Application Note

AN2284/D
Rev. 1.0, 09/2003

Interfacing the MC9328MX1 with Micron SyncFlash

By Michael Kjar

Contents

1 Introduction

This application note provides details about how to interface and use the Freescale DragonBall™ MC9328MX1 processor with the Micron® SyncFlash® by describing the hardware interface between the MC9328MX1 and the SyncFlash. Software examples are also provided that show how to erase and program the SyncFlash memory. The examples shown in this application note assume the use of two MT28S4M16LC devices (4M x 16 x 2 devices) that form one 32-bit device with a total memory size of 16 Mbytes. Because the SyncFlash features an SDRAM compatible interface, it is assumed that the reader has a good working knowledge of SDRAM operation. This document does not describe SDRAM operational details.

1.1 MC9328MX1 SDRAM Controller Overview

The MC9328MX1 is an application processor targeted for low power, portable applications. Part of the rich feature set of the MC9328MX1 includes the on-chip SDRAM controller. The MC9328MX1 SDRAM controller includes the following features:

- Supports 4 banks of 64, 128, or 256 Mbit synchronous DRAMs
- Includes 2 independent chip selects
  - Up to 64 Mbytes per chip select
  - Up to four banks simultaneously active per chip select
  - JEDEC standard pinout/operation
- Supports the Micron SyncFlash SDRAM-interface burst flash memory
  - Boot capability from CSD1
- PC100 compliant interface
  - 100 MHz system clock achievable with “-8” option PC100 compliant memories
  - Single and fixed-length (8-word) word access
  - Typical access time of 8-1-1-1 at 100 MHz

- Software configurable bus width, row and column sizes, and delays for differing system requirements
- Hardware supported self-refresh entry and exit which keeps data valid during system reset and low power modes
- Auto-powerdown (clock suspend) timer

Figure 1 shows the MC9328MX1 SDRAM Controller block diagram. This figure shows the muxing of internal signals of the SDRAM controller to the external signals of the MC9328MX1 bus. Refer to the *MC9328MX1 Reference Manual* for more detailed information on the SDRAM Controller.

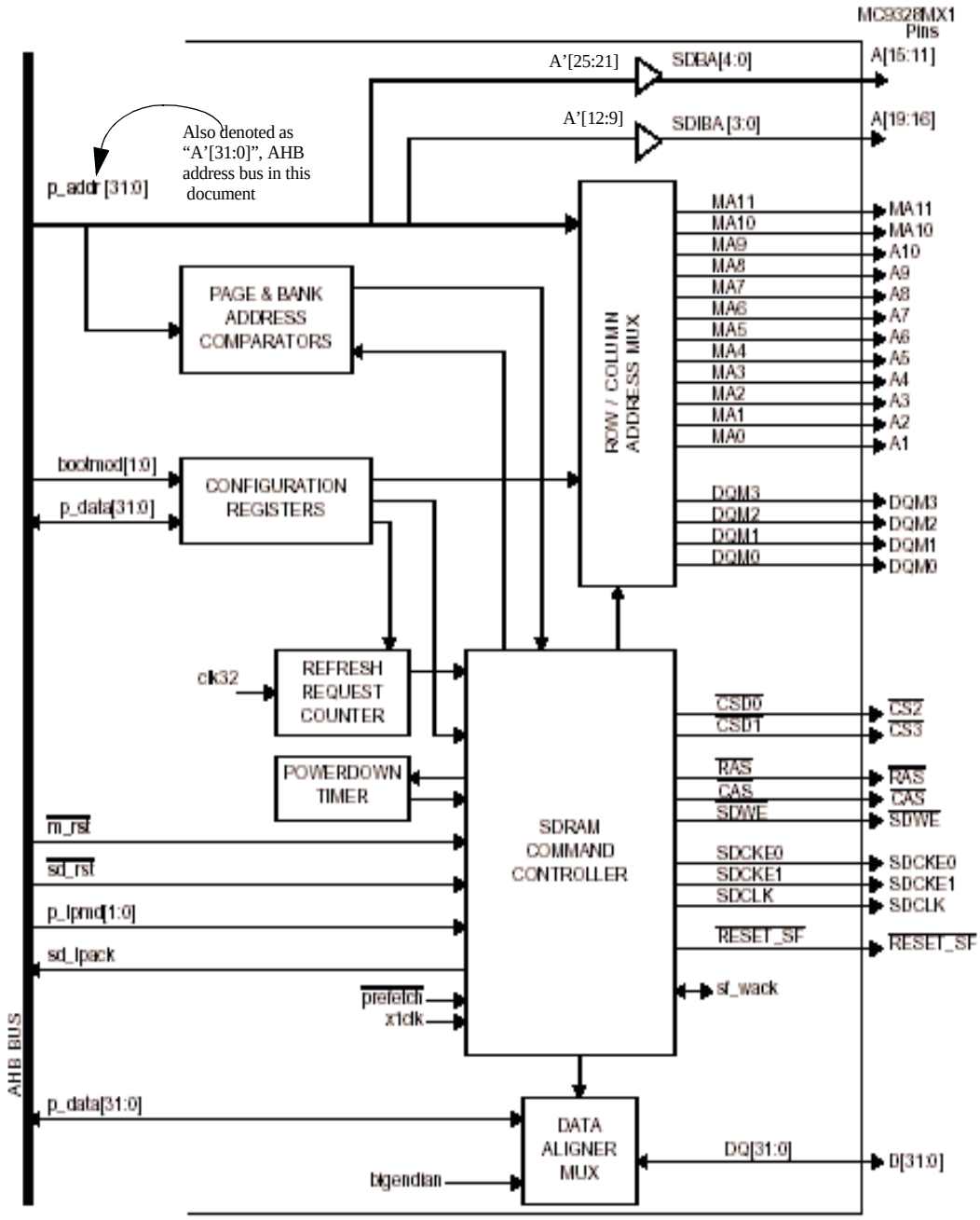

Figure 1. SDRAM Controller Block Diagram

1.2 Micron SyncFlash Overview

The Micron SyncFlash® is a nonvolatile, electrically sector-erasable, (Flash) programmable memory designed to operate in low-power systems. The Micron SyncFlash uses a standard NOR-based flash memory array, but with a standard SDRAM interface. This allows for READ cycles identical to standard SDRAMs, while allowing programming operations to share the same command encoding as SDRAM devices.

The SyncFlash part number used in this document is the MT28S4M16LC. This is a 4M x 16 SyncFlash device that operates at 3.3V. Refer to http://www.micron.com to search for the latest MT28S4M16LC data sheet.

2 Hardware Interface

This section describes the hardware interface between the MC9328MX1 and the SyncFlash device.

2.1 Interfacing the MC9328MX1 With the SyncFlash

Interfacing the MC9328MX1 with the SyncFlash is a relatively simple process. Again, this note assumes that users are employing two (2) MT28S4M16LC devices (4M x 16 x 2 devices) which will provide a total of 128 Mbits or 16 Mbytes of memory.

The MC9328MX1 SDRAM controller supports the bank-interleaving feature which allows the addresses to flow through one page in the first bank, to one page in the second, to one page in the third, etc. This allows these banks to alternate at each SDRAM page boundary allowing for an increase in memory access speed performance because it eliminates the need (and time) to close one page before attempting to access another page. Essentially, this allows that page in a particular bank to remain open while accessing another page on a different bank. However, due to the block orientated nature of erase/program operations in flash memories, interfacing to the SyncFlash in bank-interleaved mode may not be possible if the user wants to execute out of the device being programmed. Also, if the device is programmed outside the system, the data must be shuffled to account for the interleaved blocks. Therefore, Freescale recommends that users interface to the SyncFlash device in the non-bank-interleaved mode, sometimes called linear addressing. The examples shown in this document assume that the non-bank-interleaved (linear addressing) mode is used. Figure 2 shows the hardware interface between the MC9328MX1 and the Micron SyncFlash.

