Freescale Semiconductor
Application Note

AN2934
Rev. 0, 8/2005

Using the MAC7100 EIM (External Interface Module)

by: Alasdair Robertson
TSPG MCD Applications Engineering

1 Introduction

The Freescale Semiconductor MAC7100 family of microcontrollers, incorporating the ARM7TDMI-S™ processor, features an External Interface Module, or EIM.

This application note details the operation of the MAC7100 EIM from a user's perspective. After reading this document, you should have a fundamental understanding of the EIM module, and its registers and control signals; you should know how to configure the EIM chip selects and how to configure and connect the EIM to an external peripheral such as SRAM or FLASH. To aid understanding, example hardware and software configurations are detailed along with screen shots of actual EIM bus transfers captured on a logic analyser.

Note that for a full understanding of the EIM module, you are advised to read this document along with the *MAC7100 Microcontroller Family Reference Manual* (MAC7100RM, available via the Freescale web site: www.freescale.com/mac7100).

Table of Contents

freescale™ semiconductor

1.1 Background

Derived from the Freescale ColdFire® External Interface Module, the MAC7100 EIM provides a truly glueless bus interface, allowing seamless connection to a number of SRAM and FLASH memories and other peripherals.

The MAC7100 EIM supports 8-bit and 16-bit port sizes (data bus width) with a data transfer size of byte (8 bits), half-word (16 bits) and word (32 bits). Where the data transfer size is larger than the port size, back-to-back transfers are performed automatically.

NOTE

The ColdFire EIM supports fully burstable, 32-bit data port transfers. In current implementations of the MAC7100, the upper 16 bits of the data bus, along with some of the burst signals, have not been bonded out. This limits the data width to 16 bits; consequently, burst mode is not fully supported or discussed in this application note.

1.2 Conceptual Overview

Before describing the EIM signal descriptions and configuration, it is important to have a fundamental understanding of what an EIM provides.

Essentially, the EIM offers a user-definable external memory space, or address range. Whenever one of the addresses in the range is accessed (by simply reading or writing to the physical address), the EIM automatically activates a chip select for that address range. By connecting a peripheral device, such as an SRAM, to the appropriate EIM signals, this device is then "mapped" to the defined external memory space.

On the MAC7100 EIM, there are three independent chip selects, allowing up to three memory mapped peripherals to be connected to the bus. An example of this is shown in Figure 1-1.

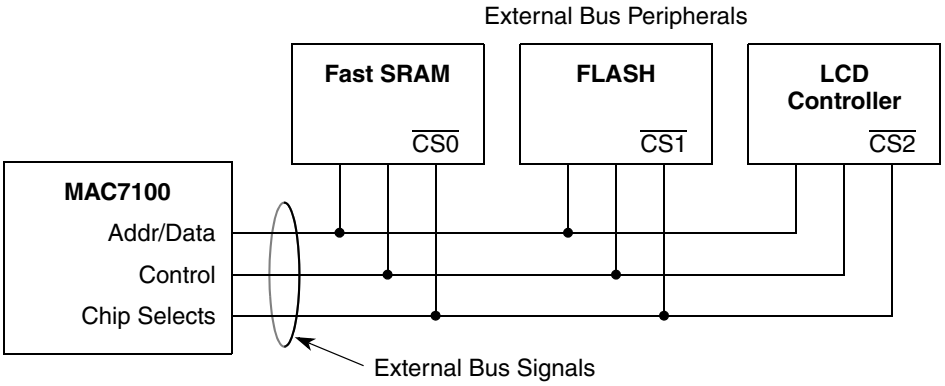


Figure 1-1. External Bus Structure

Note that this is an example and does not mean that the chip selects must be connected as shown. Any chip select can be used to drive any peripheral.

1.3 Signal Description

Before the configuration and setup of the EIM is discussed, it is worthwhile detailing the actual EIM interface signals on the MAC7100 family.

Table 1-1. MAC7100 EIM signal Descriptions

EIM Signal	GPIO Port	Description	Comments
Addr [21..0]	PortD[10..5], PortC[15..0]	Address Bus	22-bit address bus
Data [15..0]	PortA[15..0]	Data Bus	16-bit data bus
$\overline{TA}$ [1]		Transfer Acknowledge	MCU input signal to indicate that peripheral has completed requested transfer.
$\overline{AS}$ [1]		Address Strobe	Indicates that a bus cycle has been initiated and the address is stable
$\overline{BS0}$, $\overline{BS1}$	PortD[0..1]	Byte Select	Determines which bytes of the external data bus are activated during an EIM transfer
CLKOUT	PortD[2]	MCU Clock Out	Represents CPU clock
$\overline{OE}$	PortD[11]	Output Enable	Indicates that external device can drive the data bus (for read operations)
$\overline{CS2}$, $\overline{CS1}$, $\overline{CS0}$	PortD[12..14]	Chip Select	Used to enable multiple external peripherals. The EIM will trigger the relevant chip select when the MCU performs an external access to the relevant address.
R/$\overline{W}$	PortD[15]	Read / Write Line	Indicates if current cycle is a read (logic 1) or write (logic 0)

NOTES:
1. The $\overline{TA}$ and $\overline{AS}$ signals are multiplexed on the MAC7100 family. This means that only one of these functions may be used at a time.

As can be seen from Table 1-1, with the exception of $\overline{TA}$ and $\overline{AS}$, all of the EIM signals are multiplexed with GPIO ports A, C, and D. When the MCU is reset into expanded mode, these ports default to peripheral mode, with all of the EIM signals available and active. If a particular EIM signal is not being used, it is possible to reclaim the respective port pin by setting the port back to I/O mode. See Section 6.1, "Disabling EIM Signals in Expanded Mode," for more details.

1.3.1 Address Bus

The MAC7100 EIM provides 22 address lines, Addr[21..0], with Addr[0] being the least significant bit (LSB). This gives an addressable block size of 2^{22} bytes (or 4 Mbytes). In terms of physical address mapping, this corresponds to a range of 0x0040_0000.

1.3.2 Data Bus

There are 16 data lines, Data[15..0], with Data[0] being the LSB. The MAC7100 data format is big-endian. Understanding the "endian" format is a fundamental part of a design using the EIM, so an overview of big-endian format is given below:

Consider a 16-bit data value, “0x1234”. This is stored in memory as two bytes, “0x12” and “0x34”. Byte “0x12” is the most significant byte (MSB). Big-endian storage format means that the bytes are stored sequentially in memory (from low address to high address) in the order MSB to LSB. In little-endian format, the storage order of the bytes is reversed. This is shown in Figure 1-2.

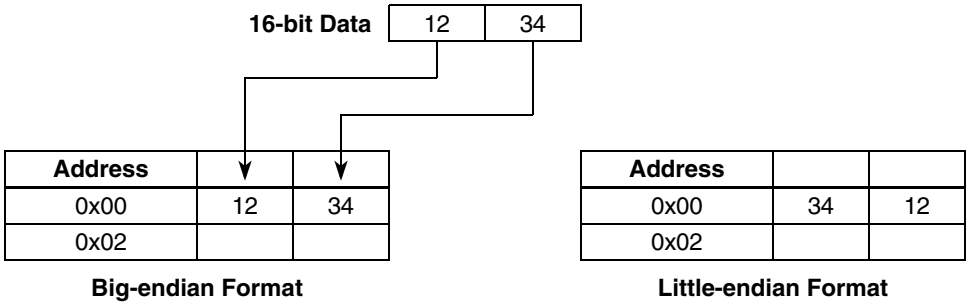


Figure 1-2. Big-endian Data Format

1.3.3 Transfer Acknowledge

The EIM provides an active low $\overline{\text{TA}}$ (Transfer Acknowledge) input signal. This is used by an external peripheral to indicate to the MAC7100 that it has completed its requested data transfer (read or write), and that the transfer cycle can be terminated.

After the MCU has started an external data transfer, with the EIM asserting the relevant chip select (see Section 1.3.8, “Chip Selects”), the $\overline{\text{TA}}$ signal is sampled by the EIM on every rising edge of CLKOUT. On detection of an asserted $\overline{\text{TA}}$ signal (logic low), the MCU either reads the data on the bus (for a data read), or latches the data onto the bus (for a data write), then completes the current transfer cycle. This feature is extremely useful as it allows an external device to have a variable response time, terminating the transfer when it is ready to do so.

