## MicroZed/PicoZed: Hello World



27 March 2017 Version 2016\_4.01

## Overview

Once a Zynq Hardware Platform is created and exported from Vivado, the next step is to create an application targeted at the platform and see it operating in hardware. This tutorial will show how to do that with the simplest of all software applications – Hello World.

## Objectives

When this tutorial is complete, you will be able to:

- Import a Zynq Hardware Platform into SDK
- Create a BSP
- Add a new application based on a Xilinx-provided template in SDK
- Run the application on the MicroZed or PicoZed hardware

© 2016 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.



## Experiment Setup

### Software

The software used to test this reference design is:

- Windows-7 64-bit
- Xilinx SDK 2016.4
- Silicon Labs CP201x USB-to-UART Bridge Driver
  - <u>www.microzed.org</u> → Support → Documentation → MicroZed
     Silicon Labs CP210x USB-to-UART Setup Guide
  - Note that MicroZed and both PicoZed FMC Carriers all use the same Silicon Labs CP2104 device, so the setup is the same.

### Hardware

The hardware setup used to test this reference design includes:

- Win-7 PC with the following recommended memory<sup>1</sup>:
  - 1.6 GB RAM available for the Xilinx tools to complete a XC7Z010 design
  - o 2.3 GB RAM available for the Xilinx tools to complete a XC7Z015 design
  - o 1.9 GB RAM available for the Xilinx tools to complete a XC7Z020 design
  - o 2.7 GB RAM available for the Xilinx tools to complete a XC7Z030 design
- One of the following:
  - Avnet MicroZed 7010 or 7020
  - Avnet PicoZed 7010, 7015, 7020, or 7030 with either the PicoZed FMC Carrier V1 or PicoZed FMC Carrier V2
- USB cable (Type A to Micro-USB Type B)
- JTAG Programming Cable (Platform Cable USB, Digilent HS1, HS2, or HS3 cable)
  - If you don't already have a JTAG Cable, Avnet recommends the Digilent HS3 Cable
  - o <u>http://www.em.avnet.com/jtaghs3</u>

<sup>&</sup>lt;sup>1</sup> Refer to <u>www.xilinx.com/design-tools/vivado/memory.htm</u>



## **Experiment 1: Import the Hardware Platform**

The first requirement within SDK is to import a hardware platform.

1. Launch SDK by selecting Start → All Programs → Xilinx Design Tools → SDK 2016.4 → Xilinx SDK 2016.4.

27 March 2017

2. Select a workspace. Click OK.

Workspace Launcher						
Select a workspace						
Xilinx SDK st Choose a wo	ores your projects in a folder called a workspace. orkspace folder to use for this session.					
Workspace:	C:\Avnet\MicroZed\Applications\MZ_Basic_System	Browse				
🔲 Use this a	s the default and do not ask again					
		OK Cancel				

Figure 1 – SDK Workspace

3. If at any time you get a Windows Security Alert, select the first two checkboxes, then click **Allow access**, then click **Yes**.

🔗 Windows Secur	ity Alert		8			
Windo	ws Firewa	II has blocked some features of this program				
Windows Firewall h	as blocked som	e features of eclipse.exe on all public, private and domain				
	Name:	eclipse.exe				
	Publisher:	Unknown				
	Path:	C:\xilinx\sdk\2015.2\eclipse\win64.o\eclipse.exe				
Allow eclipse.exe b Domain netw	o communicate vorks, such as a	on these networks: a workplace network				
🔽 Private netw	Private networks, such as my home or work network					
Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)						
What are the risks of allowing a program through a firewall?						
		Reference Cancel Cancel	el			

Figure 2 – Windows Security Alert from SDK

4. Close the *Welcome* screen by clicking the  $\bowtie$  next to *Welcome* on the tab.



Now we will import the Zynq hardware platform that was designed and built during the first tutorial.

- 5. Select File  $\rightarrow$  New  $\rightarrow$  Project.
- 6. Expand the *Xilinx* item, and select **Hardware Platform Specification**. Click **Next >**.

SOK New Project	
Select a wizard	
Wizards:	
type filter text	
<ul> <li>▷ General</li> <li>▷ C/C++</li> <li>▷ Java</li> <li>▷ Tracing</li> <li>▲ ▷ Xilinx</li> <li>▲ Application Project</li> <li>▲ ARM Trusted Firmware Project</li> <li>▲ Board Support Package</li> <li>➡ Hardware Platform Specification</li> <li>▲ Library Project</li> <li>☑ SPM Project</li> </ul>	
? < Back Next > Finish	Cancel

Figure 3 – Creating a New Hardware Platform



If you had simply launched SDK from Vivado, it would have automatically named and imported your hardware platform for you. The disadvantage in this method is that it obscures how the hardware platform gets imported. For consistency, this tutorial will use the same default name that Vivado would have used.