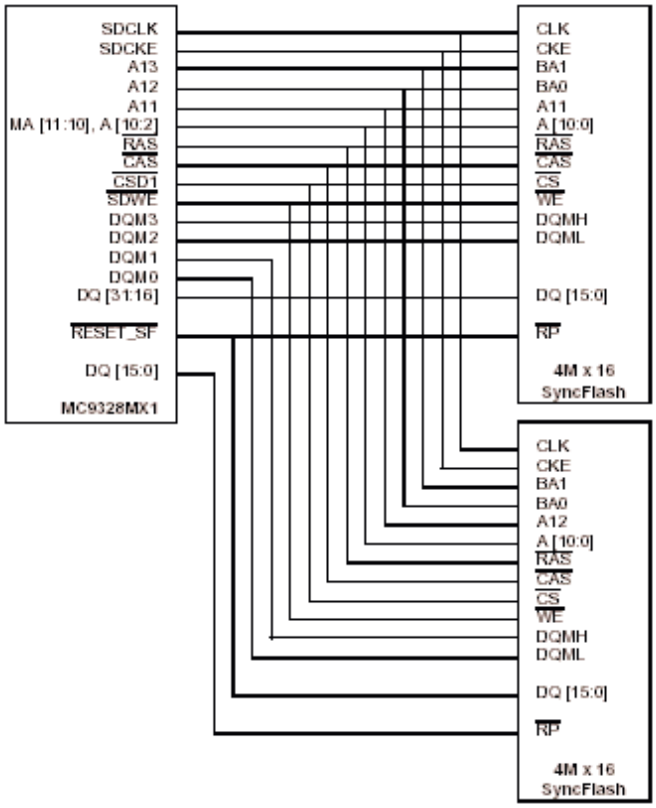

Figure 2. Dual 64 Mbit SyncFlash Connection Diagram

2.2 MC9328MX1 SDRAM Controller Address Mux

The previous section described how to interface the SyncFlash memory to the MC9328MX1. This section describes the MC9328MX1 SDRAM Controller address muxing scheme. Understanding the muxing scheme will help users determine the correct “internal” MC9328MX1 AHB address in order to access the desired “external” address to the SyncFlash memory.

Table 1 shows the MC9328MX1 SDRAM Controller address muxing scheme for a memory configuration of 4M x 16 x 2 in non-bank-interleaved mode (SDCTL1[IAM] = 0). The same SyncFlash memory configuration used for this application note. Also, following the JEDEC standard for a 4M x16 memory configuration, the SyncFlash memory has twelve row address bits, eight column address bits, and two bank address bits. Notice that the internal AHB address bits denoted as “A’xx” are different than the external address bits, denoted simply as “Axx”, and thus there is a level of translation or mapping that must be performed between the internal AHB address bits and the external address bits.

Internal addresses A’[25:21] are mapped directly to external addresses A[15:11]

Internal addresses A’[12:9] are mapped directly to external addresses A[19:16]

Figure 1 shows this address translation in detail.

Table 1. MC9328MX1 SDRAM Controller Address Muxing Scheme for a 4M x16 Memory Configuration

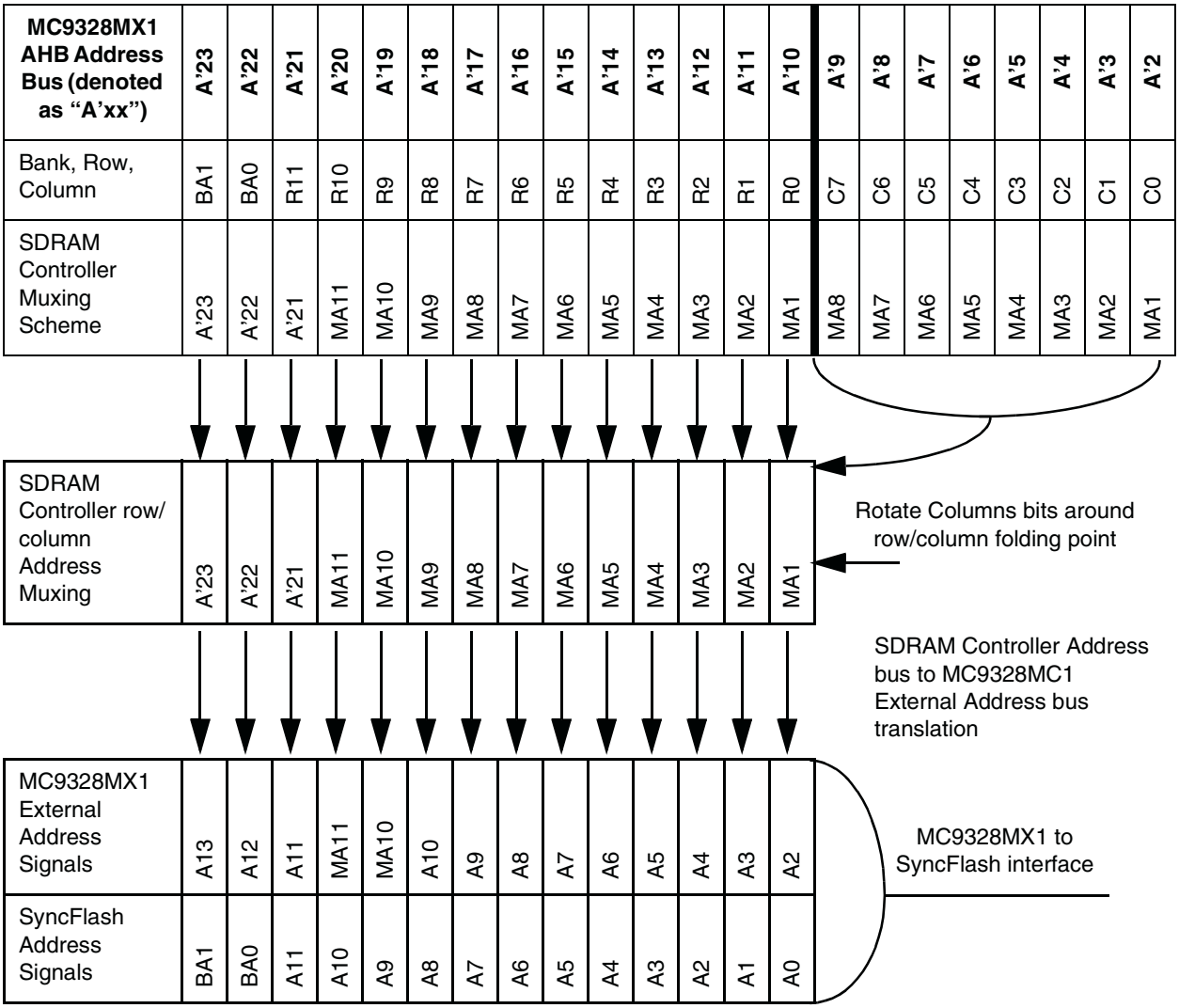

Table 1 shows how the row and column bits are positioned according to the MC9328MX1 internal AHB address bus as follows.

- C[7:0] indicate the column address bits
- R[11:0] indicate the row address bits
- BA[1:0] indicate the bank address bits

Table 1 also shows the following:

- How these bits are muxed with the SDRAM Controller Address Muxing bits MA1 through MA11.
- How these muxed addresses are "folded" on the border of the row and columns bits to form one row of muxed address bits (MA1 through MA11). Some of the internal AHB address bus bits are not multiplexed with the SDRAM Controller address mux bits, particularly address bits A'23 through A'21.

One more translation is completed as the SDRAM Controller address mux bits are brought out to the external address bus of the MC9328MX1. Refer to Figure 1 on page 2 to see how the SDRAM Controller address mux bits are translated to the MC9328MX1 address bus.

The final two rows of Table 1 show the connection interface between the MC9328MX1 and the SyncFlash memory shown in Figure 2.

2.3 Mode Register Programming

This section provides an overview of the SyncFlash mode register and how to program it.

2.3.1 Mode Register Overview

In keeping with the SDRAM standard, the SyncFlash memory has a mode register that defines its specific mode of operation. The mode register defines the SyncFlash operation by specifying the burst length, the burst type, and the CAS latency. To program the mode register, the MC9328MX1 must first issue the LOAD MODE REGISTER command, then the data to be written to the mode register is transferred across the address bus. The mode register retains this programmed data until it is reprogrammed or until the device loses power. Therefore, the mode register contents may be copied into the nvmode (non-volatile mode) register. After power is cycled, the mode register settings are loaded into the mode register from the nvmode register upon device initialization. The process of programming the mode register is described in Section 2.3.2, “Programming the SyncFlash Mode Register” on page 6. The process of programming the nvmode register is described in Section 3.2.3, “NVMODE Register Erase and Program” on page 10.

2.3.2 Programming the SyncFlash Mode Register

The previous section provided an overview of the SyncFlash mode register and the process by which it is programmed. In summary, the LOAD MODE REGISTER command must first be issued by the MC9328MX1, followed by placing the data to be programmed into the mode register onto the address bus. This section describes how the data is written onto the address bus given the internal address muxing of the MC9328MX1.

Once the mode register value is determined, it must be converted into an address. After issuing the LOAD MODE REGISTER command, any READ/WRITE access to this specific address places the contents of the address into the SyncFlash mode register. The SyncFlash mode register bits are mapped to specific address pins of the SyncFlash memory. Table 2 shows this configuration.

