Freescale Semiconductor
Application Note

Document Number: AN2416
Rev. 1, 11/2005

NAND Flash Access

MC9328MX1, MC9328MXL, and MC9328MXS

1 Abstract

This document describes how to access NAND Flash memory from the i.MX through the i.MX processor's External Interface Module (EIM). It also provides the requirements and typical configuration of the i.MX for this access.

This document applies to the following i.MX devices, collectively called i.MX throughout:

- MC9328MX1
- MC9328MXL
- MC9328MXS

1.1 NAND-Flash

NAND Flash is one type of the Flash memory with the memory array made up of 16 cells that are serially connected to form a NAND structure. Each of 16 cells resides in a different page. A block consists of 32 pages formed by two NAND structures. Figure 1 and Figure 2 graphically illustrates the NAND Flash memory structures.

Contents

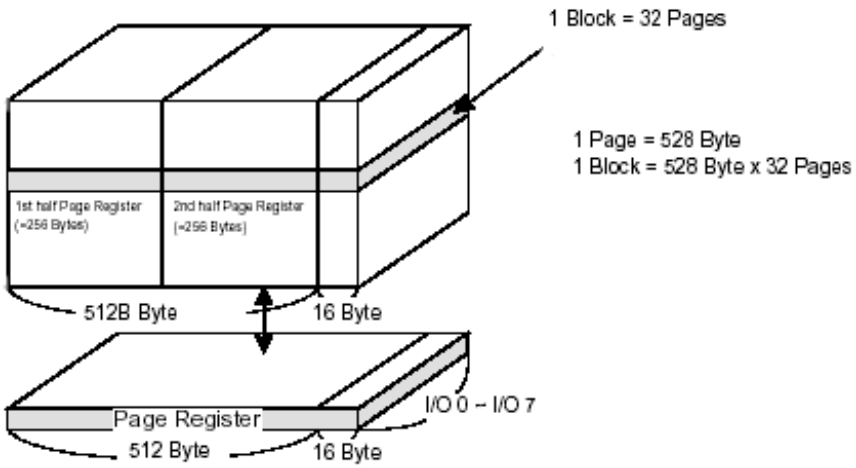

Figure 1. Array Organization of Typical NAND Flash Memory

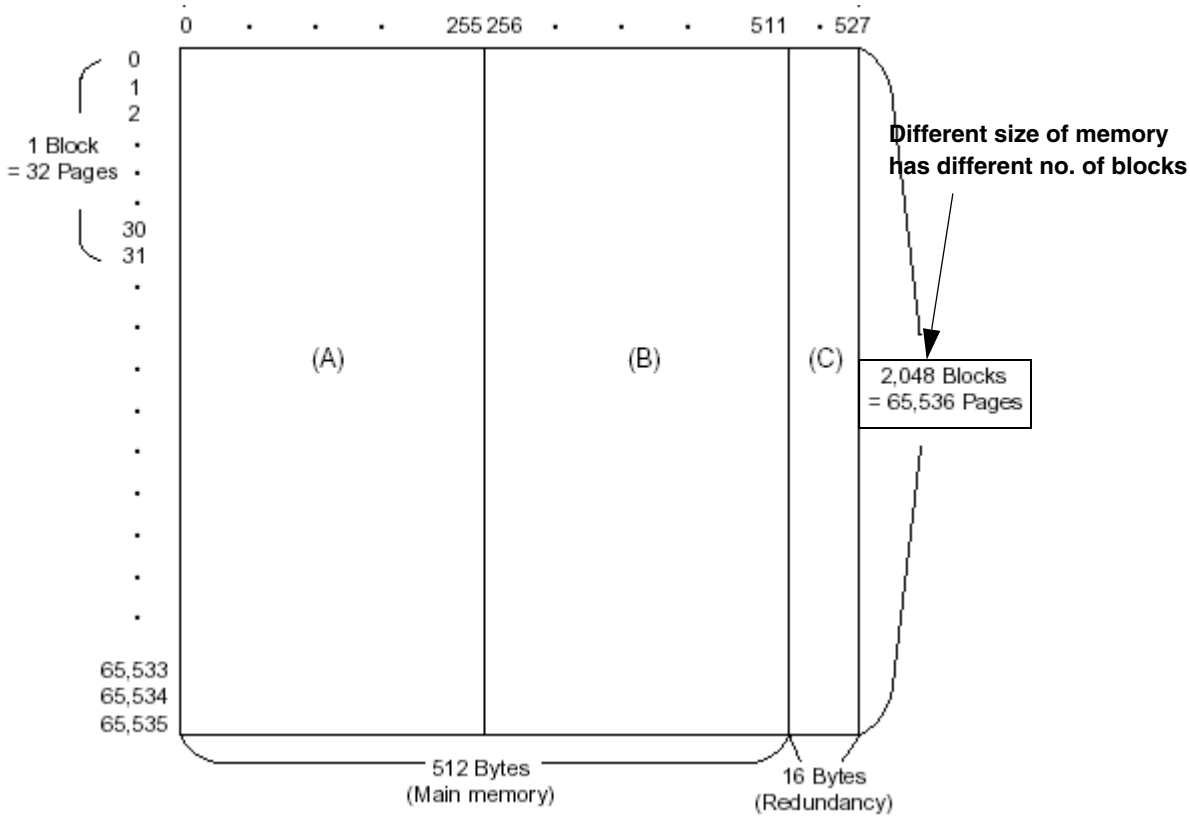

Figure 2. Array Organization in 2D Environment

1.2 SmartMedia Cards

A SmartMedia card is a type of memory card that is commonly used in portable device for mobile storage. It is a NAND Flash based memory. In this application note, the SmartMedia card was chosen for testing because it requires minimal hardware configuration. The Samsung, K9S1208V0M (64 Mbytes), SmartMedia card was chosen for our evaluation.

2 Hardware Configuration

Figure 3 shows the block diagram connection for the i.MX applications processor to the NAND SmartMedia card.

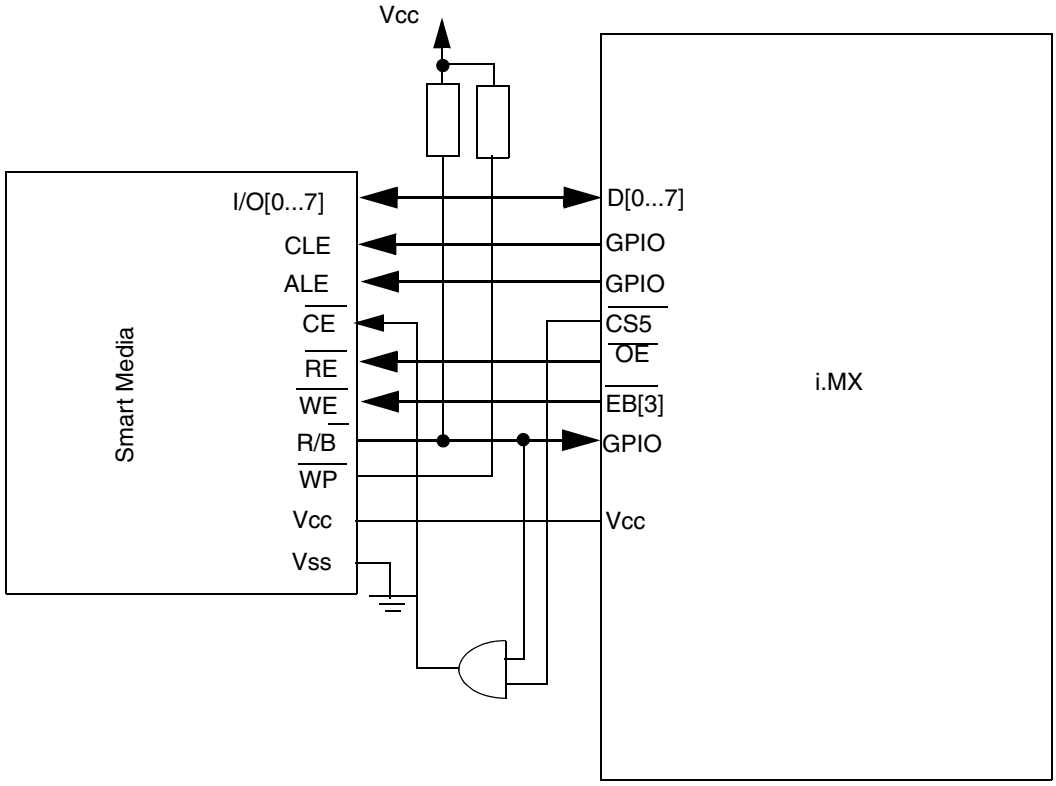

Figure 3. SmartMedia Card Interface with i.MX

NOTE

R/$\overline{B}$ and CS5 are connected to an AND gate to provide chip enable of the SmartMedia card.

The block diagram shown Figure 3 is the configuration between the SmartMedia card and the EIM of the i.MX processor. The following sections describe the requirements of all test operations and how this configuration meets the timing specifications of the SmartMedia card.

2.1 Pin Description

Figure 4 shows the pad assignment diagram of the SmartMedia card followed by Table 1 which provides the pin descriptions.

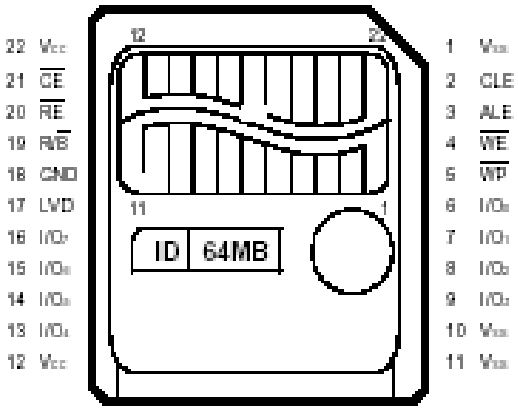

Figure 4. SmartMedia Card Pad Assignment

Table 1. SmartMedia Card Pin Description

Pin Name	Pin Function
I/O [0:7]	I/O pins used to: • Input command, address, and data • Output data during read operations
CLE	Command Latch Enable Controls the activating path for commands sent to SM
ALE	Address Latch Enable Controls the activating path for address to SM
$\overline{CE}$	Chip Enable Device selection control
$\overline{RE}$	Read Enable Serial data-out control
$\overline{WE}$	Write Enable Controls write to the I/O ports
$\overline{WP}$	Write Protect
R/$\overline{B}$	Read/Busy output • Indicates the status of device operation • When low, it indicates that a program, erase or random read operation is in process and returns to high state upon completion.
VCC	Power
VSS	Ground

2.2 Timing Characteristics for Command, Address, and Data Input

The timing characteristics for the SmartMedia card input and output are provided in this section.

