

Overview

With the Hello World application operational, we will now move on to more advanced test applications. Xilinx provides a Memory Test as well as a Peripherals Test in the built-in templates for example applications.

This Tutorial assumes that you have already completed the Hardware Platform and Hello World tutorials. Your starting point will be the SDK project after the Hello World tutorial is complete.

Objectives

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

- Add the Memory Test application
- Add the Peripherals Test application
- Run both test applications
- Edit the memory test to increase the test range

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
 - www.microzed.org → 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 memory¹:
 - 1.6 GB RAM available for the Xilinx tools to complete a XC7Z010 design
 - 2.3 GB RAM available for the Xilinx tools to complete a XC7Z015 design
 - 1.9 GB RAM available for the Xilinx tools to complete a XC7Z020 design
 - 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, Digilent HS1, HS2, or HS3 cable)
 - If you don't already have a JTAG Cable, Avnet recommends the Digilent HS3 Cable
 - <http://www.em.avnet.com/itaghs3>

¹ Refer to www.xilinx.com/design-tools/vivado/memory.htm

Experiment 1: Create Memory and Peripherals Test Applications

Similar to Hello World, use templates to create two very useful test applications.

1. Launch SDK and open the workspace from the Hello World project.
2. In SDK, select **File** → **New** → **Application Project**.
3. In the **Project Name** field type in `Mem_Test`. Change the **BSP** to the existing StandAlone BSP. Click **Next** >.