Note that the MCU will wait indefinitely for an assertion of the $\overline{\text{TA}}$ signal, before terminating the current transfer. This means that, if the external peripheral does not issue a $\overline{\text{TA}}$, the MCU will wait forever. At this point, no application code will be running. The regular MCU watchdog can be used to recover from such an occurrence.

The EIM module also has the capability of generating its own internal $\overline{\text{TA}}$ signal. This provides an automatic assertion of $\overline{\text{TA}}$ after a pre-defined number of clock cycles from the point the EIM transfer was started. This is useful for interfacing to external devices where the exact response time is known, for example when using asynchronous memories. This concept will be explored in more detail throughout this application note.

1.3.4 Address Strobe

The $\overline{\text{AS}}$ (Address Strobe) signal is used to indicate that the address is stable at the start of a bus cycle. The address is valid for the duration that $\overline{\text{AS}}$ is asserted. This signal is mainly used for burst type memory transfers and will not be referenced further in this application note. For more details, refer to the MAC7100 reference manual.

1.3.5 Byte Select

As previously mentioned, the MAC7100 EIM supports data port sizes of 8 bits and 16 bits, with accesses of byte (8 bits), half-word (16 bits) and word (32 bits). The active low byte select signals, $\overline{BS0}$ and $\overline{BS1}$, are used to determine the “laning”, i.e. which byte(s) from the 16-bit data port are active during the current EIM transfer.

Table 1-2 details all valid combinations of the byte select signals for both a 16-bit and 8-bit port size.

Table 1-2. EIM Byte Select Control

EIM Port Size	Transfer Type	Active Data Pins	$\overline{BS1}$	$\overline{BS0}$
16-bit	Half-word	D[15..0]	0	0
	Byte even	D[15..8]	0	1
	Byte odd	D[7..0]	1	0
8-bit	Byte	D[15..8]	0	1

If the EIM is configured as an 8-bit port, all data accesses are via data pins Data[15..8]. This is an important point to note if you are starting a hardware design with an 8-bit port configuration.

If the EIM is configured as a 16-bit port, the data is routed to the appropriate data bus pins depending on the access type:

- 16-bit accesses use data pins Data[15..0] with both byte select signals active
- 8-bit accesses of an even address (e.g. 0x0, 0x2), use data pins Data[15..8] with byte select $\overline{BS1}$ active
- 8-bit accesses of an odd address (e.g. 0x1, 0x3), use data pins Data [7..0] with byte select $\overline{BS0}$ active

To illustrate why the byte select signals are required, it is important to define exactly what a 16-bit data access entails. Table 1-3 shows both 8-bit and 16-bit accesses.

Table 1-3. 8-bit and 16-bit Access

8-bit Data Access			
Addr	**Data**	**Addr[1]**	**Addr[0]**
0x0	0x01	0	0
0x1	0x23	0	1
0x2	0x45	1	0
0x3	0x67	1	1

16-bit Data Access			
Addr	**Data**	**Addr[1]**	**Addr[0]**
0x0	0x0123	0	0
0x1	**Illegal Access (misaligned)**		
0x2	0x4567	1	0
0x3	**Illegal Access (misaligned)**		

8-bit Access — Data is accessed byte at a time. Address line Addr[0] is used to define whether an odd or even address is being accessed. Addr[0] is the LSB.

16-bit Access — Data is accessed two bytes at a time. On the MAC7100, you cannot do a 16-bit access of an odd address (called a misaligned access). As accesses are always performed on an even address, address line Addr[0] is always 0 and is therefore ignored (and subsequently not connected to an external memory). Address line Addr[1] becomes the LSB.

As stated above, the MAC7100 EIM supports 8-bit, 16-bit, and 32-bit accesses, irrespective of the width of the EIM data port. With the EIM configured as a 16-bit port, address Addr[0] is not connected to the memory. This introduces a potential problem when attempting to perform an 8-bit access of a 16-bit memory, as there is no way for the memory to differentiate between an odd and even address. An 8-bit access of address 0x0, would therefore result in the data at 0x01 being overwritten with an undefined value. The byte select signals provide the "missing" information from Addr[0], as shown in Table 1-4. In essence, the byte select signals allow the system performance of a wider 16-bit port to be maintained, while giving the flexibility of byte access.

Table 1-4. 8-bit Access to a 16-bit Port

Addr	Data	Addr[1]	$\overline{BS1}$	$\overline{BS0}$
0x00	0x01	0	0	1
0x01	0x23	0	1	0
0x02	0x45	1	0	1
0x03	0x67	1	1	0

Most external SRAM memories larger than one byte wide have individual byte enable/disable signals. These can be connected to the relevant EIM byte select signals, to prevent data corruption when performing a byte transfer.

Figure 1-3 shows how the byte select signals are used to effectively partition the blocks inside a 16-bit memory into odd and even arrays. This is exactly the same method by which two 8-bit memory devices can be connected to provide a virtual 16-bit device.

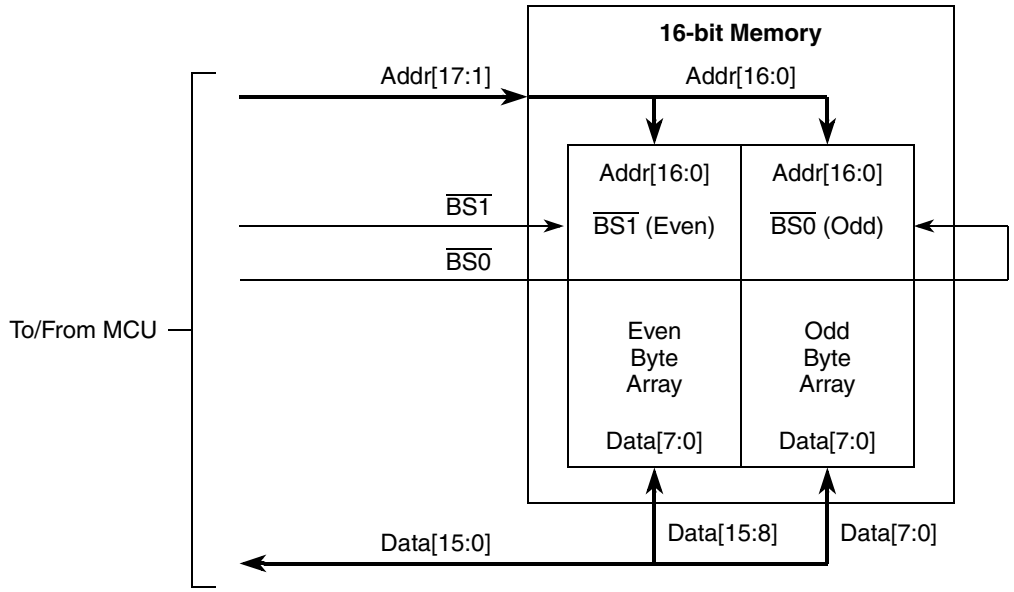


Figure 1-3. Byte Select Implementation

As can be seen in the example above, MCU address lines Addr[17..1] are connected to the memory. Address line Addr[0] is ignored as the memory is addressed in 16-bit blocks. In terms of total memory size, this is determined by:

$$2^{(\text{number of address lines})} = 2^{18} = 262{,}144 \text{ bytes or } 256 \text{ Kbytes} \qquad \textit{Eqn. 1}$$

On some 16-bit memories such as FLASH, there are no byte enable inputs. This means that only 16-bit writes are permitted, or data will be destroyed. Byte reads will result in 16-bit data being returned from the FLASH; however, the CPU will read the correct byte and ignore the unrequested byte.

CAUTION

An external memory system configured for a 16-bit data bus cannot be used with the EIM configured as an 8-bit port. As can be seen from Table 1-2 and Figure 1-3, an 8-bit access to a 16-bit port will only ever access the even byte locations. Every second sequential byte access will therefore be to the wrong location.

Note that the byte select signals can be set to activate on “read/write” operations or just “writes”. This option is selected in the chip select control register (see Section 2.3, “Chip Select Control Register (CSCRn),” for details).

