Freescale Semiconductor
Application Note

Document Number: AN3854
Rev. 1, 09/2009

Using the Mini-FlexBus External Bus Interface for ColdFire® Microcontrollers

by: Maclain Lobdell
Austin, Texas

Freescale's ColdFire® microcontrollers are available in a wide range of small form factors with rich on-chip features, such as large Flash and SRAM memory blocks. In most cases, on-chip memory provides ample room for the application, with the option to expand functionality with simple, serially-interfaced memories and/or peripherals. In some cases, however, serially-interfaced devices cannot provide the performance or memory capacity needed.

Freescale is proud to introduce the new Mini-FlexBus External Bus Interface (EBI). The Mini-FlexBus is a scaled-down version of the FlexBus that is found on other ColdFire devices.

The Mini-FlexBus is a simple EBI for microcontrollers that typically use internal memory, but need expandability. The bus interface minimizes package pinouts, while maintaining a high level of configurability and functionality.

Contents

freescale™ semiconductor

The MCF5225x family of ColdFire microcontrollers is the first to include the Mini-FlexBus, and is the focus of this application note. This application note describes how to set up and use the Mini-FlexBus on the MCF5225x and shows some implementation examples. The M52259EVB evaluation board takes advantage of one of these examples and can be used as a reference. Although the focus of the application note is the MCF5225x family, the information contained herein will also apply to future devices with the Mini-FlexBus.

1 This Application Note

1.1 What Is Covered

- Overview of features, important details, and clarifications.
- Typical examples of how to connect to asynchronous SRAM and MRAM.
- Block diagrams, PCB layout drawing examples, software setup, throughput calculations.
- Timing Explanation.

1.2 What Is not Covered

- Connecting to other memory types or peripherals such as Flash, ROM, Digital-to-Analog Converters.
- Connecting to synchronous memory.
- Using memory sizes other than 512 KByte, such as 1 MByte, 256 KByte, 128 KByte.
- PCB Layout tips such as matching trace lengths, termination resistors.
- Power Usage Implications.

2 Introduction

2.1 Mini-FlexBus Overview

- MCF5225x Mini-FlexBus Features:
  - Two user-programmable chip selects ($\overline{\text{FB\_CS0}}$ & $\overline{\text{FB\_CS1}}$).
  - 8- or 16-bit port size.
  - Multiplexed (muxed) or non-multiplexed mode (non-muxed).
  - 20 Address, eight Data or 20 Address/Data Signals.
  - Byte-, word-, longword-, and 16-byte line-sized transfers.
  - Programmable wait states, address setup, and address hold times.
  - Output Enable ($\overline{\text{FB\_OE}}$) and Read/Write (FB_$\overline{\text{RW}}$) control signals.
  - Address Latch Enable (FB_ALE) control signal for multiplexed mode.

- Differences against FlexBus:
  - The FlexBus utilizes a larger signal count to address larger memory with wider port widths.
  - The FlexBus has more chip selects and additional control signals such as byte select, transfer size, transfer acknowledge.
  - Some of these features may be available in future Mini-FlexBus implementations.
- Non-multiplexed vs. multiplexed mode:
  - Non-multiplexed:
    - Uses dedicated address signals FB_A[19:0] and dedicated data signals FB_D[7:0].
    - The port width is eight bits wide in this mode.
  - Multiplexed mode:
    - Uses FB_A[19:0] as both address and data signals. Often the signal notation is changed to FB_AD[19:0] in this mode to indicate address/data. During the address phase of the transfer, the FB_AD[19:0] signals drive the address. Later during the data phase, the FB_AD[19:0] pins drive data.
    - The port width can be either eight bits or 16 bits wide in this mode. If it is eight bits wide, then FB_AD[7:0] drive data during the data phase. If it is 16 bits wide, then FB_AD[15:0] drive data during the data phase.

2.2 Important Details and Clarifications

- The MCF5225x Mini-FlexBus can address a maximum memory size of 1 MByte per chip select. This is independent of the mode or port size used. It is solely based on the twenty address signals available (FB_A[19:0]). Twenty address signals can address 2^20 bytes, regardless of whether two bytes are accessed at a time (16-bit port size) or one byte (8-bit port size).
- The FB_ALE and FB_CS1 signals share the same pin. If the multiplexed mode is used, the FB_ALE signal may be required, therefore limiting the number of available chip selects to one. The FB_ALE or FB_CS1 function is selected in the PASPAR port-assignment register.
- Booting from the Mini-FlexBus is not supported.
- The chip-select base addresses must be 64-KByte aligned. Valid base addresses are anywhere outside the register or memory address space.
- If memory is accessed outside the chip select address range (or other valid memory mapped address space) then an access error occurs.
- Overlapping a chip select address range over another memory-mapped region (such as internal memory, registers, or another chip-select range) causes access errors.
- Executing code over the Mini-FlexBus is supported.
- Due to the 8- or 16-bit port size of the Mini-FlexBus, the memory bandwidth is lower than the 32-bit internal memory of the MCF5225x.

- Typical memory mapping on MCF52259

Internal FLASH*	0x0000_0000 — 0x0007_FFFF
Open	0x0008_0000 — 0x1FFF_FFFF
Internal SRAM*	0x2000_0000 — 0x2000_FFFF
Open	0x2001_0000 — 0x3FFF_FFFF
Registers and Reserved Space*	0x4000_0000 — 0x7FFF_FFFF
Open	0x8000_0000 — 0xFFFF_FFFF

* — The location of FLASH depends on FLASHBAR, location of SRAM depends on RAMBAR, location of Registers depends on IPSBAR.

3 Examples

Small size and flexibility are keys to the Mini-FlexBus, and the examples that follow illustrate ways, in which some common interfaces can be accomplished. The following examples use the MCF5225x, which is the first MCU available with the Mini-FlexBus. Example 2 is implemented on the M52259EVB evaluation board.

On the MCF5225x, the Mini-FlexBus can run in 1:2 or 1:1 mode, meaning it can run at half the PLL system frequency (1:2 mode) or the full system frequency (1:1 mode). This is a specific implementation on the MCF5225x family and may be different on other devices. The Mini-FlexBus frequency is controlled by the Chip Configuration Extended Register (CCE) in the chip-configuration module. The default setting is 1:2 mode.

The examples use the default 1:2 mode, and assume a PLL system frequency of 80 Mhz. Therefore the Mini-FlexBus runs at 40 Mhz.

The examples are listed in order of complexity and the advantages and disadvantages of each are discussed.

List of Examples

- Connecting to an Industry Standard x8 Asynchronous Fast SRAM using no glue logic.
- Connecting to an Industry Standard x16 Asynchronous MRAM in byte mode using minimal glue logic.
- Connecting to an Industry Standard x16 Asynchronous Fast SRAM Optimized for Performance.