Table 2. SyncFlash Mode Register Definition

SyncFlash Address	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bits	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Contents	Reserved		WB	Op Mode		CAS Latency			BT	Burst Length		

Figure 2 shows how the MC9328MX1 address bus aligns with the SyncFlash address pins, while Table 1 shows how the internal AHB address bus is multiplexed through the SDRAM Controller and on out to the external address bus. The next step is to determine how the MC9328MX1 address bus aligns with the MC9328MX1 internal AHB bus so that the correct address is written out to the SyncFlash address pins. The information shown in Table 3 takes the information shown in Table 1 a step further and shows the address translation from the MC9328MX1 internal AHB bus to the SyncFlash address pins, and summarily to the SyncFlash mode register bits.

Table 3. MC9328MX1 Address Translation to SyncFlash Mode Register Bits

MX1 Internal Address	A'31	A'30	A'29	A'28	A'27	A'26	A'25	A'24	A'23	A'22	A'21	A'20	A'19	A'18	A'17	A'16	A'15	A'14	A'13	A'12	A'11	A'10	A'9	A'8	A'7	A'6	A'5	A'4	A'3	A'2	A'1	A'0
SyncFlash Address/ Mode Register bits	x	x	x	x	CS base	CS base	x	x	BA1	BA0	A11/M11	A10/M10	A9/M9	A8/M8	A//M7	A6/M6	A5/M5	A4/M4	A3/M3	A2/M2	A1/M1	A0/M0	x	x	x	x	x	x	x	x	x	x

Table 3 shows how the internal AHB addresses are mapped to the SyncFlash memory. In this table, all cells marked with an 'x' indicate that the internal address bit is not used. The two cells labeled "CS base" indicate the base of the memory region in which the SyncFlash memory resides. On the MC9328MX1, there are two chip select signals dedicated for SDRAM operations, $\overline{CS2}$ and $\overline{CS3}$. A 64 Mbyte space is allocated to each chip select. The SDRAM arrays are mapped according to Table 4. For this example, the SyncFlash will reside on $\overline{CS3}$ with the base address of 0x0C000000.

Table 4. SDRAM Array Memory Map

Address	Use	Access
0x0800 0000 - 0x0BFF FFFF	SDRAM 0 Memory Array	R/W
0x0C00 0000 - 0x0FFF FFFF	SDRAM 1 Memory Array	R/W

Once the values for the SyncFlash mode register have been determined, these values can be inserted into Table 3 to calculate the proper address by which the MC9328MX1 would need to access after issuing the LOAD MODE REGISTER command. Section 3.2.2, "Mode Register Value," provides an example on how to program the SyncFlash mode register.

3 Erasing and Programming the SyncFlash

This section provides the procedure for erasing and programming the SyncFlash by first providing an overview of the MC9328MX1 SDRAM Control Register. Details are also provided on how to program the SyncFlash by describing the SyncFlash initialization procedure, what value to program into the mode register, how to erase and program the nvmode register, and how to erase and program the SyncFlash memory.

3.1 MC9328MX1 SDRAM Control Register Overview

In the MC9328MX1 SDRAM Controller there are two SDRAM control registers, one for each of the two memory arrays. SDCTL0 defines the operating characteristics for the SDRAM 0 region (selected by $\overline{CSD0}$ which is muxed with $\overline{CS2}$), while SDCTL1 defines the operating characteristics for the SDRAM 1 region (selected by $\overline{CSD1}$ which is muxed with $\overline{CS3}$). Bit field assignments within the registers are identical. Table 5 shows the SDRAM control register.

Table 5. SDRAM Control Register (SDCTL0 and SDCTL1)

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SDE	SMODE			SP		ROW				COL		IAM		DSIZ	
TYPE	rw	rw	rw	rw	rw	r	rw	rw	r	r	rw	rw	rw	r	rw	rw
RESET	0*	0	0	0	0	0	0	1	0	0	0	0	0	0	0*	0*
	0x0100*															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SREFR		CLKST		CI		SCL			SRP	SRCD			SRC		
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	r	rw	rw	rw
RESET	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
	0x0300															

For the example used in this note, the following settings are used. Assume maximum values for all settings to ensure proper operation. Refer to the *MC9328MX1 Reference Manual* for further details of the SDCTL0 and SDCLT1 bit definitions in the SDRAM.

- SDE (SDRAM Controller Enable), set to ‘1’ to enable.
- SMODE (SDRAM Controller Operation Mode), set to various modes depending on the desired operation. The following list describes these bit settings:
  - 000 = Normal Read/Write
  - 001 = Precharge Command
  - 010 = Auto-Refresh Command
  - 011 = Set Mode Register Command
  - 100, 101= Reserved
  - 110 = SyncFlash Load Command Register
  - 111 = SyncFlash Program Read/Write
- SP (Supervisor Protect), set to ‘0’
- ROW (Row Address Width), set to ‘01’ for the SyncFlash configuration of 12 row address bits
- COL (Column Address Width), set to ‘00’ for the SyncFlash configuration of 8 column address bits
- IAM (Bank Interleaved Address Mode), set to ‘0’ for linear bank addressing
- DSIZ (Data Width), set to ‘1x’ for 32-bit memory

- SREFR (SDRAM Refresh Rate), set to ‘00’ since SyncFlash memory does not require refresh
- CLKST (Clock Suspend Timeout), set to ‘00’ to disable
- CI (Cache Inhibit), set to ‘00’ to disable cache inhibit
- SCL (SDRAM CAS Latency), set to ‘11’ for CAS of 3 (ensure SyncFlash mode register is programmed with the same CAS latency).
- SRP (SDRAM Row Precharge Delay), set to ‘0’ for 3 clocks to be inserted between a precharge command and the next row activate
- SRCD (SDRAM Row to Column Delay), set to ‘00’ for 4 clocks to be inserted between a row activate command and a subsequent read or write command to the same bank
- SRC (SDRAM Row Cycle Delay), set to ‘000’ for 8 clocks to be inserted between a refresh and any subsequent refresh or read/write access

3.2 Programming the SyncFlash

This section describes the steps and procedures on how to initialize the SyncFlash, program the mode register, erase and program the nvmode register, and erase and program the SyncFlash memory.

3.2.1 SyncFlash Initialization

As of this publication date, the Micron SyncFlash data sheet states that the SyncFlash memory must be powered up and initialized in a pre-defined manner. To properly initialize the SyncFlash memory, power must first applied simultaneously to Vcc, VccQ, and VccP, the clock must be stable, and the RP# must be brought from low to high. The data sheet further states that a 100µs delay is required after RP# transitions high, to allow the SyncFlash memory to properly initialize. The following code listing shows the proper SyncFlash initialization. This example code listing also incorporates SyncFlash memory loading of the mode register.

Code Listing 1. Initializing the SyncFlash Memory

```
void SyncFlashInit(void)
{
              U32 tmp;

              /* Turn on CSD1 (set SDE) for negating RESETSF of SyncFLash */
              *(P_U32)SDCTL1|=0x80000000;

              /* Now call a simple DELAY routine to meet the 100us rule. Note, this
number */
              /* can be adjusted depending on the processor speed, or a more elaborate
*/
              /* delay routine can be written */
              Delay(1000);

              /* Set Load Mode Register Command in SDCTL1 */
              *(P_U32)SDCTL1=CMD_LMR; //CMD_LMR = 0xB1020300
              /* Perform a READ access to output the mode contents to the addr bus */
              tmp = (*(P_U32)(MODE_REG_VAL)); //MODE_REG_VAL = 0x0C08CC00
}

void Delay(volatile unsigned int Time)
{
              while(Time--);
}
```

3.2.2 Mode Register Value

Section 2.3, “Mode Register Programming,” described the concept and steps needed to program the SyncFlash memory mode register. To interface the two (2) MT28S4M16LC devices, the following parameters are used:

- Write Burst Mode, WB = 1 (Single Location)
- CAS Latency = ‘011’ (CAS = 3)
- Burst Type, BT = 0 (Sequential)
- Burst Length = ‘011’ (8 words to match the MC9328MX1 ARM920T™ cache line fill)

Inserting these values into Table 2 allows users to calculate the following values.

Table 6. SyncFlash Mode Register with Values

SyncFlash Address	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bits	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Contents	Reserved		WB	Op Mode		CAS Latency			BT	Burst Length		
Value	0	0	1	0	0	0	1	1	0	0	1	1

The next step is to take these values and convert them into an address to present to the SyncFlash memory following the LOAD MODE REGISTER command by inserting these values into Table 3. Table shows this example.