- 7. Insert **hw\_platform\_0** for the *Project name*.
- 8. Click **Browse** and select the System\_wrapper.hdf file generated during the Export process from Vivado. This will be included in the archive provided by the hardware engineer. Or, if you are continuing from the first tutorial, you will find it in a similar location as here:

C:\Avnet\MicroZed\Projects\MZ\_Basic\_System\MZ\_Basic\_System.sdk\

- 9. After selecting System\_wrapper.hdf, click Open.
- 10. The Bitstream is embedded in the HDF, so it is not separately specified here. Click **Finish**.

son New Hardware Project	
New Hardware Project Create a new Hardware Project.	E
Project name: hw_platform_0	
✓ Use default location	
Location: C:\Avnet\MicroZed\Applications\MZ_Basic_System\hw_platform_0 Choose file system: default Target Hardware Specification Provide the path to the hardware specification file exported from Vivado. This file usually resides in SDK/SDK_Export/hw folder relative to the Vivado project location The specification file and associated bitstream content will be copied into the workspace.	Browse
C:\Avnet\MicroZed\Projects\MZ_Basic_System\MZ_Basic_System.sdk\System_wrapper.h	Cancel

Figure 4 – Import Hardware Platform from Vivado

11. Notice the PS7 Zynq hardware platform is now visible in the *Project Explorer*.

27 March 2017



#### Figure 5 – Hardware Platform Imported and Ready for Use

If you select the HDF file, SDK will show you information about the hardware platform (not the HDF raw code itself).