3.1 Example 1 — Connecting to an Industry Standard x8 Asynchronous Fast SRAM Using No Glue Logic

In this example, the Mini-FlexBus is connected to a single Industry Standard x8 Asynchronous Fast SRAM. It uses the non-multiplexed mode with 8-bit data and 20-bit address lines.

Advantages:

- It increases available system memory by adding a 512 KByte SRAM.
- It requires no glue logic.
- It has simple PCB layout on one signal layer.

Disadvantages:

- 8-bit memory provides less performance than 16-bit.

3.1.1 Schematics

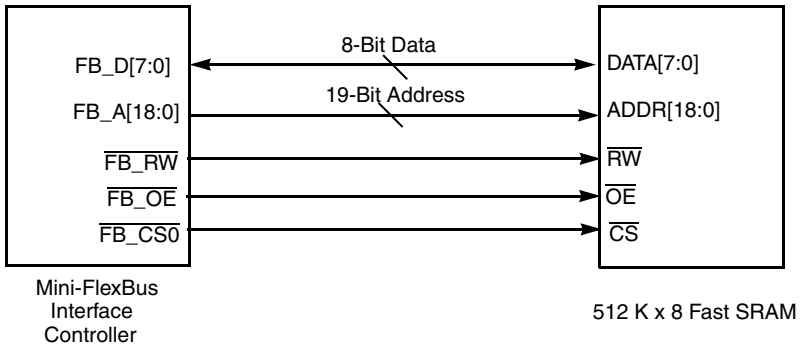

Figure 1. Non-Multiplexed x8 Mini-FlexBus to x8 Memory

3.1.2 PCB Layout

Figure 2 shows the ease, with which the MCF52259x can be connected with an industry standard x8 asynchronous fast SRAM on a single signal layer.

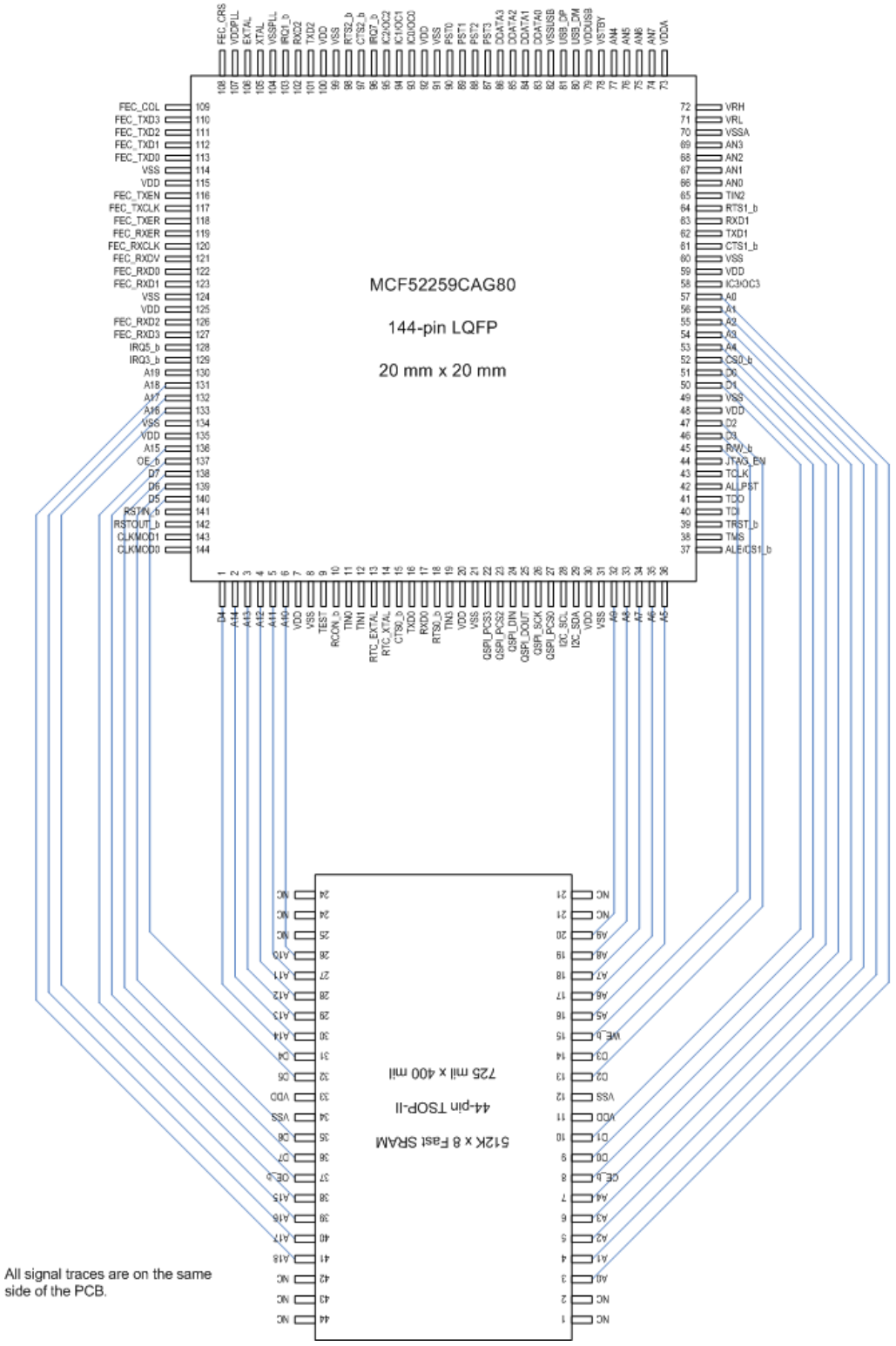

Figure 2. Mini-FlexBus to Industry Standard x8 Asynchronous Fast SRAM

3.1.3 Software Setup

1. The Mini-FlexBus pins must first be enabled via the GPIO port-assignment registers. For this example, the FB_CS1/FB_ALE pin is set to the FB_CS1 (chip select one) function, and the remaining Mini-FlexBus pins are set to their primary Mini-FlexBus function.

```
/* Write to GPIO Port Assignment Registers to enable correct pin functionality */

MCF_GPIO_PTEPAR = 0xFF; //Enable Address Lines A0-A7
MCF_GPIO_PTFPAR = 0xFF; //Enable Address Lines A8-A15
MCF_GPIO_PTGPAR = 0xFF; //Enable Address Lines A16-A19
MCF_GPIO_PTHPAR = 0x5555; //Enable Data Lines D0-D7
MCF_GPIO_PASPAR = 0x20; //Enable FB_CS1 function
```

NOTE

The C code software snippets in this application note use register macro definitions found in the header files of the MCF5225x sample projects. For register details see the *MCF52259 Reference Manual*.