Table 2. The AC timing Characteristics for Command/Address/Data Input

Parameter	Symbol	Min	Max	Unit
CLE setup time	tCLS	0	–	ns
CLE hold time	tCLH	10	–	ns
$\overline{CE}$ setup time	tCS	0	–	ns
$\overline{CE}$ hold time	tCH	10	–	ns
$\overline{WE}$ pulse width	tWP	25[1]	–	ns
ALE setup time	tALS	0	–	ns
ALE hold time	tALH	10	–	ns
Data setup time	tDS	20	–	ns
Data hold time	tDH	10	–	ns
Write cycle time	tWC	50	–	ns
$\overline{WE}$ high hold time	tWH	15	–	ns

[1] If CS is set less than 10ns, tWP must be a minimum of 35ns, otherwise tWP may be a minimum of 25ns.

2.2.1 Command Latch Cycle

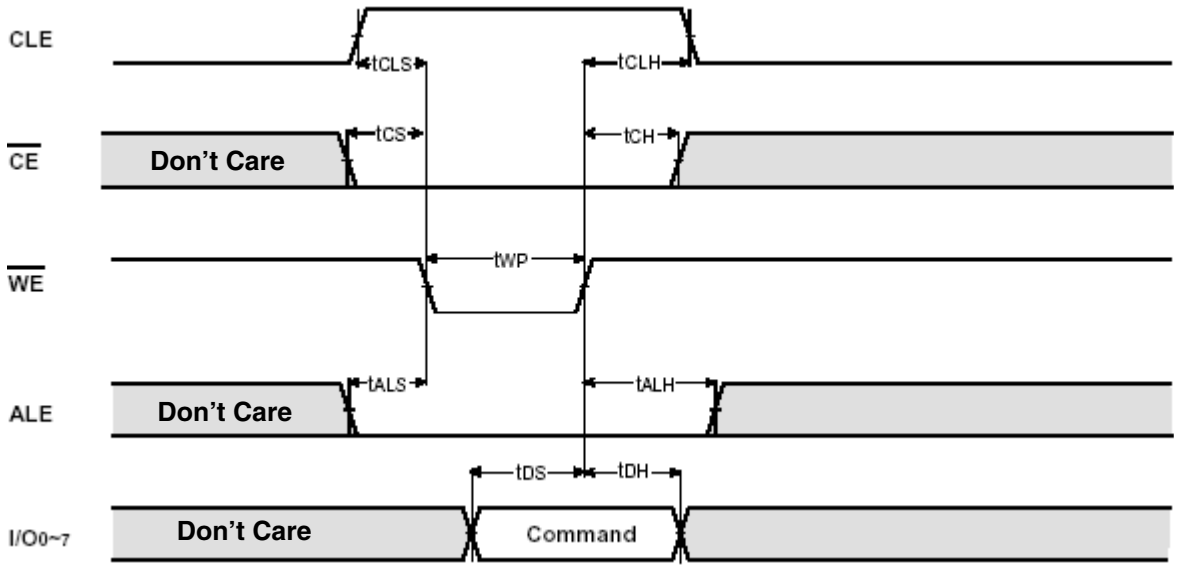

Figure 5. Command Latch Cycle Timing Diagram of SmartMedia Card

2.2.2 Address Latch Cycle

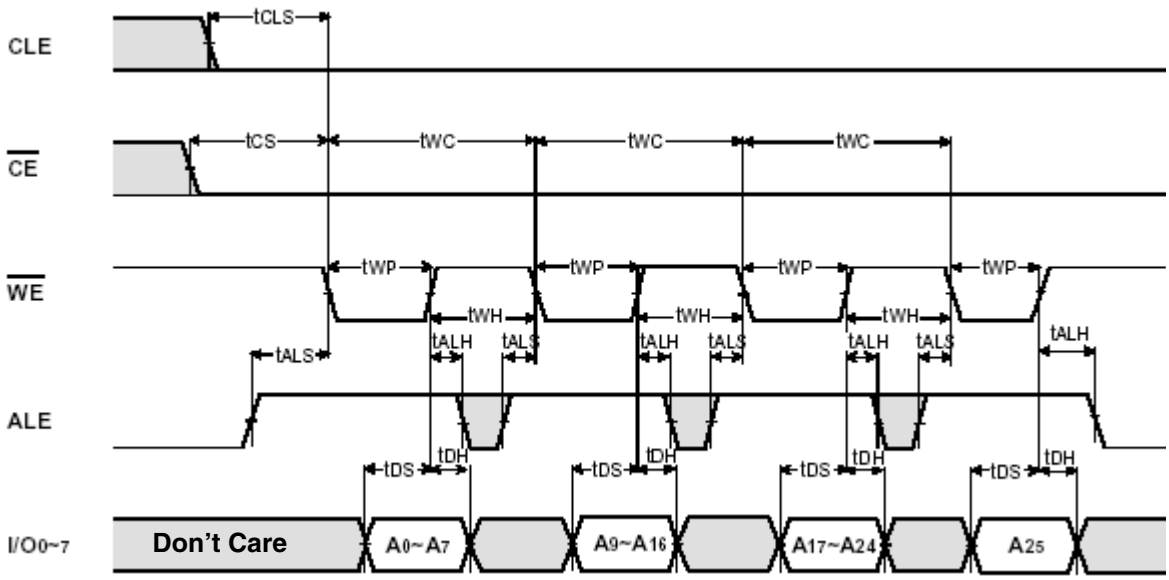

Figure 6. Address Latch Cycle Timing Diagram of SmartMedia Card

2.2.3 Input Data Latch Cycle

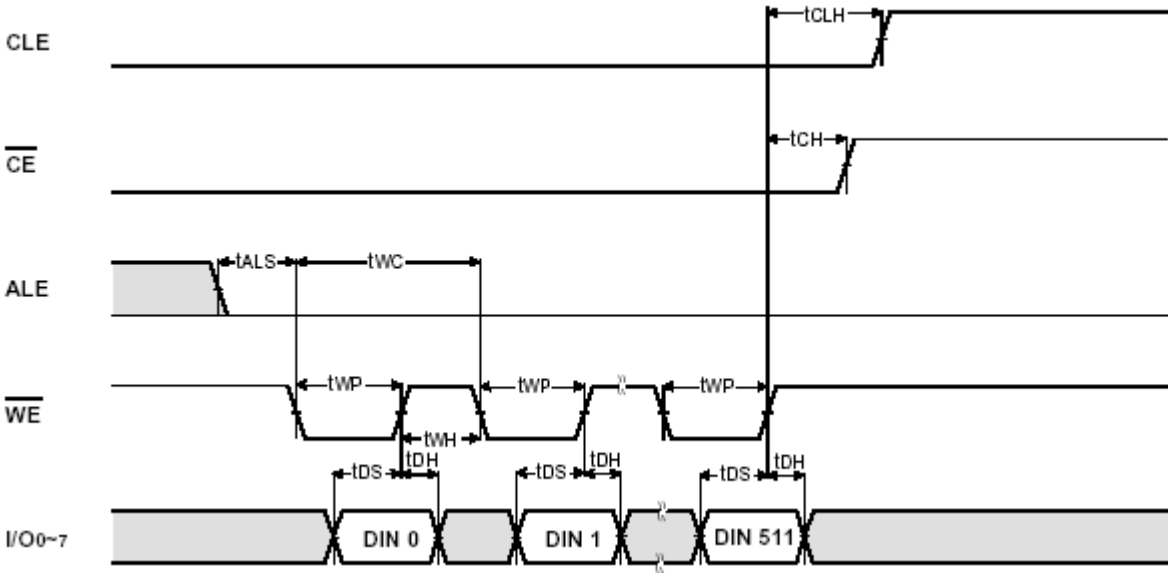

Figure 7. Input Data Latch Cycle Timing Diagram of SmartMedia Card

3 SmartMedia Card Access Operations

To access the SmartMedia card from the EIM of the i.MX processor using the i.MX processor this document presents two primary test operations.

- Read ID operation
- Read/Write operation
  - — Page write
  - — Page read

3.1 AC Characteristics for Operation

Figure 8 provides the AC timing characteristics necessary for test operations on the SmartMedia card.

Parameter	Symbol	Min	Max	Unit
Data Transfer from Cell to Register	tR	-	12	µs
ALE to $\overline{RE}$ Delay(ID read)	tAR1	10	-	ns
ALE to $\overline{RE}$ Delay(Read cycle)	tAR2	50	-	ns
Ready to $\overline{RE}$ Low	tRR	20	-	ns
$\overline{RE}$ Pulse Width	tRP	30	-	ns
$\overline{WE}$ High to Busy	tWB	-	100	ns
Read Cycle Time	tRC	50	-	ns
$\overline{RE}$ Access Time	tREA	-	35	ns
$\overline{RE}$ High to Output Hi-Z	tRHZ	15	30	ns
$\overline{CE}$ High to Output Hi-Z	tCHZ	-	20	ns
$\overline{RE}$ High Hold Time	tREH	15	-	ns
Output Hi-Z to $\overline{RE}$ Low	tIR	0	-	ns
Last RE High to Busy(at sequential read)	tRB	-	100	ns
$\overline{CE}$ High to Ready(in case of interception by $\overline{CE}$ at read)	tCRY	-	50 +tr(R/$\overline{B}$)(1)	ns
$\overline{CE}$ High Hold Time(at the last serial read)(2)	tCEH	100	-	ns
$\overline{CE}$ Access Time	tCEA	-	45	ns
$\overline{WE}$ High to $\overline{RE}$ Low	tWHR	60	-	ns
Device Resetting Time(Read/Program/Erase)	tRST	-	5/10/500(3)	µs

NOTE :
1. The time to Ready depends on the value of the pull-up resistor tied R/$\overline{B}$ pin.
2. To break the sequential read cycle, $\overline{CE}$ must be held high for longer time than tCEH.
3. If reset command(FFh) is written at Ready state, the device goes into Busy for maximum 5us.