Table 7. MC9328MX1 Address Calculation for Given Mode Register Value

MX1 Internal Address	A'31	A'30	A'29	A'28	A'27	A'26	A'25	A'24	A'23	A'22	A'21	A'20	A'19	A'18	A'17	A'16	A'15	A'14	A'13	A'12	A'11	A'10	A'9	A'8	A'7	A'6	A'5	A'4	A'3	A'2	A'1	A'0
SyncFlash Address/Mode Register Bits	x	x	x	x	CS base	CS base	x	x	BA1	BA0	A11/M11	A10/M10	A9/M9	A8/M8	A/M7	A6/M6	A5/M5	A4/M4	A3/M3	A2/M2	A1/M1	A0/M0	x	x	x	x	x	x	x	x	x	x
Mode Register Bit values	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0

Table assumes the use of $\overline{CS3}$ ($\overline{CSD1}$), so the chip-select base address bits A'27 and A'26 are set to ‘11’ for the memory map region 0x0C000000. In this example, the final value (in hexadecimal format) that is written to the MC9328MX1 internal address for proper translation to the SDRAM memory mode register is 0x0C08CC00. Next, issue a LOAD MODE REGISTER Command to the SyncFlash memory, followed by an access (either READ or WRITE) to the SDRAM memory at address 0x0C08CC00. This value will be written into the MODE_REG_VAL variable of Code Listing 1.

3.2.3 NVMODE Register Erase and Program

After the mode register has been programmed, its contents may be copied in the nvmode register. Section 2.3.1, “Mode Register Overview,” only mentioned the concept of the nvmode register. After proper programming of the nvmode register, the contents of the nvmode register are loaded into the mode register during the SyncFlash initialization. This allows the device to be powered up in the desired programmed state. This section describes how to erase and program the nvmode register.

The nvmode register must first be erased before being programmed. Both the erase and programming of the nvmode register, users must follow the proper command sequence. All erase, program, and configure commands issued to the SyncFlash memory must follow a specific three command sequence:

1. LCR
2. ACTIVE
3. READ or WRITE

The Load Command Register command, or LCR, is basically the same command as the SDRAM Auto Refresh command and it indicates to the SyncFlash that the following READ or WRITE command has special meaning. It is imperative that the three command sequence occurs in this order and that no other commands, except for NOPs, be issued during this command sequence. At the time of publication of this app note, it was found that without special consideration of closing the specific row being written to, the MC9328MX1 may inadvertently issue a PRECHARGE command during this command sequence. The following code listing takes this into account and ensures that the row is closed by reading the row then issuing a PRECHARGE command. However, future revisions of the MC9328MX1 may correct this effect such that the special row access, then PRECHARGE command, may not be needed for the LCR three command sequence. Moreover, newer revisions of the Micron SyncFlash will treat any PRECHARGE command in the command sequence as NOPs, thus allowing any device to truly treat the SyncFlash as SDRAM when issuing commands. Until this hardware is implemented, it is strongly suggested to follow the code listings provided in this document. Refer to the *MC9328MX1 Reference Manual* and the Micron SyncFlash data sheet for more details on the LCR three command sequence.

To erase the SyncFlash nvmode register, the MC9328MX1 must first issue the LCR command by writing the value ‘110’ to the SMODE bits of the SDCTL1 register followed by an access to the SyncFlash base address 0x0C000000 plus an offset indicating the input command being executed. In the case of the nvmode register erase, the command is 0x30, which after translating this through the MC9328MX1 SDRAM Controller mux and writing this to the row bits, the address written by the MC9328MX1 is actually 0xC000. Refer to Table 1 for more details on the address muxing between the internal MC9328MX1 AHB bus and the external address bus. The following code shows the nvmode register erase sequence.

Code Listing 2. NVMODE Register Erase Function

```
void SyncFlashNvmodeErase(void)
{
            U32 tmp, i;

            /* Perform a read and precharge of the bank before the LCR|ACT|WRIT */
            /* sequence to avoid the inadvertent precharge command occurring during
*/
            /* the LCR|ACT|WRIT sequence */
            /* Set the SyncFlash to Normal Mode */
            *(P_U32)SDCTL=CMD_NORMAL;

            /* Access (read) the row before issuing the precharge command */
            i = (*(P_U32)SYNCFLASH_BASE);

            /* Set Precharge Command in SDCTL1 */
            *(P_U32)SDCTL=CMD_PREC;
            /* Issue Precharge without A10 being high */
            tmp=(*(P_U32)(SYNCFLASH_BASE));

            /* LCR|ACT|WRIT sequence */
            /* Write the LCR command (110) to the SMODE bits of SDCTL1 */
            *(P_U32)SDCTL=CMD_LCR; // CMD_LCR='110'
            /* Now issue the nvmode erase command to the SyncFlash, 0x30 */
            /* Thus, LCR_ERASE_NVMODE=0xC000 */
            *(P_U32)(SYNCFLASH_BASE+LCR_ERASE_NVMODE)=0;

            /* Return to Normal Mode to issue the erase confirm */
            *(P_U32)SDCTL=CMD_NORMAL; // CMD_NORMAL='000'
            /* Now write the confirmation to the SyncFlash completing the
LCD|ACT|WRIT*/
            *(P_U32)(SYNCFLASH_BASE)=0xC0C0C0C0; // Confirm

            /* Finally, wait for the erase to take by polling the SF Status Register
*/
            while(!SyncFlashReady());
}
```

After the nvmode register has been erased, program the nvmode register. Programming the nvmode register follows the same LCR three command sequence as erasing, except that the nvmode register program command is given instead of the erase command. The nvmode register program command is 0xA0, which after going through the address translation given in Table 1, is determined to be 0x28000. Programming or writing to the nvmode register takes the current contents of the mode register and programs it into the nvmode register. It is important to properly program the mode register with the correct contents. Code Listing 1 provided an example of how to properly program the mode register.
Code Listing 3 shows the nvmode register programming sequence.

Code Listing 3. NVMODE Register Program Sequence

```
void SyncFlashNvmodeWrite(void)
{
            U32 tmp, i;

            /* Perform a read and precharge of the bank before the LCR|ACT|WRIT */
            /* sequence to avoid the inadvertent precharge command occurring during
*/
            /* the LCR|ACT|WRIT sequence */
            /* Set the SyncFlash to Normal Mode */
            *(P_U32)SDCTL=CMD_NORMAL;

            /* Access (read) the row before issuing the precharge command */
            i = (*(P_U32)SYNCFLASH_BASE);

            /* Set Precharge Command in SDCTL1 */
            *(P_U32)SDCTL=CMD_PREC;
            /* Issue Precharge without A10 being high */
            tmp=(*(P_U32)(SYNCFLASH_BASE));

            /* LCR|ACT|WRIT sequence */
            /* Write the LCR command (110) to the SMODE bits of SDCTL1 */
            *(P_U32)SDCTL=CMD_LCR; // CMD_LCR='110'
            /* Now issue the nvmode erase command to the SyncFlash, 0xA0 */
            /* Thus, LCR_PROG_NVMODE=0x28000 */
            *(P_U32)(SYNCFLASH_BASE+LCR_PROG_NVMODE)=0;

            /* Return to Normal Mode to issue the program confirm */
            *(P_U32)SDCTL=CMD_NORMAL; // CMD_NORMAL='000'
            /* Now write to the SyncFlash to complete the LCR|ACT|WRIT sequence */
            *(P_U32)(SYNCFLASH_BASE)=0xC0C0C0C0; // Confirm not needed, can be any
data

            /* Finally, wait for the erase to take by polling the SF Status Register
*/
            while(!SyncFlashReady());
}
```

The functions called in Code Listing 3 are found at the end of this note in Section 3.3, “Entire Code Listing.”

As stated before, the non-volatile register is copied into the mode register automatically during device initialization and does not require reloading prior to the first operational command. This is especially useful since the MC9328MX1 SDRAM controller is designed to permit booting from the SyncFlash device immediately out of reset. The requirement is that the SyncFlash device must be connected to $\overline{\text{CSD1}}$. Also, the BOOT [3:0] pins of the MC9328MX1 must be configured to allow instruction fetches to occur from the $\overline{\text{CSD1}}$ memory space upon system reset (configure to ‘0001’ for a x16 interface on D[15:0] or configure to ‘0010’ for a x32 interface). Refer to the *MC9328MX1 Reference Manual* for more information on the operational system boot mode of the MC9328MX1 upon system reset and how it is determined by the configuration of the four external input pins BOOT [3:0].

3.2.4 SyncFlash Erase and Program

After the nvmode register has been properly erased and programmed, the finals steps are to erase the contents of the SyncFlash, then program the SyncFlash with the desired data.

Erasing the SyncFlash is similar to erasing the nvmode register, in that they each follow the same three command sequence:

1. LCR
2. ACT

3. WRITE

The Micron 4M x 16 SyncFlash is a four-bank architecture with four erasable blocks per bank. When issuing the erase command sequence along with the address to be erased, the entire block that the address is contained in will summarily be erased. In the 4M x 16 SyncFlash architecture, each block size is 256K x 16, thus in a 4M x 16 x 2 configuration, the block size is 256K x 32. This results in block addresses issued from the MC9328MX1 internal AHB bus as shown in the Table 8.

