system_hps_0).

As noted at the beginning of Lab 2, building Preloader (boot loader) requires a Linux OS environment. This exercise uses Ubuntu 18.04 LTS running on WSL1. The following steps will be performed on the Ubuntu 18.04 LTS terminal of WSL, so please set up your WSL/Ubuntu 18.04 LTS environment beforehand.

⚠ **Note:**

* In addition to WSL 1, which was originally released, there are two different WSL environments: WSL 2 (Windows Subsystem for Linux 2), which is supported in Windows® 10 version 2004 and later. Please note that this exercise uses WSL 1.

* This document uses Ubuntu 18.04 LTS, but you should read the version of Ubuntu as appropriate for your environment.

ⓘ **Info:**

To set up the WSL 1 environment, refer to the following article.

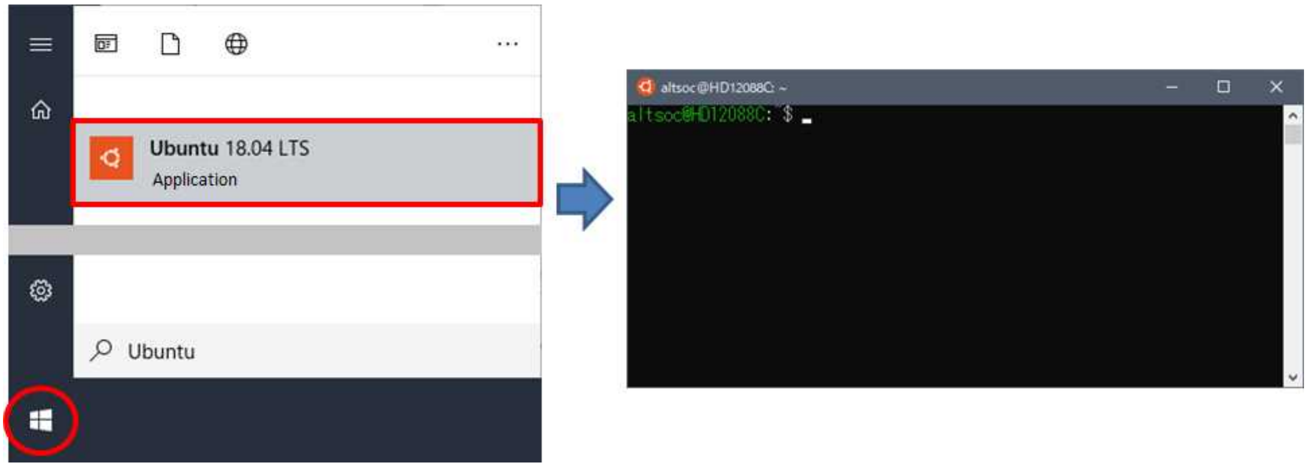
Reference:

Macnica Altera FPGA Insights "Building Preloader/U-Boot with WSL [Part 1] Environment Setup"

*** At the "SoC Startup Trial Seminar" held in our company, it will be set up beforehand.**

_____ 1. Start Ubuntu 18.04 LTS running on WSL1.

Type Ubuntu in the Windows® Start menu and click on the suggested Ubuntu 18.04 LTS app.



```
altsoc@HD12088C: $ _
```

Figure 4-3. Starting the Embedded Command Shell

ⓘ **Info:**

In your Linux OS environment, you need to install various packages necessary for building. In the case of Ubuntu 18.04 LTS, use the following command to add the necessary packages beforehand. "Do you want to continue? [Y/n] "appears, enter Y.

```
$ sudo apt update ↵
$ sudo apt upgrade ↵
$ sudo apt install build-essential bison flex ncurses-dev ↵
```

*** At the "SoC Startup Trial Seminar" held in our company, it will be set up beforehand.**

____ 2. Install the bare metal GCC toolchain on Ubuntu 18.04 LTS on WSL1.

Run the following command in Ubuntu 18.04 LTS terminal to download and extract the GCC toolchain. If already installed, proceed to the next step.

```
$ mkdir ~/toolchains ↵
$ cd ~/toolchains ↵
$ wget https://releases.linaro.org/components/toolchain/binaries/latest-7/arm-eabi/gcc-linaro-7.5.0-2019.12-x86_64_arm-eabi.tar.xz ↵
$ tar xf gcc-linaro-7.5.0-2019.12-x86_64_arm-eabi.tar.xz ↵
```

ⓘ **Info:**

Downloading the GCC toolchain takes time due to the large file size.

*** At the "SoC Startup Trial Seminar" held in our company, it will be set up beforehand.**

____ 3. Set environment variables to use the bare metal GCC toolchain.

Run the following command in the Ubuntu 18.04 LTS terminal:

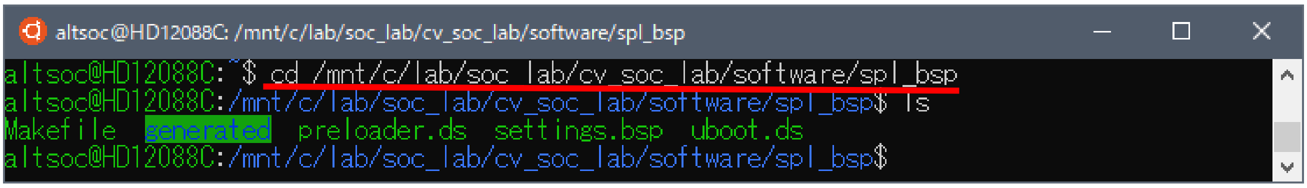
```
$ export PATH=~/toolchains/gcc-linaro-7.5.0-2019.12-x86_64_arm-eabi/bin:$PATH ↵
$ export ARCH=arm ↵
$ export CROSS_COMPILE=arm-eabi- ↵
```

4-4. Step 4: Build Preloader

Generate the bsp project needed to generate Preloader from the Quartus® Prime design handoff information (hps_isw_handoff¥soc_system_hps_0).

____ 1. Change the current directory in the Ubuntu 18.04 LTS terminal to the bsp project directory.

```
$ cd /mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp ↵
```



altsoc@HD12088C: /mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp

```
altsoc@HD12088C:~$ cd /mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp
altsoc@HD12088C:/mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp$ ls
Makefile  generated  preloader.ds  settings.bsp  uboot.ds
altsoc@HD12088C:/mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp$
```

Figure 4-4. Navigate to the bsp project directory

____ 2. Download and unpack the U-Boot/Preloader source tree under the bsp project directory. Run the following command in your Ubuntu 18.04 LTS terminal:

```
$ wget https://github.com/altera-opensource/u-boot-
socfpga/archive/ACDS20.1STD_REL_GSRD_PR.tar.gz ↵
$ tar -xzvf ACDS20.1STD_REL_GSRD_PR.tar.gz ↵
$ mv u-boot-socfpga-ACDS20.1STD_REL_GSRD_PR/ uboot-socfpga ↵
```

ⓘ **Info:**

If wget is not available, such as when working offline, use the downloaded source tree. Instead, use the following command to copy and unpack the U-Boot/Preloader source tree:

```
$ tar -xzvf /mnt/c/lab/soc_lab/cv_soc_lab/solution/uboot-
socfpga/ACDS20.1STD_REL_GSRD_PR.tar.gz ↵
$ mv u-boot-socfpga-ACDS20.1STD_REL_GSRD_PR/ uboot-socfpga ↵
```

***For the "SoC Startup Trial Seminar" in our company, follow these steps.**

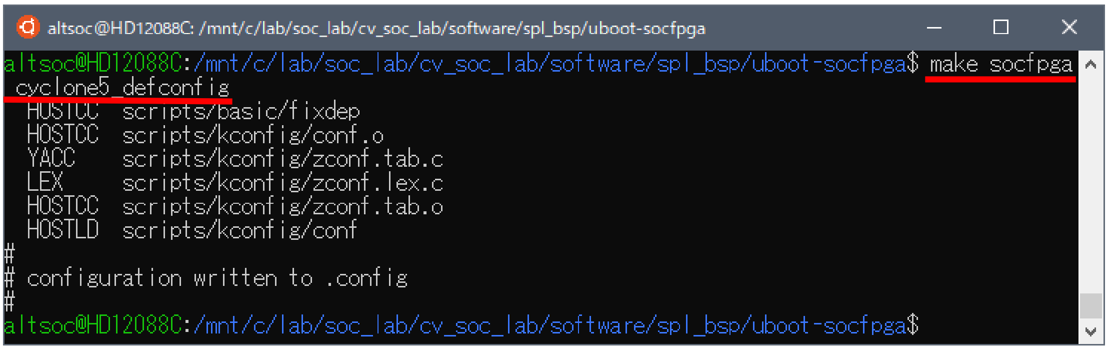
____ 3. Navigate directly to the Uboot-socfpga source tree and run qts-filter.sh.

```
$ cd uboot-socfpga ↵
$ ./arch/arm/mach-socfpga/qts-filter.sh
cyclone5 ../../../ ../ ./board/altera/cyclone5-socdk/qts/ ↵
```

This action populates the various header files generated by bsp-create-settings below the uboot-socfpga source tree.

____ 4. Configure the U-Boot/Preloader source tree for the Cyclone V SoC target.

```
$ make socfpga_cyclone5_defconfig ↵
```



altsoc@HD12088C: /mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp/uboot-socfpga

```
altsoc@HD12088C:/mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp/uboot-socfpga$ make socfpga
_cyclone5_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  YACC    scripts/kconfig/zconf.tab.c
  LEX     scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
altsoc@HD12088C:/mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp/uboot-socfpga$
```

Figure 4-5. Configuring the U-Boot/Preloader Source Tree

____ 5. Rewrite part of the U-Boot/Preloader source tree for debugging.

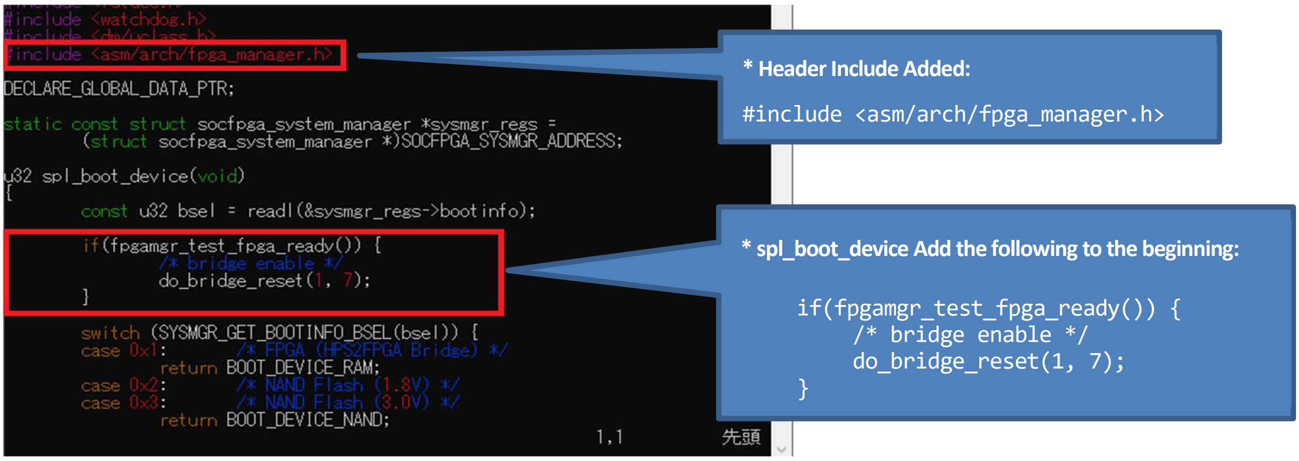
Execute the following command in the Ubuntu 18.04 LTS terminal to edit in the vi editor.

After starting the editor, type the i key to enter the input mode. Rewrite the source according to the figure below.

After editing, press the **ESC** key once, then type **: wq** to save and exit.

> ⓘ **Info:**
>
> In the vi editor, you can search in the file by entering the / (slash) key followed by a search keyword while in normal mode. If you are in input mode (with – insert – or – INSERT – displayed at the bottom left of the window), press the **ESC** key once to enter normal mode.
>
> If you cannot find the edit point in the source, use **/spl_boot_device** or **/socfpga_sdram_apply_static_cfg** to search in the file, referring to the information in the bubble below.

```
$ vi arch/arm/mach-socfpga/spl_gen5.c ↵
```



```
#include <watchdog.h>
#include <dm/uclass.h>
#include <asm/arch/fpga_manager.h>

DECLARE_GLOBAL_DATA_PTR;

static const struct socfpga_system_manager *sysmgr_regs =
        (struct socfpga_system_manager *)SOCFPGA_SYSMGR_ADDRESS;

u32 spl_boot_device(void)
{
        const u32 bsel = readl(&sysmgr_regs->bootinfo);

        if(fpgamgr_test_fpga_ready()) {
                /* bridge enable */
                do_bridge_reset(1, 7);
        }

        switch (SYSMGR_GET_BOOTINFO_BSEL(bsel)) {
        case 0x1:       /* FPGA (HPS2FPGA Bridge) */
                return BOOT_DEVICE_RAM;
        case 0x2:       /* NAND Flash (1.8V) */
        case 0x3:       /* NAND Flash (3.0V) */
                return BOOT_DEVICE_NAND;
                                                      1,1           先頭
```

*** Header Include Added:**

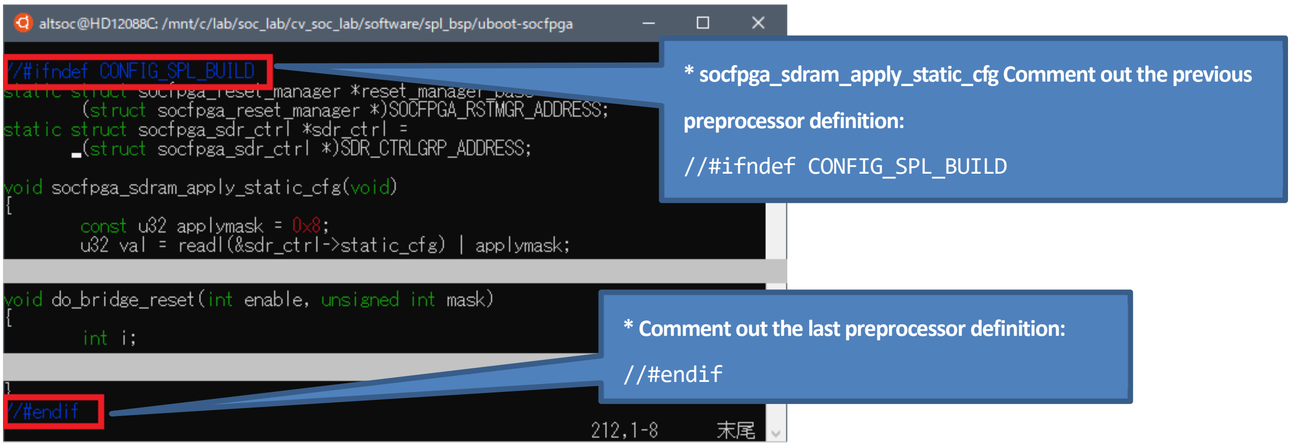
```
#include <asm/arch/fpga_manager.h>
```

*** spl_boot_device Add the following to the beginning:**

```
    if(fpgamgr_test_fpga_ready()) {
        /* bridge enable */
        do_bridge_reset(1, 7);
    }
```

Figure 4-6. Edit Arch/arm/mach-socfpga/spl_gen5.c

```
$ vi arch/arm/mach-socfpga/misc_gen5.c ↵
```



altsoc@HD12088C: /mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp/uboot-socfpga

```
//#ifndef CONFIG_SPL_BUILD
static struct socfpga_reset_manager *reset_manager_base
        (struct socfpga_reset_manager *)SOCFPGA_RSTMGR_ADDRESS;
static struct socfpga_sdr_ctrl *sdr_ctrl =
        (struct socfpga_sdr_ctrl *)SDR_CTRLGRP_ADDRESS;

void socfpga_sdram_apply_static_cfg(void)
{
        const u32 applymask = 0x8;
        u32 val = readl(&sdr_ctrl->static_cfg) | applymask;

void do_bridge_reset(int enable, unsigned int mask)
{
        int i;

}
//#endif
                                                  212,1-8       末尾
```

* socfpga_sdram_apply_static_cfg Comment out the previous preprocessor definition:

`//#ifndef CONFIG_SPL_BUILD`

* Comment out the last preprocessor definition:

`//#endif`

Figure 4-7. Edit Arch/arm/mach-socfpga/misc_gen5.c

ⓘ **Info:**

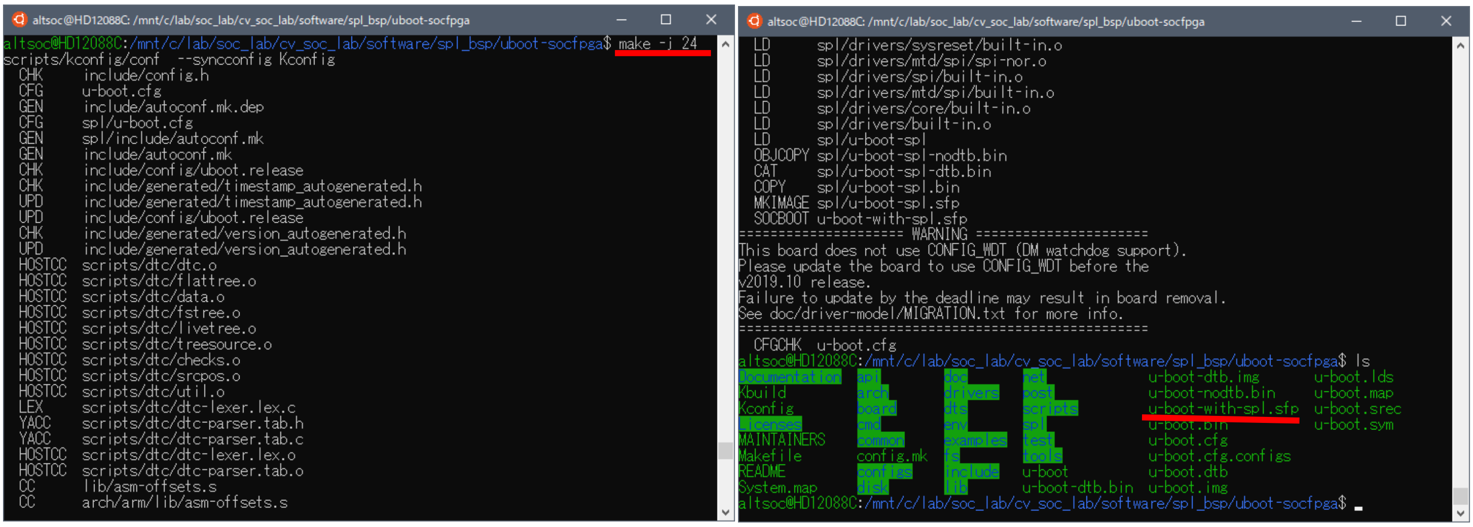
This source code edit adds processing to open the bridge interface between the HPS-FPGA during Preloader. It adds processing equivalent to what is normally done with the bridge enable command when running U-Boot.

Reference:

Macnica Altera FPGA Insights "U-Boot HPS-FPGA Bridge Open Command for SoC FPGAs"

_____ 6. Build the source tree and generate the U-Boot and Preloader.

```
$ make -j 24 ↵
```



altsoc@HD12088C: /mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp/uboot-socfpga
altsoc@HD12088C: /mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp/uboot-socfpga

```
altsoc@HD12088C:/mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp/uboot-socfpga$ make -j 24
scripts/kconfig/conf  --syncconfig Kconfig
  CHK     include/config.h
  CFG     u-boot.cfg
  GEN     include/autoconf.mk.dep
  CFG     spl/u-boot.cfg
  GEN     spl/include/autoconf.mk
  GEN     include/autoconf.mk
  CHK     include/config/uboot.release
  CHK     include/generated/timestamp_autogenerated.h
  UPD     include/generated/timestamp_autogenerated.h
  UPD     include/config/uboot.release
  CHK     include/generated/version_autogenerated.h
  UPD     include/generated/version_autogenerated.h
  HOSTCC  scripts/dtc/dtc.o
  HOSTCC  scripts/dtc/flattree.o
  HOSTCC  scripts/dtc/data.o
  HOSTCC  scripts/dtc/fstree.o
  HOSTCC  scripts/dtc/livetree.o
  HOSTCC  scripts/dtc/treesource.o
  HOSTCC  scripts/dtc/checks.o
  HOSTCC  scripts/dtc/srcpos.o
  HOSTCC  scripts/dtc/util.o
  LEX     scripts/dtc/dtc-lexer.lex.c
  YACC    scripts/dtc/dtc-parser.tab.h
  YACC    scripts/dtc/dtc-parser.tab.c
  HOSTCC  scripts/dtc/dtc-lexer.lex.o
  HOSTCC  scripts/dtc/dtc-parser.tab.o
  CC      lib/asm-offsets.s
  CC      arch/arm/lib/asm-offsets.s
```

```
  LD      spl/drivers/sysreset/built-in.o
  LD      spl/drivers/mtd/spi/spi-nor.o
  LD      spl/drivers/spi/built-in.o
  LD      spl/drivers/mtd/spi/built-in.o
  LD      spl/drivers/core/built-in.o
  LD      spl/drivers/built-in.o
  LD      spl/u-boot-spl
  OBJCOPY spl/u-boot-spl-nodtb.bin
  CAT     spl/u-boot-spl-dtb.bin
  COPY    spl/u-boot-spl.bin
  MKIMAGE spl/u-boot-spl.sfp
  SOCBOOT u-boot-with-spl.sfp
===================== WARNING ======================
This board does not use CONFIG_WDT (DM watchdog support).
Please update the board to use CONFIG_WDT before the
v2019.10 release.
Failure to update by the deadline may result in board removal.
See doc/driver-model/MIGRATION.txt for more info.
====================================================
  CFGCHK  u-boot.cfg
altsoc@HD12088C:/mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp/uboot-socfpga$ ls
Documentation  api        doc       net              u-boot-dtb.img       u-boot.lds
Kbuild         arch       drivers   post             u-boot-nodtb.bin     u-boot.map
Kconfig        board      dts       scripts          u-boot-with-spl.sfp  u-boot.srec
Licenses       cmd        env       spl              u-boot.bin           u-boot.sym
MAINTAINERS    common     examples  test             u-boot.cfg
Makefile       config.mk  fs        tools            u-boot.cfg.configs
README         configs    include   u-boot           u-boot.dtb
System.map     disk       lib       u-boot-dtb.bin   u-boot.img
altsoc@HD12088C:/mnt/c/lab/soc_lab/cv_soc_lab/software/spl_bsp/uboot-socfpga$ _
```

Figure 4-8. Generating the U-boot and Preloader

____ 7. After executing the command, confirm that it finished without any errors.

After confirming that it finished without any errors, confirm that u-boot-with-spl.sfp has been generated by the `ls` ↵ command.

This file is a binary file that contains both U-Boot and Preloader, and is written to an SD card or QSPI flash memory. The mkpimage header referenced by BootROM is attached to the Preloader portion, and the mkimage header information referenced by Preloader is attached to U-Boot.

Lab 2 is now complete.

5. Lab 3: Software Exercise (2) Bare Metal Application

In this section, you will use Arm® DS to run the Hello World sample application included with the SoC EDS and the LED Blink sample application provided for this exercise to demonstrate software development and debugging techniques.

The sample application is outlined below.

- Hello World Sample Application Overview

  This sample application uses the semi-hosting capabilities of Arm® DS to print a "Hello Tim" message to the debugger console.

  This method does not use device peripherals and all communication is done through JTAG.

  The application to run is downloaded to 64KB of on-chip RAM and starts running. This eliminates the need to configure SDRAM memory on the board.

  For these reasons, it can run on any board with Altera® SoC FPGAs.

- Overview of the LED Blink Sample Application

  This sample application uses the FPGA design created in "3. Lab 1: Hardware Exercise" and accesses the PIO peripherals on the FPGA fabric side from the Arm® processor to control the LEDs on and off.

  Before running the main application, this sample application runs the HPS block initialization software called **Preloader** to calibrate the SDRAM, set the clock, and initialize the bridge between the HPS-FPGA. This allows the main application to run with the peripherals on the FPGA fabric side accessible. The main application also loads into SDRAM and starts running.

⚠ **Note:**

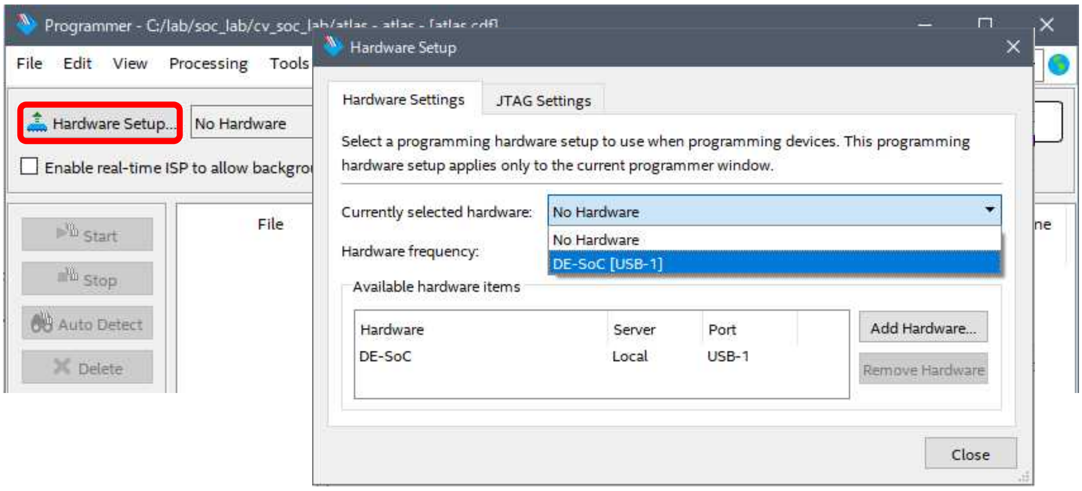
- This document uses Arm® DS version 2020.1. Use the appropriate version of Arm® DS for your environment.
- Before performing this exercise, ensure that Linux® (or another OS) is not running on the board. The OS can interfere with the ability to download and debug bare metal applications (If a microSD card is inserted, remove it).
- The descriptions, screen snapshots, and commands in this section were created using the Windows® version of SoC EDS, but they can be run in a similar manner on a Linux host PC.
- The paths in this section assume you have used the default installation path. If a non-standard location is used, adjust accordingly.
- A license is required to debug bare metal applications on an Arm® DS. The license is tied to the MAC Address. **Make sure the PC knows the network interface associated with the license.**

5-1. Download the FPGA Design

Before starting the software exercises, download the hardware design "3. Lab 1: Hardware Exercise" to the FPGA. Refer to the section on "2 Board Setup" and verify that the board setup is complete again. If the setup is OK, connect the AC adapter to the J14.

_____ 1. In Quartus® Prime, click ***Tools*** menu => ***Programmer*** or ***Programmer*** icon to start Programmer.

_____ 2. Click the [***Hardware Setup***] button in Programmer, select the DE-SoC from the ***currently selected hardware*** pull-down list in the ***Hardware Setup*** window, and close the window.



Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: No Hardware

No Hardware

DE-SoC [USB-1]

Hardware frequency:

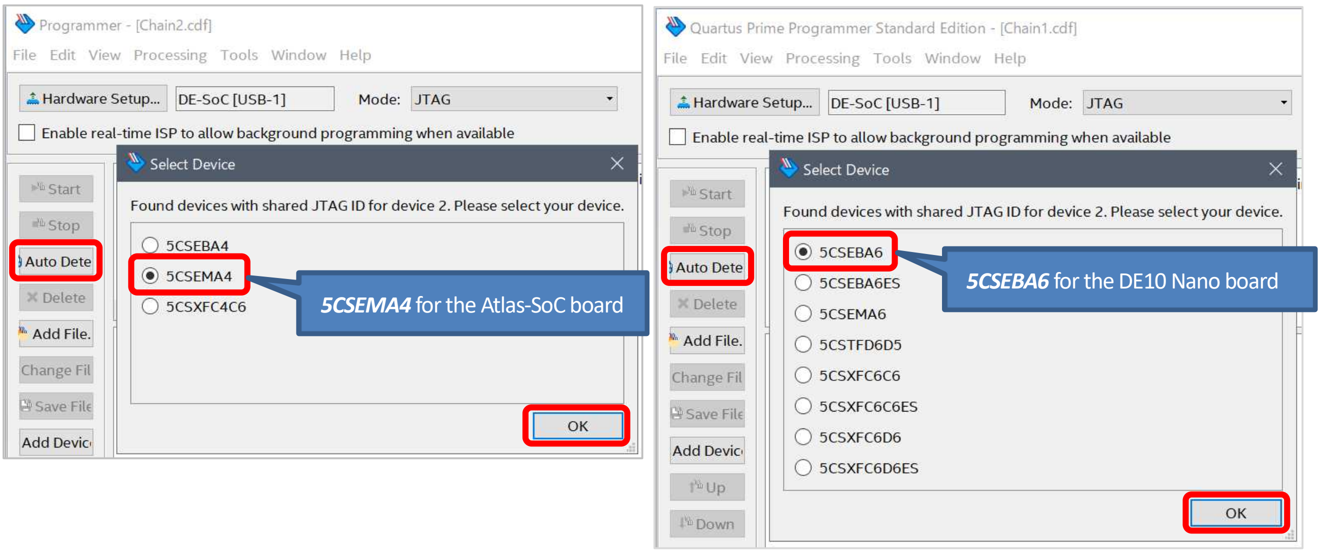
Available hardware items

Hardware	Server	Port
DE-SoC	Local	USB-1

Figure 5-1. Hardware Setup

_____ 3. Click the [***Auto Detect***] button to detect the FPGA connected to the JTAG chain on the board.

_____ 4. Select ***5CSEMA4*** for the Atlas-SoC board or ***5CSEBA6*** for the DE10 Nano board from the ***Select Device*** window, and click [***OK***].


Programmer - [Chain2.cdf]
File Edit View Processing Tools Window Help
Hardware Setup...
DE-SoC [USB-1]
Mode: JTAG
Enable real-time ISP to allow background programming when available
Start
Stop
Auto Dete
Delete
Add File.
Change Fil
Save File
Add Devic
Select Device
OK
5CSEMA4 for the Atlas-SoC board
Quartus Prime Programmer Standard Edition - [Chain1.cdf]
File Edit View Processing Tools Window Help
Hardware Setup...
DE-SoC [USB-1]
Mode: JTAG
Enable real-time ISP to allow background programming when available
Start
Stop
Auto Dete
Delete
Add File.
Change Fil
Save File
Add Devic
Up
Down
Select Device
OK
5CSEBA6 for the DE10 Nano board

Found devices with shared JTAG ID for device 2. Please select your device.

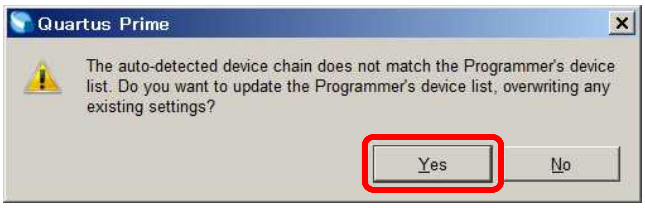
○ 5CSEBA4
● 5CSEMA4
○ 5CSXFC4C6

Found devices with shared JTAG ID for device 2. Please select your device.