					IP blocks present in the	design	
					ps7_intc_dist_0	ps7_intc_dist	1.00.a
					ps7_gpio_0	ps7_gpio	1.00.a
					ps7 scutimer 0	ps7 scutimer	1.00.a
					ns7 slcr 0	ns7 slcr	1 00 a
					ps7_sici_0	ps/_sici	1.00.0
					ps/_scuwat_0	ps/_scuwar	1.00.4
					ps/_l2cachec_0	ps/_l2cachec	1.00.a
system.hdf 🔀	Address Map for proc	essor ps7 cortexa	9 1		ps7_scuc_0	ps7_scuc	1.00.a
					ps7_qspi_linear_0	ps7_qspi_linear	1.00.a
hw_platform_0 Hardware Platform Spe	ecification Ce	II Base Addr	High Addr	Slave I/f	ps7_pmu_0	ps7_pmu	1.00.a
Design Information	ps7_afi_	0 0xf8008000	0xf8008fff		ps7 afi 1	ps7 afi	1.00.a
	ps7_afi_	1 0xf8009000	0xf8009fff		ns7 asni 0	ns7 asni	1 00 a
Target FPGA Device: 7z010 Created With: Vivada 2016 2	ps7_afi_	2 0xf800a000	0xf800afff		p37_q3pi_0	p37_q3p1	1.00.0
Created On: Thu Sep 15 11:43:23 2016	ps7_afi_	3 0xf800b000	0xf800bfff		ps7_usb_0	ps/_usb	1.00.a
	ps7_coresight_comp_	0 0xf8800000	0xf88fffff		ps/_afi_0	ps/_afi	1.00.a
Address Map for processor ps7_cortexa9_0	ps7_ddr_	0 0x00100000	0x3fffffff		ps7_afi_3	ps7_afi	1.00.a
Cell Base Adat High A	ddr Slave I/f ps7_ddrc_	0 0xf8006000	0xf8006fff		ps7_axi_interconnect_0	ps7_axi_interconnect	1.00.a
ps7_afi_0 0xf8008000 0xf8008f	ff ps7 dev cfg	0 0xf8007000	0xf80070ff		ps7 globaltimer 0	ps7 globaltimer	1.00.a
ps7_afi_1 0xf8009000 0xf8009f	ff ps7 dma n	s 0xf8004000	0xf8004fff		nc7 afi 2	nc7 afi	1 00 a
ps7_afi_2 0xf800a000 0xf800af	ff ps7 dma	• 0xf8003000	0xf8003fff		p37_011_2	p37_dm	1.00.0
ps/_an_3_0x10000000_0x100000	ff ps7 ethernet	0 0xe000b000	OxeOOObfff		ps/_dma_s	ps/_dma	1.00.a
ps7_colesignt_comp_0 0x10000000 0x3fffff	ff ps7_clobaltimer	0 0xf8f00200	Owf8f002ff		ps7_xadc_0	ps7_xadc	1.00.a
ps7_ddrc_0 0xf8006000 0xf8006f	ff ps/_globalanci_	0 0ve000a000	OveOODafff		ps7_iop_bus_config_0	ps7_iop_bus_config	1.00.a
ps7_dev_cfg_0 0xf8007000 0xf80070	)ff ps/_gpio_	0 0xf8900000	Owf89fffff		ps7_ddr_0	ps7_ddr	1.00.a
ps7_dma_ns 0xf8004000 0xf8004f	iff ps/_gpv_	0 0x10500000	0		ps7 pl310 0	ps7 pl310	1.00.a
ps/_dma_s_UX18003000_UX180031	ff ps/_intc_dist_	0 0x10101000	0x10101111		ns7 ddrc 0	ns7 ddrc	1 00 a
ps7 globaltimer 0 0xf8f00200 0xf8f002	2ff ps/_lop_bus_config_	0 0xe0200000	0xe0200111		p37_ddrc_0	p37_dure	1.00.4
ps7_gpio_0 0xe000a000 0xe000af	ff ps/_i2cacnec_	0 0x18102000	0x18102111		ps/_ocmc_u	ps/_ocmc	1.00.a
ps7_gpv_0 0xf8900000 0xf89fff	ff ps/_ocmc_	0 0x1800c000	Ux18UUcfff		ps7_uart_1	ps7_uart	1.00.a
ps7_intc_dist_0 0xf8f01000 0xf8f01f	ff ps7_pl310_	0 0x18102000	Ux18102111		ps7_coresight_comp_0	ps7_coresight_comp	1.00.a
ps/_iop_bus_config_0_0xe0200000_0xe02001	ps7_pmu_	0 0x18893000	0x18893111		ps7_ttc_0	ps7_ttc	1.00.a
ps7 ocmc 0 0xf800c000 0xf800cf	ff ps7_qspi_	0 0xe000d000	0xe000dfff		ps7 cortexa9 1	ps7 cortexa9	5.2
ps7_pl310_0 0xf8f02000 0xf8f02f	ff ps7_qspi_linear_	0 0xfc000000	Oxfcfffff		ns7 scuaic 0	ns7 scuaic	1 00 a
ps7_pmu_0 0xf8893000 0xf8893f	ff ps7_ram_	0 0x00000000	0x0002ffff		ps/_sedgre_o	ps/_scugic	1.00.4
ps7_qspi_0 0xe000d000 0xe000df	ff ps7_ram_	1 0xffff0000	Oxfffffdff		ps/_ethernet_0	ps/_ethernet	1.00.a
ps/_qspi_linear_0_UxiCUUUUUU_UxiCIIII	ps7_scuc_	0 0xf8f00000	0xf8f000fc		processing_system7_0	processing_system7	5.5
ps7 ram 1 0xffff0000 0xfffffd	1ff ps7_scugic_	0 0xf8f00100	0xf8f001ff		ps7_cortexa9_0	ps7_cortexa9	5.2
ps7_scuc_0 0xf8f00000 0xf8f000	ofc ps7_scutimer_	0 0xf8f00600	0xf8f0061f		ps7_clockc_0	ps7_clockc	1.00.a
ps7_scugic_0 0xf8f00100 0xf8f001	lff ps7_scuwdt_	0 0xf8f00620	0xf8f006ff		ps7 dev cfa 0	ps7 dev cfa	1.00.a
ps7_scutimer_0_0xf8f00600_0xf8f006	ps7_sd	0 0xe0100000	0xe0100fff		ns7 dma ns	ns7 dma	1 00 a
ps/_scuwdt_0_0xt8t00620_0xt8t006 ps7_sd_0_0xe010000_0ve0100f	ff ps7 slcr	0 0xf8000000	0xf8000fff		ps/_uma_ms	ps/_uma	1.00.4
ps7_slcr 0 0xf8000000 0xf8000f	ff ps7 ttc	0 0xf8001000	0xf8001fff		ps/_sa_u	ps/_saio	1.00.a
ps7_ttc_0 0xf8001000 0xf8001f	ff ns7 uart	1 0xe0001000	0xe0001fff		ps/_gpv_0	ps/_gpv	1.00.a
ps7_uart_1 0xe0001000 0xe0001f	iff ps7_uch	0 0xe0002000	0xe0002fff		ps7_ram_1	ps7_ram	1.00.a
ps7_usb_0 0xe0002000 0xe0002f	ff ps/_usb_	0 0xf8007100	0xf8007120		ps7_ram_0	ps7_ram	1.00.a
ps/_xadc_0_Uxf800/100_Uxf800/1	L20 ps/_xauc_	- CA1000/100	0410307120		•		

