Application Note, AN0501-1.00, April 2005

MMC*plus*™

MMCA 4.1

MultiMediaCard Association

Edition 2005-04-12
Published by MultiMediaCard Association,
P.O. Box 303
Sunol, CA 94586
USA,

Information

For further information on specification, membership and compliance please contact MultiMediaCard Association under info@mmca.org or the above mail address.

1 Comparison of MultiMediaCard Specification Version 4.x vs. 3.31

1.1 Bus Intialization

The initialization procedure of MMC*plus* is equivalent to that of cards by standard version 3.31. The major difference is that any dedicated MMC*plus* host application is restriced to have only a single card slot (in order to cope with the increased frequency requirements).

In legacy hosts the MMC*plus* is expected to operate in its single data line mode. Thus its high speed capabilities will exclusively be available to dedicated applications.

1.1.1 Power-up sequence

1. Apply power to the bus, communication voltage range (2.0V to 3.6V)
2. Set clock to 400kHz, or less
3. Wait for 1msec, then wait for 74 more clock cycles
4. Send CMD0 to reset the bus, keep DAT3 high during this step
5. Send CMD1, with the intended voltage range in the argument (either High Voltage or Dual Voltage)
6. Receive R3 (Provides the content of the OCR register)
7. Repeat steps 5 and 6 as long as the OCR busy bit is ‘0’
8. From the R3 response argument the host can learn if the card is a high voltage or dual voltage card.

R3 Response

0x80FF8000 = High Voltage Card

0x80FF8080 = Dual Voltage Card

9. If R3 returned some other value, the card is not compliant (since it should have put itself into *inactive* state, due to voltage incompatibility, and not respond); in such a case the host must power down the bus and start error recovery procedure. (the definition of error recovery procedure is host dependent)

1.1.2 Register Assignment

The MMC*plus* specification introduces the Extended CSD Register (512 Bytes). It also declares that a high speed host should have only one card slot. This implies that in multi-card systems high-speed cards should be operated only in their legacy mode (i.e. with 1 data pin and max. 20MHz) in order not to disturb other legacy cards active there.

Utilizing the common CSD register the SPEC_VERS field should be checked. If it denotes a System specification Version 4.0 or higher, this indicates a high speed card and the support of the SEND_EXT_CSD command (CMD8). This allows to verify new card properties and to select distinct operation modes. The host can change the active command set by issuing the SWITCH command (CMD6).

Segment	**Information**	**Example**
Properties	Card Capability	Card Type
Mode	Operation Mode	High Speed Interface Timing

Note: Regardless of the type of card, the maximum clock frequency is defined in the TRAN_SPEED field.

Verification Sequence (single slot host):

1. Send CMD2 to ask all cards for sending their individual Card-Identification numbers
2. Receive R2 and get the individual card’s Card-Identification
3. Select a card by sending CMD3 with a chosen RCA [value greater than 1]
4. Send CMD9 asking for the Card-Specific Data of the selected card
5. Receive R2, and determine the Card-Specific Data from it.
6. If specification version is 4.0 or higher, get EXT_CSD information and/or switch command set as required.
7. Adjust the host parameters, if necessary, according to the CSD/EXT_CSD information.

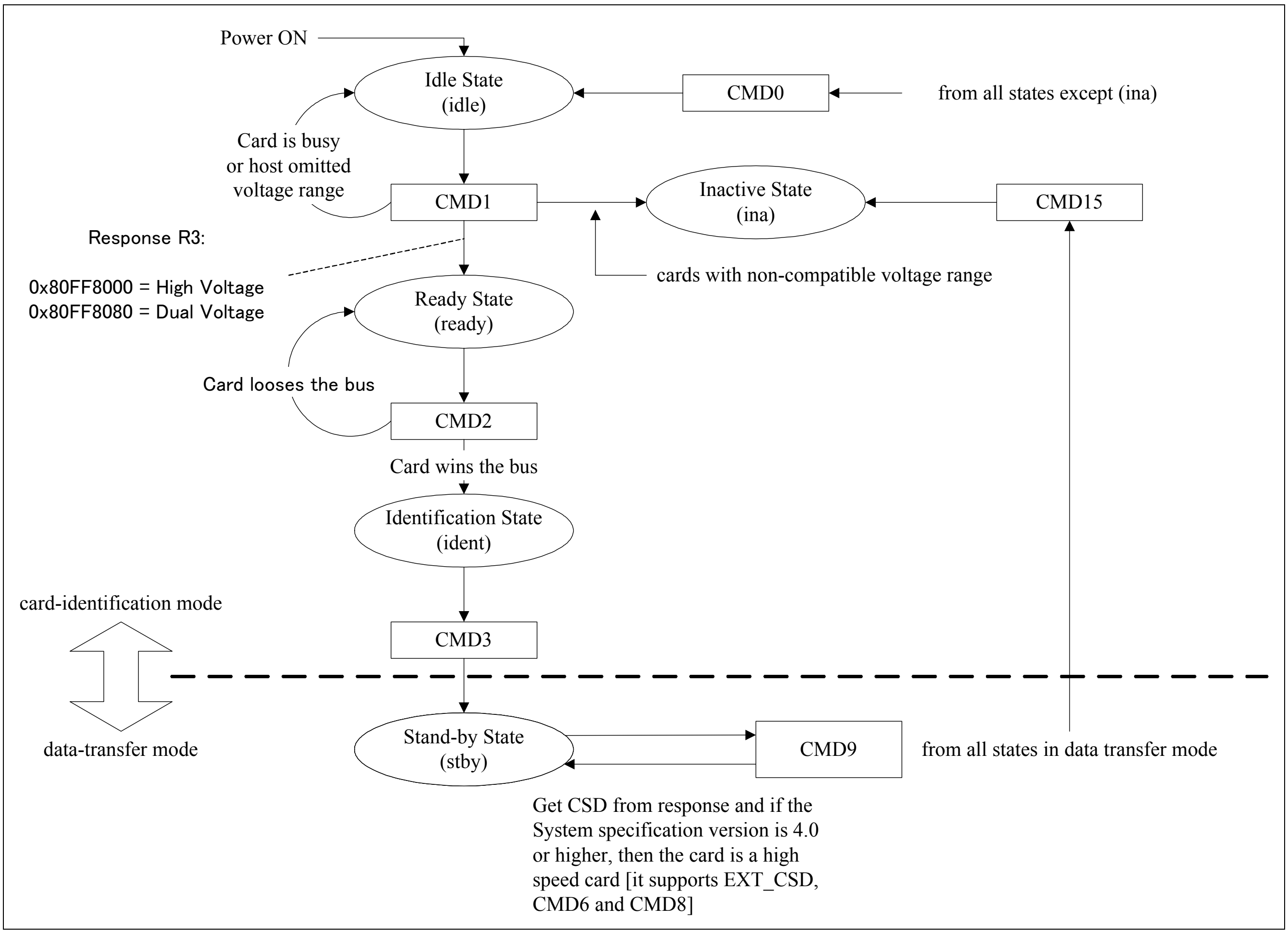
Power ON
Idle State
(idle)
CMD0
from all states except (ina)
Card is busy
or host omitted
voltage range
CMD1
Inactive State
(ina)
CMD15
Response R3:
0x80FF8000 = High Voltage
0x80FF8080 = Dual Voltage
cards with non-compatible voltage range
Ready State
(ready)
Card looses the bus
CMD2
Card wins the bus
Identification State
(ident)
card-identification mode
CMD3
data-transfer mode
Stand-by State
(stby)
CMD9
from all states in data transfer mode
Get CSD from response and if the
System specification version is 4.0
or higher, then the card is a high
speed card [it supports EXT_CSD,
CMD6 and CMD8]

Figure 1 State Diagram - Card Identification Mode

1.2 Clock Frequency

The MultiMediaCard has historically been assigned a clock frequency range of 0 to 20MHz. The initialization has been specified to be maximum 400kHz in order to cope with the initial bus state.

With the introduction of MMC*plus* the available frequencies have been extended to 26MHz and 52MHz besides the legacy value range.

1.2.1 Initial Frequency

The MMC*plus* frequency during initialization was not changed to ensure backwards compatibility. Doing so even a high speed host is potentially able to deal with legacy cards.

