Hardware System Verification (HSV)
Vertical Solutions Engineering (VSE)

MRAM
Palladium Memory Model
User Guide

Document Version: 1.6

Document Date: July 2018

Contents

General Information

The Cadence Memory Model Portfolio provides memory device models for the Cadence Palladium XP, Palladium XP II and Palladium Z1 series systems. Optimizing the acceleration and/or emulation flow on these platforms for MMP memory models may require information outside the scope of the MMP user guides and related MMP documentation.

1.1 Related Publications

For basic information regarding emulation and acceleration, please refer to the following documents:

For Palladium XP and Palladium XP II:

UXE User Guide
UXE Library Developer's Guide
UXE Known Problems and Solutions
UXE Command Reference Manual
Palladium XP Planning and Installation Guide
Palladium Target System Developer's Guide
What's New in UXE

For Palladium Z1:

VXE User Guide
VXE Library Developer's Guide
VXE Known Problems and Solutions
VXE Command Reference Manual
Palladium Z1 Planning and Installation Guide
Palladium Target System Developer's Guide
What's New in VXE

MRAM Memory Model

1. Introduction

The Cadence Palladium MRAM Model is based on *EVERSPIN* 64Mb DDR3 Spin Torque MRAM spec (Revision 0.4 9/2014).

Different data width (x4, x8, x16) are available, please consult the memory model catalog for the current available list.

2. Model Release Levels

All models in the Memory Model Portfolio are graded with a release level. This release level informs users of the current maturity and status of the model. All families in the library are graded at one of these levels.

The different levels give an overall indication of the amount of testing, level of quality and feature availability in the model. For details on supported features check the User Guide for that particular model family.

There are three release levels for models in the MMP release.

Release Level		Model Status	Available in Release	Listed in Catalog	Requires Beta Agreement
Mainstream Release	MR	Fully released and available in the catalog for all customers to use.	Yes	Yes	No
Emerging Release	ER	Model has successfully completed Beta engagement(s). Most, but not all features have been tested. Documentation is available.	No	Yes	Yes
Initial Release	IR	Model has completed initial development and has been released to Beta customer(s). The model may have missing features, may not be fully tested, may not have documentation. Model may contain defects.	No	Yes	Yes

Access to Initial and Emerging Release versions of the models will require a Beta Agreement to be signed before the model can be delivered.

3. Overview of Features

The MMP MRAM memory model is fully compatible with DDR3 standards for DRAM operation. The table below shows the MMP MRAM memory model support level for the features from the JEDEC specification *DDR3 SDRAM Standard JESD79-3D* (September 2009).

Table 1: Features List of MRAM Model

FEATURE	SUPPORT	Spec. Section + NOTE
DDR3 (JEDEC 79-3d)		
MRAM Addressing / Configurations (2.11)		
64 Mb	Yes	See MMP catalogue for specific devices
Functional (3)		
States		3.1
Power On	No	
Reset Procedure	Yes	
Initialization	Yes	
ZQ Calibration	Yes	
Idle	Yes	
MRS, MPR Write Leveling	Yes	
Self-Refresh	No	Refresh is not required for MRAM
Refreshing	Yes	Refresh is not required for MRAM
Activating	Yes	
Bank Active	Yes	
Read (RD, RDS4, RDS8)	Yes	
Read A (RDA, RDAS4, RDAS8)	Yes	
Write (WR, WRS4, WRS8)	Yes	
Write A (WRA, WRAS4, WRAS8)	Yes	
Precharging	Yes	
Basic	Yes	3.2
Reset and Initialization	Yes	3.3
Registers	Yes	3.4
Mode Register MR0	Yes	3.4.2 Partial. See section 7.1 of this user guide
Mode Register MR1	Yes	3.4.3 Partial. See section 7.2 of this user guide
Mode Register MR2	Yes	3.4.4 Partial. See section 7.3 of this user guide
Mode Register MR3	Yes	3.4.5 Partial. See section 7.4 of this user guide
MRAM SDRAM Commands (4)		
Commands	Yes	4.1 Partial. See section 8 of this user guide
CKE	Yes	4.2
No Operation Command	Yes	4.3
Deselect Command	Yes	4.4
DLL-off Mode	No	4.5
DLL on/off switching	No	4.6
Input clock frequency change	Yes	4.7
Write Leveling	Yes	4.8
Extended Temperature	No	4.9
Multi-Purpose Register	No	4.10
ACTIVE Command	Yes	4.11
PRECHARGE Command	Yes	4.12
READ Operation	Yes	4.13

FEATURE	SUPPORT	Spec. Section + NOTE
WRITE Operation	Yes	4.14
Refresh Command	Yes	4.15
Self-Refresh Operation	No	4.16
Power-Down Modes	No	4.17
ZQ Calibration Commands	Yes	4.18
DDR3 Specification Sections from *On-Die Termination* (5) through *Electrical Characteristics and AC Timing* (13) are not relevant to MMP emulation models, thus not supported.		