2. The base address is set in the base address field of the Chip Select Address Register CSAR[BA] for the corresponding chip select. In this example, the SRAM is connected to chip select zero (FB_CS0) and the base address is chosen to be 0x80000000. This address is 64-byte aligned, and it is in an available memory space.

```
/* Set Chip Select 0 Base Address */

MCF_FBCS0_CSAR = MCF_FBCS_CSAR_BA(0x80000000); //Set Base Address to 0x80000000
```

3. The next step is to write to the Chip Select Control Register (CSCR). The CSCR must be initialized to accommodate the timing of the specific memory. The following items can be adjusted to fit the application.
   - Wait States
   - Address Setup
   - Read Address Hold
   - Write Address Hold

   In this example, the default values for Address Setup, Write Address Hold, and Read Address Hold are used. See *MCF52259 Reference Manual* (MCF52259RM) for detailed descriptions of those attributes.

In this example, the SRAM returns valid data 10 ns after the chip select ($\overline{CS}$) activates and 5 ns after output enable ($\overline{OE}$) activates. Another way to interpret the timing is that the SRAM requires $\overline{CS}$ at least 10 ns and $\overline{OE}$ at least 5 ns before it returns valid data. This is illustrated in Figure 3.

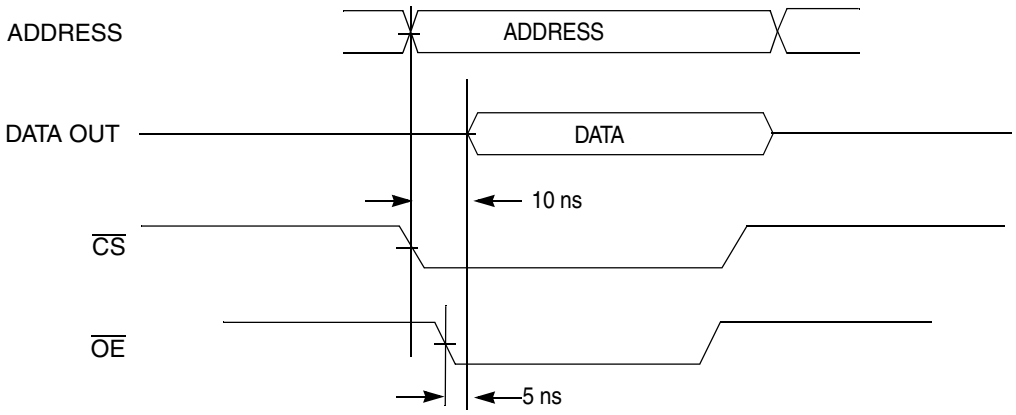

Figure 3. Example Simplified SRAM Read Cycle Timing

To determine the required adjustments to the Mini-FlexBus timing, the Mini-FlexBus chapter of the *MCF52259 Reference Manual* (MCF52259RM) and the Timing Spec in the *MCF5225x Datasheet* (MCF52259DS) are used.

The fastest Mini-FlexBus transfers take four cycles of the internal bus. The address drives in the first cycle. The chip select and output enable assert after the start of the second cycle (output valid time, 8 ns for MCF5225x). The data is required to be set up on the bus by the SRAM before the start of the third cycle (input setup time, 6 ns for MCF5225x). This is illustrated in Figure 4.

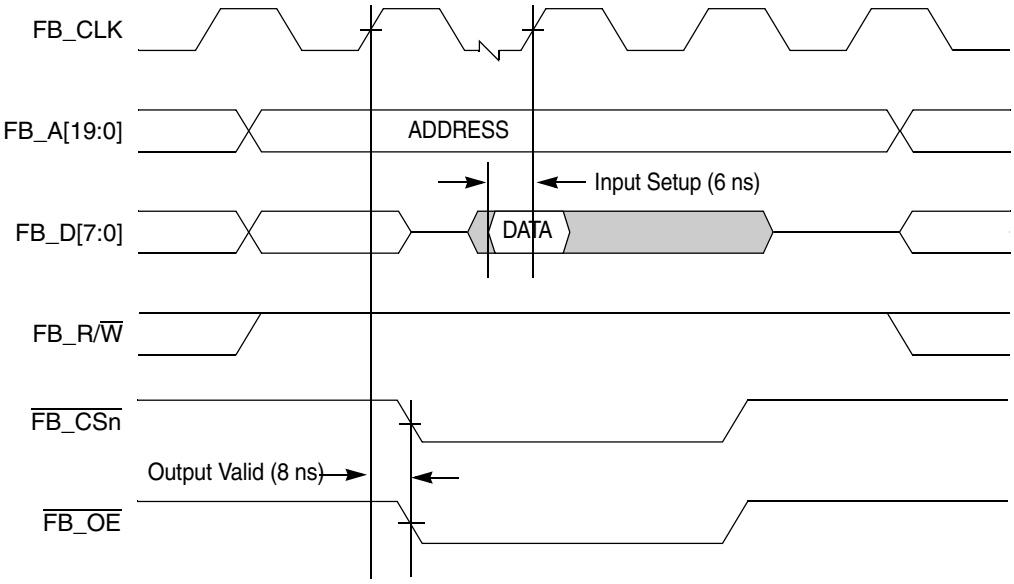

Figure 4. Simplified Mini-FlexBus Read Cycle Timing Spec

In this example, the Mini-FlexBus is in 1:2 mode, therefore it is running at 40 Mhz with 25 ns cycles. The chip select and output enable assert at the same time, therefore the output enable time is not a concern in this example. It will always assert before the 5 ns requirement.

To determine the amount of time between when the chip select asserts and the data is required, subtract the output hold and input setup time from the cycle period.

$$\textbf{25 ns – 8 ns – 6 ns = 11 ns} \qquad \textit{Eqn. 1}$$

If the SRAM does not provide the data faster than 11 ns after chip select asserts, then wait state cycles can be added to give it more time. However, in this example, the SRAM provides the data in 10 ns, therefore no wait state cycles are required.

Now that it has been determined that no wait states are required, the Chip Select Control Register (CSCR) is written. For this example, the port size is set to 8-bit and Mux bit is set to zero to enable non-multiplexed mode.

NOTE

External Transfer Acknowledges are not available on MCF5225x, therefore the CSCR[AA] bit must be set to one to enable Auto Acknowledge.

```
/* Set Chip Select 0 Control Register */
/* Note: WS, MUX, ASET, RDAH, WRAH are all cleared */

MCF_FBCS0_CSCR = MCF_FBCS_CSCR_AA //Enable Auto Acknowledge
                 | MCF_FBCS_CSCR_PS_8; //set Port Size to 01 for 8 bit
```

4. Next, write to the base address mask register to set the base address mask CSMR[BAM], write protect CSMR[WP], and valid CSMR[V] fields. CSMR[BAM] determines the address range, that is, the chip select is active. CSMR[WP] enables/disables write protect. The CSMR[V] bit is set to conclude the configuration and make the chip select settings valid.

```
/* Set Chip Select 0 Mask Register */
/* Note: WP is cleared to disable write protect */

MCF_FBCS0_CSMR = MCF_FBCS_CSMR_BAM_512K //Set Base Address Mask to 0x0007
                 | MCF_FBCS_CSMR_V; //Set Valid bit to 1
```

Setting CSMR[BAM] to 0x0007 results in a base address mask of 0x0007FFFF. With base address 0x80000000 and base address mask 0x0007FFFF, the chip select will be active for address range 0x80000000 – 0x8007FFFF.