1.2.2 Operating Frequency

Besides the legacy operating frequency range the MMC*plus* may support values of 26MHz and 52MHz. For setting the following flow may be used.

1. From EXT_CSD the host gets the power and speed class of the card.
2. In case it is enabled, the high speed interface timing can be used after using CMD6 to write 0x1 to the HS_Timing Byte of the EXT_CSD [example: use CMD6 with argument 0x03B90100]. After the command the cards is busy to configure the high speed mode. Once busy is released the card is ready for high speed timing.
3. Change the clock to the chosen frequency.

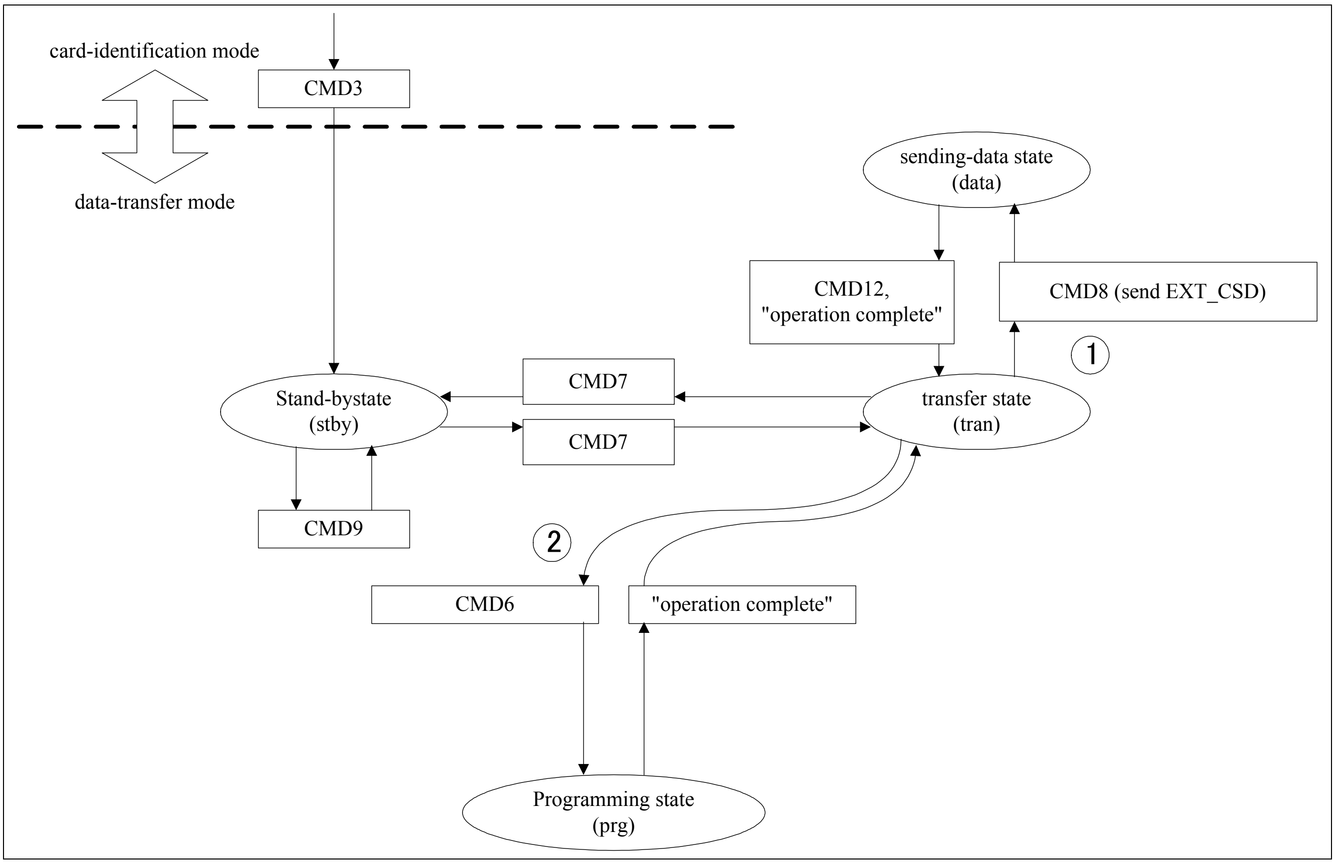

Figure 2 State Diagram - Switch to High Speed Mode

1.3 Power Class Selection

The host may change the power class of the card.

1. The default card power class is class 0 (minimum current consumption class for the card type, either High or Dual Voltage card)
2. Host sends CMD8, and determines if it will allow the card to use higher power class.
3. If a power class change is needed, send CMD6 to write POWER_CLASS in EXT_CSD register.

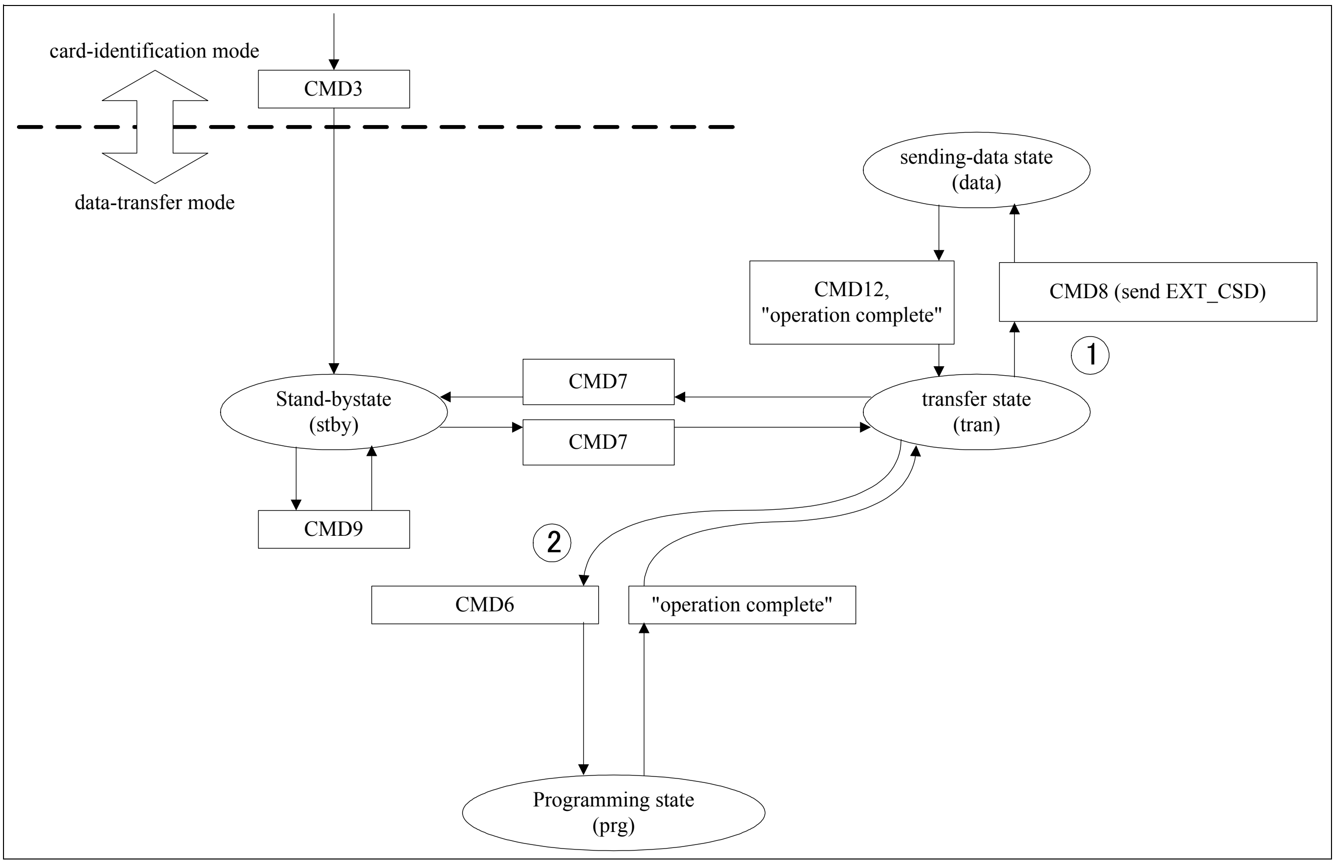

Figure 3 State Diagram - Change Power Class

1.4 Data Bus

If the cards power class allows the host to work on a wider bus, with in the host power budget, the following steps could be possible:

1. Host sends the bus test data pattern to a card
2. Host reads the reversed bus testing data pattern from a card
3. Host determines the bus line connection

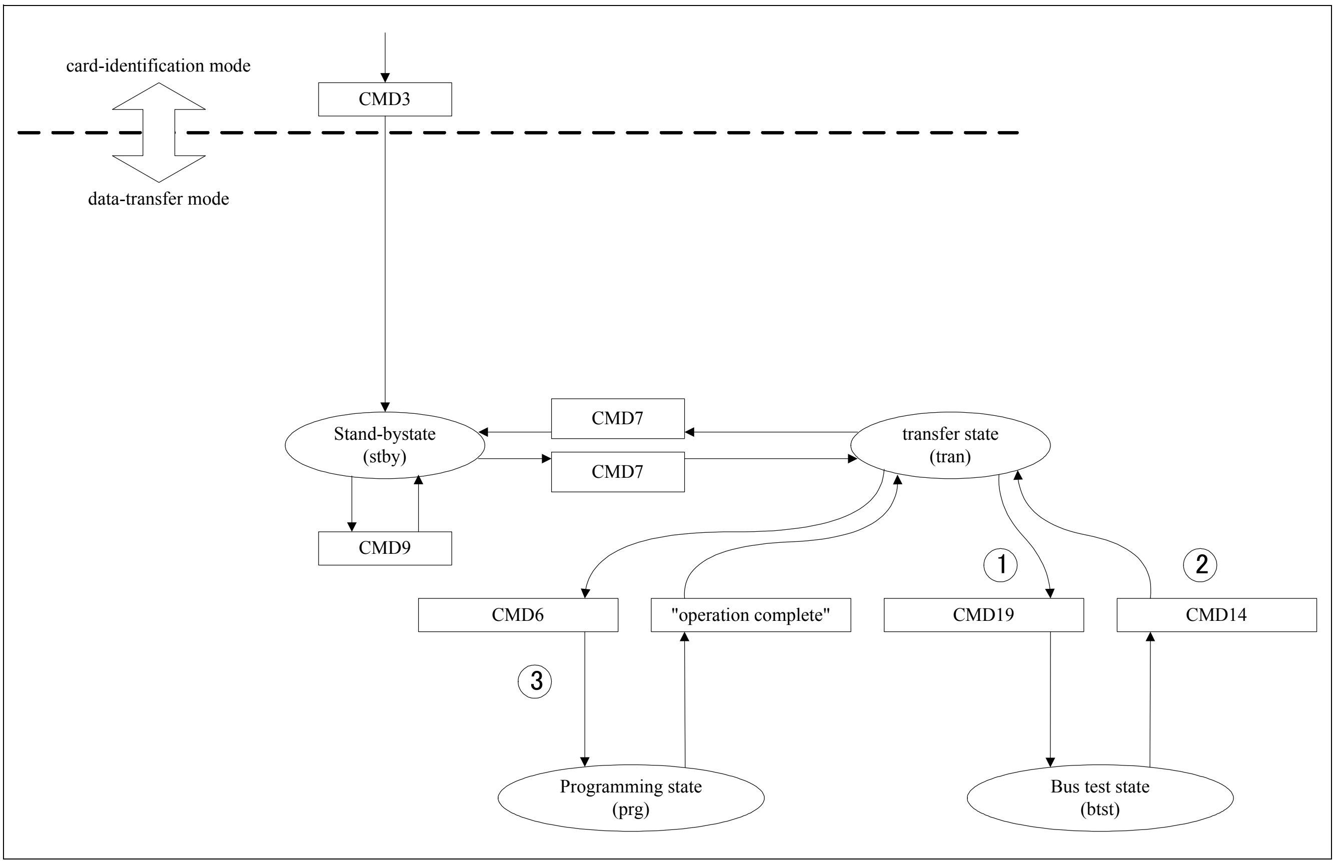

Figure 4 State Diagram - Change Data Bus Width

1.4.1 Change Bus Width Procedure - Signal Flow

1. Card is initialized just like a legacy MultiMediaCard
2. Card is directed to TransferState (by CMD7 with RCA)
3. Host sends a data pattern with a new command (CMD19), card sends a response and latches received data pattern internally
4. If the host sends another new command (CMD14), the card would send the reverse pattern of received data
5. Host determines the bus line connection or the number of the card I/O through CMD6
6. The bus is ready to exchange data using the new width configuration

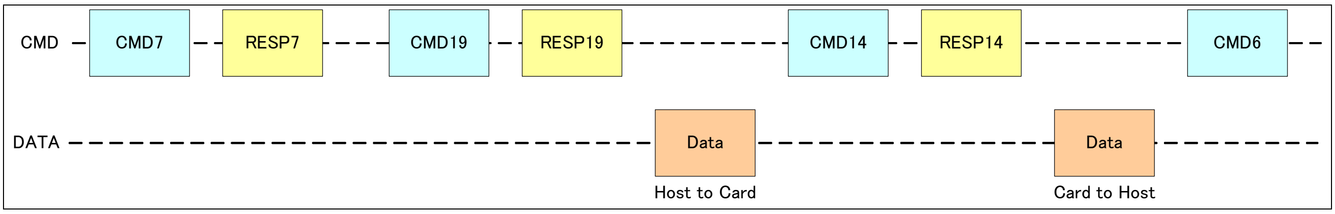

Figure 5 Signal Flow Summary - Change Data Bus Width

1.4.2 Change Bus Width Procedure - Sample Program

```
//////////////////////////////////////////////////////////////////////////////////
//     Test and switch data bus width for MMC 4.0 host
//////////////////////////////////////////////////////////////////////////////////

       SET8BIT;                                   // Set the host to 8-bit mode
       if (BUSTEST_W() || BUSTEST_R())            // If 8-bit mode access fails
              {
              SET4BIT;                            // Set the host to 4-bit mode

              if (BUSTEST_W() || BUSTEST_R())     // If 4-bit mode access failed
                     {
                     SET1BIT;                     // Set the host to 1-bit mode
                     if (BUSTEST_W() || BUSTEST_R())
                     SwitchBusWidth(0);           // Set the MMC bus to
                                                  // 1-bit mode, low speed
                     else
                            {
                            SwitchBusWidth(0);    // Set the MMC bus to 1-bit mode
                            switch_highspeed();   // Turn on high speed support
                            }
                     }
              else
                     {
                     SwitchBusWidth(1);           // Set the MMC bus to 4-bit mode
                     switch_highspeed();          // Turn on high speed support
                    }
                }
       else
              {
              SwitchBusWidth(2);                  // Set the MMC bus to 8-bit mode
              switch_highspeed();                 // Turn on high speed support
              }

//////////////////////////////////////////////////////////////////////////////////



bit BUSTEST_W()                                   // Send the bus test data
                                                  // to MMC card

{
       if (MMC_IN_8BIT_MODE)
              {
              XB_BUF0[0]=0x55;
              XB_BUF0[1]=0xaa;
              }
       else
       if (MMC_IN_4BIT_MODE) XB_BUF0[0]=0x5a;
       else  XB_BUF0[0]=0x40;                     // MMC in 1-bit mode

       MMCarg0=0x00;                              // Set up the arguments
                                                  // for MMC CMD19

       MMCarg1=19;
       MMCarg2=0x00;
       MMCarg3=0x00;
```

```
        MMCarg4=0x00;
        MMCarg5=0x00;

        if (!MMC_CMD(1) && !NO_RSP)                     // Issue CMD19
                {
                PresetTimeOut(240);
                Send_Data_to_MMC(XB_BUF);
                while (DAT0_BUSY && !TimeOut);          // Wait for MMC ready
                return 0;                               // Success
                }
        else return 1;                                  // Fail
}

bit BUSTEST_R()                                         // Read the reverse bus data
                                                        // from MMC card
{
        MMCarg0=0x00;                                   // Set up the argument
                                                        // for MMC CMD14
        MMCarg1=14;
        MMCarg2=0x00;
        MMCarg3=0x00;
        MMCarg4=0x00;
        MMCarg5=0x00;
        if (!MMC_CMD(1) && !NO_RSP)                     // Issue CMD14
                {

                Read_Data_from_MMC(XB_BUF);
                PresetTimeOut(240);
                while (DAT0_BUSY && !TimeOut);          // Wait for MMC Ready
                if (!TimeOut)
                        {
                if (MMC_IN_8BIT_MODE && (XB_BUF0[0]==0xaa)||(XB_BUF0[1]==0x55))
                                return 0;
                        else if (MMC_IN_4BIT_MODE && (XB_BUF0[0]==0xa5))
                                return 0;
                        else if ((XB_BUF0[0] & 0xc0)==0x80)
                                return 0;
                        else
                                return 1;

                        }
                else
                        return 1;
                }
        else
                return 1;

}

void switch_highspeed()
{
        MMCarg0=RSPR1B;                                 // Set up the arguments
                                                        // for MMC CMD6
        MMCarg1=6;
        MMCarg2=0x03;
```

```
       MMCarg3=185;
       MMCarg4=1;
       MMCarg5=0x00;
       MMCCMD(1);                                // Issue CMD6
}

void SwitchBusWidth(BYTE mm)
{

       if (mm==2)      SET8BIT;                   // Set the host to 8-bit mode
       else if (mm==1) SET4BIT;                   // Set the host to 4-bit mode
       else            SET1BIT;                   // Set the host to 1-bit mode

       MMCarg0=RSPR1B;                            // Set up the arguments
                                                  // for MMC CMD6
       MMCarg1=6;
       MMCarg2=0x03;
       MMCarg3=183;
       MMCarg4=mm;
       MMCarg5=0x00;
       MMCCMD(1);                                // Issue CMD6
}
```