Figure 8. AC Timing Characteristics for Operation on SmartMedia Card

3.2 Read ID Operation

For each SmartMedia card, there exists a device code to identify the card model. For the K9S1208V0M, the device code ID is 76h. The Read ID timing diagram is shown in Figure 9.

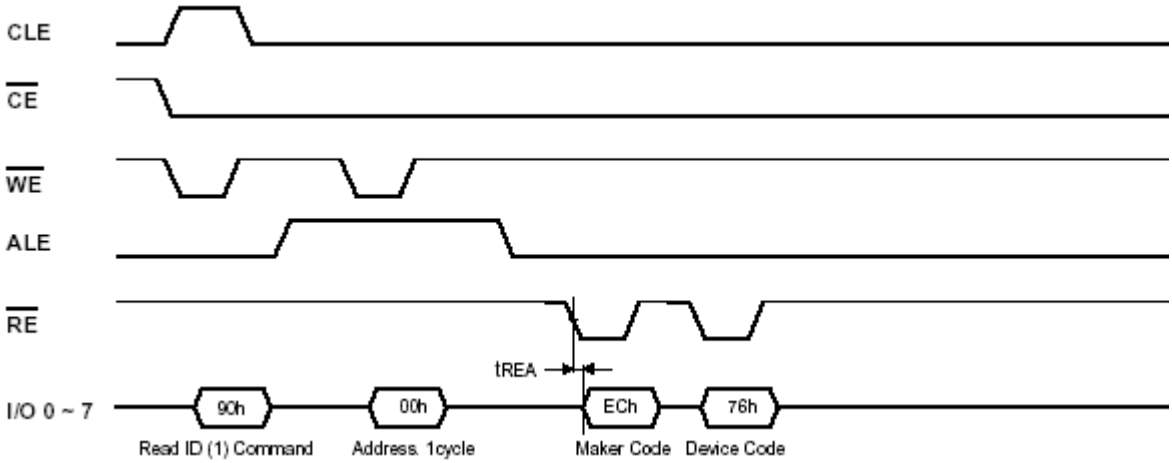

Figure 9. Read ID Timing Diagram of SmartMedia Card

3.3 Read/Write Operation

The command and address latch cycle shown in the Figure 10 timing diagram, must fulfill the AC characteristics for the test operation, especially t_{PROG} which indicates the programming time. Otherwise, the I/O will display irregular data instead of the marker and device code.

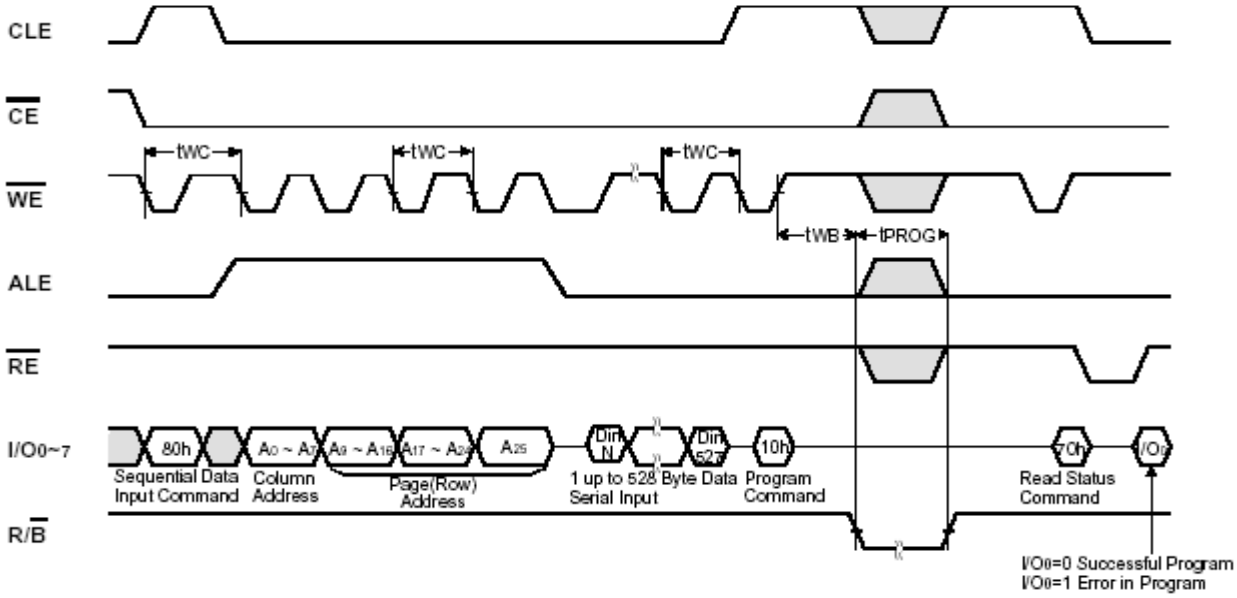

Figure 10. Timing Diagram of Page Write Operation

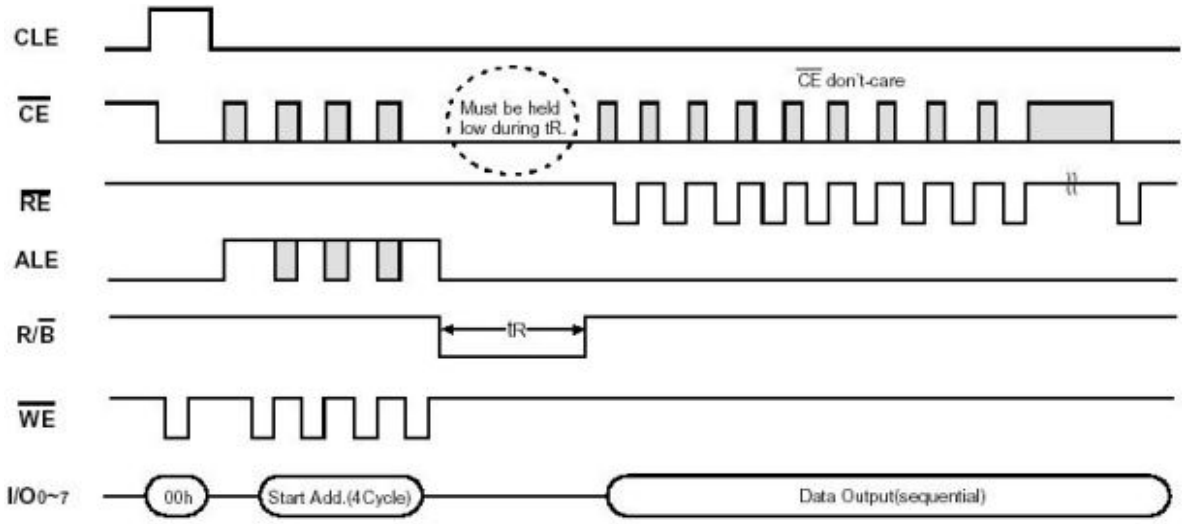

Figure 11. Timing Diagram of Page Read Operation

***Problem*:**

During the Read operation, the $\overline{CE}$ of the SmartMedia card must be held low during $t_R(R/\overline{B})$. However, the i.MX processor's EIM cannot respond to this action and therefore the chip-select remains high during that time.

***Solution*:**

Add an external 2-input AND gate. Signals R/B and CS feed the AND gate inputs. The AND gate output feeds $\overline{CE}$ of the SmartMedia card. This configuration only affects the EIM operation during tR.

4 Software Configuration

SmartMedia cards have command, address, and data multiplexed I/O ports. There are several command sets for the operation of a SmartMedia card. The software initialization for our purposes involves the following two steps as described in this section.

1. Configure EIM of the i.MX processor.
2. Configure GPIO for initialization:
   Initialize the GPIO pins on i.MX for the CLE, ALE, and R/$\overline{B}$ bits of the SmartMedia card.

The test operations of the SmartMedia card that will be performed are:

- Read/Write test of the SmartMedia card to the i.MX processor.
  - — Page read operation
  - — Page write operation
- Read ID operation

4.1 EIM Configuration for i.MX

The software configuration begins by configuring the EIM GPIO pins of the i.MX processor. Configure the GPIO pins for the chip-select output port as shown in Table 3.

Table 3. GPIO Pin Configuration of the i.MX Processor

Pins	Setting	Configuration Procedure
$\overline{CS}$ [5:4]	Primary function of GPIO PortA [23:22]	1. Set bit [23:22] of Port A GPIO Data Direction Register (PTA_DDIR) to configure as output pin 2. Clear bit [23:22] of Port A GPIO In Use Register (PTA_GIUS) 3. Clear bit [23:22] of Port A GPIO General Purpose Register (PTA_GPR)
$\overline{CS}$ [3]	Primary function of pin shared with SDRAM's $\overline{CSD1}$	1. Clear bit 1 (SDCS1_SEL) of Function Muxing Control Register (FMCR)
$\overline{CS}$ [2]	Primary function of pin shared with SDRAM's $\overline{CSD0}$	1. Clear bit 0(SDCS1_SEL) of Function Muxing Control Register (FMCR)
$\overline{CS}$ [1:0]	Not Multiplexed	

The External Interface Module (EIM) is the interface between the i.MX processor and external memory devices. Configure one of the chip-select outputs to communicate with the SmartMedia card. Both the EIM and the GPIO of i.MX processor must be configured as follows. See Example 1.

1. Set CSEN (chip-select enable)
2. Configure the selected chip-select ($\overline{CS}$) as a 8- or 16-bit, according to the NAND Flash memory
3. Configure WSC (EIM_CS5H [45:40]) for the number of wait-states to access an external device
4. Configure OEA (EIM_CS5L [31:28]) and OEN (EIM_CS5L [27:24]) for $\overline{OE}$ timing
5. Configure EBC (EIM_CS5L [11]), WEA(EIM_CS5L [31:28]), and WEN (EIM_CS5L [27:24]) for $\overline{EB}$ timing

Timing of the $\overline{OE}$, $\overline{EB}$[3] for 8-bit NAND Flash or $\overline{EB}$[2] for16-bit NAND Flash must be configured correctly so that it matches within the required timing for the read program on the NAND Flash memory according to the specification.

