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

SD CARD
Palladium Memory Model
User Guide

Document Version: 2.7

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

SD Card Palladium Memory Models

1. Introduction

The Cadence Palladium SD card Model is based on the following specification:

SD Physical Layer Specification version 4.20 (September 2013)

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 and may not have documentation. Model may contain defects.	No	Yes	Yes

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

3. Features

The Cadence SD CARD Palladium memory model is based upon *SD Specifications Part 1 Physical Layer Specification Version 4.20 (September 2013).* The SD CARD features and the level of support provided in the MMP memory model are listed below.

Feature	Support	Spec. Section
Protocols		
SD bus protocol	Yes	3.6.1
SPI bus protocol	No	3.6.2
UHS-II bus protocol	No	3.6.3
Bus Speed Mode		
Default speed	Yes	3.9
High speed	Yes	3.9
SDR12	Yes	3.9
SDR25	Yes	3.9
SDR50	Yes	3.9
SDR104	Yes	3.9
DDR50	Yes	3.9
Functions		
Card reset	Yes	4.2.1
Operating condition validation	Yes	4.2.2
Card initialization and identification process	Yes	4.2.3
Bus signal voltage switch	Yes	4.2.4
Wide bus selection/deselection	Yes	4.3.1
Read/Write	Yes	4.3.3 & 4.3.4
Erase	Yes	4.3.5
Write protect management	Yes	4.3.6
Card lock/unlock	Yes	4.3.7
APP command	Yes	4.3.9
Switch function command	Yes	4.3.10
Clock control	Yes	4.4
CRC	Yes	4.5
Card status and SD status	Yes	4.10
Card Registers		
OCR registers	Yes	5.1
CID registers	Yes	5.2
CSD (v1.0 and v2.0) registers	Yes	5.3
RCA registers	Yes	5.4
DSR registers	Yes	5.5
SCR registers	Yes	5.6
EXT registers	Yes	5.7
Hardware Interface		
Hot insertion and removal	Yes	6.1
Card detection	Yes	6.2
Power protection	No	6.3
Power scheme	No	6.4
ESD requirement	No	6.8

4. Model Configurations

The SD Card model is currently provided and configured in generic sizes described by the standard specification. The SD Card model comprises a protected RTL source file (SDcard_pd.vp in <release>/SDcard) along with a selection of wrapper files (*.v files in <release>/SDcard/wrappers) that correspond to the available configurations. The table below lists the available generic configurations.

Model Wrapper Name	Size	Generic Configuration
sd_card_256m	256MB	256MB SDSC 4.2
sd_card_512m	512MB	512MB SDSC 4.2
sd_card_1g	1GB	1GB SDSC 4.2
sd_card_2g	2GB	2GB SDSC 4.2
sd_card_4g	4GB	4GB SDHC 4.2
sd_card_8g	8GB	8GB SDHC 4.2
sd_card_16g	16GB	16GB SDHC 4.2
sd_card_32g	32GB	32GB SDHC 4.2
sd_card_64g	64GB	64GB SDXC 4.2

Please see section 10 Memory Arrays in the SD card Model for a description of any model side files.

5. Model Pin Description

In addition to the standard IO pins for the SD card as per the specification there are several other IO on the SD card Palladium Memory Model that are there to provide flexibility and and to support additional model features..

Typically, but not always, a controller will support pullups on the CMD and DAT buses. For this model there is a Verilog macro to add pullups for the case where a controller which DOES NOT support pullups. The Verilog macro is ***MMP_SDCARD_HOST_NO_PULLUP***. If there is no pullup on CMD and DAT bus in the user's test environment (out of SD model), the user should define this Verilog macro.

Pin name	Direction	Description
CLK	Input	SD card CLK input
CMD	Inout	SD card CMD input
DAT0…DAT3	Inout	SD card 4bit bi-directional databus
CID_IN[127:0]	Input	Initial value for CID register. This value is latched by the model 2 clks after start of emulation
CSD_IN[127:0]	Input	Initial value for CSD register. This value is latched by the model 2 clks after start of emulation
SCR_IN[63:0]	Input	Initial value for SCR register. This value is

		latched by the model 2 clks after start of emulation
CARD_INSERT	Input	Users can force the I/O “CARD_INSERT” to simulate the situation that card is inserted or removed. It can be set to 0 when the process is running, that means the card is removed.
CARD_DETECTION_SUPPORT	Input	For users who do not need “card detection” function, force this pin to ‘0’.
INJECT_DATA_CRC_ERROR	Input	When asserted the generated CRC for a read command will have an incorrectly calculated CRC