As shown in Table 1-3, the EIM does not support any form of misaligned access. Data must be aligned to the EIM port size, i.e. with a port size of 16 bits, you cannot do a 16-bit access of an odd address. You can however perform a 32-bit access on any 16-bit boundary address.

1.3.6 CLKOUT

The CLKOUT signal provides a slightly delayed representation of the internal system clock. The EIM runs at the same clock speed as the MAC7100 CPU, using the MCU CLKOUT as its timing reference.

For a hardware designer, probably the most critical thing to note is that the rising edge of CLKOUT is the active edge. All EIM timings are derived from this edge, and the rising edge is used for data latching. For more information on CLKOUT and EIM timing diagrams, refer to the MAC7100 Hardware Specifications document.

1.3.7 Output Enable

The EIM provides an active low $\overline{\text{OE}}$ (output enable) signal which is asserted during a read cycle to indicate to an external peripheral (e.g. memory) that it is okay to assert data onto the data bus. Any peripheral capable of driving the bus should have a corresponding “bus enable” or “output enable” signal input.

1.3.8 Chip Selects

There are three active-low chip selects on the EIM: $\overline{\text{CS0}}$, $\overline{\text{CS1}}$, and $\overline{\text{CS2}}$. These allow up to three peripherals to be connected to the same external bus and controlled independently without any complex hardware decode logic. After the chip selects have been configured for a particular address range, they are automatically asserted whenever the MCU accesses this area.

Section 2, “EIM Register Description,” describes the configuration of the chip select control registers.

1.3.9 Read/Write

The R/$\overline{W}$ (read/write) signal determines whether the current EIM transfer is a read or a write. The R/$\overline{W}$ line is the first signal to be set when a transfer is started. If this line is high, the current transfer is a read; if the line is low, the current transfer is a write.

2 EIM Register Description

This section details the registers used to configure the EIM.

There are three sets of configuration registers per chip select. These are mirrored across all three chip selects, as shown in Table 2-1. For additional details on the registers and configuration, consult the MAC7100 Reference Manual.

Table 2-1. EIM Register Map

Chip Select	EIM Offset	Register Description	
		Data [31:16]	Data [15:0]
$\overline{CS0}$	0x0080	Chip Select Address Register 0 (CSAR0)	Reserved
	0x0084	Chip Select Mask Register 0 (CSMR0)	
	0x0088	Reserved	Chip Select Control Register 0 (CSCR0)
$\overline{CS1}$	0x008C	Chip Select Address Register 1 (CSAR1)	Reserved
	0x0090	Chip Select Mask Register 1 (CSMR1)	
	0x0094	Reserved	Chip Select Control Register 1 (CSCR1)
$\overline{CS2}$	0x0098	Chip Select Address Register 2 (CSAR2)	Reserved
	0x009C	Chip Select Mask Register 2 (CSMR2)	
	0x00A0	Reserved	Chip Select Control Register 2 (CSCR2)

2.1 Chip Select Address Registers (CSAR*n*)

The chip select address registers are 16 bits wide, containing only the BA (base address) field.

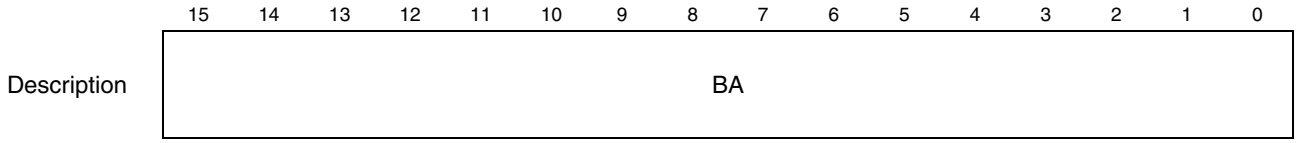


Figure 2-1. Chip Select Address Registers (CSAR*n*)

2.1.1 BA — Base Address

The 16-bit base address field (BA) defines the upper 16 bits of the start (or base) address of your block of external address space. This is compared internally with the MCU address lines Addr[31..16] to determine if there is a valid match.

Therefore, the desired 32-bit base address takes the form “0x<BA>_0000”, where <BA> is the 16-bit base address. The resolution or step size of the base address is defined as the range when the base address = 0; in other words, from 0x0000_0000 to 0x0000_FFFF, or 64 Kbytes.

For example, to define a base address of 0x0020_0000, write 0x0020 into the BA field.

NOTE

Care must be taken to ensure that the memory area defined by the base address is actually available for use and is not taken up by any other peripheral or chip select.

2.2 Chip Select Mask Registers (CSMR*n*)

The chip select mask registers are 32 bits wide, as defined below.

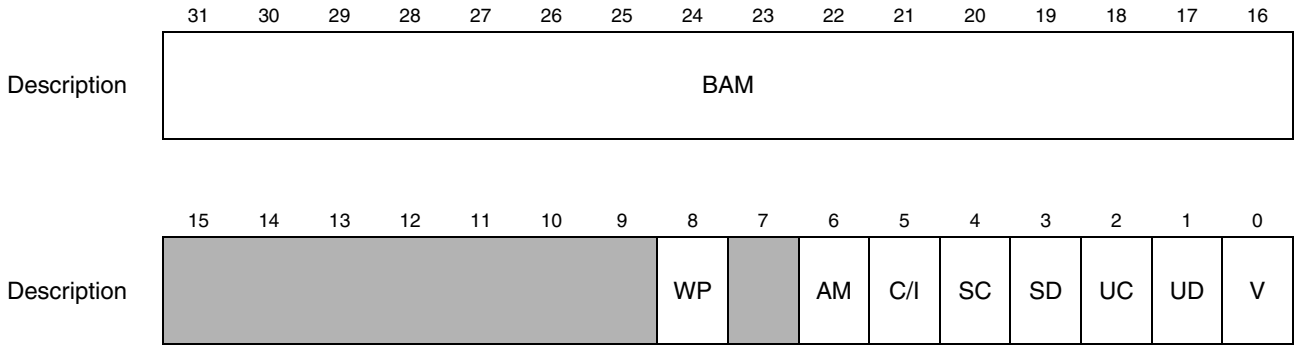


Figure 2-2. CSMR*n* Fields

2.2.1 BAM — Base Address Mask

The 16-bit base address mask (BAM) determines the size of the external memory block. As with the base address field (in the CSAR*n*), the BAM corresponds to the upper 16 bits of the block size, which is compared with the upper 16 CPU address lines and subsequently applied to the base address to give offset + size.

Configuring the BAM field requires careful consideration. Since the BAM is compared with the CPU address lines to determine the block characteristics, a contiguous block can be obtained only if the bits that are set in the BAM field are themselves contiguous. In other words, if BAM[3] is the highest BAM field bit that is set, BAM[2..0] must also be set. Table 2-2 shows the block sizes available when BAM is configured in this manner.

Table 2-2. Sequentially Filled BAM Field

BAM Value (binary)	BAM field	Block Size (Hex)	Block Size (Bytes)	
0000 0000 0000 0000	0x0000	0x0000_FFFF	64 Kbytes	Valid Block Sizes for MAC7100
0000 0000 0000 0001	0x0001	0x0001_FFFF	128 Kbytes	
0000 0000 0000 0011	0x0003	0x0003_FFFF	256 Kbytes	
0000 0000 0000 0111	0x0007	0x0007_FFFF	512 Kbytes	
0000 0000 0000 1111	0x000F	0x000F_FFFF	1 MByte	
0000 0000 0001 1111	0x001F	0x001F_FFFF	2 MBytes	
0000 0000 0011 1111	0x003F	0x003F_FFFF	4 Mbytes	
0000 0000 0111 1111	0x007F	0x007F_FFFF	8 Mbytes	Reserved
0000 0000 1111 1111	0x00FF	0x00FF_FFFF	16 Mbytes	
0000 0001 1111 1111	0x01FF	0x01FF_FFFF	32 Mbytes	
0000 0011 1111 1111	0x03FF	0x03FF_FFFF	64 Mbytes	
0000 0111 1111 1111	0x07FF	0x07FF_FFFF	128 Mbytes	
0000 1111 1111 1111	0x0FFF	0x0FFF_FFFF	256 Mbytes	
0001 1111 1111 1111	0x1FFF	0x1FFF_FFFF	512 Mbytes	
0011 1111 1111 1111	0x3FFF	0x3FFF_FFFF	1 GByte	
0111 1111 1111 1111	0x7FFF	0x7FFF_FFFF	2 Gbytes	
1111 1111 1111 1111	0xFFFF	0xFFFF_FFFF	4 Gbytes	

It may appear that the block sizes are very coarsely defined; however, if you consider that the size of an external memory is usually bounded by the number of address lines, where size in bytes equals $2^{(\text{Num Address Lines})}$, then the block sizes make perfect sense.