Table 8. MC9328MX1 Block Addresses Issued to the SyncFlash

	SyncFlash Bank 0	SyncFlash Bank 1	SyncFlash Bank 2	SyncFlash Bank 3
MC9328MX1 Internal AHB Block Addresses	0x00000000 Block 0	0x00400000 Block 4	0x00800000 Block 8	0x00C00000 Block 12
	0x00100000 Block 1	0x00500000 Block 5	0x00900000 Block 9	0x00D00000 Block 13
	0x00200000 Block 2	0x00600000 Block 6	0x00A00000 Block 10	0x00E00000 Block 14
	0x00300000 Block 3	0x00700000 Block 7	0x00B00000 Block 11	0x00F00000 Block 15

Table 8 shows the block address issued from the MC9328MX11 internal AHB address bus that will properly translate into block addresses to the SyncFlash memory device. Again, Table 1 on page 5 should help users understand the address translation shown in Table 8.

Code Listing 4 provide users an example of how to erase the entire SyncFlash, one block at a time, while Code Listing 5 shows the actual SyncFlash erase routine.

Code Listing 4. SyncFlash Erase All Routine

```
void SyncFlashEraseAll(void)
{
            /* Place the SyncFlash in Normal Mode */
            SyncFlashNormal();
            SyncFlashErase(SYNCFLASH_BASE+0x0000000);
            SyncFlashErase(SYNCFLASH_BASE+0x0100000);
            SyncFlashErase(SYNCFLASH_BASE+0x0200000);
            SyncFlashErase(SYNCFLASH_BASE+0x0300000);
            SyncFlashErase(SYNCFLASH_BASE+0x0400000);
            SyncFlashErase(SYNCFLASH_BASE+0x0500000);
            SyncFlashErase(SYNCFLASH_BASE+0x0600000);
            SyncFlashErase(SYNCFLASH_BASE+0x0700000);
            SyncFlashErase(SYNCFLASH_BASE+0x0800000);
            SyncFlashErase(SYNCFLASH_BASE+0x0900000);
            SyncFlashErase(SYNCFLASH_BASE+0x0A00000);
            SyncFlashErase(SYNCFLASH_BASE+0x0B00000);
            SyncFlashErase(SYNCFLASH_BASE+0x0C00000);
            SyncFlashErase(SYNCFLASH_BASE+0x0D00000);
            SyncFlashErase(SYNCFLASH_BASE+0x0E00000);
            SyncFlashErase(SYNCFLASH_BASE+0x0F00000);

            /* Place the SyncFlash Back into Normal Mode */
            SyncFlashNormal();

}
```

Code Listing 5. SyncFlash Erase Routine

```
void SyncFlashErase(U32 RowAddress)
{
            U32 i;
            U32 tmp;

            /* Perform a read and precharge of the bank before the LCR|ACT|WRIT */
            /* sequence to avoid the inadvertent precharge command occurring during
*/
            /* the LCR|ACT|WRIT sequence */
            /* Set the SyncFlash to Normal Mode */
            *(P_U32)SDCTL=CMD_NORMAL;

            /* Access (read) the row before issuing the precharge command */
            i = (*(P_U32)RowAddress);

            /* precharge only one bank before the LCR|ACT|WRIT sequence */
            *(P_U32)SDCTL=CMD_PREC; // Set Precharge Command
            tmp=(*(P_U32)(RowAddress)); // A10 is not explicitly high

            /* LCR|ACT|WRIT sequence */
            /* Write the LCR command (110) to the SMODE bits of SDCTL1 */
            *(P_U32)SDCTL=CMD_LCR;
            /* Now issue the erase setup command to the SyncFlash, 0x20 */
            /* Thus, LCR_ERASE_CONFIRM=0x8000 */
            *(P_U32)(RowAddress+LCR_ERASE_CONFIRM)=0;

            /* Return to Normal Mode to issue the erase confirm */
            *(P_U32)SDCTL=CMD_NORMAL;
            /* Now write the confirmation to the SyncFlash completing the
LCD|ACT|WRIT*/
            *(P_U32)(RowAddress)=0xD0D0D0D0;

            /* Finally, wait for the erase to take by polling the SF Status Register
*/
            while(!SyncFlashReady());
}
```

Once the SyncFlash has been erased, it can be programmed with the desired data. The programming sequence of the SyncFlash follows the same command sequence already shown:

1. LCR
2. ACT
3. WRITE

This time, when the data is written to the SyncFlash memory, the column address is presented on the address bus as each data word is programmed into the SyncFlash. Also, the MC9328MX1 SDRAM Controller has a special mode for programming the SyncFlash (SyncFlash Program Read/Write). To enter this mode, write the value ‘111’ to the SDCTL1[SMODE] bits. After entering this mode, write the data to the desired SyncFlash memory location, and the LCR|ACT|WRIT sequence is taken care of automatically by the MC9328MX1 SDRAM Controller as it will issue the SyncFlash Program Setup/Program command (0x40) followed by the data write to the desired memory location. Code Listing 6 on page 16 shows the SyncFlash programming sequence. This SyncFlash Program Read/Write mode (SMODE = ‘111’) can also be used to read the SyncFlash status register, which follows the LCR three command sequence:

1. LCR
2. ACT
3. READ

Perform a read of any SyncFlash address, and the MC9328MX1 will automatically issue the SyncFlash Read Status Register command (0x70), followed by a read of the status register. An example of this is shown in the function definition “SyncFlashSR(void)” located in Code Listing 7 in Section 3.3, “Entire Code Listing.”

Code Listing 6. SyncFlash Programming Routine

```
void SyncFlashWrite(U32 SourceAddress, U32 TargetAddress, U32 ByteSize)
{
              U32 i;

              for(i=0; i<ByteSize; i+=4)
              {
                     SyncFlashPrechargeAll();

                     /* Enter the SyncFlash programming mode by writing */
                     /* SDCTL1[SMODE]='111' */
                     *(P_U32)SDCTL=CMD_PROGRAM;

                     /* Now write the desired data to the SyncFlash memory */
                     *(P_U32)(TargetAddress+i)=*(P_U32)(SourceAddress+i);

                     /* Poll status register and wait for program to take */
                     while(!SyncFlashReady());

                     /* Return SyncFlash to Normal Mode */
                     SyncFlashNormal();
              }
}
```

3.3 Entire Code Listing

The code provided in this note was built using the ARM® Developer Suite™ software and Metrowerks IDE and was executed on the MC9328MX1 ADS v0.1 board provided by Freescale. The code provided is broken up into four pieces.

1. The first code listing represents the SyncFlash_4Mx16x2_IAM0_CSD1.c file which contains all the function definitions needed to support the erasing and programming of the Micron SyncFlash using the MC9328MX1.
2. The second code listing is the associated header file called stddefs.h which contains the standard definitions and function prototypes to support the code.
3. The third code listing (main.c) provides an example of how to use the function definitions given in SyncFlash_4Mx16x2_IAM0_CSD1.c to erase and program the SyncFlash. In this file, a known pattern of data is loaded into the SDRAM at CS2 which is then later programmed into the SyncFlash. Thus, after programming the SyncFlash, the code compares the data in the SyncFlash with the original data in the SDRAM and verifies whether the SyncFlash has been properly programmed.
4. The fourth code listing provides a simple init.s file which follows the templates of the init.s file provided with the ARM Developer Suite software. Following the code listing is an initialization script used by the ARM Extended Debugger (AXD) to properly initialize the MC9328MX1 prior to code download (in the AXD, go to Options, Configure Interface, Session File, then browse and choose “DBMX1_ADSv01_init.txt“). This code requires that the SDRAM memory is located on $\overline{CS2}$ (or $\overline{CDS0}$) and the SyncFlash is located on $\overline{CS3}$ (or $\overline{CSD1}$).