6. Model Parameter Descriptions

The following table provides details on the **user adjustable** parameters for the Palladium SD card Memory Model. These parameters may be modified when instantiating an MMC 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
SDHC_SUPPORT	1	Indicates if device supports SDHC or not
ADDR_WIDTH	29	Address bit width defines memory size as (1 << ADDR_WIDTH). Should match the configuration of the general purpose partition size, otherwise the ADDR_WIDTH will be mismatched
MEMCORE_DATA_WIDTH_MULT_LOG2	3	Memory core data width = 8 * (1 << MEMCORE_DATA_WIDTH_MULT_LOG2)
BLK_LEN_MAX	9	Half of buffer size for receiving data, shall be the same value as MAX_WRITE_BL_LEN in CSD
MIN_WRITE_PROTECT_GROUP_SIZE	3072	Minimum write protection group size, 512 x (SECTOR_SIZE+1) x (WP_GRP_SIZE+1)
Switch Function Parameters		
FUNCTION_GROUP_1_1_8_V	16’b1000000000011111	Supported function group1 default value if device is working on 1.8V
FUNCTION_GROUP_1_3_3_V	16’b1000000000000011	Supported function group1 default value if device is working on 3.3V
FUNCTION_GROUP_2	16’b1000000000011011	Supported function group2 default value
FUNCTION_GROUP_3	16’b1000000000001111	Supported function group3 default value
FUNCTION_GROUP_4	16’b1000000000011111	Supported function group4 default value
FUNCTION_GROUP_5	16’b1000000000000001	Supported function group5 default value
FUNCTION_GROUP_6	16’b1000000000000001	Supported function group6 default value
DATA_STRUCTURE_VER	2’b00	2’b01 -> do judge if the selected

		function is busy; other value -> do not care the busy status

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**
NID	4	Command end to response R2 or R3 start in clock cycles
NCR	8	Command end to response R1 start in clock cycles
NAC	62	Read data access time in clock cycles
R1b_READY_COUNT	4	R1b ready time in clock cycles
MEMFILE_LITTLE_ENDIAN	1	If 1, memory contents file is little endian. Byte lane is swapped. If 0, memory contents file is big endian. Byte lane is NOT swapped
EXTRA_DDR_CYCLE	1	Controls alignment of the DDR relative to the data. When set the first data is driven on the negedge of the clock so it can be latched by the Host on the posedge of the clock
VOLTAGE_2V7_3V6_SUPPORT	9'b111111111	Indicate if device support this voltage or not
VOLTAGE_SWITCH_ALLOW	1	Indicate if the voltage can be changed or not
RCA_1	16’h1234	The RCA value sent when CMD3 is issued
RCA_2	16’h5678	The RCA value sent when CMD3 is issued
RCA_3	16’h9abc	The RCA value sent when CMD3 is issued
RCA_4	16’hdef0	The RCA value sent when CMD3 is issued
USER_WRITE_RECOVERY_TIME_PDCK	100	Write recovery time for user data area, number of PD fast clock cycle (suggested not to be changed)
PWD_WRITE_RECOVERY_TIME_PDCK	100	Write recovery time for password area, number of PD fast clock cycle (suggested not to be changed)
CSD_WRITE_RECOVERY_TIME_PDCK	150	Write recovery time for CSD register, number of PD fast clock cycle (suggested not to be changed)
EXT_WRITE_RECOVERY_TIME_PDCK	550	Write recovery time for extension register area, number of PD fast clock cycle (suggested not to be changed)
CMD11_DISABLE_CLK_THRESHOLD_TIME_PDCK	2000	if CLK stays LOW during this period of time, it will be considered disabled, threshold = 2ns * CMD11_DISABLE_CLK_THRESHOLD_TIME_PDCK, measured in PD fast clk
CMD11_ENABLE_CLK_THRESHOLD_TIME_SDCLK	50	if CLK stays ACTIVE during this period of time, it will be considered enabled, threshold is measured in SD CLK

EXT_REG_MEM_ADDR_WIDTH	9+8+4	512byte x 256page x 16func, 2MB
EXT_REG_IO_ADDR_WIDTH	9+8+3	512byte x 256page x 8func, 1MB
DATA_PORT_ADDR_MEM_FUNC_0	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_1	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_2	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_3	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_4	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_5	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_6	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_7	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_8	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_9	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_A	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_B	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_C	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_D	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_E	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_MEM_FUNC_F	512	For extension register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_IO_FUNC_0	512	For extension IO register functions, indicate the default data port address of the corresponding function

DATA_PORT_ADDR_IO_FUNC_1	512	For extension IO register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_IO_FUNC_2	512	For extension IO register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_IO_FUNC_3	512	For extension IO register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_IO_FUNC_4	512	For extension IO register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_IO_FUNC_5	512	For extension IO register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_IO_FUNC_6	512	For extension IO register functions, indicate the default data port address of the corresponding function
DATA_PORT_ADDR_IO_FUNC_7	512	For extension IO register functions, indicate the default data port address of the corresponding function
SD status parameters		
SECURED_MODE_PARAM	0	Default value of the field of SD status
SD_CARD_TYPE_PARAM	16’h0000	Default value of the field of SD status
SIZE_OF_PROTECTED_AREA_PARAM	32’h10000000	Default value of the field of SD status
SPEED_CLASS_PARAM	8’h04	Default value of the field of SD status
PERFORMANCE_MOVE_PARAM	8’h00	Default value of the field of SD status
AU_SIZE_PARAM	4’h9	Default value of the field of SD status
ERASE_SIZE_PARAM	16’h0000	Default value of the field of SD status
ERASE_TIMEOUT_PARAM	6’d0	Default value of the field of SD status
ERASE_OFFSET_PARAM	2’d0	Default value of the field of SD status
UHS_SPEED_GRADE_PARAM	4’h0	Default value of the field of SD status
UHS_AU_SIZE_PARAM	4’h0	Default value of the field of SD status

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. Commands and Responses

The model supports the following commands.