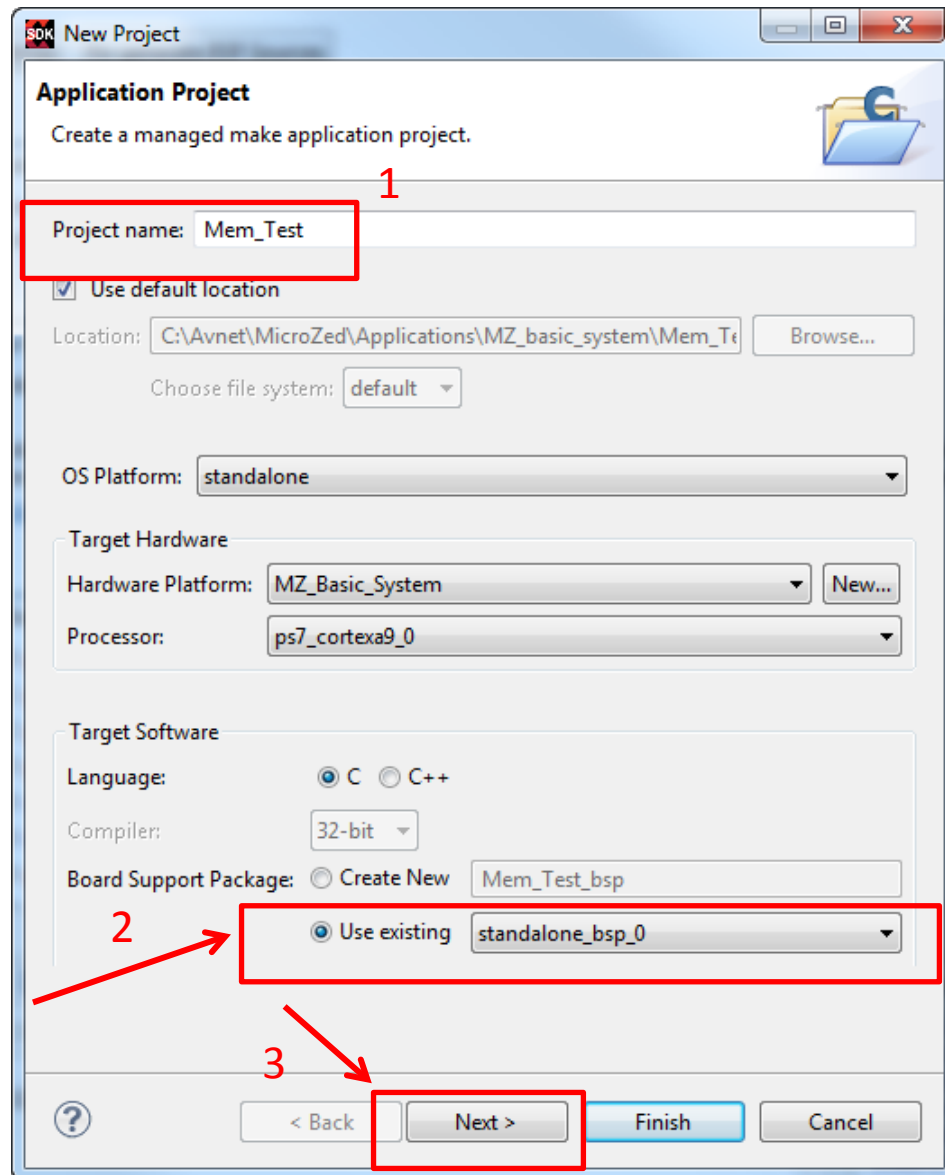


Figure 1 - New Application Wizard

4. Select **Memory Tests** from the *Available Templates* field. Click **Finish**.

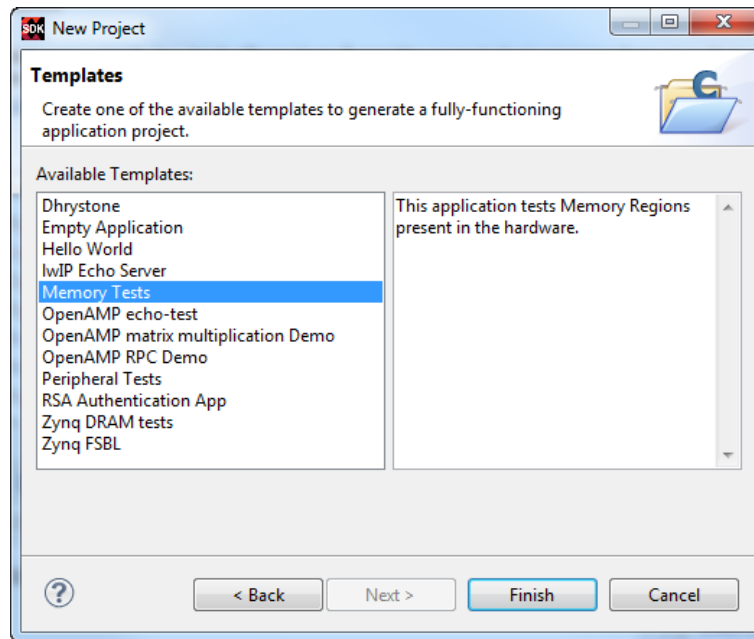


Figure 2 – New Application Project: Hello World

5. Repeat steps 2 through 4 with the following options:
 - a. Project Name = **Periph_Test**
 - b. BSP = **standalone_bsp_0**
 - c. Template = **Peripheral Tests**
6. Repeat steps 2 through 4 with the following options:
 - a. Project Name = **ZynqDRAM_Test**
 - b. BSP = **standalone_bsp_0**
 - c. Template = **Zynq DRAM tests**

When complete, *Project Explorer* should look similar to below.

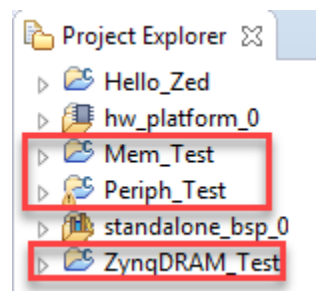


Figure 3 – Project Explorer with New Apps Highlighted

Experiment 2: Run the Applications

1. Follow the instructions in the Hello World tutorial to configure the MicroZed or PicoZed hardware for Cascaded JTAG and plug in the JTAG and USB-UART cables. Make sure to also program the bitstream so that the Blue DONE LED is lit.
2. Continue by right-clicking on the Mem_Test and Periph_Test applications selecting **Run As...**, as previously shown in the Hello World tutorial.
3. When asked to terminate the old configuration, select **Yes**.