The MAC7100 EIM has 22 available address lines. This defines the maximum addressable block size as 4 Mbytes, as shown by the shaded areas in Table 2-2. Using a BAM configuration with a size greater than 4 MBytes is not supported.

An alternative way to configure the BAM is to look at the number of address lines required to address your external memory and subtract 16. This gives the number of sequential BAM bits that must be set (starting from 0). For example, if we have a 512 Kbyte memory, this requires 19 address lines. Addr[0] is counted, even if it is not connected in the case of a 16-bit memory. Therefore, three BAM bits must be set: BAM[2..0]. This gives a BAM value of 0x0003.

There are several things that can be done incorrectly when configuring the BAM:

1. If you configure the BAM such that the block size is larger than the physical memory, overlapping memory segments will occur, causing data loss. For example, if you have a 256 Kbyte memory, sitting at base address 0x0, this would give a true addressable space of 0x0000_0000 – 0x0003_FFFF. If the BAM field is set to 0x0007 (512 Kbytes), then the area of memory visible at 0x0004_0000 – 0x0007_FFFF will be the same as that at 0x0000_0000.
2. If you enter a value in the BAM that does not have contiguous BAM bits set, unusual effects will be observed. For example, if the BAM is set to 0x0002, defining a theoretical block size of 0x0002_FFFF (192 Kbytes), and the base address is 0x0, the actual areas of accessible memory are as follows:

0x0000_0000 – 0x0000_FFFF = ACTIVE
0x0001_0000 – 0x0001_FFFF = **NOT ACTIVE**
0x0002_0000 – 0x0002_FFFF = ACTIVE

This relates back to what was mentioned earlier, where the BAM field is compared to the physical CPU address lines to determine the block size. In this case, BAM[1] is set, whereas BAM[0] is cleared. Therefore, the defined area is missing a block of memory defined by BAM[0].

2.2.2 WP — Write Protect

This single bit provides a very useful write protection mechanism for a memory (or other peripheral) controlled by the corresponding chip select.

If WP is set to ‘0’, reading and writing is permitted.

If WP is set to ‘1’, writing to the respective address will result in a Data Abort exception. Reads are unaffected and occur normally.

This function is ideal for emulating a non-volatile memory (e.g. FLASH) when the chip select is actually connected to SRAM.

2.2.3 Address Space Mask Bits (C/I, SC, SD, UC, UD)

There are five address space mask bits, as defined below. These bits allow the access privilege level of the corresponding chip select to be set.

C/I — CPU space and interrupt acknowledge cycle mask
SC — Supervisor code address space mask
SD — Supervisor data address space mask
UC — User code address space mask
UD — User data address space mask

If the address space mask is ‘0’, the specific access type can drive the chip select

If the address space mask is ‘1’, the specific access type is disabled from driving the chip select.

For example, if the UD bit is set, a user mode data fetch would not result in a chip select activation.

(Note that if the AM bit is ‘0’, the condition of the address space mask bits is ignored on an eDMA access)

2.2.4 AM — Alternate Master

This bit is used to determine whether or not the address space mask bits are ignored during an eDMA access of the corresponding EIM chip select.

If the AM bit is ‘0’, an eDMA access to the EIM will ignore the settings of the address space mask bits. All eDMA accesses of a valid chip select address will occur.

If the AM bit is ‘1’, an eDMA access of the EIM will occur only if the current access is permitted, based on the settings of the address space mask bits.

2.2.5 V — Valid Bit

The valid bit is used to “activate” the chip select. Note that the Valid bit (or the register containing the Valid bit), should be written last (i.e. after the other two registers have been written).

If the Valid bit is ‘0’, the chip select is not active or valid (CS0 special case described below).

If the Valid bit is ‘1’, the chip select is active.

The operation of the Valid bit is slightly different for $\overline{CS0}$ than for $\overline{CS1}$ or $\overline{CS2}$, as defined below.

For chip select $\overline{CS0}$:

$\overline{CS0}$ special case. When the Valid bit is cleared, the chip select is in global chip select mode (boot chip select), as defined in Section 4, “Boot Chip Select Operation.”

After the Valid bit has been set, it cannot be cleared again, without performing an MCU reset.

For chip selects $\overline{CS1}$ and $\overline{CS2}$:

The Valid bit can be set and cleared as required.

NOTE

To correctly validate chip select $\overline{CS1}$ or $\overline{CS2}$, the $\overline{CS0}$ Valid bit must be set. See Section 4, “Boot Chip Select Operation,” for details.

2.3 Chip Select Control Register (CSCR*n*)

The chip select control registers are 16 bits wide as defined below.

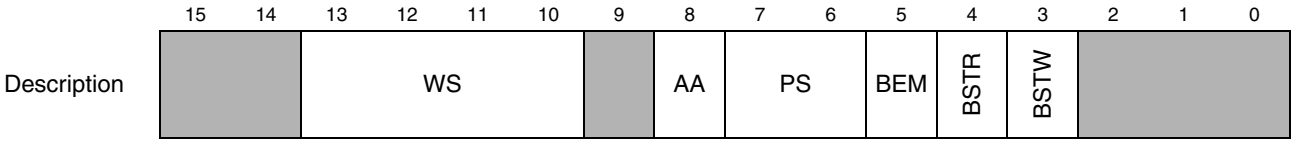


Figure 2-3. CSCR*n* Fields

2.3.1 WS — Wait States

This 4-bit field defines the number of additional wait states (clock cycles) inserted into the bus cycle before an internal $\overline{TA}$ is automatically asserted. The WS field can be set to any value between ‘0x0’ and ‘0xF’. Note that the WS field is relevant only if automated $\overline{TA}$ generation is enabled (see the description of the AA field below).

If WS = ‘0x0’, no additional wait states are inserted, and an external cycle will take 3 clock cycles

If WS = ‘0xF’, 15 additional wait states will be added, and an external cycle will take 18 clock cycles

Adding wait states allows the MAC7100 to interface to external peripherals with a slow response time. At the maximum system clock frequency of 50 MHz, 15 wait states equates to 300 ns.

See Section 5, “EIM Performance and Timing Diagrams,” for details on wait states and the implications for performance.

2.3.2 AA — Auto Acknowledge

As previously mentioned in Section 1.3.3, "Transfer Acknowledge," the EIM can be configured such that the transfer cycle is terminated by an internal, automatically generated $\overline{TA}$ (transfer acknowledge) signal, rather than requiring an external hardware $\overline{TA}$.

If AA is '0', automatic internal $\overline{TA}$ signal generation is disabled, and the $\overline{TA}$ signal must be supplied from an external device (fed into the MCU $\overline{TA}$ pin).

If AA is '1' then the EIM generates its own $\overline{TA}$ signal, based on the number of wait states defined by the WS field above.

Note that, even when AA is enabled, the external $\overline{TA}$ signal is still active and can be used to terminate the current cycle. This must be taken into consideration at the hardware design stage.

2.3.3 PS — Port Size

The port size determines the width of the data bus associated with the chip select. On the MAC7100 family, there are only two possible values for this, 8-bit and 16-bit.

Table 2-3. Port Size Configuration

Required Port Size	Value of PS[1..0]	Comments
8-bit	01	Data is driven on Data[15..8]
16-bit	1X	
32-bit	00	Invalid configuration on MAC7100 family

If an access is performed to the EIM such that the data width is larger than the port size (e.g. a 32-bit access of a 16-bit port), then back-to-back accesses will be performed. Similarly, if an access is performed on the EIM such that the data width is smaller than the port size, the byte select signals are used to determine the appropriate laning, as defined in Section 1.3.5, "Byte Select."

Note that misaligned accesses are NOT permitted. All accesses must be aligned on the port size.

