ZYNQMP DRAM DIAGNOSTICS TEST



Updated 14 November 2019

# Introduction

The Zynq MP DRAM diagnostics test is a stand-alone program running on a single Zynq MPSoC Cortex-A53 processor, executing out of OCM. The program uses the UART for interactive operation. A small menu is displayed, and the user may choose to run various memory tests.

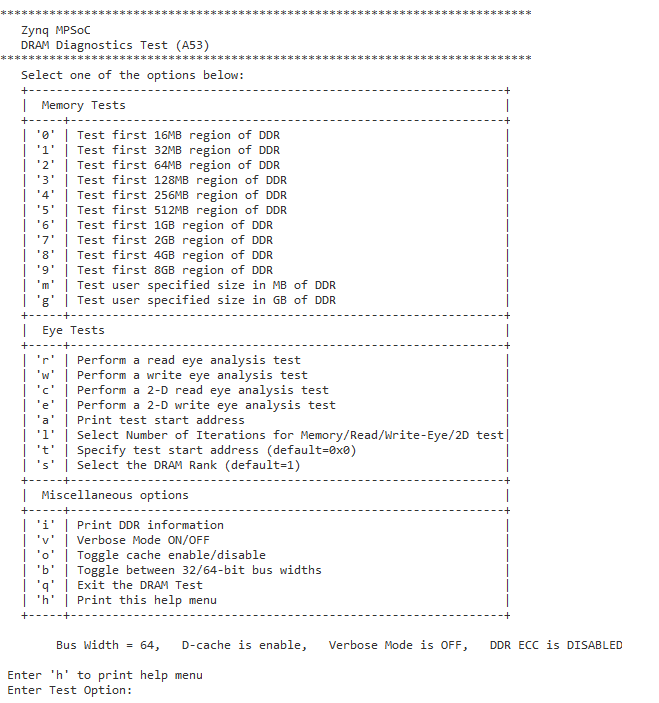
While running the test from SDK, these operations are performed by SDK, during the debug launch.

Do not do DDR remap or RAM remap.

# Running the Test

After connecting a board and launching the test from SDK, in the terminal window you should see the test menu as shown in the screen shot below.

To select a function, hit a single key without <enter>. Information is printed as the test progresses, and the menu is re-printed when the function completes.

****

## Memory Tests

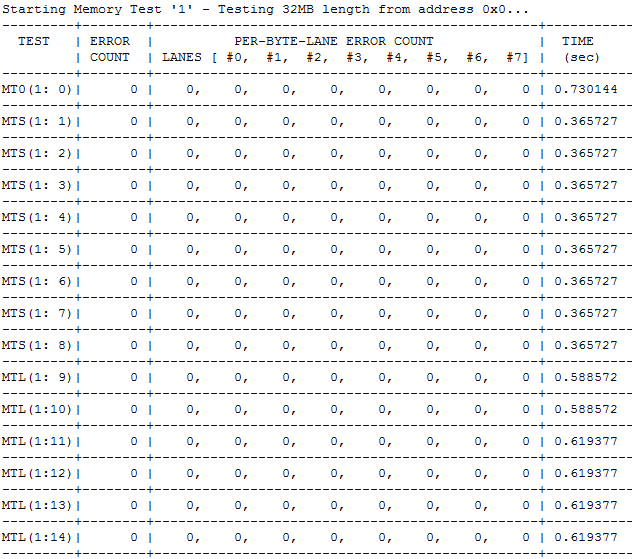
The keys 0,1,2,3,4,5,6,7,8,9,m,g select a memory test of varying lengths as shown below.

|  |  |
| --- | --- |
| Key | Test Length |
| ‘0’ | 2MB |
| ‘1’ | 32MB |
| ‘2’ | 64MB |
| ‘3’ | 128MB |
| ‘4’ | 256MB |
| ‘5’ | 512MB |
| ‘6’ | 1GB |
| ‘7’ | 2GB |
| ‘8’ | 4GB |
| ‘9’ | 8GB |
| ‘m’ | User specified integer MegaBytes |
| ‘g’ | User specified integer GigaBytes |

