

Introduction

This guide will provide examples of how to create and debug Bare Metal projects using the ARM[®] DS-5 Altera Edition included in the *Altera[®]* SoC Embedded Design Suite (SoC EDS) User Guide.

The Altera SoC EDS is a comprehensive tool suite for embedded software development on Altera SoC devices. It includes the hardware abstraction library (HWLibs), ARM DS-5 Altera Edition (DS-5 AE), tool chain, and examples for a Bare Metal development environment.

The DS-5 AE is a useful toolset that allows you to create Bare Metal applications within the DS-5 IDE, configure, and execute it on the Altera SoC target board.

For more information, refer to the "Introduction to the SoC Embedded Design Suite" and "ARM DS-5 Altera Edition" sections in the *Altera SoC Embedded Design Suite User Guide*.

Refer to the additional guidelines, such as respective *SoC Development Kit User Guide* and *USB-Blaster User Guide* when you go through the examples in this guide.

Related Information

- Cyclone V SoC Development Kit User Guide
- Arria V SoC Development Kit User Guide
- Arria 10 SoC Development Kit User Guide
- USB Blaster Download Cable User Guide
- Altera SoC Embedded Design Suite User Guide

Bare Metal Overview

Firmware applications intended to run without an operating system (OS) are referred to as Bare Metal applications. In comparison with the user application, which is managed by an OS, a Bare Metal application can interface directly to the system hardware and run without an OS.

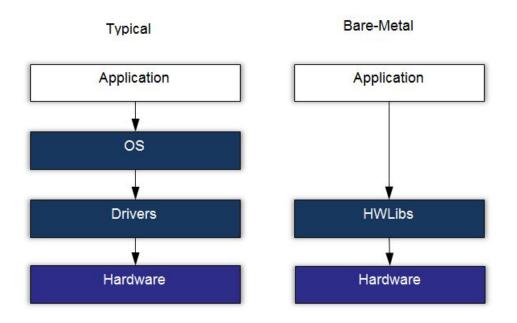
Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

*Other names and brands may be claimed as the property of others.



Figure 1: Bare Metal Application



The Bare Metal application can be invoked in one of many ways. In the following three scenarios, it is invoked after the Preloader boot stage has completed the system hardware initialization and verified the Bare Metal image or has been built as a Boot Module.

Bare Metal User Guide



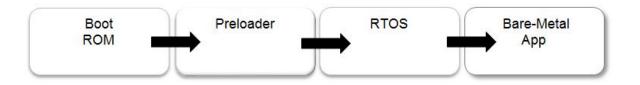
• Typical Bare Metal Application - when the Bare Metal application runs directly from the Preloader

Figure 2: Typical Bare Metal Application

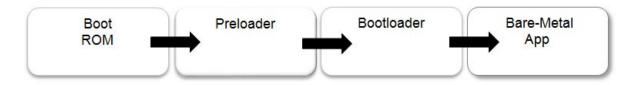


• RTOS Bare Metal Application - when the Bare Metal application runs from an RTOS

Figure 3: RTOS Bare Metal Application



• Bootloader Bare Metal Application - when the Bare Metal application runs from the Bootlooder **Figure 4: Bootloader Bare Metal Application**



The Altera SoC Embedded Design Suite (SoC EDS) User Guide provides HW abstraction Application Programming Interfaces (APIs) to simplify Bare Metal application development.

Related Information

- AN 709 HPS SoC Boot Guide For more information about boot stages.
- Minimal Preloader Example Project on page 57 For more information about the Bare Metal application built as a Boot Module.
- Altera SoC Embedded Design Suite User Guide

Prerequisites for the Bare Metal Development Environment

The following tools need to be installed before proceeding:

- Altera SoC EDS
- USB-Blaster driver •

Note: The USB-Blaster must be connected to the board and Altera SoC EDS license file must be setup correctly before proceeding.

Bare Metal User Guide

Altera Corporation



3

Bare Metal Compiler

4

The Altera SoC EDS provides the following components for a complete Bare Metal Software Development Environment:

- ARM DS-5 Altera Edition
 - ARM Compiler 5
 - GNU Compiler Collection (GCC) Bare Metal Compiler
- HWLibs
- Mkpimage tool (required by BootROM)
- Mkimage tool (required by Preloader)
- SD card image tool
- Golden Hardware Reference Design (GHRD)

For more information, refer to the "Installing the Altera SoC Embedded Design Suite" and "ARM DS-5 Altera Edition" sections in the *Altera SoC Embedded Design Suite User Guide*.

Related Information

Altera SoC Embedded Design Suite User Guide

Bare Metal Compiler

The Bare Metal Compiler that is shipped with the Altera SoC EDS is the Mentor Graphics[®] Sourcery[™] Code Bench Lite Edition. The compiler is a GCC-based **arm-altera-eabi** port. It targets the ARM processor, it assumes bare metal operation, and it uses the standard ARM embedded application binary interface (EABI) conventions. The bare metal compiler is installed as part of the Altera SoC EDS installation.

There are 2 types of bare metal compilers provided:

- ARM Compiler
- GNU Compiler Collection (GCC)

ARM compiler is supported by the Full ARM DS-5 edition (for all ARM processors) that requires a license, while the GCC is provided by DS-5 Altera Editions which is free.

For more information on the Bare Metal Compiler, refer to the "Bare Metal Compiler" chapter in the *Altera SoC Embedded Design User Guide*.

Related Information

Altera SoC Embedded Design Suite User Guide

Bare Metal Development Flow

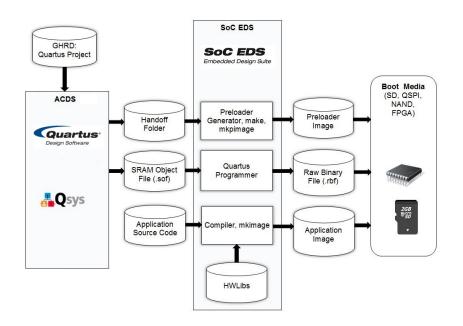
Developing SoC based Bare Metal applications involve dependencies from the FPGA generated design tools and use of the Altera SoC EDS packaged tools for building and debugging the application.

A typical Bare Metal Development flow is shown below:



Send Feedback

Figure 5: Typical Bare Metal Flow



Using DS-5 AE to Create and Manage Bare Metal Projects

Bare Metal application project can be created using DS-5 AE and compiled with ARM or GCC compiler. The application can be created to run from various target such as On-Chip RAM (OCRAM) or external memory.

The following sections will guide you through how to create, build, load and debug a simple bare metal project named "Hello World" on Cyclone V SoC Development Kit using the ARM compiler:

- To run from OCRAM
- To run from SDRAM

Note: For the GCC compiler, you can import an existing bare metal project example compiled using GCC compiler or refer to "GCC Bare-Metal Project Management" to create a simple C project manually.

Related Information

- GCC Bare-Metal Project Management
- Getting Started with Bare Metal Project Management

Simple Bare Metal Project Using On-Chip-RAM

In the following sections, you are creating, building, loading, and debugging a simple "**Hello World**" application project to run from OCRAM using ARM compiler.

Related Information

Altera SoC Embedded Design Suite User Guide

Bare Metal User Guide

Altera Corporation



Create Project

Before you begin

In Windows, go to **Windows** > **All Programs** > **ARM DS-5** > **Eclipse for DS-5** to open the ARM DS-5 tool. Select a workspace before you begin. If it is not already selected, change to C/C++ **Perspective**, located at the top right tabs of DS-5.

- 1. Create a new C project by selecting **File** > **New** > **C Project**.
- **2.** Select **Project Type** as "Hello World ANSI C Project" and **Toolchains** as "ARM Compiler 5 (DS-5 builtin)" and enter a unique project name in the **Project Name** field. For example, bare-metal-helloworld-01.

Figure 6: Creating C Project of Selected Type

🖨 C Project	
C Project Create C project of selected type	
Project name: bare-metal-hello-world-01 Use default location Location: C:\Users\mmtan\Documents\DS-5 Worksp	pace_15.1_1\bare-metal-hello-world-() Browse
Choose file system: default Project type: Choose file system: default Project type: Executable Empty Project Hello World ANSI C Project Hello World ANSI C Project Shared Library Makefile project	Toolchains: ARM Compiler 5 (DS-5 built-in) ARM Compiler 6 (DS-5 built-in) Altera Baremetal GCC GCC 4.x [arm-linux-gnueabihf] (DS-5 built-in)
Show project types and toolchains only if they are s ⑦ < <u>Back</u>	supported on the platform <u>N</u> ext > <u>Finish</u> Cancel



Note: The DS-5 is supplied with two versions of the ARM Compiler for compiling bare metal applications. ARM Compiler 5 supports all ARM architectures except ARMv8. ARM Compiler 6 supports architectures ARMv8 and ARMv7-A, as well as alpha support for architectures ARMv7-R, ARMv7-M and ARMv6-M. For Altera SoC FPGA, ARM Compiler 5 is required.

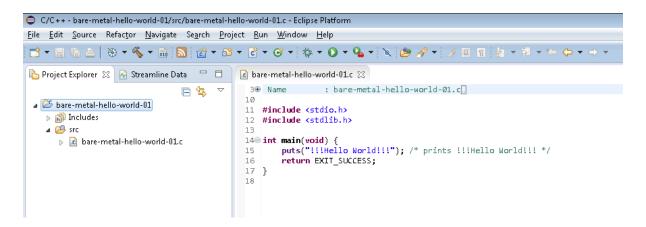
Both versions of ARM Compiler are license managed and not all editions of DS-5 include a license for it.

For any licensing information, please refer to the "Licensing" chapter in the *Altera SoC EDS User Guide*.

3. Select Finish.

The source code for bare-metal-hello-world-01.c appears in the editor view.

Figure 7: Bare Metal "Hello World - 01" Code Snippet



Create New Scatter File to Locate the Bare Metal Application in the OCRAM

Create a scatter file. Right click on the project, and select "New > Other...", then "Scatter File Editor > Scatter File".

The scatter file enables you to specify the memory map of an image to the linker using a description in a text file. It is used by the ARM compiler linker to determine the placement of the program in the target memory.

Bare Metal User Guide

7



Figure 8: Creating Scatter File

€ New	- • •
Select a wizard Create a scatter file	
<u>W</u> izards:	
type filter text	
 General C/C++ CVS DS-5 Configuration Database DS-5 Debugger DS-5 Debugger Java PyDev Remote System Explorer Scatter File Editor Scatter File Target Configuration Editor 	
(?) < <u>Back</u> <u>Next</u> > <u>Einish</u>	Cancel

2. Select the project name, bare-metal-hello-world-01, and enter the scatter file name, like scatter_OCRAM.scat



Figure 9: Scatter File Setting

🖨 New Scatter File	
Scatter File	
Create a new scatter file resource.	
Enter or select the parent folder:	
bare-metal-hello-world-01	
 Image: Image: Second Se	
File na <u>m</u> e: scatter_OCRAM.scat	
(<u>Back</u> <u>Next</u> > <u>Finish</u>	Cancel

3. Select Finish.

The new file automatically appears in the Project Explorer view.

4. In the scatter_OCRAM.scat editor view, enter the following when targeting a Cyclone V or Arria V device:

```
OCRAM 0xFFFF0000 0x10000
ÀPP_CODE +0
{
* (+RO, +RW, +ZI)
ARM_LIB_STACKHEAP 0xFFFF8000 EMPTY 0x8000 ; Application help and stack
{ }
```

Bare Metal User Guide

9

Bare Metal User Guide

Send Feedback

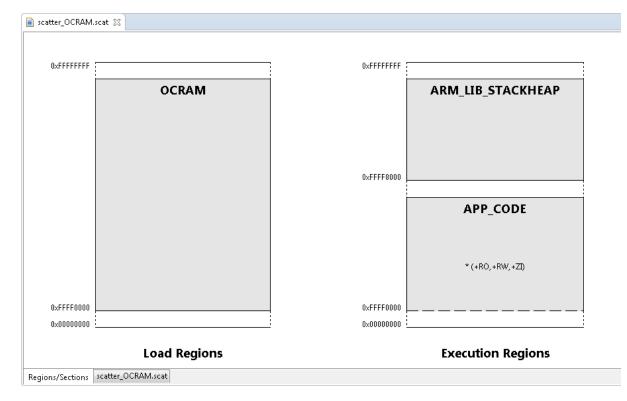
The linker script instructs the linker on how to link the application:

- Defines OCRAM base address (0xFFFF0000) and size (0x10000)
- Loads all application sections in the OCRAM
- Allocates a maximum of 32 KB (0x8000) for stack and heap starting from address 0xFFFF8000

The parameters can easily be changed for targeting Arria 10 devices, where there are 256 KB of OCRAM located at 0xFFE00000.

5. Select the **Regions/Sections** tab, located just below the scatter file to show what the memory map looks like.

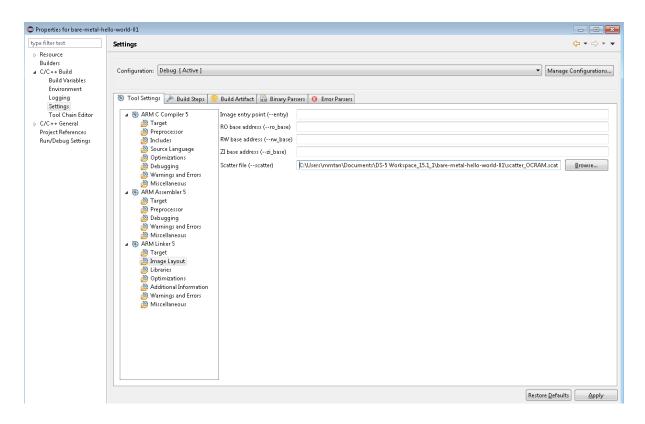
Figure 10: Scatter File Regions



- 6. Select File > Save to save the modifications.
- 7. After the scatter file has been created in the project, it needs to be associated with the project properties. Select the project name in the Project Explorer view and right-click to select **Properties**.
- 8. Go to C/C++ Build > Settings > Tool Settings > ARM Linker 5 > Image Layout.
- **9.** In the **Scatter file** (--**scatter**) text field, browse to the newly created scatter file which should now be in the project folder.