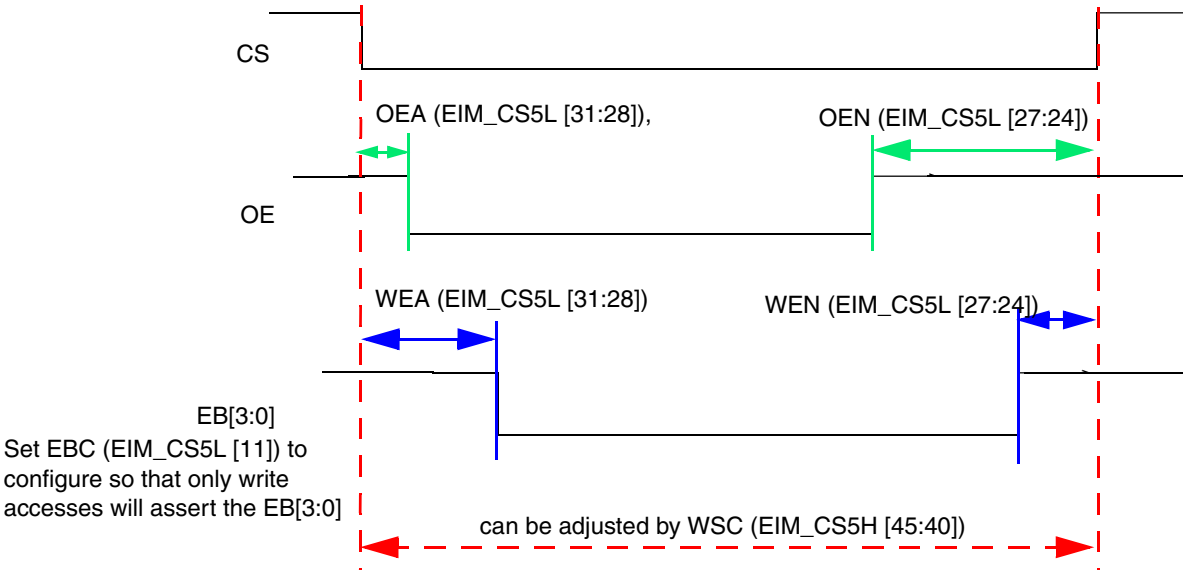

Figure 12. EIM Configuration to Match the Timing Requirement

Example 1. Configure CS5 of EIM Interface to SmartMedia Card

```
void CS5_enable(void)
{
        //CS5 control
        * (U32 *) EIM_CS5H = 0x00000c00;                    // WS
        * (U32 *) EIM_CS5L = 0x14140B01;                    // 8-bit port
                                                            // ENABLE bit control (~EB)

        //port control using CS5
        * (U32 *) PTA_DDIR |=  (0x1 << 23);                 // set output port
        * (U32 *) PTA_GIUS &= ~(0x1 << 23);                 // clear PA23
        * (U32 *) PTA_GPR  &= ~(0x1 << 23);                 // choose primary function
        return;
```

4.2 GPIO Module Configuration

The configuration procedures of the GPIO module to initialize the Address Latch Enable (ALE), Command Latch Enable (CLE), and Read/Busy (R/B) of the SmartMedia card is provided in this section. Refer also to Chapter 32 GPIO Module of the i.MX Reference Manual for additional information.

4.2.1 Configuration for ALE, CLE, and R/B

Begin the initialization of ALE and CLE by first configuring the GPIO module pins as data output followed by configuring the GPIO pins as data input for R/$\overline{B}$.

The procedures to configure Port A/B/C/D pin *i* as GPIO data output are as follows (where *i* denotes any bit from 31–0):

1. Set bit *i* of Port A/B/C/D GPIO Data Direction Register (DDIR) as an output pin.
2. Set bit *i* of Port A/B/C/D GPIO In Use Register (GIUS).

3. The value of DDIR is selected as an output:
   — If $i < 16$, set bits [2 × i +1] and [2 × i] of Output Configuration Register1 (OCR1) as b11
   — If $i \geq 16$, set bits [2 × i-32 +1] and [2 × i-32] of Output Configuration Register2 (OCR2) as b11

Now to configure for R/$\overline{B}$ as GPIO data input. The procedure to configure Port A/B/C/D pin i as GPIO data input is as follows:

4. Clear bit i of Port A/B/C/D GPIO Data Direction Register (DDIR) as input GPIO pin.

Example 2 shows the GPIO configuration of CLE, ALE, and R/$\overline{B}$ with PA1, PA2, and PB16 respectively.

Example 2. GPIO Pin Configuration for CLE, ALE, and R/$\overline{B}$

```
void GPIO_init(void)
{
        // ----------
        //       CLE
        // ----------
        // PA1 as data output
        * (U32 *) PTA_DDIR |= (0x1 << 1);
        * (U32 *) PTA_GIUS |= (0x1 << 1);
        * (U32 *) PTA_OCR1 |= 0x0000000C;// (0x11 << 18);

        // -----------
        //       ALE
        // -----------
        // PA2 as data output
        * (U32 *) PTA_DDIR |= (0x1 << 2 );
        * (U32 *) PTA_GIUS |= (0x1 << 2 );
        * (U32 *) PTA_OCR1 |= 0x00000030; //(0x11 << 20);

        // -----------
        //       R/B
        // -----------
        // PB16 as data input
        * (U32 *) PTB_DDIR &= ~(0x1 << 16);
        * (U32 *) PTB_GIUS |=  (0x1 << 16);

        CLE_disable();
        ALE_disable();

        return;
}
```

4.2.2 Pulse Generation

Figure 13 shows the pulse generation timing diagram results for the CLE and ALE using the GPIO pins of the i.MX processor.

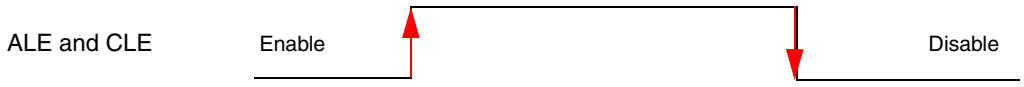

Figure 13. Pulse Generation of ALE and CLE

Example 3 provides the code to generate a pulse for Command Latch Enable (CLE) and Address Latch Enable (ALE) using the GPIO pins of the i.MX processor.

Example 3. Pulse Generation for CLE and ALE Using PA1, PA2 Respectively

```
void CLE_enable(void)
{
        // PA1 as data output
        * (U32 *) PTA_DR |= (0x1 << 1);
        return;
}
```

```
void CLE_disable(void)
{
        // PA1 as data output
        * (U32 *) PTA_DR &= ~(0x1 << 1);
        return;
}
```

```
void ALE_enable(void)
{
        // PA2  as data output
        * (U32 *) PTA_DR |= (0x1 << 2);
        return;
}
```

```
void ALE_disable(void)
{       // PA2 as data output
        * (U32 *) PTA_DR &= ~(0x1 << 2);
        return;
}
```

5 Testing Hardware and Software Configurations on the SmartMedia Card

After the hardware and software configurations are established, testing the configurations is necessary to ensure their function. This section provides the instructions and software code examples to test the capabilities of the configurations.

5.1 Read/Write Operation

Example 4 provides the Page Read/Write software code to write a specific 512 byte page address to the the SmartMedia card.

The steps performed during operation are as follows:

- Page Program—writes a specific 512 byte data into a page address of the NAND Flash memory.
- Page Read—reads back the data from that specific page address.
- Compare—write and read data are compared to verify the success of the read write test.

Example 4. Page Read/Write Operation

```
// ============================================================
//
//       READ & WRITE test on 64M x 8 bit SmartMedia Card
//
// ============================================================

void Read_Write_Test(void)
{
         U8       _read_data[528];                  // store the 528-byte read data
         U8       i = 0x0;
         U32      j  = 0x9;                         // just set it randomly

         // page program 512 byte data to one specific page
         // _gdata is pointing to the 512 byte write data
         Operation_Page_program(i, j, (U8 *) _gdata);

         // read back 528-byte data from the SM card to verify
         Operation_Read1(i, j, (U8 *) _read_data);

         // compare 512-byte data
         Compare(_gdata, _read_data);

         return;
}
```

5.2 Page Write Operation

Example 5 provides the Page Write software code to write a specific 512 byte page address to the SmartMedia card. Figure 14 illustrates the Page Write command flow.

The steps performed during the Page Write operation are as follows:

1. Send command 00h or 01h to set the pointer operation. 00h defines the starting address of the 1st half of the register and 01h defines the starting address of the 2nd half of the register.
2. Send command 80h for page program.
3. Send 4 address cycle, where the address data depends
4. Send 1 up to 528 bytes data to the I/O port.
5. Send command 10h to start page programming
6. Poll the R/$\overline{B}$ bit LOW until it becomes HIGH again.
7. Send Read Status Command 70h and read from the I/O port. $I/O_0 = 0$ indicates successful program and $I/O_0 = 1$ indicates error occur.
8. Verify that the page write operation is successful by completing the Page Read operation and the data read back and compared with the write data.