Each memory test consists of 15 sub-tests using different data patterns. In each sub-test the entire range is first written sequentially and then read and compared against the expected value. The 15 patterns are:

|  |  |
| --- | --- |
| Sub-test | Description |
| 0 | Incrementing pattern, unique value per memory location (data = address) |
| 1 | All 0 |
| 2 | All 0xffffffff |
| 3 | All 0xAAAAAAAA |
| 4 | All 0x55555555 |
| 5 | Alternating 0x00000000 and 0xFFFFFFFF |
| 6 | Alternating 0xFFFFFFFF and 0x00000000 |
| 7 | Alternating 0x55555555 and 0xAAAAAAAA |
| 8 | Alternating 0xAAAAAAAA and 0x55555555 |
| 9 | Aggressor pattern identical on all 8 bits |
| 10 | Aggressor pattern with one bit inverted, x8 times (1 per bit) |
| 11-14 | Pseudo random patterns with different seeds |

Below is a screen shot after hitting the ‘1’ key.



When running the memory test it’s best to keep the Data Cache enabled (default is enabled and ‘o’ is the toggle option) because it will give the highest performance and will result in the highest activity level of the memory interface. Having a high duty cycle on the memory interface is the ideal way to stress the system when trying to reproduce data errors. When running the memory test set the Verbose option to enabled (default is disabled and ‘v’ is the toggle option) since it’s necessary to get the data compare error details when trying to isolate failure modes.

A word error count and per-byte-lane error counts are provided in the format (Lane-0 Lane-1 Lane-2 Lane-3, Lane-4, Lane-5, Lane-6, Lane-7).

**Word Error Count:** No. of words having errors on read back and comparison with written value.  
**Per-Byte-Lane Error Counts:** No. of bytes having errors on read back and comparison with written value.

*Example:*  
Data Written : 0x01010101, Data Read Back: 0x01000001

The per-byte-lane error count reads errors on Lane-1 & Lane-2

The word error count is 1

Errors, if any, are reported for each sub-test. In verbose mode (hit the ‘v’ key), the first 10 errors in each sub-test are printed, for example:

Memtest\_l ERROR: addr=0x107154 rd/ref/xor = 0x478FDCF5 0x478FDDF5 0x00000100

Memtest\_l ERROR: addr=0x10E78C rd/ref/xor = 0x34BBA068 0x34BBA078 0x00000010

Memtest\_l ERROR: addr=0x10F00C rd/ref/xor = 0xFEF17729 0xFEF1F729 0x00008000

Memtest\_l ERROR: addr=0x10F024 rd/ref/xor = 0xBC7DCA6B 0xBC7DCA7B 0x00000010

Memtest\_l ERROR: addr=0x11304C rd/ref/xor = 0x87D46558 0x87D46758 0x00000200

Memtest\_l ERROR: addr=0x115154 rd/ref/xor = 0xD894F5B9 0xD894F5BD 0x00000004

Memtest\_l ERROR: addr=0x12F00C rd/ref/xor = 0xEFE50484 0xEFE58484 0x00008000

Memtest\_l ERROR: addr=0x12F154 rd/ref/xor = 0xA4CBAACE 0xA4CBABCE 0x00000100

Memtest\_l ERROR: addr=0x13D20C rd/ref/xor = 0xC7DF9980 0xC7DF99C0 0x00000040

Memtest\_l ERROR: addr=0x13D5EC rd/ref/xor = 0xF82DDD6F 0xF82DDF6F 0x00000200

Memtest\_l ( 0:14) Done 1 MB starting at 1 MB, 67 errors (20 40 1 6). 0.105578 sec

## Eye Tests

Prior to running the test, users must:

* Set APU frequency at >1GHz
* Initialize the DDR PLL
* Initialize the DDR controller (using FSBL or zDDR TCL scripts)

If this test is being run from SDK, these operations will be performance by SDK before launching the program.