Class	Commands
Class 0	CMD0, CMD2, CMD3, CMD4, CMD7, CMD8, CMD9, CMD10, CMD11, CMD12, CMD13, CMD15
Class 2	CMD16, CMD17, CMD18, CMD19, CMD23
Class 4	CMD16, CMD23, CMD24, CMD25, CDM27
Class 5	CMD32, CMD33, CMD38
Class 6	CMD28, CMD29, CMD30
Class 7	CMD42
Class 8	CMD55, ACMD6, ACMD13, ACMD22, ACMD23, ACMD41, ACMD42, ACMD51
Class 10	CMD6
Class 11	CMD48, CMD49, CMD58, CMD59

The model supports all response types, R1, R1b, R2, R3, R6 and R7.

The following commands are not supported:

Class	Commands
Class 2	CMD20
Class 7	CMD40
Class 8	CMD56
Class 9	CMD52, CMD53, CMD54

8. Synthesis and Compilation of the Model

The model is provided as a protected RTL source file for the core memory along with a selection of wrapper files that correspond to various configurations. Please see section 4 Model Configurations. The SDcard_pd.vp file and selected wrapper file need to be synthesized prior to compiling for Palladium, either with HDL-ICE in Classic ICE flow or with IXCOM flow.

An IXCOM compile and run example is provided below:

```
ixcom   -64bit +sv -ua \
        +dut+sd_card_2g \
        <path to MMP install>/SDcard_pd.vp \
        <path to MMP install>/SDcard/wrappers/sd_card_2g.v \
        -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 scripts below show two examples for Palladium classic ICE synthesis:

1)
```
hdlInputFile -add <selected_wrapper>.v
hdlInputFile –add SDcard_pd.vp
hdlImport -full -2001 -l qtref
hdlOutputFile -add -f Verilog <selected_wrapper>.vg
hdlSynthesize -memory -keepRtlSymbol -keepAllFlipFlop <selected_wrapper>
……
```

2)
```
vavlog      SDcard_pd.vp \
            <selected_wrapper>.v \
vaelab      -keepRtlSymbol -keepAllFlipFlop -outputVlog
<selected_wrapper>.vg <selected_wrapper>
```

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.

9. Memory Configuration

The array size of the model can be altered by the user with the parameter ADDR_WIDTH. ADDR_WIDTH should match the value of general purpose partition size configurations, or there may be bit-select or part-select index errors when the main array size is mismatched with general purpose partition size. This can be done at synthesis time with the –D switch on the hdlSynthesize command.

```
hdlSynthesize –memory –keepAllFlipFlop –DADDR_WIDTH=20 SDcard_pd
```

Alternatively it can be done by adjusting the parameter when instantiating the model. Users can also generate general purpose partitions by adjusting the related parameters or just use the default ones.

```
SDcard_pd #(.ADDR_WIDTH(20)) i1(
  .CMD(CMD),
  .CLK(CLK),
  .DAT0(DAT0),
  ……
);
```

10. Large Memory Management

Please see user guide titled MMP_MANAGING_LARGE_MEMORY_MODELS.pdf.

11. Memory Arrays in the SD card Model

There are several arrays in the SD card model, the main array, tune_pattern array, extension register arrays. These arrays can be preloaded at runtime with the Palladium runtime environment using the memory commands. The names of the areas are listed below.

Array Name in Model	Function
mem	Main Array
tune_pattern	Array contains fixed pattern for tuning sequence
ext_reg_mem	Extension memory space
ext_reg_io	Extension IO space
ext_reg_mem_func_dev_0	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_1	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_2	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_3	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_4	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_5	Corresponding function device storage for

	extension memory space access
ext_reg_mem_func_dev_6	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_7	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_8	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_9	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_a	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_b	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_c	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_d	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_e	Corresponding function device storage for extension memory space access
ext_reg_mem_func_dev_f	Corresponding function device storage for extension memory space access
ext_reg_io_func_dev_0	Corresponding function device storage for extension IO space access
ext_reg_io_func_dev_1	Corresponding function device storage for extension IO space access
ext_reg_io_func_dev_2	Corresponding function device storage for extension IO space access
ext_reg_io_func_dev_3	Corresponding function device storage for extension IO space access
ext_reg_io_func_dev_4	Corresponding function device storage for extension IO space access
ext_reg_io_func_dev_5	Corresponding function device storage for extension IO space access
ext_reg_io_func_dev_6	Corresponding function device storage for extension IO space access
ext_reg_io_func_dev_7	Corresponding function device storage for extension IO space access

The model is provided with example initialization file for tune_pattern array. The content is mandatory by SD card standard specification.

11.1. Address mapping to access memory arrays

Users generally need to pay attention to the address mapping when they access the main array --- “**mem**” with Palladium runtime load and dump commands. The address mapping for load and dump commands can be different from that in the read/write commands. For models with programmable array widths, such as this model, the address mapping for accessing the core memory array can become complicated by the fact that even though the real memory device implements an 8bit array width, for the large memory model configurations the data need to be reformatted into 32 or 64 bit load and dump files for these runtime commands. This section explain the reason for this implementation and then suggests a procedure that alleviates the need for data modification.

There is a parameter “**MEMCORE_DATA_WIDTH_MULT_LOG2**” in this model’s size/configuration wrappers which indicates the width of the real memory core implemented in Palladium. The width is:

width = 8bits x (2^ MEMCORE_DATA_WIDTH_MULT_LOG2)

For example, in the 1GB model, width = 8 x (2^3) = 64 bits. That means if users use write commands (CMD24/25) to access the memory with a logical start address like 0x1000, the actual physical address 0x200 (0x1000>>3) is accessed. Users therefore need to “dump” 0x200 to retrieve the written data instead of the expected 0x1000.

On the other hand, if the user issues mem –load for the physical address 0x200, the relevant read commands (CMD17/18) should access the core memory with the logical start address 0x1000 (0x200<<3). The data width for each address location in the preload file should be the same as the width of memcore.

Palladium runtime memory load and dump operations support various data formats which can vary the user’s load/dump performance and data file content.

For programmable array widths that are larger than 8bits, the user has the option to take advantage of the Palladium utility memTran to convert a memory data file formatted in 8bits wide data—i.e., a readmemh format—to a headerless binary formatted file—i.e. raw2 format—that can be loaded via the memory –load command without the user needing to be aware of or concerned about the memory width.

The steps are as follows for an example using readmemh formatted input:

1. User creates a “readmemh” formatted, or other Palladium supported format,” data file with 8bits wide data
2. User converts the readmemh file using Palladium’s *memTran* utility into a file with “pd_raw2” format.
3. User loads the memory at runtime using the command ‘memory –load %pd_raw2’ to load the data file into memory. Because this command does not care about the memory width and simply streams the data into the assigned memory at the specified start address, the user can perform the memory access and load without modifying the data file from the standard readmemh format. The user does need to remain cognizant of the preload start address if the start address is not address 0. The start address is the physical address of the memcore that is a 32bit or 64bit memory address. The description of how to calculate the physical address for the memory was discussed above.

To look at a specific example, the user may have a memdata file called test_load.h that contains 8 bits wide data. The following memTran command converts the test_load.h file to raw2 formatted data (pd_raw2) in a file called test_load.bin and then compares the content of the two files.