In binary, this looks like the following:

Table 1. Chip Select Base Address and Mask

Base Address	Hex	8				0				0				0				0				0				0				0			
	Binary	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base Address Mask	Hex	0				0				0				7				F				F				F				F			
	Binary	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Resulting Chip Select Decode	Binary	1	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

0 or 1 = bit is used to determine, if within chip-select address range.

x = bit is a “don’t care.”

NOTE

Select one of the following base address masks to maintain a contiguous memory space without gaps in the chip select address range.

Table 2. Suggested Chip Select Base Address Mask Values

Memory Size	CSMR[BAM]
1 Mbyte	0x000F
512 Kbyte	0x0007
256 Kbyte	0x0003
128 Kbyte	0x0001
64 Kbyte	0x0000

3.1.4 Transfer Diagrams

Read and write transfers with no wait states using non-multiplexed 8-bit mode.

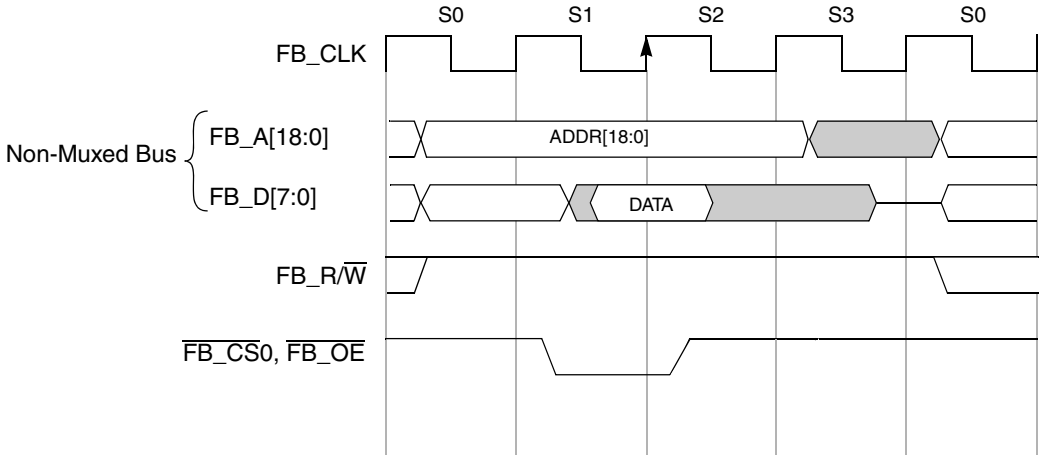

Figure 5. Read (Non-Muxed Mode 8-Bit with No Wait States)

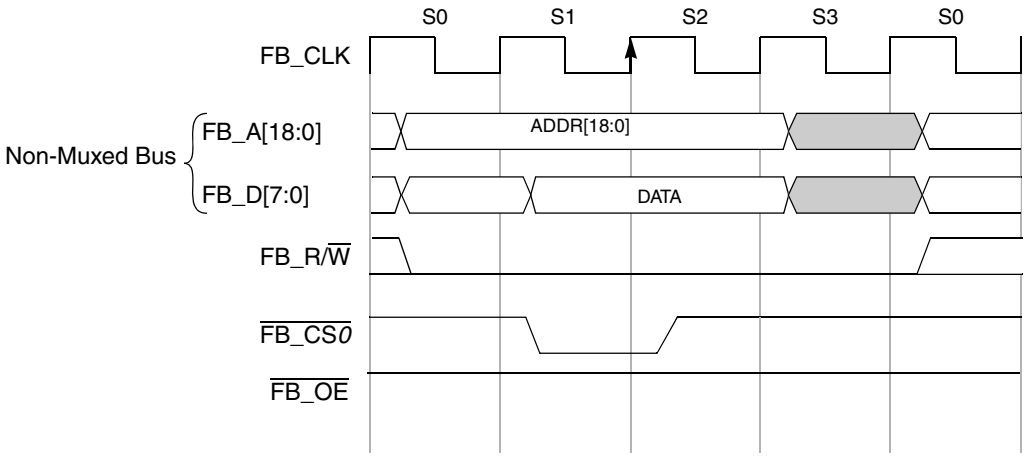

Figure 6. Write (Non-Muxed Mode 8-Bit with No Wait States)

3.1.5 Throughput Calculations

In this example, the theoretical maximal throughput is the following.

If the bus is running at 40 Mhz, with the transfers taking four cycles, the total time per transfer is 100 ns. In 8-bit non-muxed mode, you can transfer eight bits in four cycles, therefore eight bits per 100 ns. This translates to 80 Mbit / sec.

$$\frac{8\ bits}{4\ cycles \times 25\ ns} = 80\ Mbps \qquad \textit{Eqn. 2}$$

NOTE

In actual situations, the theoretical maximum throughput is not possible because the system inserts cycles in between transfers. To achieve the best results use DMA or 16-byte-line-sized transfers.

3.2 Example 2 — Connecting to an Industry Standard x16 Asynchronous MRAM in Byte Mode Using Minimal Glue Logic

In this example the Mini-FlexBus is connected to a single Magnetoresistive Random Access Memory (MRAM) chip, which is in the same footprint as an industry standard x16 asynchronous Fast SRAM. To connect to a 16-bit memory using the absolute minimum glue logic, this example also uses the non-multiplexed mode with 8-bit data and 20-bit address lines.

This solution is demonstrated on the M52259EVB evaluation board.

Advantages:

- It increases available system memory by adding a 512 KByte MRAM.
- MRAM is a non-volatile memory that does not require special erasing and programming routines.
- Only a single gate inverter is required.

Disadvantages:

- MRAM has slower access time than SRAM, and therefore the memory bandwidth is lower.
- To reduce the amount of glue logic, this method treats the MRAM as an 8-bit memory, and thus provides less performance than 16-bit.

3.2.1 Schematics

Although the MRAM is 16-bit, it is byte-read/writable. It has byte high enable and byte low enable pins to select between the upper or lower byte. These signals can be constructed from the Mini-FlexBus signals using only a single gate inverter on FB_A0. FB_A0 is used as byte high enable and the output of the inverter $\overline{\text{FB\_A0}}$ is used as byte low enable. This effectively treats the memory as an 8-bit device. Performance is sacrificed using this method, however, it minimizes glue logic, which saves on component cost.