The test runs out of OCM with the processor caches and MMU enabled. The test is agnostic to memory type (DDR3, DDR4, LPDDR4) or to the memory bus width (32 or 64-bits). This test does not measure the eye width for ECC lane. However, to ensure smooth operation of the program, user must write the entire DRAM address space to ensure valid ECC codes before running the test. The program does not use entire DRAM but this is a good practice nonetheless.

A fixed pattern that is pre-computed and is known to cause maximum errors on the data lanes as the eye width shrinks is run. The memory test is run using a single A53 processor running at speed of 1.0GHz+ with caches enabled. The memory test writes to a 1MB block of memory and as per the default behavior of the system, the block will be read in the APU’s L2 cache and it will be written there. Once the 1MB buffer is written completely, L2 cache flush routine is invocated to ensure the entire block of memory is written back to DRAM at full speed.

The test then reads back the contents of the memory that was written to using a known pattern and calculated any errors on a per-byte lane basis. This number of errors is display after the test is complete and is used to judge the edge of the eye per lane.

|  |  |
| --- | --- |
| Key | Description |
| ‘r’ | Read Eye Analysis |
| ‘w’ | Write Eye Analysis |
| ‘c’ | 2-D Read Eye Analysis |
| ‘e’ | 2-D Write Eye Analysis |
| ‘a’ | Print DRAM Test Start Address |
| ‘l’ | Select Number of Iterations for Test |
| ‘t’ | Specify DRAM Test Start Address |
| ‘s’ | Select DRAM Rank for Test |

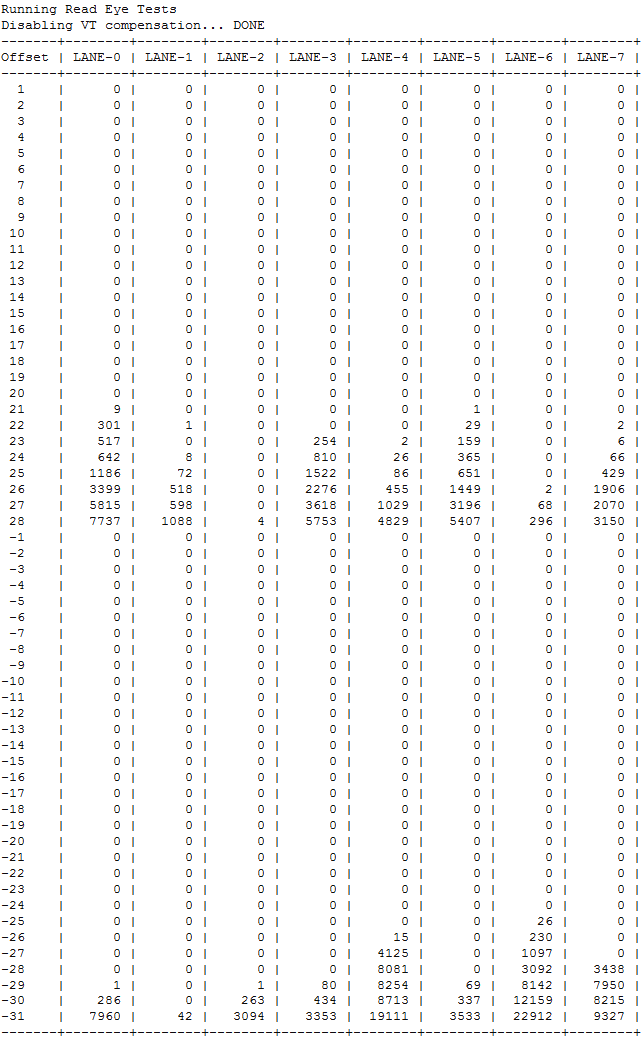
The D-Caches should be disabled in case the ECC is enabled while DDR initialization. When the cache is enabled, the ECC engine complains also about the bad writes happening to memory by cache as part of cache flushing and invalidating, which is why the D-Cache is disabled in this case to rule out the unexpected errors apart from the main test.