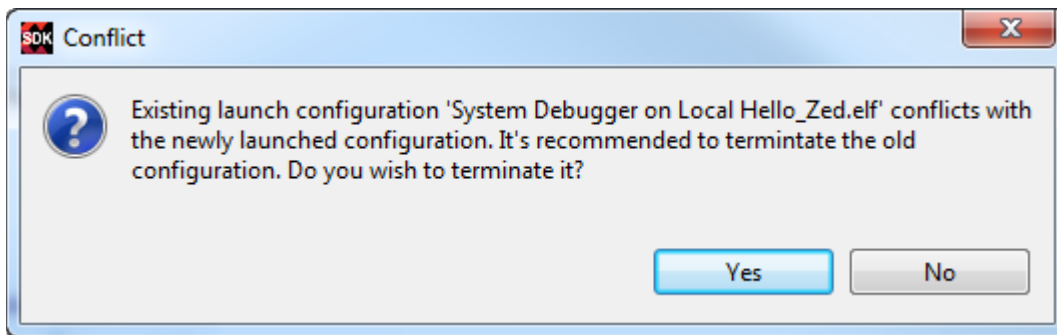
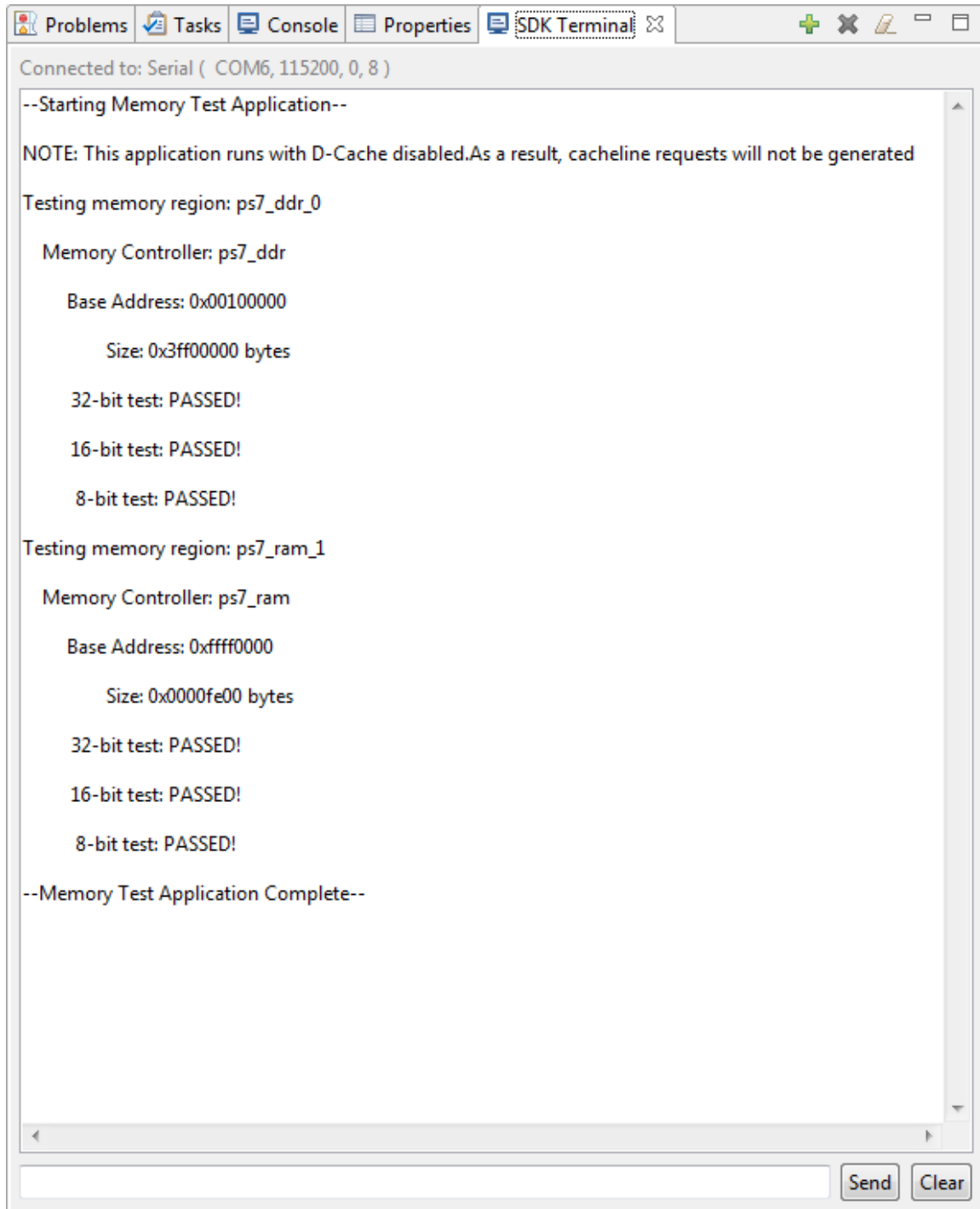