The eight data lines from the Mini-FlexBus are connected to both the upper byte and lower byte of the 16 data lines of the MRAM. The byte-enable lines that are generated by FB_A0 and $\overline{\text{FB\_A0}}$ select, which byte is driving on the data lines. When not selected, the other byte lines of the MRAM will tri-state and not interfere with the desired byte.

The memory is word-addressable (16-bit), so the address lines must be shifted such that the Mini-FlexBus word address line FB_A[1] is connected to A[0] of the memory. Thus, FB_A[18:1] are connected to A[17:0] of the memory.

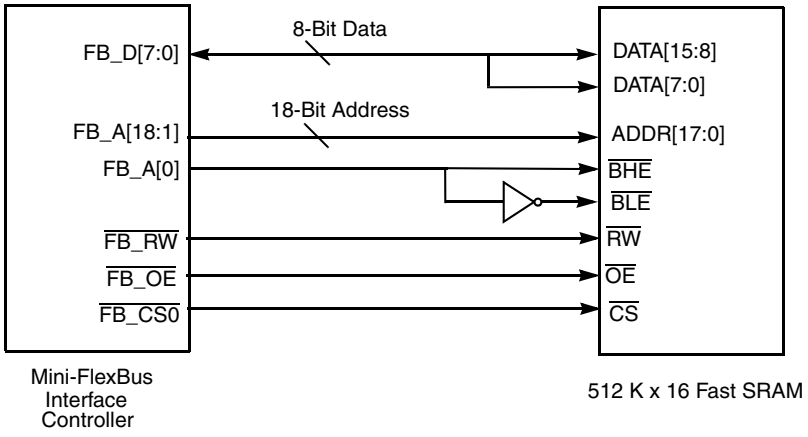

Figure 7. Non-Multiplexed x8 Mini-FlexBus to x16 Memory

3.2.2 PCB Layout

Figure 7 demonstrates the benefit of the Mini-FlexBus bonding order on the MCF5225x by showing the ease, with which an industry standard x16 asynchronous MRAM is connected. The signals can be routed on a single signal layer.

NOTE

On the provided example layout, there are several occurrences, in which the data lines aren't connected in order. This helps with the routing. It is not a problem, because the order of the data bits is irrelevant to the MRAM as long, as the bits are separated correctly according to the high byte and the low byte. On a flash memory, or another type of memory that requires special erase/programming algorithms, the data lines must be connected in order to ensure proper operation.

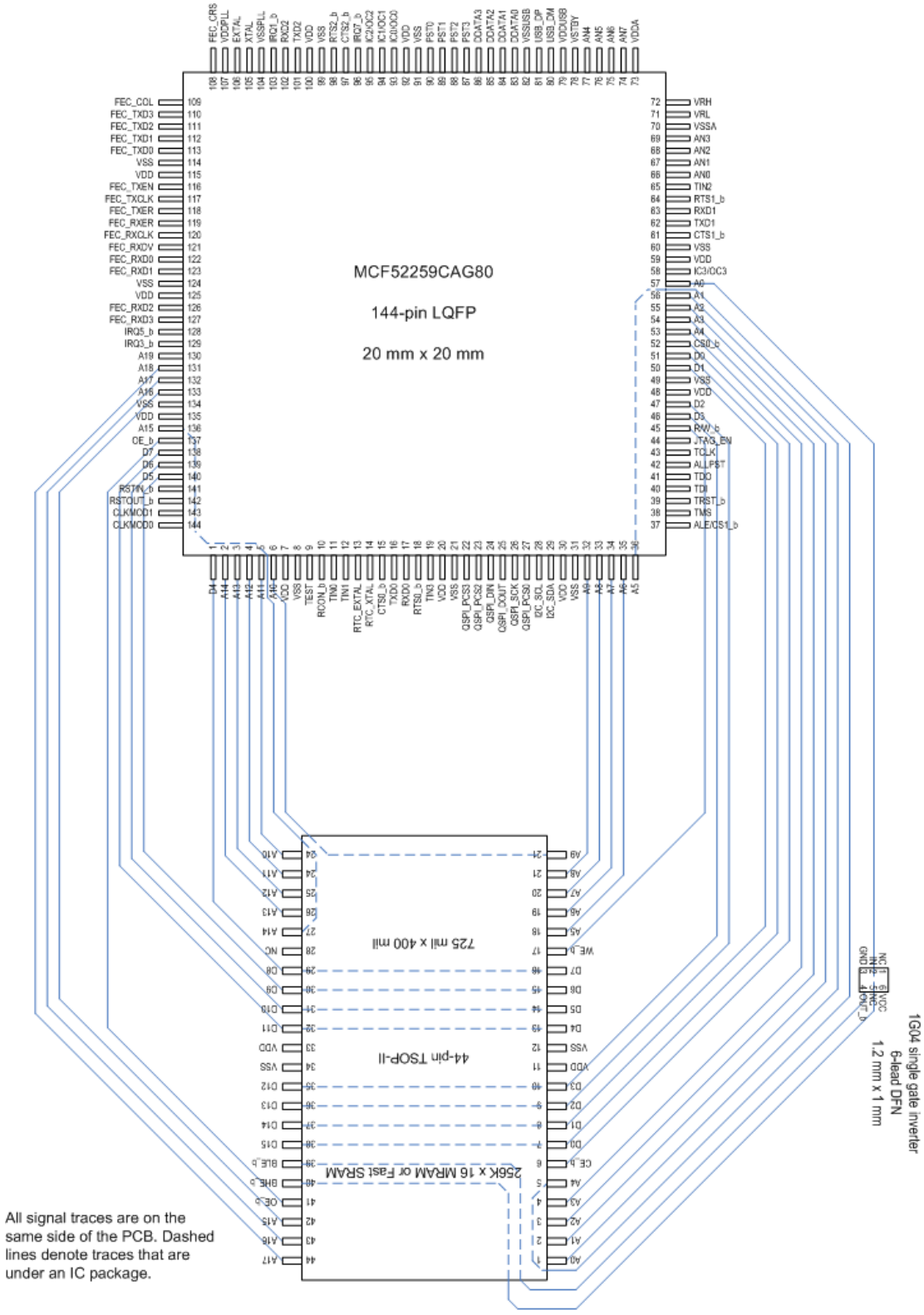

Figure 8. Mini-FlexBus to Industry Standard x16 Asynchronous MRAM

3.2.3 Software Setup

1. Enable the pins the same as in the last example.

```
/* Write to GPIO Port Assignment Registers to enable correct pin functionality */

MCF_GPIO_PTEPAR = 0xFF; //Enable Address Lines A0-A7
MCF_GPIO_PTFPAR = 0xFF; //Enable Address Lines A8-A15
MCF_GPIO_PTGPAR = 0xFF; //Enable Address Lines A16-A19
MCF_GPIO_PTHPAR = 0x5555; //Enable Data Lines D0-D7
MCF_GPIO_PASPAR = 0x20; //Enable FB_CS1 function
```