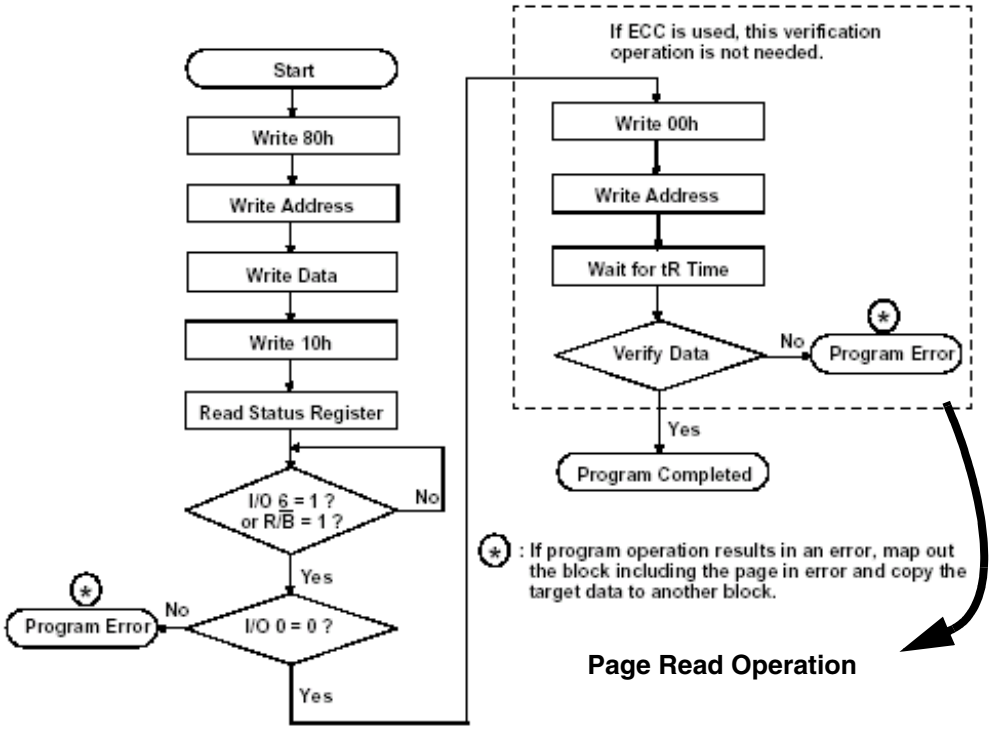

Figure 14. Page Write Flow Chart

Address Cycles:

During the ALE cycle, the data of the column address and the row address/page address (0x0 to 0x1FFFF, total = 131072 pages) will be sent to the SmartMedia card through I/O port. Column address is pointing to the starting address of the page register.

Different NAND Flash require different arrangement of the column / row address data sent through I/O port during ALE cycle.

64M × 8bit SmartMedia card/ K9S1208V0M NAND Flash is used for testing.

The K9S1208V0M is a 528 Mbit (553,648,218 bit) memory organized as 131,072 rows (pages) by 528 columns. Spare sixteen columns are located from column address of 512 to 527. Each page will have [First half of the register (256bytes) + second half of the register (256bytes) + spare register (16bytes)] = (512 +16) byte memory.

The 64 Mbyte physical space requires 26 addresses, thereby requiring four cycles for byte-level addressing: column address, low row address and high row address, in that order.

	I/O 0	I/O 1	I/O 2	I/O 3	I/O 4	I/O 5	I/O 6	I/O 7	
1st Cycle	A0	A1	A2	A3	A4	A5	A6	A7	Column Address
2nd Cycle	A9	A10	A11	A12	A13	A14	A15	A16	Row Address (Page Address)
3rd Cycle	A17	A18	A19	A20	A21	A22	A23	A24	
4th Cycle	A25	*L	*L	*L	*L	*L	*L	*L	

NOTE : Column Address : Starting Address of the Register.
00h Command(Read) : Defines the starting address of the 1st half of the register.
01h Command(Read) : Defines the starting address of the 2nd half of the register.
* A is set to "Low" or "High" by the 00h or 01h Command.
* L must be set to "Low".

Figure 15. Address Cycle for 64M x 8 bit SmartMedia Card

Note that:

00h will be pointing to the first half page.

01h will be pointing to the second half page.

50h will be pointing to the spare part.

Table 2. Destination of the pointer

Command	Pointer position	Area
00h	0 – 255 byte	1st half array(A)
01h	256 – 511 byte	2nd half array(B)
50h	512 – 527 byte	spare array(C)

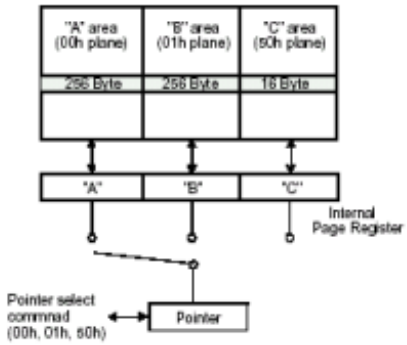

Figure 16. Block Diagram of Pointer Operation

With these commands, the starting column address can be set to any of a whole page(0–527byte).

Example 5. Program Code for Page Write (Program) Operation on 68M x 8 bit SmartMedia Card

```
// =====================================================
//
//       Page Program Operation
//
//        based on
//                64Mb x 8 bit SM card
//                K9S1208V0M-SSB0 NAND flash memory
//
// =====================================================

void Operation_Page_program(U32 column_num, U32 row_num, U8 * wrdata)
{
        U8 * _portaddr = (U8 *) SMCARD_PORTADDR;
        U8 val = 0;
        U32 i;
```

```
printf("Operation_Page_program start\n");

if ((column_num > 0xFF) || (row_num > 0x1FFFF))
        printf("Error in column, row, or block declaration\n");

// write command 00h
CLE_enable();
* _portaddr    = 0x00;
CLE_disable();

// write command 80h
CLE_enable();
* _portaddr    = 0x80;
CLE_disable();

//write 4 address cycle
// pls see the notes on address cycles
ALE_enable();
        // column addr
        * _portaddr    = (U8) column_num;
        // row addr
        * _portaddr     = (U8) ((row_num & 0x000000FF) >> 0 );
        * _portaddr     = (U8) ((row_num & 0x0000FF00) >> 8 );
        * _portaddr     = (U8) ((row_num & 0x00010000) >> 16);
ALE_disable();

//write data to 1st half and 2nd half page registers
//data will NOT be written to the spare 16 bytes
for (i = 0 ; i < 512; i++)
        * _portaddr    = (U8)  wrdata[i];

// write command 10h
CLE_enable();
* _portaddr    = 0x10;
CLE_disable();


// poll R/~B bit
while ( _getstatusSM_RB() );// Wait until 0 return

while (! _getstatusSM_RB() );// Wait until 1 return

// read status command
CLE_enable();
* _portaddr    = 0x70;
CLE_disable();

val = * _portaddr;

#ifdef DEBUG
        printf("read value = 0x%x\n", val);
#endif

if ( val & 0x01 )
        printf("error in program\n");
else
        printf("sucessful program\n");
```

```
        return;

}


// -----------------------------------------
//
//      get status of the R/~B bit (Pb16)
//      using GPIO pin of i.MX
//
// -----------------------------------------
U32 _getstatusSM_RB(void)
{
        // PB16
        U32 status = (  (* (U32 *) PTB_SSR) & (0x1 << 16) ) >> 16;

        return (status);
}
```

5.3 Page Read Operation

Example 6 provides the Page Read software code to read the specified page address of the SmartMedia card. Figure 15 illustrates the Page Read command flow.

The steps performed during the Page Read operation are as follows:

1. Send command 00h or 01h to set the pointer operation. 00h defines the starting address of the first half of the register and 01h defines the starting address of the second half of the register.
2. Send 4 address cycle, where the address data depends.
3. Poll the R/$\overline{B}$ bit LOW until it becomes HIGH again.
4. Unless the operation is aborted, bytes of data for that page address ranging from 1 byte up to 528 bytes, can be read from the I/O port.
5. Send Read Status Command 70h and read from the I/O port. $I/O_0 = 0$ indicates successful program and $I/O_0 = 1$ indicates error occur.
6. If ECC is used, verify by ECC.

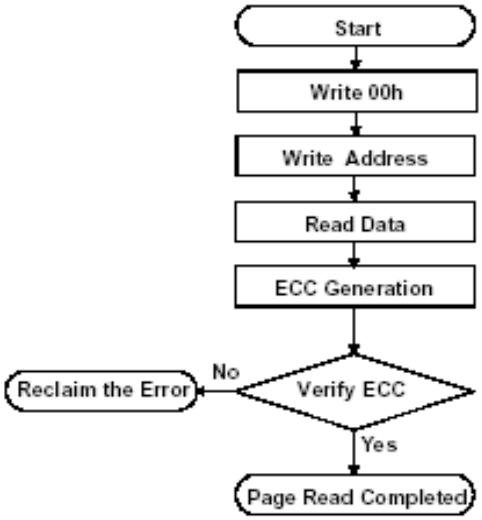

Figure 17. Flow Chart for Page Read Operation

Example 6. Program Code for Page Read Operation on 68M x 8 bit SmartMedia Card

```
// ============================================================
//
//       Page Read Operation
//
//        based on
//                64Mb x 8 bit SM card
//                K9S1208V0M-SSB0 NAND flash memory
//
// ============================================================

void Operation_Read1(U32 column_num, U32 row_num, U8 * _read_data)
{
        U8 * _portaddr = (U8 *) SMCARD_PORTADDR;
        U32 i;

        printf("Operation_Read1 start\n");

        if ((column_num > 0xFF) || (row_num > 0x1FFFF))
                printf("Error in column, row, or block declaration\n");

        // write command 00h
        CLE_enable();
        * _portaddr    = 0x00;
        CLE_disable();


        //write address
        // pls see the notes on address cycles
        ALE_enable();
                // column addr
                * _portaddr    = (U8) column_num;
                // row addr
```

```
                * _portaddr     = (U8) ((row_num & 0x000000FF) >> 0 );
                * _portaddr     = (U8) ((row_num & 0x0000FF00) >> 8 );
        * _portaddr     = (U8) ((row_num & 0x00010000) >> 16);
        ALE_disable();


        // poll R/~B bit
        while ( _getstatusSM_RB() );// Wait until 0 return

        while (! _getstatusSM_RB() );// Wait until 1 return


        // read all data from
        // 1st half page, 2nd half page and spare 16 byte
        for (i = 0 ; i < 528; i++)
                _read_data[i] = * ((U8 *) _portaddr );

        printf("Operation_Read1 done\n");

        return;
}
```

5.4 Read ID Test

The Read ID operation is a simple test to confirm that the SmartMedia card is functioning. Writing special ReadID commands provide the maker code and device code, which vary for different devices. Example 7 provides the software code for this operation.

