

## KeyStone 1

# HyperLink Self Test Kit

## User's Guide

### Revision history

Version	Date	Author	Description of Change
1.0	December 15, 2013	Brighton Feng	Initial release

Preliminary

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>3</b>
<b>2</b>	<b>Test Methods.....</b>	<b>3</b>
2.1	Integrity Test.....	3
2.1.1	Data filling Test .....	3
2.1.2	Addressing test .....	4
2.2	HyperLink Performance Test .....	4
2.2.1	HyperLink Latency Test .....	5
2.2.2	HyperLink Throughput Test.....	5
2.3	HyperLink error reporting and checking .....	6
<b>3</b>	<b>Test cases.....</b>	<b>6</b>
3.1	Test by different masters .....	6
3.2	Test through different data path.....	6
3.3	Test different memories in the system .....	7
<b>4</b>	<b>Test code/project .....</b>	<b>8</b>
4.1	CCS project.....	9
4.2	To run the test program on KeyStone 1 EVM .....	10
4.3	Test configuration .....	11
4.4	Migrate the test to custom board .....	11
	<b>References .....</b>	<b>12</b>
	<b>Appendix: Typical test output.....</b>	<b>13</b>

## Figures

<b>Figure 1.</b>	<b>Example of HyperLink mapping different memories.....</b>	<b>7</b>
<b>Figure 2.</b>	<b>Directory structure of test project.....</b>	<b>9</b>

## Tables

<b>Table 1.</b>	<b>Example of memory map at different cores.....</b>	<b>8</b>
<b>Table 2.</b>	<b>Source files of the test codes .....</b>	<b>9</b>

## 1 Introduction

HyperLink provides a highest-speed, low-latency, and low-pin-count communication interface between two Keystone DSPs.

The speed of HyperLink was designed for 12.5Gbps, on most Keystone DSPs, the speed is limited to 10Gbps because of Serdes or Layout limitation. The HyperLink is a TI-specific peripheral, compared to the traditional 8b10b encoding scheme for high speed Serdes interfaces, HyperLink reduces the encoding overhead; the efficient encoding scheme in HyperLink is equivalent to 8b9b.

The HyperLink uses similar memory map scheme like PCIE, but it provides more flexible features designed for multicore DSP. The HyperLink STK (Self Test Kit) is designed to cover following usage cases:

- Access HyperLink with DSP core or EDMA
- Access different memories through HyperLink
- Generate interrupt through HyperLink

## 2 Test Methods

### 2.1 Integrity Test

Two memory test algorithms are used to test the memory access through HyperLink, and the purpose of these test algorithms is discussed below.

#### 2.1.1 Data filling Test

The pseudocode for data pattern filling test is:

```
for(memory range under test)
    fill the memory with a value;
for(memory range under test)
    read back the memory and compare the readback value to the written value
```

Normally, this test is done several times with different test data value. The common values used for test include 0, 0xFFFFFFFF, 0x55555555, 0xAAAAAAAA.

This test can detect stuck data bit, for example, if

written value = 0, readback value = 0x8,

It indicates bit 3 sticks to 1. If

written value = 0xFFFFFFFF, readback value = 0xFFFFFFFFE,

It indicates bit 0 sticks to 0.

If this test fails, the possible reasons include HyperLink physical layer error, the HyperLink buffer error or target memory error. Trying internal loopback test against test between two devices may isolate the reason.

### 2.1.2 Addressing test

The pseudocode for addressing test is:

```
for(memory range under test)
    fill each memory unit with its address value;
for(memory range under test)
    read back the memory and compare the readback value to the written value
```

This test can detect stuck bit on address bus, for example, if

```
written value = 0 at address 0
written value = 1 at address 1
written value = 2 at address 2
written value = 3 at address 3
.....
readback value = 2 at address 0
readback value = 3 at address 1
readback value = 2 at address 2
readback value = 3 at address 3
.....
```

For this case, it indicates bit 1 of address bus sticks. So, the contents in address 0 and 2 are same, and the contents in address 1 and 3 are same.

If the data filling test passes, while this test fails, the possible reasons include HyperLink address translation error, the HyperLink buffer error or target memory error. Trying internal loopback test against test between two devices may isolate the reason.

## 2.2 HyperLink Performance Test

The performance of HyperLink is also measured. If the test shows performance degradation, normally we should check clock speed related configuration.

### 2.2.1 HyperLink Latency Test

The pseudo codes for measuring the latency of DSP core access a memory unit through HyperLink are like following:

```
flushCache();

startCycle= getTimeStampCount();

for(i=0; i< accessTimes; i++)
{
    Access Memory at address;
    address+= stride;
}

cycles = getTimeStampCount()-startCycle;

cycles/Access= cycles/accessTimes;
```

The latency of the interrupt generated by HyperLink is also measured with following pseudo code:

```
.....

startTSC= TimeStampCount;

// manually trigger the hardware event, which will generate interrupt packet to remote side
hyperLinkRegs->SW_INT= HW_EVENT_FOR_INT_TEST;

asm(" IDLE"); //wait for the queue pending interrupt

delay= intTSC - startTSC;

.....

interrupt void HyperLinkISR() //HyperLink Interrupt Service Routine
{
    intTSC= TimeStampCount; //save the Time Stamp Count when the interrupt happens
    .....
}
```

This test is done with internal loopback mode only.

### 2.2.2 HyperLink Throughput Test

The throughput is measured for different memory copy cases with DSP core or EDMA. The throughput is measured by taking total bytes copied and dividing it by the time it used.

## **2.3 HyperLink error reporting and checking**

HyperLink detects error during transfer and reports them through some error/status registers. After the test, all the error/status registers are checked and error/status information is printed/reported.

The HyperLink error can also trigger interrupt to DSP core. Interrupt Service Routine is also implemented to print/report the error information.

## **3 Test cases**

HyperLink Tests should be implemented to cover:

- Internal loopback and test between two devices
- Access HyperLink with DSP core or EDMA
- Access different memories through HyperLink

### **3.1 Test by different masters**

The test cases executed with DSP core including:

- Throughput test
- Latency of Memory access through HyperLink
- HyperLink Interrupt latency test

The test cases executed with EDMA including:

- Throughput test
- overhead of EDMA transfer through HyperLink

### **3.2 Test through different data path**

Internal loopback test is done with the TX data loopback in Serdes module. The internal loopback test can be used to check the function of HyperLink modules inside the KeyStone SOC. If internal loopback test failed, normally, we should check the HyperLink related powers and clocks.

For the test between two DSPs, DSP1's memory are mapped to DSP0 through HyperLink, so, DSP0 accesses DSP 1 through the HyperLink memory window just like access it's own memories.

To simplify the STK usage, same test program are used on two Devices test. The trick is that user must load the program into core **0** of Device0, and load same program into core **1** of Device1. The program will detect the core number at run time, if it is core 0, then it executes the configuration and functions for Device0; if it is core 1, then it executes the configuration and functions for Device1.