2. Set chip select zero base address to 0x30000000 or any 64-kByte aligned address in an available memory space.

```
/* Set Chip Select 0 Base Address */

MCF_FBCS0_CSAR = MCF_FBCS_CSAR_BA(0x30000000); //Set Base Address to 0x30000000
```

3. This example uses the same setup as before, but with one wait state added. This is because the MRAM is slower and has an access time of 35 ns. The necessary wait state is determined using the same method as the first example. Adding one wait state cycle adds one cycle (25 ns) in between, when the chip select asserts and valid data. Therefore, the Mini-FlexBus will drive the chip select and require the data 36 ns later (50 – 8 – 6 = 36).

```
/* Set Chip Select 0 Control Register */
/* Note: MUX, ASET, RDAH, WRAH are all cleared */

MCF_FBCS0_CSCR = MCF_FBCS_CSCR_WS(1) //Enable 1 Wait State
                 | MCF_FBCS_CSCR_AA //Enable Auto Acknowledge
                 | MCF_FBCS_CSCR_PS_8; //set Port Size to 01 for 8 bit
```

4. Set chip select zero base address mask to 0x0007 for 512 KByte memory size.

```
/* Set Chip Select 0 Mask Register */
/* Note: WP is cleared to disable write protect */

MCF_FBCS0_CSMR = MCF_FBCS_CSMR_BAM_512K //Set Base Address Mask to 0x0007
                 | MCF_FBCS_CSMR_V; //Set Valid bit to 1
```

3.2.4 Transfer Diagrams

Non-muxed mode with one wait state and A0 & /A0 used as byte enables.

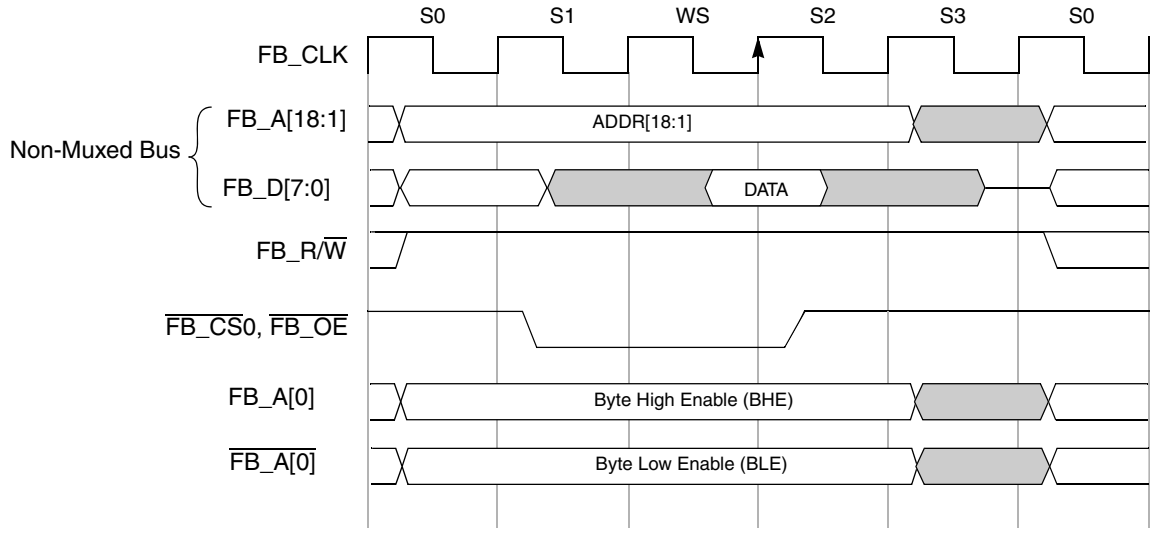

Figure 9. Read (Non-Muxed Mode 8-Bit with One Wait State and A0 and Inverted A0 Used as Byte Enables)

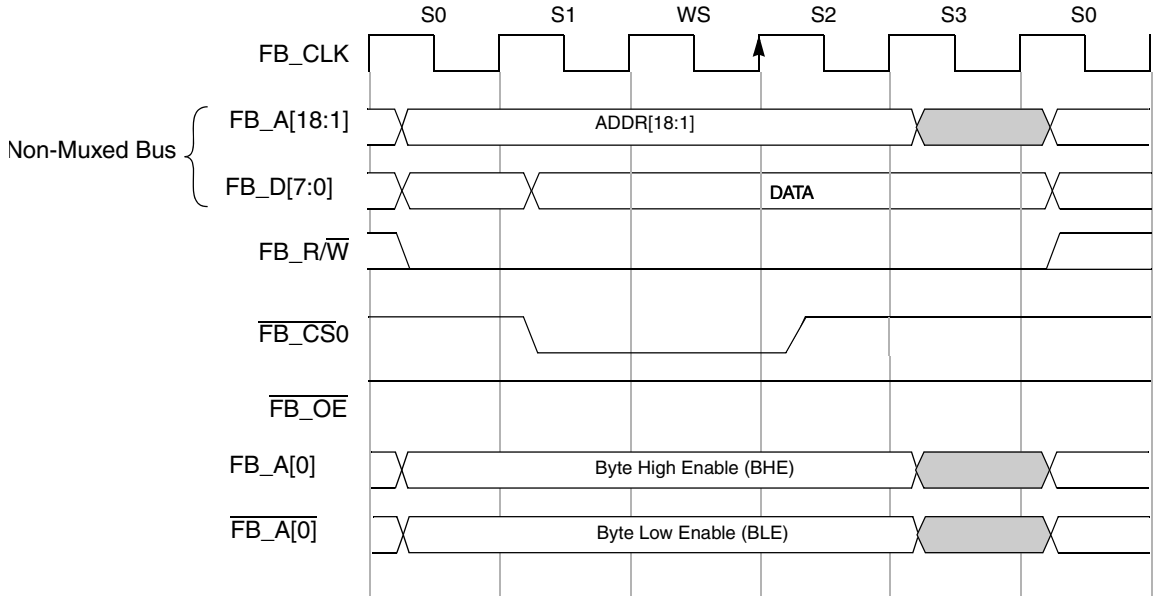

Figure 10. Write (Non-Muxed Mode 8-Bit with One Wait State and A0 and Inverted A0 Used as Byte Enables)

3.2.5 Throughput Calculation

In this example, the theoretical maximal throughput is the following.