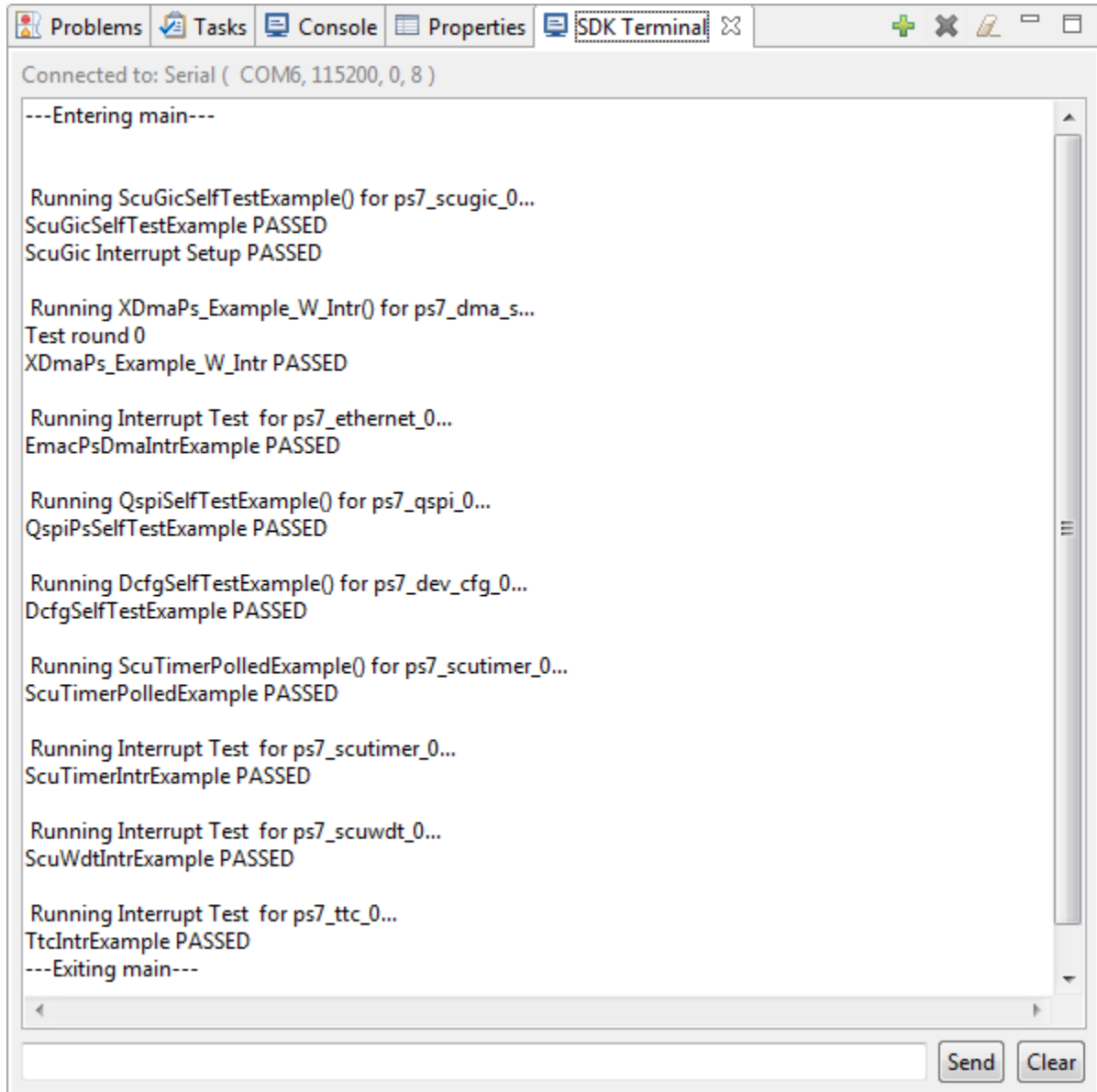
Figure 4 – Terminate Old Configuration

When done you should see these terminal messages.



```
Connected to: Serial ( COM6, 115200, 0, 8 )
--Starting Memory Test Application--
NOTE: This application runs with D-Cache disabled.As a result, cacheline requests will not be generated
Testing memory region: ps7_dds_0
Memory Controller: ps7_dds
Base Address: 0x00100000
Size: 0x3ff00000 bytes
32-bit test: PASSED!
16-bit test: PASSED!
8-bit test: PASSED!
Testing memory region: ps7_ram_1
Memory Controller: ps7_ram
Base Address: 0xffff0000
Size: 0x0000fe00 bytes
32-bit test: PASSED!
16-bit test: PASSED!
8-bit test: PASSED!
--Memory Test Application Complete--
```

Figure 5 - Memory Test Console



The screenshot shows the 'SDK Terminal' window of an IDE. The title bar includes tabs for 'Problems', 'Tasks', 'Console', 'Properties', and 'SDK Terminal'. The terminal text indicates a connection to a serial port (COM6, 115200, 0, 8) and displays the output of a test application. The output shows a series of tests for different peripherals, all of which passed. The tests include ScuGic, XDmaPs, Interrupt, Qspi, Dcfg, ScuTimer, and Ttc. The terminal window has a scrollbar on the right and a 'Send' and 'Clear' button at the bottom right.

```
Connected to: Serial ( COM6, 115200, 0, 8 )

---Entering main---

Running ScuGicSelfTestExample() for ps7_scugic_0...
ScuGicSelfTestExample PASSED
ScuGic Interrupt Setup PASSED

Running XDmaPs_Example_W_Intr() for ps7_dma_s...
Test round 0
XDmaPs_Example_W_Intr PASSED

Running Interrupt Test for ps7_ethernet_0...
EmacPsDmaIntrExample PASSED

Running QspiSelfTestExample() for ps7_qspi_0...
QspiPsSelfTestExample PASSED

Running DcfgSelfTestExample() for ps7_dev_cfg_0...
DcfgSelfTestExample PASSED

Running ScuTimerPolledExample() for ps7_scutimer_0...
ScuTimerPolledExample PASSED

Running Interrupt Test for ps7_scutimer_0...
ScuTimerIntrExample PASSED

Running Interrupt Test for ps7_scuwdt_0...
ScuWdtIntrExample PASSED

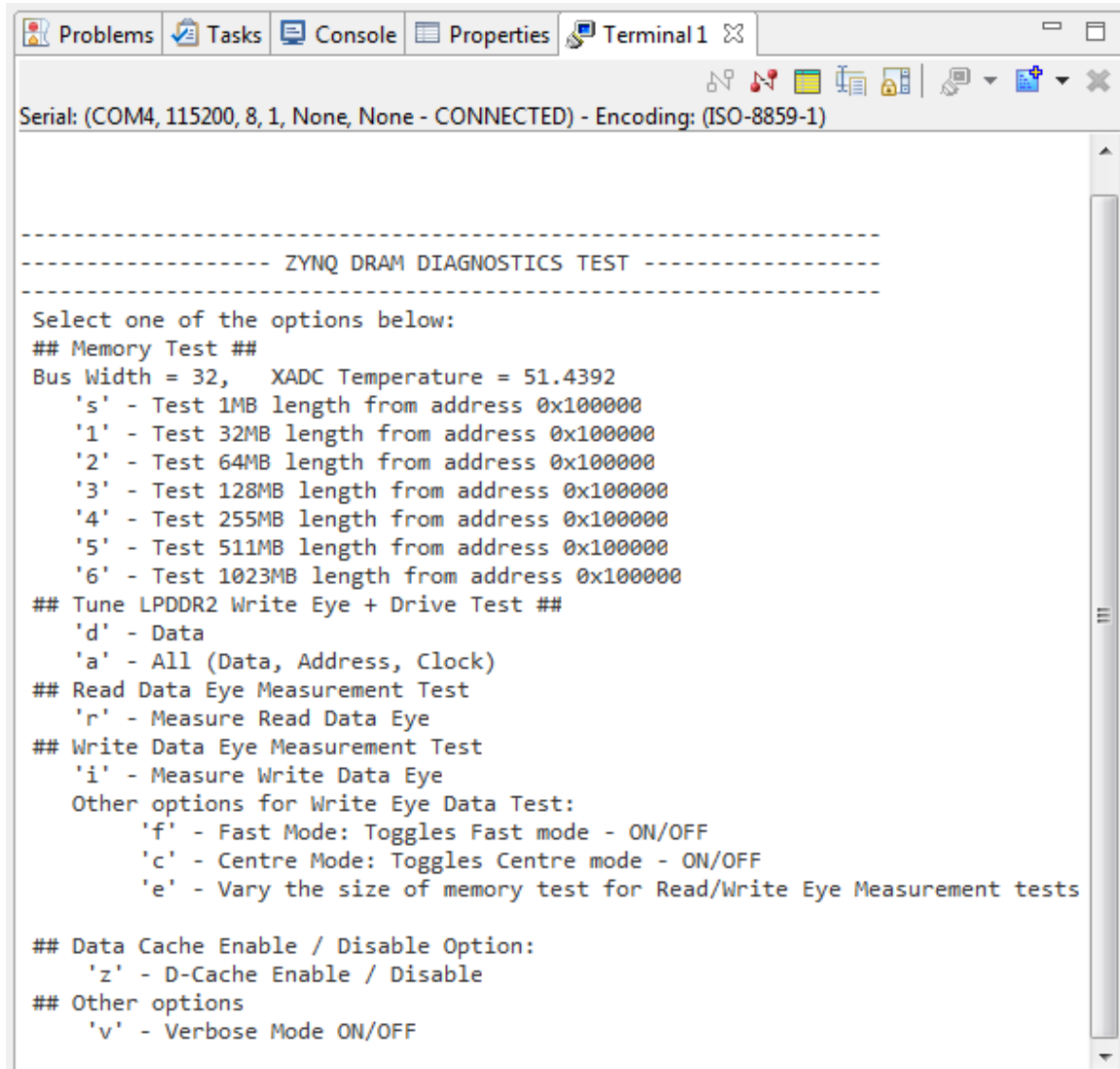
Running Interrupt Test for ps7_ttc_0...
TtcIntrExample PASSED
---Exiting main---
```