1.4.3 Bus Testing Pattern

The card ignores all but the first two bits of the data pattern.
The host ignores all but the first two bits of the reverse data pattern.

Table 1 Data Bus Width - Testing Pattern: 1 bit

Data Line	Data pattern sent by Host	Reserved pattern sent by card	Notes
DATA0	0 10xxxxxxxxxxxx [CRC16] 1	0 01000000 [CRC16] 1	Start bit defines begin of pattern
DATA1		0 00000000 [CRC16] 1	No data pattern sent
DATA2		0 00000000 [CRC16] 1	No data pattern sent
DATA3		0 00000000 [CRC16] 1	No data pattern sent
DATA4		0 00000000 [CRC16] 1	No data pattern sent
DATA5		0 00000000 [CRC16] 1	No data pattern sent
DATA6		0 00000000 [CRC16] 1	No data pattern sent
DATA7		0 00000000 [CRC16] 1	No data pattern sent

Table 2 Data Bus Width - Testing Pattern: 4 bits

Data Line	Data pattern sent by Host	Reserved pattern sent by card	Notes
DATA0	0 10xxxxxxxxxxxx [CRC16] 1	0 01000000 [CRC16] 1	Start bit defines begin of pattern
DATA1	0 01xxxxxxxxxxxx [CRC16] 1	0 10000000 [CRC16] 1	
DATA2	0 10xxxxxxxxxxxx [CRC16] 1	0 01000000 [CRC16] 1	

Table 2 Data Bus Width - Testing Pattern: 4 bits (Continued)

Data Line	Data pattern sent by Host	Reserved pattern sent by card	Notes
DATA3	0 01xxxxxxxxxxxx [CRC16] 1	0 10000000 [CRC16] 1	
DATA4		0 00000000 [CRC16] 1	No data pattern sent
DATA5		0 00000000 [CRC16] 1	No data pattern sent
DATA6		0 00000000 [CRC16] 1	No data pattern sent
DATA7		0 00000000 [CRC16] 1	No data pattern sent

Table 3 Data Bus Width - Testing Pattern: 8bits

Data Line	Data pattern sent by Host	Reserved pattern sent by card	Notes
DATA0	0 10xxxxxxxxxxxx [CRC16] 1	0 01000000 [CRC16] 1	Start bit defines begin of pattern
DATA1	0 01xxxxxxxxxxxx [CRC16] 1	0 10000000 [CRC16] 1	
DATA2	0 10xxxxxxxxxxxx [CRC16] 1	0 01000000 [CRC16] 1	
DATA3	0 01xxxxxxxxxxxx [CRC16] 1	0 10000000 [CRC16] 1	
DATA4	0 10xxxxxxxxxxxx [CRC16] 1	0 01000000 [CRC16] 1	
DATA5	0 01xxxxxxxxxxxx [CRC16] 1	0 10000000 [CRC16] 1	
DATA6	0 10xxxxxxxxxxxx [CRC16] 1	0 01000000 [CRC16] 1	
DATA7	0 01xxxxxxxxxxxx [CRC16] 1	0 10000000 [CRC16] 1	

1.5 New Commands

Commands exclusively supported by high-speed version of MultiMediaCard are CMD6, CMD8, CMD14 and CMD19. A host has to comply to the requirements of this version in order to use them properly.

Table 4 New Commands

CMD	Abbreviation	Command Description	Argument
6	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD register.	[31:26] Set to 0 [25:24] Access [23:16] Index [15:08] Value [07:03] Set to 0 [02:00] CMD set
8	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data.	[31:00] Stuff bits
14	BUSTEST_R	The host reads the reserved bus testing data pattern from the card.	[31:00] Stuff bits
19	BUSTEST_W	The host sends the bus testing data pattern to the card.	[31:00] Stuff bits

1.6 New Register

1. Extended CSD Register (512Bytes) is introduced with HS-MMC
2. Defines further card properties (320Bytes) and selected modes (192Bytes)
3. Uses SWITCH command to change modes

Table 5 Extended CSD Register (EXT_CSD)

Name	Field	Size [Bytes]	Cell Type	EXT_CSD slice [Byte]	Value
Supported command sets	S_CMD_SET	1	R	[504]	9
Power class for 26MHz @ 3.6V	PWR_CL_26_360	1	R	[203]	-
Power class for 52MHz @ 3.6V	PWR_CL_52_360	1	R	[202]	-
Power class for 26MHz @ 1.95V	PWR_CL_26_195	1	R	[201]	-
Power class for 52MHz @ 1.95V	PWR_CL_52_195	1	R	[200]	-
Card type	CARD_TYPE	1	R	[196]	-
CSD structure version	CSD_STRUCTURE	1	R	[194]	3
Extended CSD revision	EXT_CSD_REV	1	R	[192]	0
Command set	CMD_SET	1	R/W/E	[191]	-
Command set revision	CMD_SET_REV	1	R/W/E	[189]	-
Power Class	POWER_CLASS	1	R/W/E	[187]	-
High speed interface timing	HS_TIMING	1	R/W/E	[185]	-
Bus width method	BUS_WIDTH	1	W/E	[183]	-

1.7 Basic Operation

Basic operation is SAME.

- CRC calculation is used for all bus configurations.
- x4 & x8 just added CRC for each data line separately.
- 8-bit data bus is just by additional 4 data lines wider than the 4-bit bus.