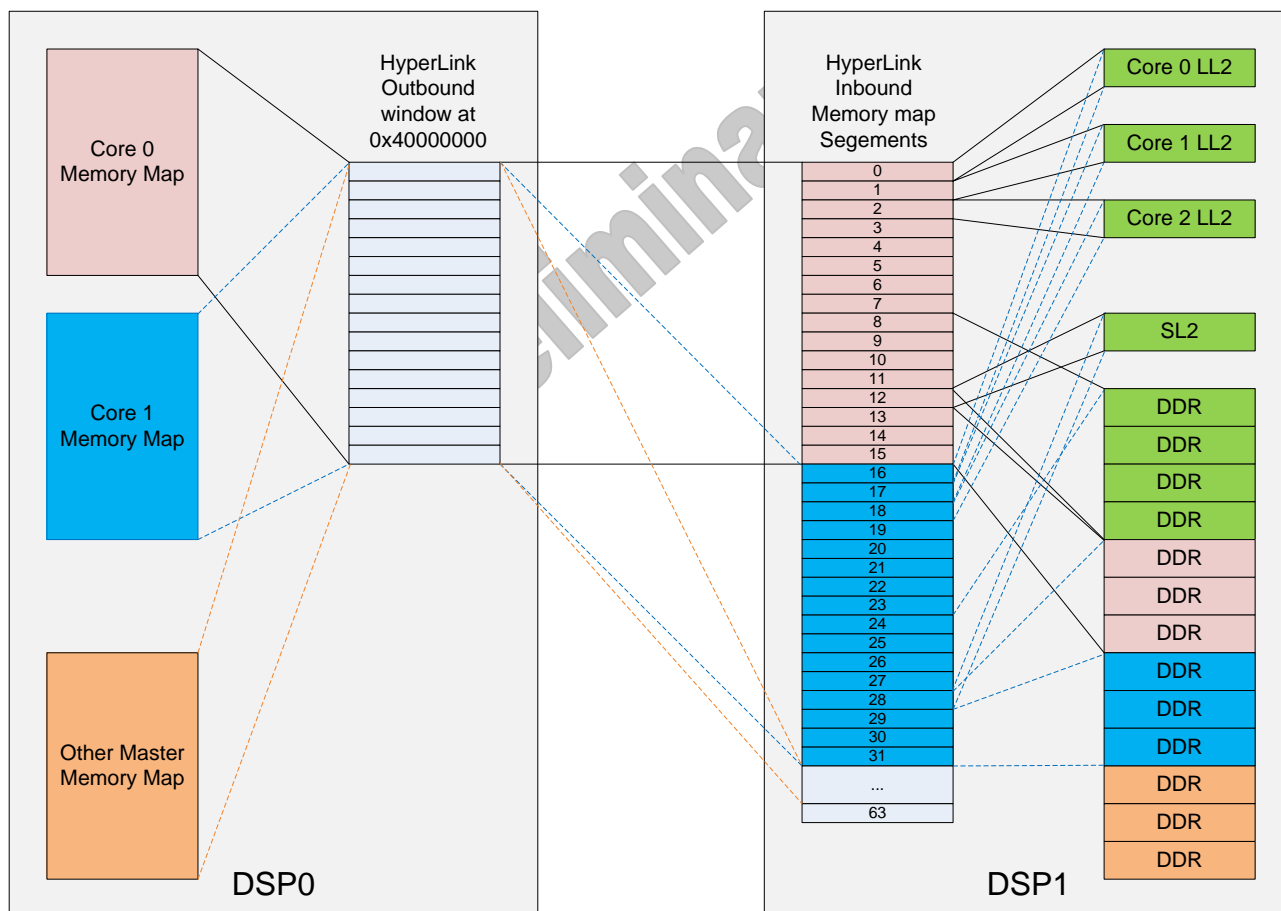
If internal loopback test passes, while the test between two devices fails, normally, we should check the signal integrity on board, and measure the eye-diagram.

Please note, the HyperLink supports automatic power down of Serdes lane, that is, if there is no data transfer over the HyperLink, one or more of the 4 serdes lanes may be powered down, and then we can not capture the eye-diagram on the Serdes lane. Two solutions may be used to overcome this behavior:

1. Continuously transfer data on the HyperLink during the eye-diagram test.
2. Disable the Serdes Lane power down feature for eye-diagram test.

### 3.3 Test different memories in the system

All memories of DSP can be accessed through HyperLink. The HyperLink memory map is very flexible. Following figure shows the configuration for the tests.



**Figure 1. Example of HyperLink mapping different memories**

In this test, DSP1's memory are mapped to DSP0's memory window from 0x40000000 to 0x50000000, the DSP0 can access all memory types in DSP1 including LL2 (Local L2), SL2 (Shared L2) and DDR just like access DSP0's local memory. In DSP0, all masters access same outbound window at 0x40000000, but they may actually access different memory space in DSP1, because the PrivID of the masters in DSP0 is carried with address, and the inbound memory segment configuration of DSP1 is set up separately for different PrivID.

The memory map for DSP core 0 and core1 in this example are summarized in following table.

**Table 1. Example of memory map at different cores**

Local Address (Size)	Address at remote DSP	
	For core 0	For core 1
0x40000000 (16MB)	0x10000000 (LL2)	0x10000000 (LL2)
0x41000000 (16MB)	0x11000000 (LL2)	0x11000000 (LL2)
0x42000000 (16MB)	0x12000000 (LL2)	0x12000000 (LL2)
.....	.....	.....
0x48000000 (16MB)	0x88000000 (DDR)	0x88000000 (DDR)
0x49000000 (16MB)	0x89000000 (DDR)	0x89000000 (DDR)
.....	.....	.....
0x4C000000 (16MB)	0x0C000000 (SL2)	0x0C000000 (SL2)
0x4D000000 (16MB)	0x8C000000 (DDR)	0x8F000000 (DDR)
0x4E000000 (16MB)	0x8D000000 (DDR)	0x90000000 (DDR)
0x4F000000 (16MB)	0x8E000000 (DDR)	0x91000000 (DDR)

With this configuration, when DSP0 accesses address 0x40800000, it actually accesses the LL2 memory of DSP1. When core 0 of DSP0 access address 0x4D000000, it actually accesses the 0x8C000000 in DDR of DSP1; while core 1 of DSP0 accesses address 0x4D000000 will actually access the 0x8F000000 in DDR of DSP1.

For internal loopback test, the memory map configuration is same, the only difference is that, DSP0 actually access its own local memory through HyperLink memory window.

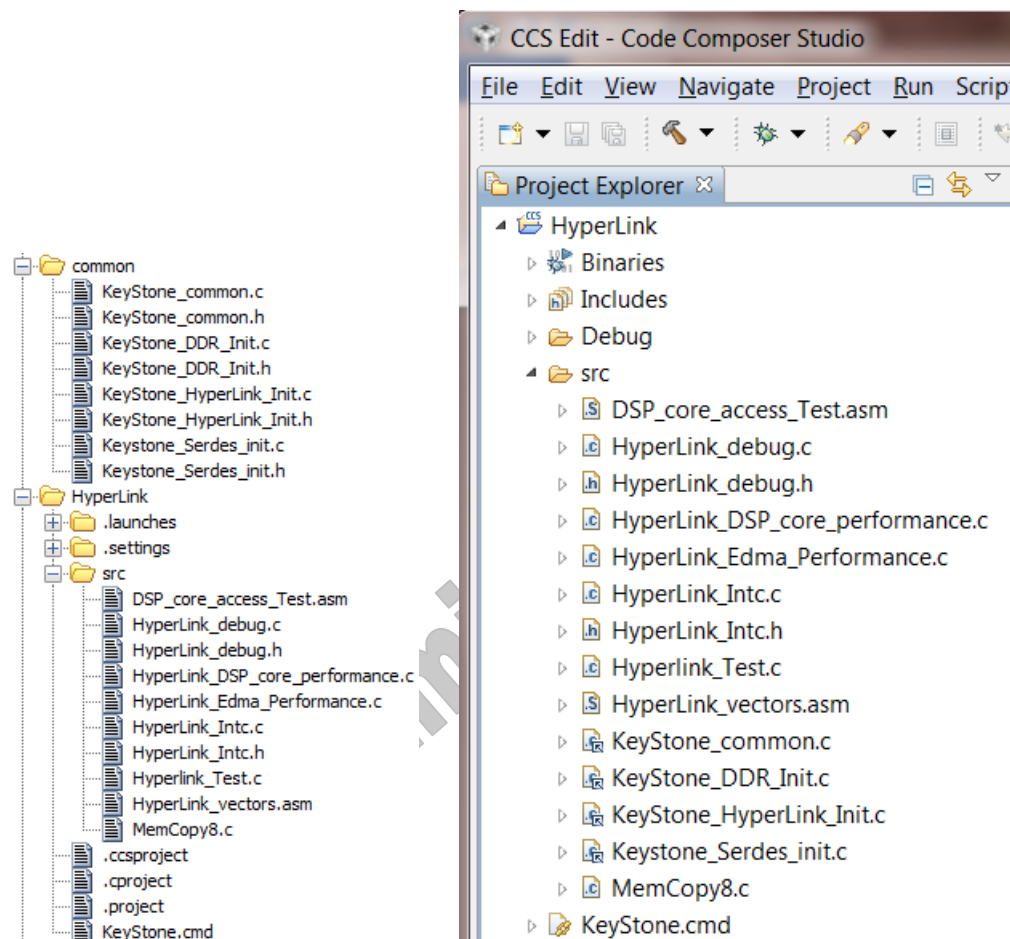
## 4 Test code/project

This section introduces the implementation of these HyperLink tests on KeyStone 1 DSP.



## 4.1 CCS project

Following figure shows the directory structure of the test project.



**Figure 2. Directory structure of test project**

The project files are in “HyperLink” folder. There is some commonly used initialization and driver code for HyperLink, DDR, EDMA, Serdes and PLL... in “common” folder. The main source code files are in the “HyperLink\src” subfolder. Below table describes these source files.