4. Configurations

The following table lists the possible configurations. Not all configurations are available from all vendors. Please consult the appropriate vendor site for details on the parts they offer.

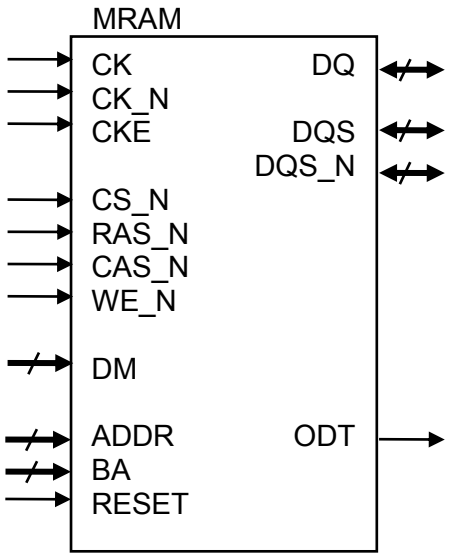
Configuration	16Mb x 4	8Mb x 8	4Mb x 16
# of Banks	8	8	8
Bank Address	BA[2:0]	BA[2:0]	BA[2:0]
Row Address	A[13:0]	A[13:0]	A[13:0]
Column Address	A[6:0]	A[5:0]	A[4:0]

NOTE: Per specification, a portion of the SDRAM address bus, in other words a slice of the column address width, is multiplexed with non-address signals. For example, in A[13:0] the following functions are shared:
A[10] serves as the auto pre-charge bit.
A[12] is BC_n

A[13:0] inputs are used for ADDR. This allocation is reflected in wrappers and core model. Please see the Input/Output Functional Description in the specification for additional detail.

5. Model Block Diagram

The widths of the Addr, Ba, Dm, Dq, and Dqs buses are dependent on the density of the part being used.



6. Model Parameter Descriptions

The following table provides details on the **user adjustable** parameters for the Palladium MRAM Memory Model. These parameters may be modified when instantiating a MRAM wrapper or, if necessary, by modifying the HDL parameter declarations and default values which are exposed for access and debug visibility.

User Adjustable Parameter	Default Value	Description
data_bits	8	Width of data bus DQ
addr_bits	14	Width of row address bus
bank_addr_width	3	Width of bank address bus
row_addr_width	14	Width of row address bus
col_addr_width	6	Width of column address bus
byte_width	8	Width of one byte or word in case of X4
tWLO_in_clocks	3	tWLO time in clocks
SMConfigSpecific_UserData (if defined)	0	Parameter for smart memory data passing. [2:0] bits can be used for device/channel/subpart; [63:61] bits can be used for extension value of total_addr_bits

The following table provides some information about exposed localparams that are NOT user adjustable. On rare occasion the user may find one of these localparam needs adjusting for their configuration. If this case arises, please contact Cadence emulation or MMP support.

Localparam	Default Value	Description
num_bytes	1	data_bits / byte_width
total_addr_bits	23	memory capacity: bank_addr_width + row_addr_width + col_addr_width

Note that there are additional exposed localparams in the model HDL that are not described here nor intended to be described here. These additional localparams are exposed for debugging purposes only and will not be described herein.

7. Address mapping

The array of the MRAM model is mapped into the internal memory of the Palladium system. This array is a single two dimensional array. The mapping of bank, row and column addresses to the internal model array is as follows:

ARRAY_ADDR = {BA, ROW, COL}

This information is required if the memory needs to be preloaded with user data.

The array name in the model hierarchy is: memcore

If {row,bank,col} addressing is needed instead of the default {bank,row,col} addressing, add +define+MMP_RBC to the vlan invocation (IXCOM flow) or to the appropriate HDL-ICE synthesis (Classical flow) command. No value is required for MMP_RBC – only the compile phase define. This option is applicable when using the <model>.vp file.