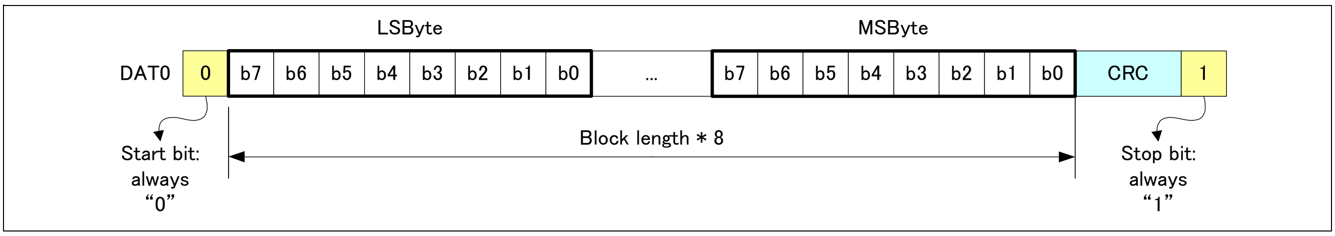

Figure 6 Signal Transfer - Data Bus Width: 1bit

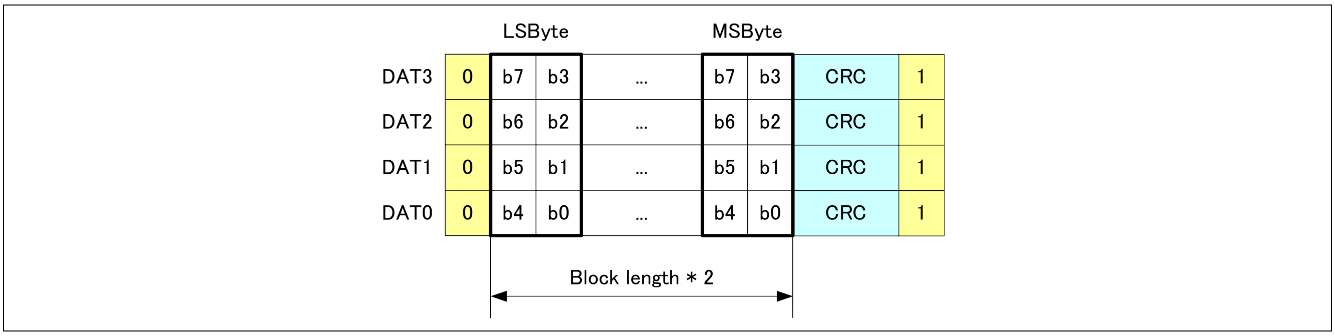
LSByte
MSByte
DAT3 0 b7 b3 ... b7 b3 CRC 1
DAT2 0 b6 b2 ... b6 b2 CRC 1
DAT1 0 b5 b1 ... b5 b1 CRC 1
DAT0 0 b4 b0 ... b4 b0 CRC 1
Block length * 2

Figure 7 Signal Transfer - Data Bus Width: 4bits

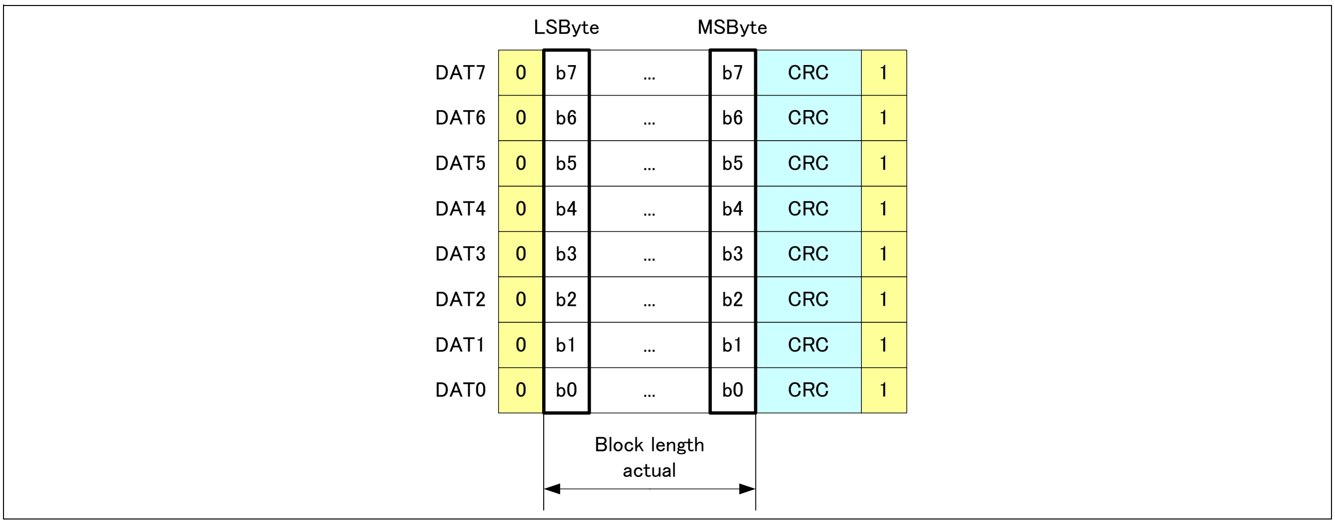

Figure 8 Signal Transfer - Data Bus Width: 8bits

1.8 Timing

High-speed MultiMediaCard timing parameters are referenced to the center of the edges of the clock signal.