The steps performed during the Read ID operation are as follows:

ReadID1 Operation:

1. Send ReadID1 Command 90h.
2. Send 1 address cycle: 00h.
3. Read the maker code and device code from the port.

For 64M × 8 bit SmartMedia card or K9S1208V0M NAND Flash,

Maker code = ECh

Device code = 76h

ReadID2 Operation:

1. Send ReadID1 Command 91h.
2. Send 1 address cycle: 00h.
3. Read the maker code from the port.

For 64M × 8 bit SmartMedia card or K9S1208V0M NAND Flash,

Maker code = 20h

Example 7. Program Code for Read ID Operation on 68M x 8 bit SmartMedia Card

```
void Operation_ReadID(void)
{
        U8 * _portaddr = (U8 *) SMCARD_PORTADDR;
        U8 makercode = 0, devicecode = 0;

        // ------------
        //       READ ID 1
        // ------------
        // send command
        CLE_enable();
        * _portaddr        = 0x90;          //Read ID command
        CLE_disable();

        // send address
        ALE_enable();
        * _portaddr        = 0x00;          //ADDRESS 1 cycle
        ALE_disable();

        // 2 read cycles
        // maker code
        makercode = * _portaddr;
        devicecode  = * _portaddr;

        if (makercode != 0xEC)
                printf("wrong manufacture code FOR READ ID 1!?\n");

        if (devicecode != 0x76)
                printf("wrong device code !?\n");

        #ifdef DEBUG
                printf("makercode = 0x%x\n", makercode);
                printf("devicecode = 0x%x\n", devicecode);
        #endif

        //delay for tAR1
        delay(1000);

        // -------------------
        //       READ ID 2
        // -------------------
        // send command
        CLE_enable();
        * _portaddr        = 0x91;          //Read ID command
        CLE_disable();

        // send address
        ALE_enable();
        * _portaddr        = 0x00;          //ADDRESS 1 cycle
        ALE_disable();

        // 2 read cycles
        // maker code
        makercode = * _portaddr;

        if (makercode != 0x20)
                printf("wrong manufacture code for read ID 2!?\n");
```

```
        #ifdef DEBUG
                printf("makercode = 0x%x\n", makercode);
        #endif

        return;
}
```

6 Lab Measurement on i.MX with NAND Flash

The laboratory results of the programming operation performed on the i.MX processor are provided in this section. The testing environment, software settings, and the specific timing results are discussed.

Testing Environment:

- i.MX processor
- Metrowerks' ® Developer Tools with Multi-ICE® Interface Unit
- SmartMedia card 64M × 8 bit or NAND Flash (K9S1208V0M-SSB0)
- Program code can be in external SRAM or embedded memory (i.MX)

Software Settings:

The following diagram are captured by logic analyzer using the following software setting:

- System clock and BCLK running at 96MHz
- Program code in external SRAM
- $\overline{\text{CS5}}$ is configured in EIM of i.MX to communicate the NAND Flash memory with i.MX.
- EIM are configured as:

  * (U32 *) EIM_CS5H = 0x00000c00; // WS

  * (U32 *) EIM_CS5L = 0x14140B01; // 8-bit port with ENABLE bit control ($\overline{\text{EB}}$) set

  With WSC(EIM_CS5H [45:40]) set as 0Ch, that is 12 wait-state, the value is for testing only. The number of wait-states can be reduced to speed up read-write process from NAND Flash, while the timing for the signal to SmartMedia card must fit its specification.

6.1 Command Latch Enable (CLE) Cycle

The Command Latch Enable "latches" the command into the command register through the I/O ports on the rising edge of the $\overline{\text{WE}}$ signal when CLE is high.

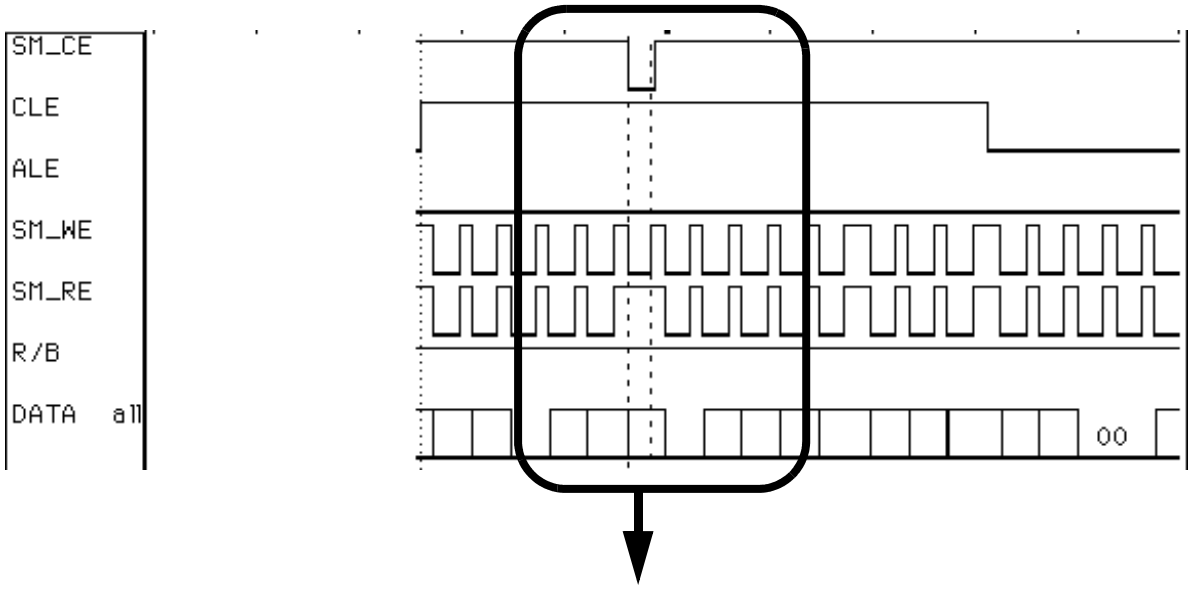

Figure 18. Timing Diagram During CLE Cycle

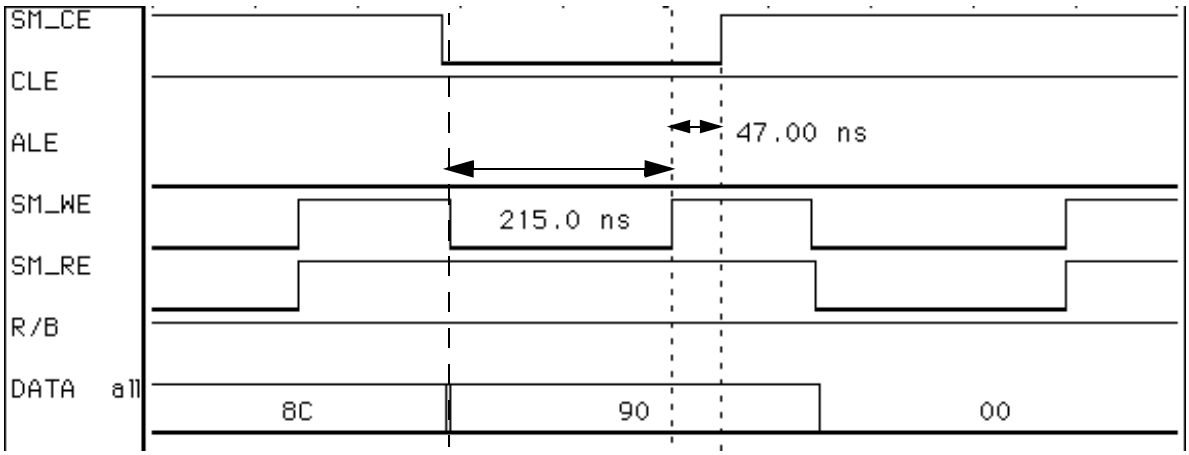

Figure 19. Detailed Timing Diagram of $\overline{WE}$ During CLE Cycle

6.2 Address Latch Enable (ALE) Cycle

The Address Latch Enable latches the address into the internal address registers through I/O port on the rising edge of $\overline{WE}$ with ALE high.