**Table 2. Source files of the test codes**

Source files	Descriptions
KeyStone_common	<p>Initialization for EDMA and PLL, and driver code for simple memory test. The key API include:</p> <pre>void KeyStone_main_PLL_init (float ref_clock_MHz,                              unsigned int multiplier, unsigned int divisor);</pre> <pre>void EDMA_init ();</pre> <pre>int Memory_quick_test(unsigned int uiStartAddress,</pre>

	unsigned int uiTotalByteCount, unsigned int uiFillByteCount, unsigned int uiDataBusWidth)
KeyStone_DDR_Init	Low level DDR initialization. The key API is:  void KeyStone_DDR_init(float ref_clock_MHz, unsigned int DDR_PLLM, unsigned int DDR_PLLD, DDR_ECC_Config * ecc_cfg)
KeyStone_HyperLink_Init	Low level HyperLink initialization. The key API is: void KeyStone_HyperLink_Init(HyperLink_Config * hyperLink_cfg);
KeyStone_Serdes_Init	Low level Serdes initialization.
HyperLink_Test	main function of the tests, HyperLink configuration, integrity test and interrupt latency test.
HyperLink_DSP_core_performance	HyperLink throughput test with DSP core; test latency of memory access through HyperLink with DSP core.
HyperLink_EDMA_performance	HyperLink throughput test with EDMA; test overhead for EDMA transfer through HyperLink
HyperLink_Intc	Interrupts configuration and ISR for HyperLink
HyperLink_debug	print HyperLink status and error information for debug

## 4.2 To run the test program on KeyStone 1 EVM

The test programs can be run on EVM of C6678, C6670 or TCI6614. The type of the device on EVM is detected by STK code automatically.

The steps to run the test cases on EVM board are:

1. extract (or install) the package, and switch CCS workspace to the extracted folder (or installation folder)
2. import the project to CCS
3. build the projects. You may need to change the CSL including path, by default the project use CSL header files in: C:\ti\pdk\_C6678\_1\_1\_2\_6\packages\ti\csf for C6678, C:\ti\pdk\_C6670\_1\_1\_2\_6\packages\ti\csf for C6670, or C:\ti\pdk\_tci6614\_1\_02\_00\_02\packages\ti\csf for TCI6614.
4. Set the boot mode of the device on EVM to no boot.
5. For loopback test, Load the program to core 0 of a DSP. For test between two DSPs, load the program to core 0 of first DSP, and load the program to core 1 of the second DSP.
6. For loopback test, run core 0; for test between two DSPs, run the core 1 of the second DSP firstly and then run the core 0 of the first DSP.
7. See output in console window.

Refer to appendix of this document to see the typical test output.

### 4.3 Test configuration

There are multiple macros defined in the source code to configure the test.

Following macro in the "HyperLink\_Test.c" is used to switch between the internal loopback test and the two devices test. The value of "1" means internal loopback test, and "0" means test between two devices.

```
#define HYPERLINK_LOOPBACK_TEST      1
```

Following macro in the "HyperLink\_Test.c" is used to set the test speed (at the unit of GHz).

```
#define HYPERLINK_SPEED_GHZ      6.25
```

The base address of the DDR be accessed through Hyperlink are defined with following macro in "HyperLink\_Test.c".

```
#define DDR_SPACE_ACCESSED_BY_HYPERLINK  0x88000000
```

The number of the Hyperlink interrupt event be used for test are defined with following macros in "HyperLink\_Test.c".

```
#define HW_EVENT_FOR_INT_TEST      0
```

### 4.4 Migrate the test to custom board

These tests are implemented based on TI's EVM boards.

In real system, the DDR configuration may be changed according to user's hardware design in "KeyStone\_DDR\_Init.c".

DSP core and DDR speed configuration may need be changed in main function like below:

```
//DSP core speed: 122.88*236/29= 999.9889655MHz
```

```
KeyStone_main_PLL_init(122.88, 236, 29);
```

```
//DDR init 66.667*20/1= 1333
```

```
KeyStone_DDR_init (66.667, 20, 1);
```

The HyperLink Serdes reference clock speed may be changed through following macro in main() function:

```
hyperLink_cfg.serdes_cfg.commonSetup.inputRefClock_MHz = 312.5 ;
```

To make your own configurations take effect, you must rebuild the project. Since CSL (Chip Support Library) header files are used by these projects, you may need change CSL including path in your system before you rebuild the project.

## **References**

1. KeyStone Architecture HyperLink User Guide (SPRUGW8)
2. KeyStone Architecture Enhanced Direct Memory Access (EDMA3) Controller User Guide (SPRUGS5)
3. TMS320C66x DSP CorePac User's Guide (SPRUGW0)

Preliminary

## Appendix: Typical test output

Below is the test output on C6678 EVM.

```
Initialize DSP main clock = 100.00MHz/1x10 = 1000MHz
Initialize DDR speed = 66.67MHzx20/1 = 1333.333MTS
HyperLink internal loopback test at 5.00GHz...
Enable Exception handling...
HyperLink memory test passed at address 0x41800000
HyperLink memory test passed at address 0x48000000
HyperLink memory test passed at address 0x4c100000
HyperLink interrupt 1 happens at TSC= 83696497
Hyperlink interrupt latency is 267 cycles
noncacheable, nonprefetchable memory copy
1002 MB/s, copy 65536 bytes from 0x820000 to 0x41800000 consumes 65373 cycles
 42 MB/s, copy 65536 bytes from 0x41800000 to 0x820000 consumes 1548317 cycles
 40 MB/s, copy 65536 bytes from 0x41800000 to 0x41810000 consumes 1621973 cycles
32KB L1D cache, prefetchable memory copy
 753 MB/s, copy 65536 bytes from 0x820000 to 0x41800000 consumes 86937 cycles
 295 MB/s, copy 65536 bytes from 0x41800000 to 0x820000 consumes 221438 cycles
 236 MB/s, copy 65536 bytes from 0x41800000 to 0x41810000 consumes 276526 cycles
32KB L1D cache, 256KB L2 cache, prefetchable memory copy
 504 MB/s, copy 65536 bytes from 0x820000 to 0x41800000 consumes 129895 cycles
 503 MB/s, copy 65536 bytes from 0x41800000 to 0x820000 consumes 130138 cycles
 331 MB/s, copy 65536 bytes from 0x41800000 to 0x41810000 consumes 197456 cycles
noncacheable, nonprefetchable memory copy
1002 MB/s, copy 65536 bytes from 0x820000 to 0x4c100000 consumes 65379 cycles
 47 MB/s, copy 65536 bytes from 0x4c100000 to 0x820000 consumes 1382441 cycles
 44 MB/s, copy 65536 bytes from 0x4c100000 to 0x4c110000 consumes 1474509 cycles
32KB L1D cache, prefetchable memory copy
 753 MB/s, copy 65536 bytes from 0x820000 to 0x4c100000 consumes 86921 cycles
 343 MB/s, copy 65536 bytes from 0x4c100000 to 0x820000 consumes 190598 cycles
 273 MB/s, copy 65536 bytes from 0x4c100000 to 0x4c110000 consumes 239638 cycles
32KB L1D cache, 256KB L2 cache, prefetchable memory copy
 589 MB/s, copy 65536 bytes from 0x820000 to 0x4c100000 consumes 111187 cycles
 587 MB/s, copy 65536 bytes from 0x4c100000 to 0x820000 consumes 111474 cycles
 380 MB/s, copy 65536 bytes from 0x4c100000 to 0x4c110000 consumes 172396 cycles
noncacheable, nonprefetchable memory copy
1002 MB/s, copy 65536 bytes from 0x820000 to 0x48000000 consumes 65379 cycles
 40 MB/s, copy 65536 bytes from 0x48000000 to 0x820000 consumes 1619853 cycles
 35 MB/s, copy 65536 bytes from 0x48000000 to 0x48010000 consumes 1835851 cycles
32KB L1D cache, prefetchable memory copy
 753 MB/s, copy 65536 bytes from 0x820000 to 0x48000000 consumes 86927 cycles
 294 MB/s, copy 65536 bytes from 0x48000000 to 0x820000 consumes 222480 cycles
 221 MB/s, copy 65536 bytes from 0x48000000 to 0x48010000 consumes 295478 cycles
32KB L1D cache, 256KB L2 cache, prefetchable memory copy
 518 MB/s, copy 65536 bytes from 0x820000 to 0x48000000 consumes 126285 cycles
 521 MB/s, copy 65536 bytes from 0x48000000 to 0x820000 consumes 125784 cycles
 364 MB/s, copy 65536 bytes from 0x48000000 to 0x48010000 consumes 179944 cycles

LDDW test: nonprefetchable, noncacheable
Memory access performance test at 0x41800000
Index= 0 Cycles/LDDW= 356.98
Index= 1 Cycles/LDDW= 314.67
Index= 2 Cycles/LDDW= 273.53
Index= 3 Cycles/LDDW= 230.79
Index= 4 Cycles/LDDW= 188.87
Index= 5 Cycles/LDDW= 225.62
Index= 6 Cycles/LDDW= 184.69
Index= 7 Cycles/LDDW= 181.96
Index= 8 Cycles/LDDW= 178.66
Index= 10 Cycles/LDDW= 178.65
Index= 12 Cycles/LDDW= 178.67
Index= 14 Cycles/LDDW= 178.69
Index= 16 Cycles/LDDW= 178.82
Index= 18 Cycles/LDDW= 178.71
Index= 20 Cycles/LDDW= 178.69
Index= 22 Cycles/LDDW= 178.71
Index= 24 Cycles/LDDW= 178.79
Index= 26 Cycles/LDDW= 178.72
Index= 28 Cycles/LDDW= 178.80
Index= 30 Cycles/LDDW= 178.74
Index= 32 Cycles/LDDW= 178.79
```