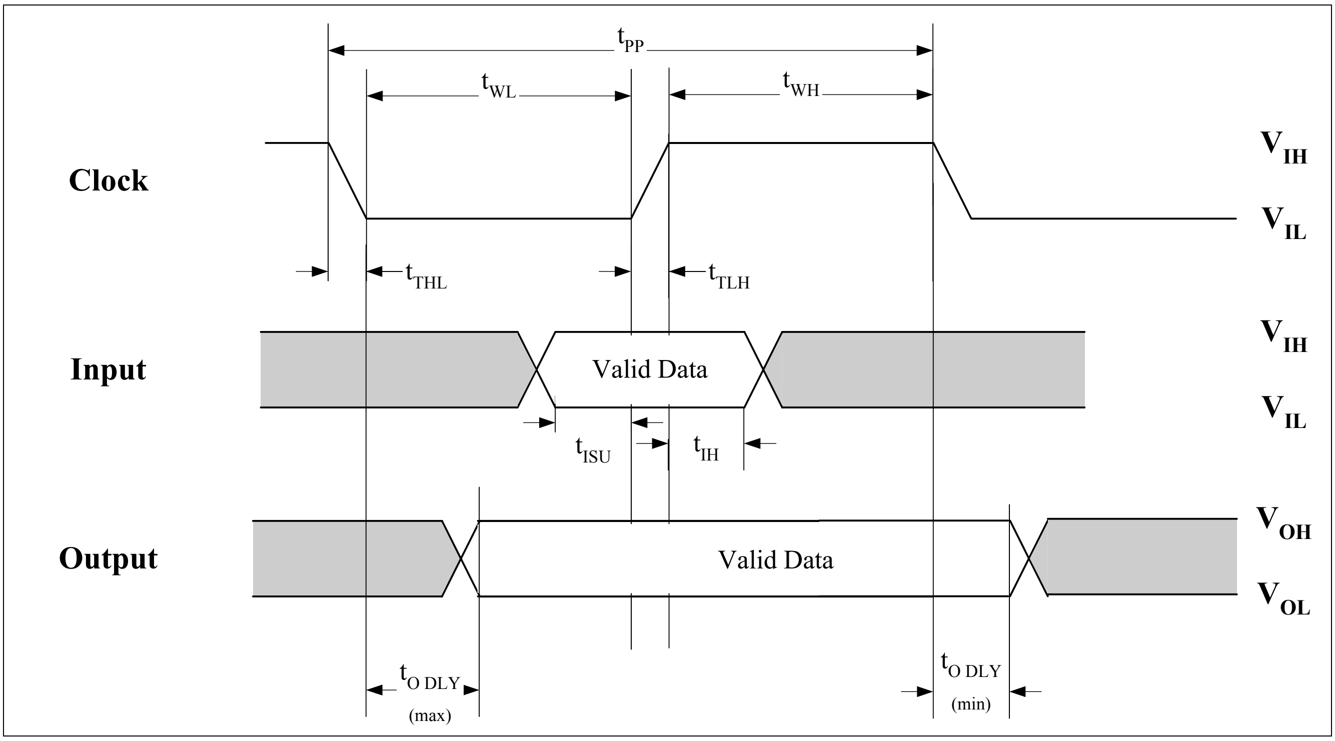

Figure 9 Timing paramters in MMCA specification version 3.31

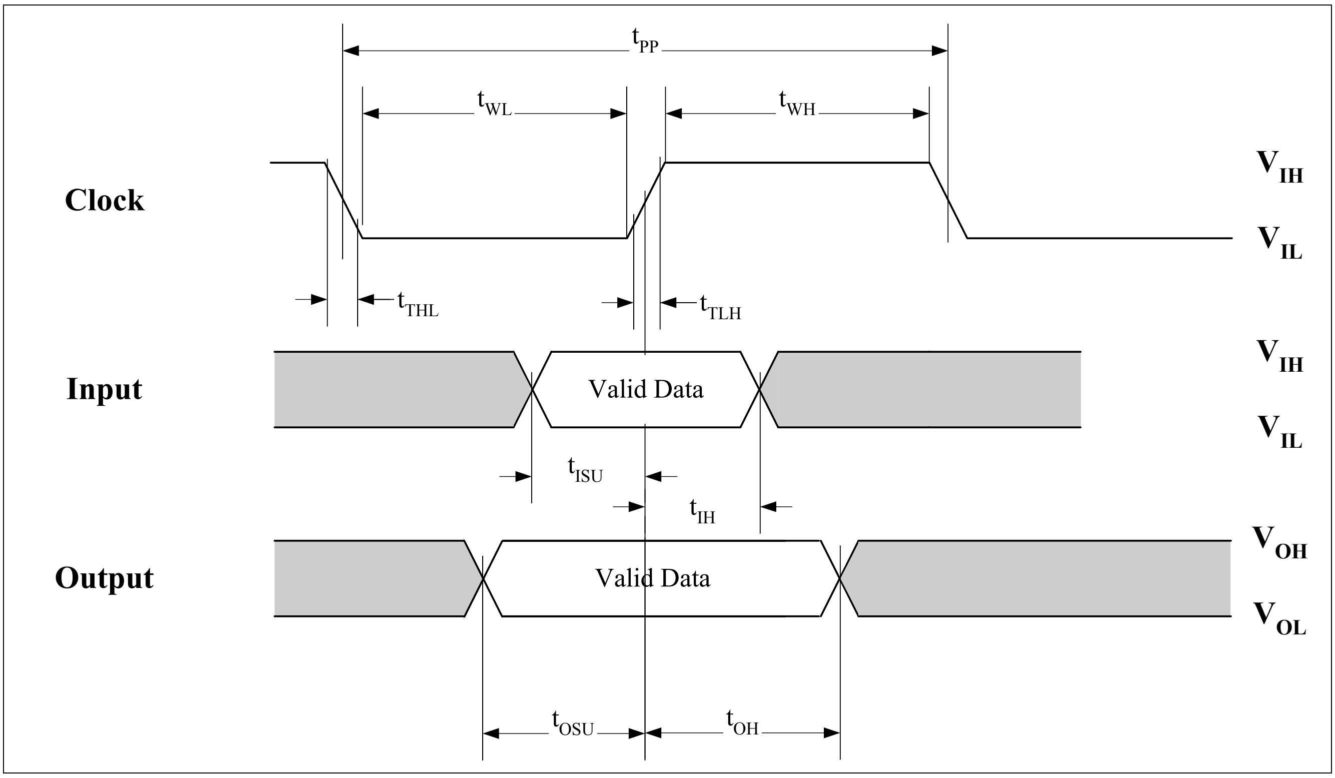

Figure 10 Timing paramters in MMCA specification version 4.0

Table 6 Timing paramters comparison MMCA Specification versions 3.31 and 4.0

Parameter		MMCA Spec ver 3.31		MMCA Spec ver 4.0	
Name	Short	Min [nsec]	Max [nsec]	Min [nsec]	Max [nsec]
Clock low time	t_{WL}	* 50 / 10		6.5	
Clock high time	t_{WH}	* 50 / 10		** 6.5	
Clock rise time	t_{TLH}		* 50 / 10		3
Clock fall time	t_{THL}		* 50 / 10		3
Input setup time	t_{ISU}	3		3	
Input hold time	tIH	3		3	
Output setup time	t_{OSU}	5		5	
Output hold time	t_{OH}	5		5	

1. **) N = 30*
2. ***) estimation - not defined in specification*

1.9 Pad-Out

High-speed MultiMediaCard pad-out is compatible to legacy MultiMediaCard and full-size SD Card. The High-speed MultiMediaCard provides two form factors (full size and reduced size). The reduced size version fits into a full-size slot with the help of a standardized passive extender

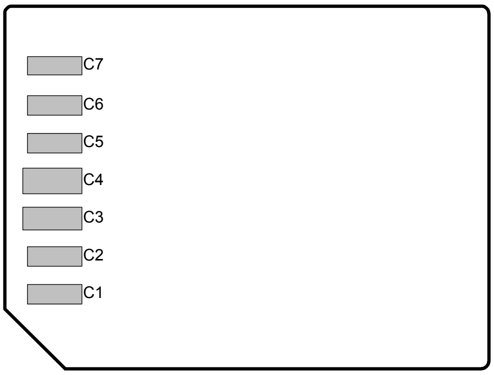

Figure 11 Pad-out of MMCA specification version 3.31

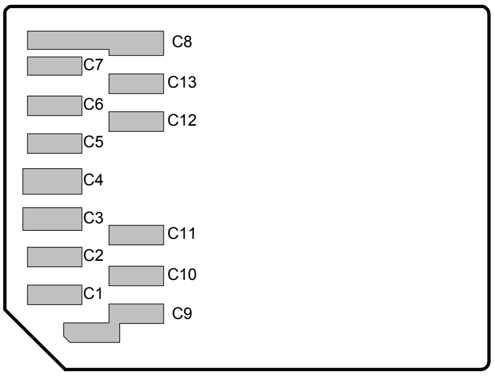

Figure 12 Pad-out of MMCA specification version 4.0 - Full Size

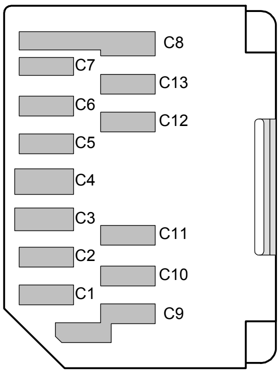

Figure 13 Pad-out scheme of MMCA specification version 4.0 - Reduced Size

Table 7 Pad-out of high-speed MultiMediaCard

Pad No.	Name	Type	Description
1	DAT3	I/O/PP	DATA
2	CMD	I/O/PP/OD	Command/Response
3	VSS1	S	Supply Voltage - Ground
4	VDD	S	Supply Voltage
5	CLK	I	Clock
6	VSS2	S	Supply Voltage - Ground
7	DAT0	I/O/PP	DATA
8	DAT1	I/O/PP	DATA
9	DAT2	I/O/PP	DATA
10	DAT4	I/O/PP	DATA
11	DAT5	I/O/PP	DATA
12	DAT6	I/O/PP	DATA
13	DAT7	I/O/PP	DATA

2 Co-existance with similar formfactor memory cards

This chapter deals mainly with the compatibilty to the SD form factor because of the evident similarity of the physical dimensions and communication protocol. For details of the SD Card form factor please refer to the according specification.

Any MultiMediaCard fits into the physical slot of a host designed to use a full size SD Card. During the course of initializing an SD Host the presence of a MultiMediaCard in the (full-size) SD Card slot can be determined. Once this is done, the host can continue to initialize the MultiMediaCard in the way described in Chapter 1.1 "Bus Intialization" on page 6 of this Application Note. Details of the MultiMediaCard initialization should be obtained from the MMCA specifciation.

The similarity of the protocols allows an easy extension of the host firmware to enable the use of the 4-data-line high speed operation mode of the MultiMediaCard.

The physical structure of dedicated miniSD and microSD card slots does not allow use of any cards in line with the MMCA standards. The same applies to the slots of any other Flash Memory Card format known today.

2.1 Initialization process (example 1)

Please find below the generalized flow leading to the routine initializing a high-speed MultimediaCard.