Figure 6 - Peripheral Test Console

The Zynq DRAM Test is a bit more complex. It is explained in detail in the **ZYNQ_DRAM_DIAGNOSTICS_TEST.docx** document that is included in the following directory:

C:\Avnet\MicroZed\Applications\MZ_Basic_System\ZynqDRAM_Test\src

A couple of the test outputs are shown below for a MicroZed 7010 Rev F01.



```
-----  
----- ZYNQ DRAM DIAGNOSTICS TEST -----  
-----  
Select one of the options below:  
## Memory Test ##  
Bus Width = 32,  XADC Temperature = 51.4392  
's' - Test 1MB length from address 0x100000  
'1' - Test 32MB length from address 0x100000  
'2' - Test 64MB length from address 0x100000  
'3' - Test 128MB length from address 0x100000  
'4' - Test 255MB length from address 0x100000  
'5' - Test 511MB length from address 0x100000  
'6' - Test 1023MB length from address 0x100000  
## Tune LPDDR2 Write Eye + Drive Test ##  
'd' - Data  
'a' - All (Data, Address, Clock)  
## Read Data Eye Measurement Test  
'r' - Measure Read Data Eye  
## Write Data Eye Measurement Test  
'i' - Measure Write Data Eye  
Other options for Write Eye Data Test:  
'f' - Fast Mode: Toggles Fast mode - ON/OFF  
'c' - Centre Mode: Toggles Centre mode - ON/OFF  
'e' - Vary the size of memory test for Read/Write Eye Measurement tests  
  
## Data Cache Enable / Disable Option:  
'z' - D-Cache Enable / Disable  
## Other options  
'v' - Verbose Mode ON/OFF
```

Figure 7 – Zynq DRAM Diagnostics Test Menu

Option Selected : 1

Starting Memory Test '1' - Testing 32MB length from address 0x100000...