2.3.4 BEM — Byte Enable Mode

This bit determines whether the byte select signals are asserted for reads as well as for writes. The required value of this bit will be determined by your specific external memory requirements.

If BEM = '0', the byte select signals are asserted for write operations only.

If BEM = '1', the byte select signals are asserted for read and write operations.

2.3.5 BSTR — Burst Read Enable

This bit determines if the associated chip select is configured to perform burst reads.

If BSTR = '0', the associated chip select will not perform burst reads.

If BSTR = '1', burst reads are enabled for the associated chip select. 16-bit reads from an 8-bit port and line reads will be accessed as a burst.

NOTE

This application note does not cover the operation of burst mode for the MAC7100, for reasons previously detailed.

2.3.6 BSTW — Burst Write Enable

This bit determines if the associated chip select is configured to perform burst writes.

If BSTW = ‘0’, the associated chip select will not perform burst writes.

If BSTW = ‘1’, burst writes are enabled for the associated chip select. 16-bit writes to an 8-bit port and line writes will be performed as a burst.

NOTE

This application note does not cover the operation of burst mode for the MAC7100 as previously detailed.

3 Configuring the EIM

This section pulls together the information discussed in the previous sections (signal and register descriptions), and shows how to configure the EIM from a user’s perspective.

The use of asynchronous memories will be the focal point of this section, as these memories are readily available and commonly used in embedded applications. As the name suggests, asynchronous memories do not require a clock input from the MCU, allowing the EIM CLKOUT pin to be disabled, to reduce EMC related noise. See Section 6.1, “Disabling EIM Signals in Expanded Mode.”

Asynchronous memories have very tightly defined read/write cycle times — it will be known exactly how long a read or write operation will take (e.g. 15 ns) from the point the EIM issues the correct control signals. These memories do not typically use hardware $\overline{TA}$ control, so automatic $\overline{TA}$ generation is required.

NOTE

Synchronous memories are generally larger, higher performance devices with a burst interface. Since the MAC7100 EIM does not fully support burst operation, interface to these memory types will not be discussed in this application note.

3.1 Design Check List

Before a design is started that uses an external memory system, the following items must be considered.

Memory Configuration: Will the memory be 8 or 16-bit wide? How much external memory is required? Note that the EIM port width must match the external memory configuration. You cannot correctly access an external 16-bit memory with an EIM port size of 8-bits and vice-versa.

Compatibility: Most asynchronous memories are compatible with the MAC7100 EIM, however the timing diagrams must be checked for compatibility. In particular, the order that the

control signals are asserted and de-asserted is important as asynchronous memories use these to determine when to start and (more importantly) stop the cycle.

The voltage levels and thresholds must also be verified as compatible. For example, a 3.3 V memory is unlikely to work when the MAC7100 is configured for 5 V I/O. See the *MAC7100 Microcontroller Family Hardware Specifications* document (MAC7100EC, available at www.freescale.com/mac7100) for details.

Speed: It is important to consider the desired speed of your external memory system. Asynchronous memories have a huge range of operating speeds, with SRAM devices being significantly faster than FLASH. A memory access time is normally quoted from the time the memory receives the valid chip select to the time data is available (for a read). The EIM can provide additional wait states (see Section 2.3, "Chip Select Control Register (CSCRn)"), to meet this timing.

3.2 Example 1: 8-bit Asynchronous SRAM

This example looks at a very simple configuration: interfacing a single, 128 Kbyte, 8-bit asynchronous SRAM, mapped at base address 0x0000_0000 and controlled by chip select $\overline{CS0}$.

3.2.1 Hardware Configuration

The following EIM signals are required to interface to the SRAM.

Addr[16..0] To address 128 Kbytes, 17 EIM address lines are required (2^{17} = 128 Kbytes).

Data[15..8] Since a single 8-bit memory is being interfaced, the EIM port size must be configured as 8 bits. As previously discussed, when the EIM is configured as an 8-bit port, data is routed to Data[15..8].

$\overline{BS1}$ Byte select $\overline{BS1}$ is active during an 8-bit port EIM cycle.

$\overline{OE}$ Output enable $\overline{OE}$ allows the SRAM to drive data onto the data bus at the correct point during a read cycle.

$\overline{CS0}$ $\overline{CS0}$ enables the SRAM automatically when an access is made to the correct address

R/$\overline{W}$ SRAM write enable signal (differentiates between read and write)

Figure 3-1. 8-bit External SRAM Example

NOTE

Most 8-bit SRAM devices will not have a “byte enable” input. This is shown for completeness. The signal names on the SRAM are likely to differ slightly from those shown.

3.2.2 Software Configuration

This section assumes that the interface between the EIM and the SRAM has already been verified as compatible.

One of the first things to calculate is the number of wait states required to allow read and write access to the SRAM. Figure 3-2 shows a simplified version of the MAC7100 EIM bus timings, taken from the *MAC7100 Microcontroller Family Hardware Specifications* document (MAC7100EC). Note that this is relevant only for configurations with auto acknowledge enabled.

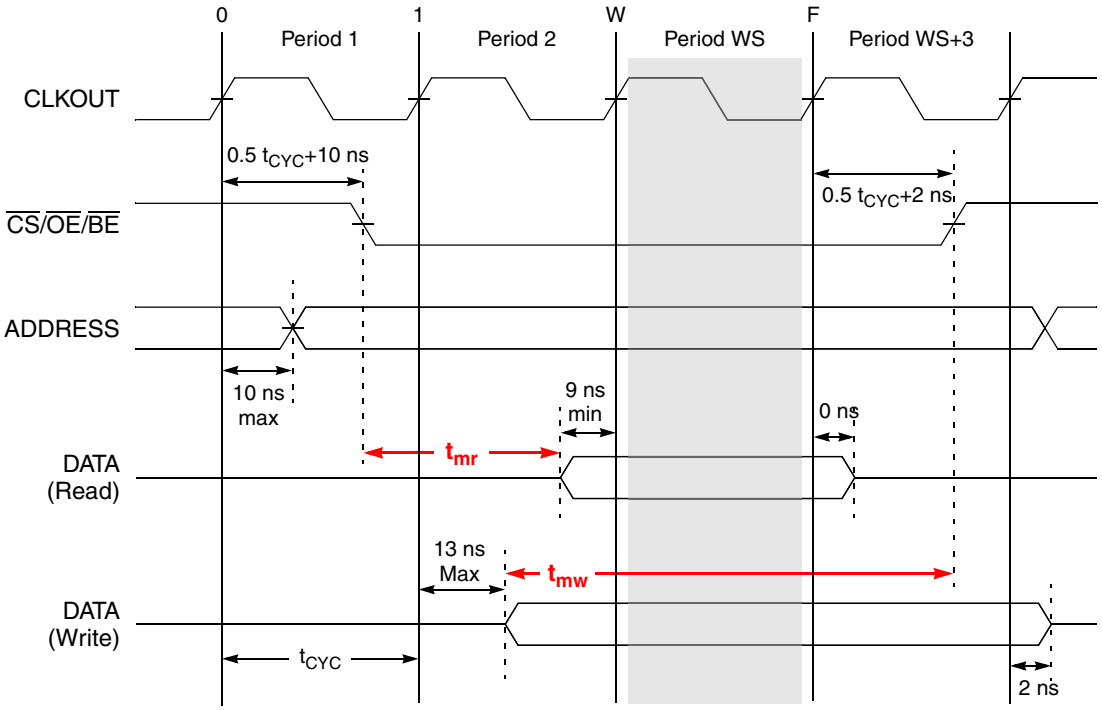


Figure 3-2. MAC7100 EIM Simplified Bus Timing

As can be seen, a complete EIM transfer cycle is made up of multiple CLKOUT cycles or periods. One CLKOUT period is defined as t_{cyc}. “Period WS” shows the area where wait states are inserted into the EIM transfer cycle if required. Note that the rising edge of CLKOUT is the “active edge” and is always used for data latch and timing reference.

If we assume that the MAC7100 CLKOUT is 40 MHz (t_{cyc} = 25 ns), and an EIM access to a valid address has been started, then:

1. The address lines become valid (correct address) at most 10 ns from edge 0 (period 1).
2. The control signals ($\overline{CS}/\overline{OE}/\overline{BE}$) are valid 22.5 ns from edge 0 (period 1).