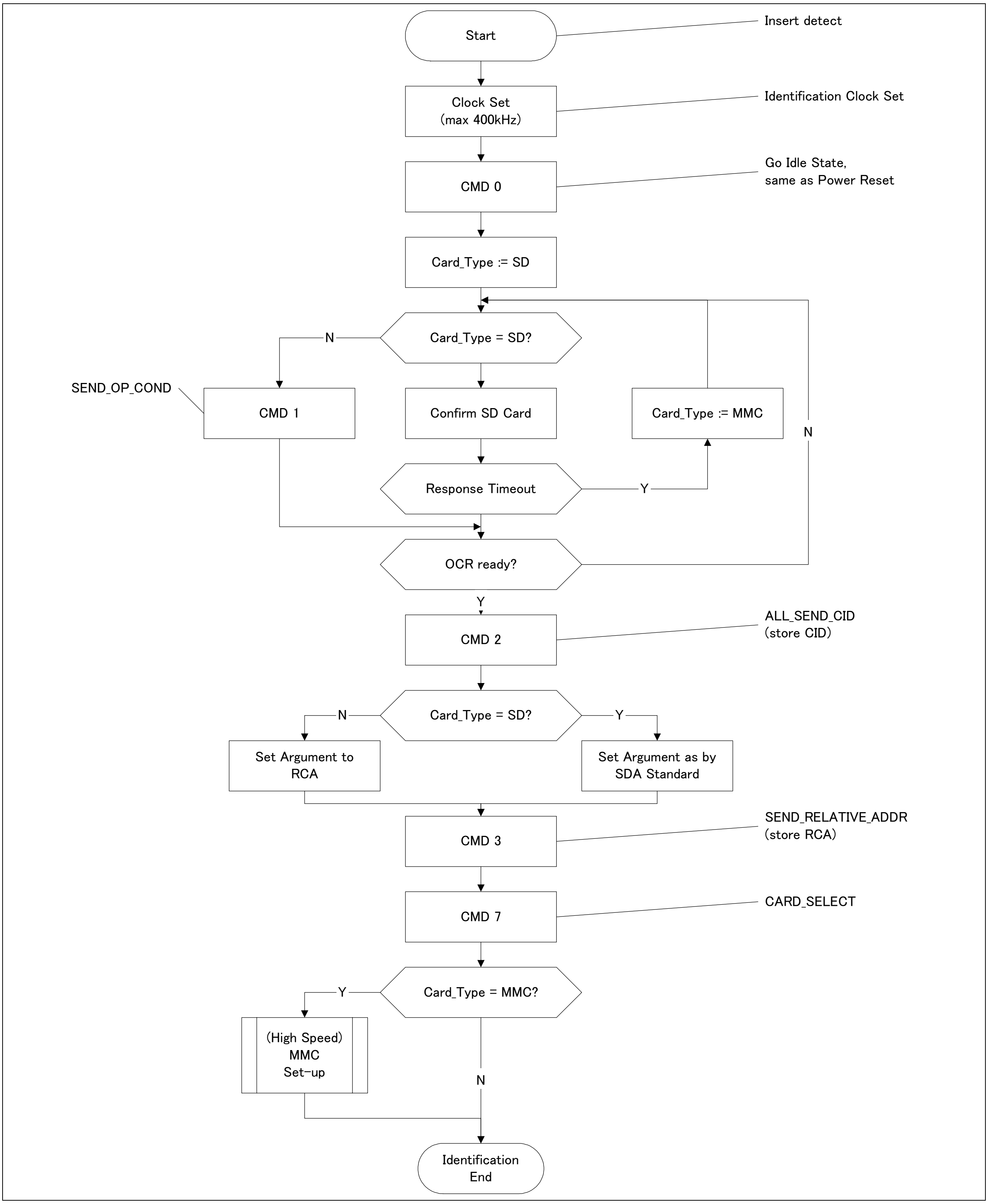

Figure 14 Initialization Flow (allowing card type identification)

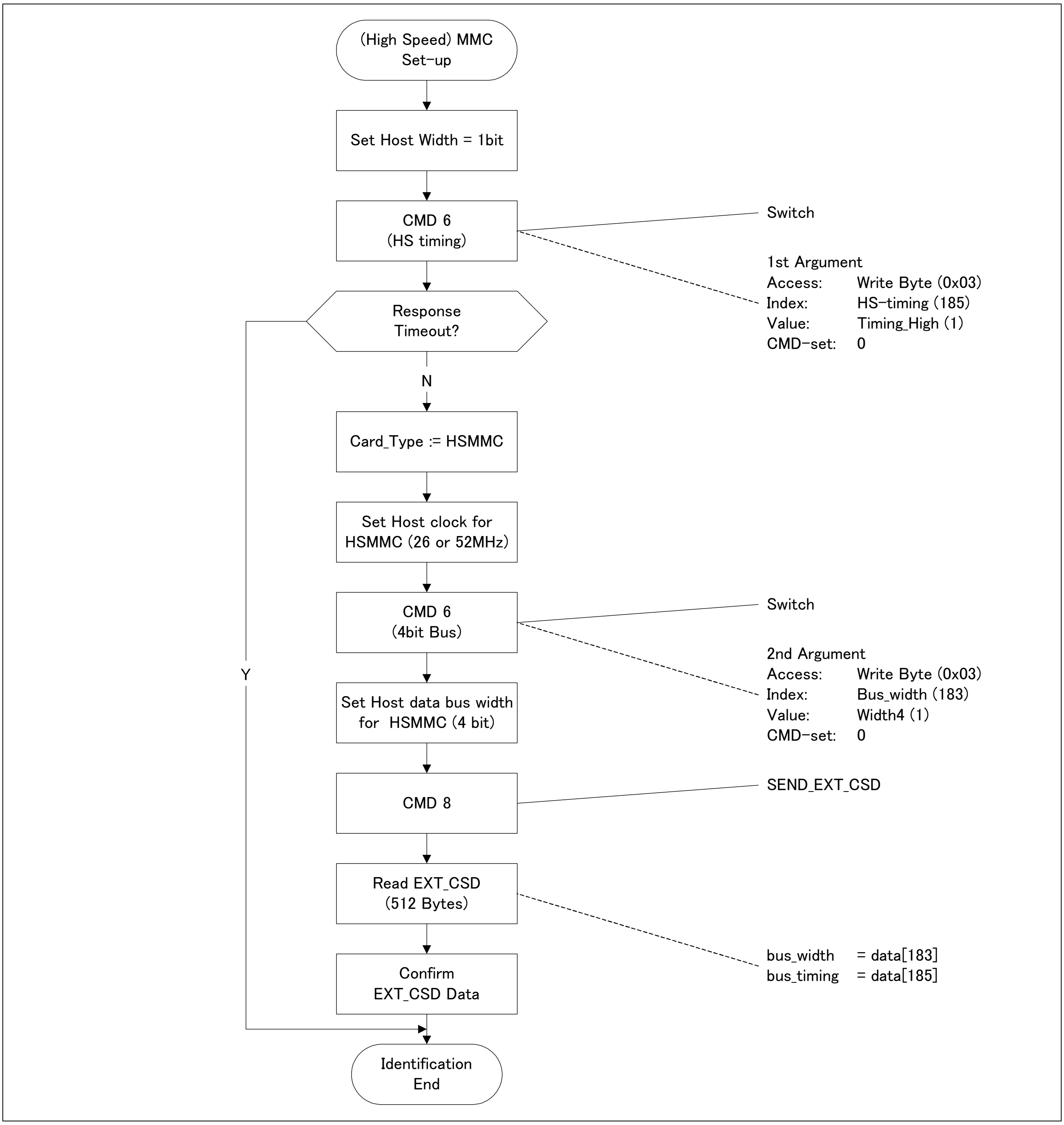

Figure 15 Setting up operation environment for high-speed MultiMediaCard

2.2 Initialization process (example 2)

Please find below the generalized flow leading to the routine initializing a high-speed MultimediaCard.