## KeyStone 1 HyperLink Self Test Kit User's Guide

```
Index= 40 Cycles/LDDW= 178.80
Index= 48 Cycles/LDDW= 178.82
Index= 56 Cycles/LDDW= 178.82
```

STDW test: nonprefetchable, noncacheable

Memory access performance test at 0x41800000

```
Index= 0 Cycles/STDW= 8.04
Index= 1 Cycles/STDW= 8.04
Index= 2 Cycles/STDW= 8.04
Index= 3 Cycles/STDW= 8.04
Index= 4 Cycles/STDW= 8.04
Index= 5 Cycles/STDW= 8.04
Index= 6 Cycles/STDW= 8.04
Index= 7 Cycles/STDW= 8.04
Index= 8 Cycles/STDW= 8.04
Index= 10 Cycles/STDW= 8.04
Index= 12 Cycles/STDW= 8.04
Index= 14 Cycles/STDW= 8.04
Index= 16 Cycles/STDW= 8.04
Index= 18 Cycles/STDW= 8.04
Index= 20 Cycles/STDW= 8.04
Index= 22 Cycles/STDW= 8.04
Index= 24 Cycles/STDW= 8.04
Index= 26 Cycles/STDW= 8.04
Index= 28 Cycles/STDW= 8.04
Index= 30 Cycles/STDW= 8.04
Index= 32 Cycles/STDW= 8.04
Index= 40 Cycles/STDW= 8.04
Index= 48 Cycles/STDW= 8.04
Index= 56 Cycles/STDW= 8.04
```

LDDW test: prefetchable, 32KB L1D cahce

Memory access performance test at 0x41800000

```
Index= 0 Cycles/LDDW= 1.86
Index= 1 Cycles/LDDW= 29.10
Index= 2 Cycles/LDDW= 55.15
Index= 3 Cycles/LDDW= 82.10
Index= 4 Cycles/LDDW= 109.84
Index= 5 Cycles/LDDW= 118.86
Index= 6 Cycles/LDDW= 127.57
Index= 7 Cycles/LDDW= 126.67
Index= 8 Cycles/LDDW= 127.98
Index= 10 Cycles/LDDW= 177.58
Index= 12 Cycles/LDDW= 182.54
Index= 14 Cycles/LDDW= 186.69
Index= 16 Cycles/LDDW= 211.05
Index= 18 Cycles/LDDW= 210.27
Index= 20 Cycles/LDDW= 210.65
Index= 22 Cycles/LDDW= 210.98
Index= 24 Cycles/LDDW= 210.63
Index= 26 Cycles/LDDW= 211.09
Index= 28 Cycles/LDDW= 210.74
Index= 30 Cycles/LDDW= 210.80
Index= 32 Cycles/LDDW= 211.63
Index= 40 Cycles/LDDW= 210.99
Index= 48 Cycles/LDDW= 210.49
Index= 56 Cycles/LDDW= 211.09
```

STDW test: prefetchable, 32KB L1D cahce

Memory access performance test at 0x41800000

```
Index= 0 Cycles/STDW= 1.19
Index= 1 Cycles/STDW= 3.58
Index= 2 Cycles/STDW= 8.88
Index= 3 Cycles/STDW= 14.18
Index= 4 Cycles/STDW= 19.51
Index= 5 Cycles/STDW= 19.51
Index= 6 Cycles/STDW= 19.51
Index= 7 Cycles/STDW= 19.52
Index= 8 Cycles/STDW= 19.53
Index= 10 Cycles/STDW= 19.49
Index= 12 Cycles/STDW= 19.51
Index= 14 Cycles/STDW= 19.50
Index= 16 Cycles/STDW= 19.52
Index= 18 Cycles/STDW= 19.52
Index= 20 Cycles/STDW= 19.49
Index= 22 Cycles/STDW= 19.51
```

```

Index= 24 Cycles/STDW= 19.52
Index= 26 Cycles/STDW= 19.52
Index= 28 Cycles/STDW= 19.52
Index= 30 Cycles/STDW= 19.53
Index= 32 Cycles/STDW= 19.52
Index= 40 Cycles/STDW= 19.50
Index= 48 Cycles/STDW= 19.53
Index= 56 Cycles/STDW= 19.51

```

LDDW test: prefetchable, 32KB L1D, 256KB L2 cahce  
Memory access performance test at 0x41800000

```

Index= 0 Cycles/LDDW= 1.90
Index= 1 Cycles/LDDW= 17.00
Index= 2 Cycles/LDDW= 32.76
Index= 3 Cycles/LDDW= 48.45
Index= 4 Cycles/LDDW= 64.89
Index= 5 Cycles/LDDW= 80.71
Index= 6 Cycles/LDDW= 96.45
Index= 7 Cycles/LDDW= 112.41
Index= 8 Cycles/LDDW= 128.21
Index= 10 Cycles/LDDW= 142.44
Index= 12 Cycles/LDDW= 156.03
Index= 14 Cycles/LDDW= 176.64
Index= 16 Cycles/LDDW= 194.74
Index= 18 Cycles/LDDW= 214.14
Index= 20 Cycles/LDDW= 232.76
Index= 22 Cycles/LDDW= 250.39
Index= 24 Cycles/LDDW= 259.71
Index= 26 Cycles/LDDW= 246.32
Index= 28 Cycles/LDDW= 241.62
Index= 30 Cycles/LDDW= 233.74
Index= 32 Cycles/LDDW= 220.91
Index= 40 Cycles/LDDW= 218.93
Index= 48 Cycles/LDDW= 232.26
Index= 56 Cycles/LDDW= 225.46

```

STDW test: prefetchable, 32KB L1D, 256KB L2 cahce  
Memory access performance test at 0x41800000

```

Index= 0 Cycles/STDW= 1.06
Index= 1 Cycles/STDW= 15.94
Index= 2 Cycles/STDW= 31.66
Index= 3 Cycles/STDW= 47.40
Index= 4 Cycles/STDW= 63.92
Index= 5 Cycles/STDW= 78.88
Index= 6 Cycles/STDW= 94.64
Index= 7 Cycles/STDW= 110.16
Index= 8 Cycles/STDW= 125.91
Index= 10 Cycles/STDW= 140.60
Index= 12 Cycles/STDW= 170.42
Index= 14 Cycles/STDW= 163.35
Index= 16 Cycles/STDW= 182.38
Index= 18 Cycles/STDW= 205.66
Index= 20 Cycles/STDW= 236.73
Index= 22 Cycles/STDW= 246.04
Index= 24 Cycles/STDW= 236.92
Index= 26 Cycles/STDW= 216.59
Index= 28 Cycles/STDW= 214.31
Index= 30 Cycles/STDW= 194.28
Index= 32 Cycles/STDW= 186.34
Index= 40 Cycles/STDW= 187.18
Index= 48 Cycles/STDW= 187.28
Index= 56 Cycles/STDW= 187.25

```

LDDW test: nonprefetchable, noncacheable  
Memory access performance test at 0x4c100000

```

Index= 0 Cycles/LDDW= 318.73
Index= 1 Cycles/LDDW= 282.73
Index= 2 Cycles/LDDW= 246.17
Index= 3 Cycles/LDDW= 205.69
Index= 4 Cycles/LDDW= 169.04
Index= 5 Cycles/LDDW= 200.67
Index= 6 Cycles/LDDW= 164.53
Index= 7 Cycles/LDDW= 163.40
Index= 8 Cycles/LDDW= 159.47
Index= 10 Cycles/LDDW= 159.53
Index= 12 Cycles/LDDW= 159.59