#### Figure 6 – system.hdf Report on Hardware Specification

## Experiment 2: BSP

Next, we will create a bare metal board support package, which Xilinx calls Standalone. This will assemble and compile various drivers that relate to the peripherals in the hardware platform for later use in our applications.

27 March 2017

- 1. In SDK select File  $\rightarrow$  New  $\rightarrow$  Board Support Package.
- 2. Accept the default settings for the standalone BSP OS. Click Finish.

New Board Su	pport Pack	age Project	
ilinx Board Su	ipport Pa	ckage Project	
Create a Board S	Support Pa	ckage.	
Project name:	standalon	e_bsp_0	
Use default	location		
Location: C:\A	vnet\Micr	oZed\Applications\MZ_Basic_System\standalone_bsp_0	owse
Choo	se file syst	em: default 💌	
Target Hardwa	are		
Hardware Plat	form: hw	_platform_0	<ul> <li>New</li> </ul>
CPU:	ps7	_cortexa9_0	-
Compiler:	32-	bit 🔻	
Board Support	t Package (	DS	
<mark>standalone</mark> freertos823_xi	linx	Standalone is a simple, low-level software layer. It provides access to basic p features such as caches, interrupts and exceptions as well as the basic feature hosted environment, such as standard input and output, profiling, abort and	rocessor es of a I exit.
?		Finish	Cancel