Code Listing 7. SyncFlash_4Mx16x2_IAM0_CSD1.c

```
/*****************************************************************************
*
* COPYRIGHT (C) 2002,  INC. ALL RIGHTS RESERVED
*
*
*****************************************************************************
*
* FILE NAME:    SyncFlash_4Mx16x2_IAM0_CSD1.c
*
* DESCRIPTION:
*            This code provides all the function definitions needed to support the
*            erasing and programming of Micron's SyncFlash using the MC9328MX1.
*
* CAUTIONS:
*            It's highly recommended that the user follow the procedure given in each
*            function definition.  Specifically, all functions requiring the LCR
*            command which activates the LCR|ACT|WRIT sequence or LCR|ACT|READ
sequence
*            to the SyncFlash device.  For proper operation, before issuing the
*            LCR sequence, a "read" of the page followed by a precharge command to
close
*            the bank must be issued before the LCR|ACT|WRIT or LCR|ACT|READ sequence
to
*            avoid an inadvertent precharge command in the middle of the sequence,
thus
*            avoiding these incorrect sequences: LCR|PRE|ACT|WRIT or
LCR|PRE|ACT|READ.
*            This code has been written to avoid these incorrect sequence.
*
*
*****************************************************************************
*
* DISCLAIMER:
*   Freescale reserves the right to make changes without further notice to any
*   product herein to improve reliability, function, or design. Freescale does
*   not assume any liability arising out of the application or use of any
*   product, circuit, or software described herein; neither does it convey any
*   license under its patent rights nor the rights of others. Freescale
*   products are not designed, intended, or authorized for use as components
*   in systems intended for surgical implant into the body, or other
*   applications intended to support life, or for any other application in
*   which the failure of the Freescale product could create a situation where
*   personal injury or death may occur. Should Buyer purchase or use Freescale
*   products for any such intended or unauthorized application, Buyer shall
*   indemnify and hold Freescale and its officers, employees, subsidiaries,
*   affiliates, and distributors harmless against all claims, costs, damages,
*   and expenses, and reasonable attorney fees arising out of, directly or
*   indirectly, any claim of personal injury or death associated with such
*   unintended or unauthorized use, even if such claim alleges that Freescale
*   was negligent regarding the design or manufacture of the part. Freescale
*   and the Freescale Logo are registered trademarks of Freescale Semiconductor Inc.
*
*****************************************************************************/

/*****************************************************************************
* I N C L U D E   F I L E S
*****************************************************************************/
#include "stddefs.h"

/*****************************************************************************
* L O C A L   D E F I N E S
*****************************************************************************/
/* 4Mx16x2 IAM=0 CSD1 */
#define SDCTL0     0x221000
```

```
#define SDCTL1      (SDCTL0+4)

/* Following is the Setting for CSD1 */
#define SDCTL       SDCTL1
#define SYNCFLASH_BASE (0x0C000000)

/* This reflects the A10 address line of the SyncFlash */
#define SYNCFLASH_A10  (0x00100000)

/* SDCTL1 SMODE settings */
#define CMD_NORMAL     (0x81020300)             /* Normal Mode                */
#define CMD_PREC       (CMD_NORMAL+0x10000000) /* Precharge Command          */
#define CMD_AUTO       (CMD_NORMAL+0x20000000) /* Auto Refresh Command       */
#define CMD_LMR        (CMD_NORMAL+0x30000000) /* Load Mode Register Command */
#define CMD_LCR        (CMD_NORMAL+0x60000000) /* LCR Command */
#define CMD_PROGRAM    (CMD_NORMAL+0x70000000) /* SyncFlash Program Mode */

/* Define the value to be programmed into the SyncFlash mode register  */
/* via the address bus */
#define MODE_REG_VAL   (SYNCFLASH_BASE+0x0008CC00) // Cas Latency 3

/* SyncFlash LCR Commands adjusted for the DBMX1 AHB internal address bus */
#define LCR_READSTATUS    (0x0001C000) // 0x70
#define LCR_ERASE_CONFIRM (0x00008000) // 0x20
#define LCR_ERASE_NVMODE  (0x0000C000) // 0x30
#define LCR_PROG_NVMODE   (0x00028000) // 0xA0
#define LCR_SR_CLEAR      (0x00014000)  //0x50

/*****************************************************************************
* F U N C T I O N S
*****************************************************************************/


__swi(0x123456)

void _Write0(unsigned int op, const char *string);

#define Write0(string) _Write0(0x4, string)



void Delay(U32 Time)
{
            while(Time--);
}


void SyncFlashSRClear(void)
{
            U32 tmp;

            /* LCR|ACT|WRIT sequence */
            /* Activate the LCR Mode in SDCTL1 (SMODE = 110) */
            *(P_U32)SDCTL=CMD_LCR;
            /* issue the Clear Status Register command */
            tmp=*(P_U32)(SYNCFLASH_BASE+LCR_SR_CLEAR);
}

U32 SyncFlashSR(void)
{
            U32 tmp;

            /* Enter the SyncFlash Program READ/WRITE mode by writing */
            /* SDCTL1[SMODE]='111' */
            *(P_U32)SDCTL=CMD_PROGRAM;
```

```
		/* Read SR of SYNC Flash */
		tmp=(*(P_U32)SYNCFLASH_BASE);

		/* Return to Normal Mode */
		*(P_U32)SDCTL=CMD_NORMAL;

		/* Clear the SyncFlash Status Register */
		SyncFlashSRClear();

		return tmp;
}

U32 SyncFlashReady(void)
{
		U32 tmp;

		/* read the SyncFlash status register */
		tmp=SyncFlashSR();

		/* check the upper 16-bit SyncFlash Status register for errors */
		if( (tmp&0x00800000) && (tmp&0x00380000) )
		{
			Write0 ("SyncFlashError, check status register! \n");
		}

		/* check the lower 16-bit SyncFlash Status register for errors  */
		if( (tmp&0x00000080) && (tmp&0x00000038) )
		{
			Write0 ("SyncFlashError, check status register! \n");
		}

		/* Test Bit 7 of SR */
		if(tmp==0x00800080)
			return 1;
		else
			return 0;
}

void SyncFlashPrechargeAll(void)
{
		U32 tmp;

		/* Set Precharge Command (001) in SDCTL1 */
		*(P_U32)SDCTL=CMD_PREC;
		/* Issue Precharge All command by accessing the SyncFlash memory */
		tmp=(*(P_U32)(SYNCFLASH_BASE+SYNCFLASH_A10));

}

void SyncFlashNormal(void)
{
		/* Call the Precharge All function */
		SyncFlashPrechargeAll();

		/* Set SMODE bits to 000  in the SDCTL1 */
		/* to place the SDRAM controller in Normal Mode */
		*(P_U32)SDCTL=CMD_NORMAL;
}

void SyncFlashInit(void)
{

		U32 tmp;

		/* Turn on CSD1 (set SDE) for negating RESETSF of SyncFLash */
		*(P_U32)SDCTL1|=0x80000000;
```

```
                /* Now call a simple DELAY routine to meet the 100us rule. Note, this
number */
                /* can be adjusted depending on the processor speed, or a more elaborate
*/
                /* delay routine can be written */
                Delay(1000);

                /* Set Load Mode Register Command in SDCTL1 */
                *(P_U32)SDCTL1=CMD_LMR; //CMD_LMR = 0xB1020300
                /* Perform a READ access to output the mode contents to the addr bus */
                tmp = (*(P_U32)(MODE_REG_VAL)); //MODE_REG_VAL = 0x0C08CC00
}

void SyncFlashWrite(U32 SourceAddress, U32 TargetAddress, U32 ByteSize)
{
                U32 i;

                for(i=0; i<ByteSize; i+=4)
                {
                        SyncFlashPrechargeAll();

                        /* Enter the SyncFlash programming mode by writing */
                        /* SDCTL1[SMODE]='111' */
                        *(P_U32)SDCTL=CMD_PROGRAM;

                        /* Now write the desired data to the SyncFlash memory */
                        *(P_U32)(TargetAddress+i)=*(P_U32)(SourceAddress+i);

                        /* Poll status register and wait for program to take */
                        while(!SyncFlashReady());

                        /* Return SyncFlash to Normal Mode */
                        SyncFlashNormal();
                }
}

U32 Compare(U32 SourceAddress, U32 TargetAddress, U32 ByteSize)
{
                U32 i;
                U32 _error=0;

                /* check each word in the SyncFlash to see if it matches the source word
*/
                for(i=0; i<ByteSize; i++)
                if(*(P_U32)(SourceAddress+i) != *(P_U32)(TargetAddress+i) )
                {
                        _error=1;
                }

                if(_error)
                        return 1;

                return 0;
}


void SyncFlashErase(U32 RowAddress)
{
                U32 i;
                U32 tmp;

                /* Perform a read and precharge of the bank before the LCR|ACT|WRIT */
                /* sequence to avoid the inadvertent precharge command occurring during
*/
                /* the LCR|ACT|WRIT sequence */
```

```
                /* Set the SyncFlash to Normal Mode */
                *(P_U32)SDCTL=CMD_NORMAL;

                /* Access (read) the row before issuing the precharge command */
                i = (*(P_U32)RowAddress);

                /* precharge only one bank before the LCR|ACT|WRIT sequence */
                *(P_U32)SDCTL=CMD_PREC; // Set Precharge Command
                tmp=(*(P_U32)(RowAddress)); // A10 is not explicitly high

                /* LCR|ACT|WRIT sequence */
                /* Write the LCR command (110) to the SMODE bits of SDCTL1 */
                *(P_U32)SDCTL=CMD_LCR;
                /* Now issue the erase setup command to the SyncFlash, 0x20 */
                /* Thus, LCR_ERASE_CONFIRM=0x8000 */
                *(P_U32)(RowAddress+LCR_ERASE_CONFIRM)=0;

                /* Return to Normal Mode to issue the erase confirm */
                *(P_U32)SDCTL=CMD_NORMAL;
                /* Now write the confirmation to the SyncFlash completing the
LCD|ACT|WRIT*/
                *(P_U32)(RowAddress)=0xD0D0D0D0;

                /* Finally, wait for the erase to take by polling the SF Status Register
*/
                while(!SyncFlashReady());
}

void SyncFlashEraseAll(void)
{
                /* Place the SyncFlash in Normal Mode */
                SyncFlashNormal();
                SyncFlashErase(SYNCFLASH_BASE+0x0000000);
                SyncFlashErase(SYNCFLASH_BASE+0x0100000);
                SyncFlashErase(SYNCFLASH_BASE+0x0200000);
                SyncFlashErase(SYNCFLASH_BASE+0x0300000);
                SyncFlashErase(SYNCFLASH_BASE+0x0400000);
                SyncFlashErase(SYNCFLASH_BASE+0x0500000);
                SyncFlashErase(SYNCFLASH_BASE+0x0600000);
                SyncFlashErase(SYNCFLASH_BASE+0x0700000);
                SyncFlashErase(SYNCFLASH_BASE+0x0800000);
                SyncFlashErase(SYNCFLASH_BASE+0x0900000);
                SyncFlashErase(SYNCFLASH_BASE+0x0A00000);
                SyncFlashErase(SYNCFLASH_BASE+0x0B00000);
                SyncFlashErase(SYNCFLASH_BASE+0x0C00000);
                SyncFlashErase(SYNCFLASH_BASE+0x0D00000);
                SyncFlashErase(SYNCFLASH_BASE+0x0E00000);
                SyncFlashErase(SYNCFLASH_BASE+0x0F00000);

                /* Place the SyncFlash Back into Normal Mode */
                SyncFlashNormal();

}

void SyncFlashNvmodeErase(void)
{
                U32 tmp, i;

                /* Perform a read and precharge of the bank before the LCR|ACT|WRIT */
                /* sequence to avoid the inadvertent precharge command occurring during
*/
                /* the LCR|ACT|WRIT sequence */
                /* Set the SyncFlash to Normal Mode */
                *(P_U32)SDCTL=CMD_NORMAL;

                /* Access (read) the row before issuing the precharge command */
```

```
                i = (*(P_U32)SYNCFLASH_BASE);

                /* Set Precharge Command in SDCTL1 */
                *(P_U32)SDCTL=CMD_PREC;
                /* Issue Precharge without A10 being high */
                tmp=(*(P_U32)(SYNCFLASH_BASE));

                /* LCR|ACT|WRIT sequence */
                /* Write the LCR command (110) to the SMODE bits of SDCTL1 */
                *(P_U32)SDCTL=CMD_LCR; // CMD_LCR='110'
                /* Now issue the nvmode erase command to the SyncFlash, 0x30 */
                /* Thus, LCR_ERASE_NVMODE=0xC000 */
                *(P_U32)(SYNCFLASH_BASE+LCR_ERASE_NVMODE)=0;

                /* Return to Normal Mode to issue the erase confirm */
                *(P_U32)SDCTL=CMD_NORMAL; // CMD_NORMAL='000'
                /* Now write the confirmation to the SyncFlash completing the
LCD|ACT|WRIT*/
                *(P_U32)(SYNCFLASH_BASE)=0xC0C0C0C0; // Confirm

                /* Finally, wait for the erase to take by polling the SF Status Register
*/
                while(!SyncFlashReady());
}

void SyncFlashNvmodeWrite(void)
{
                U32 tmp, i;

                /* Perform a read and precharge of the bank before the LCR|ACT|WRIT */
                /* sequence to avoid the inadvertent precharge command occurring during
*/
                /* the LCR|ACT|WRIT sequence */
                /* Set the SyncFlash to Normal Mode */
                *(P_U32)SDCTL=CMD_NORMAL;

                /* Access (read) the row before issuing the precharge command */
                i = (*(P_U32)SYNCFLASH_BASE);

                /* Set Precharge Command in SDCTL1 */
                *(P_U32)SDCTL=CMD_PREC;
                /* Issue Precharge without A10 being high */
                tmp=(*(P_U32)(SYNCFLASH_BASE));

                /* LCR|ACT|WRIT sequence */
                /* Write the LCR command (110) to the SMODE bits of SDCTL1 */
                *(P_U32)SDCTL=CMD_LCR; // CMD_LCR='110'
                /* Now issue the nvmode erase command to the SyncFlash, 0xA0 */
                /* Thus, LCR_PROG_NVMODE=0x28000 */
                *(P_U32)(SYNCFLASH_BASE+LCR_PROG_NVMODE)=0;

                /* Return to Normal Mode to issue the program confirm */
                *(P_U32)SDCTL=CMD_NORMAL; // CMD_NORMAL='000'
                /* Now write to the SyncFlash to complete the LCR|ACT|WRIT sequence */
                *(P_U32)(SYNCFLASH_BASE)=0xC0C0C0C0; // Confirm not needed, can be any
data

                /* Finally, wait for the erase to take by polling the SF Status Register
*/
                while(!SyncFlashReady());
}
```