```

## KeyStone 1 HyperLink Self Test Kit User's Guide

```

Index= 14 Cycles/LDDW= 159.54
Index= 16 Cycles/LDDW= 159.48
Index= 18 Cycles/LDDW= 159.52
Index= 20 Cycles/LDDW= 159.62
Index= 22 Cycles/LDDW= 159.60
Index= 24 Cycles/LDDW= 159.71
Index= 26 Cycles/LDDW= 159.52
Index= 28 Cycles/LDDW= 159.72
Index= 30 Cycles/LDDW= 159.73
Index= 32 Cycles/LDDW= 159.48
Index= 40 Cycles/LDDW= 159.71
Index= 48 Cycles/LDDW= 160.02
Index= 56 Cycles/LDDW= 160.02
Index= 64 Cycles/LDDW= 159.46
Index= 96 Cycles/LDDW= 160.59
Index= 128 Cycles/LDDW= 159.47
Index= 192 Cycles/LDDW= 161.72

```

STDW test: nonprefetchable, noncacheable

Memory access performance test at 0x4c100000

```

Index= 0 Cycles/STDW= 8.04
Index= 1 Cycles/STDW= 8.04
Index= 2 Cycles/STDW= 8.04
Index= 3 Cycles/STDW= 8.04
Index= 4 Cycles/STDW= 8.04
Index= 5 Cycles/STDW= 8.04
Index= 6 Cycles/STDW= 8.04
Index= 7 Cycles/STDW= 8.04
Index= 8 Cycles/STDW= 8.05
Index= 10 Cycles/STDW= 8.04
Index= 12 Cycles/STDW= 8.04
Index= 14 Cycles/STDW= 8.05
Index= 16 Cycles/STDW= 8.04
Index= 18 Cycles/STDW= 8.04
Index= 20 Cycles/STDW= 8.04
Index= 22 Cycles/STDW= 8.04
Index= 24 Cycles/STDW= 8.04
Index= 26 Cycles/STDW= 8.05
Index= 28 Cycles/STDW= 8.04
Index= 30 Cycles/STDW= 8.04
Index= 32 Cycles/STDW= 8.04
Index= 40 Cycles/STDW= 8.05
Index= 48 Cycles/STDW= 8.04
Index= 56 Cycles/STDW= 8.04
Index= 64 Cycles/STDW= 8.04
Index= 96 Cycles/STDW= 8.04
Index= 128 Cycles/STDW= 8.04
Index= 192 Cycles/STDW= 8.04

```

LDDW test: prefetchable, 32KB L1D cache

Memory access performance test at 0x4c100000

```

Index= 0 Cycles/LDDW= 1.74
Index= 1 Cycles/LDDW= 23.87
Index= 2 Cycles/LDDW= 45.97
Index= 3 Cycles/LDDW= 68.44
Index= 4 Cycles/LDDW= 91.52
Index= 5 Cycles/LDDW= 98.25
Index= 6 Cycles/LDDW= 102.67
Index= 7 Cycles/LDDW= 103.87
Index= 8 Cycles/LDDW= 103.64
Index= 10 Cycles/LDDW= 141.27
Index= 12 Cycles/LDDW= 150.41
Index= 14 Cycles/LDDW= 167.87
Index= 16 Cycles/LDDW= 177.38
Index= 18 Cycles/LDDW= 177.61
Index= 20 Cycles/LDDW= 178.18
Index= 22 Cycles/LDDW= 179.44
Index= 24 Cycles/LDDW= 177.72
Index= 26 Cycles/LDDW= 178.53
Index= 28 Cycles/LDDW= 180.33
Index= 30 Cycles/LDDW= 178.24
Index= 32 Cycles/LDDW= 178.63
Index= 40 Cycles/LDDW= 179.87
Index= 48 Cycles/LDDW= 179.11
Index= 56 Cycles/LDDW= 178.71
Index= 64 Cycles/LDDW= 182.21

```



```
Index= 96 Cycles/LDDW= 181.66
Index= 128 Cycles/LDDW= 179.22
Index= 192 Cycles/LDDW= 182.02
```

STDW test: prefetchable, 32KB L1D cahce  
Memory access performance test at 0x4c100000

```
Index= 0 Cycles/STDW= 1.19
Index= 1 Cycles/STDW= 3.56
Index= 2 Cycles/STDW= 8.88
Index= 3 Cycles/STDW= 14.18
Index= 4 Cycles/STDW= 19.51
Index= 5 Cycles/STDW= 19.52
Index= 6 Cycles/STDW= 19.53
Index= 7 Cycles/STDW= 19.52
Index= 8 Cycles/STDW= 19.53
Index= 10 Cycles/STDW= 19.53
Index= 12 Cycles/STDW= 19.52
Index= 14 Cycles/STDW= 19.49
Index= 16 Cycles/STDW= 19.50
Index= 18 Cycles/STDW= 19.51
Index= 20 Cycles/STDW= 19.52
Index= 22 Cycles/STDW= 19.52
Index= 24 Cycles/STDW= 19.50
Index= 26 Cycles/STDW= 19.53
Index= 28 Cycles/STDW= 19.52
Index= 30 Cycles/STDW= 19.50
Index= 32 Cycles/STDW= 19.52
Index= 40 Cycles/STDW= 19.51
Index= 48 Cycles/STDW= 19.52
Index= 56 Cycles/STDW= 19.52
Index= 64 Cycles/STDW= 19.52
Index= 96 Cycles/STDW= 19.53
Index= 128 Cycles/STDW= 19.49
Index= 192 Cycles/STDW= 19.52
```

LDDW test: prefetchable, 32KB L1D, 256KB L2 cahce  
Memory access performance test at 0x4c100000

```
Index= 0 Cycles/LDDW= 1.75
Index= 1 Cycles/LDDW= 14.42
Index= 2 Cycles/LDDW= 27.30
Index= 3 Cycles/LDDW= 40.41
Index= 4 Cycles/LDDW= 53.35
Index= 5 Cycles/LDDW= 66.88
Index= 6 Cycles/LDDW= 79.25
Index= 7 Cycles/LDDW= 92.33
Index= 8 Cycles/LDDW= 105.24
Index= 10 Cycles/LDDW= 116.07
Index= 12 Cycles/LDDW= 128.17
Index= 14 Cycles/LDDW= 141.45
Index= 16 Cycles/LDDW= 146.07
Index= 18 Cycles/LDDW= 173.01
Index= 20 Cycles/LDDW= 191.44
Index= 22 Cycles/LDDW= 202.69
Index= 24 Cycles/LDDW= 195.97
Index= 26 Cycles/LDDW= 193.76
Index= 28 Cycles/LDDW= 189.21
Index= 30 Cycles/LDDW= 185.84
Index= 32 Cycles/LDDW= 182.47
Index= 40 Cycles/LDDW= 182.32
Index= 48 Cycles/LDDW= 183.12
Index= 56 Cycles/LDDW= 182.14
Index= 64 Cycles/LDDW= 181.99
Index= 96 Cycles/LDDW= 182.35
Index= 128 Cycles/LDDW= 182.32
Index= 192 Cycles/LDDW= 182.35
```

STDW test: prefetchable, 32KB L1D, 256KB L2 cahce  
Memory access performance test at 0x4c100000

```
Index= 0 Cycles/STDW= 1.06
Index= 1 Cycles/STDW= 13.08
Index= 2 Cycles/STDW= 25.66
Index= 3 Cycles/STDW= 38.94
Index= 4 Cycles/STDW= 51.77
Index= 5 Cycles/STDW= 63.89
Index= 6 Cycles/STDW= 76.64
Index= 7 Cycles/STDW= 89.19
```

## KeyStone 1 HyperLink Self Test Kit User's Guide

```

Index= 8 Cycles/STDW= 101.95
Index= 10 Cycles/STDW= 112.45
Index= 12 Cycles/STDW= 132.49
Index= 14 Cycles/STDW= 126.96
Index= 16 Cycles/STDW= 122.60
Index= 18 Cycles/STDW= 139.25
Index= 20 Cycles/STDW= 155.16
Index= 22 Cycles/STDW= 157.36
Index= 24 Cycles/STDW= 155.08
Index= 26 Cycles/STDW= 146.18
Index= 28 Cycles/STDW= 142.81
Index= 30 Cycles/STDW= 136.48
Index= 32 Cycles/STDW= 135.97
Index= 40 Cycles/STDW= 127.93
Index= 48 Cycles/STDW= 128.22
Index= 56 Cycles/STDW= 127.95
Index= 64 Cycles/STDW= 133.33
Index= 96 Cycles/STDW= 137.01
Index= 128 Cycles/STDW= 151.75
Index= 192 Cycles/STDW= 128.35