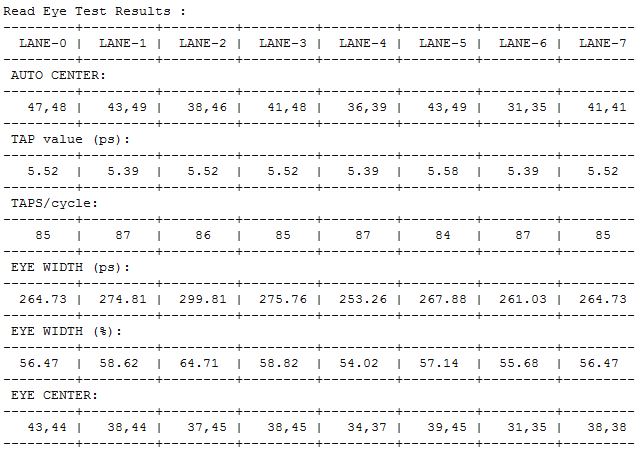
### Read Eye Analysis

This can be invoked by pressing ‘r’ key in the menu. The test will read the eye center as computed after auto training and use that as the baseline to compute the new center by moving the DQS position by one tap count at a time and running the memory test pattern. The resulting output is as displayed in the following picture.

As seen in the displayed output, the test runs for multiple tap counts in each direction to find the edge of the eye per byte lane. Once all lanes have failed the memory test in both directions, the final output is shown (per byte lane) with some key information to judge the width and center of the eye.

After the final results are displayed, the software will restore the original state of any registers. VT compensation is disabled during the test and re-enabled once completed.