8. Register Definitions

In the MRAM there are four registers, the Mode Register and three Extended Mode Register. The Palladium MRAM model implements the Mode Register and all three Extended Mode Registers.

8.1. Mode Register MR0

The mode register is implemented in the MRAM model. The model supports the parameter values as described in the tables and sub tables below. Power Down, Write Recovery, and DLL Reset are not supported in this memory model.

15-13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PD	Write Recovery			DLL Reset	Mode	CAS Latency			Burst Type	CAS Latency	Burst Length	

Bit 12	Power Down	
0	Fast Exit	Not Supported
1	Slow Exit	Not supported

Bit 11	Bit 10	Bit 9	Write Recovery
X	X	X	Not supported

Bit 8	DLL Reset	
0	No	Not Supported
1	Yes	Not supported

Bit 7	Mode	
0	Normal	Supported
1	Test	Not supported

Bit 6	Bit 5	Bit 4	Bit 2	CAS Latency
0	0	0	0	Reserved
0	0	1	0	5
0	1	0	0	6
0	1	1	0	7
1	0	0	0	8
1	0	1	0	9
1	1	0	0	10
1	1	1	0	11
0	0	0	1	12
0	0	1	1	13
0	1	0	1	14
0	1	1	1	15
1	0	0	1	16
1	0	1	1	Reserved
1	1	0	1	Reserved
1	1	1	1	Reserved

Bit 3	Burst Type	
0	Nibble Sequential	Supported
1	Interleaved	Supported

Bit 1	Bit 0	Burst Length
0	0	8 Fixed
0	1	BC4 or 8 On the Fly
1	0	BC4 Fixed
1	1	Reserved

8.2. Mode Register MR1

The mode register MR1 is implemented in the MRAM model. The model supports the following parameter values.

15-13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	QOFF	TDQS Enable	0	RTT3	0	Write Leveling	RTT2	DIC2	Additive Latency		RTT1	DIC1	DLL Enable

TDQS Enable, RTT3, RTT2, RTT1, DIC2, DIC1 and DLL Enable are not applicable to the MRAM Palladium Model.

Bit 4	Bit 3	Additive Latency
0	0	0
0	1	CL-1
1	0	CL-2
1	1	Reserved

Bit 7	Write Leveling 1
0	Write Leveling Disabled
1	Write Leveling Enabled

Bit 12	QOFF 1
0	Output Buffer Enabled
1	Output Buffer Disabled

8.3. Mode Register MR2

The mode register MR2 is implemented in the MRAM model. The model supports the following parameter values.

15-13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	RTT_WR		0	SRT	ASR	CAS Write Latency			PASR		

RTT_WR, SRT, ASR and PASR are not applicable to the MRAM Palladium Model.

Bit 5	Bit 4	Bit 3	CAS Write Latency
0	0	0	5
0	0	1	6
0	1	0	7
0	1	1	8
1	0	0	9
1	0	1	10
1	1	0	11
1	1	1	12

8.4. Mode Register MR3

The mode register MR3 is implemented in the MRAM model. The model supports the following parameter values.

15-3	2	1	0
0	MPR	MPR Location	

Bit 2	MPR 1
0	Normal Operation
1	Dataflow from MPR

Bit 1	Bit 0	MPR Location 1
0	0	Predefined Pattern
0	1	Reserved
1	0	Reserved
1	1	Reserved

9. Commands

The MRAM model provides command support as indicated in the table below.

Command	Abbreviation	Command Accepted	Functionality Supported
Mode Register Set	MRS	Yes	Yes
Refresh	REF	Yes	Yes
Self Refresh Entry	SRE	Yes	No
Self Refresh Exit	SRX	Yes	No
Single Bank Precharge	PRE	Yes	Yes
Precharge all Banks	PREA	Yes	Yes
Bank Activate	ACT	Yes	Yes
Write (Fixed BL8 or BC4)	WR	Yes	Yes
Write (BC4, on the Fly)	WRS4	Yes	Yes
Write (BL8, on the Fly)	WRS8	Yes	Yes
Write with Auto Precharge (Fixed BL8 or BC4)	WRA	Yes	Yes
Write with Auto Precharge (BC4, on the Fly)	WRAS4	Yes	Yes
Write with Auto Precharge (BL8, on the Fly)	WRAS8	Yes	Yes
Read (Fixed BL8 or BC4)	RD	Yes	Yes
Read (BC4, on the Fly	RDS4	Yes	Yes
Read (BL8, on the Fly)	RDS8	Yes	Yes
Read with Auto Precharge (Fixed BL8 or BC4)	RDA	Yes	Yes
Read with Auto Precharge (BC4, on the Fly)	RDAS4	Yes	Yes
Read with Auto Precharge (BL8, on the Fly)	RDAS8	Yes	Yes
No Operation	NOP	Yes	Yes
Device Deselected	DES	Yes	Yes
Power Down Entry	PDE	Yes	No
Power Down Exit	PDX	Yes	No
ZQ Calibration Long	ZQCL	Yes	No
ZQ Calibration Short	ZQCS	Yes	No