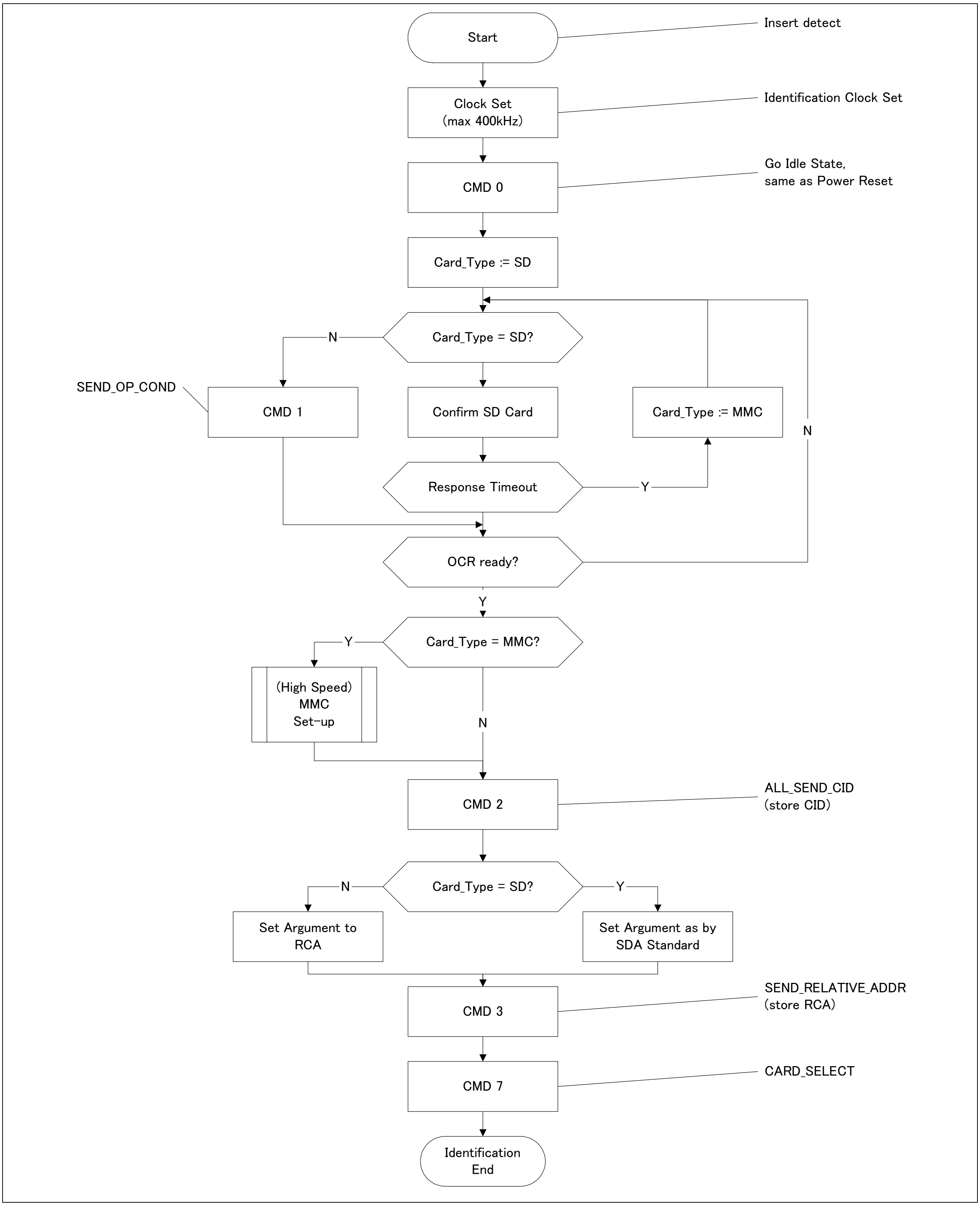

Figure 16 Initialization Flow (allowing card type identification)

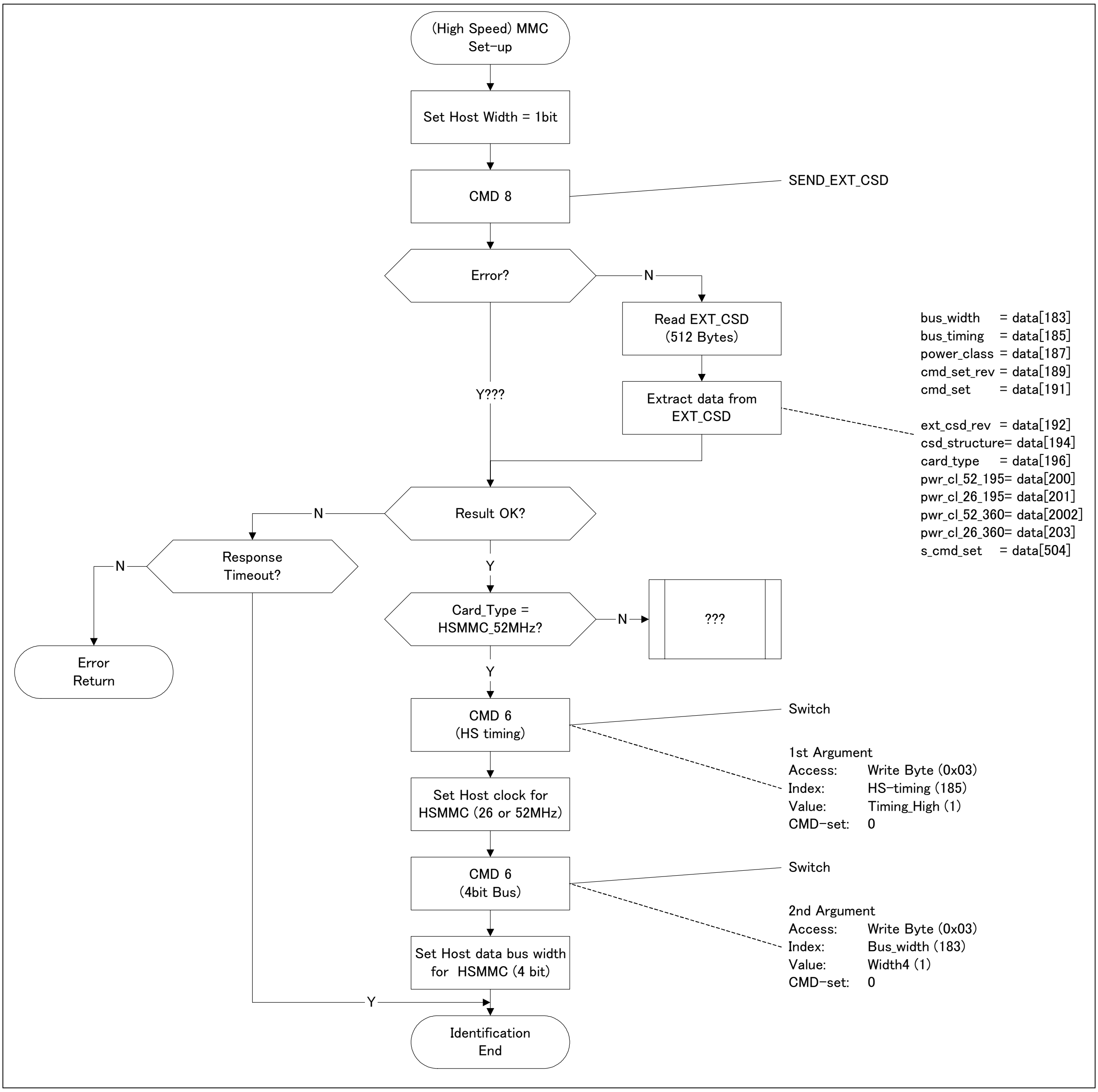

Figure 17 Setting up operation environment for high speed MultiMediaCard

3 Host setup

3.1 x4 Designs

Despite many of todays designs still utilize a single data line for card access, there in an increasing number of hosts demanding the availablity of 4 data lines to achieve minimum performance.

The high-speed MultiMediaCard specification introduces a migration path to the full speed operation. It enables the use of only 4 of its 8 data lines on a frequency of 26MHz. By that the migration from existing (single slot) full-size SD Card designs is rather easy, even if they use 4 data lines.

It should be noticed, that for this step not even the use of new connector hardware is required.

3.2 x8 Designs

A dedicated Full-speed high-speed MultiMediaCard host enables the highest possible performance of a high-speed MultiMediaCard - 52MB/sec (burst). The sustained performance will depend on the actual card design (vastly from the performance supported by the used Non-volatile Memory).

Such a host can be enabled to accept high-speed MultiMediaCard, legacy MultiMediaCard, Full-Size SD. It needs a flexible clock design in order to adjust to the given maximum frequency of each standard if the best possible performance should be achieved.

Sophisitcated connector design would allow to introduce even miniSD and eventually microSD to the same host slot. Multi-format connectors to accept further card standards are available from several suppliers and should be asked for at their side. (A list of according companies can be found on the web site of the MMCA.)

Notes:

info@mmca.org

Published by MultiMediaCard Association