● 5CSEBA6
○ 5CSEBA6ES
○ 5CSEMA6
○ 5CSTFD6D5
○ 5CSXFC6C6
○ 5CSXFC6C6ES
○ 5CSXFC6D6
○ 5CSXFC6D6ES

Figure 5-2. Device selection

____ 5. If the following dialog box appears, select [***Yes***].



Quartus Prime

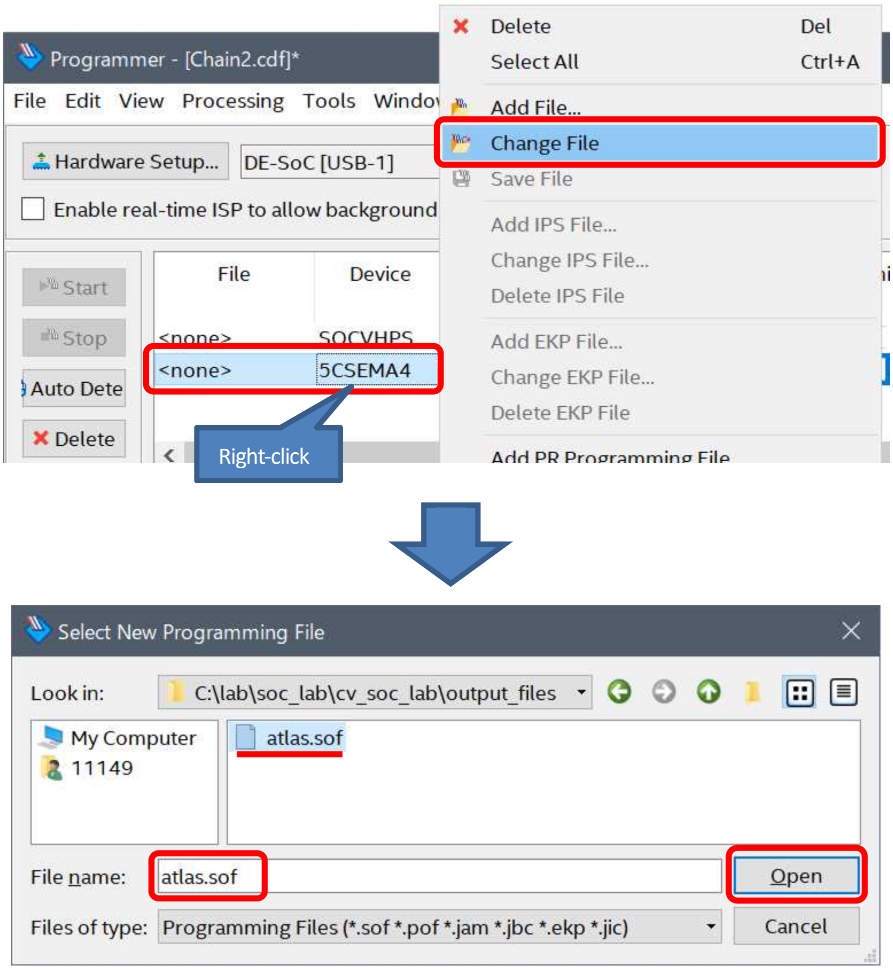
The auto-detected device chain does not match the Programmer's device list. Do you want to update the Programmer's device list, overwriting any existing settings?

Figure 5-3. Dialog box

This displays SOCVHPS and 5CSMA4/5CSEBA6 on the JTAG Chain. SOCVHPS indicates that the HPS side has been recognized, and 5CSMA4/5CSEBA6 indicates that the FPGA side has been recognized.

____ 6. Select the file to download.

Right-click 5CSEMA4/5CSEBA6 in the Device field and click ***Change File***. In the ***Select New Programming File*** dialog box, browse to c:¥lab¥soc_lab¥cv_soc_lab¥output_files and select <u>***atlas.sof*** for an Atlas-SoC board</u> or <u>***DE10 Nano.sof*** for a DE10 Nano board</u>.



Look in: C:\lab\soc_lab\cv_soc_lab\output_files

atlas.sof

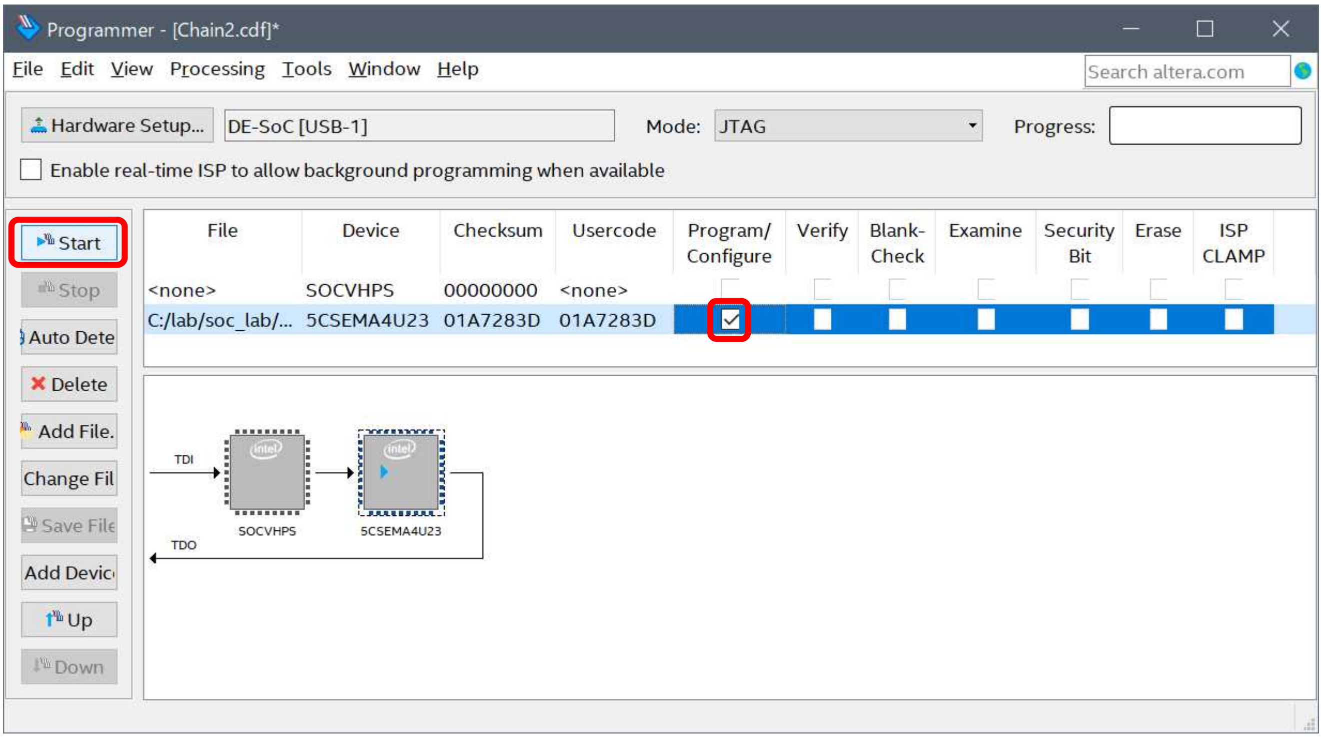
File name: atlas.sof

Files of type: Programming Files (*.sof *.pof *.jam *.jbc *.ekp *.jic)

Figure 5-4. sof file selection

____ 7. Check ***Program/Configure*** and click the [***Start***] button to perform the configuration.

When the Progress par in the upper right reaches 100%, the operation image is written to the FPGA.

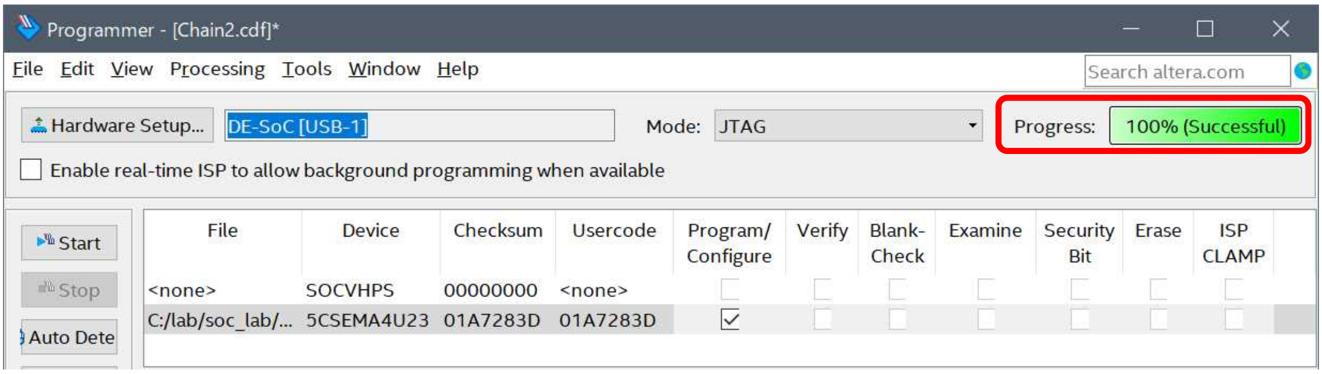


Programmer - [Chain2.cdf]*

Hardware Setup... DE-SoC [USB-1] Mode: JTAG Progress:

□ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
<none>	SOCVHPS	00000000	<none>	□	□	□	□	□	□	□
C:/lab/soc_lab/...	5CSEMA4U23	01A7283D	01A7283D	☑	□	□	□	□	□	□



Programmer - [Chain2.cdf]*

Hardware Setup... DE-SoC [USB-1] Mode: JTAG Progress: 100% (Successful)

□ Enable real-time ISP to allow background programming when available

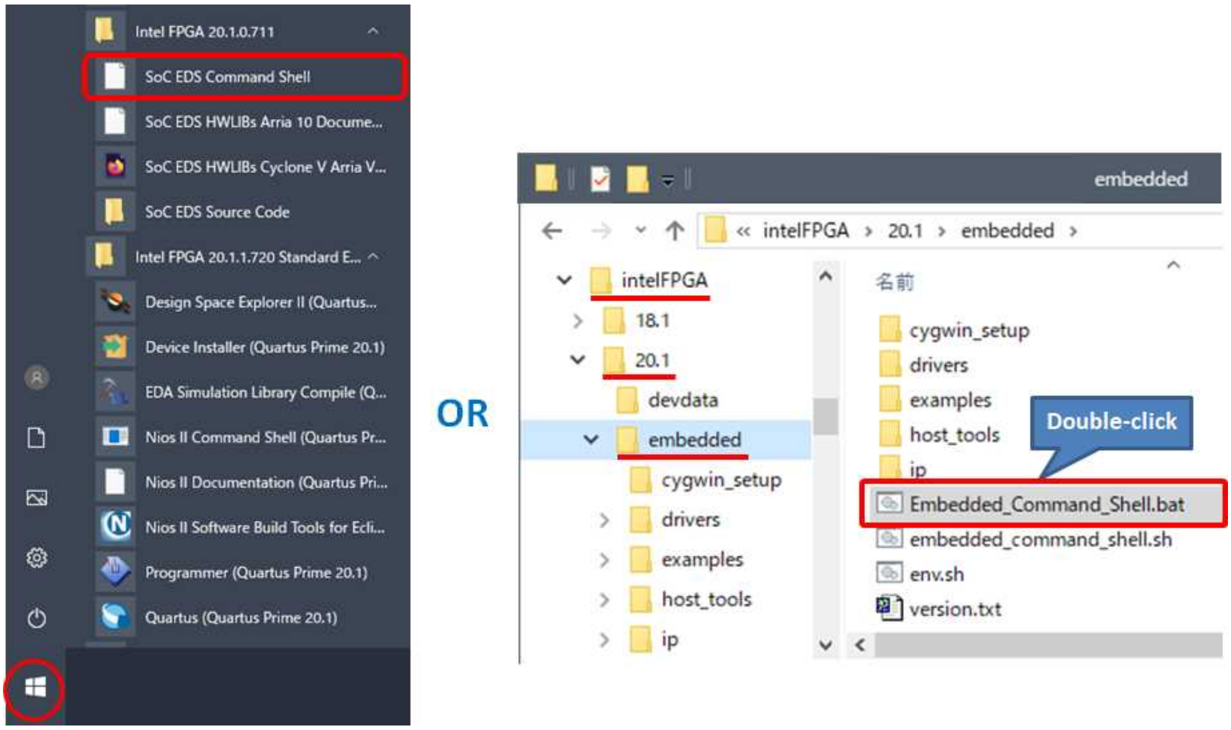
File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
<none>	SOCVHPS	00000000	<none>	□	□	□	□	□	□	□
C:/lab/soc_lab/...	5CSEMA4U23	01A7283D	01A7283D	☑	□	□	□	□	□	□

Figure 5-5. Download sof

5-2. Run the Hello World sample application

Now let's run the sample application on HPS. Start Arm® DS.

____ 1. Start Arm® DS from the Embedded Command Shell included with the SoC EDS. Start the Embedded Command Shell by double-clicking the ***Embedded_Command_Shell.bat*** startup script located in the Windows® Start menu or in the SoC EDS installation folder.

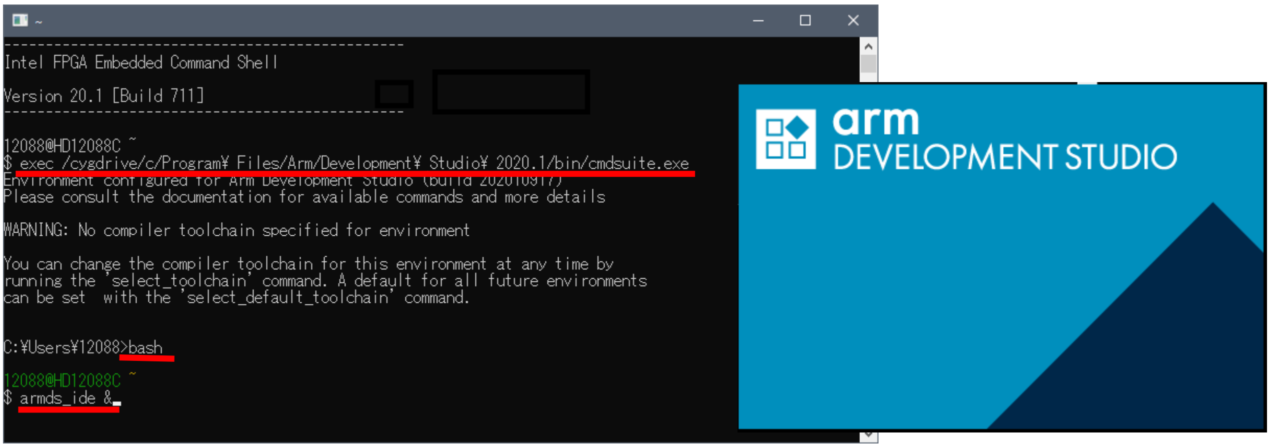


Intel FPGA 20.1.0.711
SoC EDS Command Shell
SoC EDS HWLIBs Arria 10 Docume...
SoC EDS HWLIBs Cyclone V Arria V...
SoC EDS Source Code
Intel FPGA 20.1.1.720 Standard E...
Design Space Explorer II (Quartus...
Device Installer (Quartus Prime 20.1)
EDA Simulation Library Compile (Q...
Nios II Command Shell (Quartus Pr...
Nios II Documentation (Quartus Pri...
Nios II Software Build Tools for Ecli...
Programmer (Quartus Prime 20.1)
Quartus (Quartus Prime 20.1)
OR
embedded
« intelFPGA › 20.1 › embedded ›
intelFPGA
18.1
20.1
devdata
embedded
cygwin_setup
drivers
examples
host_tools
ip
名前
cygwin_setup
drivers
examples
host_tools
ip
Embedded_Command_Shell.bat
embedded_command_shell.sh
env.sh
version.txt
Double-click

Figure 5-6. Starting the Embedded Command Shell

____ 2. Run the following command in the Embedded Command Shell to start Arm® DS.

```
$ exec /cygdrive/c/Program¥ Files/Arm/Development¥ Studio¥ 2020.1/bin/cmdsuite.exe ↵
> bash ↵
$ armds_ide & ↵
```



```
--------------------------------------------------
Intel FPGA Embedded Command Shell

Version 20.1 [Build 711]
--------------------------------------------------

12088@HD12088C ~
$ exec /cygdrive/c/Program¥ Files/Arm/Development¥ Studio¥ 2020.1/bin/cmdsuite.exe
Environment configured for Arm Development Studio (build 202010917)
Please consult the documentation for available commands and more details

WARNING: No compiler toolchain specified for environment

You can change the compiler toolchain for this environment at any time by
running the 'select_toolchain' command. A default for all future environments
can be set  with the 'select_default_toolchain' command.


C:¥Users¥12088>bash

12088@HD12088C ~
$ armds_ide &
```

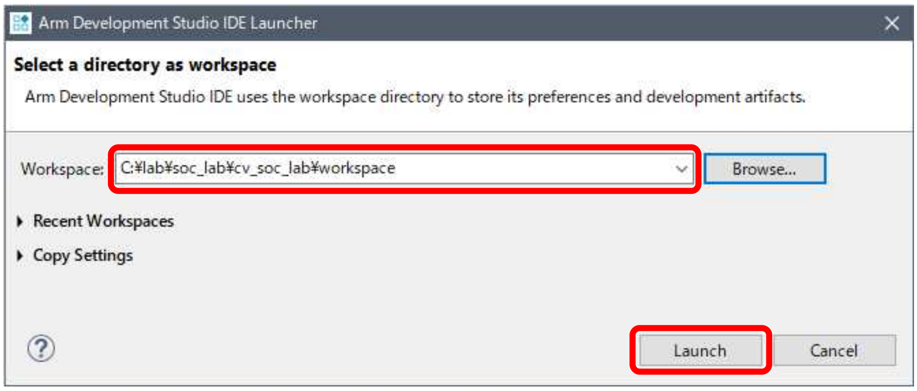
Figure 5-7. Arm® DS Launch and Launch Screen

____ 3. Set up a workspace folder for working with Arm® DS.

In this exercise, you create a workspace in the "3. Lab 1: Hardware Exercise" working folder.

Specify the following path and click [***Launch***] (if the folder does not exist, it will be created automatically):

```
C:¥lab¥soc_lab¥cv_soc_lab¥workspace
```



Arm Development Studio IDE Launcher

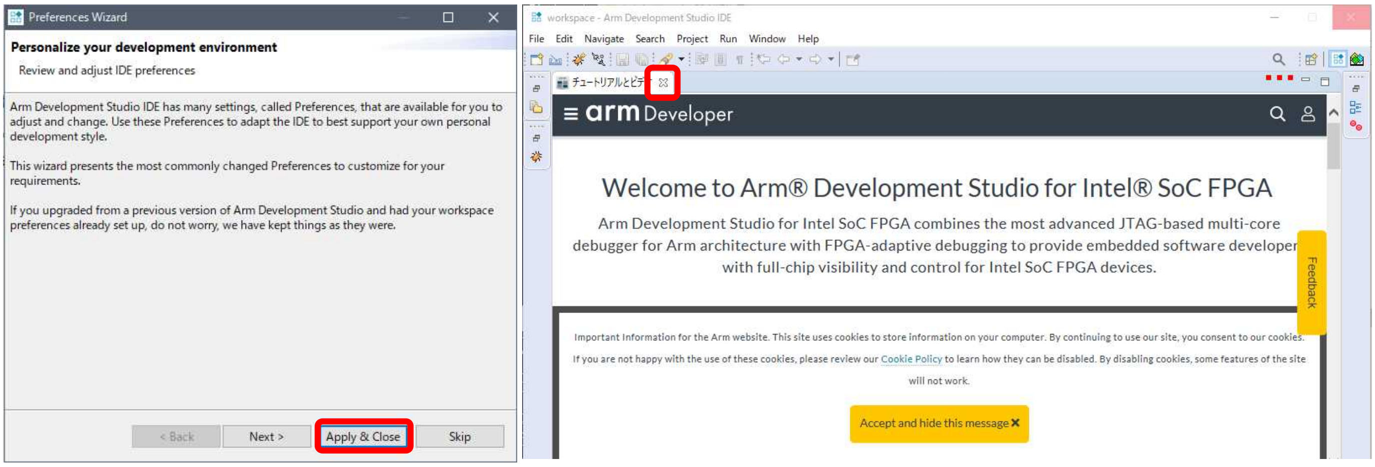
Select a directory as workspace

Arm Development Studio IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:¥lab¥soc_lab¥cv_soc_lab¥workspace

Figure 5-8 Creating a Workspace

____ 4. If the Preferences Wizard appears, confirm the contents and click [***Apply & Close***]. If the Arm® DS Welcome screen appears, click **Close** (**X**) to close it.



Preferences Wizard

Personalize your development environment

Review and adjust IDE preferences

Arm Development Studio IDE has many settings, called Preferences, that are available for you to adjust and change. Use these Preferences to adapt the IDE to best support your own personal development style.

This wizard presents the most commonly changed Preferences to customize for your requirements.

If you upgraded from a previous version of Arm Development Studio and had your workspace preferences already set up, do not worry, we have kept things as they were.

Welcome to Arm® Development Studio for Intel® SoC FPGA

Arm Development Studio for Intel SoC FPGA combines the most advanced JTAG-based multi-core debugger for Arm architecture with FPGA-adaptive debugging to provide embedded software developer with full-chip visibility and control for Intel SoC FPGA devices.

Important Information for the Arm website. This site uses cookies to store information on your computer. By continuing to use our site, you consent to our cookies. If you are not happy with the use of these cookies, please review our Cookie Policy to learn how they can be disabled. By disabling cookies, some features of the site will not work.

Figure 5-9. Preference Wizard and Welcome Screen

ⓘ **Info:**

When displaying the Welcome screen (tutorial and video) online in Arm® DS (2020.1), it has been confirmed that window operations become unresponsive. Do not force the operation until the online display is complete. After closing the Welcome screen, you will be able to operate smoothly.

If you start Arm® DS offline, the offline Welcome screen will be used to avoid this problem.

Next, import the Hello World sample application.

The Hello World sample application is included in the SoC EDS as a Software Example.

____ 5. From the Arm® DS menu, select ***File*** => ***Import***.

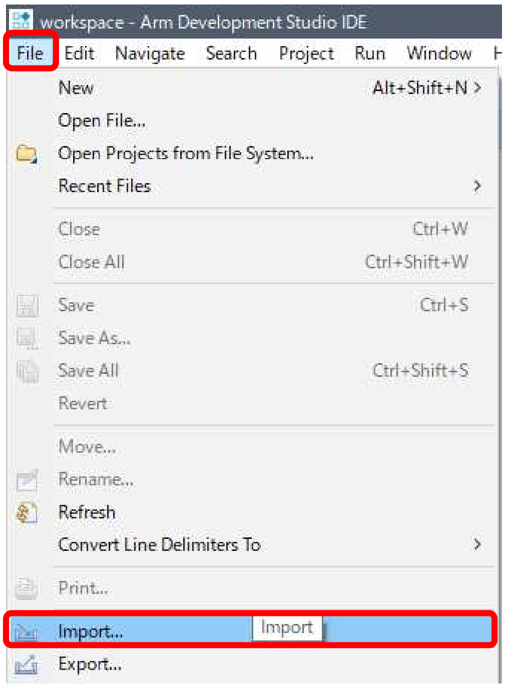


Figure 5-10. Import menu

____ 6. Select "***Existing Projects into Workspaces***" and click [***Next***].

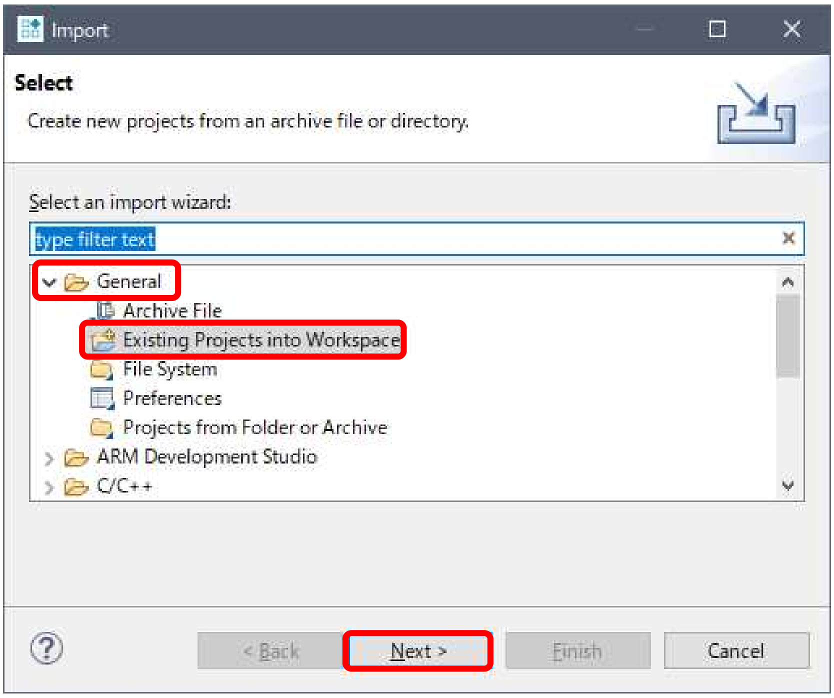


Figure 5-11. Import an existing project

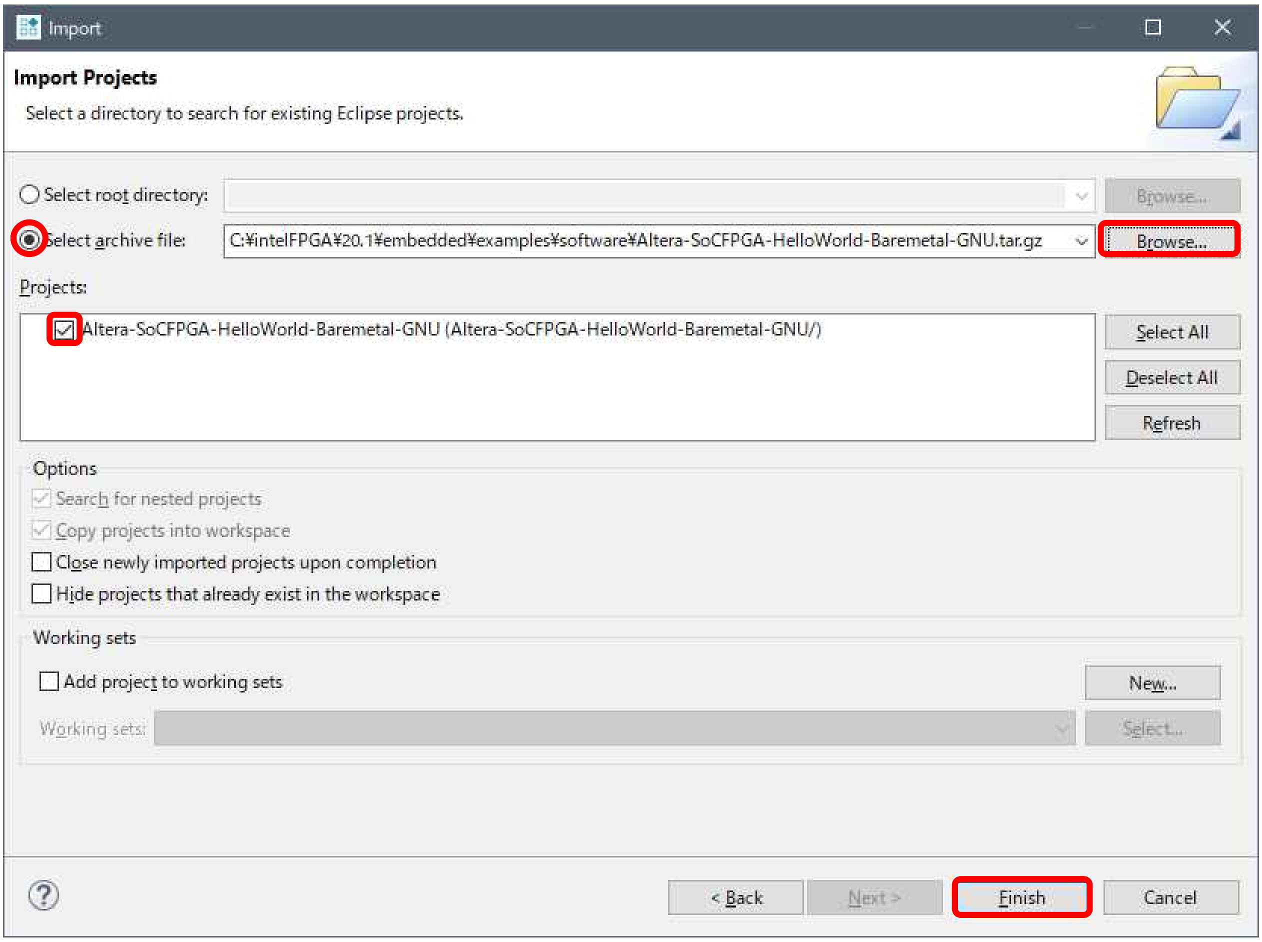
____ 7. Select archive file: option and use the [***Browse***] button to locate the sample project.

The sample project is included with SoC EDS and can be found by default in the following installation folder:

```
C:¥intelFPGA¥20.1¥embedded¥examples¥software¥Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar.gz
```

(Importing <SoC EDS installation directory >`¥examples¥software¥Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar.gz`).

Once selected, click the [***Finish***] button.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory:

● Select archive file: C:¥intelFPGA¥20.1¥embedded¥examples¥software¥Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar.gz

Projects:

☑ Altera-SoCFPGA-HelloWorld-Baremetal-GNU (Altera-SoCFPGA-HelloWorld-Baremetal-GNU/)

Options

☑ Search for nested projects

☑ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets:

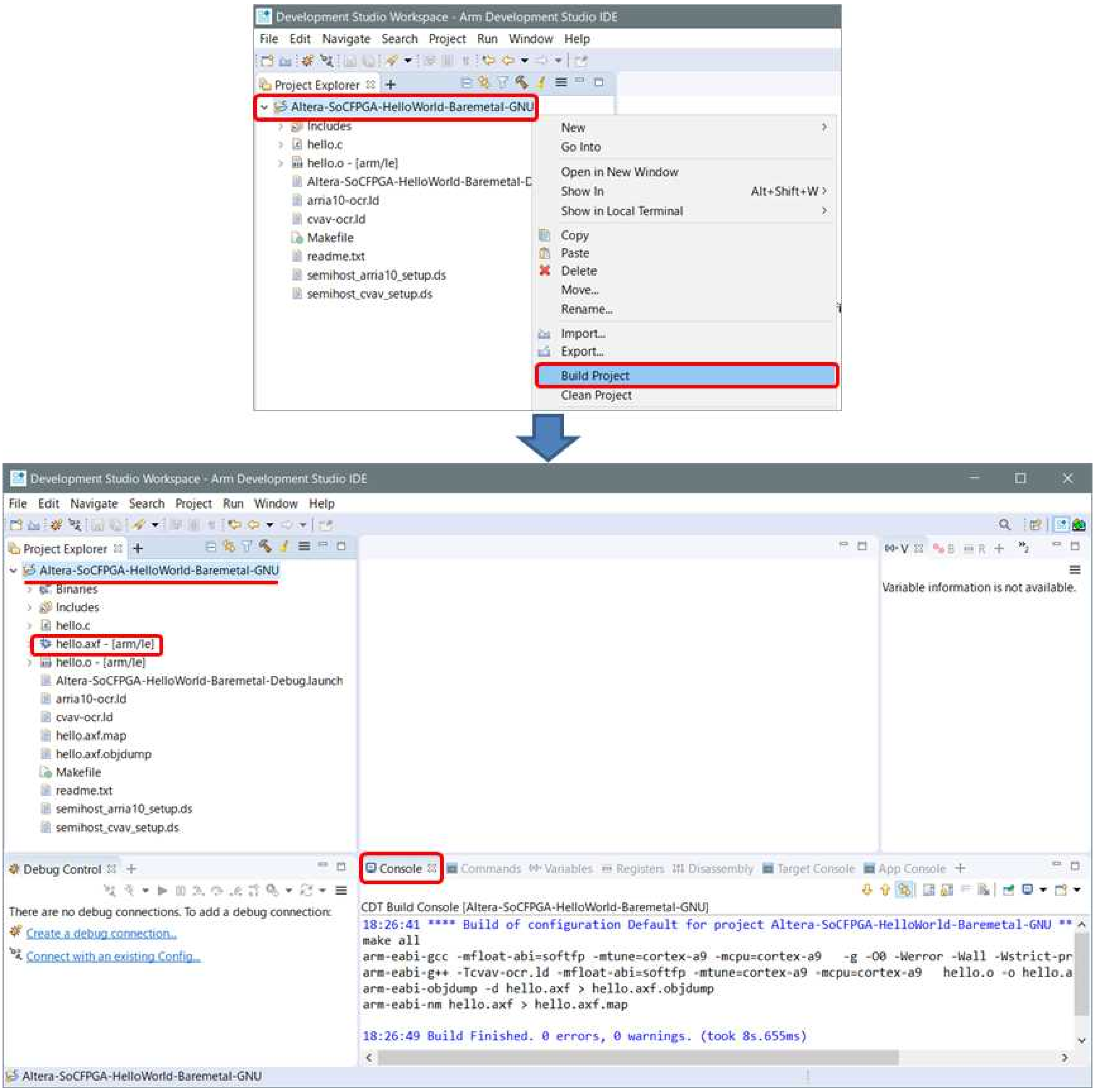
Figure 5-12. Selecting the Hello World sample application

After you complete this task, the Project Explorer on the left side of the Arm® DS window displays the various files contained in the project.

Next, compile the Hello World sample application.

____ 8. Select and highlight the project from the Project Explorer tab.

____ 9. From the Arm® DS menu, choose ***Project*** => ***Build Project***. Alternatively, select the project in the Project Explorer and **right-click** => ***Build Project***.