10. Initialization Sequence

The MRAM model requires that the memory controller follows the initialization sequence as documented in the specification. The sequence basically entails the following steps:

1. Assert RESET
2. De-assert RESET
3. Start clocks
4. Wait for CKE to asserted
5. Write to Mode Register 2
6. Write to Mode Register 3
7. Write to Mode Register 1
8. Write to Mode Register 0
9. Issue ZQC command

The model requires that these steps be performed in the correct sequence in order to complete initialization. The model will not respond to any others commands unless this sequence is completed.

11. SWI Smart Memory Interface

This MRAM model supports a fixed, Verilog macro controlled interface, or “hooks”, to one of the Smart Memory components in Cadence’s Software Integrator (SWI) product. Software Integrator (SWI) is a support library that is part of Cadence’s Hybrid Solution--a multi-tool system level solution that runs in a hybrid PXP + IES simulation mode and targets the increasing number of SoC performance conscious projects requiring the booting of commercial operating systems and the running of complex software-driven tests against an accurate model of the SoC prior to tapeout.

For additional details about the SWI product at large please consult the SWI product documentation which includes a user guide. This documentation can be accessed via support.cadence.com and is located on the Product Pages/Product Manuals link where SWI 13.1 is located with other Functional Verification products.

The user of the SWI solution who is integrating this core model to the corresponding SWI Smart Memory component side should define the Verilog macro “MMP_SM” to enable the Smart Memory interface. This will enable the portion of the interface that resides in the MMP model core, thus completing access to the implemented SWI Smart Memory functionality.

Table 2 : SWI Smart Memory Verilog Define

`define Macro purpose	Possible `define Values
Set the Smart Memory interface to compile “in” as part of the memory model	MMP_SM

The SWI Smart Memory interface includes the signals shown in Table 3: SWI Smart Memory Interface Signals. It is outside the MMP scope to treat the integration of the MMP model into a hybrid solution. For additional details, please consult the SWI documentation and other Hybrid Solution documentation.

Table 3: SWI Smart Memory Interface Signals

NAME	DECLARATION	DESCRIPTION
sm_raddr	output [(total_addr_bits-1):0] sm_raddr	Smart Memory read address
sm_re	output sm_re	Smart Memory read enable
sm_dout	input [data_bits-1:0] sm_dout	Smart Memory read data
sm_waddr	output [(total_addr_bits-1):0] sm_waddr	Smart Memory write address
sm_we	output sm_we	Smart Memory write enable
sm_din	output [data_bits-1:0] sm_din	Smart Memory write data
sm_bw	output [data_bits-1:0] sm_bw	Smart Memory data mask
verbosity	input [3:0] verbosity	Smart Memory debug info display level control

The Smart Memory interface includes a user adjustable parameter that passes user data from wrapper layers that are external to this MMP model into MMP model. The single 64-bit parameter is subdivided by field to accommodate data as shown in Table 4: SWI Smart Memory User Adjustable Parameters below. This parameter defaults to a value of '0' and is managed by the SWI product Smart Memory component. For additional details, please consult the SWI documentation and other Hybrid Solution documentation.

Table 4: SWI Smart Memory User Adjustable Parameters

PARAMETER	DESCRIPTION
parameter [63:0] SMConfigSpecific_UserData	Smart Memory User Data Passing
FIELD	**DESCRIPTION**
[2:0]	Used for specifying device/channel/subpart
[63:61]	Extension value of total_addr_bits

The Smart Memory interface includes non-user adjustable localparams and parameters as shown in Table 5: SWI Smart Memory Non-User Adjustable Parameters below. Note that a localparam of the same name (total_addr_bits) but of different size is part of the standard MMP core model.