10.Select Apply and then OK.

Figure 11: Scatter File Location Setting



Build and Debug the Project

1. Then, right click on the project and select "Build Project".

The "Debug" build directory will be created and the compiled object modules will be placed there.



Figure 12: Bare Metal "Hello World - 01" Built Done

C/C++ - bare-metal-hello-world-01/scatter_OCRA	Marcat Eclines Distance		
File Edit Source Refactor Navigate Search	•		
	· 63 • 63 • 63 • 1 ☆ • 0 • 94 • 1 ≥ 129 ∧	2 • : □ □ : 2 • 5 • 5 • 5 •	Quick Access
🎦 Project Expl 😒 📓 Streamline 😑 🗖	🗈 bare-metal-hello-world-01.c 🛛 📄 scatter_OCRAN	A.scat 23	- 0 0 × × - 0
🖻 🚖 🔻			
a 😂 bare-metal-hello-world-01	0xFFFFFFFF	0xFFFFFFF	$\overline{\nabla}$
▷ 🔊 Includes ⊿ 😕 src	0.000		▲ ● ^L OCRAM
Src b are-metal-hello-world-01.c	OCRAM	ARM_LICKHEAP	▷ ● ^E APP_CC ● ^E ARM LII
🔺 🗁 Debug			• ARM_LII
> 🗁 src			
🔜 bare-metal-hello-world-01.axf 🍙 makefile			
b objects.mk			
ources.mk		0xFFFF8000	
📄 scatter_OCRAM.scat			
		APP_CODE	
		* (+RO,+RW,+ZD)	
	0×FFFF0000	0xFFFF0000	
	0×00000000	0x0000000	
	Load Regions	Execution Regions	
	•		•
	Regions/Sections scatter_OCRAM.scat		
	🖹 Problems 🧔 Tasks 📮 Console 🛛 🔲 Propertie	es 🕹 🗘 🔂 📰 👪 🖛 🖩	🗼 🛃 🖳 🕶 🔂 🖛 🗖 🗖
	CDT Build Console [bare-metal-hello-world-01]		
			*
	Finished building target: bare-metal-hello	p-world-01.axf*	
	14:00:31 Build Finished (took 6s.670ms)		-

- 2. To download and debug the program, select **Run** > **Debug Configurations**.
- 3. Right click on DS-5 Debugger and select New to create a new debug configuration.

Bare Metal User Guide



13

Figure 13: New Debug Configuration

Debug Configurations	
Create, manage, and run configurations	
(Files): No application on target details entered	
Image: System State Sta	Name: New_configuration Image: Files Debugger OS Awareness Environment Select target Select the manufacturer, board, project type and debug operation to use. Currently selected: ARM FVP (Installed with DS-5) / Cortex-A9x1 pre-configured to boot ARM Embedded Linux / Linux Application Debug / Start gdbserver and debug targe Filter platforms Altera > Arria 10 SoC > Arria 10 SoC > Arris V SoC DS-5 Debugger will connect to the FVP target (if the FVP is not currently running it will be started by DS-5 Debugger) and start a new gdbserver session application already present on the ARM Embedded Linux filesystem or on the virtual file system. Connections gdbserver (serial) Connection details determined automatically. Use serial port /dev/ttyAMA3 if starting gdbserver manually. Model parameters Imable virtual file system support File System Host mount point: Sworkspace_loc} File System Remote target mount point: Awriteable Apply.
Filter matched 19 of 19 items	мыл
0	Debug Close

4. In the Connection tab, select Altera > Cyclone V SoC (Single Core) > Bare Metal Debug > Debug Cortex-A9_0 and select "USB-Blaster" from the Target Connection pull-down menu.



Figure 14: New Debug Configuration Target Connection Setting

Debug Configurations		×
Create, manage, and run configurations		
Onfiguration for connection type 'Bare Meta	Il Debug' is not valid - Connection cannot be empty.	-
C III ★ E IF ▼ type filter text	Name: New_configuration Name: New_configuration Re- Connection Game Files State State	
C C/C++ Application C C/C++ Attach to Application C C/C++ Postmortem Debugger C C/C++ Remote Application 4 ♣ DS-5 Debugger	Select target Select the manufacturer, board, project type and debug operation to use. Currently selected: Altera / Cyclone V SoC (Single Core) / Bare Metal Debug / Debug Cortex-A9_0	
 New_configuration IronPython Run IronPython unittest Java Applet Java Applet Java Applet Julit Jython run Jython run Jython run Jython run Jython run PyDev Joingo PyDev Joingo PyDev Google App Run Python Run Python unitest Remote Java Application 	Filter platforms	*
Filter matched 19 of 19 items		wse
?	<u>D</u> ebug	Close

5. Select an available Bare Metal Debug Connection by clicking **Browse**. This will return a list of the available debug connections.

Figure 15: Connection Browser

Connection Browser	—
Connection Browser Select a target connection	
USB-BlasterII USB-1 USB-BlasterII on localhost [USB-1]	
?	Select Cancel

6. Select the hardware and then click **Select**. Change the name of the configuration from "New_Configuration" to "bare-metal-hello-world-config", and select **Apply**.



Bare Metal User Guide



15

Figure 16: Bare Metal Hello World Config Connection Settings

Debug Configurations		
Create, manage, and run configurations () [Debugger]: Debugging from a symbol, but no) symbol files defined in the Files tab	TO.
 Image: Second Second		¢
Filter matched 19 of 19 items	Apply	Revert
?		Close

7. Go to the Files tab and select the application to download by clicking "Workspace" and select the object module that was built, bare-metal-hello-world-01.axf. It should be in the Debug folder within the project. Make sure that "Load symbols" is selected, and then click on Apply.



Figure 17: Bare Metal Hello World Config Files Settings

Debug Configurations		— ×
Create, manage, and run configurations Create, edit or choose a configuration to launch	n a DS-5 debugging session.	ñ
Image: Second Secon	Name: bare-metal-hello-world-config Image: State Target Configuration Application on host to download: Stwork:space_loc:/bare-metal-hello-world-01/Debug/bare-metal-hello-world-01.axf) File System Image: Load symbols Files Load symbols from file Image: File System Workspace Image: Image: File System Image: File System	I I I I I I I I I I I I I I I I I I I
Filter matched 19 of 19 items	Apply Revert	
?	Debug Close	

8. Go to the **Debugger** tab and make sure that under **Run control**, **Debug from symbol** is set to "main".

Bare Metal User Guide



17

Figure 18: Bare Metal Hello World Config Debugger Settings

Debug Configurations		×
Create, manage, and run configurations		-
Create, edit or choose a configuration to launch	a DS-5 debugging session.	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
🗋 🗎 🗶 📄 🎲 🗸	Name: bare-metal-hello-world-config	
type filter text C C/C++ Application	🖇 Connection 🔚 Files 🏶 Debugger 🛛 🇐 OS Awareness 🕬 Arguments 🚾 Environment	
 C C/C++ Attach to Application C C/C++ Attach to Application C C/C++ Postmortem Debugger C C/C++ Remote Application D Debugger D Debugger	Run control Connect only Debug from entry point Run target initialization debugger script (ds / .py) File System Workspace Run debug initialization debugger script (ds / .py) File System Workspace Execute debugger commands Host working directory Use default Styres_loc? Paths Source search directory	
Filter matched 19 of 19 items	Apply	Revert
?	Debug	Close

9. Select **Debug** to load and debug the application.

It runs a query to change to the DS-5 Debug Perspective. Select Yes.

Figure 19: Confirm Perspective Switch

🖨 Con	firm Perspective Switch		×
\bigcirc	This launch is associated with the DS-5 Debug p	erspective.	
	Do you want to open this perspective now?		
🗖 <u>R</u> er	nember my decision		
		<u>Y</u> es	<u>N</u> o

10.It switches to the DS-5 Debug Perspective and then download and begin to run the application.

The program stops at main and waits there. In the Commands view, notice the entry point is **0xFFFF0000**, which is the start address specified in the scatter file for the on-chip RAM.



Figure 20: OCRAM Debug Stop at Main

ŝ • 🗄 🕼 🛎 📾 🕸 • 🔕 隆 • 🙋 🖋 • 🌶	[a] * [a] * (b) < (b) * (b) *				Quick Acc	ess 🔡 🖻 🖬	4
🕻 Debug C 😢 🎦 Project E 📲 Remote S 😐 🗖	🗖 Commands 🛛 🖬 History	y 🤏 Scripts 👘 📓 🕞	🚮 🤣 - 🌞 🖓	🗖 😡= Va 🔀 💁	Br ₫10 Re ^{¥+9} / _{??} Ex	f()Fu	
🖻 💐 🧏 🛎 🍇 🔿 🕶 🎜 🕶 🕨 💷 👁 🕼 1	🔩 Linked	d: bare-metal-hello-world-co	nfig 🕶			× 🖑	۲
~	loadfile "C:\Users\mmtar				nked: bare-metal-hello-v	vorld-config 🕶	
🕞 bare-metal-hello-world-config connected		Loaded section APP_CODE: S:0xFFFF0000 ~ S:0xFFFF1233 (si: Loaded section APP CODE: S:0xFFFF1234 ~ S:0xFFFF1243 (si:		Indific	Value	Type Count Size	
Gortex-A9_0 #1 stopped on breakpoint	Entry point S:0xFFFF000		FFF1243 (size 0	🖃 🗁 Locais	0 variables		
= main	Semihosting server socke	et created at port 800			Variables 0 of 0 variables		
<pre>rt entry main+0x4</pre> Semihosting enables		omatically due to semi	hosting symbol (det 🛛 🖻 🗁 Globals	0 of 0 variables		
	set debug-from main start						
	Starting target with ima	age C:\Users\mmtan\Doc	uments\DS-5 Worl	kst			
	Running from entry point			-			
	wait			=			
	Execution stopped in SVG In bare-metal-hello-worl		: S:0xFFFF00C0				
	S:0xFFFF00C0 14,16						
	Deleted temporary break			*			
C bare-metal-hello-world-config connected	Deleted temporary break	point: 1		•			
bare-metal-hello-world-config connected No OS Support	Deleted temporary break	point: 1	Sut	 Add Variable 	III	Brows	15
No OS Support	Deleted temporary breaks	point: 1 for Content Assist					
No OS Support bare-metal-hello-world-01.c 😒 📄 scatter_OCRAM.scat	Deleted temporary breaks	point: 1 for Content Assist	Memory 📒 N	Aodules 📰 Events 🔡	Outline		
No OS Support bare-metal-hello-world-01c SS scatter_OCRAM.scat 3% Name : bare-metal-hello-world-01.c[Deleted temporary breaks	for Content Assist	E Memory E M	Aodules 📰 Events 📑 inked: bare-metal-hello-w	Outline vorld-config ▼		<i>(</i> 5
No OS Support bare-metal-hello-world-01.c [2] scatter_OCRAM.scat 3@ Name : bare-metal-hello-world-01.c[] 0 1 #include <stdio.h></stdio.h>	Deleted temporary breaks	point: 1 for Content Assist Image:	E Memory E M	Modules 📰 Events 🔡 inked: bare-metal-hello-w	Outline vorld-config ▼		
No OS Support bare-metal-hello-world-01c [3] scatter_OCRAM.scat 3@ Name : bare-metal-hello-world-01.c[0 1 #include <stdio.h> 2 #include <stdib.h></stdib.h></stdio.h>	Deleted temporary breaks	for Content Assist	Hemory E M Linstruction>	Modules 📰 Events 🔐 inked: bare-metal-hello-w 10 ssembly	Outline vorld-config ▼ 10		
No OS Support bare-metal-hello-world-Dlc [S] in scatter_OCRAM.scat 30 Name : bare-metal-hello-world-Ol.c[] 1 #include <stdio.h> 2 #include <stdib.h> 3</stdib.h></stdio.h>	Deleted temporary breaks	point: 1 for Content Assist Image: state	Memory E N Struction> Opcode Disas EB0003BF BL	Modules 📰 Events 🔡 inked: bare-metal-hello-w	Outline vorld-config ▼ 10		
No OS Support bare-metal-hello-world-01c [3] scatter_OCRAM.scat 3@ Name : bare-metal-hello-world-01.c[1 #include <stdio.h> 2 #include <stdib.h> 3 4 int main(void) {</stdib.h></stdio.h>	Deleted temporary breakp	for Content Assist	Hemory E M Linstruction>	Addules Events E inked: bare-metal-hello-w 10 ssembly 	Outline vorld-config ▼ 10		
No OS Support bare-metal-hello-world-01c S3 scatter_OCRAM.scat 30 Name : bare-metal-hello-world-01.c[0 1 #include <stdio.h> 2 #include <stdib.h> 34 int main(woid) { 5 puts("!!!Hello World!!!"); /* prints !! 6 return EXT_SUCCESS;</stdib.h></stdio.h>	Deleted temporary breakp	for Content Assist	Memory Note: Not	Aodules Events E inked: bare-metal-hello-w ssembly _sys_exit ; 0xF {r4, tr} r0, {pc}+0x10 ;	Outline vorld-config • 10 FFFØFCØ ØxffffØØd4		
No OS Support bare-metal-hello-world-Dic [2] in scatter_OCRAM.scat 3# Name : bare-metal-hello-world-01.c[] 1 #include <stdlib.h> 3 4 int main(woid) { 5 puts["11Hello World111"); /* prints 11 6 return EXIT_SUCCESS; 7 }</stdlib.h>	Deleted temporary breakp	for Content Assist	Memory E N Memory E N Norther Struction> Opcode Disas EB00036F BL main PUSH E28F0008 ADR EB000006 BL	Aodules Events EE inked: bare-metal-hello-w 10 ssembly sys_exit ; 0xF {r4, lr} r0, (pc)+0x10 ; puts ; 0xFFF00	Outline vorld-config • 10 FFFØFCØ ØxffffØØd4		
No OS Support bare-metal-hello-world-Dic [2] in scatter_OCRAM.scat 3# Name : bare-metal-hello-world-01.c[] 1 #include <stdlib.h> 3 4 int main(woid) { 5 puts["11Hello World111"); /* prints 11 6 return EXIT_SUCCESS; 7 }</stdlib.h>	Deleted temporary breakp	for Content Assist	Memory E N Cpcode Disas EB0003BF BL main PUSH E28F0006 BL E3A00000 BU	Aodules Ε Events Ε inked: bare-metal-hello-w 11 ssembly _sys_exit ; 0xFF {74, Lr} rθ, {pc}+0x10 ; puts ; 0xFFFF00 rθ, θ0	Outline vorld-config • 10 FFFØFCØ ØxffffØØd4		
No OS Support bare-metal-hello-world-Dic [2] in scatter_OCRAM.scat 3# Name : bare-metal-hello-world-01.c[] 1 #include <stdlib.h> 3 4 int main(woid) { 5 puts["11Hello World111"); /* prints 11 6 return EXIT_SUCCESS; 7 }</stdlib.h>	Deleted temporary breakp	for Content Assist	Memory N N Memory N N Notruction> Opcode Disas EB0003BF BL main PUSH E28F0006 ADR E5000006 BL E3A00000 ROV E88D8010 POP	Adodules E Events EE inked: bare-metal-hello-w sys_exit ; 0xFF {rd,tr} r0,{pc}+0x10 ; puts ; 0xFFF00 r0,#0 {r4,pc}	Outline vorld-config • 10 FFFØFCØ ØxffffØØd4		
No OS Support bare-metal-hello-world-Dic [2] in scatter_OCRAM.scat 3# Name : bare-metal-hello-world-01.c[] 1 #include <stdlib.h> 3 4 int main(woid) { 5 puts["illHello World!!!"); /* prints !! 6 return EXIT_SUCCESS; 7 }</stdlib.h>	Deleted temporary breakp	for Content Assist	Memory E N Cpcode Disas EB0003BF BL main PUSH E28F0006 BL E3A00000 BU	Aodules E Events E inked: bare-metal-hello-w 11 ssembly {74, Lr} rθ, {pc}+θx10; puts; 0xFFFF00 rθ,θ0	Outline vorld-config • 10 FFFØFCØ ØxffffØØd4		
No OS Support bare-metal-hello-world-01c [23] in scatter_OCRAM.scat 340 Name : bare-metal-hello-world-01.c[] 1 #include <stdib.h> 3 44 int main(woid) { 5 puts("illHello World!!!"); /* prints !! 6 return EXIT_SUCCESS; 7 }</stdib.h>	Deleted temporary breakp	point: 1 for Content Assist □ It? Disassembly 23 Address S: 0xFFFF00C4 S: 0xFFFF00C4 S: 0xFFFF00C4 S: 0xFFFF00C4	Memory N Memory N	Aodules Events Einked: bare-metal-hello-w	Outline vorld-config • ID FFFØFCØ ØxffffØØd4 E8	A ve	
No OS Support bare-metal-hello-world-01c 23 scatter_OCRAM.scat 3@ Name : bare-metal-hello-world-01.c[1 #include <stdib.h> 2 #include <stdib.h> 4 int main(woid) { 5 puts("IIIHello WorldIII"); /* prints II</stdib.h></stdib.h>	Deleted temporary breakp	for Content Assist	Memory N N Memory N	Aodules Events Einked: bare-metal-hello-w	Outline vorld-config • ID FFFØFCØ ØxffffØØd4 E8		