Development Studio Workspace - Arm Development Studio IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
Altera-SoCFPGA-HelloWorld-Baremetal-GNU
Includes
hello.c
hello.o - [arm/le]
Altera-SoCFPGA-HelloWorld-Baremetal-D
arria10-ocr.ld
cvav-ocr.ld
Makefile
readme.txt
semihost_arria10_setup.ds
semihost_cvav_setup.ds
New
Go Into
Open in New Window
Show In Alt+Shift+W
Show in Local Terminal
Copy
Paste
Delete
Move...
Rename...
Import...
Export...
Build Project
Clean Project
Binaries
hello.axf - [arm/le]
Altera-SoCFPGA-HelloWorld-Baremetal-Debug.launch
hello.axf.map
hello.axf.objdump
Variable information is not available.
Debug Control
There are no debug connections. To add a debug connection:
Create a debug connection...
Connect with an existing Config...
Console
Commands
Variables
Registers
Disassembly
Target Console
App Console

CDT Build Console [Altera-SoCFPGA-HelloWorld-Baremetal-GNU]

```
18:26:41 **** Build of configuration Default for project Altera-SoCFPGA-HelloWorld-Baremetal-GNU **
make all
arm-eabi-gcc -mfloat-abi=softfp -mtune=cortex-a9 -mcpu=cortex-a9   -g -O0 -Werror -Wall -Wstrict-pr
arm-eabi-g++ -Tcvav-ocr.ld -mfloat-abi=softfp -mtune=cortex-a9 -mcpu=cortex-a9   hello.o -o hello.a
arm-eabi-objdump -d hello.axf > hello.axf.objdump
arm-eabi-nm hello.axf > hello.axf.map

18:26:49 Build Finished. 0 errors, 0 warnings. (took 8s.655ms)
```

Figure 5-13. Building the Hello World project

The project is compiled, and the Project Explorer contains a file named ***hello.axf,*** as shown in the figure above. The executable binary on Arm® DS is output.

The Console window shows the command executed to generate the executable binary.

Finally, run the Hello World sample application (***hello.axf***) generated earlier.

____ 10. Choose ***Run*** menu => ***Debug Configuration (B).***

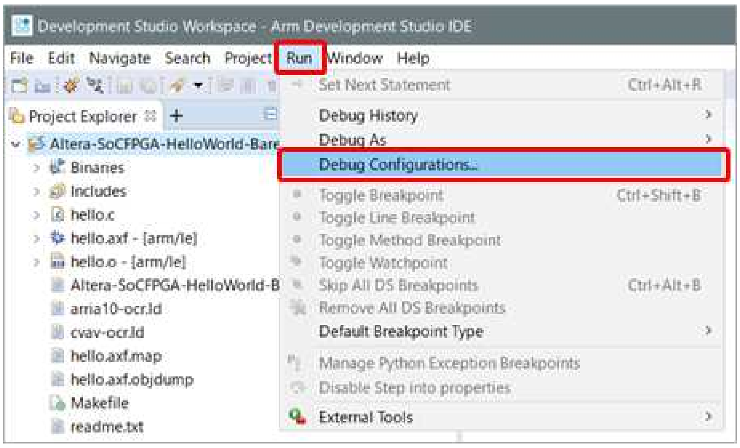


Figure 5-14. Selecting the Debug Configuration

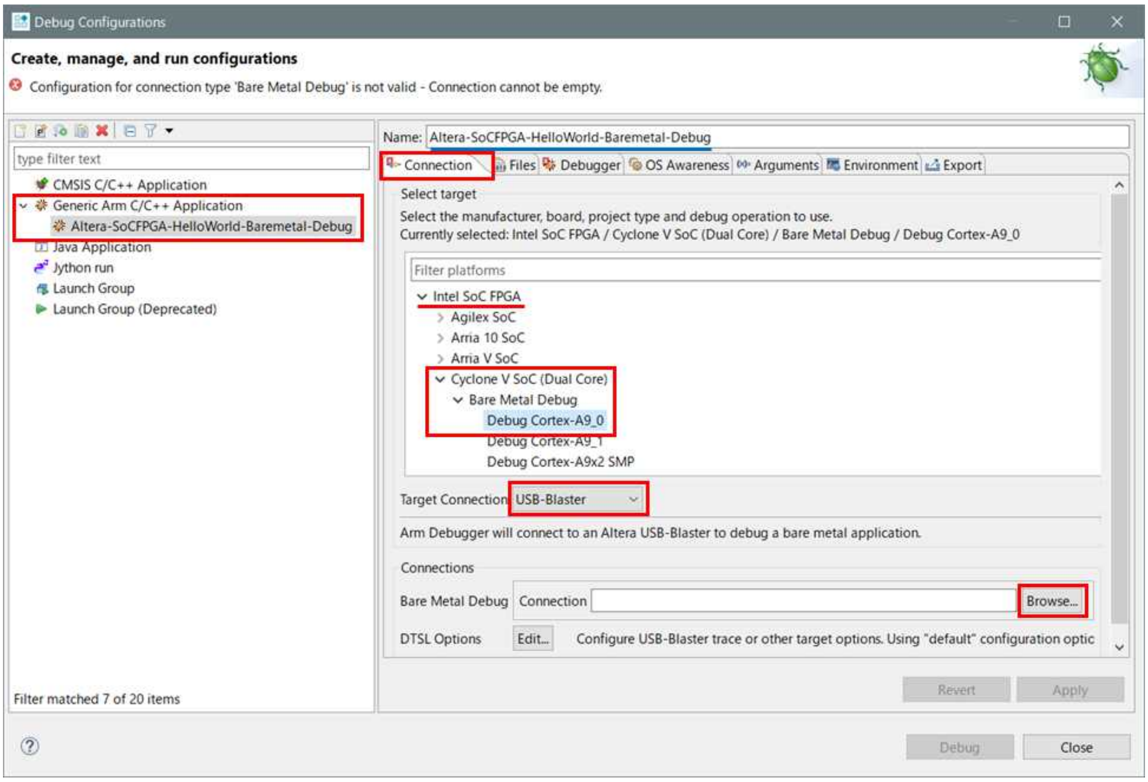
____ 11. From the left panel of the Debug Configurations window, select

Generic ARM C/C++ Applications => ***Altera-SoCFPGA-HelloWorld-Baremetal-Debug*** (If you do not see it, click (>) next to Generic ARM C/C++ Application).

The target connection uses the Intel ® FPGA Download Cable (USB-Blaster™).

Intel SoC FPGA => ***Cyclone V SoC (Dual Core)*** => ***Bare Metal Debug*** => ***Debug Cortex-A9_0***.

____ 12. Press the [***Browse***] button to the right of the connection section to bring up the selection screen for the USB-Blaster™ connection.



Debug Configurations

Create, manage, and run configurations

Configuration for connection type 'Bare Metal Debug' is not valid - Connection cannot be empty.

Name: Altera-SoCFPGA-HelloWorld-Baremetal-Debug

Select target

Select the manufacturer, board, project type and debug operation to use.
Currently selected: Intel SoC FPGA / Cyclone V SoC (Dual Core) / Bare Metal Debug / Debug Cortex-A9_0

Target Connection: USB-Blaster

Arm Debugger will connect to an Altera USB-Blaster to debug a bare metal application.

Connections

Bare Metal Debug Connection:

DTSL Options Edit... Configure USB-Blaster trace or other target options. Using "default" configuration optic

Filter matched 7 of 20 items

Figure 5-15. Debug configuration (connection tab)

____ 13. In the connection browser window, highlight the desired USB-Blaster™ (in this example, DE-SoC on localhost) and click [***Select***].

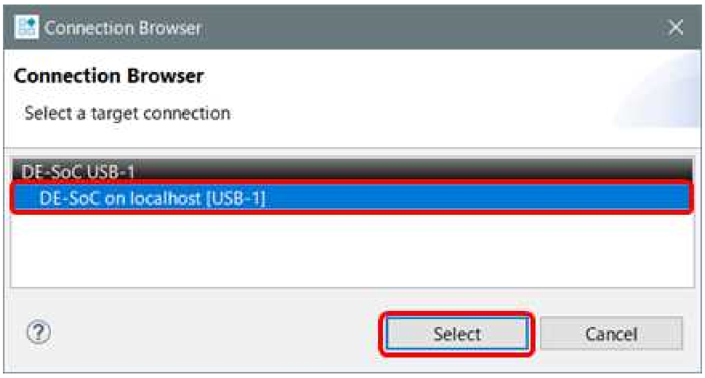
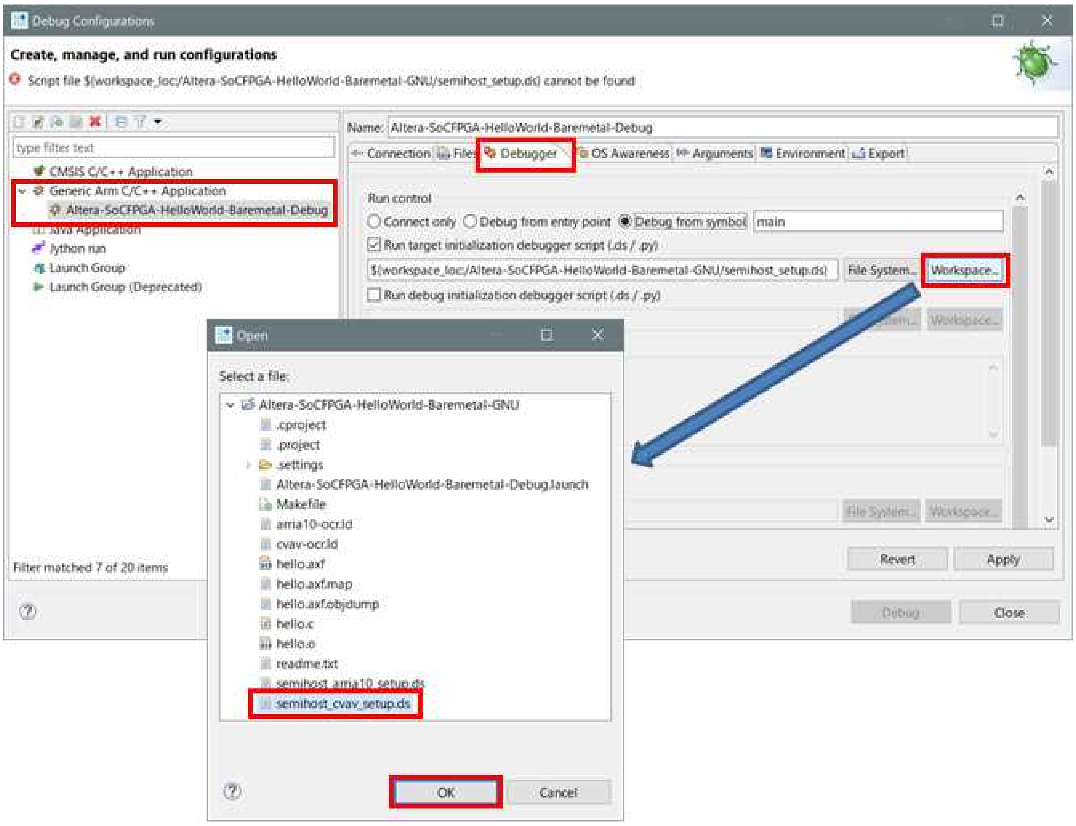


Figure 5-16. Select debug cable

____ 14. Modify semihost_setup.ds specified in the target initialization debugger script (.ds/.py) in the Debugger tab. Select semihost_cvav_setup.ds included in the project from the [***Workspace***] button and click [***OK***].


Create, manage, and run configurations

Script file ${workspace_loc:/Altera-SoCFPGA-HelloWorld-Baremetal-GNU/semihost_setup.ds} cannot be found

Name: Altera-SoCFPGA-HelloWorld-Baremetal-Debug

○ Connect only ○ Debug from entry point ● Debug from symbol main

☑ Run target initialization debugger script (.ds / .py)

${workspace_loc:/Altera-SoCFPGA-HelloWorld-Baremetal-GNU/semihost_setup.ds}

□ Run debug initialization debugger script (.ds / .py)

Filter matched 7 of 20 items

Figure 5-17. Debug configuration (Debugger tab)

____ 15. Click the Debug button in the lower right corner of the [***Debug***] configuration window.

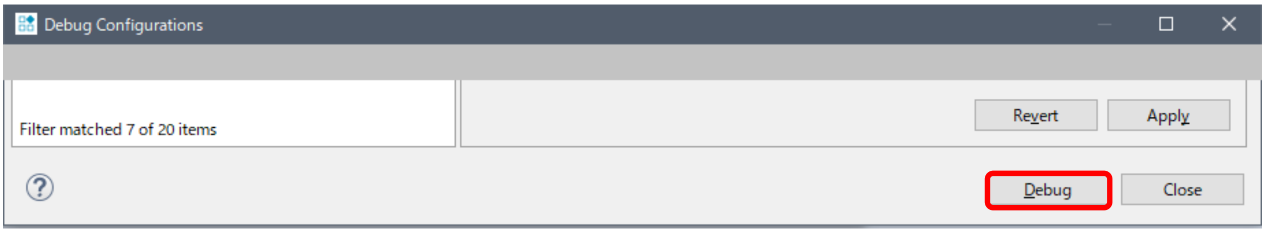
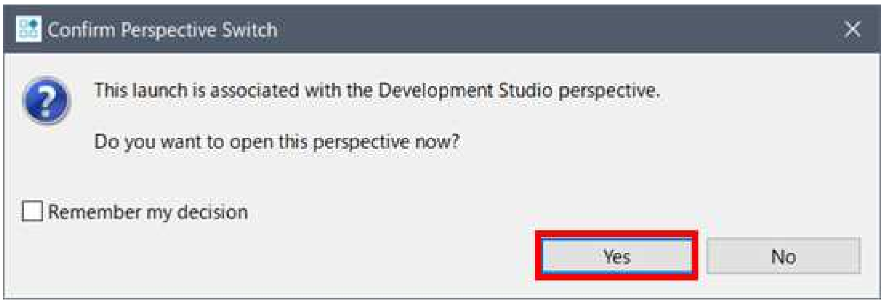

Filter matched 7 of 20 items

Figure 5-18. Running debugging

____ 16. If prompted for a perspective switch, click [***Yes***] to accept it.



Confirm Perspective Switch

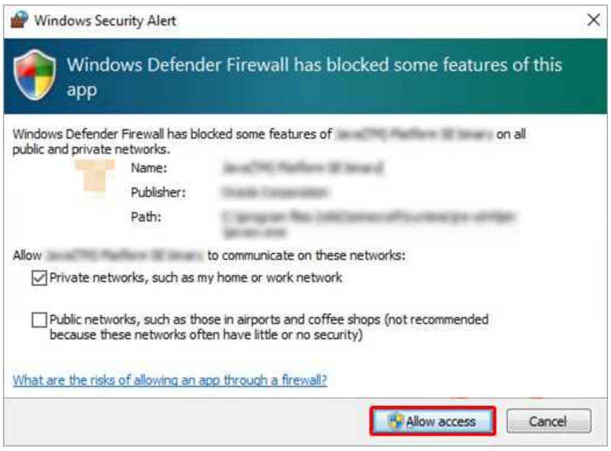
This launch is associated with the Development Studio perspective.

Do you want to open this perspective now?

□ Remember my decision

Figure 5-19. Check the perspective switch

If you receive a Windows Defender Firewall warning, click [***Allow access***].



Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of [redacted] on all public and private networks.

Name: [redacted]

Publisher: [redacted]

Path: [redacted]

Allow [redacted] to communicate on these networks:

☑ Private networks, such as my home or work network

□ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

Figure 5-20. Security warning

ⓘ **Info:**

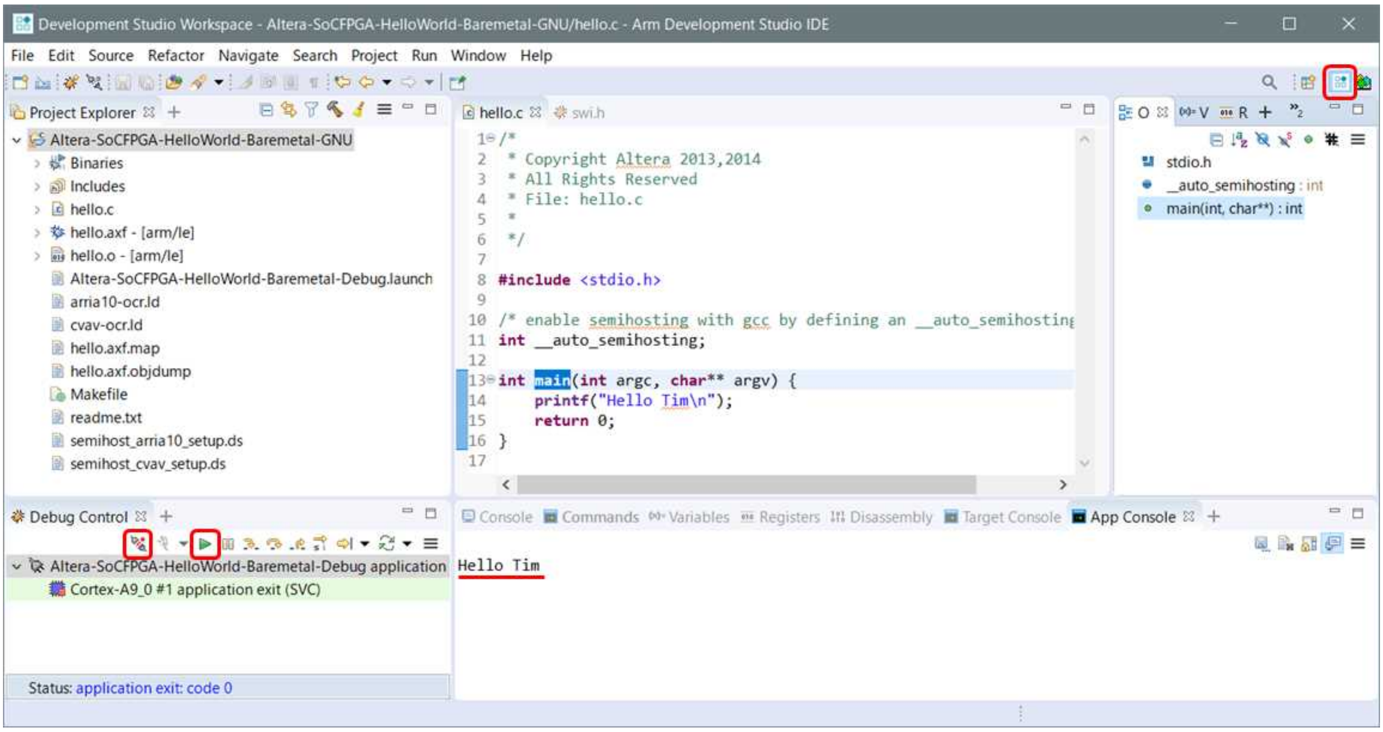
If you receive a download error, check the following:

(1) Make sure that the network interface (for example, USB-Ethernet Interface Adapter) to which the Arm® DS is licensed is enabled.

(2) Make sure that the evaluation board is powered off and that the PC is rebooted. If the evaluation board is powered off, remember to download the FPGA data again.

The debugger will follow the instructions in the startup script to enable the semi-hosting feature and then download the application to the board via JTAG. When the program counter reaches the main function, it breaks and you are ready to start debugging. At this point, you can use all the debugging features of Arm® DS (View and edit registers and variables, reference disassembly code, etc.).

____ 17. The green ***Continue*** button ▶ Click (or press F8) to run the application. This will display a **Hello Tim** message in the ***App Console***.


Development Studio Workspace - Altera-SoCFPGA-HelloWorld-Baremetal-GNU/hello.c - Arm Development Studio IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Altera-SoCFPGA-HelloWorld-Baremetal-GNU
Binaries
Includes
hello.c
hello.axf - [arm/le]
hello.o - [arm/le]
Altera-SoCFPGA-HelloWorld-Baremetal-Debug.launch
arria10-ocr.ld
cvav-ocr.ld
hello.axf.map
hello.axf.objdump
Makefile
readme.txt
semihost_arria10_setup.ds
semihost_cvav_setup.ds
hello.c
swi.h
O
V
R
stdio.h
_auto_semihosting : int
main(int, char**) : int
Debug Control
Altera-SoCFPGA-HelloWorld-Debug application
Cortex-A9_0 #1 application exit (SVC)
Console
Commands
Variables
Registers
Disassembly
Target Console
App Console

```
/*
 * Copyright Altera 2013,2014
 * All Rights Reserved
 * File: hello.c
 *
 */

#include <stdio.h>

/* enable semihosting with gcc by defining an __auto_semihosting
int __auto_semihosting;

int main(int argc, char** argv) {
    printf("Hello Tim\n");
    return 0;
}
```

```
Hello Tim
```

Status: application exit: code 0

Figure 5-21. Displaying Hello Tim

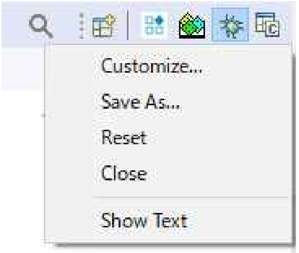
____ 18. Disconnect button to disconnect from the CPU.

____ 19. If you are changing the perspective in the upper right corner of the screen, click to return to the main screen.

ⓘ **Info:**

You can use the perspective menu in the upper right corner of the Arm® DS window to store several different screen layouts. You can switch between perspectives depending on what you are doing.

button to open a new perspective. The perspectives are listed in icon form in the upper right corner and can be switched by clicking the icon. You can also reset the selected perspective by right-clicking it.



5-3. Running the LED Blink Sample Application

Import the pre-installed LED Blink sample application, similar to the Hello World sample application, into Arm® DS.

____ 1. From the Arm® DS menu, select ***File*** => ***Import***.

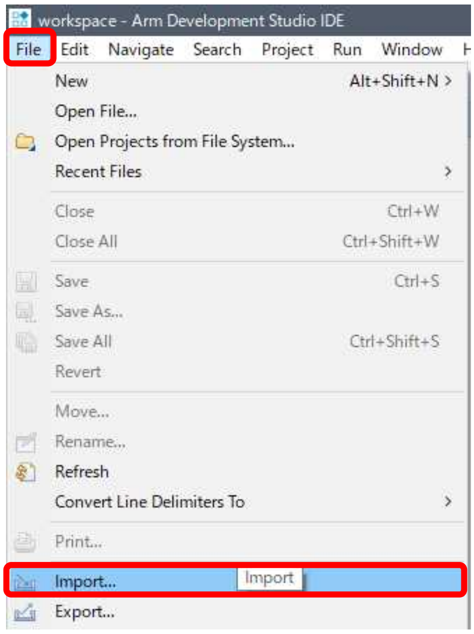


Figure 5-22. Import menu

____ 2. ***General*** => "***Existing Projects into Workspaces***" Select and click [***Next***].

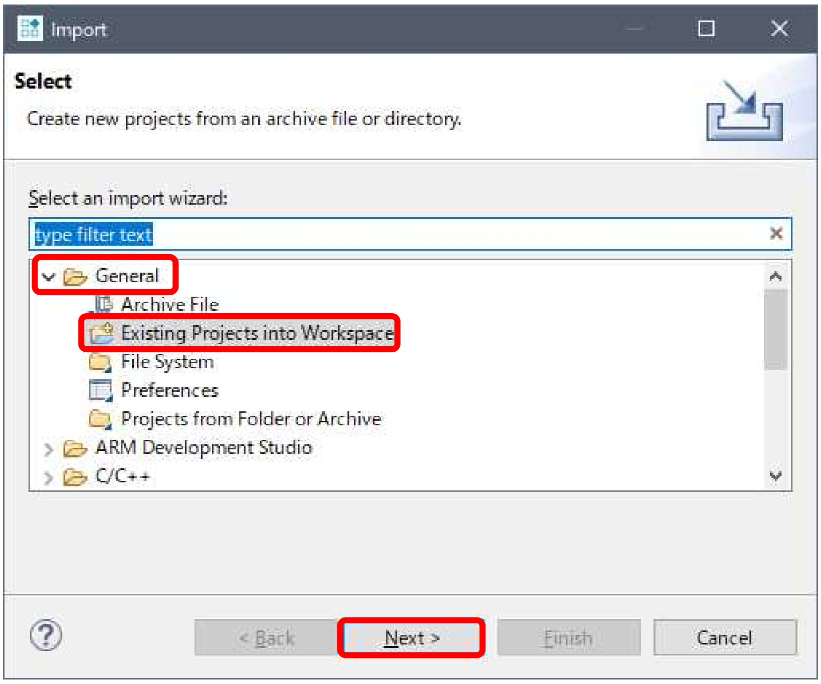


Figure 5-23. Import an existing project

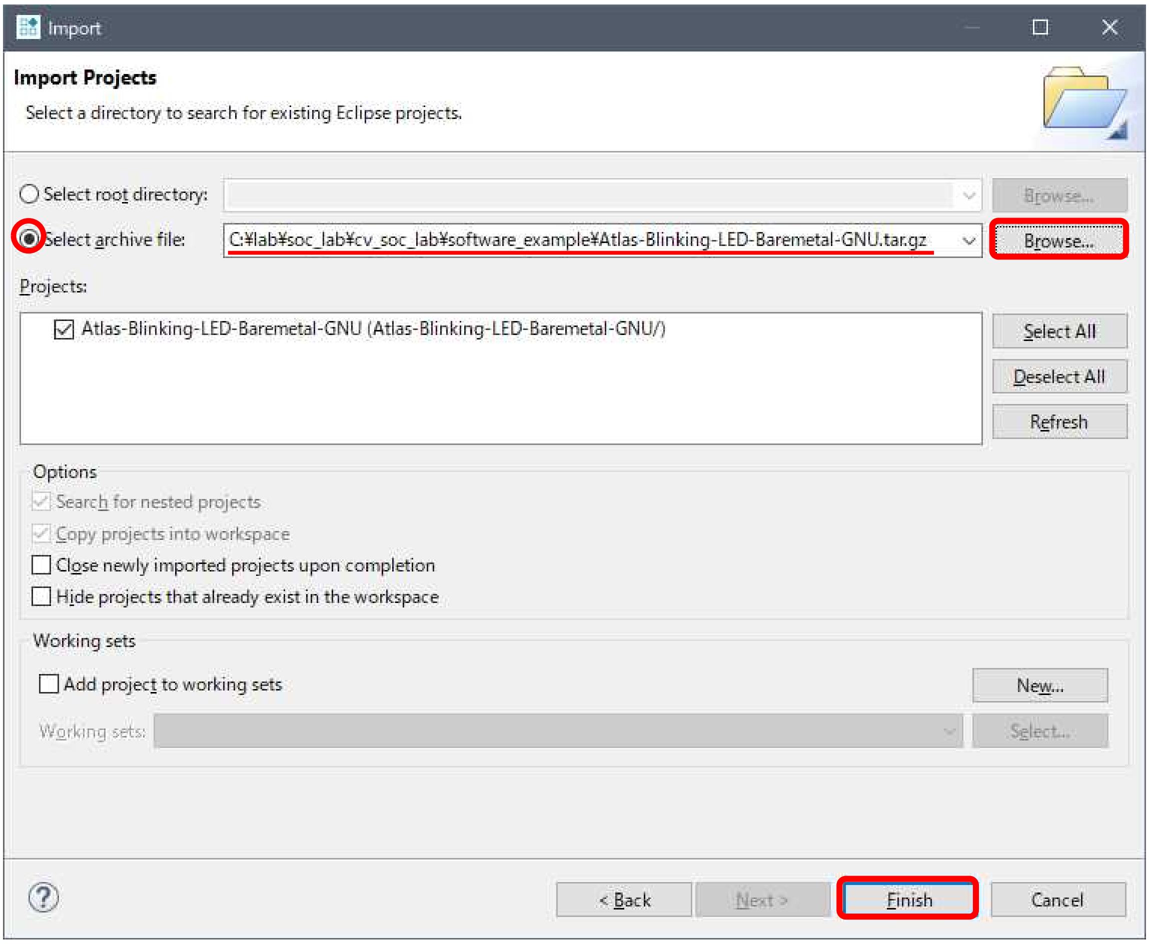
____ 3. Select the *Select archive file:* option and use the [***Browse***] button to locate the sample project.

```
C:¥lab¥soc_lab¥cv_soc_lab¥software_example¥Atlas-Blinking-LED-Baremetal-GNU.tar.gz
```

ⓘ **Info:**

Note that this is **under the exercise data directory**, not the tool installation directory.

After making your selections, press the [***Finish***] button.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory:

● Select archive file: C:¥lab¥soc_lab¥cv_soc_lab¥software_example¥Atlas-Blinking-LED-Baremetal-GNU.tar.gz

Projects:

☑ Atlas-Blinking-LED-Baremetal-GNU (Atlas-Blinking-LED-Baremetal-GNU/)

Options

☑ Search for nested projects

☑ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets:

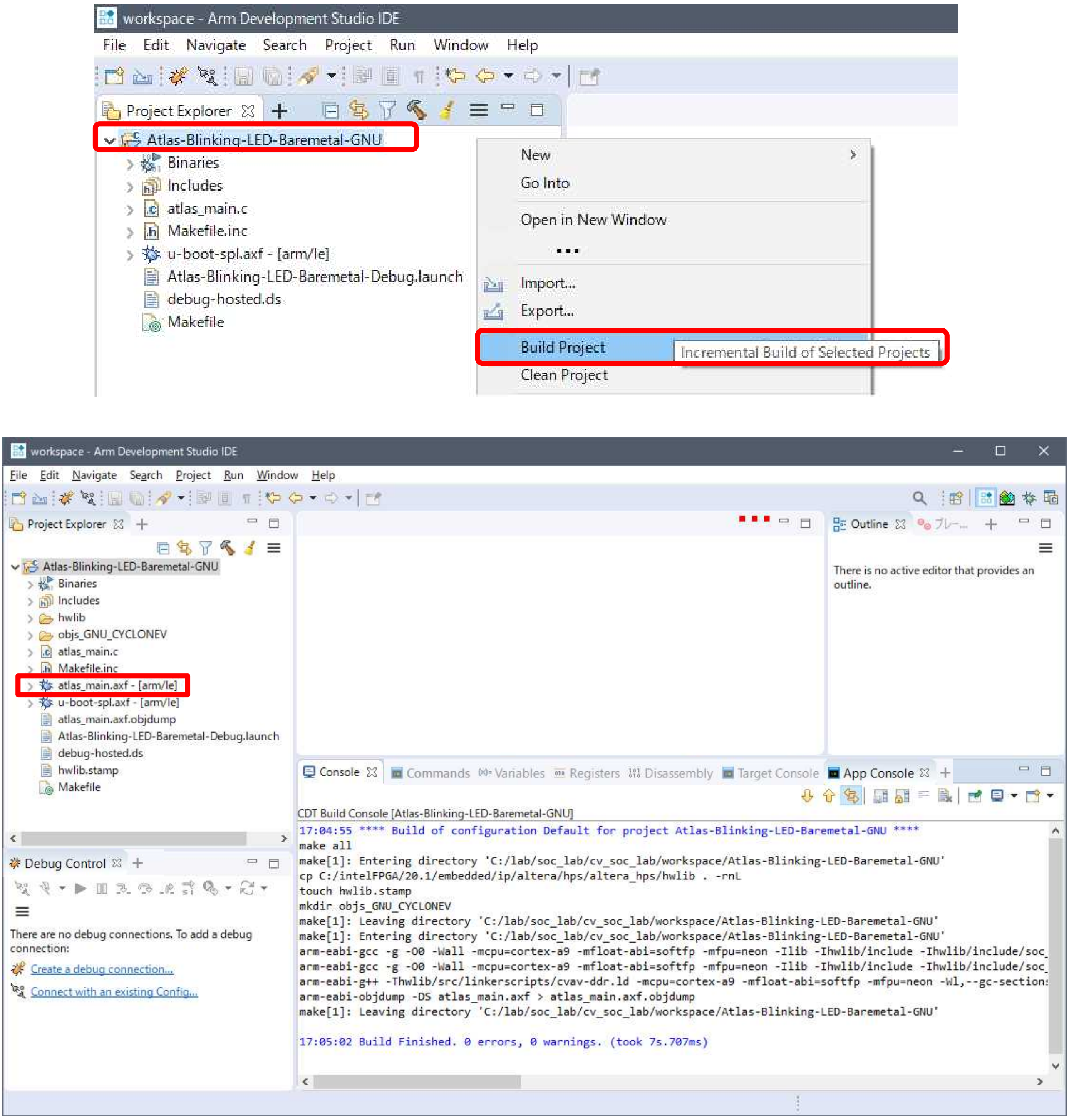
Figure 5-24. Selecting the LED Blink sample application

After completing this task, the Project Explorer on the left side of the Arm® DS window displays the various files contained in the project.