If you run the bus at 40 Mhz, with transfer cycles taking four cycles plus one wait state cycle. In 8-bit non-muxed mode you can transfer eight bits in five cycles, therefore eight bits in 125 ns. This translates to 64 Mbit / sec.

$$\frac{8\ bits}{5\ cycles \times 25\ ns} = 64\ Mbps \qquad \textit{Eqn. 3}$$

3.3 Example 3 — Industry Standard x16 Asynchronous Fast SRAM Optimized for Performance

This example connects to an Industry Standard x16 Asynchronous Fast SRAM. The x16 interface SRAM is similar to the MRAM of the previous example, however, in this example uses multiplexed mode with 20-bit address shared with 16-bit data. This configuration allows for higher performance, but it requires the use of an external latch component. During the address phase of the transfer, the ALE pin activates the latch, which captures and holds the address. In this example the SRAM is connected to $\overline{\text{FB\_CS0}}$. Chip select one $\overline{\text{FB\_CS1}}$ is not available, because the pin is in use as FB_ALE.

Advantages:

- Frees up the dedicated FB_D[7:0] data lines for use as GPIO.
- Improves performance by allowing 16-bit transfers.

Disadvantages:

- Additional cost added due to the external latch component.
- Increases signal-routing complexity.
- Byte read/writes are not supported in this method.
- Only one chip select is available.

3.3.1 Schematics

In multiplexed mode, the FB_A[19:0] pins are notated as FB_AD[19:0], because they now are both address and data signals. In this example, FB_AD[15:0] are connected to DATA[15:0] of the SRAM. The SRAM is word-addressable (16-bit), so the address lines must be shifted such that the Mini-FlexBus word address line FB_AD[1] is connected to A[0] of the memory. Thus, FB_AD[15:1] are connected to A[14:0] of the memory through the latch, and FB_AD[18:16] are connected to A[17:15] directly.

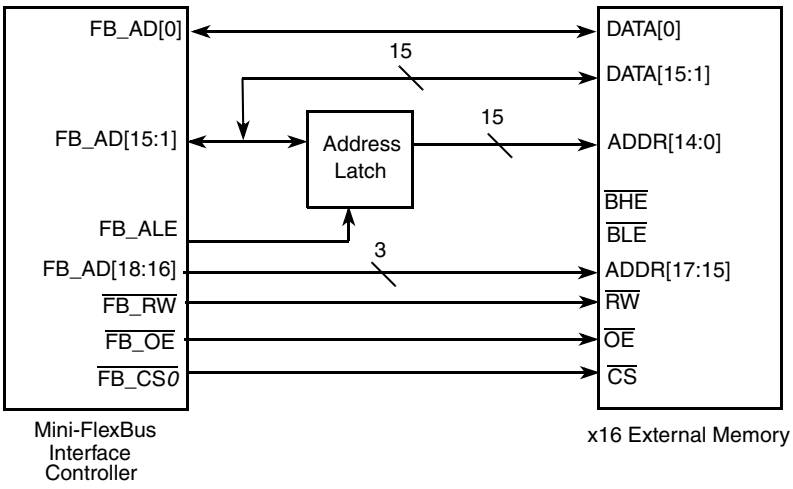

Figure 11. Multiplexed x16 Mini-FlexBus to x16 Memory

3.3.2 PCB Layout

The PCB layout picture below is a proof of concept, and to show, how the signals could potentially be routed.