11. Click the Step Over Source Line icon or press F6.

12. Then "Step" again.

The output appears on the App Console view.

Bare Metal User Guide



19

Figure 21: "App Console" Output

Debug C 🙁 🏠 Project E 📕 Remote S 🖳 🗖	 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ Commands 🔅 💼 History 🤻 	Scripts 🖳 🖍	- 🗟 🥩 🕶	*	(x)= Va 😒	9 ₀ Br <u>010</u> F	Re [¥] ? ⁹ Ex	ess 🕴 🖻	🔤 (
a 💘 💘 🗶 🙀 🛋 + 🖓 + 🕨 🔈 👁 🖉	📩 🤤 Linked: bar	e-metal-hello-world-co	nfig 🔻					×	S.
7	Starting target with image C:\Users\mmtan\Documents\DS-5 Works		-5 Works; ^	🔄 Linked: bare-metal-hello-world-config 🕶					
k bare-metal-hello-world-config connected	wait	unning from entry point ait			Nam 🕞 🍋 Locals	-	Value 0 variables	Type Count	it Size
Cortex-A9_0 #1 stopped	Execution stopped in SVC mo		: S:0xFFF	FØØCØ	E Cocais				
≡ main+0xC	In bare-metal-hello-world-0 S:0xFFFF00C0 14,16 int n				E C Globals		of 0 variables of 0 variables		
	Deleted temporary breakpoin wait next Execution stopped in SVC mo S:0xFFF00C4 15,2 puts(' wait next Execution stopped in SVC mo S:0xFFFF00CC 16,2 retur	de at S:0xFFFF00C4 '!!!Hello World!!! de at S:0xFFFF00CC	"); /* pr	ints lllt					
k bare-metal-hello-world-config connected	·			•	•	III			
No OS Support	Command: Press (Ctrl+Space) for C	ontent Assist		Submit	Add Variable			Ĩ	Brow
<pre>DW Name : bare-metal-hello-world-01.c 0 #include <stdio.h> 2 #include <stdib.h> 3 #int main(void) { 5 puts("!!!Hello World!!!"); /* prints ! 6 return EXIT_SUCCESS;</stdib.h></stdio.h></pre>		Address S:0xFFFF00C4 S:0xFFFF00C4 S:0xFFFF00C5	EB000006 EBA00000	Disassembl ADR r BL p MOV r		100 ; ØxffffØØ	, 		
7 }		S:0xFFFF00D4 S:0xFFFF00D8 S:0xFFFF00DC S:0xFFFF00E0	6F6C6C65 726F5720	DCD Ø	x48212121 x6F6C6C65 x726F5720 x2121646C				
		🖬 App Console 🖂	🔟 Target		-	-world-confic			7 🗆
				Tinked:	bare-metal-nelit	-wona-conn <u>e</u>	, •		

13.Select **Continue** to exit out of the application.

14.Create a breakpoint by double clicking on the left margin of the bare-metal-hello-world-0.1.c Source view. If the Breakpoints view is not shown, then open it by selecting: View > Show View > Breakpoints.



Figure 22: Debug Breakpoint Setup

🝷 🔚 🕼 🖢 🗟 🐐 🕈 🔽 💁 📲 🥔 🖋 🔽] ½ ▼ 🖓 ▼ 🏷 🗢 ▼ ⇒ ▼				Quick Access	😰 🔂 🕯
Debug C 😢 🍋 Project E 🚚 Remote S 😐 🗖	🗖 Commands 🔀 💼 History 🖣	🌾 Scripts 🛛 🖳 🕞 🚮	🤣 - 🌞 😑	🗖 🚺 🖬 Va 🔀 💁 Br	010 Re *** Ex f() Fi	• E
🖄 💥 💥 🐳 🔿 + 🖓 + 🕨 💷 👁e.	🔄 🔄 🔄	are-metal-hello-world-config	•			x 🖉 .
	Execution stopped in SVC m	ode at S:0xFFFF00C4		🔺 🔄 🔄	bare-metal-hello-world-co	nfig •
浓 bare-metal-hello-world-config application exit: code 0	S:0xFFFF00C4 15,2 puts	("!!!Hello World!!!");	/* prints !!	1) Name	Value Type C	ount Size L
Cortex-A9_0 #1 application exit	wait next			🖙 🗁 Locals	0 variables	
= main+0xC	Execution stopped in SVC m	ode at S:0xFFFF00CC		🖶 🗁 File Static Variab		
	S:0xFFFF00CC 16,2 retu	rn EXIT_SUCCESS;		🕀 🗁 Globals	0 of 0 variables	
	wait continue					
	NORMAL TERMINATION					
	Execution stopped in SVC m	ode at S:0xFFFF0FC8 due	e to applicat	i		
	In _sys_exit (no debug inf					
	S:0xFFFF0FC8 SVC #0 break -p "C:/Users/mmtan/D	x123456	a 15 1 1/bana	E		
	Breakpoint 2 at S:0xFFFF00		2_15.1_1/bare			
		lo-world-01.c, line 15		*		_
🔯 bare-metal-hello-world-co application exit: code 0					m	
No OS Support	Command: Press (Ctrl+Space) for	Content Assist	Subr	mit Add Variable		Brows
bare-metal-hello-world-01.c 🔀 📄 scatter_OCRAM.scat] 🕴 Disassembly 🖾 🔚	Memory 🚪 Mi	odules 🔚 Events 👫 Outlin	ie 🔗	~ - 6
8⊕ Name : bare-metal-hello-world-01.c.	A		🔁 Lin	ked: bare-metal-hello-world-	config 🕶	
<pre>3⊕ Name : bare-metal-hello-world-01.c. 2 4 #include <stdio.h></stdio.h></pre>	· · · · · · · · · · · · · · · · · · ·	🖹 🚽 🗸 Next Instru		ked: bare-metal-hello-world- 100	config •	
2 L #include <stdio.h> 2 #include <stdlib.h></stdlib.h></stdio.h>	· · · · · · · · · · · · · · · · · · ·			100	contig •	
<pre>// #include <stdio.h> // #include <stdlib.h> // #include <stdlib.h #include="" <stdli<="" <stdlib.h="" td=""><td>· · · · · · · · · · · · · · · · · · ·</td><td>Address Op S:0xFFFF0FC0 E59</td><td>iction> pcode Disass PF1008 LDR</td><td>100 embly r1,[pc,#8] ; [ØxFFF</td><td></td><td></td></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdio.h></pre>	· · · · · · · · · · · · · · · · · · ·	Address Op S:0xFFFF0FC0 E59	iction> pcode Disass PF1008 LDR	100 embly r1,[pc,#8] ; [ØxFFF		
) #include <stdio.h> 2 #include <stdlib.h> 3 4 6 int main(void) {</stdlib.h></stdio.h>		Address Op S:0xFFF0FC0 E59 S:0xFFF0FC4 E3A	iction > bcode Disass 0F1008 LDR 00018 MOV	100 embly r1,[pc,#8] ; [0xFFf r0,#0x18		
<pre>// #include <stdio.h> // #include <stdlib.h> // #include <stdlib.h #include="" <stdli<="" <stdlib.h="" td=""><td></td><td>Address Op S:0xFFF0FC0 E59 S:0xFFF0FC4 E34 S:0xFFF0FC8</td><td>iction > bcode Disass DF1008 LDR NOV SVC</td><td>100 embly r1,[pc,#8];[0xFFf r0,#0x18 #0x123456</td><td>FØFDØ] = Øx20026</td><td></td></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdio.h></pre>		Address Op S:0xFFF0FC0 E59 S:0xFFF0FC4 E34 S:0xFFF0FC8	iction > bcode Disass DF1008 LDR NOV SVC	100 embly r1,[pc,#8];[0xFFf r0,#0x18 #0x123456	FØFDØ] = Øx20026	
<pre>/ #include <stdio.h> / #include <stdib.h> // #include <stdib.h> // for main(void) { // puts("!!!Hello World!!!"); /* prints !! // return EXIT_SUCCESS; // }</stdib.h></stdib.h></stdio.h></pre>		Address Op S:0xFFF0FC0 E59 S:0xFFF0FC4 E3A	iction> bcode Disass DF1008 LDR MOV SVC SFFFFE B	100 embly r1,[pc,#8] ; [0xFFf r0,#0x18	FØFDØ] = Øx20026	
9 1 #include <stdio.h> 2 #include <stdlib.h> 3 i≕int main(void) { 5 preturn EXT_SUCCESS; 7 return EXT_SUCCESS;</stdlib.h></stdio.h>		Address Op S:0xFFFF0FC0 E55 S:0xFFFF0FC4 E34 S:0xFFFF0FC6 S:0xFFFF0FC6 S:0xFFFF0FC0 0000	Inction> Disease F1008 LDR MOV SVC FFFFE B D20026 DCD Heep	100 embly r1,[pc,#8] ; [0xFFi r0,#0x18 #0x123456 	FØFDØ] = Øx20026	
<pre>/ #include <stdio.h> / #include <stdib.h> // #include <stdib.h> // for main(void) { // puts("!!!Hello World!!!"); /* prints !! // return EXIT_SUCCESS; // }</stdib.h></stdib.h></stdio.h></pre>		Address Op S:0xFFFF0FC0 ESS S:0xFFFF0FC4 ESA S:0xFFFF0FC6 S:0xFFFF0FC6 S:0xFFFF0FC6 S:0xFFFF0FC6 S:0xFFFF0FC6 S:0xFFFF0FC6 S:0xFFFF0FC6 S:0xFFFF0FC6	Iction> Socode Disass F1008 LDR NOV SVC FFFFE B D20026 DCD Heap N03000 Mov	100 embly r1,[pc,#8];[0xFFf r0,#0x18 #0x123456 _sys_exit+12;0xFF 0x00020026 Providetemory r3,r0	FØFDØ] = Øx20026	
<pre>/ #include <stdio.h> / #include <stdib.h> // #include <stdib.h> // for main(void) { // puts("!!!Hello World!!!"); /* prints !! // return EXIT_SUCCESS; // }</stdib.h></stdib.h></stdio.h></pre>		Address Op S:0xFFFF0FC0 E55 S:0xFFFF0FC4 E34 S:0xFFFF0FC6 S:0xFFFF0FC6 S:0xFFFF0FC0 0000	Iction> Socode Disass F1008 LDR NOV SVC FFFFE B D20026 DCD Heap N03000 Mov	100 embly r1,[pc,#8] ; [0xFFi r0,#0x18 #0x123456 	FØFDØ] = Øx20026	
<pre>/ #include <stdio.h> / #include <stdib.h> // #include <stdib.h> // for main(void) { // puts("!!!Hello World!!!"); /* prints !! // return EXIT_SUCCESS; // }</stdib.h></stdib.h></stdio.h></pre>		Address Op S:0xFFFF0C0 ESS S:0xFFFF0C0 ESS S:0xFFFF0FC8 S:0xFFFF0FC8 S:0xFFFF0FC0 000 S:0xFFFF0FC0 000 S:0xFFFF0FD0 ESS	Interference Disass Accode Disass Br1008 LDR MOV SVC FFFFE B MOV DCD MOV LDR MOV LDR MOV LDR MOV LDR	100 embly r1,[pc,#8];[0xFFI r0,#0x13456 #0x123456 _sys_exit+12;0xFF 0x00020026 ProvideHemory r3,r0 r0,[r0,#4]	FØFDØ] = Øx20026	
<pre>/ #include <stdio.h> / #include <stdib.h> // #include <stdib.h> // for main(void) { // puts("!!!Hello World!!!"); /* prints !! // return EXIT_SUCCESS; // }</stdib.h></stdib.h></stdio.h></pre>		Address Op S:0xFFF6C0 ESS S:0xFFF6C4 ESG S:0xFFF6C4 ESG S:0xFFF6C4 ESG S:0xFFF6C4 ESG S:0xFFF6C4 ESG S:0xFFF6C6 ESG S:0xFFF6C6 ESG S:0xFFF6C6 ESG	International Construction State Sta	100 r1,[pc,#8];[@xFFI r0,#0x123456 #0x123456 gys_exit+12; 0xFF 0x00020026 ProvideHemory r3,r0 r0,[r0,#4] € ErrorLog	FF0FD0] = 0x20026 FF0FCC	~ - (
<pre>/ #include <stdio.h> / #include <stdib.h> // #include <stdib.h> // for main(void) { // puts("!!!Hello World!!!"); /* prints !! // return EXIT_SUCCESS; // }</stdib.h></stdib.h></stdio.h></pre>		Address Op S:0xFFFF0C0 ESS S:0xFFFF0C0 ESS S:0xFFFF0FC8 S:0xFFFF0FC8 S:0xFFFF0FC0 000 S:0xFFFF0FC0 000 S:0xFFFF0FD0 ESS	International Construction State Sta	100 embly r1,[pc,#8];[0xFFI r0,#0x13456 #0x123456 _sys_exit+12;0xFF 0x00020026 ProvideHemory r3,r0 r0,[r0,#4]	FF0FD0] = 0x20026 FF0FCC	⊽ □ [