```
memTran -translate %readmemh memdata/test_load.h %pd_raw2 test_load.bin -width 8 -depth 256
memTran -compare   %readmemh memdata/test_load.h %pd_raw2 test_load.bin -width 8 -depth 256
```

During runtime, the memory –load command can be used as below:

```
    memory -load %pd_raw2 sd_card_256m.dut.mem -file test_load.bin –nochecksize
```

And the memory –dump command as below:

```
    memory -dump %pd_raw2 sd_card_256m.dut.mem -file test_dump.bin
```

The dump command may be followed by the appropriate memTran commands to convert the binary formatted file back to 8-bit width format.

```
    memTran -translate %pd_raw2 test_dump.bin %readmemh test_dump.bin.postconv.h -width 64
    memTran -compare %pd_raw2 test_dump.bin %readmemh test_dump.bin.postconv.h -width 64 -depth
335
```

An additional consideration is that load/dump operations are typically faster with binary formatted files such as raw2 formats.

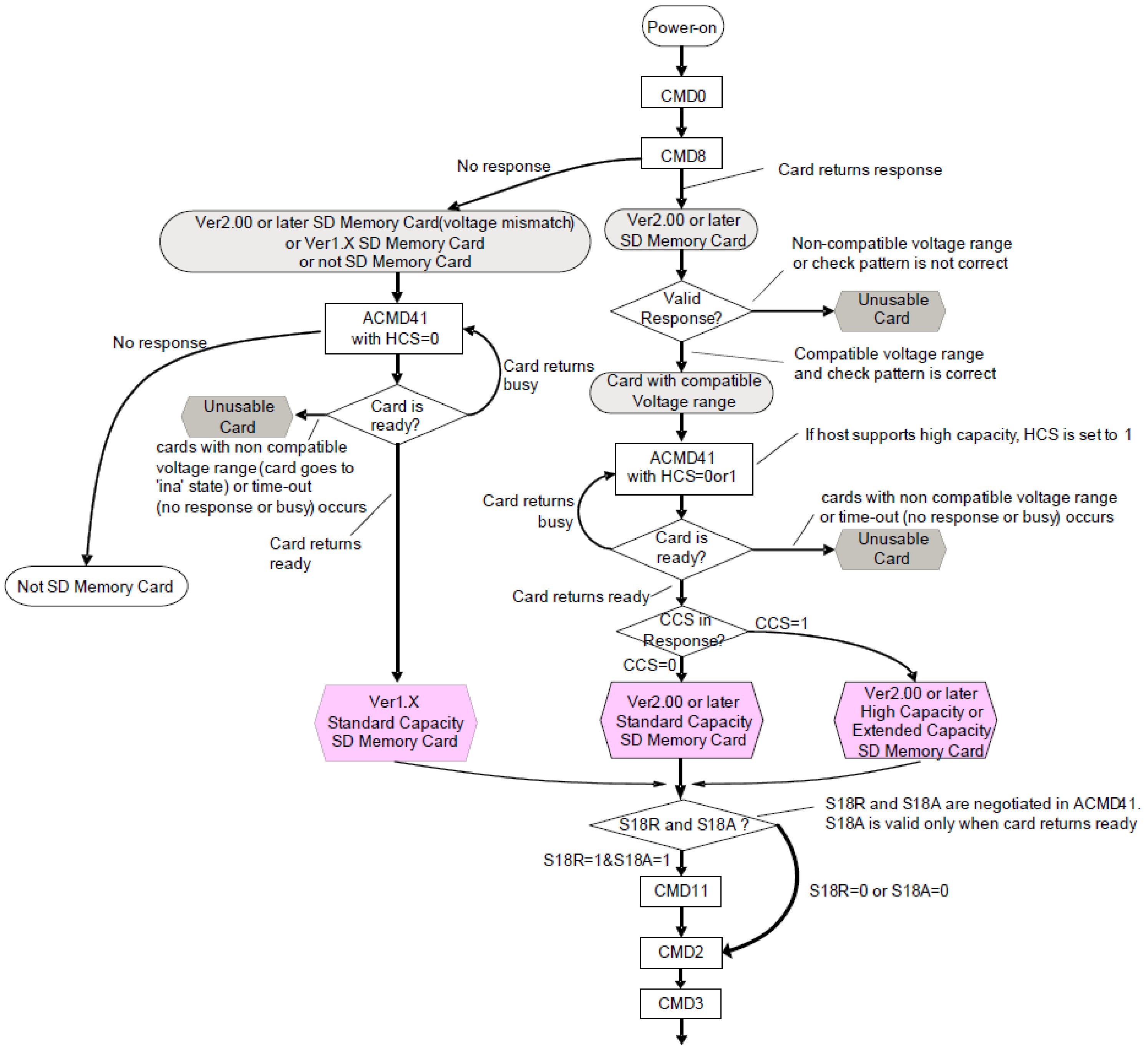
Please reference the UXE/VXE user guide and command reference manual for more detailed information regarding the memTran utility which translates between raw format and other file formats. Referencing the following sections, among others, may be helpful: “Compiling and Running Designs with Memories,” “Using Memory Streaming,” and “Translating Memory Format Using the memTran Utility.”

In the SD card model with the exception of core memory array “mem”, which is **NOT** 8 bits, all other arrays are per device specification.

12. Tuning Sequence

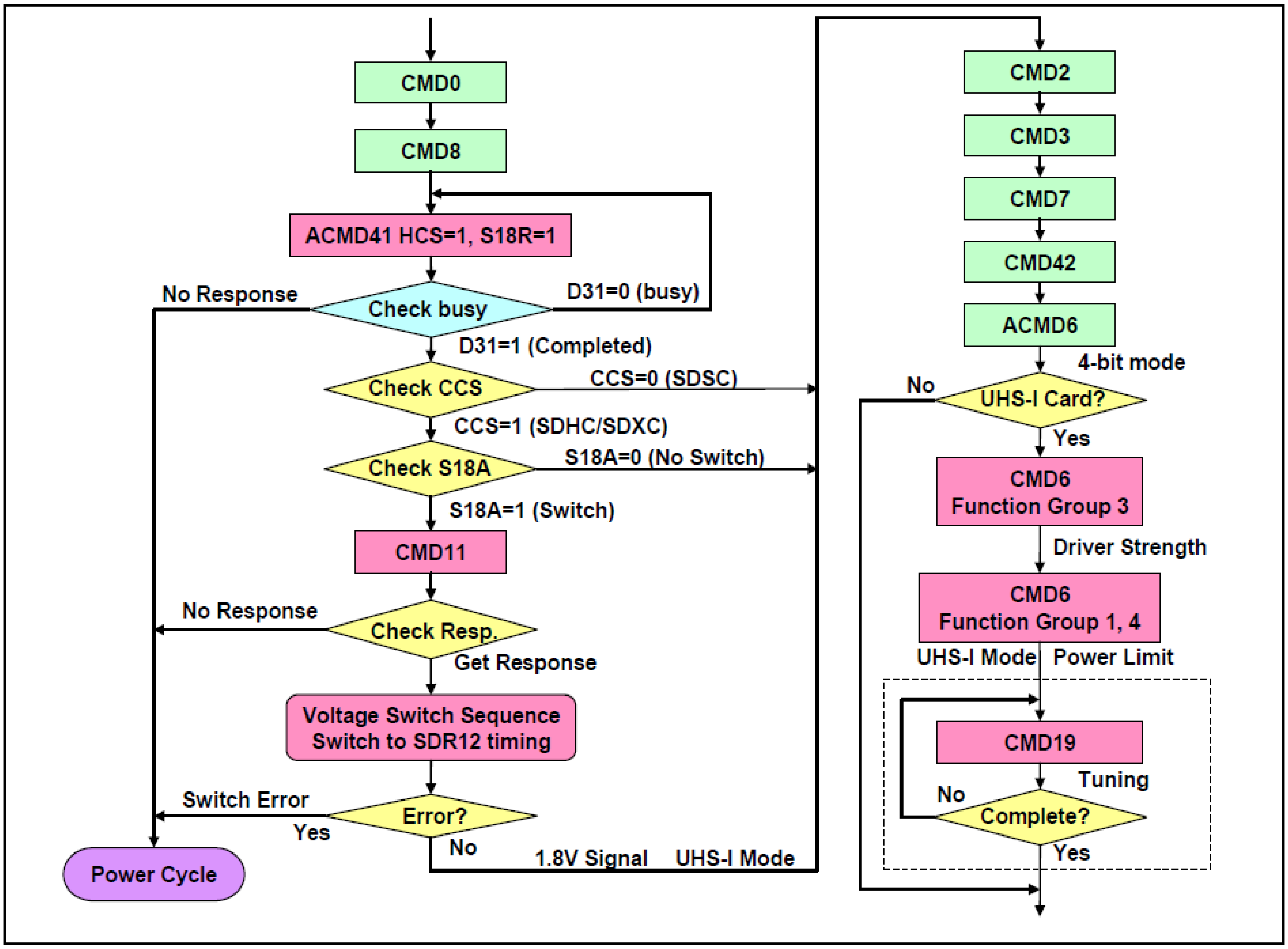
The tuning sequence pattern is stored in the Palladium SD card model in the array tune_pattern. This array must be loaded using the Palladium memory command with the provided tune.hex memory initialization file before CMD19 is issued.

13. Initialization Sequence


Power-on
CMD0
CMD8
No response
Card returns response
Ver2.00 or later SD Memory Card(voltage mismatch) or Ver1.X SD Memory Card or not SD Memory Card