#### Figure 7 – Standalone BSP



3. In the *Board Support Package Settings*, accept the default settings. Click **OK**.

ontrol various settings of	your Board Support Pack	age.		11
Overview standalone a drivers ps7_cortexa9_0	standalone_bsp_0 OS Type: stando OS Version: 6.1 Target Hardware Hardware Specifica Processor:	alone ▼ stion: C:\Avnet\Mir ps7_cortexa9	Standalone is a simple, low-level software layer. It provides acce features such as caches, interrupts and exceptions as well as the environment, such as standard input and output, profiling, abo croZed\Applications\MZ_Basic_System\hw_platform_0\system.I 0	ess to basic processor e basic features of a hoste rt and exit. hdf
	-Supported Librarie	s		
	Check the box new navigator on the l	t to the libraries yo eft. Version	u want included in your Board Support Package.You can config	jure the library in the
	Check the box new navigator on the l	t to the libraries yo eft. Version	u want included in your Board Support Package.You can config Description	jure the library in the
	Check the box neo navigator on the l Name ilibmetal	t to the libraries yo eft. Version 1.1 1.7	u want included in your Board Support Package.You can config Description Libmetal Library by D.C.P./D.Stack librarg, by D.d. 4.1	jure the library in the
	Check the box new navigator on the l Name libmetal hvip141	t to the libraries yo eft. Version 1.1 1.7 1.2	u want included in your Board Support Package.You can config Description Libmetal Library IwIP TCP/IP Stack library: IwIP v1.4.1 OpenAmp Library	gure the library in the
	Check the box new navigator on the l Name iibmetal wip141 viffc	t to the libraries yo eft. Version 1.1 1.7 1.2 3.5	u want included in your Board Support Package.You can config Description Libmetal Library IwIP TCP/IP Stack library: IwIP v1.4.1 OpenAmp Library Generic Fat File System Library	jure the library in the
	Check the box new navigator on the l libmetal wip141 openamp xilffs xilffash	t to the libraries yo eft. Version 1.1 1.7 1.2 3.5 4.2	u want included in your Board Support Package.You can config Description Libmetal Library IwIP TCP/IP Stack library: IwIP v1.4.1 OpenAmp Library Generic Fat File System Library Xilinx Flash library for Intel/AMD CFI compliant paral	gure the library in the
	Check the box new navigator on the l Name libmetal vip141 openamp xilffs xilfash xilisf	t to the libraries yo eft. Version 1.1 1.7 1.2 3.5 4.2 5.7	u want included in your Board Support Package.You can config Description Libmetal Library IwIP TCP/IP Stack library: IwIP v1.4.1 OpenAmp Library Generic Fat File System Library Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx In-system and Serial Flash Library	gure the library in the
	Check the box new navigator on the l libmetal libmetal openamp xilffs xilffash xilisf	t to the libraries yo eft. Version 1.1 1.7 1.2 3.5 4.2 5.7 2.2	u want included in your Board Support Package.You can config Description Libmetal Library IwIP TCP/IP Stack library: IwIP v1.4.1 OpenAmp Library Generic Fat File System Library Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx In-system and Serial Flash Library Xilinx Memory File System	gure the library in the
	Check the box new navigator on the l libmetal viip141 openamp xilffs xilfash xilisf xilinfs xilmfs	t to the libraries yo eft. Version 1.1 1.7 1.2 3.5 4.2 5.7 2.2 2.0	u want included in your Board Support Package.You can config Description Libmetal Library IwIP TCP/IP Stack library: IwIP v1.4.1 OpenAmp Library Generic Fat File System Library Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx Memory File System Power Management API Library for ZyngMP	gure the library in the
	Check the box new navigator on the l libmetal viip141 openamp xilffs xilfash xilisf xilisf xilmfs xilmfs xilmfs	t to the libraries yo eft. Version 1.1 1.7 1.2 3.5 4.2 5.7 2.2 2.0 1.2	u want included in your Board Support Package.You can config Description Libmetal Library WIP TCP/IP Stack library: IwIP v1.4.1 OpenAmp Library Generic Fat File System Library Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx Memory File System Power Management API Library for ZynqMP Xilinx RSA Library	gure the library in the
	Check the box new navigator on the l libmetal wip141 openamp xilffs xilfash xilisf xilisf xilirsa xilpm xilrsa xilskey	t to the libraries yo eft. Version 1.1 1.7 1.2 3.5 4.2 5.7 2.2 2.0 1.2 6.1	u want included in your Board Support Package.You can config Description Libmetal Library WIP TCP/IP Stack library: IwIP v1.4.1 OpenAmp Library Generic Fat File System Library Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx In-system and Serial Flash Library Xilinx Memory File System Power Management API Library for ZynqMP Xilinx RSA Library Xilinx Secure Key Library	gure the library in the
	Check the box new navigator on the l libmetal wip141 openamp xilffs xilfash xilfash xilisf xilmfs xilmfs xilms xilma xilrsa xilrsa	t to the libraries yo eft. Version 1.1 1.7 1.2 3.5 4.2 5.7 2.2 2.0 1.2 6.1	u want included in your Board Support Package.You can config Description Libmetal Library WIP TCP/IP Stack library: IwIP v1.4.1 OpenAmp Library Generic Fat File System Library Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx Memory File System Power Management API Library for ZynqMP Xilinx RSA Library Xilinx Secure Key Library	gure the library in the

Figure 8 – Board Support Package Settings

Based on the default settings in SDK, the BSP will automatically be built once it is added to the project. This may take a minute to compile the new BSP. The progress may be seen in the *Console* tab.