Table 5: SWI Smart Memory Non-User Adjustable Parameters

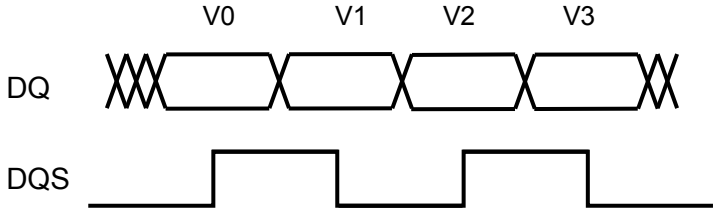
LOCALPARAM	VALUE	DESCRIPTION
total_addr_bits	bank_addr_width+row_addr_width +col_addr_width + SMConfigSpecific_UserData[63:61]	Memory capacity width

The SWI Smart Memory interface has dependencies on the inclusion of external file(s). For additional details about purpose and content of any Smart Memory related external files, please consult the SWI documentation and other Hybrid Solution documentation.

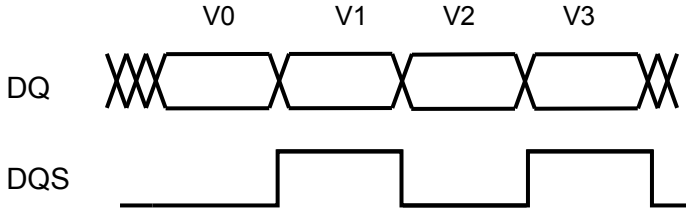
```
`ifdef MMP_SM
   `include "cdn_sm_mapDRBCToLinAdr.vh"
`endif
```

12. Handling DQS in Palladium Memory Models

For writes to a DDR memory, industry datasheets show each DQS edge centered within the corresponding valid period (v0, v1, v2, etc.) of DQ, as in the following diagram.

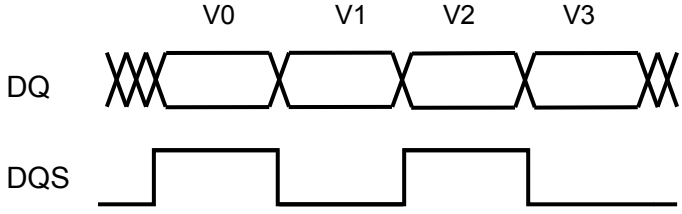


For DDR models provided by Cadence for Palladium, if the design drives DQ and DQS signals with the above timing, the DDR memory will behave correctly. However, to obtain this timing in Palladium, the fastest design clock must toggle twice as frequently as the DQS signal. If this faster clock is not needed for any other reason, the presence of the faster clock will usually cause an unnecessary 2X slowdown in emulation speed. To eliminate the need for a faster clock, you can have the design generate each DQS edge at the end of the corresponding DQ valid period (rather than the middle), as in the following diagram:



Note that the first DQS edge is at the *end* of first valid DQ, not at the beginning.

For reads from the DDR model, the DDR model will drive DQ and DQS with the first DQS edge at the *beginning* of the first valid data, not at the end:



The DDR model behaves this way to conform with industry datasheets for DDR memories. The design reading the data from the DDR model must delay the DQS signal, and use the delayed-DQS signal to sample the DQ. A delay of one Q_FDP0B should work fine, even in CAKE 1X mode. If you are using CAKE 1X mode and the DDR clock is the fastest design clock, the DQ signal will change twice per FCLK, and the Q_FDP0B delaying DQS will provide one-half FCLK delay, so that each delayed-DQS edge is at the end of the corresponding data valid period.

To delay the DQS signal, a commonly used approach is to create a special pad cell for DQS, that has a Q_FDP0B delay cell inserted on the path that leads from the DDR memory into the design.