Ver2.00 or later SD Memory Card
Non-compatible voltage range or check pattern is not correct
Valid Response?
Unusable Card
Compatible voltage range and check pattern is correct
Card with compatible Voltage range
ACMD41 with HCS=0
No response
Card returns busy
Card is ready?
Unusable Card
cards with non compatible voltage range(card goes to 'ina' state) or time-out (no response or busy) occurs
Card returns ready
Not SD Memory Card
If host supports high capacity, HCS is set to 1
ACMD41 with HCS=0or1
Card returns busy
cards with non compatible voltage range or time-out (no response or busy) occurs
Card is ready?
Unusable Card
Card returns ready
CCS in Response?
CCS=1
CCS=0
Ver1.X Standard Capacity SD Memory Card
Ver2.00 or later Standard Capacity SD Memory Card
Ver2.00 or later High Capacity or Extended Capacity SD Memory Card
S18R and S18A ?
S18R and S18A are negotiated in ACMD41. S18A is valid only when card returns ready
S18R=1&S18A=1
CMD11
S18R=0 or S18A=0
CMD2
CMD3

13.1. UHS-I initialization

If the card is UHS-I, the initialization sequence shall be as follow:



When signaling level is 3.3V, host repeats to issue ACMD41 with HCS=1 and S18R=1 until the response indicates ready. The argument (HCS and S18R) of the first ACMD41 is effective but the all following ACMD41 should be issued with the same argument. If Bit 31 indicates ready, host needs to check CCS and S18A. The card indicates S18A=0, which means that voltage switch is not allowed and the host needs to use current signaling level. S18A=1 means that voltage switch is allowed and host issues CMD11 to invoke voltage switch sequence. By receiving CMD11, the card returns R1 response and start voltage switch sequence. No response of CMD11 means that S18A was 0 and therefore host should not have sent CMD11. Completion of voltage switch sequence is checked by high level of DAT[3:0]. Any bit of DAT[3:0] can be checked depends on ability of the host.

14. Card Detection