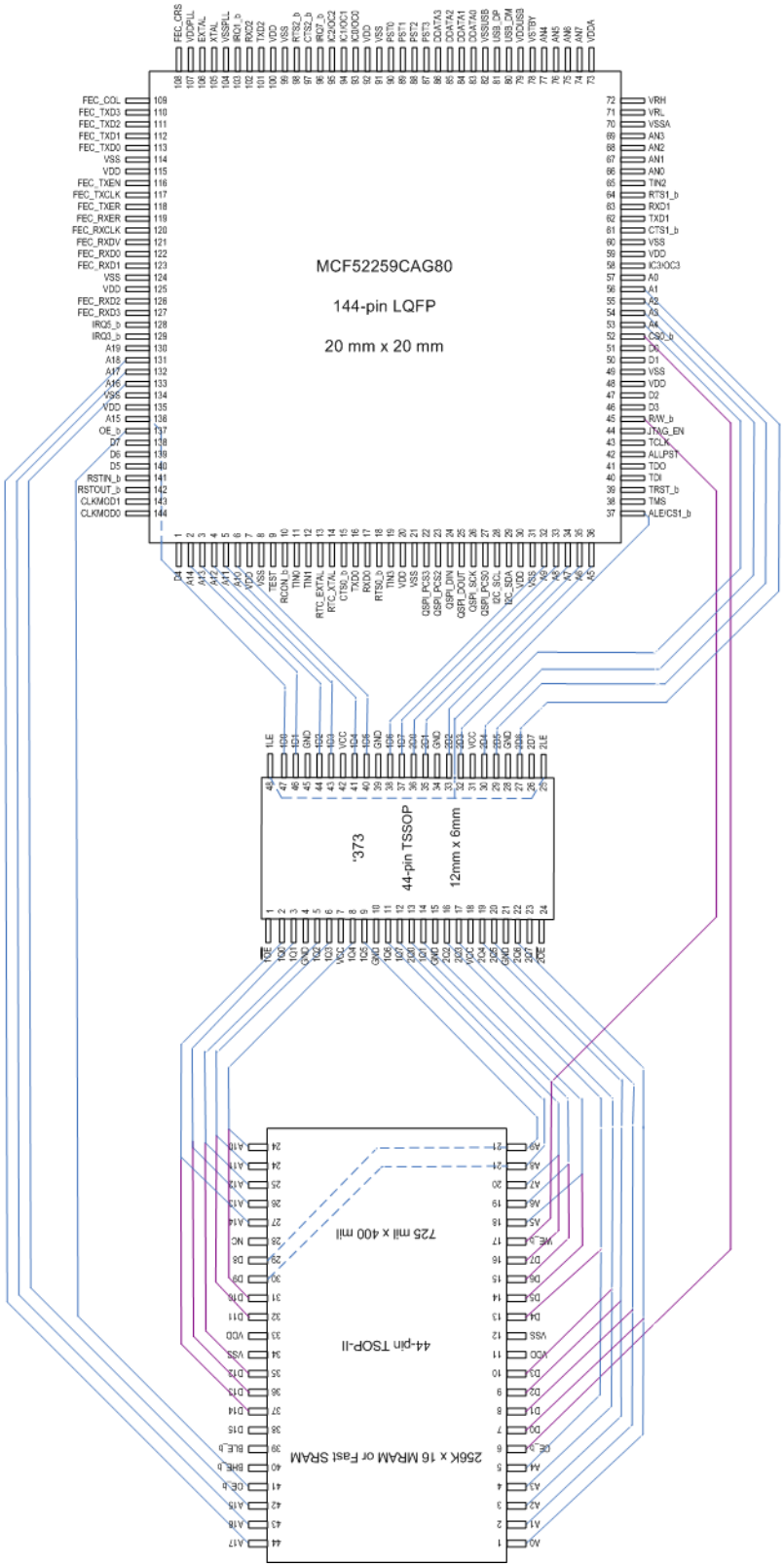

Figure 12. Mini-FlexBus to Industry Standard x16 Asynchronous Fast SRAM

3.3.3 Software Setup

1. The Mini-FlexBus pins must first be enabled via the GPIO port assignment registers. In this example, FB_D[7:0] are not required, therefore Port TH is configured to be used as GPIO. Port AS is configured such that the FB_CS1/FB_ALE pin is set to the FB_ALE (Address Latch Enable) function.

```
/* Write to GPIO Port Assignment Registers to enable correct pin functionality */

MCF_GPIO_PTEPAR = 0xFF; //Enable Address Lines A0-A7
MCF_GPIO_PTFPAR = 0xFF; //Enable Address Lines A8-A15
MCF_GPIO_PTGPAR = 0xFF; //Enable Address Lines A16-A19
MCF_GPIO_PTHPAR = 0x0000; //Set Data Lines to general purpose input/output (GPIO)
MCF_GPIO_PASPAR = 0x10; //Enable FB_ALE function
```

2. Set chip select zero base address to 0x30000000, or any 64-kByte aligned address in an available memory space.

```
/* Set Chip Select 0 Base Address */

MCF_FBCS0_CSAR = MCF_FBCS_CSAR_BA(0x30000000); //Set Base Address to 0x30000000
```

3. In this example the port size is set to 16-bit. Multiplexed mode is enabled by writing to the CSCR[MUX] bit. Just as in the first example, no wait states or other timing adjustments are required.

```
/* Set Chip Select 0 Control Register */
/* Note: WS, ASET, RDAH, WRAH are all cleared */

MCF_FBCS0_CSCR = MCF_FBCS_CSCR_AA //Enable Auto Acknowledge
                 | MCF_FBCS_CSCR_PS_16; //set Port Size to 10 for 16 bit
                 | MCF_FBCS_CSCR_MUX; //Enable Multiplexed Mode
```

4. Set chip select zero base address mask to 0x0007 for 512 KByte memory size.

```
/* Set Chip Select 0 Mask Register */
/* Note: WP is cleared to disable write protect */

MCF_FBCS0_CSMR = MCF_FBCS_CSMR_BAM_512K //Set Base Address Mask to 0x0007
                 | MCF_FBCS_CSMR_V; //Set Valid bit to 1
```

3.3.4 Transfer Diagrams

During the address phase, the address information on FB_AD[15:0] are latched. During the data phase, the data is driven on FB_AD[15:0].

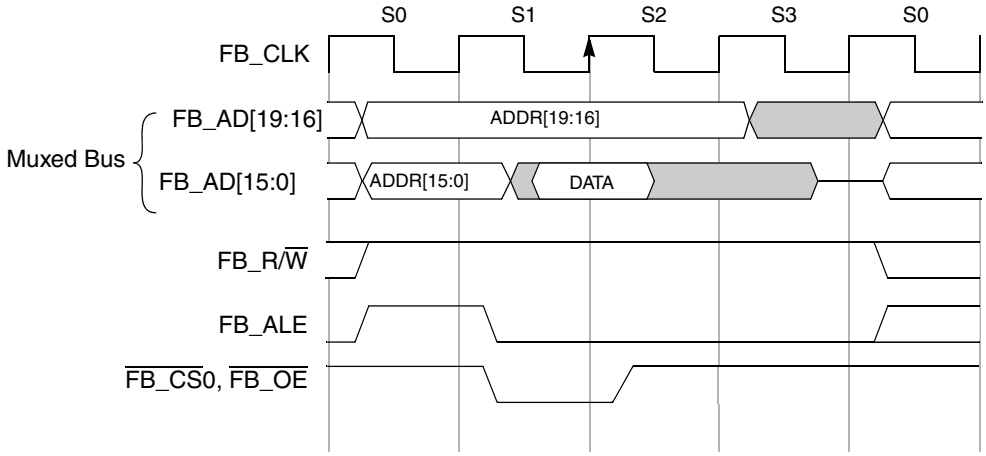

Figure 13. Read (Muxed Mode 16-Bit Data with no Wait States)

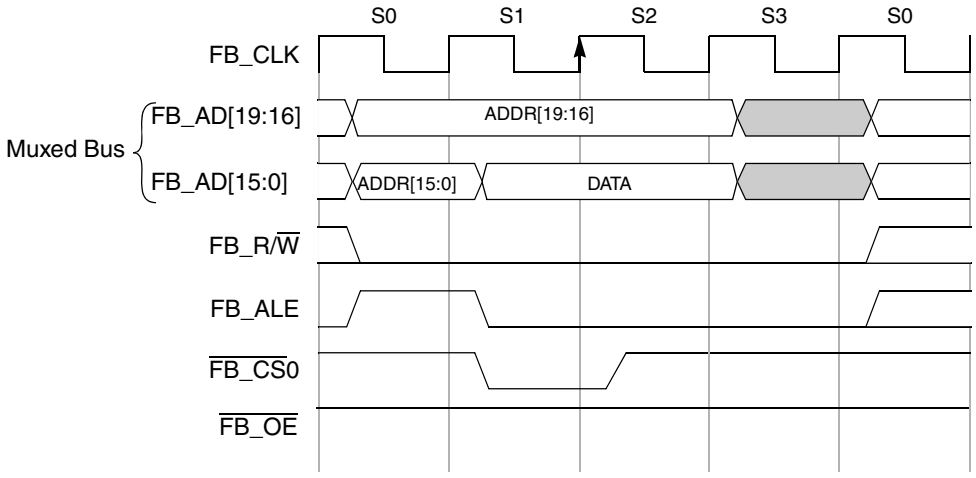

Figure 14. Write (Muxed Mode 16-Bit Data with no Wait States)

3.3.5 Throughput Calculation

In this example the theoretical maximal throughput is the following.

If the bus is running at 40 Mhz, with the transfers taking four cycles, the total time per transfer is 100 ns. In 16-bit muxed mode, you can transfer 16 bits in four cycles, therefore 16 bits per 100 ns. This translates to 160 Mbit / sec.

$$\frac{16\ bits}{4\ cycles \times 25\ ns} = 160\ Mbps \qquad \textit{Eqn. 4}$$

3.4 Document Revision History

Table 3 provides a sample revision history table. Most of our documents include such table, though it may often be conditionalized for internal reference only.

NOTE

Do not use cross references in the Document Revision History table.

Table 3. Document Revision History

Rev. No.	Substantive Change(s)
0	Initial release.
1	The following statement was removed because this feature is not supported by the MCF5225x. “When internal flash security is enabled, memory accesses through the Mini-FlexBus are blocked.”

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3854
Rev. 1
09/2009

freescale™
semiconductor