TEST	WORD ERROR COUNT	PER-BYTE-LANE ERROR COUNT				TIME (sec)
		[LANE-0]	[LANE-1]	[LANE-2]	[LANE-3]	
Memtest_0 (0: 0)	0	[0]	[0]	[0]	[0]	1.10946
Memtest_s (0: 1)	0	[0]	[0]	[0]	[0]	0.731382
Memtest_s (0: 2)	0	[0]	[0]	[0]	[0]	0.731382
Memtest_s (0: 3)	0	[0]	[0]	[0]	[0]	0.731382
Memtest_s (0: 4)	0	[0]	[0]	[0]	[0]	0.731382
Memtest_s (0: 5)	0	[0]	[0]	[0]	[0]	0.731382
Memtest_s (0: 6)	0	[0]	[0]	[0]	[0]	0.731382
Memtest_s (0: 7)	0	[0]	[0]	[0]	[0]	0.731382
Memtest_s (0: 8)	0	[0]	[0]	[0]	[0]	0.731382
Memtest_p (0: 9)	0	[0]	[0]	[0]	[0]	1.10592
Memtest_p (0:10)	0	[0]	[0]	[0]	[0]	1.10238
Memtest_l (0:11)	0	[0]	[0]	[0]	[0]	1.07761
Memtest_l (0:12)	0	[0]	[0]	[0]	[0]	1.07761
Memtest_l (0:13)	0	[0]	[0]	[0]	[0]	1.07761
Memtest_l (0:14)	0	[0]	[0]	[0]	[0]	1.07761

Figure 8 – Test #1, 32MB test

Option Selected : r

Running Read Eye Measurement now ...

TEST	WORD ERROR COUNT	PER-BYTE-LANE ERROR COUNT				TIME (sec)
		[LANE-0]	[LANE-1]	[LANE-2]	[LANE-3]	
Test offset 64	0	[0]	[0]	[0]	[0]	0.103219
Test offset 68	0	[0]	[0]	[0]	[0]	0.103219
Test offset 72	0	[0]	[0]	[0]	[0]	0.103219
Test offset 76	0	[0]	[0]	[0]	[0]	0.102629
Test offset 80	0	[0]	[0]	[0]	[0]	0.103219
Test offset 84	0	[0]	[0]	[0]	[0]	0.103219
Test offset 88	0	[0]	[0]	[0]	[0]	0.103219
Test offset 92	0	[0]	[0]	[0]	[0]	0.103219
Test offset 96	6629	[0]	[0]	[0]	[6629]	0.103809
Test offset 100	21945	[0]	[0]	[0]	[21945]	0.105578
Test offset 104	167779	[3700]	[28103]	[0]	[154158]	0.122094
Test offset 108	354211	[28556]	[116049]	[1185]	[332086]	0.143327
Test offset 60	0	[0]	[0]	[0]	[0]	0.103219
Test offset 56	0	[0]	[0]	[0]	[0]	0.103219
Test offset 52	0	[0]	[0]	[0]	[0]	0.103219
Test offset 48	0	[0]	[0]	[0]	[0]	0.102629
Test offset 44	0	[0]	[0]	[0]	[0]	0.103219
Test offset 40	0	[0]	[0]	[0]	[0]	0.103219
Test offset 36	0	[0]	[0]	[0]	[0]	0.103219
Test offset 32	0	[0]	[0]	[0]	[0]	0.102629
Test offset 28	0	[0]	[0]	[0]	[0]	0.102629
Test offset 24	0	[0]	[0]	[0]	[0]	0.102629
Test offset 20	0	[0]	[0]	[0]	[0]	0.102629
Test offset 16	0	[0]	[0]	[0]	[0]	0.102629
Test offset 12	115873	[101190]	[0]	[171]	[86632]	0.116195
Test offset 8	143934	[132762]	[183]	[53348]	[115630]	0.120324

Read Eye Result:

[128 units = 1 bit time (ideal eye width)]

Description	LANE-0	LANE-1	LANE-2	LANE-3
EYE [MIN-MAX] :	[16,100]	[12,100]	[16,104]	[16, 92]
EYE CENTER :	58/128	56/128	60/128	54/128
EYE WIDTH :	65.62%	68.75%	68.75%	59.38%

Figure 9 – Measure Read Data Eye

```
Option Selected : i


Running Write Eye Measurement now ...
** read all ddrc regs
** read all ddriob regs
```

TEST	WORD ERROR COUNT	PER-BYTE-LANE ERROR COUNT				TIME (sec)
		[LANE-0]	[LANE-1]	[LANE-2]	[LANE-3]	
Test offset 64	0	[0]	[0]	[0]	[0]	0.103219
Test offset 68	0	[0]	[0]	[0]	[0]	0.103219
Test offset 72	0	[0]	[0]	[0]	[0]	0.103219
Test offset 76	0	[0]	[0]	[0]	[0]	0.103219
Test offset 80	0	[0]	[0]	[0]	[0]	0.103219
Test offset 84	0	[0]	[0]	[0]	[0]	0.103219
Test offset 88	0	[0]	[0]	[0]	[0]	0.103219
Test offset 92	0	[0]	[0]	[0]	[0]	0.103219
Test offset 96	0	[0]	[0]	[0]	[0]	0.103219
Test offset 100	0	[0]	[0]	[0]	[0]	0.103219
Test offset 104	2220	[7]	[0]	[0]	[2220]	0.103219
Test offset 108	36880	[11730]	[1]	[1683]	[35041]	0.108528
Test offset 60	0	[0]	[0]	[0]	[0]	0.103219
Test offset 56	0	[0]	[0]	[0]	[0]	0.103219
Test offset 52	0	[0]	[0]	[0]	[0]	0.103219
Test offset 48	0	[0]	[0]	[0]	[0]	0.103219
Test offset 44	0	[0]	[0]	[0]	[0]	0.103219
Test offset 40	0	[0]	[0]	[0]	[0]	0.103219
Test offset 36	0	[0]	[0]	[0]	[0]	0.103219
Test offset 32	0	[0]	[0]	[0]	[0]	0.103219
Test offset 28	0	[0]	[0]	[0]	[0]	0.103219
Test offset 24	0	[0]	[0]	[0]	[0]	0.103219
Test offset 20	0	[0]	[0]	[0]	[0]	0.103219
Test offset 16	1276	[0]	[0]	[0]	[1276]	0.103219
Test offset 12	88858	[14295]	[12951]	[1001]	[82967]	0.106758

```
Write Eye Result:
[128 units = 1 bit time (ideal eye width)]
```