Preloader

The Preloader is a boot-strap program that configures some components of the board including the memory controller during start up. It normally runs automatically if it were burned into one of the Flash devices on the board.

For testing purposes or before the Preloader has been burned into Flash, the Preloader can be run by downloading and executing it from the DS-5.

Loading and Running the Default Preloader

There are a few options for running the Preloader.

- It can be ran through a DS-5 command script.
- It can be imported into a DS-5 project and launched similarly to any other Bare Metal application.
- It can be launched by the DS-5 Run Control (Debug Control) independent from the DS-5 projects.

This section demonstrates how the preloader is launched independently from the DS-5 project for Cyclone V SoC Development Kit. To optionally import the Preloader into a DS-5 project, refer to the "Importing Preloader into a DS-5 Project" section.

This process uses the **Run Configuration (Debug Configuration)** of DS-5 to download and execute the Preloader.

- 1. To begin, create a new **Debug Configuration**. **Run** > **Debug Configurations**.
- 2. Select DS-5 Debugger and click on New icon, which is located just above the "type filter text" field.

Altera Corporation

Bare Metal User Guide



3. On the Connection tab, select Altera > Cyclone V SoC (Single Core) > Bare Metal Debug > Debug Cortex-A9 and then select "USB-Blaster" from the Target Connection pull-down menu.

Figure 23: RunPreloaderDefault Target Connection Setting

Debug Configurations	X
Create, manage, and run configuratio ② Configuration for connection type 'Bare	ns Metal Debug' is not valid - Connection cannot be empty.
	Name: RunPreloaderDefault
type filter text © C/C++ Application © C/C++ Attach to Application © C/C++ Postmortem Debugger © C/C++ Remote Application © DS-5 Debugger Image: DS-5 Debugger <td>Connection Files Convection Connection Connection Files Connection Connection Files Connection Connection Files Fi</td>	Connection Files Convection Connection Connection Files Connection Connection Files Connection Connection Files Fi
?	Debug Close

4. To select the physical debug connection, after the **Connections** field, click on **Browse** to select the specific Debug Hardware connection.

Note: If there is only one debugger connected, then only one will show up in the list (as shown).



UG-01165 2015.11.30

Figure 24: Connection Browser

Connection Browser	—
Connection Browser	
Select a target connection	
USB-BlasterII USB-1	
USB-BlasterII on localhost [USB-1]	
?	Select Cancel

Figure 25: RunPreloaderDefault Connection Settings

Debug Configurations	A CARLON AND A C	×
Create, manage, and run configurations [Debugger]: Debugging from a symbol, but no sym 	bol files defined in the Files tab	Č.
	Name: RunPreloaderDefault	
type filter text	🐢 Connection 🖉 Files 👫 Debugger 🏀 OS Awareness 🕬 Arguments 🖉 Environment	
© C/C++ Application © C/C++ Attach to Application © C/C++ Postmortem Debugger © C/C++ Remote Application	Select target Select the manufacturer, board, project type and debug operation to use. Currently selected: Altera / Cyclone V SoC (Single Core) / Bare Metal Debug / Debug Cortex-A9_0	
A 4 DS-5 Debugger bare-metal-hello-world-config	Filter platforms	
A New_configuration		*
a ^I IronPython Run	▷ Arria 10 SoC ▷ Arria V SoC	-
a [¥] IronPython unittest ፼ Java Applet	Arma V Soc > Cyclone V SoC (Dual Core)	=
Java Applet	∠ Cyclone V SoC (Single Core)	
Ju JUnit	Bare Metal Debug	
a ⁷ Jython run	Debug Cortex-A9_0	
nittest 🥙 Jython unittest	Linux Application Debug	
Launch Group	Linux Kernel and/or Device Driver Debug Dual Arria V SoC (2 Dual Core SoCs)	
PyDev Django	blad star a soc (z bad core socs) blad cyclone V Soc (2 bad core socs)	
為 PyDev Google App Run ┛ Python Run	Advard Application Debug	
Python unittest	ARM Development Boards	
Remote Java Application	> ARM FVP	-
	- ADM installation (in the DC S)	÷
	Target Connection USB-Blaster DTSL Options Edit Configure USB-Blaster trace or other target options. Using "default" configuration options	
	DS-5 Debugger will connect to an Altera USB-Blaster to debug a bare metal application.	-
	Connections	
	Bare Metal Debug Connection USB-BlasterII on localhost [USB-1]:USB-BlasterII USB-1	
Filter matched 20 of 20 items	Apply Rey	ert
(?)	Cebug	lose

5. Select the Files tab and use "File System..." to browse to the Preloader image ("u-boot-spl").

23

The Preloader image can typically be found in the following location: <*SoC EDS installation folder*>\ embedded\examples\hardware\cv_soc_devkit_ghrd\software\preloader\uboot-socfpga\spl\u-boot-spl.

Note: Uncheck the "Load symbols" option.

Figure 26: RunPreloaderDefault Files Settings

Name: RunPreloaderDefault	
🦇 Connection 🔚 Files 🛛 🏘 Debugger 🆓 OS Awareness 🕺 Arguments 🚾 Environment	
Target Configuration Application on host to download: C:\altera\15.1\embedded\examples\hardware\cv_soc_devkit_ghrd\software\preloader\uboot-socfpga\spl\u-boot-spl File System Workspace Load symbols from file • File System Workspace	
	Ŧ

- **6.** On the **Debugger** tab, select "Debug from Entry Point" under **Run Control** and select **Execute debugger commands** to enable the following commands:
 - Run
 - Pause 1s
 - Interrupt
 - Quit



Figure 27: RunPreloaderDefault Debugger Settings

Connection 🔚 Files 🏘 Debugger 👋 OS Awareness 😡 Arguments 🚾 Enviro	onment
Run control	
🗇 Connect only 💿 Debug from entry point 💿 Debug from symbol 🛛 🔤 main	
🗌 Run target initialization debugger script (.ds / .py)	
	File System Workspace
Run debug initialization debugger script (.ds / .py)	
	File System Workspace
Execute debugger commands	
Pause 1s Interrupt Quit	*
Host working directory ✔ Use default	
\${workspace_loc}	File System Workspace
Paths Source search directory	
-	

7. Select Apply, Debug, and then "Yes" to switch to the DS-5 Debug perspective, if queried.

Importing Preloader into a DS-5 Project

Instead of loading and running the default preloader directly, as described in the "Loading and Running the Default Preloader" chapter, you can import the Preloader executable (**u-boot-spl**) into your project within your workspace and browse to it by clicking on **Workspace**, as shown in the following sections. This isolates your project from any changes that you may make to the default preloader.

Import Preloader

Before you begin

To do so, first you have to import the Preloader into your project.

 Right click on the project and select Import. Then select General > File System and click Next when you are done.

Bare Metal User Guide



25

Figure 28: Import File System

🖨 Import	- • •
Select Import resources from the local file system into an existing project.	Ľ
Select an import source:	
type filter text	
 ✓ General ✓ Archive File ✓ Existing Projects into Workspace ✓ File System ✓ Preferences ✓ C/C++ ✓ CVS ✓ Install ✓ Remote Systems ✓ Run/Debug ✓ Scatter File Editor ✓ Target Configuration Editor ✓ Team 	
(?) < <u>Back</u> <u>Next > Einish</u>	Cancel

- 2. Browse to the Preloader executable (u-boot-spl) or enter the full path to the directory, for example: <*SoC EDS installation path*>\embedded\examples\hardware\cv_soc_devkit_ghrd\ software\preloader\uboot-socfpga\spl.
- 3. Check the box next to the file name and then select **Finish**.



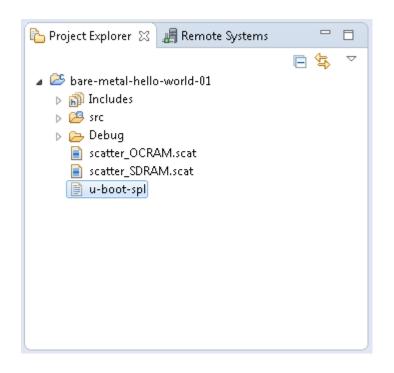
Figure 29: Import u-boot-spl

🖨 Import	
File system Import resources from the local file system.	
From directory: C:\altera\15.1\embedded\examples\hardv	ware\cv_soc_devkit_ghrd\sof 👻 <u>Br</u> owse
▶ ■	 igitignore Makefile u-boot-spl u-boot-spl.bin u-boot-spl.lds u-boot-spl.map u-boot.lst
Filter Types Select All Deselect All	
Into fo <u>l</u> der: bare-metal-hello-world-01 Options	Bro <u>w</u> se
Overwrite existing resources without warning	
<u>Create top-level folder</u> <u>Advanced >></u>	
? < <u>B</u> ack	Next > Einish Cancel

4. Make sure the newly imported file is shown in the Project Explorer.



Figure 30: u-boot-spl file in Project Explorer



Create New Debug Configurations and Debug Preloader

1. Select "USB-Blaster" from the Target Connection pull-down menu and then select an available Bare Metal Debug Connection by clicking **Browse**.

27

Figure 31: Run Preloader Hello Connection Settings

:: Run Preloader Hello	
ionnection 🔪 🔚 Files 👫 Debugger 🍓 OS Awareness 🕺 Arguments 🚾 Environment	
lect target	
lect the manufacturer, board, project type and debug operation to use. Currently selected:	
ltera / Cyclone V SoC (Single Core) / Bare Metal Debug / Debug Cortex-A9_0	
Filter platforms	
⊿ Altera	*
⊳ Arria 10 SoC	=
⊳ Arria V SoC > Coulors V SoC (Dual Care)	
▶ Cyclone V SoC (Dual Core) ▲ Cyclone V SoC (Single Core)	
∠ Bare Metal Debug	
Debug Cortex-A9_0	
Linux Application Debug	
▷ Linux Kernel and/or Device Driver Debug ▷ Dual Arria V SoC (2 Dual Core SoCs)	
 Dual Arria V SUC (2 Dual Core SUCS) Dual Cyclone V SoC (2 Dual Core SuCs) 	
· · · · · · · · · · · · · · · · · · ·	•
rget Connection USB-Blaster 👻	
TSL Options Edit Configure USB-Blaster trace or other target options. Using "default" configurat	tion options
S-5 Debugger will connect to an Altera USB-Blaster to debug a bare metal application.	
onnections	
re Metal Debug Connection USB-BlasterII on localhost [USB-1]:USB-BlasterII USB-1	Browse

- 2. On the Files tab, select Workspace and browse to the project and then the Preloader file "u-boot-spl".
- 3. Uncheck the check box next to "Load Symbols".