For a **Data READ** — The memory will return data X ns later, where X is the speed of the memory. For this data to be successfully latched into the MCU, it must be returned (valid) in time t_{mr}, 9 ns BEFORE edge W. If this is not possible, wait states must be added. To check if a wait state is required, calculate the time between the EIM control signals activating and the data read setup time. For 40 MHz CLKOUT, calculating t_{mr} for 0 wait states gives:

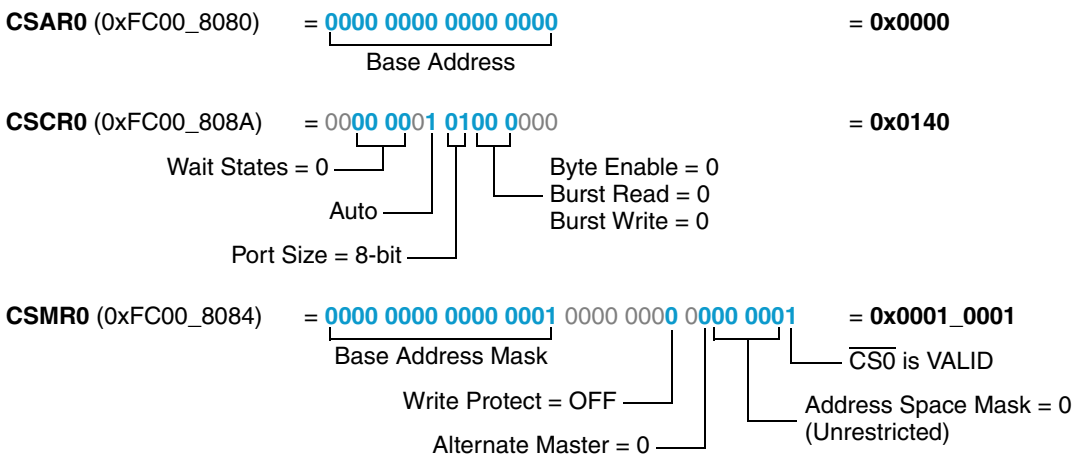
$$t_{mr} = (t_{cyc} - (0.5t_{cyc} + 10)) + (t_{cyc} - 9) + (\text{num wait states} \times t_{cyc}) = 18.5 \text{ ns} \qquad \textit{Eqn. 2}$$

Therefore, any memory with a read time of less than 18.5 ns would function correctly without adding any wait states. If we had a slow memory, of 100 ns, this would require four wait states to read correctly (18.5 + 25 + 25 + 25 + 25).

For a **Data WRITE** — The external memory requires data to be available roughly X ns (where X is the speed of the memory) before the de-assertion of the control signals (the exact details of this are memory dependent). In Figure 3-2, this corresponds to time t_{mw}. Again, if this is not attainable, wait states must be added.

$$t_{mw} = (t_{cyc} - 13) + (\text{num wait states} \times t_{cyc}) + (0.5t_{cyc} + 2) = 26.5; \text{ 40 MHz CLKOUT, 0 wait states} \qquad \textit{Eqn. 3}$$

In our example, we have a 12 ns memory, so no wait states are required (read or write) for operation at 40 MHz. The configuration of the chip select is therefore:



Note that the SCMR0 register (with the Valid bit) is written last.

3.3 Example 2: 16-bit Asynchronous SRAM

To make the most of the external interface, the full 16-bit data port should be used. This example shows how to connect an external asynchronous SRAM to the EIM in this configuration. It is assumed that chip select $\overline{CS0}$ has already been configured and validated for another peripheral, therefore chip select $\overline{CS1}$ will be configured to control this memory, with base address 0x0040_0000.

3.3.1 Hardware Configuration

The hardware configuration is slightly different than the previous example. With a 16-bit SRAM, the least significant address bit (normally called Addr[0] on the SRAM) is 16-bit aligned, and therefore connected to EIM address line Addr[1].

There are two possible hardware implementations of a 16-bit memory, using either a single 16-bit device or two 8-bit devices connected together as shown in Figure 1-3. The operation is identical in either case. To provide correct byte access over the 16-bit port, the byte select signals are required.

In this example, a 256 Kbyte memory is shown, requiring 18 EIM address lines, Addr[17..0]. Addr[0] is however not connected as detailed above.

Figure 3-3. 16-Bit External SRAM Example

NOTE

If the external memory system is configured for 16-bit data, as shown above, this will only function with a 16-bit EIM port size. If the EIM is re-configured as an 8-bit data port and used with a 16-bit memory system, half of the data will be lost since A0 is not connected.

Table 3-1 details some examples of asynchronous SRAM devices that may be interfaced to the MAC7100 EIM without glue logic. These devices have been successfully used on the Freescale MAC7100EVB. Note that these devices are 5 V I/O.

Table 3-1. Sample SRAM Part Numbers

Device Part Number	Size	Configuration
Cypress CYC1020B	64 KBytes	32 K x 16-Bits
Cypress CYC1021B	128 KBytes	64 K x 16-Bits
IDT IDT71016	128 KBytes	64 K x 16-Bits

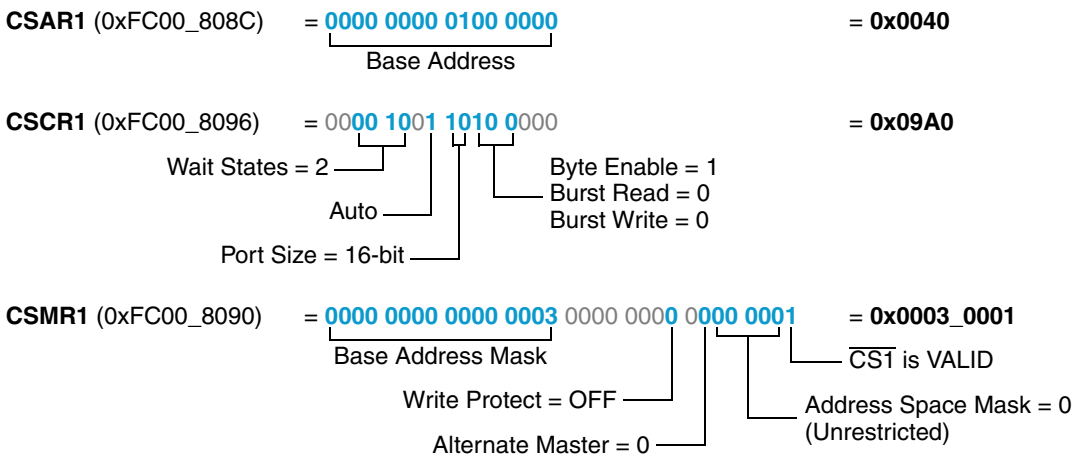
3.3.2 Software Configuration

It is assumed that the number of required wait states have already been calculated, based on the CPU system frequency and the bus timings. For the purposes of the example, the number of wait states will be set to 2.

The configuration of the chip select registers is very similar to the first example, with the following difference:

- Wait States = 2
- Base Address = 0x0040_0000
- Port Size = 16-bit
- Base Address Mask = 0x0003 (256 Kbytes)

The registers are configured as shown below. Note this time, $\overline{CS1}$ is used.



NOTE

The memory range defined for chip select $\overline{CS1}$ does not overlap the previous chip select $\overline{CS0}$ memory range. This is a requirement of the configuration. Overlapping memory ranges must be avoided or unpredictable results will occur.

3.4 Example 3: 16-bit Asynchronous FLASH

The final example shows how to connect an external 16-bit asynchronous FLASH memory. To complement the other examples, this will be configured to use chip select $\overline{CS2}$, with a base address of 0x0080_0000. Again, it is assumed that chip select $\overline{CS0}$ is already activated.

3.4.1 Hardware Configuration

The hardware configuration is very similar to example 2, with one exception. Typically, external asynchronous FLASH devices do not support byte writes to a 16-bit device and therefore have no byte enable inputs. Byte reads are however supported, since the MCU will simply ignore the irrelevant data.

A typical asynchronous flash will be between 256 Kbytes and 2 Mbytes in size. In this case, a 512 Kbyte device is shown, requiring 19 EIM address lines, Addr[18..0]. As with the previous example, the FLASH is addressed in 16-bit mode, so EIM address line Addr[0] is not connected.