Customers can force the I/O “CARD_INSERT” to simulate the situation that card is inserted or removed.

14.1. CARD_INSERT = 0

“CARD_INSERT” can be set to 0 when the process is running, that means the card is removed. Thus card will give no reaction to any command and DAT3 will be low. Customers can issue ACMD42 with arg[0]=1, if card is removed DAT3 will be low.

14.2. CARD_INSERT = 1

Normally “CARD_INSERT” is set to 1. After power up if the card is inserted, DAT3 will be automatically connected to the pull-up resistor and DAT3 will be high at the very beginning, customers MUST issue ACMD42 with arg[0]=0 to do the disconnection before regular data transfer. Also customers can re-issue ACMD42 with arg[0]=1 to connect DAT3 to the resistor again to detect whether the card is still inserted or not.
If the current state is “card is removed”, Forcing “CARD_INSERT” to 1 is treated as a reset in this model.

15. Accessing extension registers

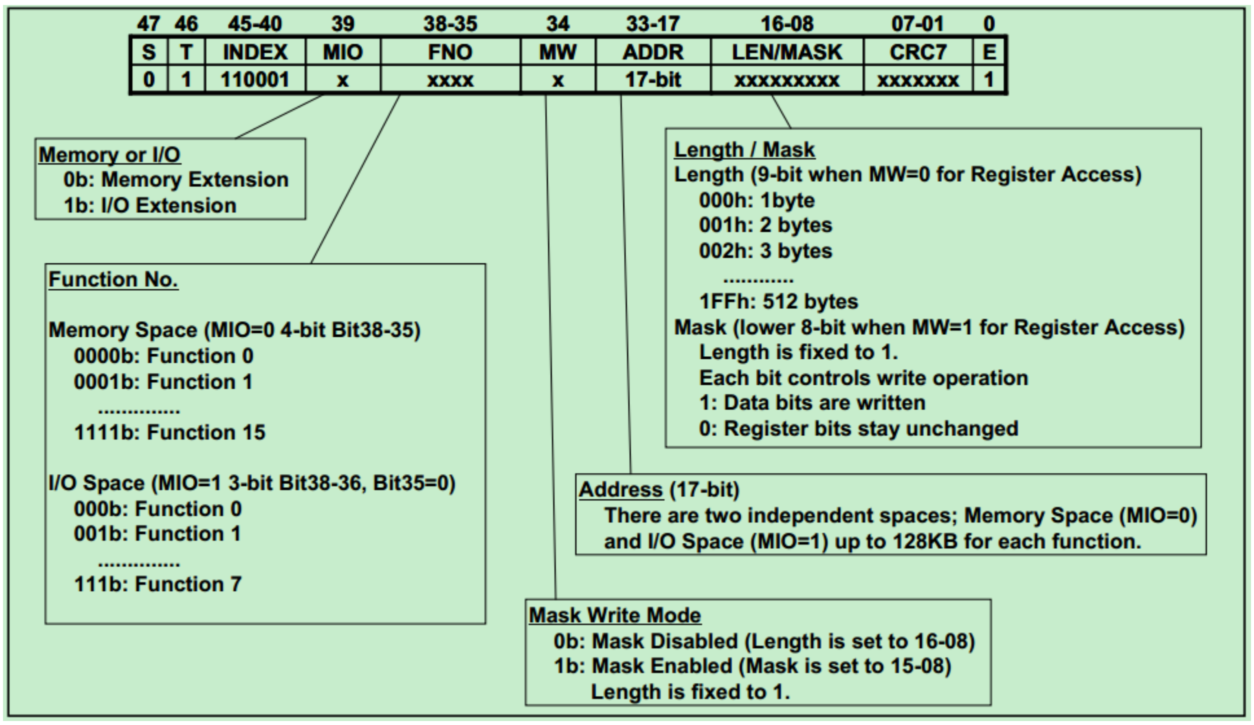
15.1. Extension registers

1) **ext_reg_mem**: this is extension memory space which is a 8 bits width array. The depth of it is defined by parameter “EXT_REG_MEM_ADDR_WIDTH” whose default value is 9+8+4 which means 512bytes x 256pages x 16functions.
2) **ext_reg_io**: this is extension IO space which is a 8 bits width array. The depth of it is defined by parameter “EXT_REG_IO_ADDR_WIDTH” whose default value is 9+8+3 which means 512bytes x 256pages x 8functions.
3) **ext_reg_mem_func_dev_x**: these are function device space for accessing extension memory. They are 8bit width, depth is 512bytes x 16
4) **ext_reg_io_func_dev_x**: these are function device space for accessing extension IO. They are 8bit width, depth is 512bytes x 16

all the arrays above can be preloaded before running emulation in PD.

15.2. Access registers

Use CMD49 as an example (access extension registers, single block write). The figure below shows the command format.


47 46 45-40 39 38-35 34 33-17 16-08 07-01 0