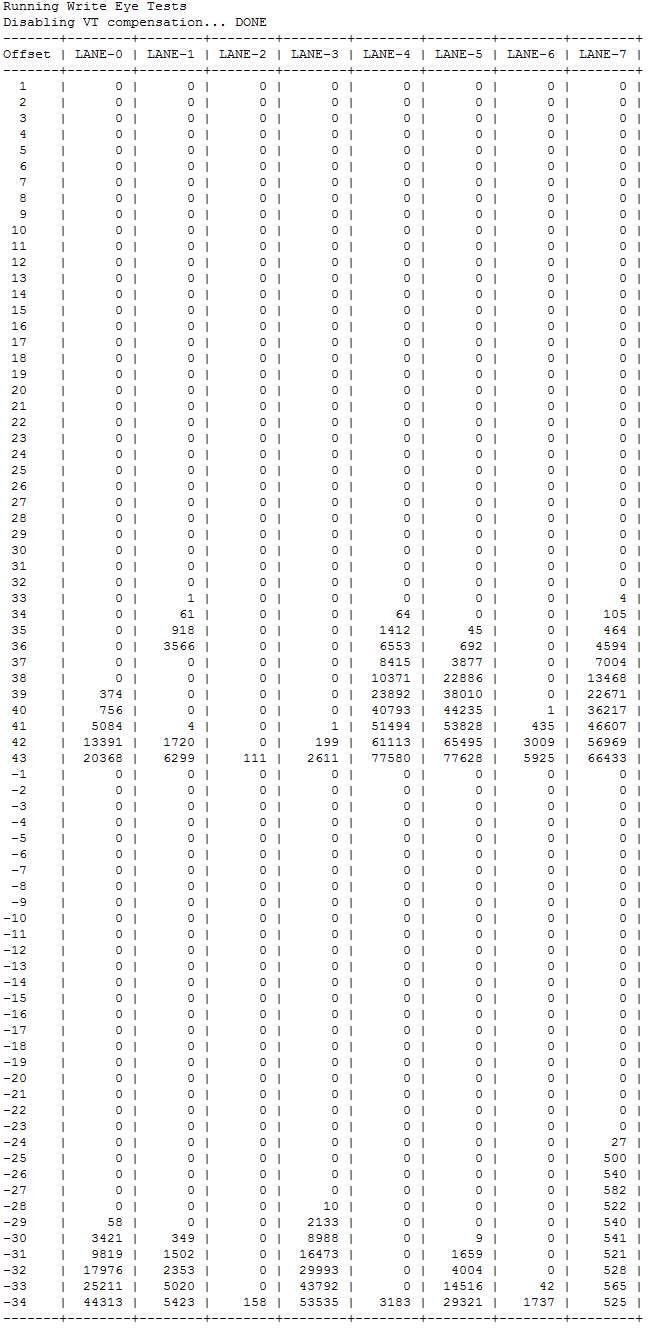


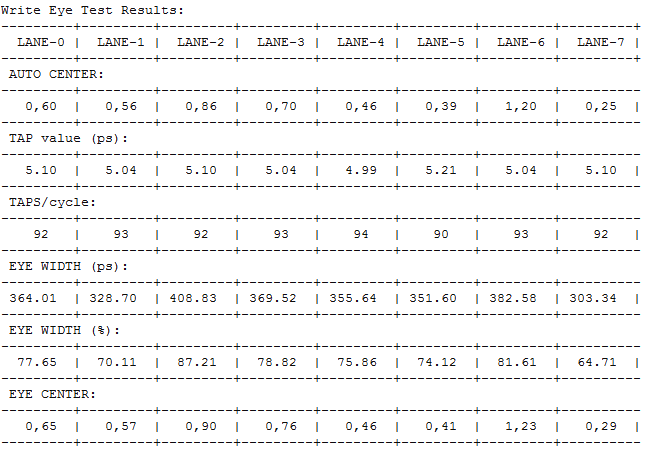
### Write Eye Analysis

This test can be invoked by pressing ‘w’ key in the menu. The test then reads the eye center as computed after auto training and use that as the baseline to compute the new center by moving the DQS position by one tap count at a time and running the memory test pattern. The resulting output is as displayed in the picture above.

Similar to the read eye measurement test, the write eye measurement also runs for multiple tap counts in each direction to find the edge of the eye per byte lane. Once all lanes have failed the memory test in both directions, the final output is shown (per byte lane) with some key information to judge the width and center of the eye.

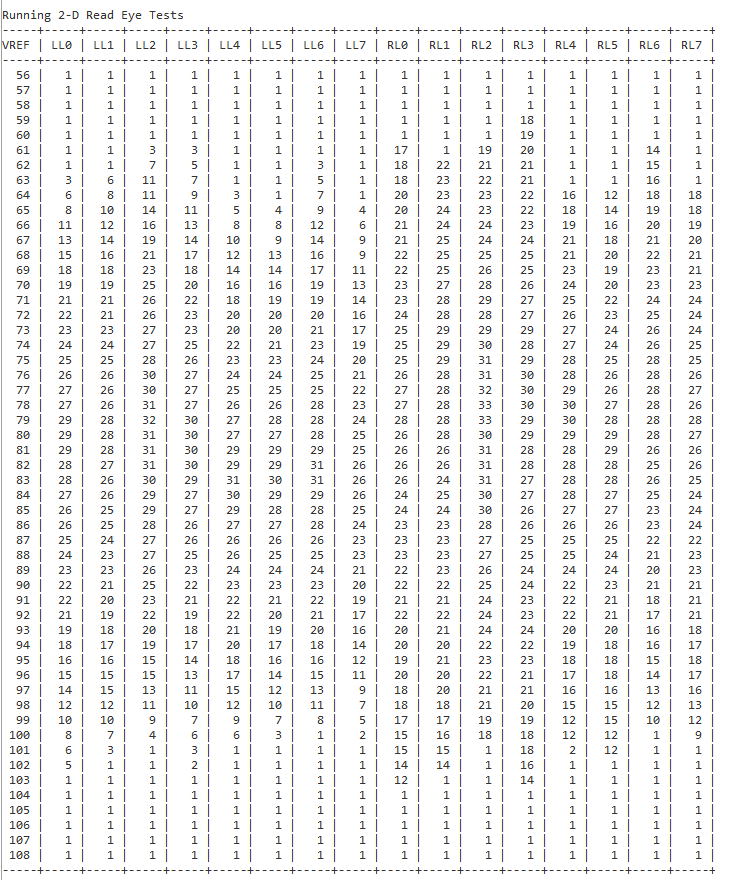
After the final results are displayed, the software will restore the original state of any registers. VT compensation is disabled during the test and re-enabled once completed.