Next, compile the LED Blink sample application.

____ 4. From the *Project Explorer* tab, select and highlight the Atlas-Blinking-LED-Baremetal-GNU project.

____ 5. From the Arm® DS menu, select ***Project*** => ***Build Project***. Alternatively, select the project in the Project Explorer and **right-click** => ***Build Project***.


workspace - Arm Development Studio IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
Atlas-Blinking-LED-Baremetal-GNU
Binaries
Includes
atlas_main.c
Makefile.inc
u-boot-spl.axf - [arm/le]
Atlas-Blinking-LED-Baremetal-Debug.launch
debug-hosted.ds
Makefile
New
Go Into
Open in New Window
...
Import...
Export...
Build Project
Incremental Build of Selected Projects
Clean Project
workspace - Arm Development Studio IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
Atlas-Blinking-LED-Baremetal-GNU
Binaries
Includes
hwlib
objs_GNU_CYCLONEV
atlas_main.c
Makefile.inc
atlas_main.axf - [arm/le]
u-boot-spl.axf - [arm/le]
atlas_main.axf.objdump
Atlas-Blinking-LED-Baremetal-Debug.launch
debug-hosted.ds
hwlib.stamp
Makefile
Outline
There is no active editor that provides an outline.
Debug Control
There are no debug connections. To add a debug connection:
Create a debug connection...
Connect with an existing Config...
Console
Commands
Variables
Registers
Disassembly
Target Console
App Console

CDT Build Console [Atlas-Blinking-LED-Baremetal-GNU]

```
17:04:55 **** Build of configuration Default for project Atlas-Blinking-LED-Baremetal-GNU ****
make all
make[1]: Entering directory 'C:/lab/soc_lab/cv_soc_lab/workspace/Atlas-Blinking-LED-Baremetal-GNU'
cp C:/intelFPGA/20.1/embedded/ip/altera/hps/altera_hps/hwlib . -rnL
touch hwlib.stamp
mkdir objs_GNU_CYCLONEV
make[1]: Leaving directory 'C:/lab/soc_lab/cv_soc_lab/workspace/Atlas-Blinking-LED-Baremetal-GNU'
make[1]: Entering directory 'C:/lab/soc_lab/cv_soc_lab/workspace/Atlas-Blinking-LED-Baremetal-GNU'
arm-eabi-gcc -g -O0 -Wall -mcpu=cortex-a9 -mfloat-abi=softfp -mfpu=neon -Ilib -Ihwlib/include -Ihwlib/include/soc_
arm-eabi-gcc -g -O0 -Wall -mcpu=cortex-a9 -mfloat-abi=softfp -mfpu=neon -Ilib -Ihwlib/include -Ihwlib/include/soc_
arm-eabi-g++ -Thwlib/src/linkerscripts/cvav-ddr.ld -mcpu=cortex-a9 -mfloat-abi=softfp -mfpu=neon -Wl,--gc-section
arm-eabi-objdump -DS atlas_main.axf > atlas_main.axf.objdump
make[1]: Leaving directory 'C:/lab/soc_lab/cv_soc_lab/workspace/Atlas-Blinking-LED-Baremetal-GNU'

17:05:02 Build Finished. 0 errors, 0 warnings. (took 7s.707ms)
```

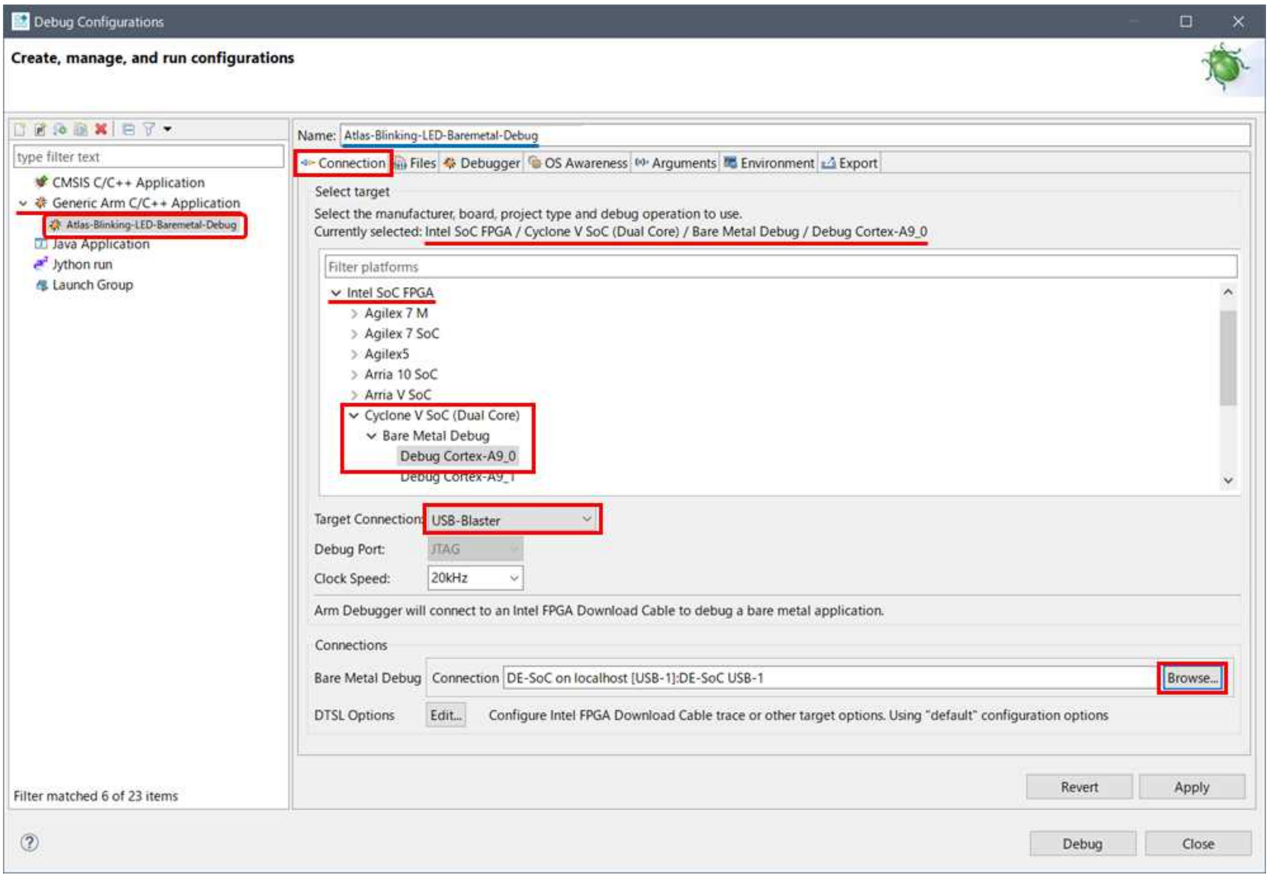
Figure 5-25. Building the LED Blink Sample Application

Finally, run the LED Blink sample application.

____ 6. Choose ***Run*** menu => ***Debug Configuration (B).*** The sample project comes with pre-configuration to run on an Atlas-SoC board.

____ 7. From the left panel of the Debug Configurations window, select
Generic ARM C/C++ Application => ***Atlas-Blinking-LED-Baremetal-Debug*** (If you do not see it, click (>) next to Generic ARM C/C++ Application).

The target connection is already configured using USB-Blaster™.
Intel SoC FPGA => Cyclone V SoC (Dual Core) => Bare Metal Debug => Debug Cortex-A9_0.



Debug Configurations

Create, manage, and run configurations

type filter text

- CMSIS C/C++ Application
- Generic Arm C/C++ Application
  - Atlas-Blinking-LED-Baremetal-Debug
- Java Application
- Jython run
- Launch Group

Filter matched 6 of 23 items

Name: Atlas-Blinking-LED-Baremetal-Debug

Connection | Files | Debugger | OS Awareness | Arguments | Environment | Export

Select target

Select the manufacturer, board, project type and debug operation to use.
Currently selected: Intel SoC FPGA / Cyclone V SoC (Dual Core) / Bare Metal Debug / Debug Cortex-A9_0

Filter platforms

- Intel SoC FPGA
  - Agilex 7 M
  - Agilex 7 SoC
  - Agilex5
  - Arria 10 SoC
  - Arria V SoC
  - Cyclone V SoC (Dual Core)
    - Bare Metal Debug
      - Debug Cortex-A9_0
      - Debug Cortex-A9_1

Target Connection: USB-Blaster
Debug Port: JTAG
Clock Speed: 20kHz

Arm Debugger will connect to an Intel FPGA Download Cable to debug a bare metal application.

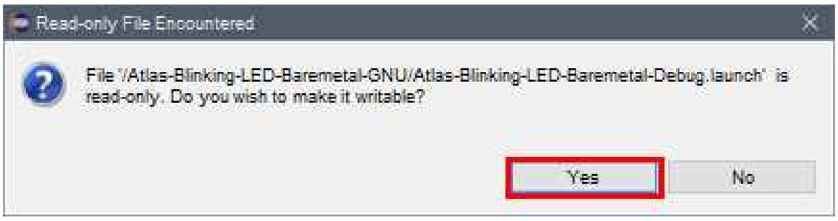
Connections

Bare Metal Debug Connection DE-SoC on localhost [USB-1]:DE-SoC USB-1 Browse...

DTSL Options Edit... Configure Intel FPGA Download Cable trace or other target options. Using "default" configuration options

Figure 5-26. Debug configuration of the LED Blink sample application

____ 8. If the following confirmation popup appears, select [***Yes***].



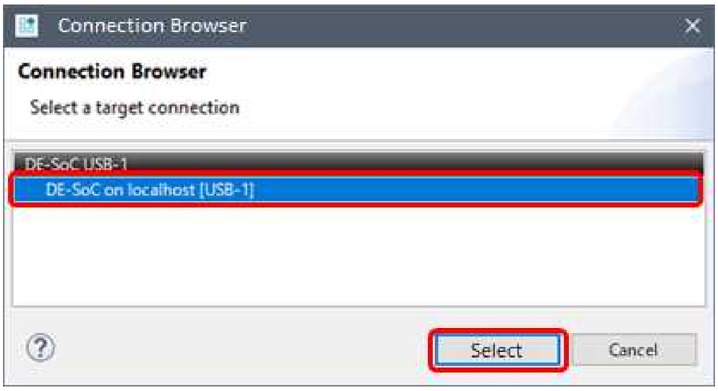
Read-only File Encountered

File '/Atlas-Blinking-LED-Baremetal-GNU/Atlas-Blinking-LED-Baremetal-Debug.launch' is read-only. Do you wish to make it writable?

Figure 5-27. Confirmation popup

____ 9. Press the [***Browse***] button to the right of the connection section and select the USB-Blaster™ connection.

____ 10. In the connection browser window, highlight the desired USB-Blaster™ (DE-SoC on localhost in this example) and click Select. Click [***Select***].



Connection Browser

Connection Browser

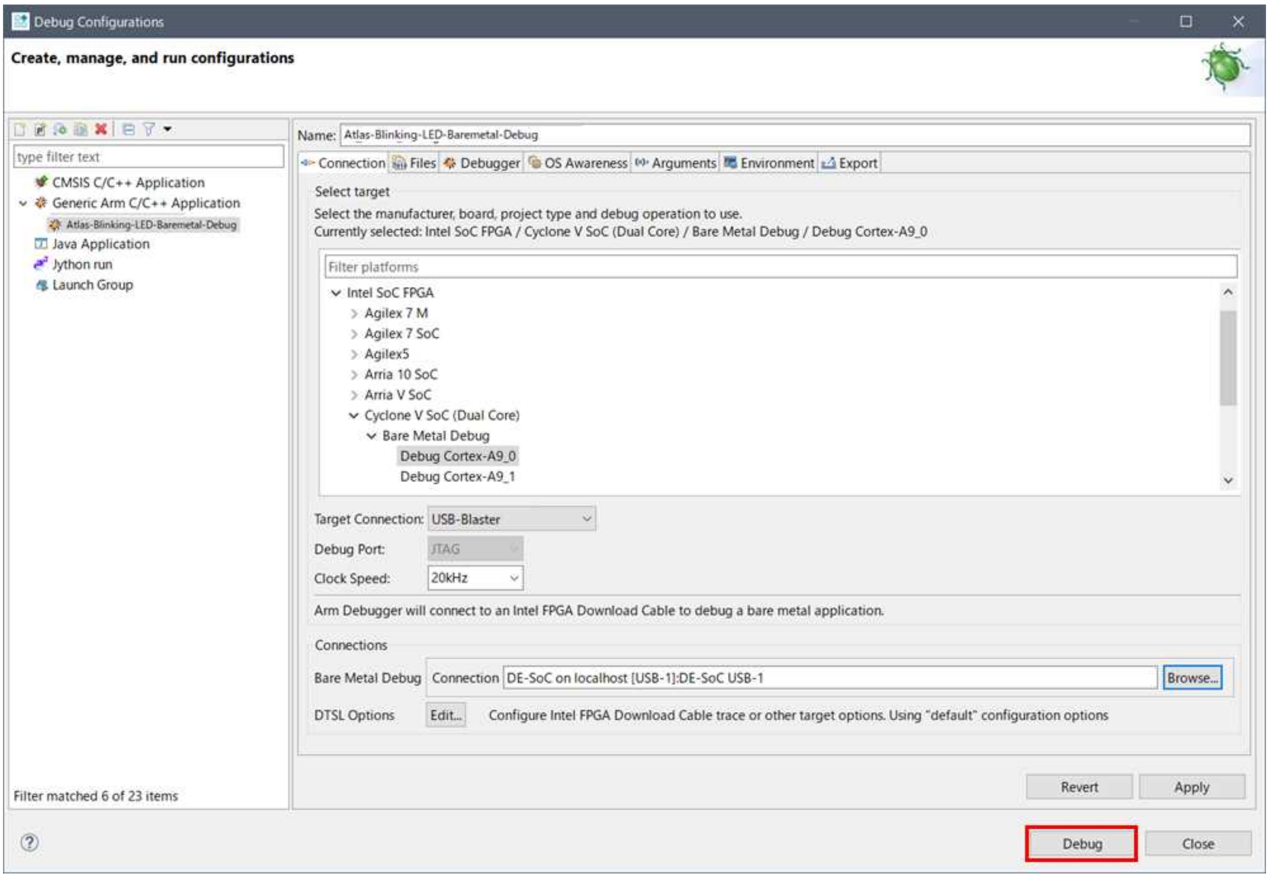
Select a target connection

DE-SoC USB-1

DE-SoC on localhost [USB-1]

Figure 5-28. Selecting a Debug Cable

____ 11. Click the [***Debug***] button in the lower right corner of the ***Debug Configurations*** window.



Debug Configurations

Create, manage, and run configurations

type filter text

CMSIS C/C++ Application
Generic Arm C/C++ Application
Atlas-Blinking-LED-Baremetal-Debug
Java Application
Jython run
Launch Group

Filter matched 6 of 23 items

Name: Atlas-Blinking-LED-Baremetal-Debug

Select target

Select the manufacturer, board, project type and debug operation to use.
Currently selected: Intel SoC FPGA / Cyclone V SoC (Dual Core) / Bare Metal Debug / Debug Cortex-A9_0

Filter platforms

Intel SoC FPGA
Agilex 7 M
Agilex 7 SoC
Agilex5
Arria 10 SoC
Arria V SoC
Cyclone V SoC (Dual Core)
Bare Metal Debug
Debug Cortex-A9_0
Debug Cortex-A9_1

Target Connection: USB-Blaster
Debug Port: JTAG
Clock Speed: 20kHz

Arm Debugger will connect to an Intel FPGA Download Cable to debug a bare metal application.

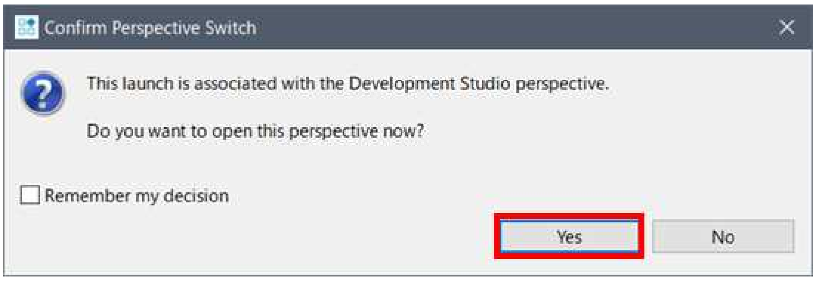
Connections

Bare Metal Debug Connection DE-SoC on localhost [USB-1]:DE-SoC USB-1 Browse...

DTSL Options Edit... Configure Intel FPGA Download Cable trace or other target options. Using "default" configuration options

Figure 5-29. Debugging the LED Blink sample application

____ 12. If prompted to confirm the perspective switch, click [***Yes***] to accept it.



Confirm Perspective Switch

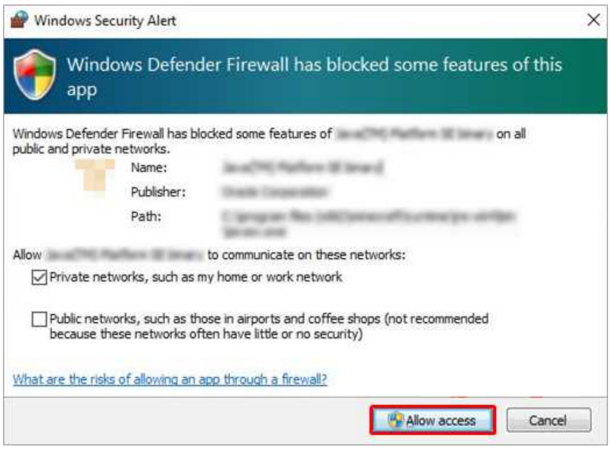
This launch is associated with the Development Studio perspective.

Do you want to open this perspective now?

□ Remember my decision

Figure 5-30. Checking the perspective switch

If you receive a Windows Defender Firewall warning, click [***Allow access***].



Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of [redacted] on all public and private networks.

Name: [redacted]

Publisher: [redacted]

Path: [redacted]

Allow [redacted] to communicate on these networks:

☑ Private networks, such as my home or work network

□ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

Figure 5-31. Security warning

ⓘ **Info:**

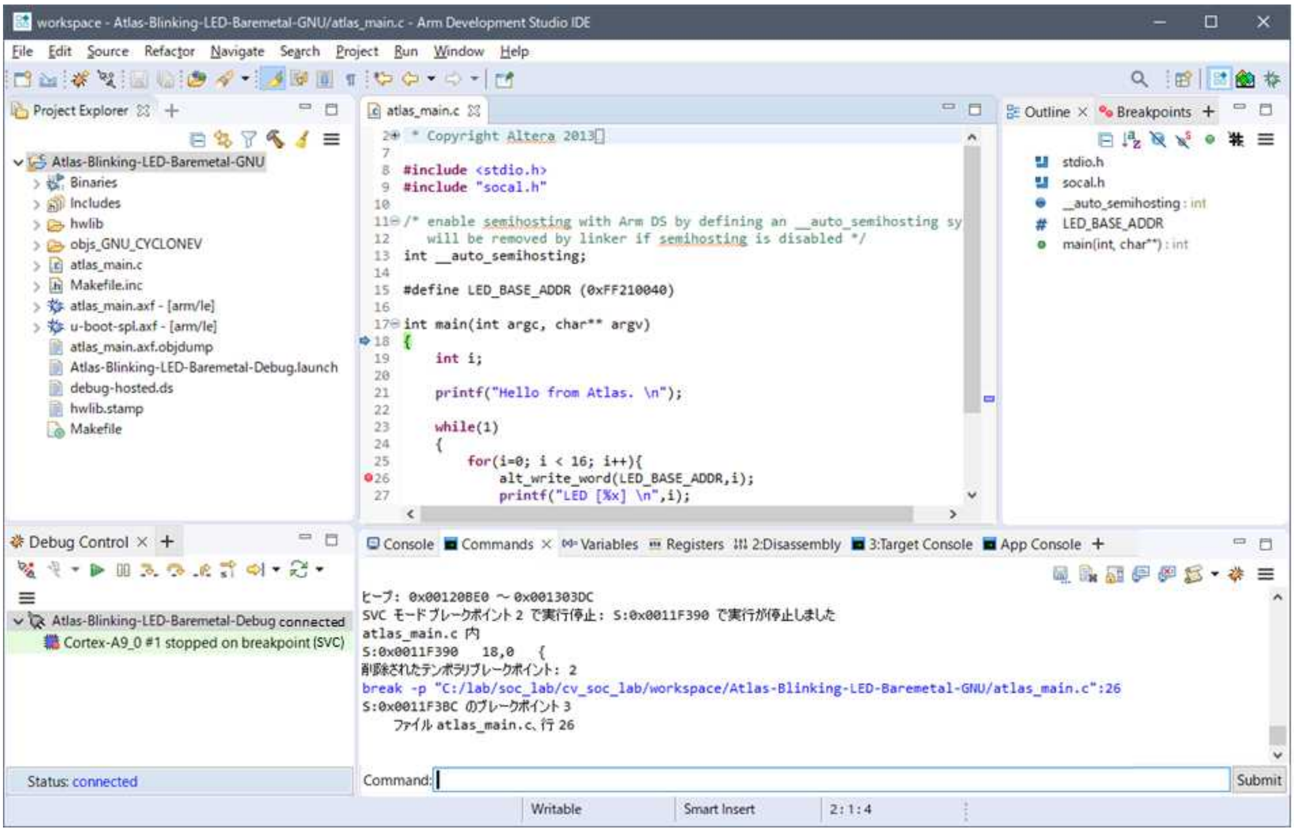
If you receive a download error, check the following:

(1) Make sure that the network interface (for example, USB-Ethernet Interface Adapter) to which the Arm® DS is licensed is enabled.

(2) Check if turning the evaluation board off and restarting the PC will not recover. If you turn off the evaluation board, remember to download the FPGA data again.

____ 13. Set a breakpoint.

Set a breakpoint on line 26 of **atlas_main.c**. You can set it by double-clicking the space to the left of the line number display.



```
 2⊕ * Copyright Altera 2013
 7
 8 #include <stdio.h>
 9 #include "socal.h"
10
11⊖/* enable semihosting with Arm DS by defining an __auto_semihosting sy
12    will be removed by linker if semihosting is disabled */
13 int __auto_semihosting;
14
15 #define LED_BASE_ADDR (0xFF210040)
16
17⊖int main(int argc, char** argv)
18 {
19     int i;
20
21     printf("Hello from Atlas. \n");
22
23     while(1)
24     {
25         for(i=0; i < 16; i++){
26             alt_write_word(LED_BASE_ADDR,i);
27             printf("LED [%x] \n",i);
```

```
ヒープ: 0x00120BE0 ~ 0x001303DC
SVC モードブレークポイント 2 で実行停止: S:0x0011F390 で実行が停止しました
atlas_main.c 内
S:0x0011F390   18,0   {
削除されたテンポラリブレークポイント: 2
break -p "C:/lab/soc_lab/cv_soc_lab/workspace/Atlas-Blinking-LED-Baremetal-GNU/atlas_main.c":26
S:0x0011F3BC のブレークポイント 3
    ファイル atlas_main.c、行 26
```

Figure 5-32. Setting breakpoints

____ 14. green ***Continue*** button ▶ to run the application (or press F8). This displays a **Hello from Atlas** message in the App Console. A message is displayed.

____ 15. Click the green ***Continue*** button ▶ twice again (or press F8) to run the application. This displays an **LED [0]** message in the **application console** and you can see that the user LED (LED [3: 0]) on the Atlas-SoC board changes state.

____ 16. Furthermore, confirm that the LED state changes each time you click the ***Continue*** button.

____ 17. ***Disconnect*** button to disconnect from the CPU.

This completes Lab 3. Thank you!

There are optional exercises from the next page. If you have time, try them too.

5-4. Initialize with Preloader created in Lab 2 (optional exercise)

In Lab 3, you initialized HPS using the preloaded Preloader.

Now you will initialize HPS with the Preloader created in "4. Lab 2 - Software Exercise (1) Generate Preloader".

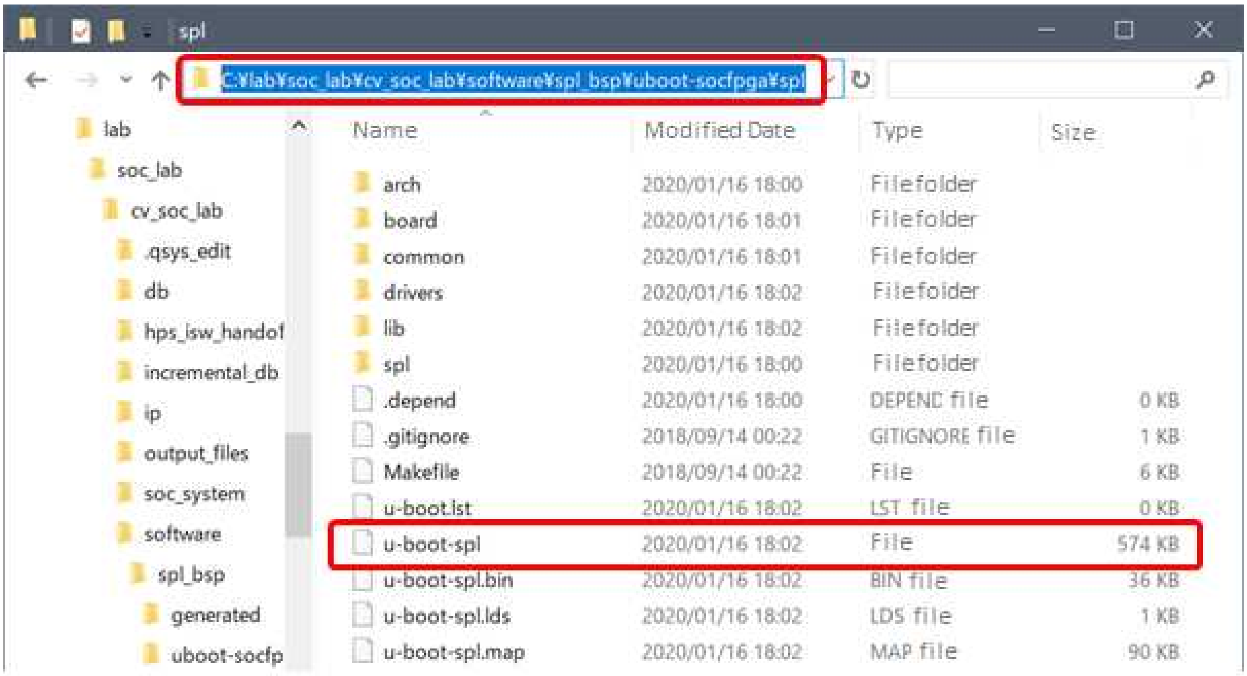
____ 1. Verify that the Preloader image was created in Lab 2.

The Preloader should be created under the

`C:¥lab¥soc_lab¥cv_soc_lab¥software¥spl_bsp¥uboot-socfpga¥spl`

directory and named u-boot-spl. Verify that this file has been generated. Also, make sure that a device tree "u-boot-spl.dtb" has been created in the same directory as the Preloader itself.

If not, repeat Lab 2.



Name	Modified Date	Type	Size
arch	2020/01/16 18:00	Filefolder	
board	2020/01/16 18:01	Filefolder	
common	2020/01/16 18:01	Filefolder	
drivers	2020/01/16 18:02	Filefolder	
lib	2020/01/16 18:02	Filefolder	
spl	2020/01/16 18:00	Filefolder	
.depend	2020/01/16 18:00	DEPEND file	0 KB
.gitignore	2018/09/14 00:22	GITIGNORE file	1 KB
Makefile	2018/09/14 00:22	File	6 KB
u-boot.lst	2020/01/16 18:02	LST file	0 KB
u-boot-spl	2020/01/16 18:02	File	574 KB
u-boot-spl.bin	2020/01/16 18:02	BIN file	36 KB
u-boot-spl.lds	2020/01/16 18:02	LDS file	1 KB
u-boot-spl.map	2020/01/16 18:02	MAP file	90 KB

Figure 5-33. Checking the U-boot-spl File

ⓘ **Info:**

The u-boot-spl file you just checked is an Arm® Executable and Linkable Format (ELF) file.

It is read by the Arm® DS initialization script and executed before running the user application.

For details, please refer to the chapter "How to support custom boards" in the reference article below.

(The reference article is for DS-5™, but the concept is similar for Arm® DS.)

Reference: Getting Started with SoC – Bare Metal Application Debugging with DS-5 (in Japanese)

____ 2. Rename the “u-boot-spl.axf” file used in Lab 3.

C:¥lab¥soc_lab¥cv_soc_lab¥workspace¥Atlas-Blinking-LED-Baremetal-GNU contains the file “u-boot-spl.axf”.

____ 3. Copy the files “u-boot-spl.dtb” and "u-boot-spl" created in Lab 2.

Copy the files “u-boot-spl.dtb” and "u-boot-spl" under the directory
`C:¥lab¥soc_lab¥cv_soc_lab¥software¥spl_bsp¥uboot-socfpga¥spl`
to the directory
`C:¥lab¥soc_lab¥cv_soc_lab¥workspace¥Atlas-Blinking-LED-Baremetal-GNU.`

____ 4. Rename the copied “u-boot-spl.axf” file.

You have now modified the Preloader you will use when debugging.
Let's see if it works.

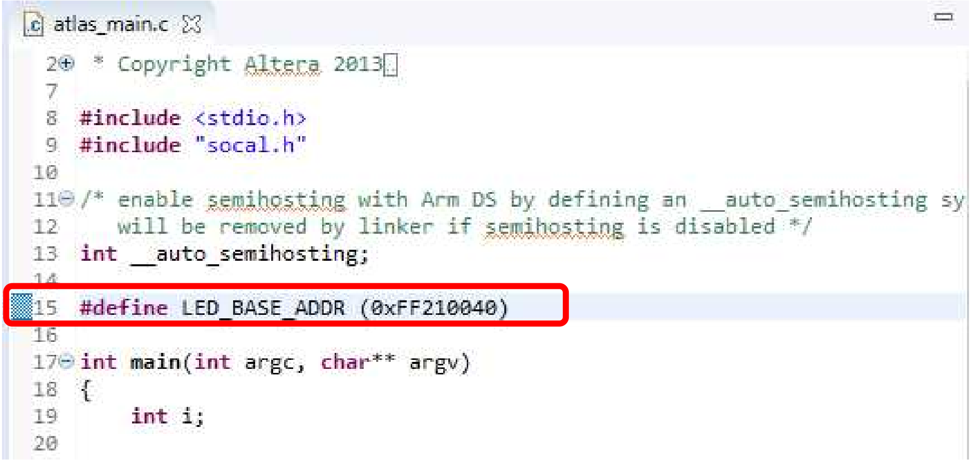
____ 5. Run the LED Blink sample application again.

Run it from “____6 Choose *Run menu* => Debug Configuration (*B*). The sample project comes with pre-configuration to run on an Atlas-SoC board..” on page 75.

If the LED Blink sample application runs as before, the Preloader created in Lab 2 is working correctly.