Code Listing 8. stddefs.h

```
/*****************************************************************************
*
* COPYRIGHT (C) 2002,  INC. ALL RIGHTS RESERVED
*
*****************************************************************************
*
* FILE NAME:    stddefs.h
*
* DESCRIPTION:
*            This is a header file for the SyncFlash erase and program routine.  It
*            provides standard definitions and function prototypes to support the
code.
*
*
*****************************************************************************/
#ifndef STDDEFS_INCLUDED
#define STDDEFS_INCLUDED

/*****************************************************************************
* D E F I N E S
*****************************************************************************/

/* Typedefs for integer types */
typedef volatile unsigned char U8; /*  unsigned 8 bit data  */
typedef volatile unsigned short U16; /*  unsigned 16 bit data */
typedef volatile unsigned int U32; /*unsigned 32 bit data */
typedef volatile char S8; /*  signed 8 bit data  */
typedef volatile short S16; /*  signed 16 bit data */
typedef volatile int S32; /*  signed 32 bit data */

typedef U8 * P_U8; /*  unsigned 8 bit data  */
typedef U16 * P_U16; /*  unsigned 16 bit data */
typedef U32 * P_U32; /*  unsigned 32 bit data */
typedef S8 * P_S8; /*  signed 8 bit data  */
typedef S16 * P_S16; /*  signed 16 bit data */
typedef S32 * P_S32; /*  signed 32 bit data */

typedef U16 TEXT; /* 16-bit text data */
typedef P_U16 P_TEXT; /* 16-bit text data */

typedef S16 STATUS; /* status word is a signed short */

typedef void * P_VOID; /* pointer to void */

/*****************************************************************************
* P R O T O T Y P E S
*****************************************************************************/

void Delay(U32 Time);
void SyncFlashSRClear(void);
U32 SyncFlashSR(void);
U32 SyncFlashReady(void);
void SyncFlashPrechargeAll(void);
void SyncFlashNormal(void);
void SyncFlashInit(void);
void SyncFlashWrite(U32 SourceAddress, U32 TargetAddress, U32 ByteSize);
U32 Compare(U32 SourceAddress, U32 TargetAddress, U32 ByteSize);
void SyncFlashErase(U32 RowAddress);
void SyncFlashEraseAll(void);
void SyncFlashNvmodeErase(void);
void SyncFlashNvmodeWrite(void);

#endif // STDDEFS_INCLUDED
```