### 2-D Read Eye Analysis

This can be invoked by pressing ‘c’ key in the menu. This test works similarly to the regular read eye analysis except it will test across a range of VREF values during the FPGA read data capture. The test will begin at the bottom of the VREF range and will then try to find the edge of the data valid window by moving the DQS position left and right of the center starting point. The results are show in a table which indicates the VREF value and the number of taps left and right of the center for each byte lane while still detecting valid data.



From the figure above the data is presented as follows:

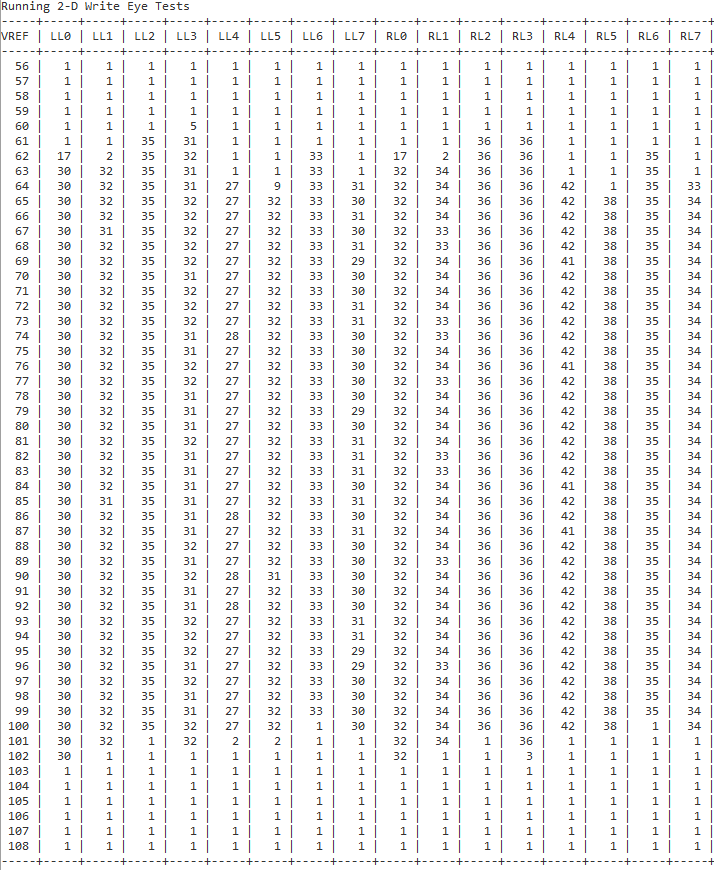
LL0 means Left Byte Lane 0  
LL1 means Left Byte Lane 1  
RL0 means Right Byte Lane 0  
RL1 means Right Byte Lane 1

To evaluate the eye width at a single VREF point for a single byte lane you would add the Left and Right values together. For example using the VREF value of 80 from the figure above we can see LL0 is 29 taps and RL0 is 26 taps, for a total eye width of 55 taps at that VREF point. The tap size is reported in the regular read eye analysis results.