```

LDDW test: nonprefetchable, noncacheable

Memory access performance test at 0x48000000

```

Index= 0 Cycles/LDDW= 375.90
Index= 1 Cycles/LDDW= 329.76
Index= 2 Cycles/LDDW= 284.43
Index= 3 Cycles/LDDW= 241.20
Index= 4 Cycles/LDDW= 197.66
Index= 5 Cycles/LDDW= 236.27
Index= 6 Cycles/LDDW= 193.06
Index= 7 Cycles/LDDW= 190.41
Index= 8 Cycles/LDDW= 187.55
Index= 10 Cycles/LDDW= 187.76
Index= 12 Cycles/LDDW= 187.42
Index= 14 Cycles/LDDW= 187.94
Index= 16 Cycles/LDDW= 187.62
Index= 18 Cycles/LDDW= 187.47
Index= 20 Cycles/LDDW= 187.50
Index= 22 Cycles/LDDW= 187.56
Index= 24 Cycles/LDDW= 187.36
Index= 26 Cycles/LDDW= 187.17
Index= 28 Cycles/LDDW= 187.63
Index= 30 Cycles/LDDW= 187.94
Index= 32 Cycles/LDDW= 187.83
Index= 40 Cycles/LDDW= 187.56
Index= 48 Cycles/LDDW= 188.21
Index= 56 Cycles/LDDW= 188.10
Index= 64 Cycles/LDDW= 189.04
Index= 96 Cycles/LDDW= 188.83
Index= 128 Cycles/LDDW= 191.10
Index= 192 Cycles/LDDW= 191.52
Index= 256 Cycles/LDDW= 195.68
Index= 384 Cycles/LDDW= 196.74
Index= 512 Cycles/LDDW= 200.67
Index= 768 Cycles/LDDW= 201.05
Index=1024 Cycles/LDDW= 201.14
Index=1536 Cycles/LDDW= 201.21
Index=2048 Cycles/LDDW= 401.94
Index=3072 Cycles/LDDW= 201.23

```

STDW test: nonprefetchable, noncacheable

Memory access performance test at 0x48000000

```

Index= 0 Cycles/STDW= 8.04
Index= 1 Cycles/STDW= 8.04
Index= 2 Cycles/STDW= 8.04
Index= 3 Cycles/STDW= 8.04
Index= 4 Cycles/STDW= 8.04
Index= 5 Cycles/STDW= 8.04
Index= 6 Cycles/STDW= 8.04
Index= 7 Cycles/STDW= 8.04
Index= 8 Cycles/STDW= 8.04
Index= 10 Cycles/STDW= 8.04
Index= 12 Cycles/STDW= 8.04
Index= 14 Cycles/STDW= 8.04
Index= 16 Cycles/STDW= 8.04
Index= 18 Cycles/STDW= 8.04

```

```

Index= 20 Cycles/STDW= 8.04
Index= 22 Cycles/STDW= 8.04
Index= 24 Cycles/STDW= 8.04
Index= 26 Cycles/STDW= 8.04
Index= 28 Cycles/STDW= 8.04
Index= 30 Cycles/STDW= 8.04
Index= 32 Cycles/STDW= 8.04
Index= 40 Cycles/STDW= 8.04
Index= 48 Cycles/STDW= 8.04
Index= 56 Cycles/STDW= 8.04
Index= 64 Cycles/STDW= 8.04
Index= 96 Cycles/STDW= 8.04
Index= 128 Cycles/STDW= 8.04
Index= 192 Cycles/STDW= 8.04
Index= 256 Cycles/STDW= 8.04
Index= 384 Cycles/STDW= 8.04
Index= 512 Cycles/STDW= 8.04
Index= 768 Cycles/STDW= 8.04
Index=1024 Cycles/STDW= 8.04
Index=1536 Cycles/STDW= 8.04
Index=2048 Cycles/STDW= 8.44
Index=3072 Cycles/STDW= 8.04

```

LDDW test: prefetchable, 32KB L1D cahce  
Memory access performance test at 0x48000000

```

Index= 0 Cycles/LDDW= 1.87
Index= 1 Cycles/LDDW= 27.90
Index= 2 Cycles/LDDW= 52.84
Index= 3 Cycles/LDDW= 79.02
Index= 4 Cycles/LDDW= 105.76
Index= 5 Cycles/LDDW= 111.78
Index= 6 Cycles/LDDW= 116.22
Index= 7 Cycles/LDDW= 116.88
Index= 8 Cycles/LDDW= 117.19
Index= 10 Cycles/LDDW= 162.80
Index= 12 Cycles/LDDW= 168.80
Index= 14 Cycles/LDDW= 173.86
Index= 16 Cycles/LDDW= 206.46
Index= 18 Cycles/LDDW= 206.68
Index= 20 Cycles/LDDW= 206.74
Index= 22 Cycles/LDDW= 206.86
Index= 24 Cycles/LDDW= 205.97
Index= 26 Cycles/LDDW= 206.75
Index= 28 Cycles/LDDW= 207.68
Index= 30 Cycles/LDDW= 207.00
Index= 32 Cycles/LDDW= 207.42
Index= 40 Cycles/LDDW= 207.64
Index= 48 Cycles/LDDW= 207.08
Index= 56 Cycles/LDDW= 207.48
Index= 64 Cycles/LDDW= 208.69
Index= 96 Cycles/LDDW= 207.63
Index= 128 Cycles/LDDW= 210.82
Index= 192 Cycles/LDDW= 209.98
Index= 256 Cycles/LDDW= 219.00
Index= 384 Cycles/LDDW= 215.15
Index= 512 Cycles/LDDW= 220.15
Index= 768 Cycles/LDDW= 219.56
Index=1024 Cycles/LDDW= 219.67
Index=1536 Cycles/LDDW= 219.64
Index=2048 Cycles/LDDW= 439.02
Index=3072 Cycles/LDDW= 219.75

```

STDW test: prefetchable, 32KB L1D cahce  
Memory access performance test at 0x48000000

```

Index= 0 Cycles/STDW= 1.19
Index= 1 Cycles/STDW= 3.57
Index= 2 Cycles/STDW= 8.88
Index= 3 Cycles/STDW= 14.20
Index= 4 Cycles/STDW= 19.49
Index= 5 Cycles/STDW= 19.52
Index= 6 Cycles/STDW= 19.52
Index= 7 Cycles/STDW= 19.50
Index= 8 Cycles/STDW= 19.52
Index= 10 Cycles/STDW= 19.51
Index= 12 Cycles/STDW= 19.51
Index= 14 Cycles/STDW= 19.52

```

## KeyStone 1 HyperLink Self Test Kit User's Guide

```

Index= 16 Cycles/STDW= 19.53
Index= 18 Cycles/STDW= 19.51
Index= 20 Cycles/STDW= 19.53
Index= 22 Cycles/STDW= 19.52
Index= 24 Cycles/STDW= 19.50
Index= 26 Cycles/STDW= 19.52
Index= 28 Cycles/STDW= 19.52
Index= 30 Cycles/STDW= 19.52
Index= 32 Cycles/STDW= 19.53
Index= 40 Cycles/STDW= 19.52
Index= 48 Cycles/STDW= 19.52
Index= 56 Cycles/STDW= 19.51
Index= 64 Cycles/STDW= 19.51
Index= 96 Cycles/STDW= 19.52
Index= 128 Cycles/STDW= 19.52
Index= 192 Cycles/STDW= 19.52
Index= 256 Cycles/STDW= 19.51
Index= 384 Cycles/STDW= 19.52
Index= 512 Cycles/STDW= 19.53
Index= 768 Cycles/STDW= 19.50
Index=1024 Cycles/STDW= 19.50
Index=1536 Cycles/STDW= 19.52
Index=2048 Cycles/STDW= 19.51
Index=3072 Cycles/STDW= 19.52

```

LDDW test: prefetchable, 32KB L1D, 256KB L2 cahce  
Memory access performance test at 0x48000000

```

Index= 0 Cycles/LDDW= 1.91
Index= 1 Cycles/LDDW= 15.79
Index= 2 Cycles/LDDW= 30.51
Index= 3 Cycles/LDDW= 45.15
Index= 4 Cycles/LDDW= 60.34
Index= 5 Cycles/LDDW= 75.35
Index= 6 Cycles/LDDW= 89.99
Index= 7 Cycles/LDDW= 104.38
Index= 8 Cycles/LDDW= 117.60
Index= 10 Cycles/LDDW= 129.33
Index= 12 Cycles/LDDW= 138.15
Index= 14 Cycles/LDDW= 154.90
Index= 16 Cycles/LDDW= 159.74
Index= 18 Cycles/LDDW= 181.60
Index= 20 Cycles/LDDW= 194.66
Index= 22 Cycles/LDDW= 214.62
Index= 24 Cycles/LDDW= 225.14
Index= 26 Cycles/LDDW= 222.77
Index= 28 Cycles/LDDW= 219.03
Index= 30 Cycles/LDDW= 214.79
Index= 32 Cycles/LDDW= 209.10
Index= 40 Cycles/LDDW= 208.80
Index= 48 Cycles/LDDW= 210.71
Index= 56 Cycles/LDDW= 209.92
Index= 64 Cycles/LDDW= 210.11
Index= 96 Cycles/LDDW= 211.11
Index= 128 Cycles/LDDW= 214.09
Index= 192 Cycles/LDDW= 214.73
Index= 256 Cycles/LDDW= 219.17
Index= 384 Cycles/LDDW= 220.60
Index= 512 Cycles/LDDW= 227.59
Index= 768 Cycles/LDDW= 228.18
Index=1024 Cycles/LDDW= 228.71
Index=1536 Cycles/LDDW= 228.61
Index=2048 Cycles/LDDW= 456.05
Index=3072 Cycles/LDDW= 228.68