5-5. Address resolution using system header files (optional exercise)

The LED Blink sample application in Lab 3 addressed the LED PIO directly in the source code.



```
 * Copyright Altera 2013

#include <stdio.h>
#include "socal.h"

/* enable semihosting with Arm DS by defining an __auto_semihosting sy
   will be removed by linker if semihosting is disabled */
int __auto_semihosting;

#define LED_BASE_ADDR (0xFF210040)

int main(int argc, char** argv)
{
    int i;

```

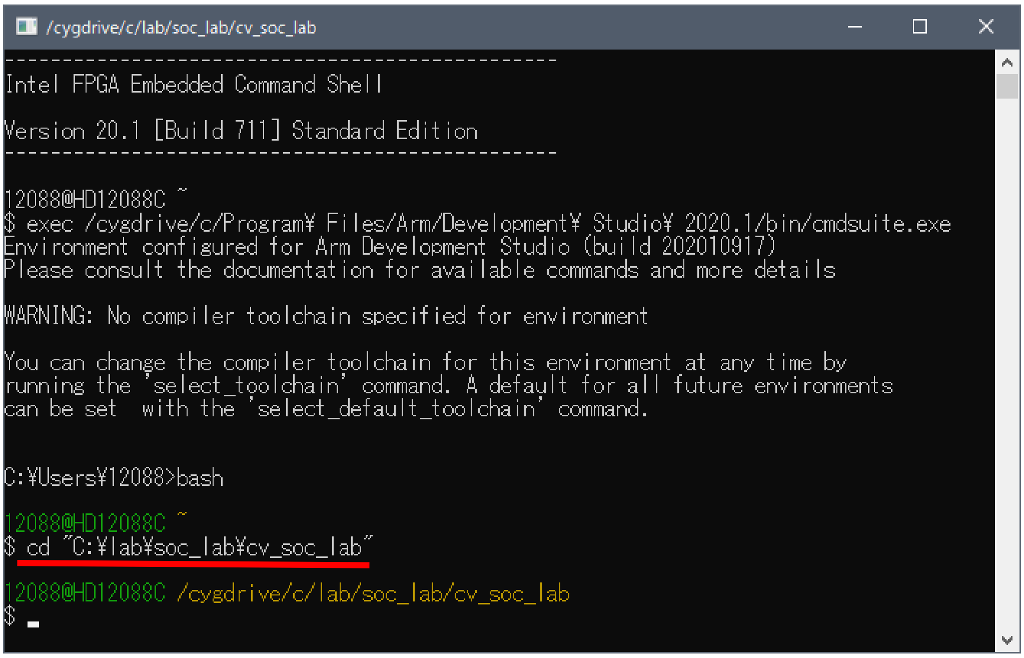
Figure 5-34. How to address so far

Let's use the SoC EDS system header file generation command (sopc-create-header-files) to generate and use the system header file.

____ 1. Start the Embedded Command Shell if it is not already running.

____ 2. Navigate to C:¥lab¥soc_lab¥cv_soc_lab.

```
$ cd "C:¥lab¥soc_lab¥cv_soc_lab" ↵
```

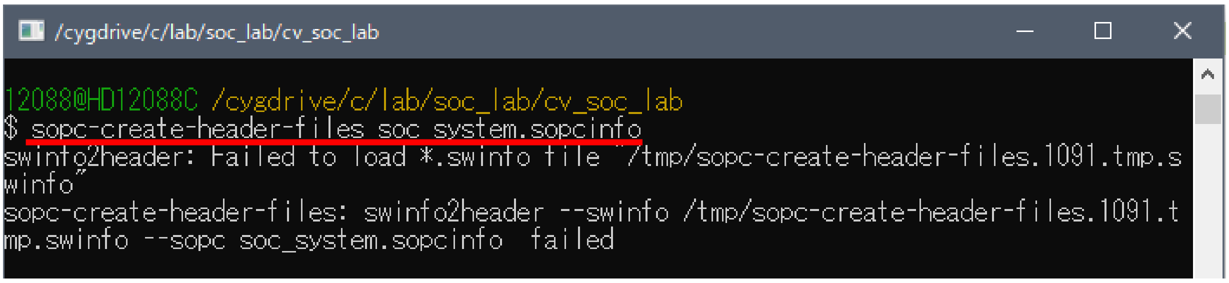


```
------------------------------------------------
Intel FPGA Embedded Command Shell

Version 20.1 [Build 711] Standard Edition
------------------------------------------------

12088@HD12088C ~
$ exec /cygdrive/c/Program¥ Files/Arm/Development¥ Studio¥ 2020.1/bin/cmdsuite.exe
Environment configured for Arm Development Studio (build 202010917)
Please consult the documentation for available commands and more details

WARNING: No compiler toolchain specified for environment

You can change the compiler toolchain for this environment at any time by
running the 'select_toolchain' command. A default for all future environments
can be set  with the 'select_default_toolchain' command.


C:¥Users¥12088>bash

12088@HD12088C ~
$ cd "C:¥lab¥soc_lab¥cv_soc_lab"

12088@HD12088C /cygdrive/c/lab/soc_lab/cv_soc_lab
$ 
```

Figure 5-35. Moving directories

____ 3. In the Embedded Command Shell, run the system header file generation command (sopc-create-header-files).

```
$ sopc-create-header-files soc_system.sopcinfo ↵
```



```
12088@HD12088C /cygdrive/c/lab/soc_lab/cv_soc_lab
$ sopc-create-header-files soc_system.sopcinfo
swinfo2header: Failed to load *.swinfo file "/tmp/sopc-create-header-files.1091.tmp.s
winfo"
sopc-create-header-files: swinfo2header --swinfo /tmp/sopc-create-header-files.1091.t
mp.swinfo --sopc soc_system.sopcinfo  failed
```

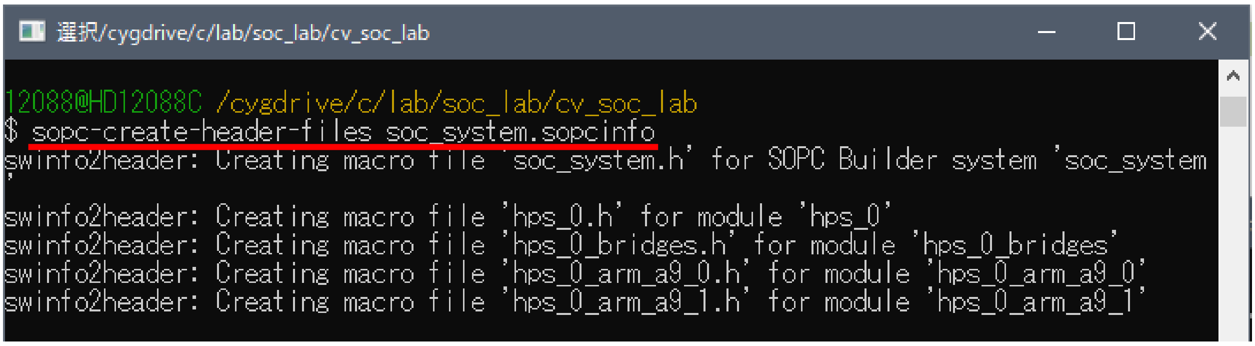
Figure 5-36. Executing system header file generation command (error)

⚠ **Note:**

If your operating system is Windows® 10, the error shown above may occur. If the error occurs, take the corrective action described in the following reference site, and then re-execute sopc-create-header-files.

Reference:

Macnica Altera FPGA Insights "Workaround for sopc-create-header-files execution error in SoC EDS environment"



```
12088@HD12088C /cygdrive/c/lab/soc_lab/cv_soc_lab
$ sopc-create-header-files soc_system.sopcinfo
swinfo2header: Creating macro file 'soc_system.h' for SOPC Builder system 'soc_system
'

swinfo2header: Creating macro file 'hps_0.h' for module 'hps_0'
swinfo2header: Creating macro file 'hps_0_bridges.h' for module 'hps_0_bridges'
swinfo2header: Creating macro file 'hps_0_arm_a9_0.h' for module 'hps_0_arm_a9_0'
swinfo2header: Creating macro file 'hps_0_arm_a9_1.h' for module 'hps_0_arm_a9_1'
```

Figure 5-37. Executing the system header file generation command (success)

Verify that 5 files have been generated.

soc_system.h : Define module information for all masters in Platform Designer

hps_0.h : Define module information connected to each bridge (H2F, LWH2F) in HPS

hps_0_bridges.h : Define module information connected to each bridge (F2H, H2F, LWH2F) in HPS

hps_0_arm_a9_0.h : Define module information for hps_0_arm_a9_0. Offsets for each bridge are added.

hps_0_arm_a9_1.h : Define module information for hps_0_arm_a9_1. Offsets for each bridge are added.

We will use hps_0_arm_a9_0.h.

____ 4. Copy the system header file to the LED Blink sample application project.

File name : **`hps_0_arm_a9_0.h`**
From : **`C:¥lab¥soc_lab¥cv_soc_lab`**
To : **`C:¥lab¥soc_lab¥cv_soc_lab¥workspace¥Atlas-Blinking-LED-Baremetal-GNU`**

____ 5. Modify the LED Blink sample application source code **atlas_main.c**.

Select Yes if you see a "Do you want to make it writable?" pop-up when you make the change.

Description added:

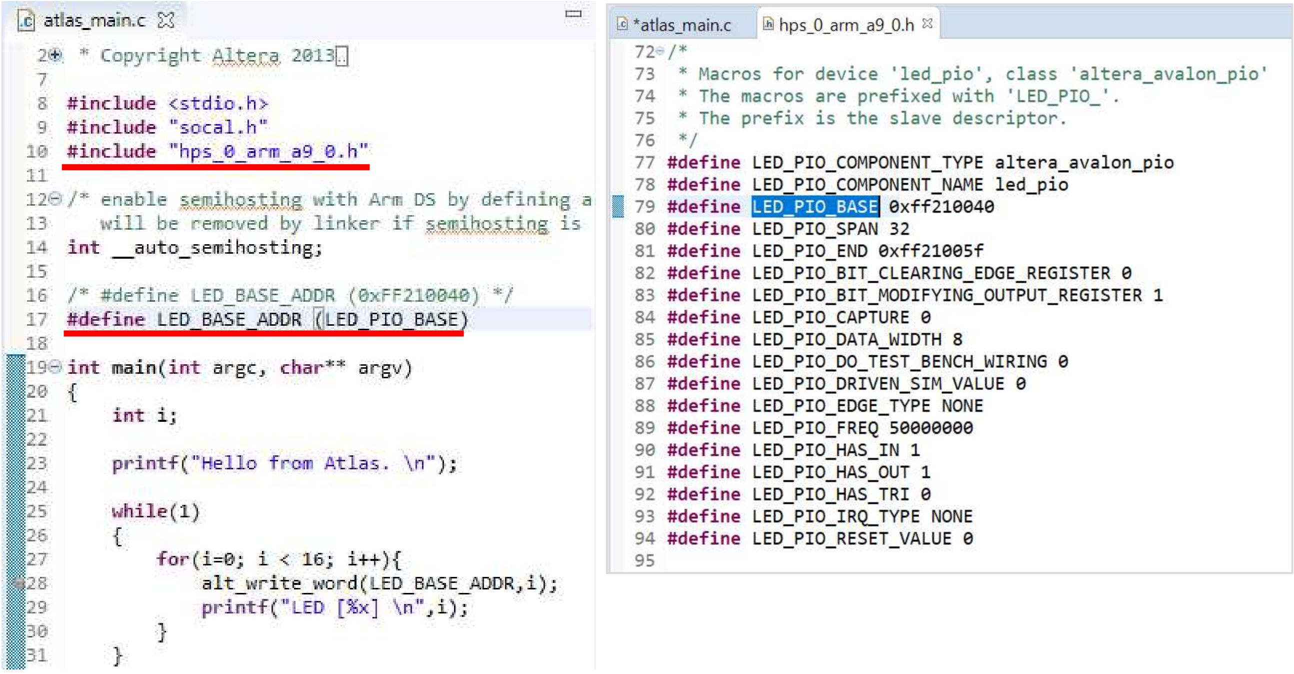
```
#include “hps_0_arm_a9_0.h”
```

Description changed:

< before change > `#define LED_BASE_ADDR (0xFF210040)`

< after change > `#define LED_BASE_ADDR LED_PIO_BASE`

In the following figure, the previous LED_BASE_ADDR description has been commented out for ease of comparison. For reference, the corresponding parts of "hps _0_arm_a9_0 .h" are also shown.



```
 * Copyright Altera 2013

#include <stdio.h>
#include "socal.h"
#include "hps_0_arm_a9_0.h"

/* enable semihosting with Arm DS by defining a
   will be removed by linker if semihosting is
int __auto_semihosting;

/* #define LED_BASE_ADDR (0xFF210040) */
#define LED_BASE_ADDR (LED_PIO_BASE)

int main(int argc, char** argv)
{
    int i;

    printf("Hello from Atlas. \n");

    while(1)
    {
        for(i=0; i < 16; i++){
            alt_write_word(LED_BASE_ADDR,i);
            printf("LED [%x] \n",i);
        }
    }
```

```
/*
 * Macros for device 'led_pio', class 'altera_avalon_pio'
 * The macros are prefixed with 'LED_PIO_'.
 * The prefix is the slave descriptor.
 */
#define LED_PIO_COMPONENT_TYPE altera_avalon_pio
#define LED_PIO_COMPONENT_NAME led_pio
#define LED_PIO_BASE 0xff210040
#define LED_PIO_SPAN 32
#define LED_PIO_END 0xff21005f
#define LED_PIO_BIT_CLEARING_EDGE_REGISTER 0
#define LED_PIO_BIT_MODIFYING_OUTPUT_REGISTER 1
#define LED_PIO_CAPTURE 0
#define LED_PIO_DATA_WIDTH 8
#define LED_PIO_DO_TEST_BENCH_WIRING 0
#define LED_PIO_DRIVEN_SIM_VALUE 0
#define LED_PIO_EDGE_TYPE NONE
#define LED_PIO_FREQ 50000000
#define LED_PIO_HAS_IN 1
#define LED_PIO_HAS_OUT 1
#define LED_PIO_HAS_TRI 0
#define LED_PIO_IRQ_TYPE NONE
#define LED_PIO_RESET_VALUE 0
```

Figure 5-38. Source code changes and system header files

____ 6. Save the modified **atlas_main.c** and build the LED Blink sample application.

____ 7. After the build, run the LED Blink sample application and verify that the results are similar to those in Lab 3.

This completes Lab 3 (optional).

6. Lab 4: Linux Application Exercise (optional exercise)

In this exercise you will run and debug Hello World, one of the Linux applications, from Arm® DS.

ⓘ **Info:**

For this exercise, if you have a SoC FPGA Seminar ina Box provided by our company, you will use the included microSD card.
This microSD card contains a design to boot the Linux OS.
If you are not using a SoC FPGA Seminar ina Box to perform this exercise, please prepare your own microSD card by following the instructions in "6-1. Preparing the microSD card" below.

6-1. Preparing the microSD card

If you are using the microSD card supplied with the “SoC FPGA Seminar in a Box”, skip this section and use the "**6-2. Linux Boot and Login**" below. If you want to burn the microSD card yourself, follow the steps below.

____ 1. Download the SD card image file for your board from the following site.
Extract the downloaded file to a folder of your choice. Make sure that the .img image file exists in the extracted folder.

- Atlas-SoC board SD card image file
- DE10 Nano board SD card image file

____ 2. Windows® users use general-purpose software to write SD card image files. This section introduces Win32 Disk Imager. You can download it from:

- Win32 Disk Imager

____ 3. Insert the microSD card (8GB or more recommended) into the SD card slot of the PC (or use a USB card reader/writer). Check the drive (In this example, drive E) assigned to the microSD card.

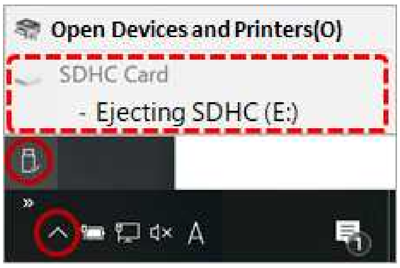


Figure 6-1. Check the drive assigned to the microSD card

____ 4. Launch the Win32 Disk Imager that you previously installed on your PC.

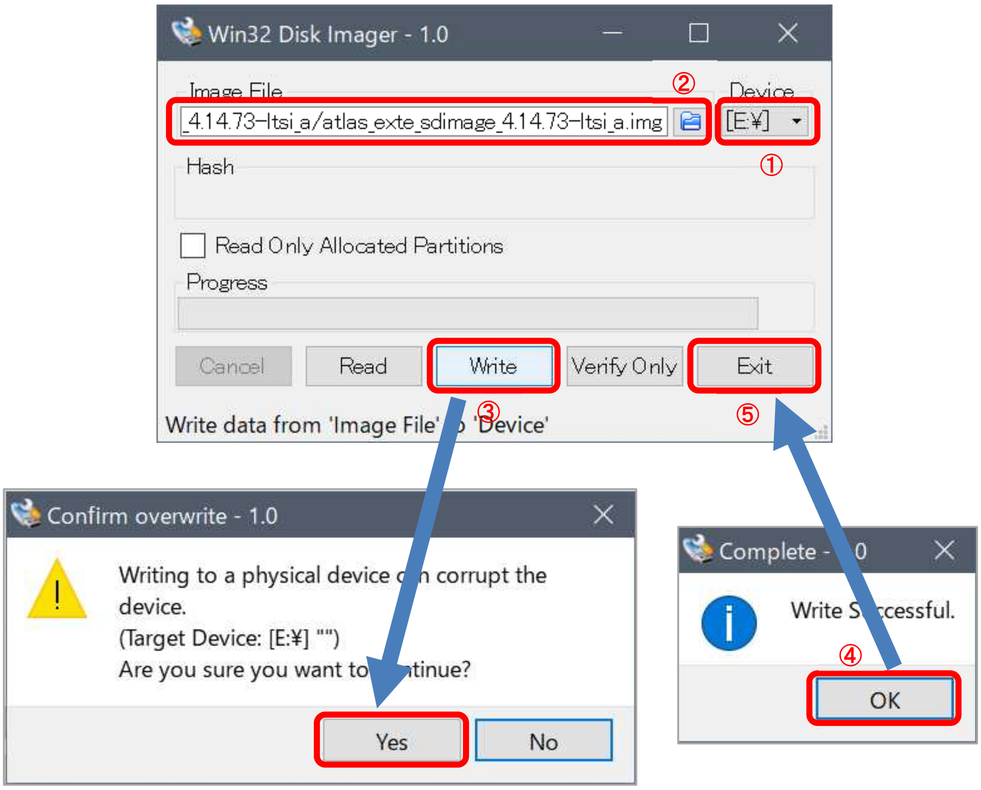
① Make sure the drive of the microSD card that you inserted into your PC is selected as the Device.

② Select and open the SD card image file that you extracted earlier.

③ Click the [***Write***] button to write the image file.

④ Click the [***OK***] button when writing is complete.

⑤ Click the [***Exit***] button to exit Win32DiskImager.



4.14.73-ltsi_a/atlas_exte_sdimage_4.14.73-ltsi_a.img

[E:¥]

□ Read Only Allocated Partitions

Write data from 'Image File' to 'Device'

Writing to a physical device can corrupt the device.
(Target Device: [E:¥] "")
Are you sure you want to continue?

Write Successful.

Figure 6-2. Win32 Disk Imager

____ 5. Safely remove the microSD card from the PC.

⚠ **Note:**

If the host PC is running Windows® 10 and you are writing an SD card and there is a non-FAT partition (volume) in the card, the following symptoms may occur:

- Warning window appears when inserting the card
- SD card image fails to write

For information on how to handle these symptoms, see the following reference sites:

📄 **Reference:**

Macnica Altera FPGA Insights "What to do if writing an SD card image fails in Windows® 10"

6-2. Linux Boot and Login

This exercise uses the following interfaces.

The following is an example of an Atlas-SoC board, but the DE10 Nano board is essentially the same.

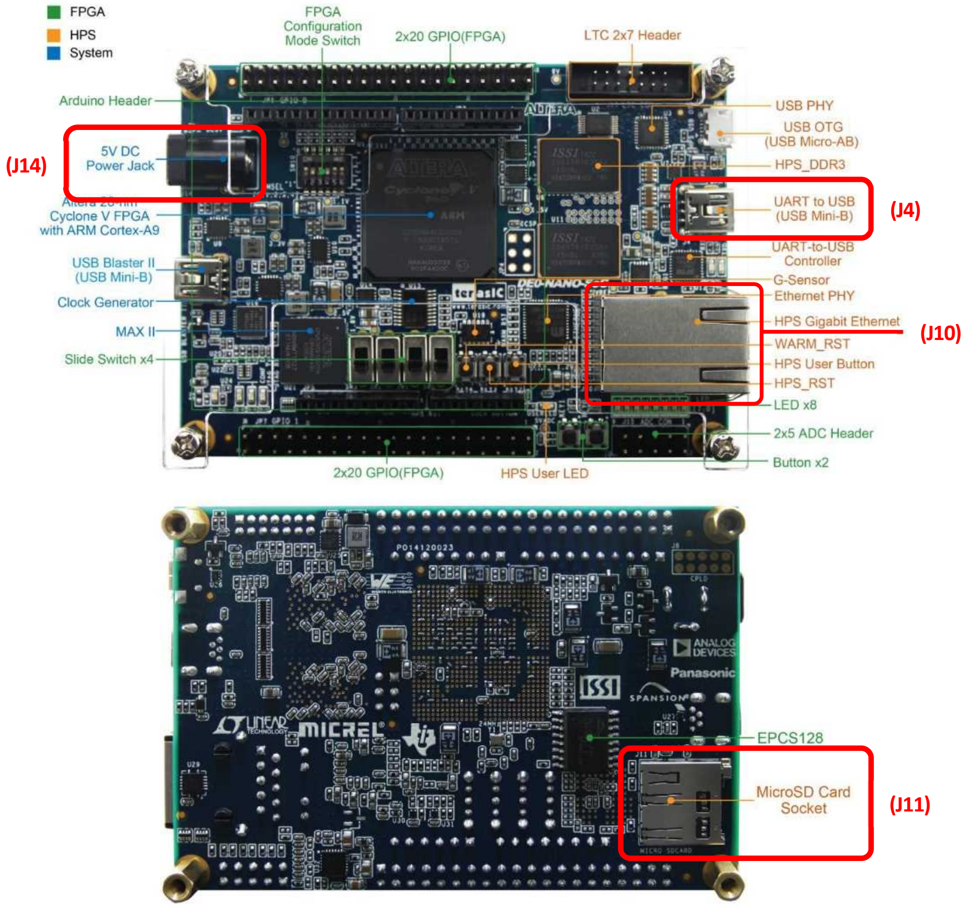


Figure 6-3. Interface used in this exercise (Atlas-SoC board)

____ 1. If the power adapter is connected to the 5 V DC jack (J14) on the board, unplug it.

____ 2. Connect the USB Mini-B cable to the UART USB connector (J4) on the board. Connect the other end of the cable to the USB connector on the PC.

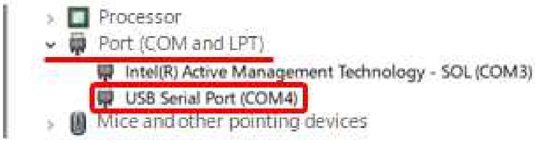
____ 3. Connect the Ethernet cable to the HPS Ethernet connector (J10) on the board. Connect the other end of the cable to the Ethernet connector on the PC.

____ 4. Insert the microSD card into the microSD card slot (J11) on the back of the board.

____ 5. Connect the power adapter cable to the 5 V DC jack (J14) on the board and power on the board.

____ 6. Open Windows® ***Device Manager***. Expand ***Ports* (COM & LPT)** in ***Device Manager*** and check the COM port number of the UART on the board (COM 4 in this example).

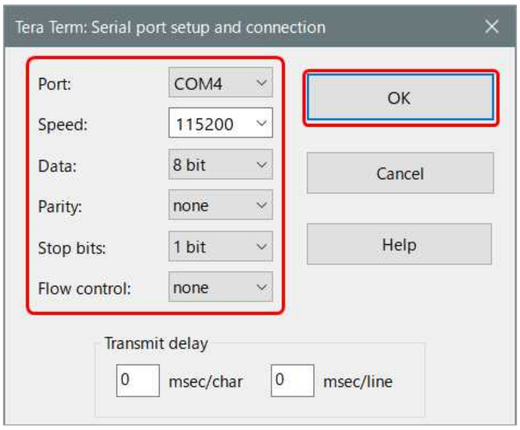
Close ***Device Manager***.



Processor
Port (COM and LPT)
Intel(R) Active Management Technology - SOL (COM3)
USB Serial Port (COM4)
Mice and other pointing devices

Figure 6-4. Checking the COM Port

____ 7. Start the terminal software that you have previously installed, and set the serial port. Select the COM port that you just checked, and set it as shown below (COM 4 in this example).



Tera Term: Serial port setup and connection

Port: COM4
Speed: 115200
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Figure 6-5. Serial Port Settings

____ 8. Press the WARM reset button (KEY3) on the board. The terminal displays a boot message.

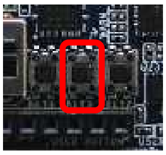
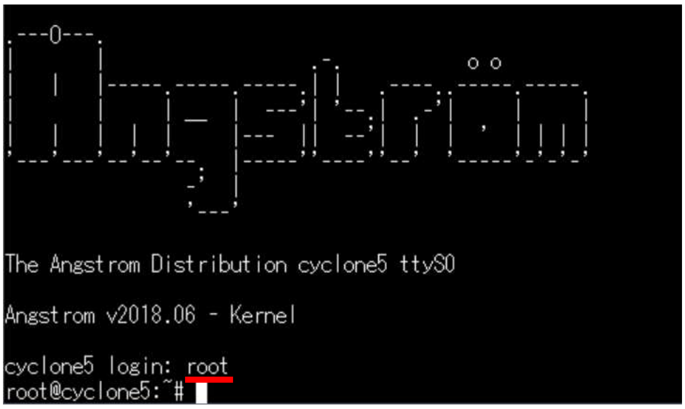


Figure 6-6. WARM reset button (KEY3)

____ 9. After the Linux kernel boots, log in as **root** ↵



```
.---O---.
|       |                  .-.           o o
|   |   |-----.-----.-----.| |   .----..-----.-----.
|       |     | __  |  ---'| '--.|  .-'|     |     |
|   |   |  |  |     |---  ||  --'|  |  |  '  | | | |
'---'---'--'--'--.  |-----''----''--'  '-----'-'-'-'
                -'  |
                '---'

The Angstrom Distribution cyclone5 ttyS0

Angstrom v2018.06 - Kernel

cyclone5 login: root
root@cyclone5:~#
```

Figure 6-7. Login as Root

6-3. Setting the IP Address and Password on Linux

____ 1. From a terminal, use the `ifconfig` command to set the IP address of the board (192.168.1.30 in this example).

```
# ifconfig eth0 192.168.1.30 ↵
```

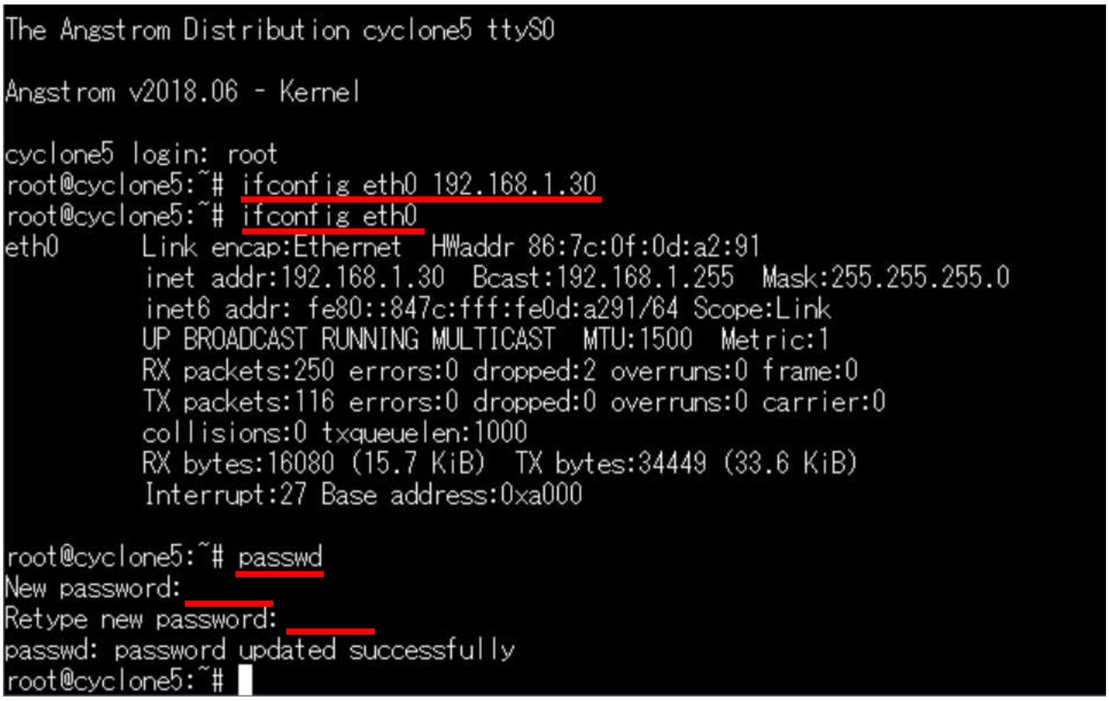
____ 2. Then use the ifconfig command to verify the settings.

```
# ifconfig eth0 ↵
```

____ 3. Use the Passwd command to set the password of your choice. This password will be used later for remote system debugging.

```
# passwd ↵
```

____ 4. Enter the password again.



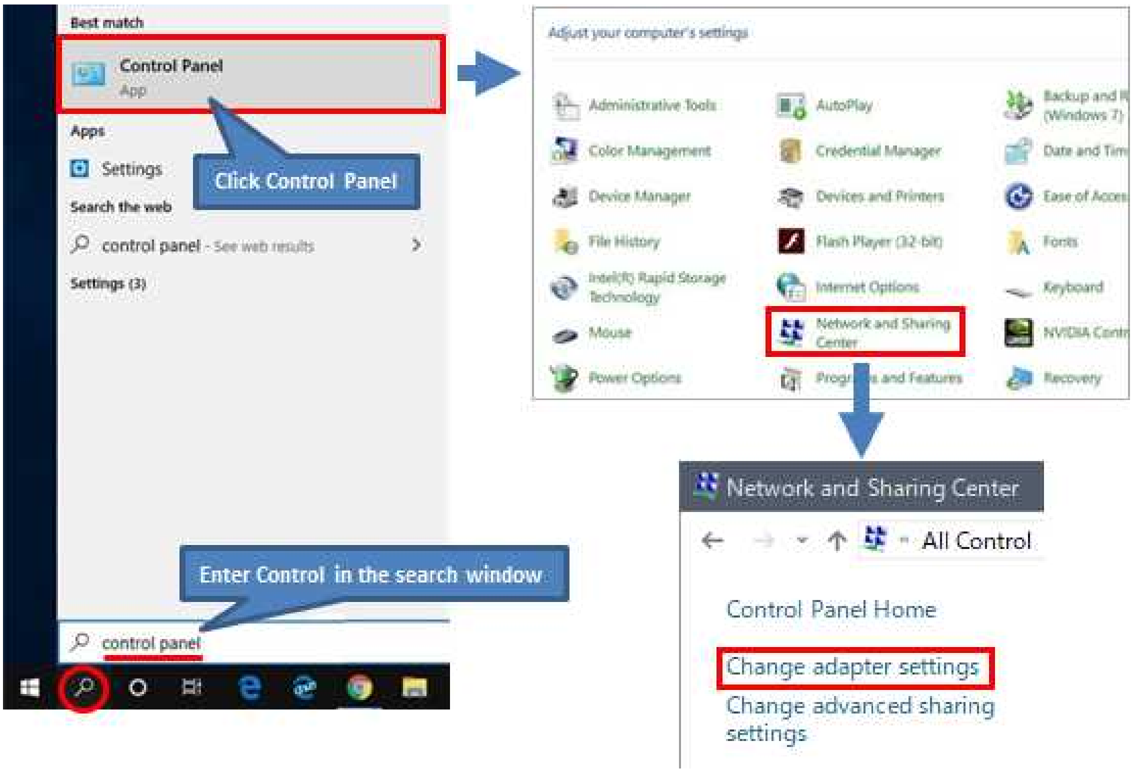
```
The Angstrom Distribution cyclone5 ttyS0

Angstrom v2018.06 - Kernel

cyclone5 login: root
root@cyclone5:~# ifconfig eth0 192.168.1.30
root@cyclone5:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 86:7c:0f:0d:a2:91
          inet addr:192.168.1.30  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::847c:fff:fe0d:a291/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:250 errors:0 dropped:2 overruns:0 frame:0
          TX packets:116 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16080 (15.7 KiB)  TX bytes:34449 (33.6 KiB)
          Interrupt:27 Base address:0xa000

root@cyclone5:~# passwd
New password:
Retype new password:
passwd: password updated successfully
root@cyclone5:~#
```

Figure 6-8. Setting the IP Address and Password

6-4. Network Settings on the Host PC

To run and debug Linux applications using Remote System Explorer (RSE) on the Arm® DS, set the network settings on the host PC.

____ 1. First, set the IP address of the host PC. From the ***Control Panel***, click ***Network and Sharing Center***, then click ***Change Adapter Settings*** on the left.