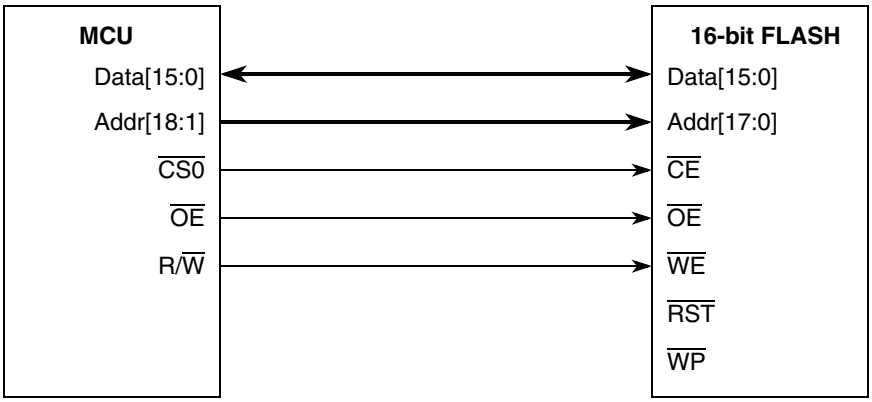


Figure 3-4. 16-bit External FLASH Example

Usually, the asynchronous external FLASH device will also have some additional inputs which are described below:

1. **Reset** (RST) — Connected to the "reset-out" of the MCU. Check that the active state of MCU and FLASH reset are the same.
2. **Write Protect** (WP) — This input pin provides a hardware write protection mechanism, if required.
3. **BYTE** — Some 16-bit FLASH devices allow you to select between 8-bit and 16-bit operating modes. Due to the pins involved, this is normally a design-time decision, rather than something that can be re-configured once implemented.

Table 3-2 details some examples of asynchronous FLASH devices that may be interfaced to the MAC7100 EIM without any glue logic. As with the SRAM, these devices have been successfully used on the Freescale MAC7100EVB. Note that these devices are 5 V I/O.

Table 3-2. Sample FLASH Part Numbers

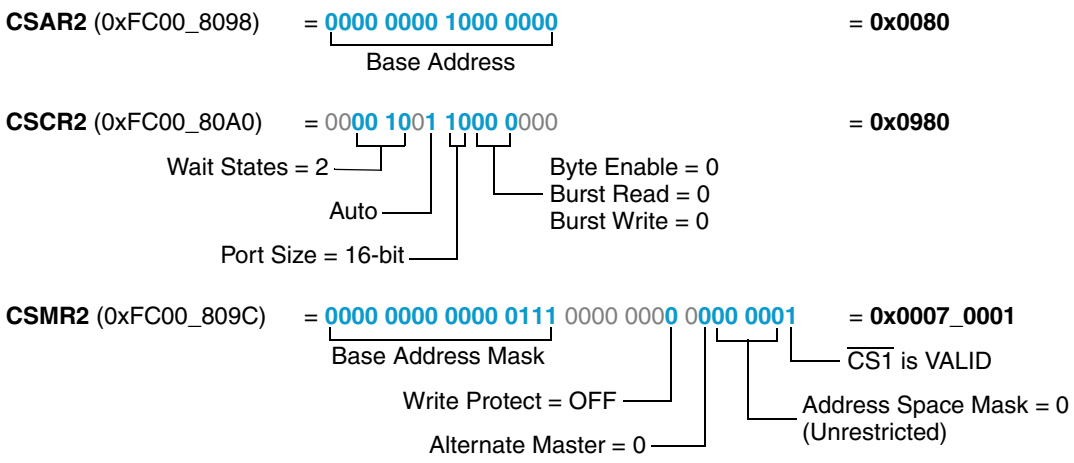
Device Part Number	Size	Configuration Options
AM29F400B	512 Kbytes	512 K x 8-bits / 256 K x 16-bits
AM29F800B	1 Mbytes	1 M x 8-bits / 512 K x 16-bits
AM29F160D	2 Mbytes	2 M x 8-bits / 1 M x 16-bits

3.4.2 Software Configuration

Asynchronous FLASH memory is typically much slower than SRAM. (A "fast" FLASH device would be defined as one that had an access time of less than 60 ns.) This means that the number of wait states used will be greater than that for SRAM. If we assume a 60 ns FLASH device and a 40 MHz CPU speed, two wait states are required (see Section 3.2.2, "Software Configuration," for details).

For 512 Kbytes, the address range is 0x0000_0000 – 0x0007_FFFF, corresponding to a base address mask of 0x0007 (see Section 2.2, "Chip Select Mask Registers (CSMRn)," for details)

The register configuration for this example is therefore:



3.5 Important Configuration Notes

To reiterate the critical configuration points:

1. There must be NO overlapping memory ranges between the chip selects; otherwise, the chip selects will not behave as expected. The exact behaviour is dependent on the particular configuration.
2. Chip select $\overline{\text{CS0}}$ must have been manually configured and activated before $\overline{\text{CS1}}$ or $\overline{\text{CS2}}$ can be used (see Section 4, "Boot Chip Select Operation," for more details).

4 Boot Chip Select Operation

The MAC7100 EIM supports boot operation from an external memory mapped peripheral such as external FLASH.

When the MCU is reset into expanded mode, chip select $\overline{\text{CS0}}$ is configured as the "Boot Chip Select". This provides a default external memory space at base address 0x0, allowing the MCU to start execution at external address 0x0 in the normal manner. To boot successfully from an external memory:

1. Chip select $\overline{\text{CS0}}$ must be connected to the external boot memory.
2. External memory address 0x0 must contain a branch instruction to the reset handler.

To provide as much flexibility and choice as possible for the type of external boot memory used, the boot chip select is configured by default in burst inhibit (asynchronous) mode, with the maximum number of wait states. The port width and auto acknowledge settings are user configurable using a reset configuration pin as shown in Table 4-1.

Table 4-1. Chip Select $\overline{\text{CS0}}$ Reset Configuration

MCU Pin	EIM Reset Function	Reset Value
PA14 / Data14 / PS	Determines size of port	0 = 8-bit port 1 = 16-bit port
PA15 / Data15 / AA	Determines if auto $\overline{\text{TA}}$ is enabled	0 = $\overline{\text{TA}}$ Supplied from external peripheral 1 = EIM supplies $\overline{\text{TA}}$ signal (auto acknowledge)

Chip select $\overline{CS0}$ remains active and in the boot chip select state until it is manually configured and the Valid bit set in the CSMR0 register (see Section 2.2, "Chip Select Mask Registers (CSMRn)"). Up to this point, $\overline{CS1}$ and $\overline{CS2}$ will not function.

NOTE

After the $\overline{CS0}$ Valid bit has been written, it can be cleared again ONLY by an MCU reset.

As the boot chip select is configured for maximum compatibility, with the maximum number of wait states, the performance of the external memory may be limited. It may be necessary to reconfigure chip select $\overline{CS0}$ after the system has booted. To do this, it is advised to copy a program, from the boot device into internal RAM, that will do this re-configuration. Operation would then be transferred to internal RAM, the chip select re-configured, and operation restored to external memory. This avoids any potential synchronisation problems.

5 EIM Performance and Timing Diagrams

As detailed previously, the MAC7100 EIM requires at least three clock cycles per asynchronous transfer. If fast termination is used (see the MAC7100 Reference manual for details), this can be reduced to two cycles, but this restricts the types of memories that can be used. If a 32-bit access is performed from internal memory, this will take one or two clock cycles (depending on the memory accessed and whether a FLASH access is sequential — see the MAC7100 Reference Manual). If the same type of access is made to an external 3-cycle memory, this mandates two back-to-back 16-bit accesses, requiring six clocks.

As a reference point, it is often useful to see actual timing diagrams for the EIM. These can be used as a baseline for comparison with your own external bus configuration, and they help to describe the signals in more detail.

To capture the bus timings, a small piece of test code was run from internal MCU RAM. This wrote and then read back, from external SRAM at address 0x100, a 32-bit value (0x0123_4567). 8-bit, 16-bit and 32-bit writes and reads were performed with the EIM configured as a 16-bit port. The effect of adding additional wait states was also investigated. A logic analyser was used to capture the results, triggering on either the falling or rising edge of the read/write line, R/$\overline{W}$.