```

STDW test: prefetchable, 32KB L1D, 256KB L2 cahce  
Memory access performance test at 0x48000000

```

Index= 0 Cycles/STDW= 1.06
Index= 1 Cycles/STDW= 14.50
Index= 2 Cycles/STDW= 28.84
Index= 3 Cycles/STDW= 43.61
Index= 4 Cycles/STDW= 58.55
Index= 5 Cycles/STDW= 72.37
Index= 6 Cycles/STDW= 87.27
Index= 7 Cycles/STDW= 101.34
Index= 8 Cycles/STDW= 115.86
Index= 10 Cycles/STDW= 127.28

```

**Error! No text of specified style in document.** 21

transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x11840000,	consumes	583	cycles,	achieve	bandwidth	13	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c0800000	to 0x42840000,	consumes	253	cycles,	achieve	bandwidth	31	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c0800000	to 0x4c140000,	consumes	253	cycles,	achieve	bandwidth	31	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c0800000	to 0x48300000,	consumes	253	cycles,	achieve	bandwidth	31	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x c0c0000,	consumes	583	cycles,	achieve	bandwidth	13	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c100000	to 0x c0c0000,	consumes	517	cycles,	achieve	bandwidth	15	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x c0c0000,	consumes	583	cycles,	achieve	bandwidth	13	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x42840000,	consumes	319	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x4c140000,	consumes	319	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x48300000,	consumes	319	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x88100000,	consumes	583	cycles,	achieve	bandwidth	13	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c100000	to 0x88100000,	consumes	517	cycles,	achieve	bandwidth	15	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x88100000,	consumes	649	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x42840000,	consumes	583	cycles,	achieve	bandwidth	13	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c100000	to 0x4c140000,	consumes	517	cycles,	achieve	bandwidth	15	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x48300000,	consumes	583	cycles,	achieve	bandwidth	13	MB/s
Throughput test with EDMA0 TC1															
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x42840000,	consumes	36157	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x4c140000,	consumes	36157	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x48300000,	consumes	36157	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x11840000,	consumes	87967	cycles,	achieve	bandwidth	745	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c100000	to 0x11840000,	consumes	58003	cycles,	achieve	bandwidth	1129	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x11840000,	consumes	64603	cycles,	achieve	bandwidth	1014	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c0800000	to 0x42840000,	consumes	36091	cycles,	achieve	bandwidth	1815	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c0800000	to 0x4c140000,	consumes	36091	cycles,	achieve	bandwidth	1815	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c0800000	to 0x48300000,	consumes	36091	cycles,	achieve	bandwidth	1815	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x c0c0000,	consumes	87901	cycles,	achieve	bandwidth	745	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c100000	to 0x c0c0000,	consumes	58003	cycles,	achieve	bandwidth	1129	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x c0c0000,	consumes	64471	cycles,	achieve	bandwidth	1016	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x88000000	to 0x42840000,	consumes	36157	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x88000000	to 0x4c140000,	consumes	36223	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x88000000	to 0x48300000,	consumes	36157	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x88100000,	consumes	87967	cycles,	achieve	bandwidth	745	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c100000	to 0x88100000,	consumes	57937	cycles,	achieve	bandwidth	1131	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x88100000,	consumes	67177	cycles,	achieve	bandwidth	975	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x42840000,	consumes	99715	cycles,	achieve	bandwidth	657	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c100000	to 0x4c140000,	consumes	84733	cycles,	achieve	bandwidth	773	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x48300000,	consumes	101893	cycles,	achieve	bandwidth	643	MB/s	
Overhead test with EDMA1 TC0															
transfer	1	*	8	Bytes	with index=	8	from 0x11820000	to 0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x11820000	to 0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x11820000	to 0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c100000	to 0x11840000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c0800000	to 0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c0800000	to 0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c0800000	to 0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c100000	to 0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x c0c0000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x42840000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x4c140000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x48300000,	consumes	412	cycles,	achieve	bandwidth	19	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x88100000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c100000	to 0x88100000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x88100000,	consumes	718	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x42840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c100000	to 0x4c140000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x48300000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
Throughput test with EDMA1 TC0															
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x42840000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x4c140000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x48300000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x11840000,	consumes	92722	cycles,	achieve	bandwidth	706	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c100000	to 0x11840000,	consumes	66814	cycles,	achieve	bandwidth	980	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x11840000,	consumes	69619	cycles,	achieve	bandwidth	941	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c0800000	to 0x42840000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c0800000	to 0x4c140000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c0800000	to 0x48300000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x c0c0000,	consumes	92773	cycles,	achieve	bandwidth	706	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c100000	to 0x c0c0000,	consumes	66865	cycles,	achieve	bandwidth	980	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x c0c0000,	consumes	69568	cycles,	achieve	bandwidth	942	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x88000000	to 0x42840000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x88000000	to 0x4c140000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x88000000	to 0x48300000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	

transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x88100000,	consumes	92773	cycles,	achieve	bandwidth	706	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x88100000,	consumes	66865	cycles,	achieve	bandwidth	980	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x88100000,	consumes	79870	cycles,	achieve	bandwidth	820	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x42840000,	consumes	99964	cycles,	achieve	bandwidth	655	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x4c140000,	consumes	84919	cycles,	achieve	bandwidth	771	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x48300000,	consumes	102259	cycles,	achieve	bandwidth	640	MB/s	
Overhead test with EDMA1 TC1																		
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x11840000,	consumes	565	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x c0c0000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x c0c0000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x42840000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x4c140000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x48300000,	consumes	412	cycles,	achieve	bandwidth	19	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x88100000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x88100000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x88100000,	consumes	718	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x42840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x4c140000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x48300000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
Throughput test with EDMA1 TC1																		
transfer	4	*	16384	Bytes	with	index=16384	from	0x11820000	to	0x42840000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x11820000	to	0x4c140000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x11820000	to	0x48300000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x11840000,	consumes	97108	cycles,	achieve	bandwidth	674	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x11840000,	consumes	68956	cycles,	achieve	bandwidth	950	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x11840000,	consumes	74260	cycles,	achieve	bandwidth	882	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x c080000	to	0x42840000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x c080000	to	0x4c140000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x c080000	to	0x48300000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x c0c0000,	consumes	97108	cycles,	achieve	bandwidth	674	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x c0c0000,	consumes	69007	cycles,	achieve	bandwidth	949	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x c0c0000,	consumes	74209	cycles,	achieve	bandwidth	883	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x88000000	to	0x42840000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x88000000	to	0x4c140000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x88000000	to	0x48300000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x88100000,	consumes	97159	cycles,	achieve	bandwidth	674	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x88100000,	consumes	69007	cycles,	achieve	bandwidth	949	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x88100000,	consumes	78901	cycles,	achieve	bandwidth	830	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x42840000,	consumes	103789	cycles,	achieve	bandwidth	631	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x4c140000,	consumes	88744	cycles,	achieve	bandwidth	738	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x48300000,	consumes	132043	cycles,	achieve	bandwidth	496	MB/s	
Overhead test with EDMA1 TC2																		
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x11840000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x c0c0000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x c0c0000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x42840000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x4c140000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x48300000,	consumes	412	cycles,	achieve	bandwidth	19	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x88100000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x88100000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x88100000,	consumes	718	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x42840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x4c140000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x48300000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
Throughput test with EDMA1 TC2																		
transfer	4	*	16384	Bytes	with	index=16384	from	0x11820000	to	0x42840000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x11820000	to	0x4c140000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x11820000	to	0x48300000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	

transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x11840000,	consumes	92722	cycles,	achieve	bandwidth	706	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x11840000,	consumes	66865	cycles,	achieve	bandwidth	980	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x11840000,	consumes	69619	cycles,	achieve	bandwidth	941	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x c080000	to	0x42840000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x c080000	to	0x4c140000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x c080000	to	0x48300000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x c0c0000,	consumes	92773	cycles,	achieve	bandwidth	706	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x c0c0000,	consumes	66865	cycles,	achieve	bandwidth	980	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x c0c0000,	consumes	69466	cycles,	achieve	bandwidth	943	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x88000000	to	0x42840000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x88000000	to	0x4c140000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x88000000	to	0x48300000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x88100000,	consumes	92773	cycles,	achieve	bandwidth	706	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x88100000,	consumes	66916	cycles,	achieve	bandwidth	979	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x88100000,	consumes	79768	cycles,	achieve	bandwidth	821	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x42840000,	consumes	99964	cycles,	achieve	bandwidth	655	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x4c140000,	consumes	84919	cycles,	achieve	bandwidth	771	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x48300000,	consumes	102259	cycles,	achieve	bandwidth	640	MB/s	
Overhead test with EDMA1 TC3																		
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x11840000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x c0c0000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x42840000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x4c140000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x48300000,	consumes	412	cycles,	achieve	bandwidth	19	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x88100000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x88100000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x88100000,	consumes	718	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x42840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x4c140000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x48300000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
Throughput test with EDMA1 TC3																		
transfer	4	*	16384	Bytes	with	index=16384	from	0x11820000	to	0x42840000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x11820000	to	0x4c140000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x11820000	to	0x48300000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x11840000,	consumes	97159	cycles,	achieve	bandwidth	674	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x11840000,	consumes	68956	cycles,	achieve	bandwidth	950	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x11840000,	consumes	74617	cycles,	achieve	bandwidth	878	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x c080000	to	0x42840000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x c080000	to	0x4c140000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x c080000	to	0x48300000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x c0c0000,	consumes	97159	cycles,	achieve	bandwidth	674	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x c0c0000,	consumes	69007	cycles,	achieve	bandwidth	949	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x c0c0000,	consumes	74617	cycles,	achieve	bandwidth	878	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x88000000	to	0x42840000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x88000000	to	0x4c140000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x88000000	to	0x48300000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x88100000,	consumes	97159	cycles,	achieve	bandwidth	674	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x88100000,	consumes	69007	cycles,	achieve	bandwidth	949	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x88100000,	consumes	79003	cycles,	achieve	bandwidth	829	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x42820000	to	0x42840000,	consumes	103738	cycles,	achieve	bandwidth	631	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x4c100000	to	0x4c140000,	consumes	88744	cycles,	achieve	bandwidth	738	MB/s	
transfer	4	*	16384	Bytes	with	index=16384	from	0x48200000	to	0x48300000,	consumes	131890	cycles,	achieve	bandwidth	496	MB/s	
Overhead test with EDMA2 TC0																		
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x11820000	to	0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x11840000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x c080000	to	0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x42820000	to	0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x4c100000	to	0x c0c0000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x48200000	to	0x c0c0000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with	index=	8	from	0x88000000	to	0x42840000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s



**Error! No text of specified style in document.** 25

transfer	1	*	8	Bytes	with index=	8	from 0x11820000	to 0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x11820000	to 0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c140000	to 0x11840000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c080000	to 0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c080000	to 0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c080000	to 0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c140000	to 0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x c0c0000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x42840000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x4c140000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x48300000,	consumes	412	cycles,	achieve	bandwidth	19	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x88100000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c140000	to 0x88100000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x88100000,	consumes	718	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x42840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c140000	to 0x4c140000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x48300000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
Throughput test with EDMA2 TC2															
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x42840000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x4c140000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x48300000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x11840000,	consumes	97108	cycles,	achieve	bandwidth	674	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c140000	to 0x11840000,	consumes	68956	cycles,	achieve	bandwidth	950	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x11840000,	consumes	74617	cycles,	achieve	bandwidth	878	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c080000	to 0x42840000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c080000	to 0x4c140000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c080000	to 0x48300000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x c0c0000,	consumes	97159	cycles,	achieve	bandwidth	674	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c140000	to 0x c0c0000,	consumes	69007	cycles,	achieve	bandwidth	949	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x c0c0000,	consumes	74362	cycles,	achieve	bandwidth	881	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x88000000	to 0x42840000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x88000000	to 0x4c140000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x88000000	to 0x48300000,	consumes	36265	cycles,	achieve	bandwidth	1807	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x88100000,	consumes	97108	cycles,	achieve	bandwidth	674	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c140000	to 0x88100000,	consumes	69007	cycles,	achieve	bandwidth	949	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x88100000,	consumes	79156	cycles,	achieve	bandwidth	827	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x42840000,	consumes	103738	cycles,	achieve	bandwidth	631	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c140000	to 0x4c140000,	consumes	88744	cycles,	achieve	bandwidth	738	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x48300000,	consumes	131992	cycles,	achieve	bandwidth	496	MB/s	
Overhead test with EDMA2 TC3															
transfer	1	*	8	Bytes	with index=	8	from 0x11820000	to 0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x11820000	to 0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x11820000	to 0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c140000	to 0x11840000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x11840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c080000	to 0x42840000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c080000	to 0x4c140000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x c080000	to 0x48300000,	consumes	310	cycles,	achieve	bandwidth	25	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c140000	to 0x c0c0000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x c0c0000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x42840000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x4c140000,	consumes	361	cycles,	achieve	bandwidth	22	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x88000000	to 0x48300000,	consumes	412	cycles,	achieve	bandwidth	19	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x88100000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c140000	to 0x88100000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x88100000,	consumes	718	cycles,	achieve	bandwidth	11	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x42820000	to 0x42840000,	consumes	616	cycles,	achieve	bandwidth	12	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x4c140000	to 0x4c140000,	consumes	565	cycles,	achieve	bandwidth	14	MB/s
transfer	1	*	8	Bytes	with index=	8	from 0x48200000	to 0x48300000,	consumes	667	cycles,	achieve	bandwidth	11	MB/s
Throughput test with EDMA2 TC3															
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x42840000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x4c140000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x11820000	to 0x48300000,	consumes	36163	cycles,	achieve	bandwidth	1812	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x11840000,	consumes	92722	cycles,	achieve	bandwidth	706	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c140000	to 0x11840000,	consumes	68865	cycles,	achieve	bandwidth	980	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x48200000	to 0x11840000,	consumes	70639	cycles,	achieve	bandwidth	927	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c080000	to 0x42840000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c080000	to 0x4c140000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x c080000	to 0x48300000,	consumes	36214	cycles,	achieve	bandwidth	1809	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x42820000	to 0x c0c0000,	consumes	92773	cycles,	achieve	bandwidth	706	MB/s	
transfer	4	*	16384	Bytes	with index=16384	from 0x4c140000	to 0x c0c0000,	consumes	68865	cycles,	achieve	bandwidth	980	MB/s	

```

transfer 4 * 16384 Bytes with index=16384 from 0x48200000 to 0x c0c0000, consumes 69823 cycles, achieve bandwidth 938 MB/s
transfer 4 * 16384 Bytes with index=16384 from 0x88000000 to 0x42840000, consumes 36265 cycles, achieve bandwidth 1807 MB/s
transfer 4 * 16384 Bytes with index=16384 from 0x88000000 to 0x4c140000, consumes 36265 cycles, achieve bandwidth 1807 MB/s
transfer 4 * 16384 Bytes with index=16384 from 0x88000000 to 0x48300000, consumes 36265 cycles, achieve bandwidth 1807 MB/s
transfer 4 * 16384 Bytes with index=16384 from 0x42820000 to 0x88100000, consumes 92773 cycles, achieve bandwidth 706 MB/s
transfer 4 * 16384 Bytes with index=16384 from 0x4c100000 to 0x88100000, consumes 66916 cycles, achieve bandwidth 979 MB/s
transfer 4 * 16384 Bytes with index=16384 from 0x48200000 to 0x88100000, consumes 79564 cycles, achieve bandwidth 823 MB/s
transfer 4 * 16384 Bytes with index=16384 from 0x42820000 to 0x42840000, consumes 99964 cycles, achieve bandwidth 655 MB/s
transfer 4 * 16384 Bytes with index=16384 from 0x4c100000 to 0x4c140000, consumes 84919 cycles, achieve bandwidth 771 MB/s
transfer 4 * 16384 Bytes with index=16384 from 0x48200000 to 0x48300000, consumes 102157 cycles, achieve bandwidth 641 MB/s
EDMA test complete
-----HyperLink status when test completes-----
a request has been detected on the Tx VBUSM slave interface.
the Tx PLS layer has linked to the remote device.
lane zero has been identified during training
TX Serdes lane 0 is enabled.
RX Serdes lane 0 is enabled.
TX Serdes lane 1 is enabled.
RX Serdes lane 1 is enabled.
TX Serdes lane 2 is enabled.
RX Serdes lane 2 is enabled.
TX Serdes lane 3 is enabled.
RX Serdes lane 3 is enabled.

```

Preliminary