The user may insert delays into pad cells (or elsewhere in the design) using the below code example which leverages ixc_pulse, an internal primitive that can be used to access FCLK and to create controlled delay, for IXCOM flow and leverages the Q_FDP0B primitive for delay generation in the Classic ICE flow. For more detailed information about ixc_pulse please reference the *UXE User Guide* section called *Generating Pulses*. There is no need for the user to define IXCOM_UXE for the Verilog macro; it is predefined for the user in IXCOM flow. Note that in UXE 13.1.0 and prior the equivalent pulse generating function was named axis_pulse.

```
// Flow independent delay cell
module pxp_fclk_delay (in, out_delay);
input in;
output out_delay;

reg out_delay;

`ifdef IXCOM_UXE
  wire VCC=1'b1;
  ixc_pulse #(1)(Fclk,VCC);
  always @(posedge Fclk)
    out_delay <= in;
`else
  Q_FDP0B fclk_dly (.D(in), .Q(out_delay));
`endif

endmodule
```

13. Configuration Parameters

There are a few parameters that can be set to adjust the configuration of a model. One limitation is that the sum of bank_addr_width, row_addr_width, and col_addr_width should be less than 32 bits due to the 2G address limit in HDL-ICE. The following parameters are available in the protected rtl version (<model_name>.vp file) of a model:

```
parameter   data_bits = 16;  // width of DQ bus
parameter   addr_bits = 13;  // row address width
parameter   bank_addr_width = 3;  // bank address width
parameter   row_addr_width = addr_bits;  // row address width
parameter   col_addr_width = 10;  // column address width
parameter   byte_width = 8;   // byte_width = 4 if DQ width is 4; otherwise byte_width = 8
parameter   tWL0_in_clocks = 3;   // tWL time in number of clocks
```

14. Compile and Emulation

Each model is provided as protected RTL files (*.vp). The files need to be synthesized prior to the back-end Palladium compile. An example of the command for compilation (including synthesis) and run of this model in the IXCOM flow is shown below.

```
ixcom -64 +sv -ua +dut+emd3d064m04 \
    ./emd3d064m04.vp \
    -incdir ../../../utils/cdn_mmp_utils/sv \
    ../../../utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
    ……

xeDebug -64 --ncsim \
    -sv_lib ../../../utils/cdn_mmp_utils/lib/64bit/libMMP_utils.so -- \
    -input auto_xedebug.tcl
```

The script below shows two example for Palladium classic ICE synthesis:

```
1)
hdlInputFile emd3d064m04.vp
hdlImport -full -2001 -l qtref
hdlOutputFile -add -f verilog emd3d064m04.vg
hdlSynthesize -memory -keepVhdlCase -keepRtlSymbol -keepAllFlipFlop
emd3d064m04
……

2)
vavlog emd3d064m04.vp

vaelab -keepRtlSymbol -keepAllFlipFlop -outputVlog emd3d064m04.vg
emd3d064m04
……
```

NOTE: It is common for Palladium flows to require –keepallFlipFlop since it removes optimizations that are in place by default. For example, without –keepAllFlipFlop, HDL-

ICE can remove flops with constant inputs and merge equivalent FF. The picture above is modified a bit when ICE ATB mode (–atb) is used since then a constant input FF is only optimized out when there is no initial value for it or the initial value is the same as the constant input value.

It is also common for Palladium flows to require –keepRtlSymbol. This option enables the HDL Compiler to keep original VHDL RTL symbols, such as ".", whenever possible. In other words, it maps VHDL RTL signal name a.b to the netlist entry, \a.b. Without this modifier, the signal name would otherwise be converted to a_b in the netlist.

If the recommended compile script includes the aforementioned options, the user must include them to avoid affecting functionality of the design.

15. Debugging

Below is a list of recommended debugging features.

- For issues that may not be MRAM specific please review the *Memory Model Portfolio FAQ for All Models User Guide.*

- **Debug Display:** The Palladium MRAM memory model has available a built-in debug methodology called MMP Debug Display that is based on the Verilog system task $display. Please see the *Palladium Memory Model Debug Display User Guide* in the release docs directory for additional information. Some of the debug information displayed when the Debug Display macro options are enabled includes:

    - MRAM commands received
    - MRAM command decoding and states
    - MRAM mode register operation
    - MRAM memory accesses
    - MRAM error conditions

- **Golden waveform:** A package with a reference waveform is available which shows the following command sequence:
    - setting MR2
    - setting MR3
    - setting MR1
    - setting MR0 // initialization complete
    - -> normal READ and WRITE commands

16. Revision History

The following table shows the revision history for this document

Date	Version	Revision
June 2015	1.0	Initial Release
July 2015	1.1	Update Cadence naming on front page
January 2016	1.2	Update for Palladium-Z1 and VXE
April 2016	1.3	Update SM interface sm_we
July 2016	1.4	Remove hyphen in Palladium naming
January 2018	1.5	Modify header and footer
July 2018	1.6	Update for new utility library