Code Listing 9. main.c

```
/*****************************************************************************
*
* COPYRIGHT (C) 2002,  ALL RIGHTS RESERVED
*
*
*****************************************************************************
*
* FILE NAME:    main.c
*
* DESCRIPTION:
*            This purpose of this file is to provide an example of how to erase and
*            program Micron's SyncFlash using the MC9328MX1.  This code will make
*            function calls to SyncFlash_4Mx16x2_IAM0_CSD1.c demonstrating the
*            proper SyncFlash erase and program procedure.  Essentially, this code
will:
*            1. Place a known data pattern in the SDRAM
*            2. Program the SyncFlash Mode Register
*            3. Erase the nvmode register
*            4. Program the nvmode register
*            5. Erase the SyncFlash and verify the erase
*            6. Program the SyncFlash with known data and verify the data
*
*            This code has been written for Freescale's MC9328MX1 ADS v0.1 board
*            which contains:
*            16 Mbtyes of SyncFlash on CS3/CSD1
*            64 MBytes of SDRAM on CS2/CSD0
*
* CAUTIONS:
*            This code was built for the DBMX1 ADS board with the memory requirements
*            above. It's the responsibility of the user to port this code to their
*            system which may contain a different memory configuration.
*            Furthermore, this code is provided as mainly an example of how to use the
*            functions listed in SyncFlash_4Mx16x2_IAM0_CSD1.c
*
*****************************************************************************
*
* DISCLAIMER:
*   Freescale reserves the right to make changes without further notice to any
*   product herein to improve reliability, function, or design. Freescale does
*   not assume any liability arising out of the application or use of any
*   product, circuit, or software described herein; neither does it convey any
*   license under its patent rights nor the rights of others. Freescale
*   products are not designed, intended, or authorized for use as components
*   in systems intended for surgical implant into the body, or other
*   applications intended to support life, or for any other application in
*   which the failure of the Freescale product could create a situation where
*   personal injury or death may occur. Should Buyer purchase or use Freescale
*   products for any such intended or unauthorized application, Buyer shall
*   indemnify and hold Freescale and its officers, employees, subsidiaries,
*   affiliates, and distributors harmless against all claims, costs, damages,
*   and expenses, and reasonable attorney fees arising out of, directly or
*   indirectly, any claim of personal injury or death associated with such
*   unintended or unauthorized use, even if such claim alleges that Freescale
*   was negligent regarding the design or manufacture of the part. Freescale
*   and the Freescale Logo are registered trademarks of Freesale Semieconductor Inc.
*
*****************************************************************************/

/*****************************************************************************
* I N C L U D E   F I L E S
*****************************************************************************/
#include "stddefs.h"

/*****************************************************************************
```

```
* L O C A L   D E F I N E S
*******************************************************************************/
#ifndef SYNCFLASH_BASE
            #define SYNCFLASH_BASE (0x0C000000)
#endif

#define SDCTL0     0x221000
#define SDCTL1     (SDCTL0+4)

#define ONES_PATTERN 0x11111111

#define ERASE // uncomment to erase syncflash
#define PROGRAM // uncomment to program the syncflash
#define PUT_DATA_IN_SDRAM // uncomment to place data in SDRAM that
                                    // will be programmed into syncflash

/*******************************************************************************
* G L O B A L   V A R I A B L E S
*******************************************************************************/
U32 gFailCount;

/*******************************************************************************
* F U N C T I O N S
*******************************************************************************/
__swi(0x123456)

void _Write0(volatile unsigned int op, const char *string);

#define Write0(string) _Write0(0x4, string)

void Error(void)
{
            Write0("Error in Flash Programming!!!\n");

            return;
}


int main(void)
{
            U32 SourceAddress = 0x08800000;
            U32 TargetAddress = 0x0C000000;
            U32 ByteSize = 0x1000000; //16 MBytes
            U32 i;
            U32 pattern = 0;
            U32 *Pntr;

#ifdef PUT_DATA_IN_SDRAM
            /* fill the SDRAM memory with 32-bit data */
            /* SDRAM memory located at CS2/CSD0, 0x08000000 */
            for (i = 0; i < ByteSize; i = i+4)
            {
                  Pntr = (P_U32)(SourceAddress + i);
                  *Pntr = pattern;
                  pattern += ONES_PATTERN;
                  Pntr++;
            }

            pattern = 0;
            gFailCount = 0;

            /* validate the SDRAM contents */
            for (i = 0; i < ByteSize; i = i+4)
            {
                  Pntr = (P_U32)(SourceAddress + i);
                  if(*Pntr != pattern)
```

```
                {
                        gFailCount++;
                        Write0("SDRAM program failed.\n");
                }
                pattern += ONES_PATTERN;
                Pntr++;
        }

        if (gFailCount == 0)
        {
                Write0("SDRAM programmed correctly.\n");
        }

#endif

        Write0("Flashing in progress....\n");

        SyncFlashInit();

        // place SyncFlash into Normal mode
        SyncFlashNormal();

        Write0("Erasing and Programming the nvmode register.\n");

        SyncFlashNvmodeErase();
        SyncFlashNvmodeWrite();
        SyncFlashNormal();

#ifdef ERASE
        Write0("Erasing the SyncFlash.\n");

        SyncFlashEraseAll();
        Write0("Erase All completed.\n");

        SyncFlashNormal();

        /* initialize gFailCount variable */
        gFailCount = 0;

        /* verify erase took by checking to see if each word */
        /* in the SyncFlash equals 0xFFFFFFFF */
        for (i = 0; i < ByteSize; i = i+4)
        {
                Pntr = (P_U32)(TargetAddress + i);

                if (*Pntr != 0xFFFFFFFF)
                {
                        gFailCount++;
                }
        }

        if (gFailCount > 0)
        {
                Write0("Erase didn't take! \n");
        }
        else
        {
                Write0("Erase OK. \n");
        }

#endif

#ifdef PROGRAM
        Write0("\nBeginning Syncflash programming. \n");

        SyncFlashWrite(SourceAddress, TargetAddress, ByteSize);
```

```
            Write0("Flash operation completed.\n");

            SyncFlashNormal();
            Write0("Validating Data.....\n");

            if(Compare(SourceAddress, TargetAddress, ByteSize))
    {
            Write0("Flash programming not successful");
                  Error();
    }
    else
    {

                  Write0("Flash Successfully Completed.\n");
    }
#endif

            return 0;
}
```

Code Listing 10. init.s

```
            EXPORT __main
            EXPORT _main
            IMPORT main
            IMPORT |Image$$ZI$$Base|
            IMPORT |Image$$ZI$$Limit|


            AREA Init, CODE, READONLY

            CODE32
            ENTRY

            LDR sp, =0x320000

            LDR r0,=0x21B000 ;; Set FCLK to 192MHz, BCLK to 48MHz
            LDR r1,=0x0F008403
            STR r1,[r0,#0]

            mrc p15, 0, r1, c1, c0, 0
            orr r1,r1, #0xC0000000
            mcr p15, 0, r1, c1, c0, 0

_main
__main
            B main

            END
```

Code Listing 11. ARM Developer Suite AXD Initialization Script, "DBMX1_ADSv01_init.txt"

```
comment ### Init code for the DBMX1 ADS v0.1

comment ### Select CLKO mux to output HCLK(BCLK) ###
comment setmem 0x21B000 0x2F00AC03, 32

comment ### Change BCLK (CPUCLK) to 16MHz
comment setmem 0x21B000 0x2F009403, 32

comment ### Change BCLK (CPUCLK) to 24MHz
comment setmem 0x21B000 0x2F008C03, 32

comment ### Change BCLK (CPUCLK) to 32MHz
comment setmem 0x21B000 0x2F008803, 32

comment ### Change BCLK (CPUCLK) to 48MHz
setmem 0x21B000 0x2F008403, 32

comment ### Change BCLK (CPUCLK) to 96MHz
comment setmem 0x21B000 0x2F008003, 32

comment ### CS0 - SRAM on SODIMM, 10 wait states, 32-bit ###
setmem 0x220000 0x00000A00, 32
setmem 0x220004 0x11110601, 32

comment ### CS1 - SRAM, 10 wait states, 32-bit ###
setmem 0x220008 0x00000A00, 32
setmem 0x22000C 0x11110601, 32

comment ### CS4 - External UART, 10 wait states, 8-bit ###
setmem 0x220020 0x00000A00, 32
setmem 0x220024 0x11110301, 32


comment ### SDRAM Init, non-bank-interleaved mode
comment ### assumes 16Mx16x2 configuration
comment ### Set Precharge Command
setmem 0x221000 0x9212C200, 32
comment ### Issue Precharge all Command (assert SDRAM memory A10)
mem    0x08200000, +4 32
comment ### Set AutoRefresh Command
setmem 0x221000 0xA212C200, 32
comment ### Issue AutoRefresh Command
mem    0x08000000, +4 32
mem    0x08000000, +4 32
mem    0x08000000, +4 32
mem    0x08000000, +4 32
mem    0x08000000, +4 32
mem    0x08000000, +4 32
mem    0x08000000, +4 32
mem    0x08000000, +4 32
comment ### Set Mode Register
setmem 0x221000 0xB212C200, 32
comment ### Issue Mode Register Command, Burst length = 8, CAS = 2
mem    0x08111800, +4 32
comment ### Set to Normal Mode,row:13,col:9,IAM:0,SREFR:8192,CAS:2
setmem 0x221000 0x8212C200, 32
```

4 References

For the latest MC9328MX1 reference manual and information, go to:

http://e-www.freescale.com/

Then search for MC9328MX1.

For the latest Micron SyncFlash data sheet and other technical information, go to:

http://www.syncflash.com

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
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™ semiconductor

AN2284/D