Click Control Panel

Enter Control in the search window

Figure 6-9. Changing Adapter Settings

____ 2. Double-click Ethernet.

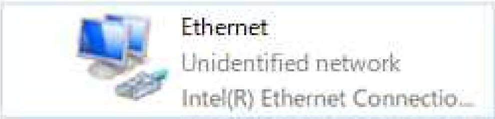
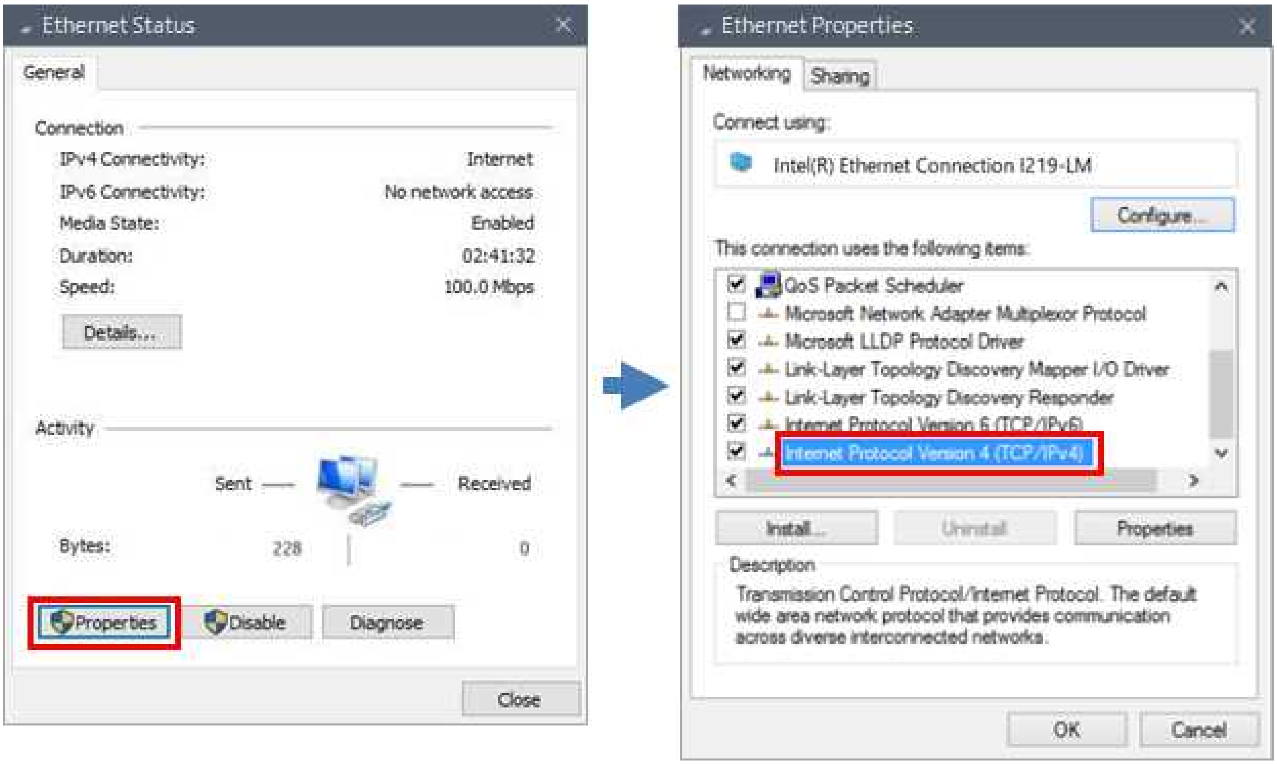


Figure 6-10. Double-click Ethernet

____ 3. Click the [***Properties***] button.

____ 4. Double-click ***Internet Protocol Version 4*** **(TCP/Ipv4)**.



Ethernet Status

General

Connection

IPv4 Connectivity: Internet

IPv6 Connectivity: No network access

Media State: Enabled

Duration: 02:41:32

Speed: 100.0 Mbps

Details...

Activity

Sent — — Received

Bytes: 228 | 0

Properties Disable Diagnose

Close

Ethernet Properties

Networking Sharing

Connect using:

Intel(R) Ethernet Connection I219-LM

Configure...

This connection uses the following items:

- ☑ QoS Packet Scheduler
- □ Microsoft Network Adapter Multiplexor Protocol
- ☑ Microsoft LLDP Protocol Driver
- ☑ Link-Layer Topology Discovery Mapper I/O Driver
- ☑ Link-Layer Topology Discovery Responder
- ☑ Internet Protocol Version 6 (TCP/IPv6)
- ☑ Internet Protocol Version 4 (TCP/IPv4)

Install... Uninstall Properties

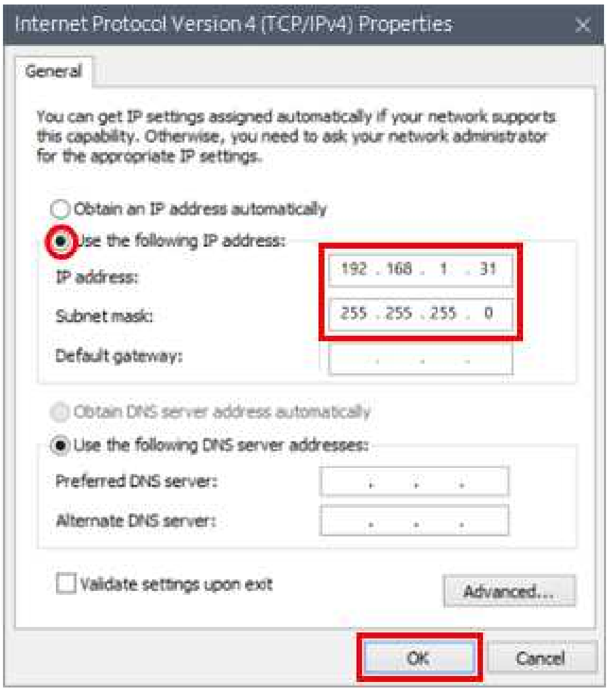
Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

OK Cancel

Figure 6-11. Local Area Connection Properties

____ 5. Select the "***Use the following IP address:***" checkbox and specify the "***IP address***" and "***Subnet mask***" (This example sets the IP address to 192.168.1.31 and the subnet mask to 255.255.255.0.).
Click [***OK***] when finished.



Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address: 192 . 168 . 1 . 31

Subnet mask: 255 . 255 . 255 . 0

Default gateway: . . .

○ Obtain DNS server address automatically

● Use the following DNS server addresses:

Preferred DNS server: . . .

Alternate DNS server: . . .

□ Validate settings upon exit

Advanced...

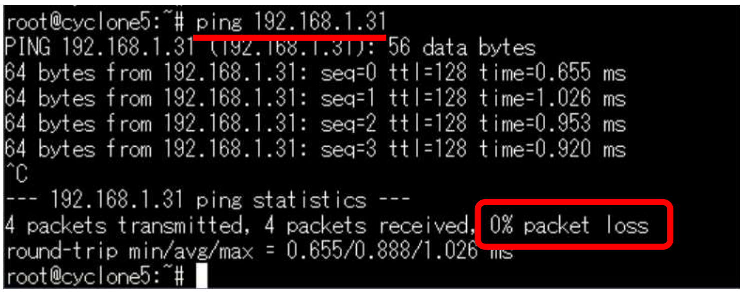
OK Cancel

Figure 6-12. Setting the "IP address" and "Subnet mask"

____ 6. Check network connectivity. Try pinging the host PC from Linux on the board (In this example, the IP address of the PC is set to 192.168.1.31.).

```
# ping 192.168.1.31 ↲
```

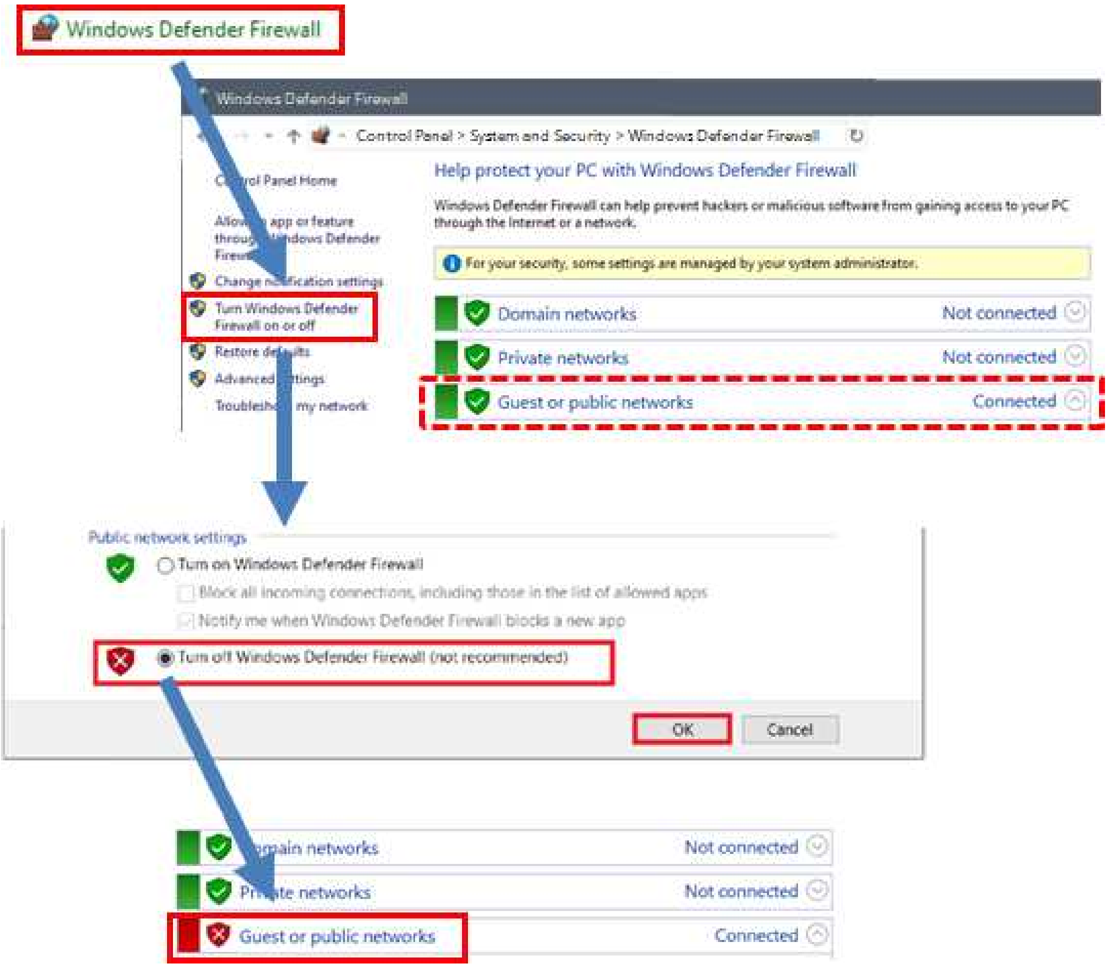
____ 7. Ctrl + C to stop the ping.



```
root@cyclone5:~# ping 192.168.1.31
PING 192.168.1.31 (192.168.1.31): 56 data bytes
64 bytes from 192.168.1.31: seq=0 ttl=128 time=0.655 ms
64 bytes from 192.168.1.31: seq=1 ttl=128 time=1.026 ms
64 bytes from 192.168.1.31: seq=2 ttl=128 time=0.953 ms
64 bytes from 192.168.1.31: seq=3 ttl=128 time=0.920 ms
^C
--- 192.168.1.31 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.655/0.888/1.026 ms
root@cyclone5:~#
```

Figure 6-13. Ping the PC to check connectivity

____ 8. If there is no ping response, check the Windows Defender Firewall settings.
Check the ***Public Network Settings*** and if Windows Defender Firewall is set to "**Enabled**," set it to "**Disabled**" and ping again to check connectivity.



Windows Defender Firewall

Control Panel > System and Security > Windows Defender Firewall

Help protect your PC with Windows Defender Firewall

Windows Defender Firewall can help prevent hackers or malicious software from gaining access to your PC through the Internet or a network.

For your security, some settings are managed by your system administrator.

Domain networks — Not connected
Private networks — Not connected
Guest or public networks — Connected

Control Panel Home
Allow an app or feature through Windows Defender Firewall
Change notification settings
Turn Windows Defender Firewall on or off
Restore defaults
Advanced settings
Troubleshoot my network

Public network settings

○ Turn on Windows Defender Firewall
□ Block all incoming connections, including those in the list of allowed apps
☑ Notify me when Windows Defender Firewall blocks a new app
● Turn off Windows Defender Firewall (not recommended)

Domain networks — Not connected
Private networks — Not connected
Guest or public networks — Connected

Figure 6-14. Windows Firewall settings

6-5. Start Arm® DS and import and build the Linux sample application

____ 1. Start the Embedded Command Shell by double-clicking the startup script ***Embedded_Command_Shell.bat*** from the Windows® Start menu or the SoC EDS installation folder (intelFPGA ¥embedded).

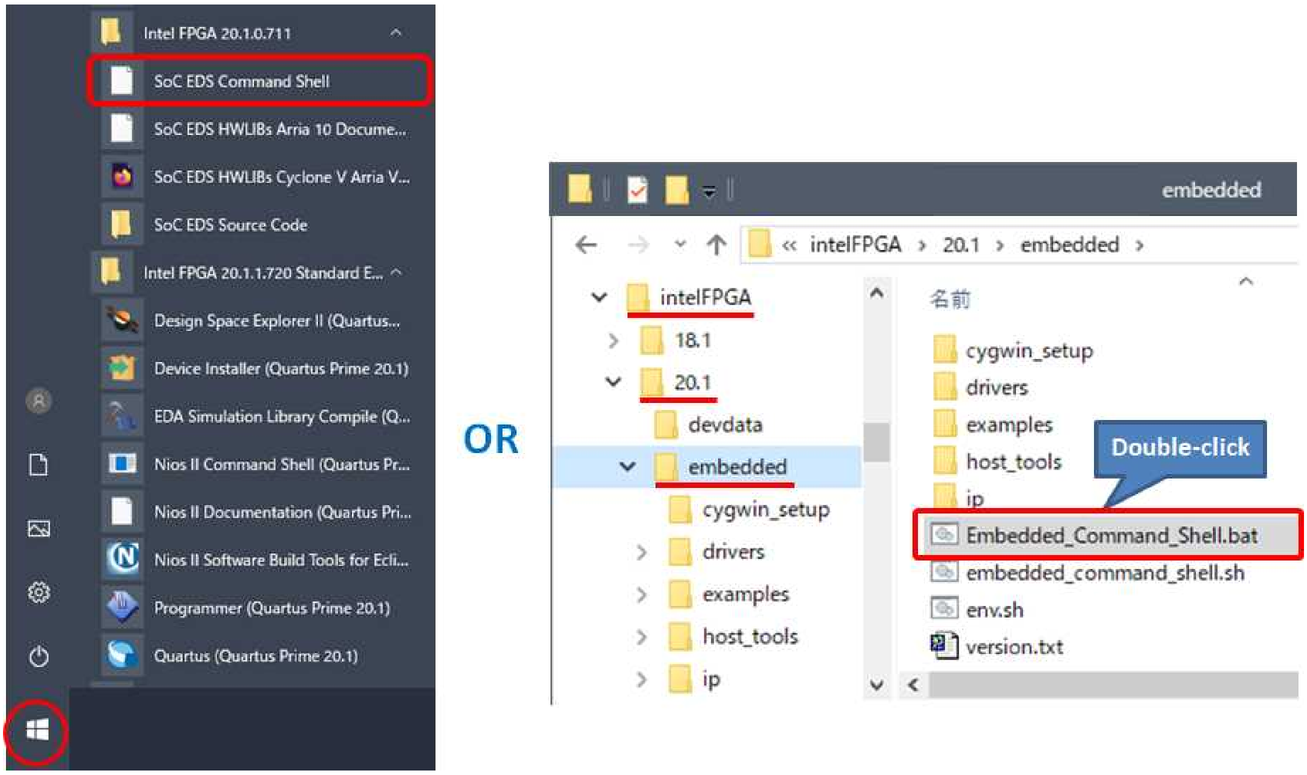
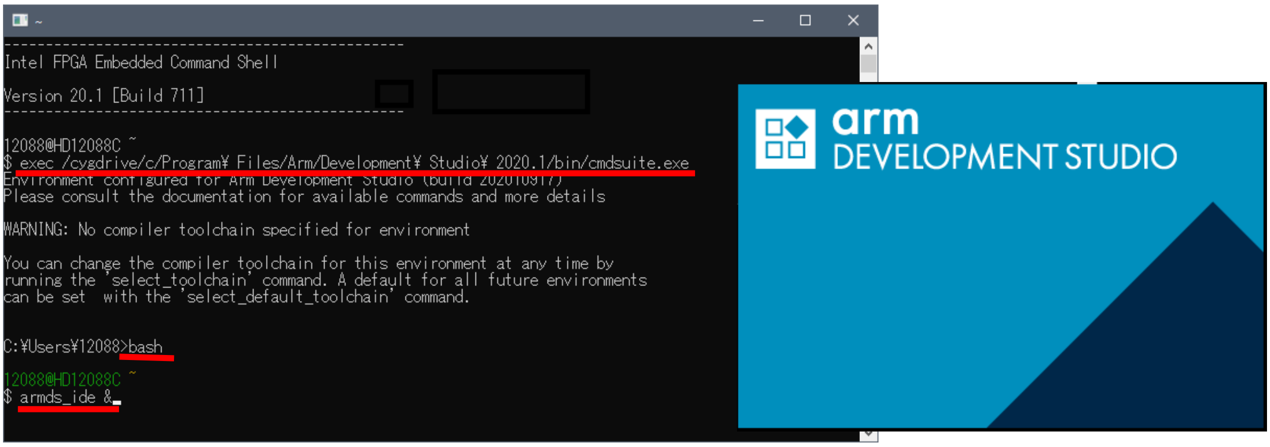

Intel FPGA 20.1.0.711
SoC EDS Command Shell
SoC EDS HWLIBs Arria 10 Docume...
SoC EDS HWLIBs Cyclone V Arria V...
SoC EDS Source Code
Intel FPGA 20.1.1.720 Standard E...
Design Space Explorer II (Quartus...
Device Installer (Quartus Prime 20.1)
EDA Simulation Library Compile (Q...
Nios II Command Shell (Quartus Pr...
Nios II Documentation (Quartus Pri...
Nios II Software Build Tools for Ecli...
Programmer (Quartus Prime 20.1)
Quartus (Quartus Prime 20.1)
OR
embedded
« intelFPGA › 20.1 › embedded ›
intelFPGA
18.1
20.1
devdata
embedded
cygwin_setup
drivers
examples
host_tools
ip
名前
cygwin_setup
drivers
examples
host_tools
ip
Embedded_Command_Shell.bat
embedded_command_shell.sh
env.sh
version.txt
Double-click

Figure 6-15. Launch Embedded Command Shell

____ 2. Run the following command on the Embedded Command Shell to launch Arm® DS:

```
$ exec /cygdrive/c/Program¥ Files/Arm/Development¥ Studio¥ 2020.1/bin/cmdsuite.exe ↵
> bash ↵
$ armds_ide & ↵
```



```
-------------------------------------------------
Intel FPGA Embedded Command Shell

Version 20.1 [Build 711]           [redacted]  [redacted]
-------------------------------------------------

12088@HD12088C ~
$ exec /cygdrive/c/Program¥ Files/Arm/Development¥ Studio¥ 2020.1/bin/cmdsuite.exe
Environment configured for Arm Development Studio (build 202010917)
Please consult the documentation for available commands and more details

WARNING: No compiler toolchain specified for environment

You can change the compiler toolchain for this environment at any time by
running the 'select_toolchain' command. A default for all future environments
can be set  with the 'select_default_toolchain' command.

C:¥Users¥12088>bash

12088@HD12088C ~
$ armds_ide &_
```

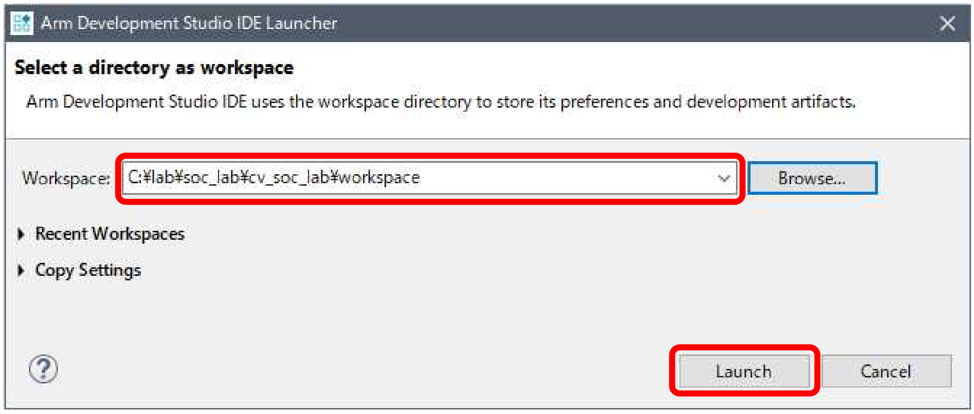
Figure 6-16. Launch Arm® DS from the Embedded Command Shell

____ 3. Set up a workspace folder for working with Arm® DS.

In this exercise, you will create a workspace in the working folder of "3. Lab 1: Hardware Exercise".

Specify the following path and click [***Launch***] (if the folder does not exist, it will be created automatically):

```
C:¥lab¥soc_lab¥cv_soc_lab¥workspace
```



Arm Development Studio IDE Launcher

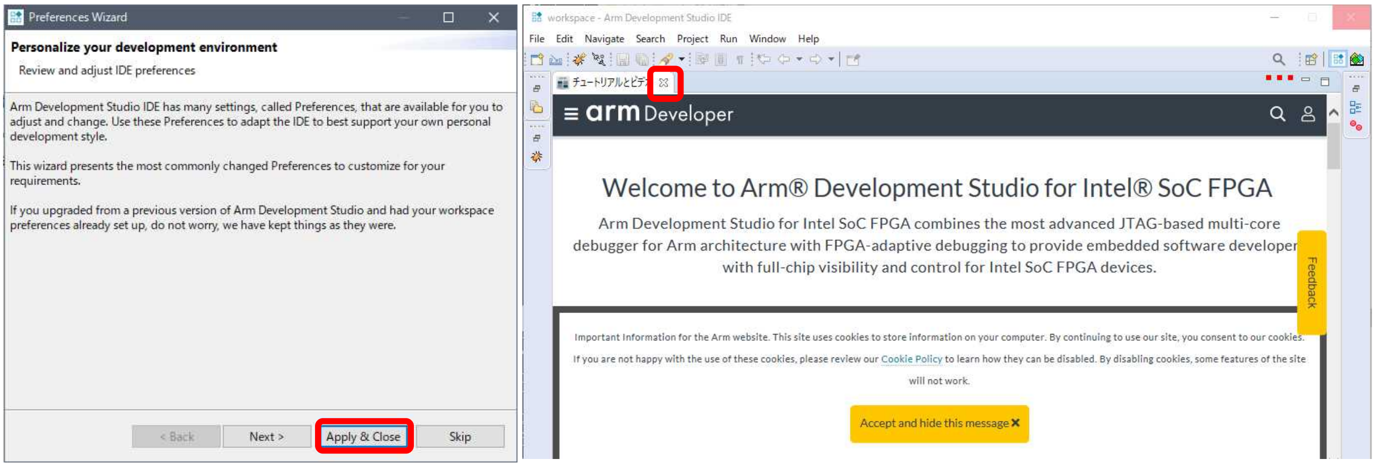
Select a directory as workspace

Arm Development Studio IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:¥lab¥soc_lab¥cv_soc_lab¥workspace

Figure 6-17. Creating a Workspace

____ 4. If the Preferences Wizard appears, confirm it and click [***Apply & Close***]. If the Arm® DS Welcome screen appears, click **Close** (**X**) to close it.



Preferences Wizard

Personalize your development environment

Review and adjust IDE preferences

Arm Development Studio IDE has many settings, called Preferences, that are available for you to adjust and change. Use these Preferences to adapt the IDE to best support your own personal development style.

This wizard presents the most commonly changed Preferences to customize for your requirements.

If you upgraded from a previous version of Arm Development Studio and had your workspace preferences already set up, do not worry, we have kept things as they were.

workspace - Arm Development Studio IDE

Welcome to Arm® Development Studio for Intel® SoC FPGA

Arm Development Studio for Intel SoC FPGA combines the most advanced JTAG-based multi-core debugger for Arm architecture with FPGA-adaptive debugging to provide embedded software developer with full-chip visibility and control for Intel SoC FPGA devices.

Important Information for the Arm website. This site uses cookies to store information on your computer. By continuing to use our site, you consent to our cookies. If you are not happy with the use of these cookies, please review our Cookie Policy to learn how they can be disabled. By disabling cookies, some features of the site will not work.

Figure 6-18. Preference Wizard and Welcome Screen

ⓘ **Info:**

In Arm® DS (version 2020.1), it has been confirmed that window operation becomes slow when the Welcome screen (tutorial and video) is displayed online. Please wait until the online display is complete without forcing the operation. After closing the Welcome screen, operation will be smooth.

If you start Arm® DS offline, the offline Welcome screen is used, and this problem can be avoided.

____ 5. From the Arm® DS menu, select ***File*** => ***Import***.

____ 6. Select ***General*** => "***Existing Projects into Workspaces***" and click [***Next***].

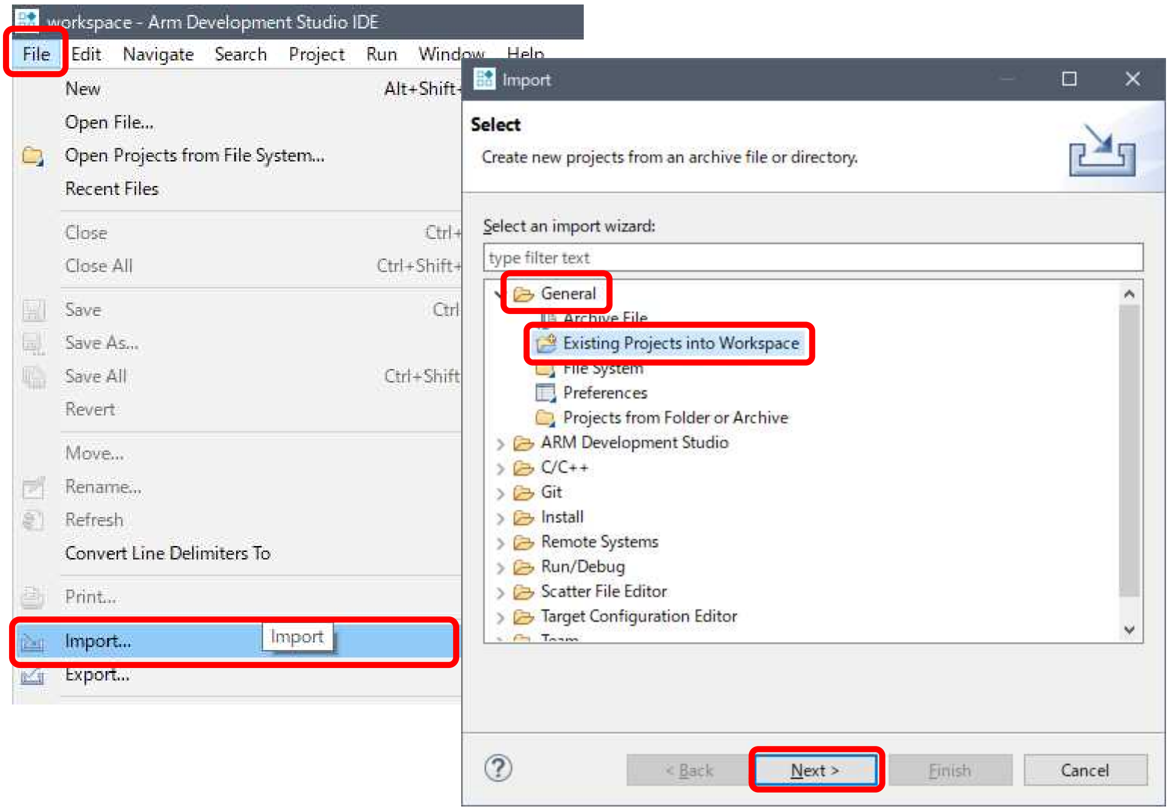


Figure 6-19. File => Import

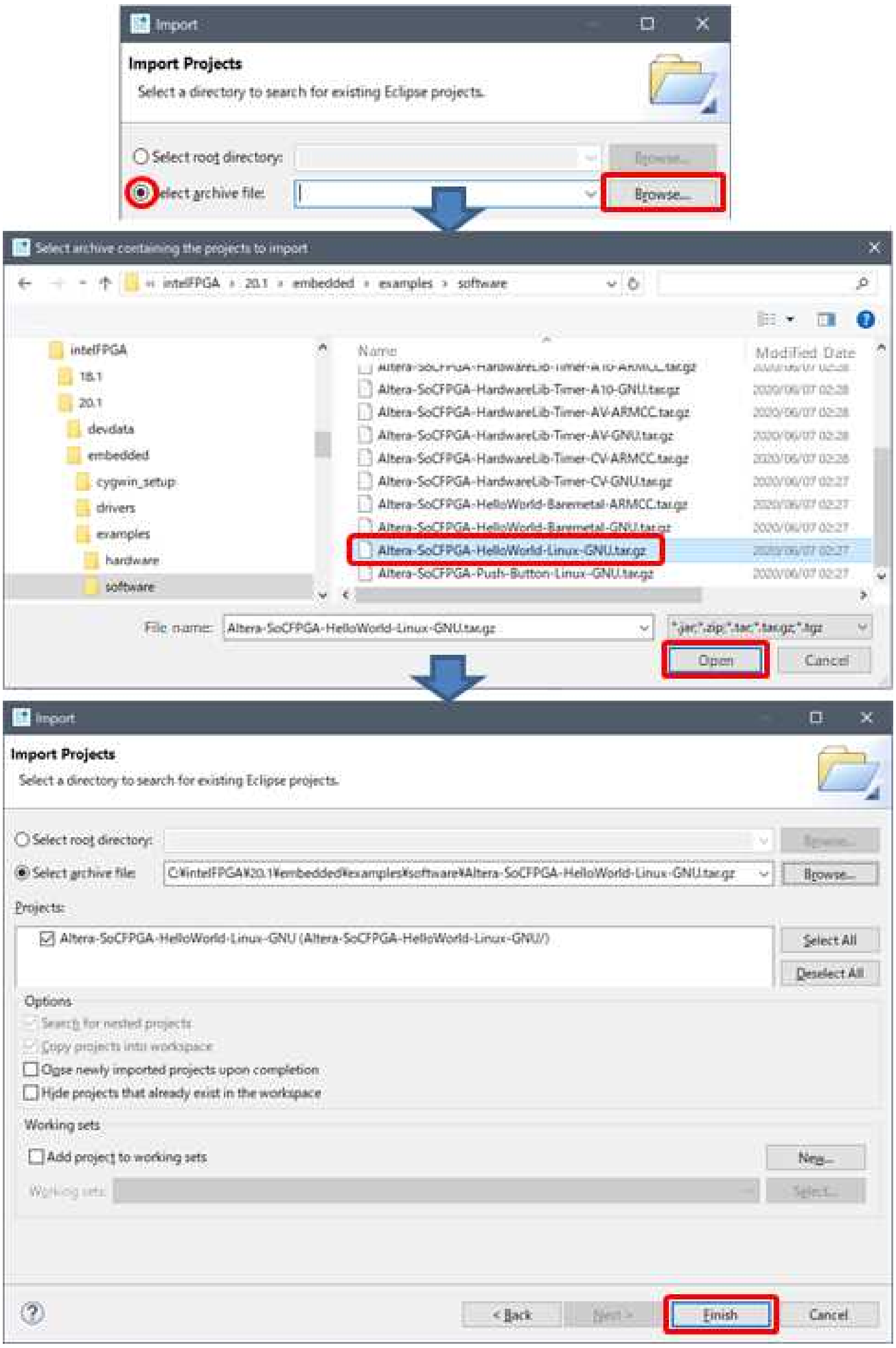
____ 7. ***Select archive file:*** option and use the [***Browse***] button to locate the sample project.

The sample project is included with SoC EDS and can be found by default in the following installation folder:

```
C:¥intelFPGA¥20.1¥embedded¥examples¥software¥Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz
```

(Importing <SoC EDS installation directory >`¥examples¥software¥Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz`).