Description	LANE-0	LANE-1	LANE-2	LANE-3
EYE [MIN-MAX] :	[16,100]	[16,104]	[16,104]	[20,100]
EYE CENTER :	58/128	60/128	60/128	60/128
EYE WIDTH :	65.62%	68.75%	68.75%	62.50%
EYE ADJUSTED :	0	0	0	0

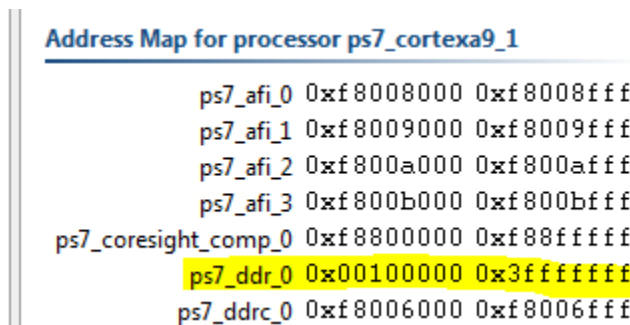
Figure 10 – Measure Write Data Eye

- When finished with the Zynq DRAM Test, click  to disconnect the terminal. Switch to the *Console* tab.

Experiment 3: Edit Memory Test to Expand the Range

MicroZed and PicoZed contain 1 GB of DDR3 RAM, configured as 256M x 32-bits. You may have noticed that the Memory Test application actually runs three different memory tests – 32-bit, 16-bit, and 8-bit. These tests completed very quickly, which should be an indication that the entire memory range was not tested.

1. Open the system.hdf in the hw_platform_0 to investigate the memory map for the DDR.



ps7_afi_0	0xf8008000	0xf8008fff
ps7_afi_1	0xf8009000	0xf8009fff
ps7_afi_2	0xf800a000	0xf800afff
ps7_afi_3	0xf800b000	0xf800bfff
ps7_coresight_comp_0	0xf8800000	0xf88fffff
ps7_ddr_0	0x00100000	0x3fffffff
ps7_ddrc_0	0xf8006000	0xf8006fff

Figure 11 – DDR3 Memory Map

Notice that the address range is 0x00100000 to 0x3fffffff, which is 0x3FF00000 or 1,072,693,248 bytes. (For an explanation on where the lowest 1 MB of DDR3 went, see the Zynq TRM, *On-Chip Memory (OCM)*.)

2. Browse to the C source code for the Memory Test application in the *Project Explorer* at Mem_Test → src

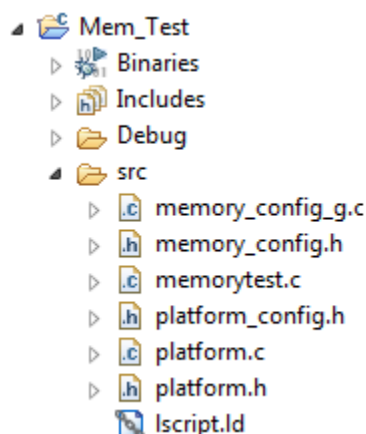


Figure 12 – Memory Test C Sources

3. The main() function is located in memorytest.c. Open that source by double-clicking it.

In `main()`, a for loop iterates on a variable `n_memory_ranges` to run function `test_memory_range`. The `n_memory_ranges` will allow this application to test both the on-chip-memory (OCM) for `cpu1` as well as the DDR3. The `cpu0` OCM is not tested as that is the memory used to store and execute the application (as shown in source `Iscrip.td`).

Looking up further in the file, you will notice the `test_memory_range()` function. To make it easier to reference code, we'll turn on line numbers now.

4. Turn on line numbers by right-clicking in the left-hand column, or use the **Window** → **Preferences** dialog. Go to **General** → **Editors** → **Text Editors** and then check the box for *Show line numbers*. Click **OK**.