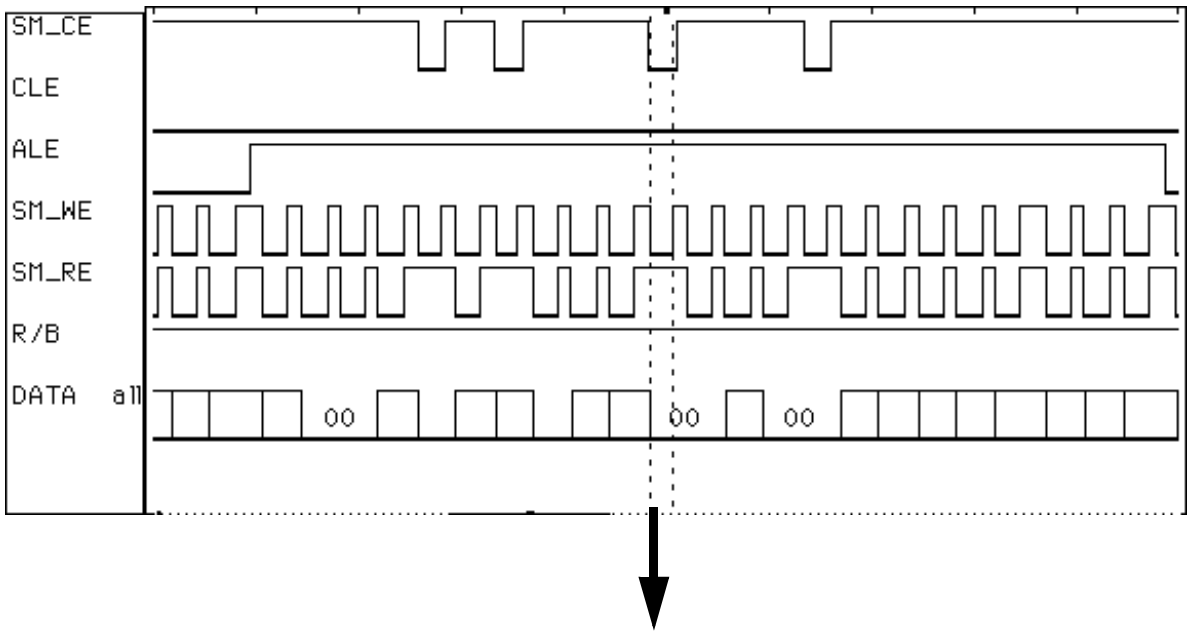

Figure 20. Timing Diagram of ALE Cycle

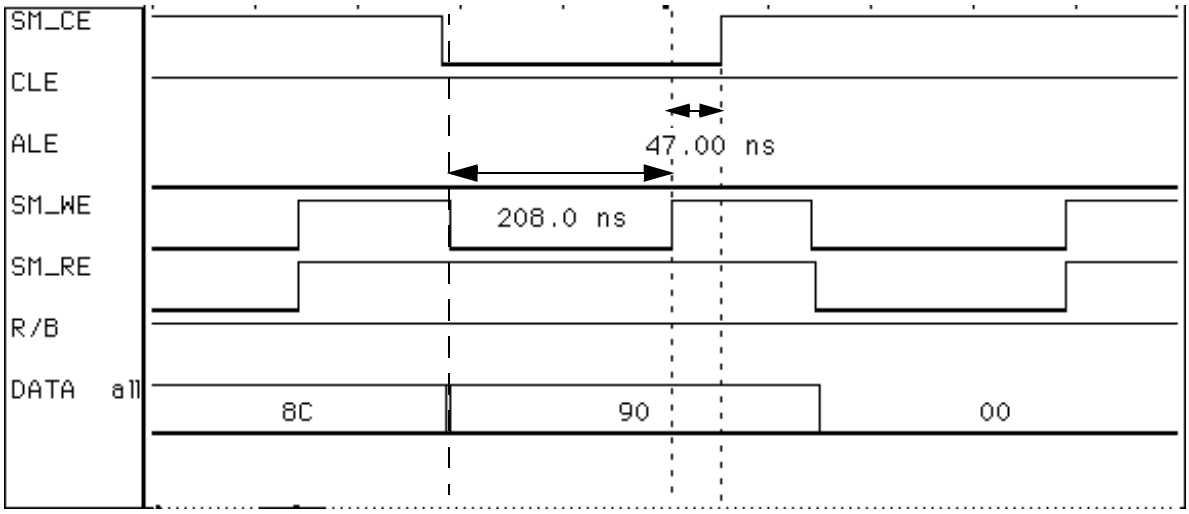

Figure 21. Detailed Diagram Showing the Timing of $\overline{WE}$ during ALE Timing

6.3 Page Write Test Operation

To program data into the page address of NAND Flash.

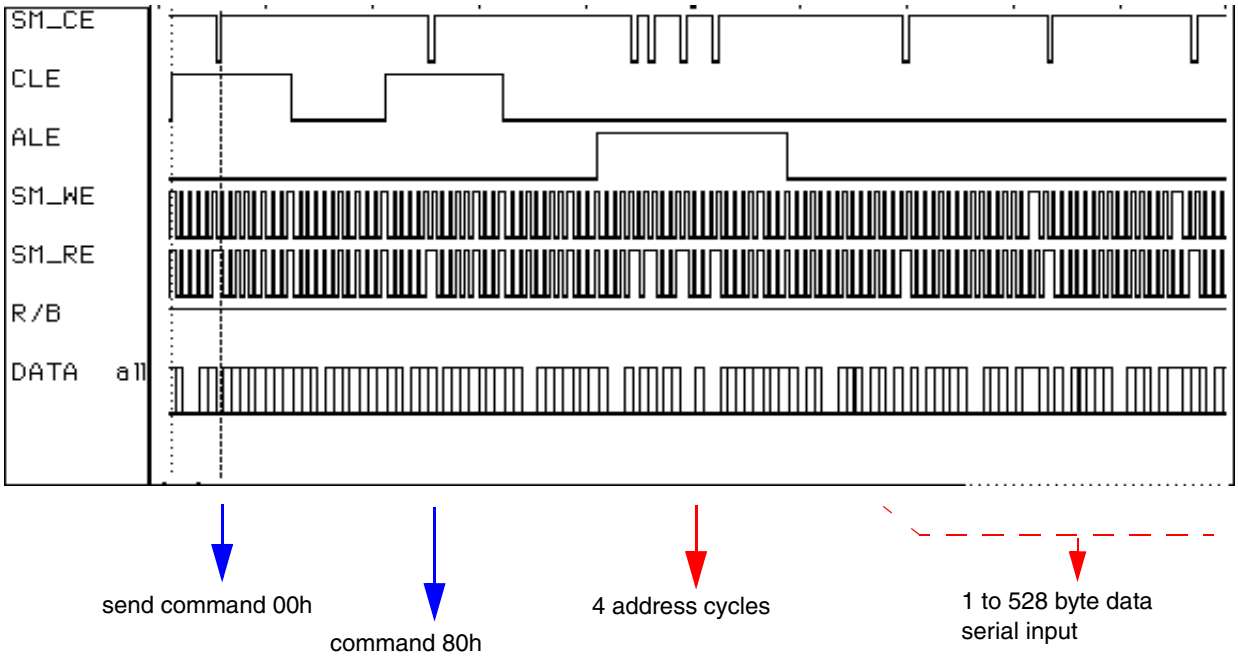

Figure 22. Timing Diagram for Page Program Operation on SmartMedia Card with i.MX

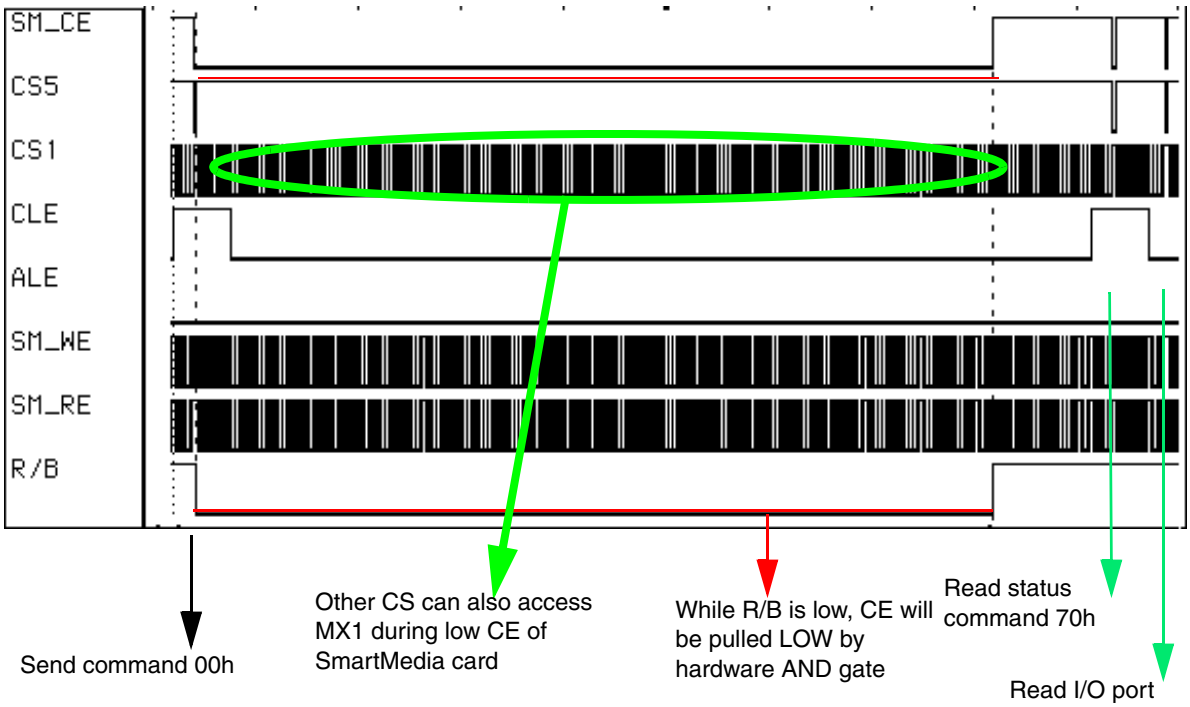

Figure 23. Page-Programming on SmartMedia Card with i.MX

Note that other $\overline{CS}$ can still communicate with i.MX with no bus crash when R/$\overline{B}$ and CE of SmartMedia card is LOW.

6.4 Page Read Test Operation

To page-read from NAND Flash.

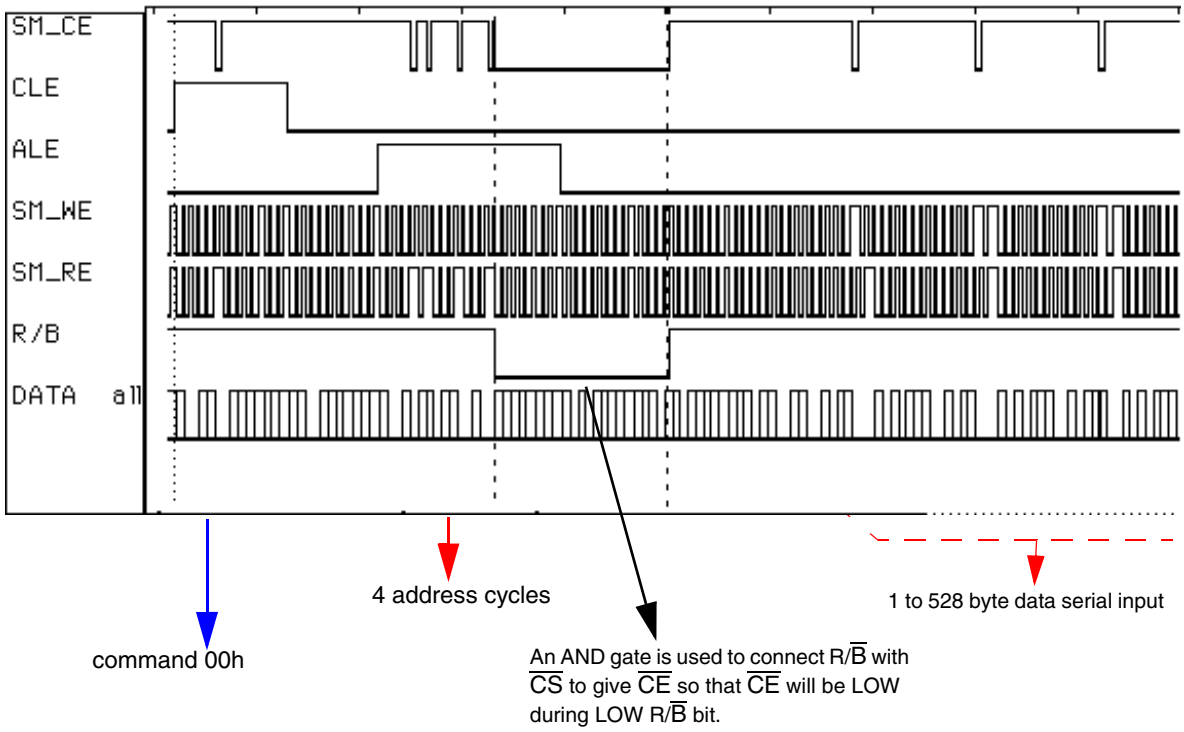

Figure 24. Timing Diagram for Read Operation in 64M x 8 bit SmartMedia Card

Other Chip-Select (CS) can communicate with i.MX during LOW $\overline{CE}$ of SmartMedia card and LOW R/$\overline{B}$ bit.