The VREF value is an extrapolation of the VREF setting starting at the bottom of the range to the top of the range. The 2-D eye test results will give you enough information to know the width of the data valid eye for each byte lane as well as the total height of the valid eye. Using the data from above as an example the VREF values of 68 through 97 across all byte lanes report healthy eye widths. If the test results show any byte lanes with significant differences in the healthy VREF range, or if the total eye width is narrower than the other bytes, or if the results for any one byte are asymmetric you can then use this information to try and isolate the cause to the hardware.

### 2-D Write Eye Analysis

This can be invoked by pressing ‘e’ key in the menu. This test works similarly to the regular write eye analysis except it will test across a range of VREF values during the data writes to the DRAM. The test will begin at the bottom of the VREF range and will then try to find the edge of the data valid window by moving the DQS position left and right of the center starting point. The results are show in a table which indicates the VREF value and the number of taps left and right of the center for each byte lane while still detecting valid data.



From the figure above the data is presented as follows:

LL0 means Left Byte Lane 0  
LL1 means Left Byte Lane 1  
RL0 means Right Byte Lane 0  
RL1 means Right Byte Lane 1

To evaluate the eye width at a single VREF point for a single byte lane you would add the Left and Right values together. For example using the VREF value of 78 from the figure above we can see LL0 is 30 taps and RL0 is 32 taps, for a total eye width of 62 taps at that VREF point. The tap size is reported in the regular write eye analysis results.

The VREF value is an extrapolation of the VREF setting starting at the bottom of the range to the top of the range. The 2-D eye test results will give you enough information to know the width of the data valid eye for each byte lane as well as the total height of the valid eye. Using the data from above as an example the VREF values of 65 through 99 across all byte lanes report healthy eye widths. If the test results show any byte lanes with significant differences in the healthy VREF range, or if the total eye width is narrower than the other bytes, or if the results for any one byte are asymmetric you can then use this information to try and isolate the cause to the hardware.

### DRAM Test Start Address

The DRAM test start address can be changed by selecting the ‘t’ option while in the menu.

The DRAM test start address can be printed by selecting the ‘a’ option while in the menu.

### Select Number of Iterations for Test

By default each Memory or Eye test will run only once, however, there is an option to change the number of iterations by selecting the ‘l’ option (‘l’ for **l**oops) in the menu. This option will then prompt the user to enter an integer value for the number of loops which will be run for each test. This feature is most useful when running the write/read Memory Test since it can be set to loop over the entire memory range for multiple hours or even days depending on the loop count. This is useful when trying to reproduce data errors when the hardware has a low error rate or long time to failure. Make sure the Data Cache is enabled to heavily load the memory interface and the Verbose option ‘v’ is enabled so the data compare error details are logged.

### Select the DRAM Rank

If the memory interface is a dual rank design then the rank selection option can be used to isolate tests to the second rank of the design. By default the first rank is always select and the second rank can be selected by using the ‘s’ option and entering 2. To go back to the first rank enter the ‘s’ option when in the menu and then press 1.

## Miscellaneous Options

Other menu items are described below.

|  |  |  |
| --- | --- | --- |
| Key | Name | Description |
| ‘i’ | Information | Print DDR information |
| ‘v’ | Verbose | Toggle verbose mode on/off. If enabled and errors occur during a memory test then the first 10 errors in each sub-test are printed in detail. |
| ‘o’ | D-Cache | Toggle D-Cache enabled or disabled. Keep enabled for highest load on memory interface. |
| ‘b’ | Bus Width | Toggle between 32/64-bit bus widths. Only available when using a 64-bit memory interface so the test can be isolated to the lower 32-bits. |
| ‘q’ | Quit | Exit the DRAM test |
| ‘h’ | Help | Print the DRAM test menu |