Figure 32: Run Preloader Hello Files Settings

Vame	e: Run Preloader Hello
⊲⊳ (Connection 🔚 Files 🛛 🏘 Debugger 🆓 OS Awareness 🕅 Arguments 🔤 Environment
l n	Target Configuration
ł	Application on host to download:
	C:\altera\15.1\embedded\examples\hardware\cv_soc_devkit_ghrd\software\preloader\uboot-socfpga\spl\u-boot-spl
	File System Workspace Load symbols
F	Files
	Load symbols from file 🔹
ſ	
U	File System Workspace
(+

- **4.** On the **Debugger** tab, select "Debug from Entry Point" under **Run Control** and select **Execute debugger commands** to enable the following commands:
 - Run
 - Pause 1s
 - Interrupt
 - Quit



Bare Metal User Guide

Send Feedback

Figure 33: Run Preloader Hello Debugger Settings

Connection 🔚 Files 👫 Debugger 🛛 🆓 OS Awareness 🕺 Arguments 🚾 Environ	nment
Run control	
Connect only Debug from entry point Debug from symbol main	
Run target initialization debugger script (.ds / .py)	
	File System Workspace
Run debug initialization debugger script (.ds / .py)	
	File System Workspace
Execute debugger commands	
Run Pause 1s Interrupt Quit	•
Host working directory V Use default	
Section of the sectio	File System Workspace
Paths	
Source search directory	

5. Select Apply and then Debug, then "Yes" to switch to the DS-5 Debug perspective, if queried. Figure 34: Confirm Perspective Switch

Conf	firm Perspective Switch		x	
\bigcirc	This launch is associated with the DS-5 Debug perspective.			
•	Do you want to open this perspective now?			
🗖 <u>R</u> en	nember my decision			
		<u>Y</u> es	<u>N</u> o	

6. DS-5 AE should load and run the Preloader in the on-chip RAM, similarly to how the simple Bare Metal example is ran. This initializes the SDRAM memory controller and then stop and wait. The display should look something like this:

Figure 35: Run Preloader Hello Debug View

🛊 Debug Control 🙁 🎦 Project Explorer 🛛 📕 Remote Systems	😑 🗖 🗖 Commar	ds 🕄 📷 History 🦓 Scripts	🖳 🗟 🔻 🆑 🗖
□ 🕱 💥 💥 🕪 - 음 - ▶ 🗉 ౫ ۞ 止言 🔍	14 6 91 V	🚖 Linked: Run Prelo	oader Hello 🕶
Run Preloader Hello disconnected bare-metal-hello-world-config disconnected		<pre>stopped in SVC mode at S:0xFFFF49E4 "C:\altera\15.1\embedded\ds-5\sw\debugger 9E4 B {pc}; 0xffff49e4</pre>	r\configdb\Scripts\altera_target_check.py"
	No SYSID	registers could be found. Has a peripheral	l description file been supplied?
	Loaded se Loaded se Loaded se Entry poi	<pre>altera\15.1\embedded\examples\hardware\cv tion .text: 5:0xFFFF0000 ~ 5:0xFFFF7IF3 tion .rodata: 5:0xFFFF7IF4 ~ 5:0xFFFF914 tion .data: 5:0xFFFF9150 ~ 5:0xFFFFA023 (t 5:0xFFFF0000 -from *\$ENTRYPOINT</pre>) (size Øx1F58)
	Starting wait	arget with no symbolic information loaded stopped in SVC mode at S:0xFFFF0000 300 B {pc}+0x70 ; 0xffff0070	3
	Starting Pause 1s Interrupt Quit	arget with no symbolic information loaded	a
	S:0xFFF4	stopped in SVC mode at S:0xFFFF49E4 3E4 B {pc} ; 0xffff49e4 sed from stopped target Altera - Cyclone Y	/ SoC (Single Core) on TCP:localhost
Run Preloader Hello disconnected No OS Support	Command:	Press (Ctrl+Space) for Content Assist	Subr

This Run Control can now be used to launch the Preloader.

Although this method associates the Preloader image with a specific DS-5 project, it can still be used with other projects. However, it is important to remember which project contains the Preloader image.

It should load and run the Preloader in the on-chip RAM, similarly to how the simple Bare Metal example was run. This configures the SDRAM memory controller and then stops.

After that, you can download and run the "**Hello World**" example in the SDRAM memory. The same Debug Configuration created earlier for the **Hello World** example can be used.

Download and Debug "Hello World" Example in the SDRAM Memory

 Select Run > Debug Configurations. Select the same debug configuration.



Figure 36: Bare Metal "Hello World" Config Debugger

Debug Configurations		×
Create, manage, and run configurations		Ť.
	Name: bare-metal-hello-world-config Connection Files * Debugger OS Awareness * Arguments Fervironment Select target Select target Select target Select target Altera / Cyclone V SoC (Single Core) / Bare Metal Debug / Debug Cortex-A9_0 Filter platforms Altera Altera Arria 10 SoC Arria V SoC Cyclone V SoC (Dual Core) Cyclone V SoC (Single Core) Bare Metal Debug Debug Cortex-A9_0 Linux Application Debug Dista Cortey Contection USB-Blaster Target Connection USB-Blaster To debug a bare metal application. Connections Bare Metal Debug Connection USB-BlasterI on localhost [USB-1]:USB-BlasterII USB-1	Browse

2. Select **Debug** to launch.

Figure 37: Run "Hello World" from SDRAM Stops at Main

DS-5 Debug - bare-metal-hello-world-01/src/bare-metal-hello-world-01.c - Eclipse Platform			×					
File Edit Source Refactor Navigate Search Project Run Window Help								
🏘 Debug Control 🕴 🏠 Project Explorer 📲 Remote Systems 📃 🗖	🔳 Commands 🕴 📷 History 🧠 Scripts	🖳 🖺 🚮 🍫 🔻 🛱 🗖 🔲 😡- V 😒 🍡 👘						
📄 🖄 💥 🍇 🌒 = 옷 = 🕒 🔊 👁 😥 👘 🖉 또 한 개 🔻	🔄 Linked: bare-metal-hello-world-conf	fig • 🗶 🦑	$\overline{\nabla}$					
🔭 Run Preloader Hello disconnected	Connected to stopped target Altera - Cyclone V SoC (Single Con cd "C:\Users\mmmtan\Documents\DS-5 Workspace 15.1 1"	re) on TCP:localhost iked: bare-metal-hello-we	/orld-					
kare-metal-hello-world-config connected Contex-A9 0 #1 stopped on breakpoint	Working directory "C:\Users\mmtan\Documents\DS-5 Workspace_15.	.1_1" Name						
a 1 g cortex-M9_0 +1 stopped on breakpoint ≡ main	Execution stopped in SVC mode at S:0xFFFF49E4 S:0xFFFF49E4 B {pc}; 0xffff49e4	Locals Brie Static Variable	0.0.0					
Intermediate and a second	<pre>source /v "C:\altera\15.1\embedded\ds-5\sw\debugger\configdb\!</pre>		0 c					
	No SYSID registers could be found. Has a peripheral description	on file been supplied?						
	<pre>loadfile "C:\Users\mmtan\Documents\D5-5 Workspace_15.1_\bare Loaded section APP_CODE: S:002000000 ~ S:0020001233 (size 0x Entry point S:002000000 ~ S:002001234 (size 0x Entry point S:002000000 ~ S:002001243 (size 0x Entry point S:002000000 ~ S:002001243 (size 0x et debug=from main start debug=from main start debug=from main starting target with image C:\Users\mmtan\Documents\D5-5 Works Waning from entry point Execution stopped in SVC mode at breakpoint 1: S:002000000 Ha/16 int main(void) { Deleted temporary breakpoint: 1</pre>	1234) 10) etected in image 'bare-metal-hello space_15.1_1\bare-metal-hello-worl						
k bare-metal-hello-world-config connected	• [• • • • • • • • • • • • • • • • • • •	P.					
No OS Support	Command: Press (Ctrl + Space) for Content Assist	Submit Add Variable Brows	/se					
scatter_SDRAM.scat	- 8	112 Dis 23 11 Me E Mo E Eve 15 Outl						
3® Name : bare-metal-hello-world-01.c.			_					
10		1971						
11 #include <stdio.h> 12 #include <stdlib.h></stdlib.h></stdio.h>		💼 App C 💼 Target 👰 Error L 📮 Console 💥 📟	8					
13 ⇒ 14⊖ int main(void) {			8 -					
15 puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */		No consoles to display at this time.	· .					
16 return EXIT_SUCCESS; 17 }								
18								
	*							
	,							
🔖 bare-metal-hello-world-config connected (Altera - Cyclone V SoC (Single Core))								

Bare Metal User Guide



You should notice in the Commands view that the entry point is now S:0x02000000, which is the beginning of the mapped SDRAM.

- **3.** Click on the **"Step Over Source Line"** icon or press **F6** to see the program counter progress to the next source line.
- 4. Click it again to see the "!!!Hello World!!!" message in the App Console view. If the App Console view is not currently selected, then the letters are highlighted in bold letters to

indicate that there is a message. Select the App Console view to see the output.

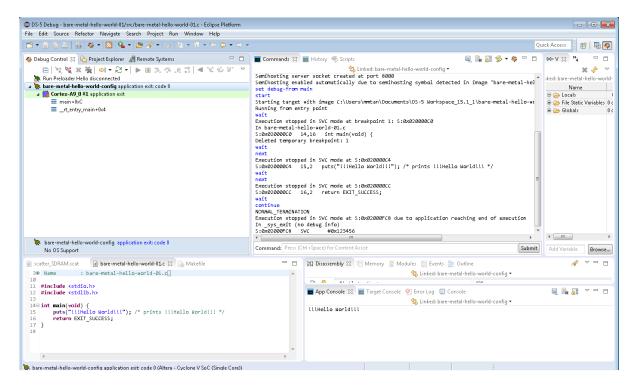
Figure 38: Bare Metal "Hello World" App Console

C DS-5 Debug - bare-metal-hello-world-01/src/bare-metal-hello-world-01							
File Edit Source Refactor Navigate Search Project Run Window Help							
				Qu	iick Access 🔡 🔛	R 🐺	
🏘 Debug Control 🖾 🏠 Project Explorer 📲 Remote Systems 📃 🗖	🖬 Commands 🔀	📷 History 🦓 Scripts		🖳 🗟 🚮 🤣 * # 🖓 🗆	(×)= V ⊠ ²⁰ 4		
📄 💐 💥 💥 🏟 이 🗸 🖓 🕨 🔈 👁 😥 😭 🔍 일 관 관 가 🔻			ked: bare-metal-hello-world-co		×	🛷 🤝	
🔭 Run Preloader Hello disconnected	No SYSID registe	No SYSID registers could be found. Has a peripheral description file been supplied?					
bare-metal-hello-world-config connected				e-metal-hello-world-01\Debug\bare	Name		
Cortex-A9_0 #1 stopped main+0xC			0 ~ S:0x02001233 (size 0 4 ~ S:0x02001243 (size 0		🖶 🗁 Locals 🖶 🗁 File Static Va	l de la la de	
Internet v main +0x4	Entry point S:00	02000000		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	H 🗁 Globals	nables 0 c 0 c	
		er socket created at		detected in image 'bare-metal-he]			
	set debug-from n		e to seminosting sympol	detected in image bare-metai-nel			
	start						
	Starting target Running from ent		mmtan\Documents\DS-5 Wor	<pre>rkspace_15.1_1\bare-metal-hello-wc</pre>			
	wait						
	Execution stoppe In bare-metal-he		akpoint 1: S:0x020000C0				
	S:0x020000C0 1	4,16 int main(void) {	E			
	Deleted temporar wait	Deleted temporary breakpoint: 1					
	next						
		d in SVC mode at S:0	x020000C4 World!!!"); /* prints !				
	wait	5,2 puts("IIIHeIIC	worldili"); /" prints :	TITHEITO WORLDITT "7			
	next						
		d in SVC mode at S:0 .6,2 return EXIT SU		-			
bare-metal-hello-world-config connected	•	·	,	- F	<	F.	
No OS Support	Command: Press (C	trl+Space) for Content Ass	st	Submit	Add Variable	Browse	
scatter_SDRAM.scat		111 Disassembly 😒 🗏	Memory 🚊 Modules 📰 Ev	ents 🔠 Outline	1		
3⊕ Name : bare-metal-hello-world-01.c.	*		🖾 Linked	: bare-metal-hello-world-config •			
			and the second s				
<pre>11 #include <stdio.h> 12 #include <stdlib.h></stdlib.h></stdio.h></pre>		🗖 App Console 🔀 🗖	Target Console 🧕 Error Log	📃 Console	🗎 🕞 🚮 🗸		
13			🗟 Linked	: bare-metal-hello-world-config =			
14⊖ int main(void) { 15 puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */		!!!Hello World!!!					
16 return EXIT_SUCCESS;							
17 }							
**							
	-						
4	Þ						

5. Select **Continue** to finish executing the program.



Figure 39: Bare Metal "Hello World" App Console End



6. To rerun, select the debug connection from the list in the Debug Control view. In this case, it is "bare-metal-hello-world-01"; and click the yellow arrow.

This runs the same debug configuration again (and run from Main).

7. When you are finished with this application, you can disconnect it from the target. To disconnect, right click on the **Debug Connection** and then click on the "Disconnect from Target".