Once selected, click the [***Finish***] button.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory:

● Select archive file:

Select archive containing the projects to import

intelFPGA › 20.1 › embedded › examples › software

Name	Modified Date
Altera-SoCFPGA-HardwareLib-Timer-A10-ARMCC.tar.gz	2020/06/07 02:28
Altera-SoCFPGA-HardwareLib-Timer-A10-GNU.tar.gz	2020/06/07 02:28
Altera-SoCFPGA-HardwareLib-Timer-AV-ARMCC.tar.gz	2020/06/07 02:28
Altera-SoCFPGA-HardwareLib-Timer-AV-GNU.tar.gz	2020/06/07 02:28
Altera-SoCFPGA-HardwareLib-Timer-CV-ARMCC.tar.gz	2020/06/07 02:28
Altera-SoCFPGA-HardwareLib-Timer-CV-GNU.tar.gz	2020/06/07 02:27
Altera-SoCFPGA-HelloWorld-Baremetal-ARMCC.tar.gz	2020/06/07 02:27
Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar.gz	2020/06/07 02:27
Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz	2020/06/07 02:27
Altera-SoCFPGA-Push-Button-Linux-GNU.tar.gz	2020/06/07 02:27

File name: Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz

.jar;.zip;*.tar;*.tar.gz;*.tgz

Import

Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory:

● Select archive file: C:¥intelFPGA¥20.1¥embedded¥examples¥software¥Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz

Projects:

☑ Altera-SoCFPGA-HelloWorld-Linux-GNU (Altera-SoCFPGA-HelloWorld-Linux-GNU/)

Options

☑ Search for nested projects

☑ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets:

Figure 6-20. Importing the sample project

____ 8. The Altera-SoCFPGA-HelloWorld-Linux-GNU project has been added to the Project Explorer on the left side of the Arm® DS window. Click and expand Altera-SoCFPGA-HelloWorld-Linux-GNU to view the various files contained in the project. If the project icon is marked with a red X, the tool chain is configured. If the X is not marked, ____ 11 Go to.

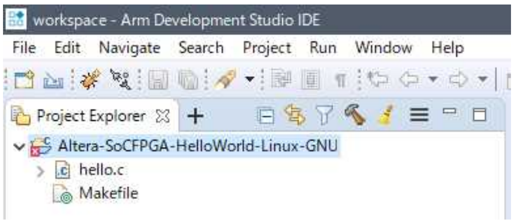
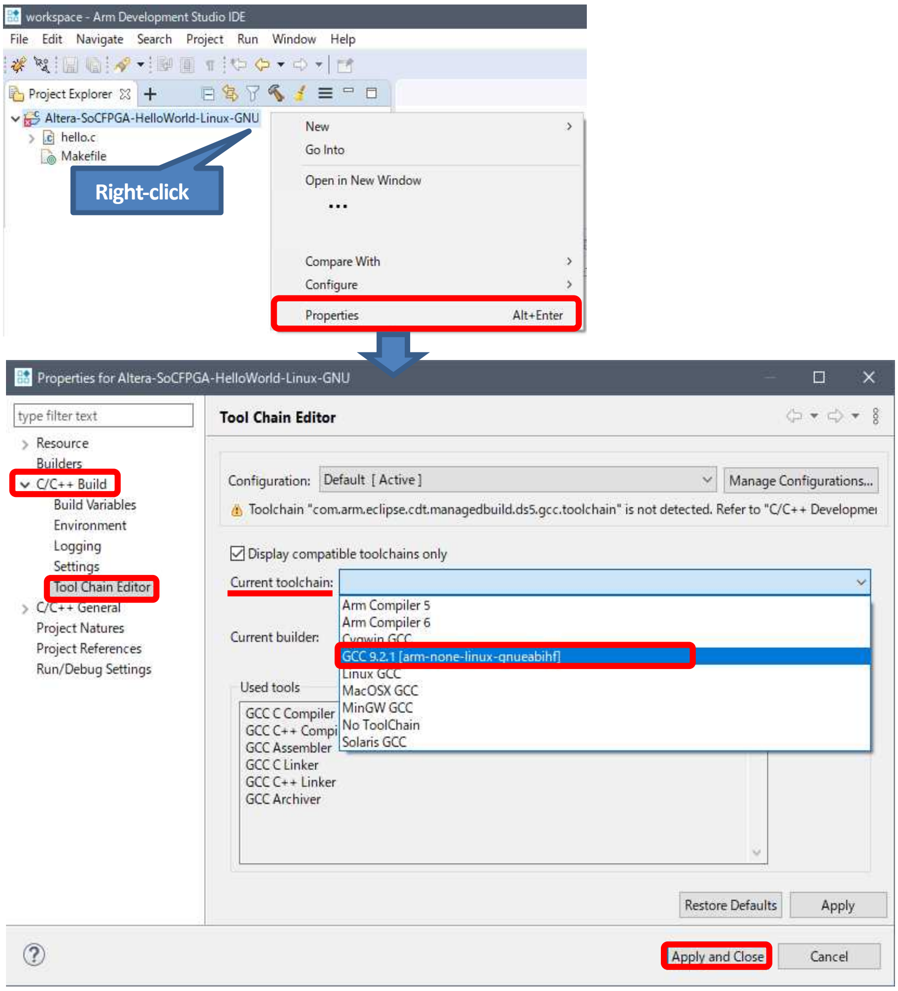

workspace - Arm Development Studio IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
Altera-SoCFPGA-HelloWorld-Linux-GNU
hello.c
Makefile

Figure 6-21. Altera-SoCFPGA-HelloWorld-Linux-GNU project added (toolchain not found)

____ 9. In the Project Explorer, select Properties from the right-click menu for Altera-SoCFPGA-HelloWorld-Linux-GNU to view the project properties. In ***C/C++ Build*** => ***Tool Chain Editor***, ***Current toolchain:*** is unselected, so select arm-one-linux-gnueabihf or arm-linux-gnueabihf and click [***Apply and Close***]. If you have selected a toolchain, go to ____ 11 Go to. If arm-one-linux-gnueabihf or arm-linux-gnueabihf is not a candidate, follow 0 and then repeat this procedure.



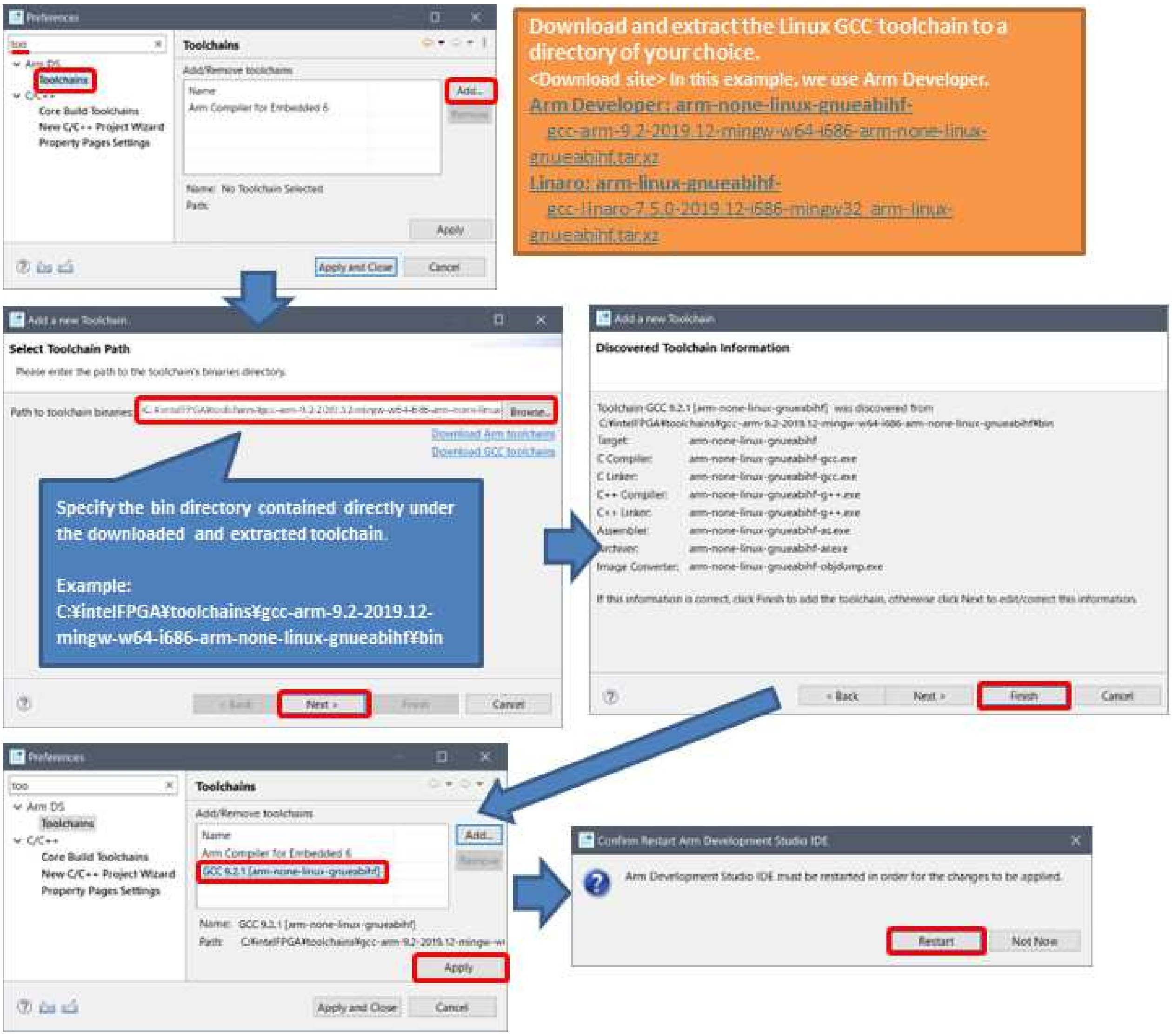
Configuration: Default [Active]

Toolchain "com.arm.eclipse.cdt.managedbuild.ds5.gcc.toolchain" is not detected. Refer to "C/C++ Developmei

☑ Display compatible toolchains only

Figure 6-22. Selecting the Linux GCC toolchain

____ 10. Select "***Window***" menu => "***Preferences***," then "***ARM DS***" => "***Toolchains***." Since GCC (arm-linux-gnueabihf or arm-none-linux-gnueabihf) for Linux is not in the list, use the [***Add***] button and follow the instructions in the window to add the toolchain. When the addition is complete, restart Arm® DS.



Download and extract the Linux GCC toolchain to a directory of your choice.
<Download site> In this example, we use Arm Developer.
Arm Developer: arm-none-linux-gnueabihf-
gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux-gnueabihf.tar.xz
Linaro: arm-linux-gnueabihf-
gcc-linaro-7.5.0-2019.12-i686-mingw32_arm-linux-gnueabihf.tar.xz

Name: No Toolchain Selected
Path:

Select Toolchain Path

Please enter the path to the toolchain's binaries directory.

Path to toolchain binaries: C:¥intelFPGA¥toolchains¥gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux

Specify the bin directory contained directly under the downloaded and extracted toolchain.

Example:
C:¥intelFPGA¥toolchains¥gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux-gnueabihf¥bin

Discovered Toolchain Information

```
Toolchain GCC 9.2.1 [arm-none-linux-gnueabihf]  was discovered from
 C:¥intelFPGA¥toolchains¥gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux-gnueabihf¥bin
Target:            arm-none-linux-gnueabihf
C Compiler:        arm-none-linux-gnueabihf-gcc.exe
C Linker:          arm-none-linux-gnueabihf-gcc.exe
C++ Compiler:      arm-none-linux-gnueabihf-g++.exe
C++ Linker:        arm-none-linux-gnueabihf-g++.exe
Assembler:         arm-none-linux-gnueabihf-as.exe
Archiver:          arm-none-linux-gnueabihf-ar.exe
Image Converter:   arm-none-linux-gnueabihf-objdump.exe
```

If this information is correct, click Finish to add the toolchain, otherwise click Next to edit/correct this information.

Toolchains: Arm Compiler for Embedded 6, GCC 9.2.1 [arm-none-linux-gnueabihf]

Name: GCC 9.2.1 [arm-none-linux-gnueabihf]
Path: C:¥intelFPGA¥toolchains¥gcc-arm-9.2-2019.12-mingw-wi

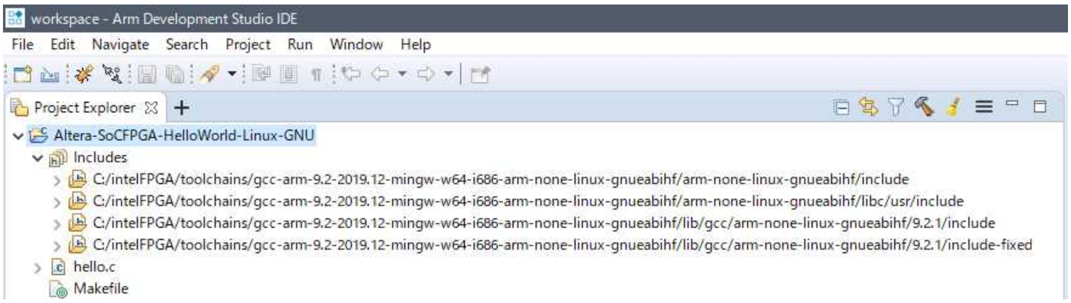
Confirm Restart Arm Development Studio IDE

Arm Development Studio IDE must be restarted in order for the changes to be applied.

Restart Not Now

Figure 6-23. Adding the Linux GCC toolchain

____ 11. The Altera-SoCFPGA-HelloWorld-Linux-GNU project has been added to the Project Explorer. Click on Altera-SoCFPGA-HelloWorld-Linux-GNU to expand it and view the various files contained in the project.


workspace - Arm Development Studio IDE
File Edit Navigate Search Project Run Window Help
Project Explorer

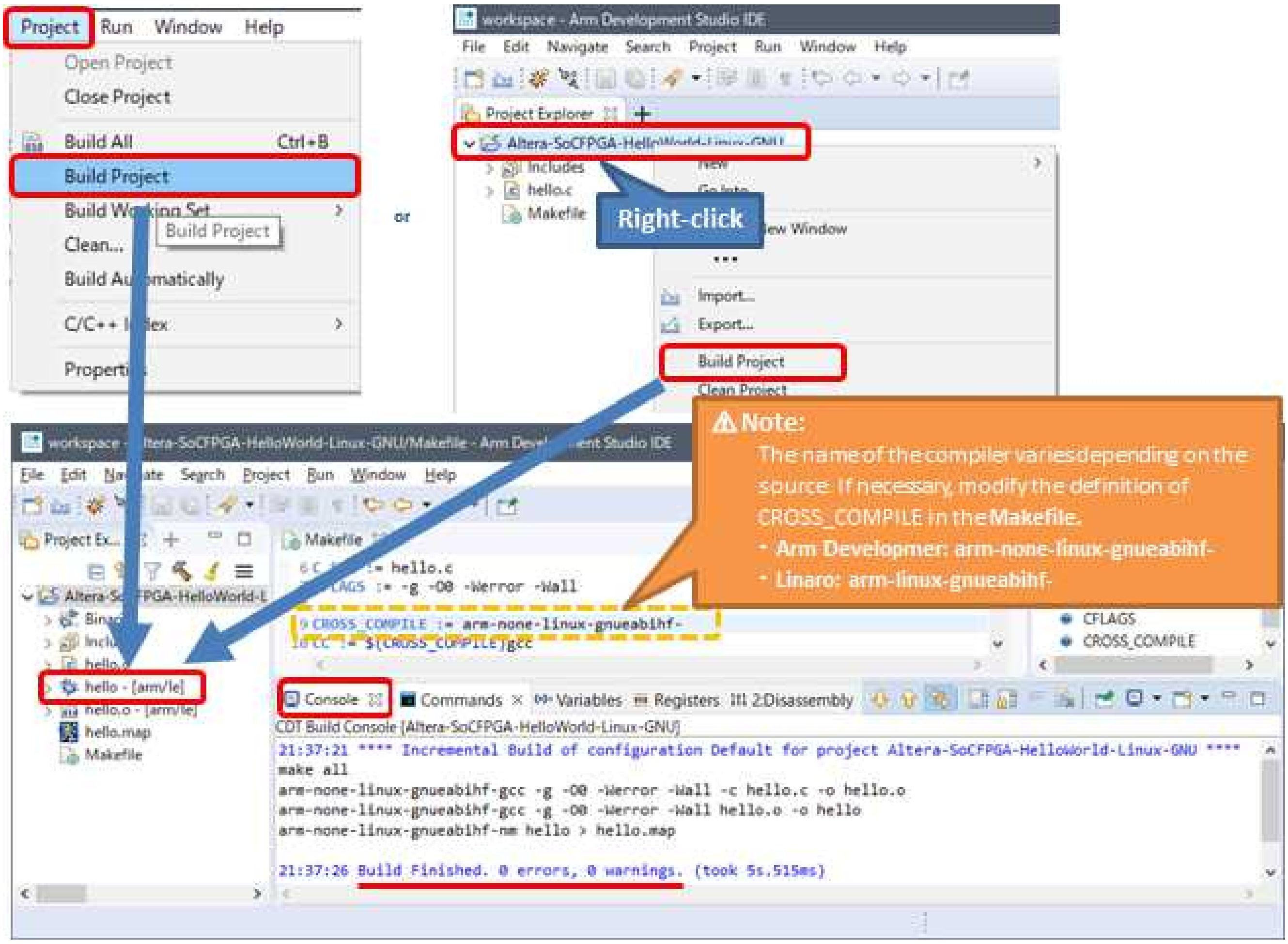
```
Altera-SoCFPGA-HelloWorld-Linux-GNU
  Includes
    C:/intelFPGA/toolchains/gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux-gnueabihf/arm-none-linux-gnueabihf/include
    C:/intelFPGA/toolchains/gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux-gnueabihf/arm-none-linux-gnueabihf/libc/usr/include
    C:/intelFPGA/toolchains/gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux-gnueabihf/lib/gcc/arm-none-linux-gnueabihf/9.2.1/include
    C:/intelFPGA/toolchains/gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux-gnueabihf/lib/gcc/arm-none-linux-gnueabihf/9.2.1/include-fixed
  hello.c
  Makefile
```

Figure 6-24. Altera-SoCFPGA-HelloWorld-Linux-GNU project added (normal)

____ 12. Build the Altera-SoCFPGA-HelloWorld-Linux-GNU application.

Highlight the Altera-SoCFPGA-HelloWorld-Linux-GNU project in the Project Explorer and choose ***Project*** => ***Build Project***. Alternatively, select the project in the Project Explorer and **right-click *Build Project***.

The newly generated hello executable is output in the Project Explorer.



⚠Note:
The name of the compiler varies depending on the source. If necessary, modify the definition of CROSS_COMPILE in the **Makefile**.
- **Arm Developmer: arm-none-linux-gnueabihf-**
- **Linaro: arm-linux-gnueabihf-**

```
C_SRC := hello.c
CFLAGS := -g -O0 -Werror -Wall
CROSS_COMPILE := arm-none-linux-gnueabihf-
CC := $(CROSS_COMPILE)gcc
```

CDT Build Console [Altera-SoCFPGA-HelloWorld-Linux-GNU]

```
21:37:21 **** Incremental Build of configuration Default for project Altera-SoCFPGA-HelloWorld-Linux-GNU ****
make all
arm-none-linux-gnueabihf-gcc -g -O0 -Werror -Wall -c hello.c -o hello.o
arm-none-linux-gnueabihf-gcc -g -O0 -Werror -Wall hello.o -o hello
arm-none-linux-gnueabihf-nm hello > hello.map

21:37:26 Build Finished. 0 errors, 0 warnings. (took 5s.515ms)
```

Figure 6-25. Building the project

6-6. Setting up Remote System Explorer (RSE)

With Arm® DS, you can use Remote System Explorer (RSE) to run and debug Linux application programs on the target.

____ 1. Choose ***Window*** => ***Perspective*** => ***Open Perspective*** => ***Other***.

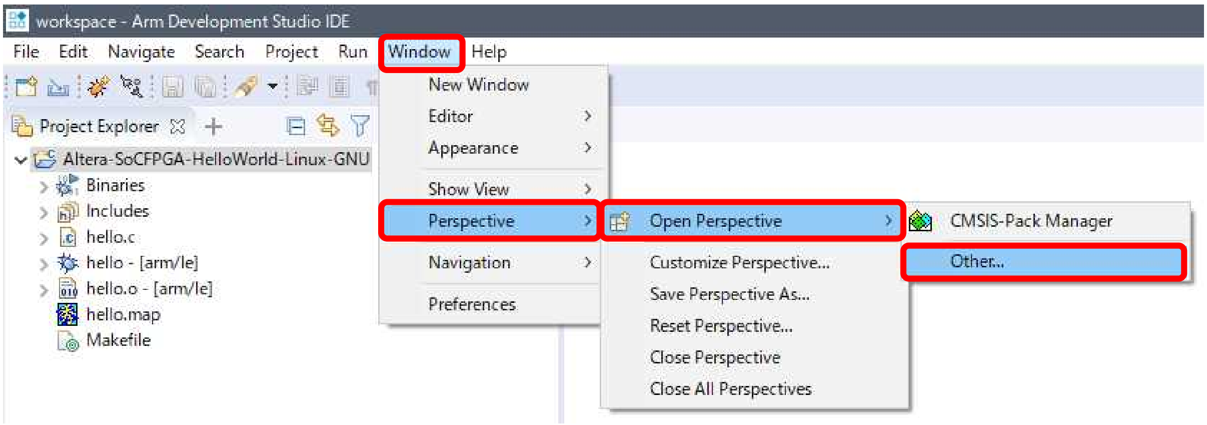


Figure 6-26. Choose Open Perspective -> Other

____ 2. Select "***Remote System Explorer***" and click [***Open***].

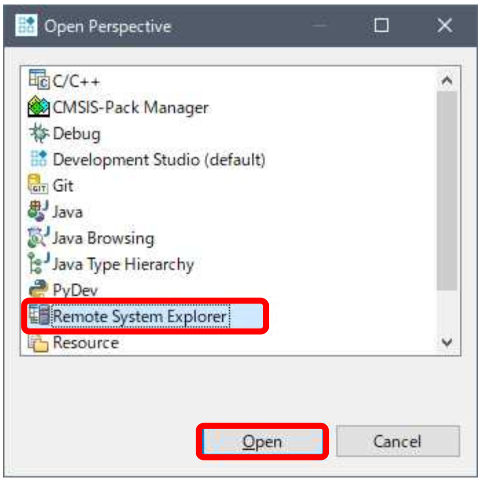


Figure 6-27. Selecting Remote System Explorer

____ 3. In the Remote System Explorer view button or right-click a blank area and select ***New Connection***.

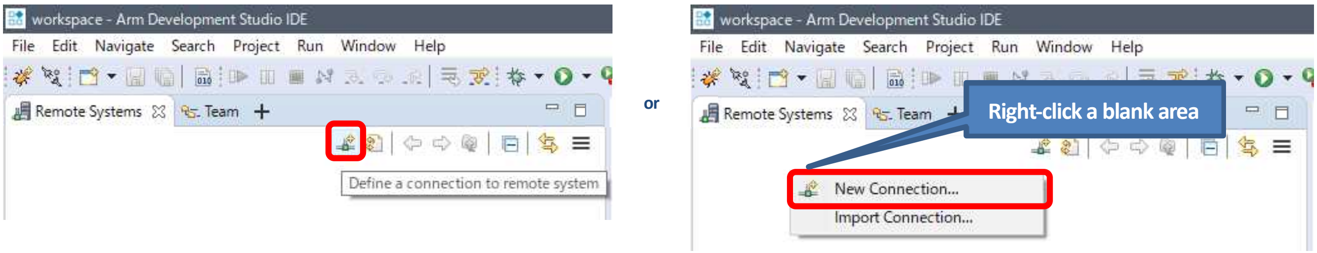
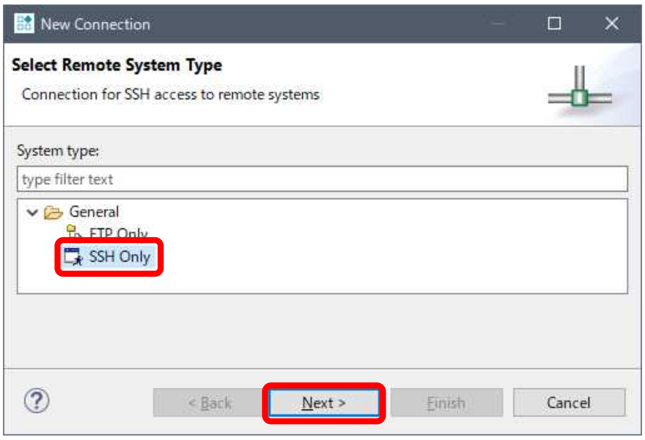

workspace - Arm Development Studio IDE
File Edit Navigate Search Project Run Window Help
Remote Systems
Team
Define a connection to remote system
or
workspace - Arm Development Studio IDE
File Edit Navigate Search Project Run Window Help
Remote Systems
Team
Right-click a blank area
New Connection...
Import Connection...

Figure 6-28. New connection in Remote System Explorer

____4. Select ***SSH Only*** in the Select Remote System Type view and click [***Next***].



New Connection

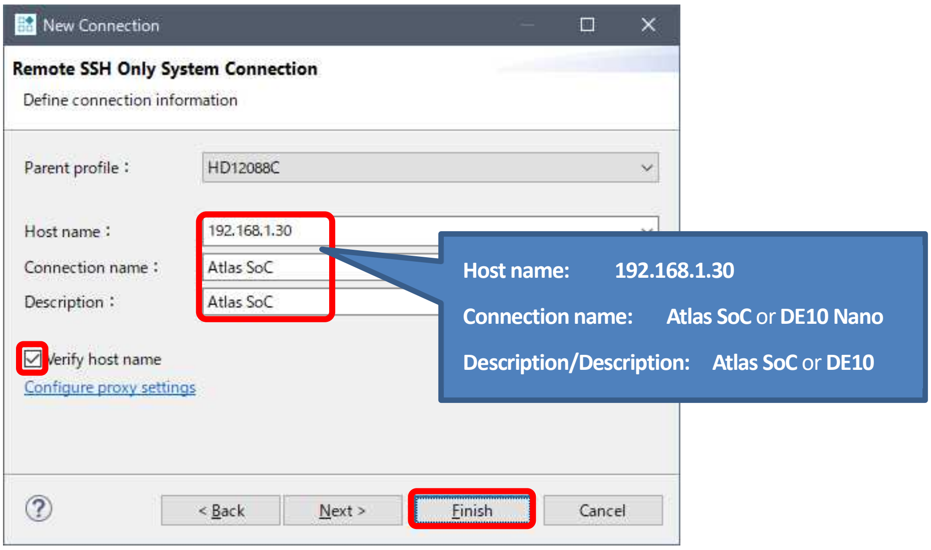
Select Remote System Type

Connection for SSH access to remote systems

System type: type filter text

Figure 6-29. Select SSH Only

____5. In the Host name: field, enter the IP address of the board (192.168.1.30 in this example). In the Connection name: and Description: field, enter **Atlas SoC** or **DE10 Nano**. Check Verify host name and click [***Finish***].



New Connection

Remote SSH Only System Connection

Define connection information

Parent profile : HD12088C

Host name : 192.168.1.30

Connection name : Atlas SoC

Description : Atlas SoC

☑ Verify host name

Configure proxy settings

Host name: **192.168.1.30**

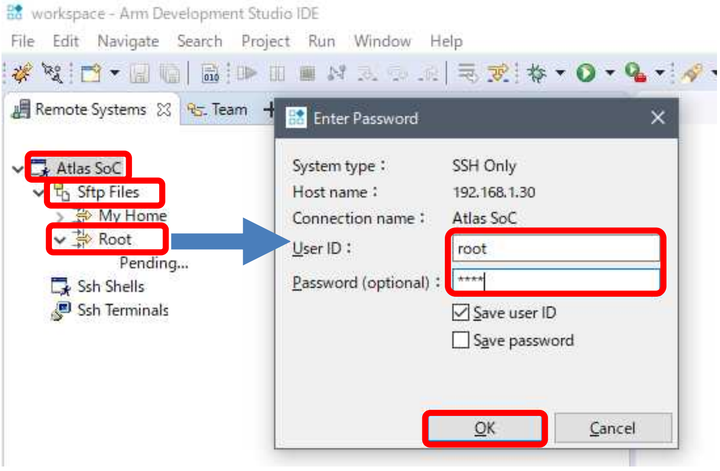
Connection name: **Atlas SoC** or **DE10 Nano**

Description/Description: **Atlas SoC** or **DE10**

Figure 6-30. Connection Settings

____ 6. In the Remote System Explorer view, click ***Atlas SoC*** (or ***DE10 Nano***) => ***Sftp Files*** => ***Root***. A window for entering the user ID and password will appear.

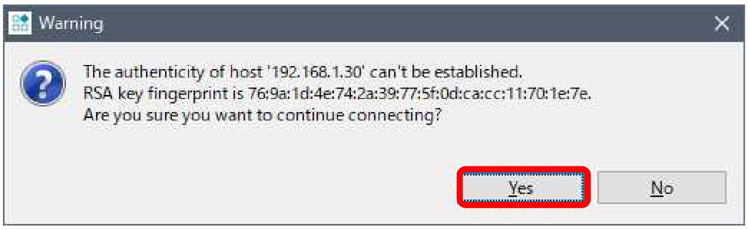
____ 7. Enter "**root**" for "User ID:" and the password you set for "***Password***" and click [***OK***].



System type : SSH Only
Host name : 192.168.1.30
Connection name : Atlas SoC
User ID : root
Password (optional) : ****
☑ Save user ID
□ Save password

Figure 6-31. Enter the User ID and Password

____ 8. If the warning shown below appears, click [***Yes***].



The authenticity of host '192.168.1.30' can't be established.
RSA key fingerprint is 76:9a:1d:4e:74:2a:39:77:5f:0d:ca:cc:11:70:1e:7e.
Are you sure you want to continue connecting?

Figure 6-32. Warning display

____ 9. If the connection is successful, Remote System Explorer displays the files on the current board.

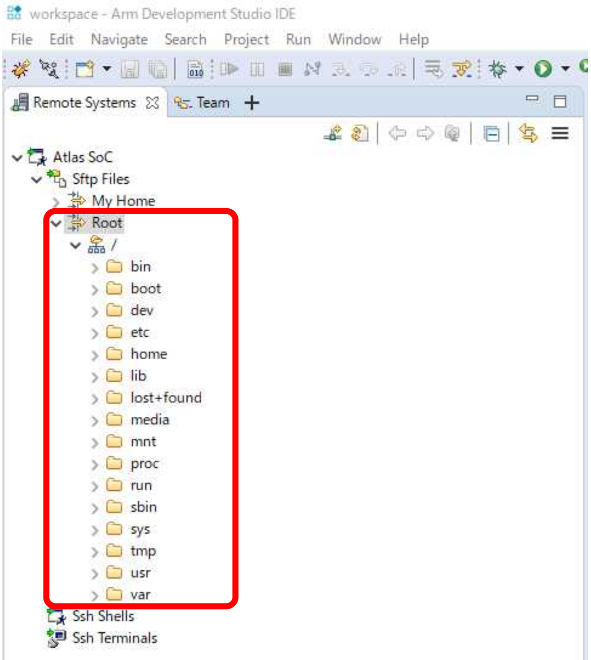


Figure 6-33. Remote System Explorer displays the files on the current board

____ 10. If you are unable to connect with an error, it may be a problem with the proxy settings on the host PC. In this case, click "***Control Panel***" -> "***Internet Options***," then click "***LAN Settings***" in the "Connections" tab.