S T INDEX MIO FNO MW ADDR LEN/MASK CRC7 E
0 1 110001 x xxxx x 17-bit xxxxxxxxx xxxxxxx 1
Memory or I/O
0b: Memory Extension
1b: I/O Extension
Function No.
Memory Space (MIO=0 4-bit Bit38-35)
0000b: Function 0
0001b: Function 1
....
1111b: Function 15
I/O Space (MIO=1 3-bit Bit38-36, Bit35=0)
000b: Function 0
001b: Function 1
....
111b: Function 7
Length / Mask
Length (9-bit when MW=0 for Register Access)
000h: 1byte
001h: 2 bytes
002h: 3 bytes
....
1FFh: 512 bytes
Mask (lower 8-bit when MW=1 for Register Access)
Length is fixed to 1.
Each bit controls write operation
1: Data bits are written
0: Register bits stay unchanged
Address (17-bit)
There are two independent spaces; Memory Space (MIO=0) and I/O Space (MIO=1) up to 128KB for each function.
Mask Write Mode
0b: Mask Disabled (Length is set to 16-08)
1b: Mask Enabled (Mask is set to 15-08)
Length is fixed to 1.

Normally function extension commands will acess ext_reg_mem or ext_reg_io, but when address field in the command is a data port address, it will access function device space. These special data ports address of each function device are defined by parameters “DATA_PORT_ADDR_MEM_FUNC_X” and “DATA_PORT_ADDR_IO_FUNC_X”. the mechanism of accessing ext_reg_mem(io) and function device space is shown below.

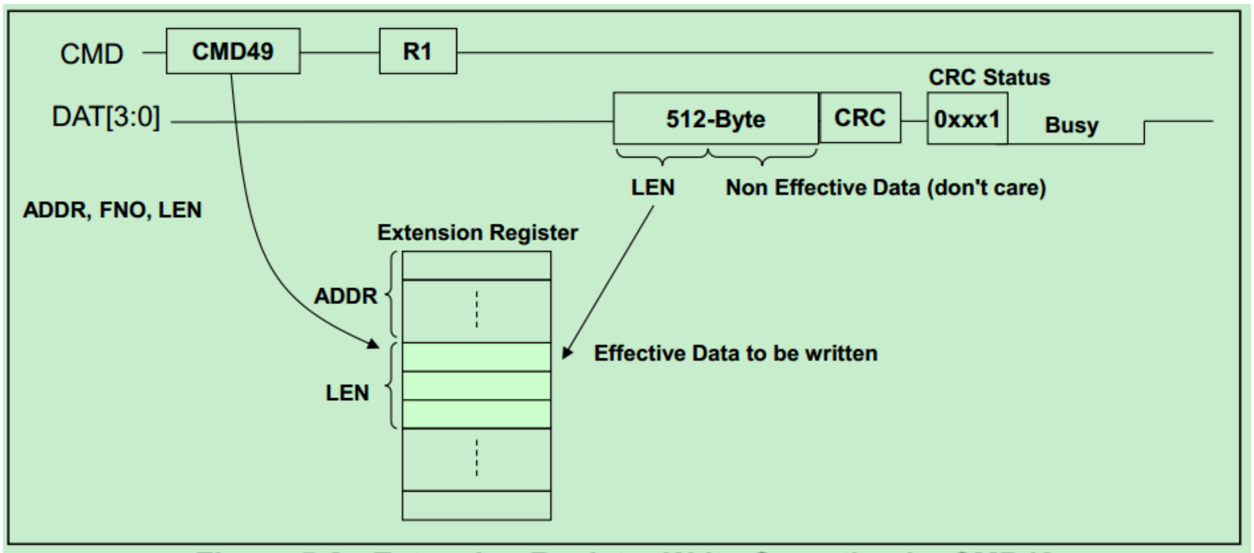


Figure 5-8 : Extension Register Write Operation by CMD49

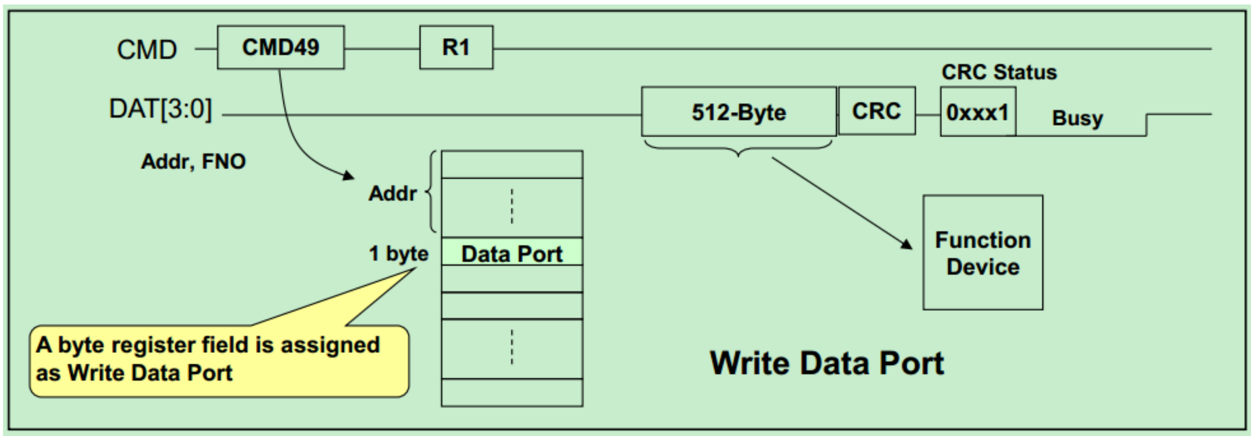


Figure 5-9 : Data Port Write Operation by CMD49

16. Special Notes

1) DDR50 mode can be selected only when the bus width is 4 bits.
2) Write protection features can only be supported in version 1.0 device. Ver2.0 and later version devices do not support write protection features.
3) Only following commands are valid when the card is locked: CMD0, CMD2, CMD3, CMD4, CMD7, CMD8, CMD9, CMD10, CMD11, CMD12, CMD13, CMD15, CMD16, CMD42; ACMD41, ACMD42.