# MicroZed/PicoZed: Hello World v2016\_4.01

27 March 2017



🖹 Problems 🖉 Tasks 🖳 Console 🛛 🔲 Properties 📮 SDK Terminal 🕹 🗘 😭 🖓 📳 📓 🗉 🖷 🙀 🛫 🗂 🕶 🖓	
CDT Build Console [standalone hsp 0]	
"Compiling scugic"	
"Running Make libs in ps7 cortexa9 0/libsrc/scutimer v2 1/src"	
make -C ps7 cortexa9 0/libsrc/scutimer v2 1/src -s libs "SHELL=CMD" "COMPILER=arm-xilinx-eabi-gcc" "ARCHIVER=arm-xilinx-	ei
"Compiling scutimer"	
"Running Make libs in ps7 cortexa9 0/libsrc/scuwdt v2 1/src"	
make -C ps7_cortexa9_0/libsrc/scuwdt_v2_1/src -s libs "SHELL=CMD" "COMPILER=arm-xilinx-eabi-gcc" "ARCHIVER=arm-xilinx-ea	b:
"Compiling scuwdt"	
"Running Make libs in ps7_cortexa9_0/libsrc/sdps_v2_6/src"	
make -C ps7_cortexa9_0/libsrc/sdps_v2_6/src -s libs "SHELL=CMD" "COMPILER=arm-xilinx-eabi-gcc" "ARCHIVER=arm-xilinx-eabi	-1
"Compiling sdps"	
"Running Make libs in ps7_cortexa9_0/libsrc/standalone_v5_3/src"	
make -C ps7_cortexa9_0/libsrc/standalone_v5_3/src -s libs "SHELL=CMD" "COMPILER=arm-xilinx-eabi-gcc" "ARCHIVER=arm-xilin	x
"Compiling standalone"	
"Running Make libs in ps7_cortexa9_0/libsrc/ttcps_v3_0/src"	
make -C ps7_cortexa9_0/libsrc/ttcps_v3_0/src -s libs "SHELL=CMD" "COMPILER=arm-xilinx-eabi-gcc" "ARCHIVER=arm-xilinx-eab	i
"Compiling ttcps"	
"Running Make libs in ps7_cortexa9_0/libsrc/uartps_v3_1/src"	
<pre>make -C ps7_cortexa9_0/libsrc/uartps_v3_1/src -s libs "SHELL=CMD" "COMPILER=arm-xilinx-eabi-gcc" "ARCHIVER=arm-xilinx-ea</pre>	b:
"Compiling wartps"	
"Running Make libs in ps7_cortexa9_0/libsrc/usbps_v2_2/src"	. —
make -C ps7_cortexa9_0/libsrc/usbps_v2_2/src -s libs "SHELL=CMD" "COMPILER=arm-xilinx-eabi-gcc" "ARCHIVER=arm-xilinx-eab	i
"Compiling usbps"	
"Running Make libs in ps7_cortexa9_0/libsrc/xadcps_v2_2/src"	
make -C ps/_cortexa9_0/libsrc/xadcps_v2_2/src -s libs "SHELL=CMD" "COMPILER=arm-xilinx-eabi-gcc" "ARCHIVER=arm-xilinx-ea	b:
"Compiling xadcps"	=
'Finished building libraries'	
12:57:20 Puild Sinished (teck 26: 572ms)	
12:57:50 DUILU FINISHEU (COOK 205.525MS)	
	-

Figure 9 – BSP Building

The standalone\_bsp\_0 is now visible in the *Project Explorer*.

4. Expand **standalone\_bsp\_0** under the *Project Explorer*.



Figure 10 – BSP Added to the Project



## **Experiment 3: Add Application**

With a Hardware Platform and BSP, we are now ready to add an application and run something on the board.

- 1. In SDK, select File  $\rightarrow$  New  $\rightarrow$  Application Project.
- 2. In the **Project Name** field type in Hello\_Zed. Change the **BSP** to the existing StandAlone BSP. Click **Next** >.

New Project				
Application Project				
Create a managed make application project.				
1				
Project name: Hello_Zed				
Use default location				
Location: C:\Avnet\MicroZed\Applications\MZ_Basic_System\Hello_Z( Browse				
Choose file system: default				
OS Platform: standalone 🗸				
Target Hardware				
Processor: ps7_cortexa9_0				
Target Software				
Language: O C C++				
Board Support Package: O Create New Hello_Zed_bsp				
2  Our Standalone_bsp_0				
3				
(?) < Back Next > Finish Cancel				

Figure 11 - New Application Wizard



3. Select Hello World from the Available Templates field. Click Finish.

SOK New Project	
Templates Create one of the available templates to gene application project.	rate a fully-functioning
Available Templates: Dhrystone Empty Application Hello World IwIP Echo Server Memory Tests OpenAMP echo-test OpenAMP matrix multiplication Demo OpenAMP RPC Demo Peripheral Tests RSA Authentication App Zynq DRAM tests Zynq FSBL	Let's say 'Hello World' in C.
Reck Network	ext > Finish Cancel

Figure 12 – New Application Project: Hello World

4. Notice that the Hello\_Zed application is now visible in *Project Explorer*. By default, SDK will build the application automatically after it is added.