Figure 40: Bare Metal "Hello World" App Console Disconnected

DS-5 Debug - bare-metal-hello-world-01/src/bare-metal-hello-world-01.c - Eclipse Platform							-	- • • ×		
File Edit Source Refactor Navigate Search Project Run Window Help										
≅ ▼ 🖩 % ≏ 📾 巻 ▼ 🔕 隆 ▼ 🅭 🖋 ▼ 🥖 ½ ▼ 🖻 ▼ ↔ ▼ ↔ ▼						Qu	ck Access	😰 🗟 🐺		
🌸 Debug Control 🕺 🏠 Project Explorer 📲 Remote Systems 📃 🗖	🗖 Commands 😒	🧰 History Scripts		R. 6	🕯 🚮 🗇 = 🏘 =		(x)= V 🙁 👋	4 - 0		
		🚖 Linke	ed: bare-metal-hello-wo	rld-config 🕶				\bigtriangledown		
🔭 Run Preloader Hello disconnected	set debug-from main						iked: bare-metal-hello-w			
Ann Preissor Preissor Preissor Preissor Preissor Preissor Preisson Preis	<pre>start Starting target with image C:\Users\mmtan\Documents' Running from entry point wait Execution stopped in SVC mode at breakpoint 1: S:0xi In bare-metal-hello-world-01.c S:0x02000C0 14,16 int main(void) { Deleted temporary breakpoint: 1 wait Execution stopped in SVC mode at S:0x020000C4 S:0x020000CC 16,2 puts("I!!Hello World!!!"); /* wait Execution stopped in SVC mode at S:0x020000CC S:0x020000CC 16,2 return EXIT_SUCCESS; wait Execution stopped in SVC mode at S:0x02000FC8 due to In mext Execution stopped in SVC mode at S:0x02000FC8 due to In mext Execution stopped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in SVC mode at S:0x02000FC8 due to In mext topped in topp</pre>		<pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>	:8x8220000C0		=======================================		tar-nello-world-		
🚴 bare-metal-hello-world-config disconnected	Commandi Brass (0	m Ctrl+Space) for Content Assist				▶ bmit				
No OS Support	Command: Hest (ourropace/ror content/Assist			Ju	onnic				
📄 scatter_SDRAM.scat 🛛 🖻 bare-metal-hello-world=01.c 🔀 🗋 Makefile	- 8	111 Disassembly 🛛 🗏 M	1emory 🚪 Modules	📰 Events 🛛 🔡 Outline				~		
3⊕ Name : bare-metal-hello-world-01.c.	*		🔩 L	inked: bare-metal-hello-	world-config 🕶					
10 11 #include (stdio ,h)		Disaccombly information is	not susilable							
11 #include <stalo.n> 12 #include <stalib.h></stalib.h></stalo.n>		🔳 App Console 🔀 🔳 T	Farget Console 🥺 Erro	r Log 📃 Console			🖳 🕞 🐻			
13			包 1	inked: bare-metal-hello-	world-config •					
<pre>14= int main(void) { 15 puts("iiiHello Worldill"); /* prints !!!Hello Worldill */ 16 return EXIT_SUCCESS; 17 } 18 4</pre>	*	!!!Hello World!!!								
🚴 bare-metal-hello-world-config disconnected (Altera - Cyclone V SoC (Single Core))										

It is not necessary to remove the disconnected debug connection from the Debug Control view; however, it can be removed since it only needs to run once.

Note: If you do not remove a disconnected debug connection, then you can re-launch the configuration by selecting it from the Debug Control view and clicking the **Connect to Target** icon above it.

Don't forget that if you reboot the board, then you have to run the Preloader to configure the memory controller before running an application in SDRAM. One quick way to do this is to keep the disconnected debug connection in the Debug Control view; then select it and click on "Connect to Target". This reproduces the previous launch.

- 8. When you are completely finished with this application, you can remove it from the Debug Control view. The connections must be disconnected in order to remove them. To disconnect, select the **Debug Connection** and then click on the "Disconnect from Target". Once disconnected you can select "Remove Connection" or "Remove All Connections".
- **9.** After removing the Debug Connection from the Debug Control view, rerun the Preloader by using the Debug Configuration that was created for the Preloader. Rerun the demo, using the Debug Configuration that was created for the **"Hello World"** application.

Modify Project to Run from SDRAM Instead of On-Chip RAM

After completing the process of creating a simple "**Hello World**" application and downloading and debugging it in on-chip RAM of the Altera SoC, the next step is to configure the same project to run in SDRAM instead of on-chip RAM.

Bare Metal User Guide

Altera Corporation



36 Create a New Scatter File to Locate the Bare Metal Application in the SDRAM

To use any SDRAM, the SDRAM controller needs to be configured. This is done by loading and running the Preloader.

For more information about loading and running the Preloader, refer to the "Preloader" section.

Related Information

Preloader on page 20

Create a New Scatter File to Locate the Bare Metal Application in the SDRAM

In the DS-5 ARM compiler projects, it is the scatter file that specifies the addresses that are used to locate the code in the required portion of the memory map.

In this step, use the same "**Hello World**" application, but build it located in the SDRAM memory instead of the on-chip RAM.

To use SDRAM, you must run the Preloader to configure the SDRAM memory controller. If the SDRAM is not configured before you start running the project from SDRAM, the following error messages appears:

```
ERROR(CMD16-TAD274-NAL22):
! Failed to load "bare metal-hello-world-01.axf"
! Failed to write 4,896 bytes to address S:0x02000000 while writing block of 4,096
bytes to address S:0x02000000
! General error on memory or register access.
```

- Create a scatter file for SDRAM. Go to File > New > Others... and select Scatter File Editor > Scatter File.
- 2. Click Next and enter the file name of the new scatter file as scatter_SDRAM.scat and click Finish.
- 3. In the scatter_SDRAM.scat editor view, enter the following:

```
SDRAM 0x02000000 0x02000000 ; 32M SDRAM
{
    APP_CODE + 0
    {
        * (+ R0 , + RW , + ZI )
    }
    ARM_LIB_STACKHEAP 0x03000000 EMPTY 0x0x01000000 ; Application heap and
stack
    {
    }
}
```

Note: If your HWLibs project needs interrupt support, you have to add a "VECTOR" section at the beginning of the **scatter** file. For example:

```
SDRAM 0x00100000 0x40000000
{
    VECTORS +0
    {
        * (VECTORS, +FIRST)
    }
    APP_CODE +0
    {
        * (+R0, +RW, +ZI)
    }
    ARM_LIB_STACKHEAP +0 EMPTY (0x40000000 - ImageLimit(APP_CODE)) ;
Application heap and stack
    {
    }
}
```



This is the standard scatter file for a project that runs from SDRAM in most of the HWLibs examples in the "*SoC EDS installation path*>\embedded\examples\software". Please note that you have to compile and link "**alt_interrupt_armcc.s**" in your project. This is needed because the ARMCC toolchain does not provide the vectors at the start of the program automatically.

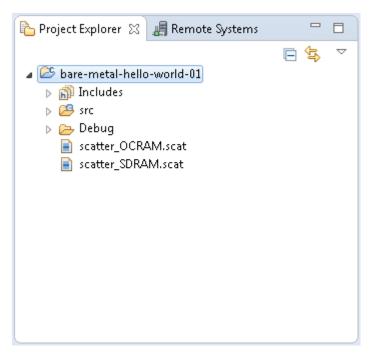
Note: For this section, the Vector section is not included since we are running a simple "Hello World" project only.

Figure 41: SDRAM Scatter File Code Snippet



4. Make sure the new scatter file appears in the Project Explorer view.

Figure 42: SDRAM Scatter file in Project Explorer



- 5. Associate the new scatter file with the project. Right-click on the project and select **Properties** > C/C++ Build > Settings > ARM Linker 5 > Image Layout.
- 6. Browse to the new scatter file.
- 7. Alternatively, instead of browsing, enter a path, like: \${workspace_loc}\bare-metal-hello-world-01\scatter_SDRAM.scat); select Apply and then OK.

Bare Metal User Guide

Altera Corporation

37



Figure 43: SDRAM Scatter File Location Setting

Tool Settings Build Artifact Binary Parsers Error Parsers Image entry point (entry) Image entry point (entry) Image entry point (entry) RO base address (ro_base) Image entry point (entry) Image entry point (entry) Image entry point (entry) Image entry point (entry) <tr< th=""><th>ttings</th><th></th><th></th><th></th><th>¢</th><th>• <> •</th></tr<>	ttings				¢	• <> •
ARM C Compiler 5 Preprocessor Miscellaneous Marget Preprocessor Miscellaneous Miscellaneous	nfiguration: Debug [Active]				 Manage Cor 	figuration
	 Tool Settings Build Steps Tool Settings Build Steps Target Target Preprocessor Includes Source Language Optimizations Debugging Warnings and Errors Preprocessor Preprocessor Debugging Warnings and Errors Debugging Warnings and Errors Carget Preprocessor ARM Assembler 5 Target Preprocessor Wiscellaneous ARM Linker 5 Target Target Debuguing Miscellaneous Additional Information Warnings and Errors 	Image entry point (entry) RO base address (ro_base) RW base address (rw_base) ZI base address (zi_base)		e_15.1_1\bare-metal-hell		

Right-click on the project and select Build Project to rebuild with the new scatter file. Use the same debug configuration created earlier to download and run the application.
 The program stops at main and waits there. In the Commands view, notice the entry point is 0x02000000, which is the start address specified in the scatter file for the SDRAM.

Altera Corporation

Bare Metal User Guide



Figure 44: Program Stops at SDRAM Start Address

	<u>R</u> un <u>W</u> indow <u>H</u> elp			
i • 🖫 🖻 📄 🏶 • 🔊 🂁 • 🙋 🖋 • 🌛	[] <u>2</u>] × ²] × ⁴ ← → ×			Quick Access 🔡 😰 🖬
Debug C 🙁 🍋 Project E 📕 Remote 👘 🗖	🗖 Commands 🕺 📷 History 🤏	Scripts 🛛 📓 📓 🤣 🔻 🏶 🖻	" 🗖 ⋈= Va 🔀 💁 Br 👳	🖲 Re 👬 Ex f() Fu 🗖
i ½ % × 🙀 4) + C + ▶ III 3. ⊙ . (c)	🔄 Linked: bai	re-metal-hello-world-config 🔹		x 🖑
	He siste i eBracei a conta ac		🔺 🔄 🔄	re-metal-hello-world-config 🔻
🔭 RunPreloaderDefault disconnected		<pre>cuments\DS-5 Workspace_15.1_1\bar 0x020000000 ~ S:0x02001233 (size 0</pre>		Value Type Count Size
bare-metal-hello-world-config connected		0x020000000 ~ 5:0x02001233 (S1Ze 0 0x02001234 ~ 5:0x02001243 (size 0	Iv10 🗁 🦢 Locais	0 variables
Cortex-A9_0 #1 stopped on breakpoint	Entry point 5:0x02000000			0 of 0 variables 0 of 0 variables
= main	Semihosting server socket c	reated at port 8000 ically due to semihosting symbol		o or o variables
_rt_entry_main+0x4	set debug-from main	ically are to senthosting symbol	act	
	start			
		C:\Users\mmtan\Documents\DS-5 Wor	ksp≡	
	Running from entry point			
	wait			
	Execution stopped in SVC mo	de at breakpoint 1: S:0x020000C0		
	S:0x020000C0 14,16 int	main(void) {		
	Deleted temporary breakpoin	t:1	·	
No. and the second second	< III		•	III
bare-metal-hello-world-config connected	Command: Press (Ctrl+Space) for C	Content Assist	ubmit Add Variable	Brow
No OS Support				ВЮ
scatter_SDRAM.scat	2	111 Disassembly 🖾 🗐 Memory 🚊 M	1odules 🔚 Events 📑 Outline	1
3⊕ Name : bare-metal-hello-world-01.c			.inked: bare-metal-hello-world-cor	
0 see : bare-metal-hello-world-01.cl	<u>^</u>			fig ¥
1 #include <stdio.h></stdio.h>		😫 🞒 👻 <next instruction=""></next>	100	
		Address Opcode Disas	sembly	
		S:0x020000BC EB0003BF BL	_sys_exit ; 0x2000FC0	
3				
3 4⊖ int main(void) {		main		
3 40 int main(void) { 5 puts("!!!Hello World!!!"); /* prints !!	!Hello World!!! */	S:0x020000C0 09204010 PUSH	{r4, lr}	
3 40 <mark>int main(void) {</mark> 5 puts("!!!Hello World!!!"); /* prints !! 6 return EXIT_SUCCESS;	!Hello World!!! *∕	 S:0x020000C0 E0004010 PUSH S:0x020000C4 E28F0008 ADR 	r0,{pc}+0x10 ; 0x2000	0d4
3 40 <mark>int main(void) {</mark> 5 puts("!!!Hello World!!!"); /* prints !! 6 return EXIT_SUCCESS; 7 }	!Hello World!!! */	S:0x020000C0 PUSH S:0x020000C4 E28F0008 ADR S:0x020000C8 EB000006 BL	r0,{pc}+0x10 ; 0x2000 puts ; 0x20000E8	0d4
3 49 int main(void) { 5 puts("!!!Hello World!!!"); /* prints !! 6 return EXIT_SUCCESS; 7 }	!Hello World!!! */	➡ S:0x02000000 PUSH S:0x02000004 E28F0008 ADR S:0x020000068 E8000006 BL S:0x0200000CC E3A00000 MOV	r0,{pc}+0x10 ; 0x20000 puts ; 0x20000E8 r0,#0	d4
3 40 <mark>int main(void) {</mark> 5 puts("!!!Hello World!!!"); /* prints !! 6 return EXIT_SUCCESS; 7 }	!Hello World!!! */	S:0x020000C0 PUSH S:0x020000C4 E28F0008 ADR S:0x020000C8 EB000006 BL	r0,{pc}+0x10 ; 0x2000 puts ; 0x20000E8)d4
3 40 <mark>int main(void) {</mark> 5 puts("!!!Hello World!!!"); /* prints !!	HHello World‼! */	\$:0x020000C0 PUSH \$:0x020000C4 £2870008 ADR \$:0x020000C6 E8000006 BL \$:0x020000C6 E8000006 BL \$:0x020000C6 E8000006 How \$:0x020000C0 E800000 How	r0,{pc}+0x10 ; 0x20000 puts ; 0x20000E8 r0,#0 {r4,pc}	
3 40 <mark>int main(void) {</mark> 5 puts("!!!Hello World!!!"); /* prints !! 6 return EXIT_SUCCESS; 7 }	Hello World!!! */	➡ S:0x02000000 PUSH S:0x02000004 E28F0008 ADR S:0x020000068 E8000006 BL S:0x0200000CC E3A00000 MOV	r0,{pc}+0x10 ; 0x20000 puts ; 0x20000E8 r0,#0 {r4,pc}	2d4

Alternative Way to Create a Simple Bare Metal Project Running from On-Chip-RAM

To create a simple bare metal project that runs from OCRAM, you can also choose to import the Bare Metal example that is included with the Altera SoC EDS tools into the ARM DS-5 environment. For the simple "**Hello World**" bare metal project, you can import either one of the following from the "*SoC EDS installation path*>\embedded\examples\software":

- Altera-SoCFPGA-HelloWorld-Baremetal-ARMCC.tar (using ARM compiler)
- Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar (using GNU CC compiler)

Note: The above bare metal project examples can run on Arria 10, Arria V or Cyclone V SoC Development Kits.