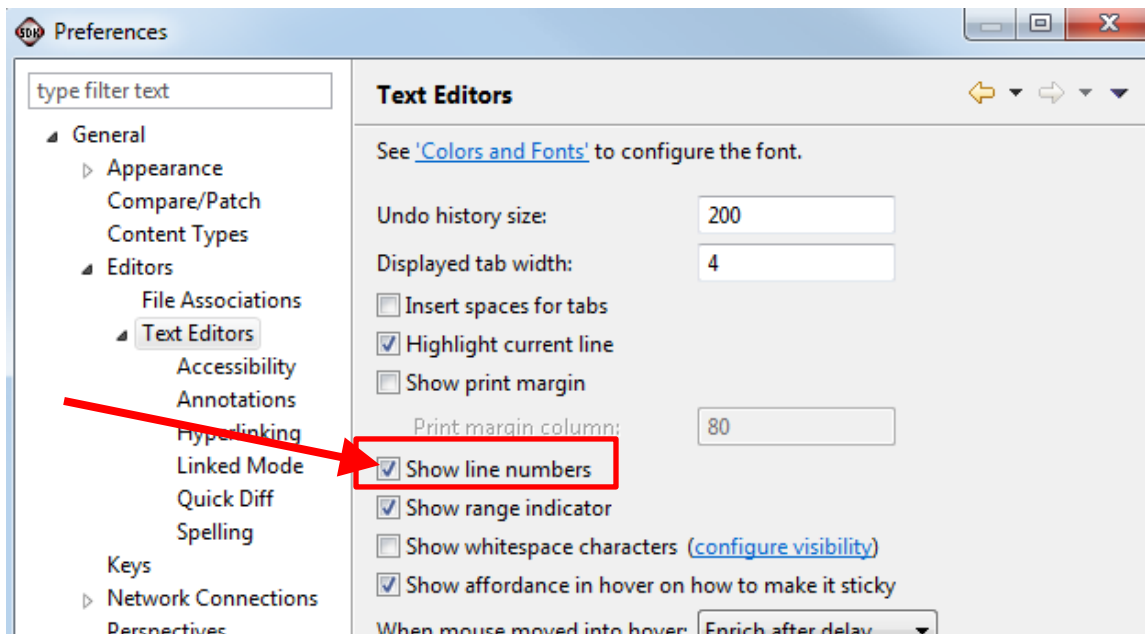


Figure 13 – Show Line Numbers


5. Find lines 79, 82, and 85. You will see that the default function only tests the first 4K bytes:
 - 1024 locations in the 32-bit (4-bytes) test
 - 2048 locations in the 16-bit (2-bytes) test
 - 4096 locations in the 8-bit (1 byte) test

Since the cpu0 OCM is used to execute the code, there is no consequence to testing the entire range other than it will take much, much longer. We will change this to test the full DDR3. However, remember that this function is universally used to test 4KB on both cpu1 OCM and DDR3. Since cpu1 OCM doesn't have 1 GB, if we just change the range in the test function, it will cause the OCM to fail. Therefore, we will change the test to test only DDR3, and we will extend the range.

6. Open memory_config_g.c, which defines the memory_range_s structure.
7. Comment out lines 12 through 17. This can easily be done by selecting the range with your mouse then using Ctrl / on your keyboard.
8. Change the n_memory_ranges to 1.
9. Return to memorytest.c. Make the following edits:
 - Line 79: replace 1024 with 1072693248/4
 - Line 82: replace 2048 with 1072693248/2
 - Line 85: replace 4096 with 1072693248

```
79 status = Xil_TestMem32((u32*)range->base, 1072693248/4, 0xAAAA5555, XIL_TESTMEM_ALLMEMTESTS);
80 print("      32-bit test: "); print(status == XST_SUCCESS? "PASSED!":"FAILED!"); print("\n\r");
81
82 status = Xil_TestMem16((u16*)range->base, 1072693248/2, 0xAA55, XIL_TESTMEM_ALLMEMTESTS);
83 print("      16-bit test: "); print(status == XST_SUCCESS? "PASSED!":"FAILED!"); print("\n\r");
84
85 status = Xil_TestMem8((u8*)range->base, 1072693248, 0xA5, XIL_TESTMEM_ALLMEMTESTS);
86 print("      8-bit test: "); print(status == XST_SUCCESS? "PASSED!":"FAILED!"); print("\n\r");
```

Figure 14 – Modified Memory Test

10. Save all files using the  icon, which will cause a re-build.
11. In the Console, notice the size of the application is ~44KB. Since cpu0 OCM has 196K useable bytes, we are well within the limits.

```
'Invoking: ARM v7 Print Size'
arm-none-eabi-size Mem_Test.elf |tee "Mem_Test.elf.size"
  text    data    bss     dec      hex filename
 28480    1176    14388   44044   ac0c Mem_Test.elf
```

Figure 15 – Mem_Test Built

12. Reconnect the terminal and re-run this edited and newly built Mem_Test. Be patient as the test times are significantly longer.
 - 32-bit test: ~1:45
 - 16-bit test: ~3:20
 - 8-bit test: ~6:20

Total elapsed time will be about 11.5 minutes.

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 for Vivado 2014.1
11 Jun 2014	2014_2.01	Update for Vivado 2014.2
29 Jun 2015	2015_1.01	Update for Vivado 2015.1. Added support for PicoZed.
15 Jul 2015	2015_2.01	Update for Vivado 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. Clarified terminal disconnect function.
15 Sept 2016	2016_2.01	Updated to 2016.2
20 Jan 2017	2016_4.01	Updated to 2016.4