# MicroZed/PicoZed: Hello World v2016\_4.01

27 March 2017



🖹 Problems 🖉 Tasks 📃 Console 🛛 🔲 Properties 🖳 SDK Terminal CDT Build Console [Hello\_Zed] 13:00:44 \*\*\*\* Build of configuration Debug for project Hello\_Zed \*\*\*\* make all 'Building file: ../src/helloworld.c' 'Invoking: ARM gcc compiler' arm-xilinx-eabi-gcc -Wall -O0 -g3 -c -fmessage-length=0 -MT"src/helloworld.o" -I../../standalone\_bsp\_0/ps7\_cortexa9\_0/incl 'Finished building: ../src/helloworld.c' 'Building file: ../src/platform.c' 'Invoking: ARM gcc compiler' arm-xilinx-eabi-gcc -Wall -O0 -g3 -c -fmessage-length=0 -MT"src/platform.o" -I../../standalone\_bsp\_0/ps7\_cortexa9\_0/include 'Finished building: ../src/platform.c' 'Building target: Hello\_Zed.elf' 'Invoking: ARM gcc linker' arm-xilinx-eabi-gcc -Wl,-T -Wl,../src/lscript.ld -L../../standalone\_bsp\_0/ps7\_cortexa9\_0/lib -o "Hello\_Zed.elf" ./src/hel 'Finished building target: Hello\_Zed.elf' 'Invoking: ARM Print Size' arm-xilinx-eabi-size Hello\_Zed.elf |tee "Hello\_Zed.elf.size" text data bss dec hex filename 22888 1152 22564 46604 b60c Hello\_Zed.elf 22888 'Finished building: Hello\_Zed.elf.size' 13:00:46 Build Finished (took 1s.300ms) •

Figure 14 – Hello World Application Automatically Built



## **Experiment 4: Run on Hardware**

- 1. Set the Boot Mode jumpers to Cascaded JTAG Mode.
  - a. MicroZed: MIO[5:2] = GND. Set JP3, JP2, and JP1 to positions 1-2.



Figure 15 – Cascaded JTAG Boot Mode on MicroZed

b. PicoZed: Set both switches on SW1 on the SOM <u>away</u> from the SW1 silkscreen



Figure 16 - PicoZed SW1 Set to Cascaded JTAG Boot Mode



Neither MicroZed nor PicoZed has on-board USB JTAG programming. Thus it requires an external JTAG programmer, such as the Digilent HS3.

- 2. Connect a Platform Cable or Digilent Programming cable from your PC to the 2x7 JTAG socket.
  - a. MicroZed:
    - i. Use J3
  - b. PicoZed FMC Carrier V1:
    - i. Use J12 PC4\_JTAG
  - c. PicoZed FMC Carrier V2:
    - i. Use J7 JTAG
- 3. Power the board and connect a USB cable from your PC to the USB-UART port.
  - a. MicroZed:
    - i. Plug the micro-USB into J2.
    - ii. The USB cable will power MicroZed. You should see the Green Power Good LED (D5) and the Red User LED (D3) light.



Figure 17 – MicroZed Powered and Connected to Digilent HS2 and USB-UART



- b. PicoZed:
  - i. Make sure the PZCC-FMC power switch (SW7) is OFF.
  - ii. Insert the PicoZed module onto the PZCC-FMC.
  - iii. Set the on-board jumpers as follows
    - 1. JP1 is open
    - 2. JP3 is closed in position 1-2
    - 3. JP4 is closed
    - 4. JP6 is open
    - 5. J9 is closed in positions 3-5 and 4-6
    - 6. CON2 is open, which sets V\_ADJ to 1.8V
  - iv. Insert the appropriate country plug into the 12V AC/DC adapter.
     Plug it into the J14 2x3 power connector. (NOTE this 2x3 connector is NOT compatible with ATX power supplies.)
  - v. Turn the PZCC-FMC power switch (SW7) to the ON position.
  - vi. Plug in the micro-USB cable to PZCC-FMC USB-UART port (J6). (The reason for waiting until AFTER power is applied to the board is explained in the <u>PZCC-FMC Errata</u>.)
  - vii. After 1-2 seconds, you will notice five LEDs that are lit:
    - a. D1 (green) on PicoZed, indicating Power Good
    - b. D19 (green) on PZCC-FMC, indicating Vin is on
    - c. D14 (green) on PZCC-FMC, which is the PG\_MODULE handshake between the SOM and the Carrier indicating that the SOM power is good
    - d. D21 (blue) on PZCC-FMC indicating that the Zynq PL configuration is DONE
    - e. D6 (amber) indicating the USB-UART is connected



Figure 18 – PicoZed / PZCC-FMC Powered On with LEDs

www.microzed.org



If this is the first time you've connected the MicroZed, PicoZed, and/or the JTAG cable to this computer, you may see Windows install device drivers for the USB-UART and/or the JTAG cable. You should have previously installed the driver for the Silicon Labs CP2104 USB-UART. The Platform Cable and Digilent USB-JTAG drivers were installed during the Xilinx tool installation.