Figure 45: Embedded Software Example Design List

♥ 🕖 🗢 🚺 ≪ altera 🕨 15.1	embedded 🕨 examples 🕨 software 👻 🍫 Sec	arch software	3
Organize 👻 🛛 💐 Open with V	inZip 🔻 Print New folder	B • 🗖	0
🔆 Favorites	Name	Date modified	Ту
🧮 Desktop	Altera-SoCFPGA-Blinking-LED-Linux-GNU.tar	10/9/2015 5:50 AM	W
〕 Downloads	Altera-SoCFPGA-HardwareLib-16550-CV-ARMCC.tar	10/9/2015 5:50 AM	W
🔛 Recent Places	💐 Altera-SoCFPGA-HardwareLib-16550-CV-GNU.tar	10/9/2015 5:50 AM	W
🎍 Personal Documents	💐 Altera-SoCFPGA-HardwareLib-ECCL2-CV-ARMCC.tar	10/9/2015 5:50 AM	W
	💐 Altera-SoCFPGA-HardwareLib-ECCL2-CV-GNU.tar	10/9/2015 5:50 AM	W
词 Libraries	💐 Altera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC.tar	10/9/2015 5:50 AM	W
Documents	💐 Altera-SoCFPGA-HardwareLib-FPGA-CV-GNU.tar	10/9/2015 5:50 AM	W
🎝 Music	💐 Altera-SoCFPGA-HardwareLib-MPL.tar	10/9/2015 5:50 AM	W
📔 Pictures 💡	💐 Altera-SoCFPGA-HardwareLib-SPI-CV-ARMCC.tar	10/9/2015 5:50 AM	W
🚼 Videos	💐 Altera-SoCFPGA-HardwareLib-SPI-CV-GNU.tar	10/9/2015 5:50 AM	W
	💐 Altera-SoCFPGA-HardwareLib-Timer-CV-ARMCC.tar	10/9/2015 5:50 AM	W
🖳 Computer	🍕 Altera-SoCFPGA-HardwareLib-Timer-CV-GNU.tar	10/9/2015 5:50 AM	W
🏭 Local Disk (C:)	💐 Altera-SoCFPGA-HelloWorld-Baremetal-ARMCC.tar	10/9/2015 5:50 AM	W
💼 System Reserved (D:)	💐 Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar	10/9/2015 5:50 AM	W
👝 Local Disk (E:)	💐 Altera-SoCFPGA-HelloWorld-Linux-GNU.tar	10/9/2015 5:50 AM	W
🖵 mmtan (\\cmode.pg-itcr	Altera-SoCFPGA-Push-Button-Linux-GNU.tar	10/9/2015 5:50 AM	W

These examples are **make-based** examples where the projects compile based on the Makefile settings or configurations. Creating a simple Bare Metal project manually as shown in the "Simple Bare-Metal Project Using On-Chip-RAM" section is creating a **managed-make** project where the **makefile** is auto-generated.

For more information on how to import, build and debug the project, you can refer to "Importing, Building and Debugging in a Make-Based Example" section.

Related Information

- Importing, Building and Debugging in a Make-Based Example on page 40
- Simple Bare Metal Project Using On-Chip-RAM on page 5

Importing, Building and Debugging in a Make-Based Example

The import, build and debug of the Altera SoC EDS Make-based example covers importing and debugging an example included with the Altera SoC EDS tools into the ARM DS-5 environment.

The Make-based example that is included in the following sections, loads the Golden Hardware Reference Design (GHRD) FPGA image on Cyclone V SoC Development Kit and blinks LEDs to test that the HPS can control peripherals in the FPGA fabric. This example compiles using the ARM compiler.

Bare Metal User Guide



Import the Project

- 1. From within **DS-5**, select **File** > **Import...**
- 2. In the resulting dialog box, select "Existing Projects into Workspace" and click Next.

Figure 46: Import Existing Projects

⊜ Import	- • •
Select Create new projects from an archive file or directory.	Ľ
Select an import source:	
type filter text	
 ✓ General ✓ Archive File ✓ Existing Projects into Workspace ✓ File System ✓ Preferences ✓ C/C++ ✓ CVS ✓ Install ✓ Remote Systems ✓ Run/Debug ✓ Scatter File Editor ✓ Target Configuration Editor ✓ Team 	
A Back <u>Next > Einish</u>	Cancel

3. Select "Select archive file" and browse to the software examples directory of your installation, as shown.



Figure 47: Embedded Software Example Design List

Organize 🔻 🛛 💐 Open with WinZip	 Print Burn New folder 	≣ ▼ 🔲 (
🔶 Favorites 🔒	Name	Date modified
🔜 Desktop	🔍 Altera-SoCFPGA-Blinking-LED-Linux-GNU.tar	10/9/2015 5:50 A
📜 Downloads	Altera-SoCFPGA-HardwareLib-16550-CV-ARMCC.tar	10/9/2015 5:50 A
📃 Recent Places	Altera-SoCFPGA-HardwareLib-16550-CV-GNU.tar	10/9/2015 5:50 A
퉬 Personal Documents	💐 Altera-SoCFPGA-HardwareLib-ECCL2-CV-ARMCC.tar	10/9/2015 5:50 A
	💐 Altera-SoCFPGA-HardwareLib-ECCL2-CV-GNU.tar	10/9/2015 5:50 A
🗃 Libraries	🔍 Altera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC.tar	10/9/2015 5:50 A
Documents	💐 Altera-SoCFPGA-HardwareLib-FPGA-CV-GNU.tar	10/9/2015 5:50 A
🎝 Music	🌉 Altera-SoCFPGA-HardwareLib-MPL.tar	10/9/2015 5:50 A
📔 Pictures 📃	Altera-SoCFPGA-HardwareLib-SPI-CV-ARMCC.tar	10/9/2015 5:50 A
📑 Videos	🂐 Altera-SoCFPGA-HardwareLib-SPI-CV-GNU.tar	10/9/2015 5:50 A
	🂐 Altera-SoCFPGA-HardwareLib-Timer-CV-ARMCC.tar	10/9/2015 5:50 A
💺 Computer	🍕 Altera-SoCFPGA-HardwareLib-Timer-CV-GNU.tar	10/9/2015 5:50 A
🚢 Local Disk (C:)	🂐 Altera-SoCFPGA-HelloWorld-Baremetal-ARMCC.tar	10/9/2015 5:50 A
👝 System Reserved (D:)	🂐 Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar	10/9/2015 5:50 A
👝 Local Disk (E:)	🍕 Altera-SoCFPGA-HelloWorld-Linux-GNU.tar	10/9/2015 5:50 A
坖 mmtan (\\cmode.pg-itcrnv-na	🍕 Altera-SoCFPGA-Push-Button-Linux-GNU.tar	10/9/2015 5:50 A
坖 mmtan (\\mysitespg\personal)		
雬 swdev (\\pg-itnas03b) (S:)		
🖢 Network 🗸 👻	•	

Note: The file can be found in *<SoC EDS installation path>*\embedded\examples\software. You can import other examples for reference according to your usage.

- 4. Choose the "Altera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC.tar.gz" archive and select Open.
- 5. Click Finish to complete the import process.

Build the Project

Now that the project is imported, make sure the current toolchain (ARM Compiler 5 (DS-5 built-in)) is selected correctly in the Tool Chain Editor. This can be done with the following steps:

- 1. Right click on the project and select "Properties".
- **2.** Go to C/C++ Build > Tool Chain Editor.

Next, right click on the project and select "Build Project". This initiates a Make-based build that does the following:

- Copies additional source files from HWLibs into the project.
- Creates an object file of the FPGA image using standard Altera tools and objcopy.
- Compiles and links everything into an AXF executable (ELF-compatible).

Note: For details of what is happening, browse the Makefile that is part of the project.

Altera Corporation

Bare Metal User Guide



Included with the project are a number of key items that are used to properly initialize the ARM core:

• A scatter file (scatter.scat) that the ARM tools use for linking

Note: Much simpler that GNU linker syntax

- A DS script file (debug-hosted.ds) that controls debug flow
 - Loads and runs the Preloader
 - Loads the project executable (HWLIB.axf) and stops at the "main" symbol
- A readme.txt file which describes more about this example

Debug the Project

1. Right-click on the project and select **Debug** > **Debug** As > **Debug** Configurations, as shown:

Figure 48: "Altera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC-Debug" Debug Configurations

Debug Configurations	
Create, manage, and run configurations Onfiguration for connection type 'Bare Me	tal Debug' is not valid - Connection cannot be empty.
Image: Source of the second	Name: Altera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC-Debug Image: Connection Connection
?	Debug Close

2. Next to the **Connections** field, select **Browse** and select the hardware you are using (Altera USB Blaster or DStream) from the available connections and click **Select**.

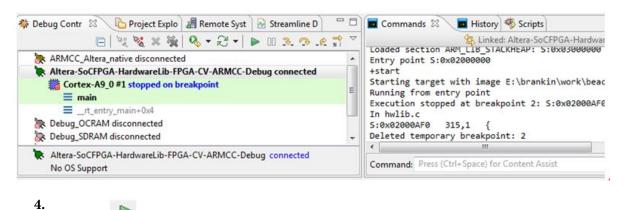


Figure 49: Connection Browser

Connection Browser	•••
Connection Browser	
Select a target connection	
USB-BlasterII USB-1	
USB-BlasterII on localhost [USB-1]	
?	Select Cancel

3. Select **Apply** and click **Debug** to start the debug session which will configure the processor, load the software and execute it, stopping at the **"main"** symbol as the default.

Figure 50: Debug Stopping at Main



Click the button to start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to completion. Other options (Defined a start running the code. The code should run to complete a start running the code. The code should run to complete a start running the code. The code should run to complete a start running the code. The code should run to complete a start running the code. The code should run to complete a start running the code. The code should run to code should run to complete a start running the code should run to code should

Bare Metal User Guide



Figure 51: Application Exit Displayed in App Console

🖬 App Console 🕮 🛛 🖬 Target Console 🧐 Error Log	🐐 Debug Contr 🖾 🔁 Project Explo 🛛 🖉 Remote Syst 🛛 🔂 Streamline D 👘 🗖
Linked: Altera-SoCFPGA-HardwareLib-FPG/ INFO: Gray code(i=0xa) => 0xf [1111].	🕒 🧏 🍇 🍇 💊 • 중 • 🕨 🗷 👁 正言 🦈
INFO: Gray code(i=0xb) => 0xe [1110].	🚴 ARMCC_Altera_native disconnected
INFO: Gray code(i=0xc) => 0xa [1010]. INFO: Gray code(i=0xd) => 0xb [1011].	RItera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC-Debug application exit: co
INFO: Gray code(i=0xe) => 0x9 [1001].	Cortex-A9_0 #1 application exit
INFO: Gray code(i=0xf) => 0x8 [1000]. INFO: LEDs should have blinked.	No Stack
	Debug_OCRAM disconnected
(NFO: Cleanup of Bridge [2]	🔭 Debug_SDRAM disconnected
INFO: Cleanup of FPGA INFO: Cleaning up DMA System	FPGA_conf_ds-5 disconnected
RESULT: Example completed successfully.	Altera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC-De application exit: code 0 No OS Support

DS-5 ARM HWLIBs Project Derived from Make-Based Project

This section takes a makefile-based example project that comes with SoC EDS and converts it to a managed make project. "Managed Make" in Eclipse means that the IDE (DS-5 in this case) takes care of generating and maintaining any Makefiles.

When you use this method, you are able to change the project settings from the GUI.

Note: You cannot automate generic steps, like converting the FPGA SOF file to C object code to be linked into the application.

A managed make project can be compiled from the command line, since there is a makefile. Therefore, automated builds can be done using commands.

Create Project

Before you begin

Before you start creating a project, make sure you have gone through the "Importing, Building and Debugging in a Make-Based Example" section and the example project compiled successfully.

- 1. Create a new "managed make" project by selecting File > New > C Project.
- 2. Fill in the resulting dialog box, as shown, naming it whatever you prefer, for example: armcc_fpga_managed_make.



Figure 52: Create New ARMCC FPGA Managed Make Project

🖨 C Project				
C Project Create C project of selected type				
Project name: armcc_fpga_managed_make Image: Intersection Location: C:\Users\mmtan\Documents\DS-5 Work	space_15.1_3\armcc_fpga_manag Browse			
Choose file system: default 💌 Project type:	Toolchains:			
 Executable Empty Project Hello World ANSI C Project Shared Library Static Library Makefile project 	ARM Compiler 5 (DS-5 built-in) ARM Compiler 6 (DS-5 built-in) Altera Baremetal GCC GCC 4.x [arm-linux-gnueabihf] (DS-5 built-in)			
Show project types and toolchains only if they are supported on the platform ? < Back Next > Finish Cancel				

Related Information Importing, Building and Debugging in a Make-Based Example on page 40

Bare Metal User Guide



Copy Files

Make sure the HWLibs FPGA ARMCC project has been imported and compiled successfully before you proceed. The compiled project creates the FPGA object code file, previously, auto generated using the makefile.

Before you begin

Copy all C code (files with a .c extension), **alt_interrupt_armcc.s**, the scatter file and the FPGA object code (**soc_system_dc.o**) from the Altera SoC EDS example project.