4) All of the controllers shall support pullup on CMD and DAT bus. There is a macro for the controllers which DO NOT support pullup --- ***MMP_SDCARD_HOST_NO_PULLUP***. If there is no pullup on CMD and DAT bus in user's test environment (out of SD model), please define the macro.

17. Revision History

The following table shows the revision history for this document

Date	Version	Revision
Jan 2014	1.0	Initial release
Feb 2014	1.1	Added Card Insert function
Apr 2014	1.2	Add description of accessing extension registers
July 2014	1.3	Repaired doc title property.
September 2014	1.4	Remove version from UG file name. Update UXE / IXE documentation reference titles. Added missing CMD11 to supported table. Removed Beta watermark. Update tables describing user adjustable parameters and localparams to align with RTL changes. Add two localparams that were introduced in v1.3 voltage switching fix. Added a couple missing items in model pin description table.
March 2015	1.5	Update related publications list. Update feature list. Update Configurations section and compile info.
March 2015	1.6	Change default value in localparam table of R1b_READY_COUNT from 64 to 4
April 2015	1.7	Add some generic SD Card model configurations to UG and catalogue
July	1.8	Added description for pullups and ***HOST_NO_PULLUP*** macro Added pointer to large memory user rguide.
July 2015	1.9	Update Cadence naming on front page
August 2015	1.91	Correct 512MB/1GB/2GB card configuration description from SDHC to correct description SDSC (CCR# 1459829)
September 2015	2.0	Modify compile notes to reflect *.vp as sole model format and add note about synthesis options.
December 2015	2.1	Add introduction of address mapping when accessing memory arrays
January 2016	2.2	Update for Palladium-Z1 and VXE. Expand “address Mapping to access Memory arrays” section.
July 2016	2.3	Remove hyphen in Palladium naming
September 2016	2.4	Fixed path in synthesis example
October 2017	2.5	Update macro name to MMP_SDCARD_HOST_NO_PULLUP
January 2018	2.6	Modify header and footer
July 2018	2.7	Update for new utility library