- Use Device Manager to determine the COM port for the Silicon Labs CP201x USB-UART. In Windows 7, click Start → Control Panel, and then click Device Manager. Click Yes to confirm.
- 5. Expand *Ports*. Note the COM port number for the SiLabs Serial device. This example shows COM4.

Ports (COM & LPT)
 Intel(R) Active Management Technology - SOL (COM3)
 Silicon Labs CP210x USB to UART Bridge (COM4)

Figure 19 – Find the COM port number for the SiLabs USB-UART device

6. Open a serial communication utility for the COM port assigned on your system. SDK provides a serial terminal utility. See the SDK Terminal tab in the center bottom window.

E	Problems	🙆 Tasks	📮 Console	Properties	🖳 SDK Terminal	53	+	х	B_	
	Click on + bu	tton to add	l a port to the	terminal.						
										*
	1									

Figure 20 – Terminal Window Header Bar



- Click the button to open the Terminal Settings dialog box.
   Change the settings as shown below. Click OK.

sok Connect to s	erial port					
-Basic Setting	Basic Settings					
Port:	Port: COM6 🔻					
Baud Rate: 1	15200 👻					
▼ Advance S	Settings					
Data Bits:	8 🔻					
Stop Bits:	1 -					
Parity:	None 🔻					
Flow Contro	l: None 🔻					
Timeout (see	c): 5					
ОК	Cancel					

Figure 21 – Terminal Settings Dialog Box



Program the PL first by clicking the <sup>♣</sup> icon or selecting Xilinx Tools → Program FPGA. The default options are acceptable. Click Program. When complete, the Blue DONE LED should light.

Sok Program FPGA						
Program FPGA						
Specify the bitstream and the ELF files that reside in BRAM memory						
Hardware Configuration						
Hardware Platform:	hw_platform_0					
Connection:	Local		▼ New	]		
Device:	Auto Detect Select			]		
Bitstream:	System_wrapper.bit Search Browse			Browse		
Partial Bitstream	Partial Bitstream					
BMM/MMI File:			Search	Browse		
Software Configuration						
Processor		ELF/MEM File to Initialize	in Block RAM			
?		Pro	ogram	Cancel		

Figure 22 – Program FPGA

10. Right-click on the Hello\_Zed application and select **Run As** → 1 Launch on Hardware (System Debugger).

	Run As		E TEF	1 Launch on Hardware (System Debugger)
	Compare With Restore from Local History		₩ 	2 Start Performance Analysis 3 Launch on Hardware (System Debugger on QEMU)
	Create Boot Image Change Referenced BSP	3	SDB C	4 Launch on Hardware (GDB) 5 Local C/C++ Application
5	Generate Linker Script			Run Configurations

Figure 23 – Launch on Hardware (GDB)



11. The tools will now initialize the processor, download the Hello\_Zed.elf to DDR, and then run Hello\_Zed. This takes a few seconds to complete, depending on the USB traffic in your system. You can follow the progress in the lower right corner of SDK.



Figure 24 – Launching Hello\_Zed Progress

SDK will download the Hello World ELF to the DDR3, and the ARM cpu0 begins executing the code. On MicroZed, you will notice that the Red User LED will go out, which is expected since that GPIO is now properly configured without a pull-up. The application standard output is displayed in the SDK Terminal. If SDK automatically switches to the *Console* tab, click on the *Terminal* tab to see the output.





Figure 25 – Hello Zed Complete

You have now booted Zynq hardware on MicroZed or PicoZed! The Terminal can be disconnected by clicking the solution.

## **Revision History**

Date	Version	Revision
23 Aug 2013	2013_2.01	Initial Avnet release for Vivado 2013.2
09 Jun 2014	2014_1.01	Update to 2014.1
11 Jun 2014	2014_2.01	Update to 2014.2
29 Jun 2015	2015_1.01	Update to 2015.1. Add support for PicoZed.
15 Jul 2015	2015_2.01	Update to 2015.2.
06 Apr 2016	2015_4.01	Update to 2015.4. Add support for PZCC-FMC-V2.
01 Jun 2016	2015_4.02	Update to 2015.4. Add picoZed JTAG Boot picture.
29 Aug 2016	2015_4.03	Clarified JTAG port for PZ FMC Carrier Card V2
09 Sept 2016	2016_2.01	Updated to 2016.2
20 Jan 2017	2016_4.01	Updated to 2016.4