1. Select the files, right-click and click Copy.



Figure 53: Files to Copy from the SoC EDS Example Project

်ြ Project	Explorer 🛿 📕 Remote Systems 📃					
a 🌮 Alte	a 💕 Altera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC					
	🕞 📸 Includes					
	alt_address_space.c					
	alt_bridge_manager.c					
	alt_cache.c					
	alt_clock_manager.c					
⊳ 💽	alt_dma_program.c					
⊳ 💽	alt_dma.c					
⊳ 💽	alt_fpga_manager.c					
⊳ 💽	alt_hps_detect.c					
Þ []h	alt_hps_detect.h					
	alt_interrupt_armcc.s					
⊳ 🚾	hwlib.c					
_	alt_address_space.o					
_	alt_bridge_manager.o					
_	alt_cache.o					
_	alt_clock_manager.o					
_	alt_dma_program.o					
_	alt_dma.o					
	alt_fpga_manager.o					
_	alt_hps_detect.o					
	alt_interrupt_armcc.o					
	Altera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC-Debug.launch					
	debug-hosted.ds					
	hwlib.axf					
	hwlib.axf.map					
	hwlib.axf.objdump					
	hwlib.o					
-	Makefile					
_	readme.txt					
	scatter.scat					
	soc_system_dc.o					
	soc_system_dc.rbf					
	soc_system.sof u-boot-spl.axf					
010	u-boot-spildxi	_				

2. Right-click the new project (armcc_fpga_managed_make) and select Paste. All of the files shown in the image should have been copied into the new project.

The C code is built and linked with the FPGA object file, automatically, after setting up the build system in the next section.

Configure Build Settings

Get ready to build.

Right-click on the project in DS-5's Project Explorer window and select **Properties** then browse to C/C++ Build and select Settings.

Altera Corporation

Bare Metal User Guide



Figure 54: ARMCC FPGA Managed Make Settings

Properties for armcc_fpga_man	aged_make 🗖 🔲 💌
type filter text	Settings 🗘 🕆 🖒 👻 💌
 Resource Builders C/C++ Build Build Variables Environment Logging Settings Tool Chain Editor C/C++ General Project References Run/Debug Settings 	Configuration: Debug [Active] Includes Build Artifact Binary Parsers ARM C Compiler S Command: arrncc Target Preprocessor All options: -00 -g Source Language Optimizations -00 -g

Compiler, assembler and linker settings changes are necessary. Please follow the steps in the "ARM C Compiler Settings" section.

Related Information

ARM C Compiler Settings

ARM C Compiler Settings

1. In "Target", set Target CPU" to "Cortex-A9" and check the "Disable unaligned accesses" box.



Figure 55: ARMCC Target Settings

610	🖇 Tool Settings 🎤 Build Steps 🚇	Build Artifact 🛛 🗟 Binary Parsers	8 Error Parsers
	4 🛞 ARM C Compiler 5	Target CPU (cpu)	Cortex-A9
	🖄 Target 🍅 Preprocessor	Byte order	Default 🔹
	🖄 Includes	Instruction set	Default 🔹
	i Source Language	☐ Interworking (apcs=/interwor ✓ Disable unaligned accesses (r Target FPU (fpu) Flacting point mode (formed)	o_unaligned_access)
	ARM Assembler 5 Arget	Floating-point mode (fpmode) Floating-point PCS (apcs)	Default
🖉 Preprocessor 🆄 Debuqqinq		Generate enumerations as integ	[]
	🖉 Warnings and Errors 🖄 Miscellaneous	Wide character size	Default 🔹

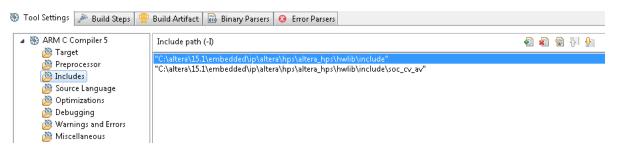
2. Add the preprocessor macro "ALT_FPGA_ENABLE_DMA_SUPPORT=1" to ensure that the code responsible for configuring the FPGA uses the DMA in the HPS.

Figure 56: ARMCC Preprocessor Settings



- 3. Add the preprocessor macro "soc_cv_av" to set the Cyclone V device family.
- **4.** Select the next category down (**"Includes"**) and click the **"+"** button to the right to add the include paths to **Build Settings**.

Figure 57: ARMCC Include Path Settings



5. Select "Source Language" and set "Source language mode" to "-c99".

Bare Metal User Guide



Figure 58: ARMCC Source Language Settings

🛞 Tool Settings 🎤 Build Steps 🚇	Puild Artifact 🛛 🗟 Binary Parsers 🛛 😣 Error Parsers
 ARM C Compiler 5 Target Preprocessor Includes Source Language Optimizations Debugging Warnings and Errors Miscellaneous 	Source language mode C99 (c99) Enable GNU extensions (gnu) Strict language conformance Default Enable C++ exceptions (exceptions)

Leave "**Optimizations**", "**Debugging**", "**Warnings and Errors**", and "**Miscellaneous**" settings to default values.

ARM Assembler Settings

Change the settings in "Target" to match the following:

Figure 59: ARM Assembler Target Settings

🛞 Tool Settings 🎤 Build Steps 🚇 Build	ld Artifact 🗟 Binary Parsers	8 Error Parsers
 ARM C Compiler 5 Target Preprocessor Includes Instructions Optimizations Debugging Warnings and Errors Miscellaneous ARM Assembler 5 	get CPU (cpu) e order ruction set nterworking (apcs=/interword)isable unaligned accesses (n get FPU (fpu) iting-point mode (fpmode)	Cortex-A9 Default Default k)

ARM Linker Settings

1. Define the "Target CPU (--cpu)" as "Cortex-A9" in the Target section.



Figure 60: ARM Linker Target Settings

2. "Define the Image entry point (--entry) as "alt_interrupt_vector" and add the Scatter file (--scatter) location in the "Image Layout" section.

Figure 61: ARM Linker Image Layout Settings

🛞 Tool Settings 🎤 Build Steps Build Artifact 🗟 Binary Parsers 🥝 Error Parsers	
 Tool Settings Build Steps Build Artifact Binary Parsers Error Parsers ARM C Compiler 5 Target Preprocessor Maines and Errors Marcelaneous Sarget ARM Assembler 5 Target Target Marcelaneous ARM Linker 5 Target Marcelaneous Marcelaneous 	

- **3.** Leave "Libraries", "Optimization", "Additional Information", and "Warnings and Errors" with default values.
- 4. Add the FPGA object file in the Miscellaneous settings section.
- **5.** Click on the **Add...** icon and Browse to the location of the FPGA object file under the "armcc_fpga_managed_make" Workspace.

Bare Metal User Guide



Figure 62: ARM Linker Miscellaneous Settings

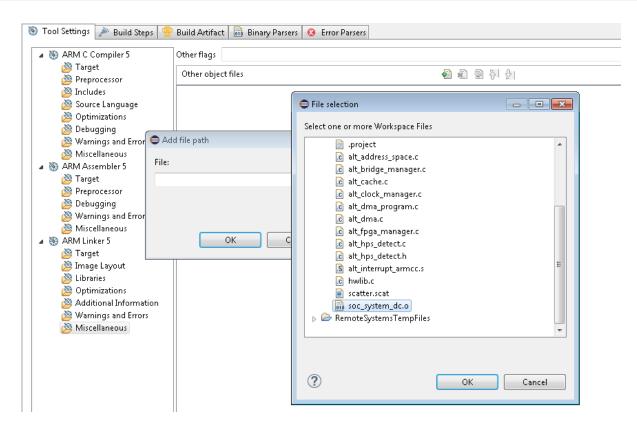
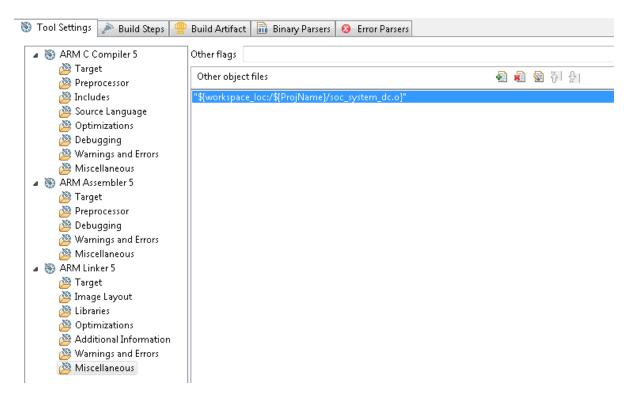




Figure 63: ARM Linker Miscellaneous Settings - Part 2



6. Click Apply and then OK to apply settings and return.

Build Project

Right-click on the project you created and pull-down to the "Build Project" option. This starts building the project in the default build directory in the project.

Run/Debug Project

Create a Debug Configuration

- Right-click on the project and pull-down to the Debug As > Debug Configurations option. This opens the Debug Configurations dialog box.
- 2. Create a new configuration and setup the debug hardware as shown.

Bare Metal User Guide



Figure 64: ARMCC FPGA Managed Make Debug Connection Settings

Debug Configurations	
Create, manage, and run configurations ② Configuration for connection type 'Bare M	etal Debug' is not valid - Connection cannot be empty.
Image: Society of Societ	Name: armcc_fpga_managed_make_blebug
?	Debug Close

3. Select an available Bare Metal Debug Connection by clicking **Browse**. This returns a list of the available debug connections.

Figure 65: Connection Browser

Connection Browser	—
Connection Browser	
Select a target connection	
USB-BlasterII USB-1	
USB-BlasterII on localhost [USB-1]	
?	Select Cancel

4. Click **OK** and then go to the **Files** tab on the top right. Select "**Workspace...**" and look for the **AXF** file in the project **Debug** sub-directory.



Figure 66: ARMCC FPGA Managed Make Files Settings

Name: armcc_fpga_managed_make_debug	
🙅 Connection 🔚 Files 🛛 🏘 Debugger 🦓 OS Awareness 🕺 Arguments 🚾 Environment	
Target Configuration Application on host to download: \${workspace_loc:/armcc_fpga_managed_make/Debug/armcc_fpga_managed_make.axf} File System Workspace I Load symbols Files	E
Load symbols from file File System Workspace	
•	-
Apply Reyert	

5. Click on Apply and then Debug to start the debugging session.

Minimal Preloader

The Minimal Preloader (MPL) is an alternative for the General Public License (GPL) Preloader. It uses the BSD license and may be freely distributed and modified according to the terms of that license. The MPL supports a subset of features supported by Altera's GPL Preloader.

The MPL initializes the PLLs, reset signals, configures IOCSR and pin MUXing, and performs other configuration-based on the preloader generator file settings. It can also load the FPGA from a boot source, if desired. It then reads a secondary image from a boot source into RAM and hands control to that image.

This version of MPL supports booting from QSPI, SD/MMC and FPGA.

Note: NAND boot is not supported.

The MPL uses Altera HWLib drivers for most of its functionality. It also uses Altera HWLib SoCAL folders for the memory map definitions and basic read and write commands.

The MPL supports both the ARMCC and GNU GCC compilers. The MPL supports both Cyclone V SoC and Arria V SoC devices.

Note: Arria 10 SoC devices are not supported.

The example project in the following section is for Cyclone V SoC. Please modify the example as needed to select appropriate file names.

Bare Metal User Guide



Minimal Preloader Example Project

Before you begin

The Minimal Preloader (MPL) project provided in the Altera SoC EDS is a non-GPL Preloader that uses the Altera SoC EDS HWLIBS. To build the MPL, either the ARM Compiler Toolchain or the GCC must be greater than v14.1.

Import the MPL project by using the steps in the "Importing an Existing Bare Metal Project into DS-5" chapter.

1. Go to File > Import and select General > Existing Projects into Workspace.

Figure 67: Import Existing Projects

€ Import	- • •
Select Create new projects from an archive file or directory.	Ľ
Select an import source:	
type filter text	
 General Archive File Existing Projects into Workspace File System Preferences C/C++ CVS Install Remote Systems Run/Debug Scatter File Editor Scatter File Editor Target Configuration Editor Team 	
(?) < <u>Back</u> <u>Next > <u>Finish</u></u>	Cancel

2. Select the Altera-SoCFPGA-HardwareLib-MPL.tar project from the <SoC EDS installation path>\ embedded\examples\software and then click Finish to import the project.



Figure 68: Import MPL Project File

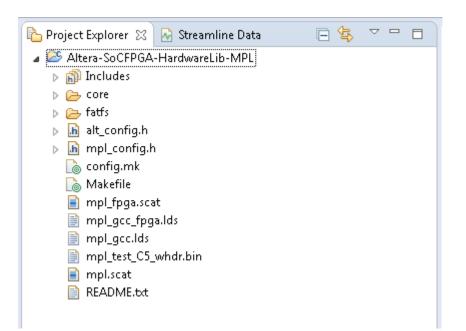
🖨 Import		- 0 x
Import Projects Select a directory to sear	ch for existing Eclipse projects.	
 Select root directory: Select archive file: 	C:\altera\15.1\embedded\examples\software\Altera-SoCFPGA-F 👻	Browse Browse
Projects:	HardwareLib-MPL (Altera-SoCFPGA-HardwareLib-MPL)	Select All Deselect All Refresh
Options Search for nested pro Copy projects into w Hide projects that alr Working sets Add project to work Working sets:	orkspace ready exist in the workspace	Select
?	< Back Next > Finish	Cancel

3. From the **Project Explorer** tab, verify all project files are present.

Bare Metal User Guide



Figure 69: MPL Project File List



For more information about building and debugging, refer to the "Importing, Building and Debugging in a Make-Based Example" section.

Related Information

Importing, Building and Debugging in a Make-Based Example on page 40

Appendix: Troubleshooting

Debug Cable Does Not Work

Be sure that the USB-Blaster II driver is installed and that it is functional. Generally, following the development kit installation instructions and going through a few of the recommended examples is good enough to solve this issue.

FPGA Is Not Programmed Successfully

If this occurs, refer to specifics on your development kit, but usually this is due to a mismatched MSEL (programming mode).

Temporary Directories not Writable

Though uncommon, it is possible for temporary directories that the Eclipse-based DS-5 debugger relies on being writable are sometimes un-writable. Also, if you run into strange "permission denied" issues when starting debug sessions, search through all temporary directories in your environment (**tmp**, **TMP**, **temp**, **TEMP**) and change or set them to a writable directory.

Bare Metal User Guide

Altera Corporation