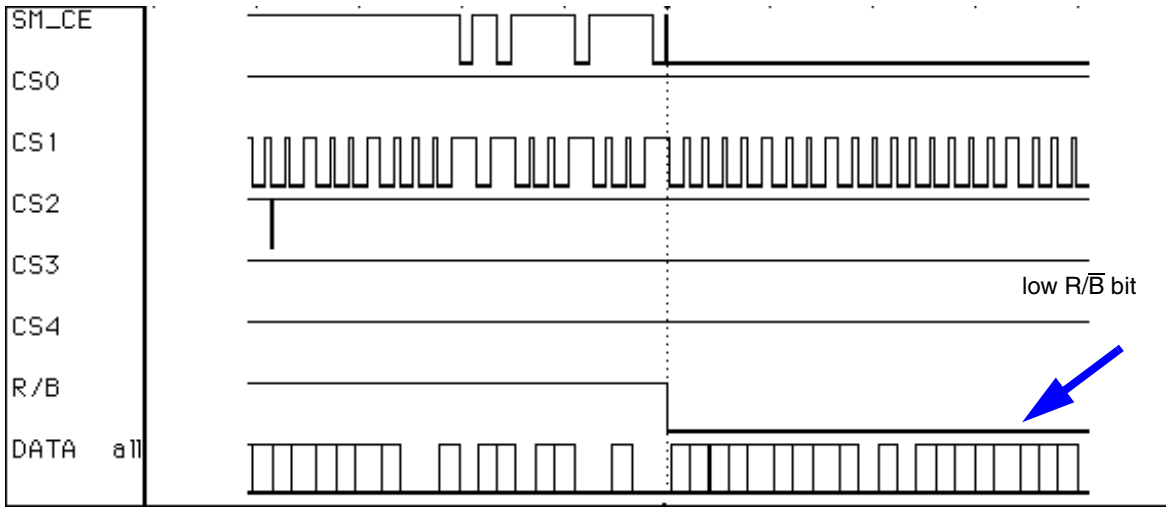

Figure 25. Access of Other Chip-Select (CS) of i.MX during Low R/$\overline{B}$ bit of SmartMedia Card

6.5 Read ID Test Operation

Simple test to read ID value from NAND Flash.

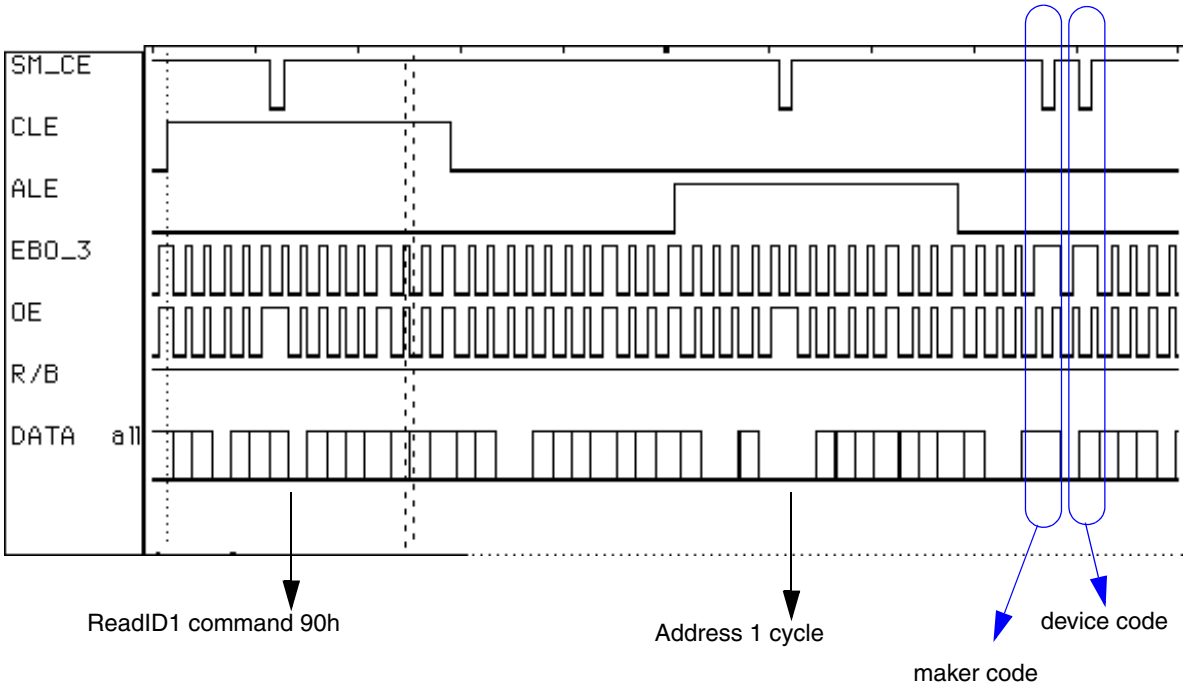

Figure 26. Timing Diagram for ReadID1 Operation on SmartMedia Card with i.MX

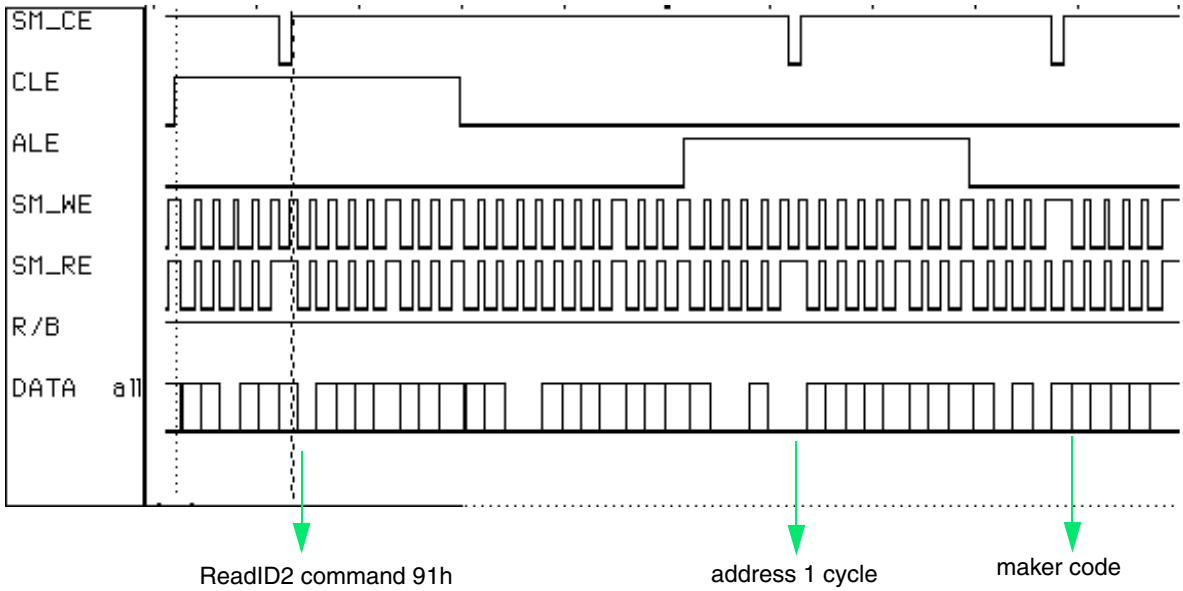

Figure 27. Timing Diagram for ReadID2 Operation on 64M x 8bit SmartMedia Card with i.MX

7 Conclusion

The SmartMedia card has only one I/O port and one command/address/data multiplexed I/O port, in addition to simple command sets which are provided for the different operations.

The i.MX applications processors communicate with NAND Flash through the EIM chip-selects with the GPIO suitably configured. However, the timing of EIM within the i.MX processor must be properly configured to meet the timing requirements of the NAND Flash.

SmartMedia cards contain invalid blocks of NAND Flash to reduce the cost. Invalid blocks must be clearly identified by the software—that is, the invalid block information. Because this application note is primarily concentrated on the Read/Write operations on the NAND Flash using the i.MX processor, program code for invalid block identification is NOT included, please refer to the specification of your particular NAND Flash memory or SmartMedia card for details.

8 References

The following documents can be used for additional information:

4. *MC9328MX1 i.MX Integrated Portable System Processor Reference Manual* (order number: MC9328MX1RM)
5. *MC9328MXL i.MX Integrated Portable System Processor Data Sheet* (order number: MC9328MXLRM)
6. *MC9328MXS i.MX Integrated Portable System Processor Data Sheet* (order number: MC9328MXSRM)

For these and other technical documents about the i.MX products, go to www.freescale.com/imx.

9 Revision History

This revision is for the purpose of applying the Freescale template and does not include technical content changes.

NOTES

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only*:**
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN2416
Rev. 1
11/2005

freescale™
semiconductor