____ 11. If "***Use proxy server for LAN***" is checked, uncheck it and click [***OK***].

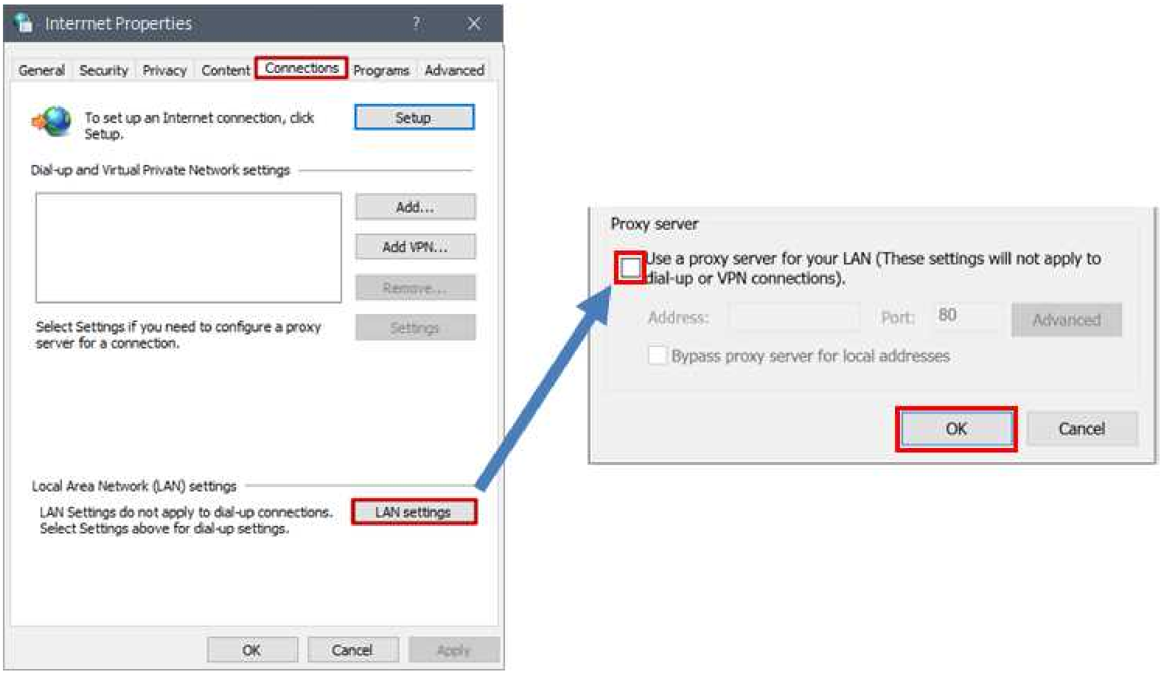

Internet Properties
General
Security
Privacy
Content
Connections
Programs
Advanced
To set up an Internet connection, click Setup.
Setup
Dial-up and Virtual Private Network settings
Add...
Add VPN...
Remove...
Select Settings if you need to configure a proxy server for a connection.
Settings
Local Area Network (LAN) settings
LAN Settings do not apply to dial-up connections. Select Settings above for dial-up settings.
LAN settings
OK
Cancel
Apply
Proxy server
Use a proxy server for your LAN (These settings will not apply to dial-up or VPN connections).
Address:
Port:
80
Advanced
Bypass proxy server for local addresses
OK
Cancel

Figure 6-34. Proxy Server Settings

____ 12. Try connecting to the root of the **Atlas SoC** (or **DE10 Nano**) again.

6-7. Running and Debugging Linux Applications

This section describes how to set up the debugger and how to run and debug it.

____ 1. Click the button in the menu bar (top right of the window) to return to the main perspective of Development Studio. From the Project Explorer tab, right-click Altera-SoCFPGA-HelloWorld-Linux-GNU.

____ 2. . From the Project Explorer tab, **right-click *Altera-SoCFPGA-HelloWorld-Linux-GNU***

Select ***Debug As*** => ***Debug Configurations***.

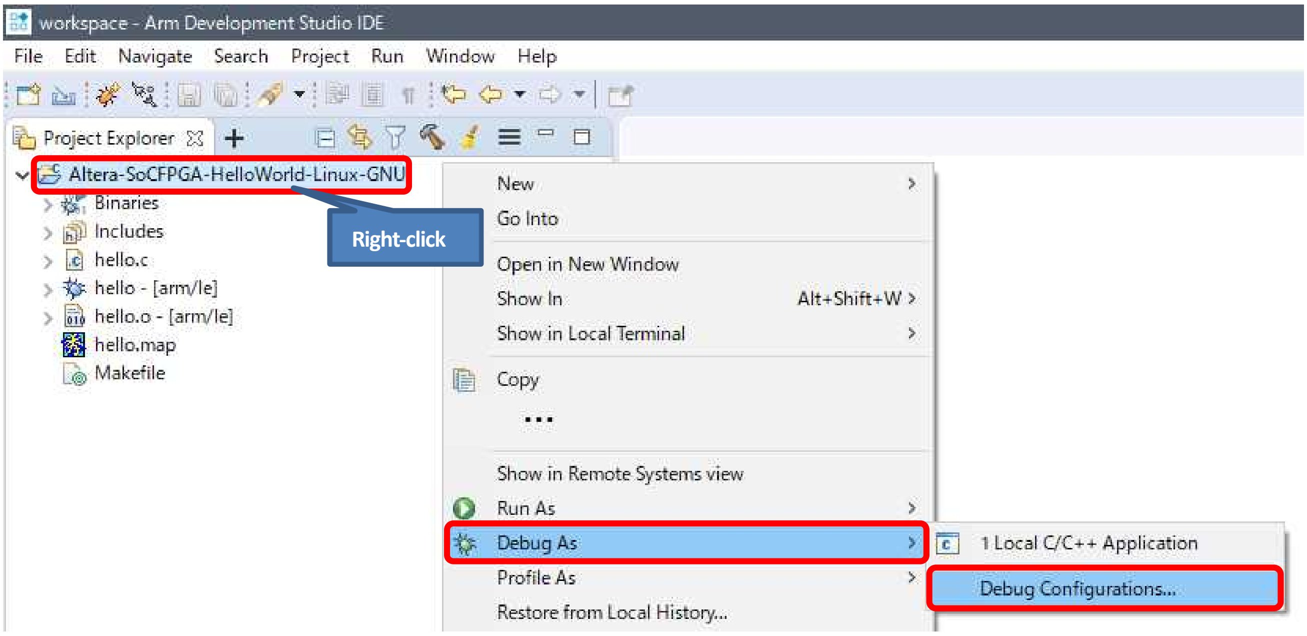


Figure 6-35. Select “Debug” => “Debug Configuration”

____ 3. **Right-click** “***Generic ARM C/C++ Application***” and select “***New Configuration***” to create a new debug configuration.

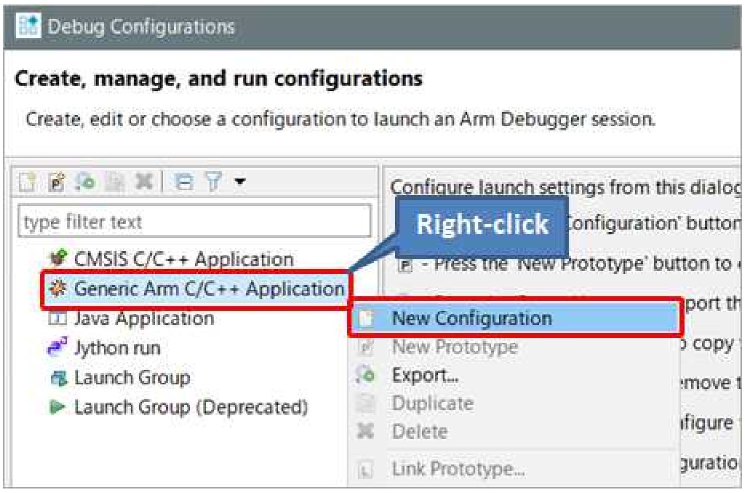

Debug Configurations
Create, manage, and run configurations
Create, edit or choose a configuration to launch an Arm Debugger session.
type filter text
CMSIS C/C++ Application
Generic Arm C/C++ Application
Java Application
Jython run
Launch Group
Launch Group (Deprecated)
Right-click
New Configuration
New Prototype
Export...
Duplicate
Delete
Link Prototype...
Configure launch settings from this dialog
onfiguration' button
- Press the 'New Prototype' button to
port th
copy
move
figure
guratio

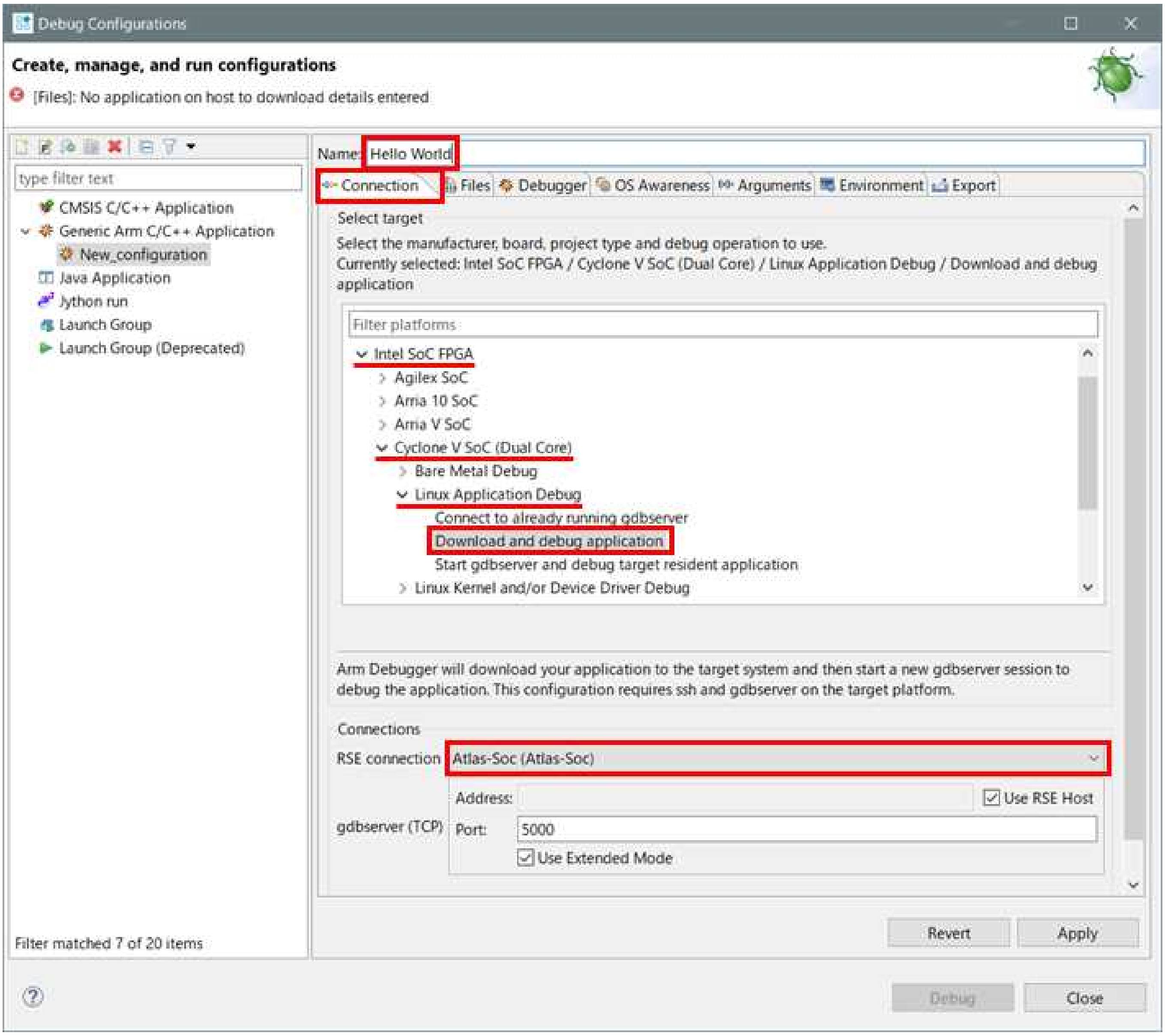
Figure 6-36. Creating a new debug configuration

____ 4. Type "**HelloWorld**" in the Name field.

____ 5. In the ***Select Target*** field of the Connections tab, select

Intel SoC FPGA => Cyclone V SoC (Dual Core) => Linux Application Debug => Download and debug application.

____ 6. In the ***Connection*** field, select the generated RSE connection (**Atlas SoC** in this example) and accept the other defaults.



Debug Configurations

Create, manage, and run configurations

[Files]: No application on host to download details entered

Name: Hello World

Select target

Select the manufacturer, board, project type and debug operation to use.
Currently selected: Intel SoC FPGA / Cyclone V SoC (Dual Core) / Linux Application Debug / Download and debug application

Arm Debugger will download your application to the target system and then start a new gdbserver session to debug the application. This configuration requires ssh and gdbserver on the target platform.

Connections

RSE connection: Atlas-Soc (Atlas-Soc)

gdbserver (TCP) Address: ☑ Use RSE Host

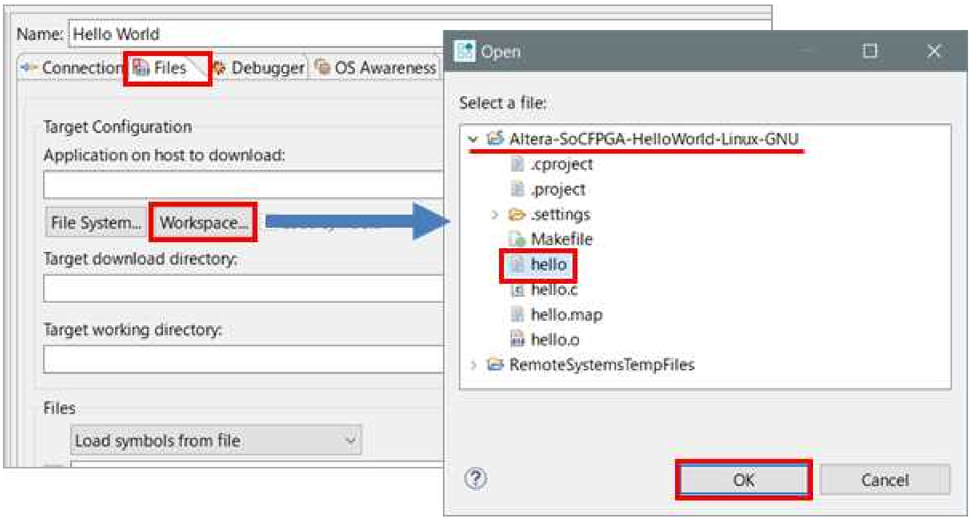
Port: 5000

☑ Use Extended Mode

Filter matched 7 of 20 items

Figure 6-37. Set debug configuration (1)

____ 7. In the ***File*** tab, set the **Application on host to download** to the Hello World executable. Select ***hello*** using the [***Workspace***] button and click [***OK***].



Name: Hello World

Connection | Files | Debugger | OS Awareness

Target Configuration

Application on host to download:

File System... | Workspace...

Target download directory:

Target working directory:

Files

Load symbols from file

Open

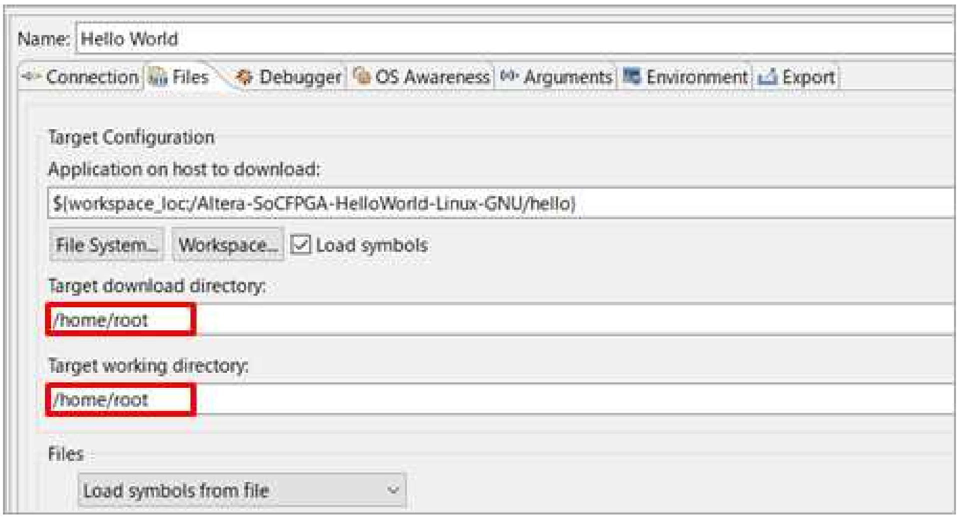
Select a file:

- Altera-SoCFPGA-HelloWorld-Linux-GNU
  - .cproject
  - .project
  - .settings
  - Makefile
  - hello
  - hello.c
  - hello.map
  - hello.o
- RemoteSystemsTempFiles

OK | Cancel

Figure 6-38. Setting the debug configuration (2)

____ 8. Set ***Target Download Directory:*** and ***Target Working Directory:*** to**/home/root**.



Name: Hello World

Connection | Files | Debugger | OS Awareness | Arguments | Environment | Export

Target Configuration

Application on host to download:

${workspace_loc:/Altera-SoCFPGA-HelloWorld-Linux-GNU/hello}

File System... | Workspace... | ☑ Load symbols

Target download directory:

/home/root

Target working directory:

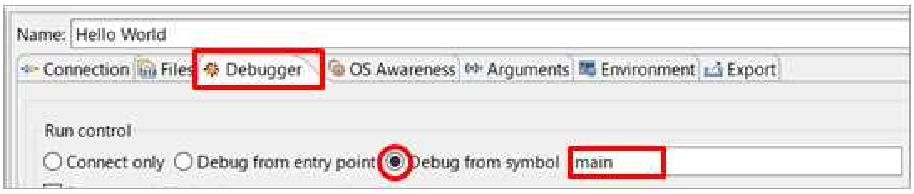
/home/root

Files

Load symbols from file

Figure 6-39. Setting the debug configuration (3)

____ 9. In the ***Debugger*** tab, select ***Debug from symbols*** as the execution control field and enter **main** as the symbol name.



Name: Hello World

Connection | Files | Debugger | OS Awareness | Arguments | Environment | Export

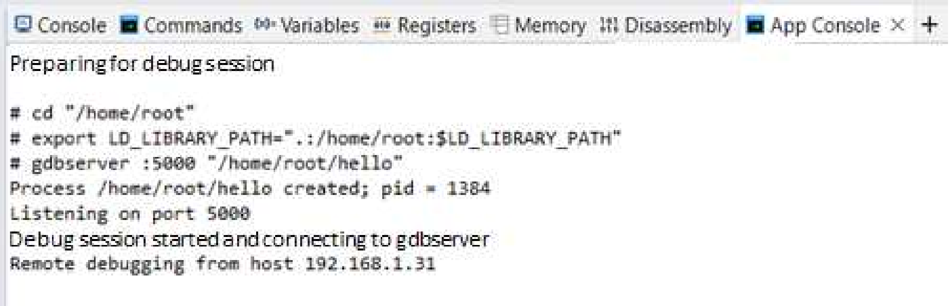
Run control

○ Connect only ○ Debug from entry point ● Debug from symbol main

Figure 6-40. Setting the debug configuration (4)

____ 10. Click the [***Debug***] button Debug to start the debugging session.

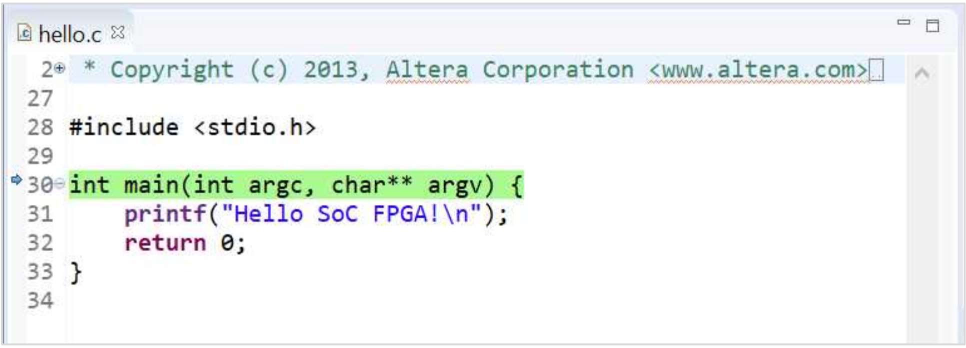
____ 11. When a debugging session starts, the App Console displays a log similar to the following:



```
Preparing for debug session

# cd "/home/root"
# export LD_LIBRARY_PATH=".:/home/root:$LD_LIBRARY_PATH"
# gdbserver :5000 "/home/root/hello"
Process /home/root/hello created; pid = 1384
Listening on port 5000
Debug session started and connecting to gdbserver
Remote debugging from host 192.168.1.31
```

Figure 6-41. App Console at the start of a debugging session

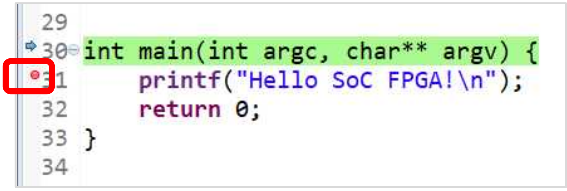
____ 12. The application is loaded and then breaks in the main function.



```
 * Copyright (c) 2013, Altera Corporation <www.altera.com>

#include <stdio.h>

int main(int argc, char** argv) {
    printf("Hello SoC FPGA!\n");
    return 0;
}

```

Figure 6-42. Break in Main function

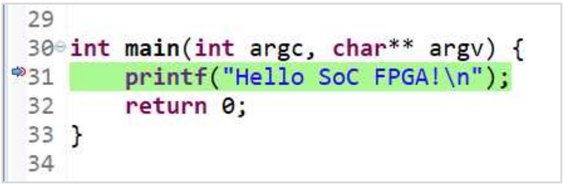
____ 13. Double-click the left margin of the source code and the debugger will set a breakpoint there, as indicated by the red dot ●.



```

int main(int argc, char** argv) {
    printf("Hello SoC FPGA!\n");
    return 0;
}

```

Figure 6-43. Setting breakpoints

____ 14. ***Continue*** button ▶ (or F8) to run the application and stop at the breakpoint.



```

int main(int argc, char** argv) {
    printf("Hello SoC FPGA!\n");
    return 0;
}

```

Figure 6-44. Stop at breakpoint

____ 15. Double-click the breakpoint indicated by a red dot ● in the left margin of the source code. To cancel the breakpoint setting, double-click the breakpoint indicated by a red dot.

____ 16. ***Step Source Line*** button (or F5) advances the execution code one line.

____ 17. You can add various views useful for debugging by selecting "***Window***" menu => "***Show View***" => "***Other***". Select the view you want to display from the items listed under "***ARM Debugger***" and click [***Open***].

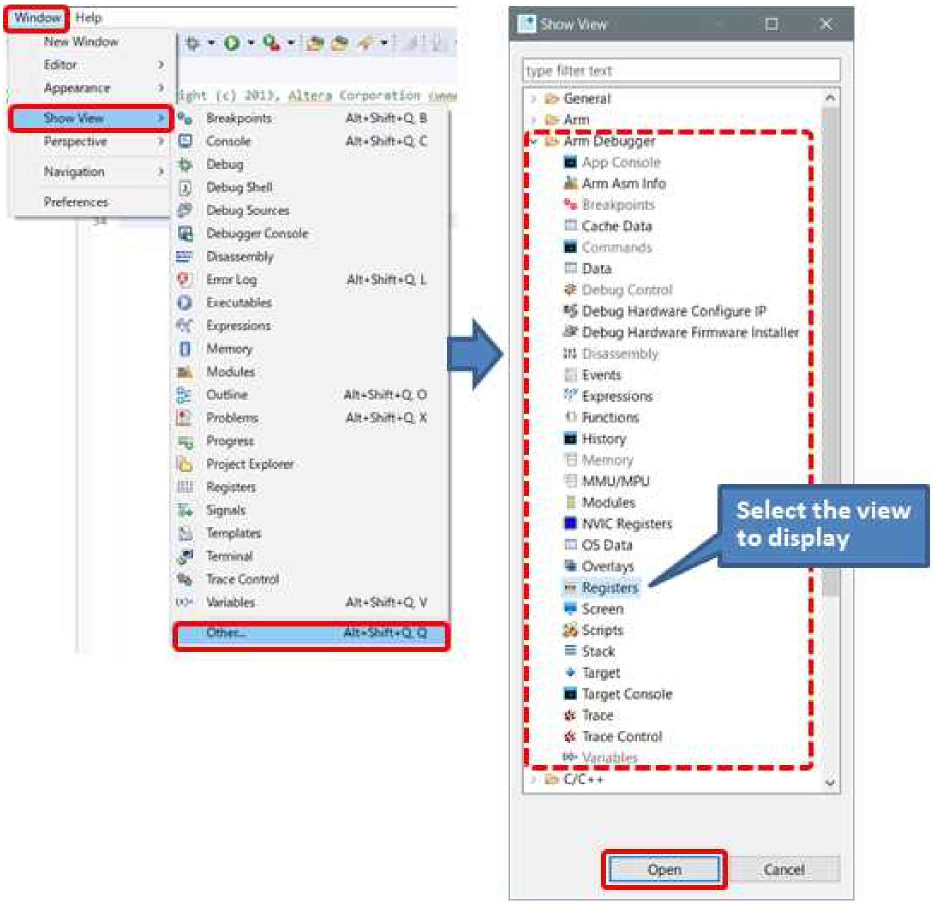
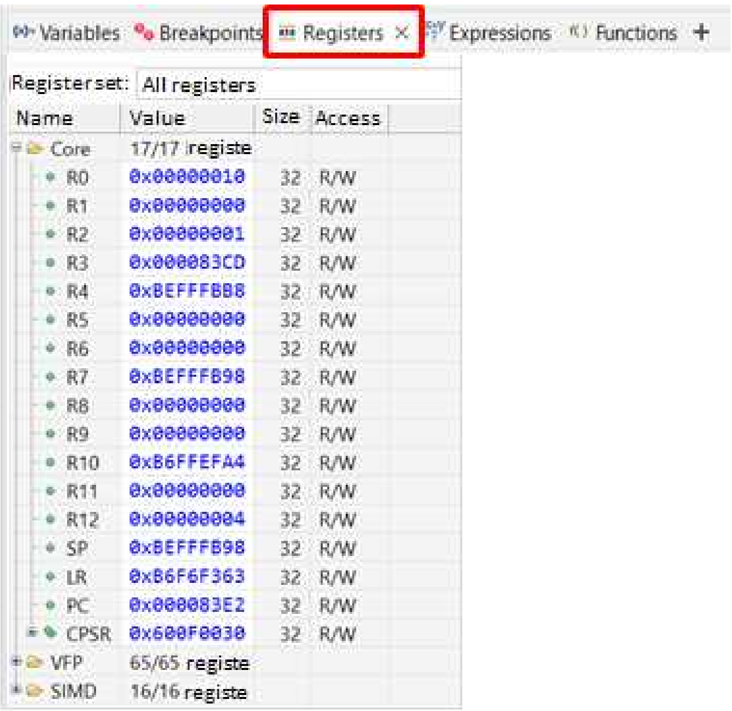


Figure 6-45. Adding a view

____ 18. The ***Registers*** view displays the contents of the target register. You can also change the value of a writable register.

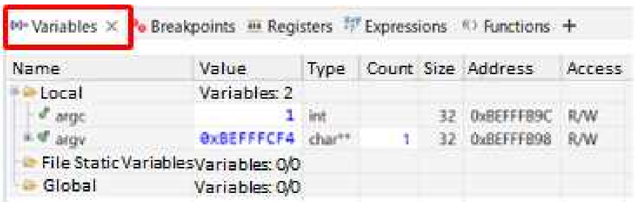


Registerset: All registers

Name	Value	Size	Access
Core	17/17 registe		
R0	0x00000010	32	R/W
R1	0x00000000	32	R/W
R2	0x00000001	32	R/W
R3	0x000083CD	32	R/W
R4	0xBEFFFBB8	32	R/W
R5	0x00000000	32	R/W
R6	0x00000000	32	R/W
R7	0xBEFFFB98	32	R/W
R8	0x00000000	32	R/W
R9	0x00000000	32	R/W
R10	0xB6FFEFA4	32	R/W
R11	0x00000000	32	R/W
R12	0x00000004	32	R/W
SP	0xBEFFFB98	32	R/W
LR	0xB6F6F363	32	R/W
PC	0x000083E2	32	R/W
CPSR	0x600F0030	32	R/W
VFP	65/65 registe		
SIMD	16/16 registe		

Figure 6-46. Registers view

____ 19. The ***Variables*** view displays the contents of a variable that is currently in its valid range. You can also change the value of a variable that is currently in its valid range.

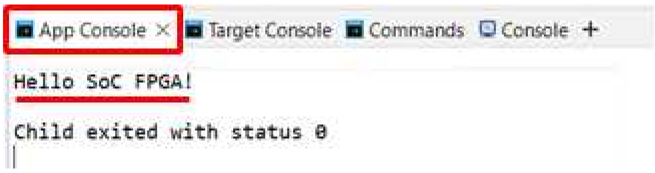


Name	Value	Type	Count	Size	Address	Access
Local	Variables: 2					
argc	1	int		32	0xBEFFFB9C	R/W
argv	0xBEFFFCF4	char**	1	32	0xBEFFFB98	R/W
File Static Variables	Variables: 0/0					
Global	Variables: 0/0					

Figure 6-47. Variables view

____ 20. The ***App Console*** (application console) view enables you to use the console I/O functionality provided by the semi-hosting implementation in the Arm C library. The contents of the print statement in the application are displayed.

____ 21. ***Continue*** button ▶ Press the button to continue the application and display Hello SoC FPGA! is displayed.



```
Hello SoC FPGA!

Child exited with status 0
```

Figure 6-48. App Console view

____ 22. Clicking ⇨| ***Debug from main()*** will break the program back to the beginning of the application, main.

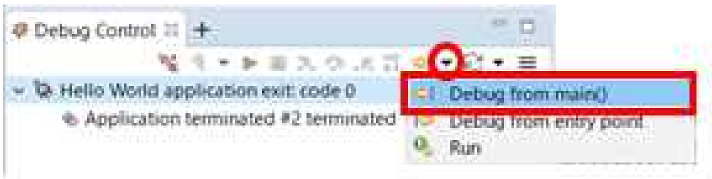

Debug Control
Hello World application exit code 0
Application terminated #2 terminated
Debug from main()
Debug from entry point
Run

Figure 6-49. Debug Control Debug from main()

____ 23. Press the ***Continue*** ▶ button again. Press the Continue button again to run the application from the top, and the ***App Console*** view Hello SoC FPGA! is displayed.

____ 24. ***Disconnect from Target*** button click to end the debugging session.

____ 25. ***Delete All Connections*** button to delete the debugging session.

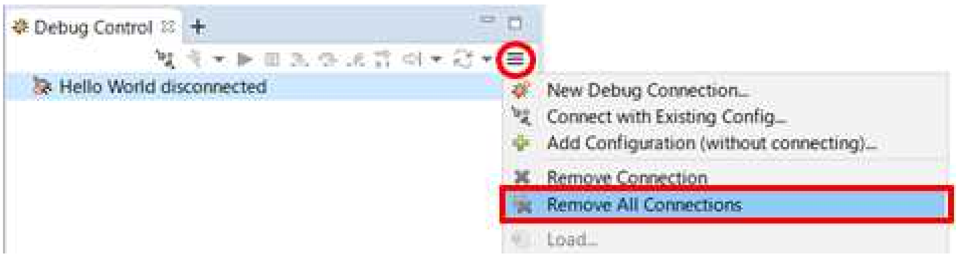


Figure 6-50. Debug Control Delete All Connections

____ 26. Right-click the button at the top right of the window and select **Reset** to restore the initial screen layout.

Congratulations! All exercises have now been completed.

7. Future reference materials

This exercise focused on learning the basic operations of the Quartus® Prime development software, which is an Altera® SoC FPGA development environment, the Platform Designer system integration tool, and the SoC EDS software development environment. If you want to improve your knowledge in the future, you can use various information sources.

Please refer to the "SoC Beginner's Guide" for the same information.

Reference:

- SoC Beginner's Guide

  Altera® SoC FPGA Summary Page (in Japanese)

  Beginner's Guide to SoC - Accessing HPS-FPGA (in Japanese)

  Beginner's Guide to SoC - How to use Preloader Generator

  Beginner's Guide to SoC – Bare Metal Application Debugging with DS-5 (in Japanese)

- SoC Information

  Macnica Altera FPGA Insights

  Macnica Altera FPGA Insights Technical Content Page

  Macnica Altera FPGA Insights FAQ Page

  Macnica Semiconductor Business: SoC FPGA-related articles and resources (in Japanese)

  Macnica Semiconductor Business: SoC FPGA-related FAQs (in Japanese)

  Macnica Semiconductor Business: Altera® FPGA-related FAQs (in Japanese)

- Description of devices and tools

  Altera® FPGAs and Programmable Devices - FPGA Documentation

- Various documents and projects that are useful for using Linux on SoC devices

  RocketBoards.org

  Altera Opensource

Revision History

Revision	Years	Overview
v20.1 r3	December 2020	Updated for SoC EDS v20.1
v20.1 r4	April 2025	Format changes, content revisions, and URL corrections

DISCLAIMER AND PRECAUTIONS

If you have obtained this document from our company, please read the following precautions before using it.

1. This document is not for sale. Resale and reproduction without permission are prohibited.
2. This document is subject to change without notice.
3. We have made every effort to prepare this document. However, if you notice any unclear points, errors, omissions, etc., please contact the distributor from whom you obtained this document.
4. Please note that we are not responsible for the effects of operation of the circuits, technologies, and programs described in this document.
5. This document is a supplementary document for using the product. When using the product, please also use the English version of the document issued by the manufacturer.