Table 5-1. EIM Test Data

	Address 0x100	Address 0x101	Address 0x102	Address 0x103
Data	01	23	45	67

NOTE

All timing diagrams have been captured using auto acknowledge enabled.

5.1 EIM Test Code

The EIM test code is shown below. This is extremely simple and involves using a cast to write then read data to/from external memory. For each test, the relevant read/write section was uncommented. Note that the actual chip select configuration is not shown, as this was set up within a debugger script environment.

```
typedef unsigned int        uint32_t;
typedef unsigned short      uint16_t;
typedef unsigned char       uint8_t;

int main(void)
  {
   volatile uint32_t readval_32;
   volatile uint16_t readval_16;
   volatile uint8_t readval_8;

    // 8-Bit Data Write
    *((uint8_t *) 0x100) = 0x01;
    *((uint8_t *) 0x101) = 0x23;
    *((uint8_t *) 0x102) = 0x45;
    *((uint8_t *) 0x103) = 0x67;

    // 16-Bit Data Write
    // *((uint16_t *) 0x100)= 0x1234;
    // *((uint16_t *) 0x102)= 0x4567;

    // 32-Bit Data Write
    // *((uint32_t *) 0x100)= 0x01234567;

    // 8-Bit Data Read
    readval_8 = *(( uint8_t *) 0x100);
    readval_8 = *(( uint8_t *) 0x101);
    readval_8 = *(( uint8_t *) 0x102);
    readval_8 = *(( uint8_t *) 0x103);

    // 16-Bit Data Read
    // readval_16 = *(( uint16_t *) 0x100);
    // readval_16 = *(( uint16_t *) 0x102);

    // 32-Bit Data Read
    // readval_32 = *((uint32_t *) 0x100);
  }
```

Figure 5-1. EIM Write / Read Test Code

5.2 8-bit Write / Read

In this case, the test data 0x0123_4567 was written, then read back as individual 8-bit values. As with all of the examples, the EIM is configured as a 16-bit port.

Test Case 1 (8-bit Write) — Figure 5-2 shows the first test case. This reflects the boot chip select configuration, with the maximum number of wait states (0xF) added to the cycle. The logic analyser was set to trigger on the first write. This illustrates the ability of the MAC7100 EIM to cope with extremely slow external peripherals. The maximum number of wait states in combination with the 8-bit data width illustrate the worst case scenario for automated $\overline{TA}$ transfers.

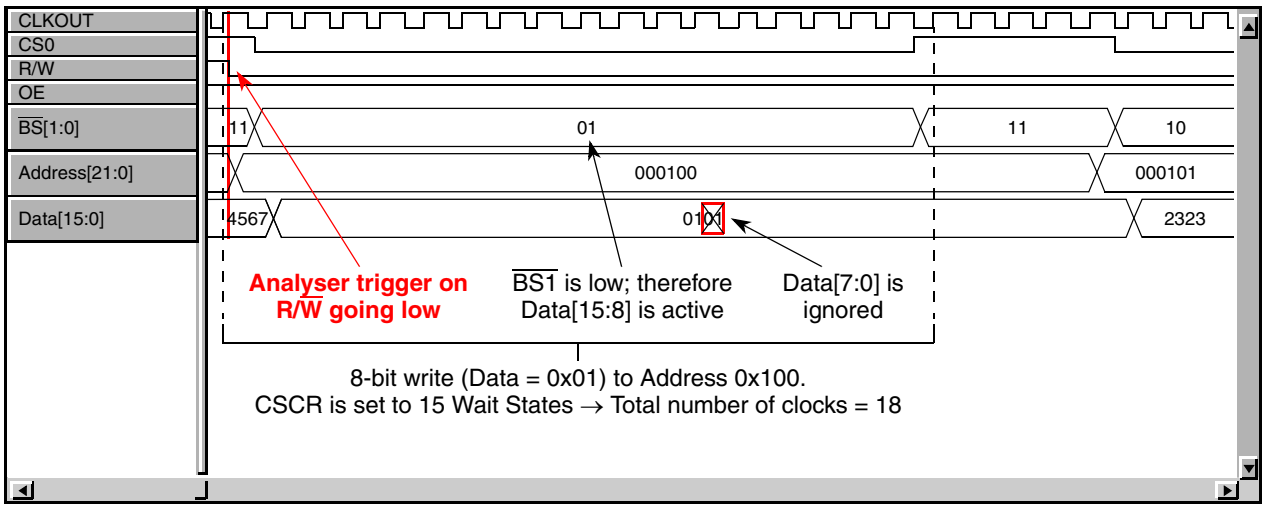


Figure 5-2. Test Case 1 (8-bit Write to 0x100 With Default $\overline{CS0}$ Configuration, Maximum Wait States)

There are several things to note in this screen shot:

- Although the data access is 8-bit, the EIM is still configured as 16-bit. This means that the byte select signals (shown as BS on the screen shot) will enable upper and lower bytes as defined in Section 1.3.5, "Byte Select." For the first write to 0x100, byte select $\overline{BS1}$ is active, and the data valid on Data[15..8]. Data[7..0] is ignored by the SRAM.
- There is a gap between the first and second byte transfers, where the chip select is high (deactivated). This corresponds to the time where the core is fetching the next instructions, setting up the next 8-bit write.

Test Case 2 (8-bit Write) — Figure 5-3 details a "zoomed-out" version of Figure 5-2. This time, all four 8-bit writes are shown. Note the substantial number of clocks required to complete the full 32-bit transfer.

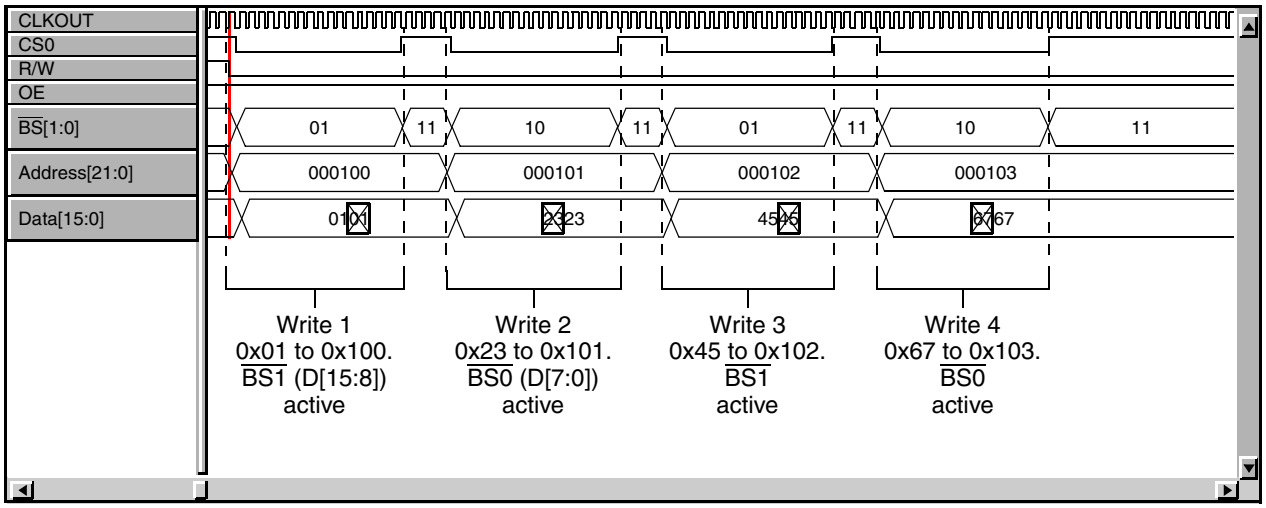


Figure 5-3. Test Case 2 (4 x 8-bit Write With Default $\overline{CS0}$ Configuration, Maximum Wait States)

Note the byte select signals changing for odd and even accesses. The ignored data has been lightly crossed out for illustrative purposes.

Test Case 3 (8-bit Read) — Figure 5-4 shows the full 32-bit read sequence, this time with only a single wait state added to the chip select transfer cycle. As can be seen, this significantly speeds up the process, with only four clock cycles required instead of 18. In this case, the analyser was set to trigger on the read/write line, R/$\overline{W}$, going high.

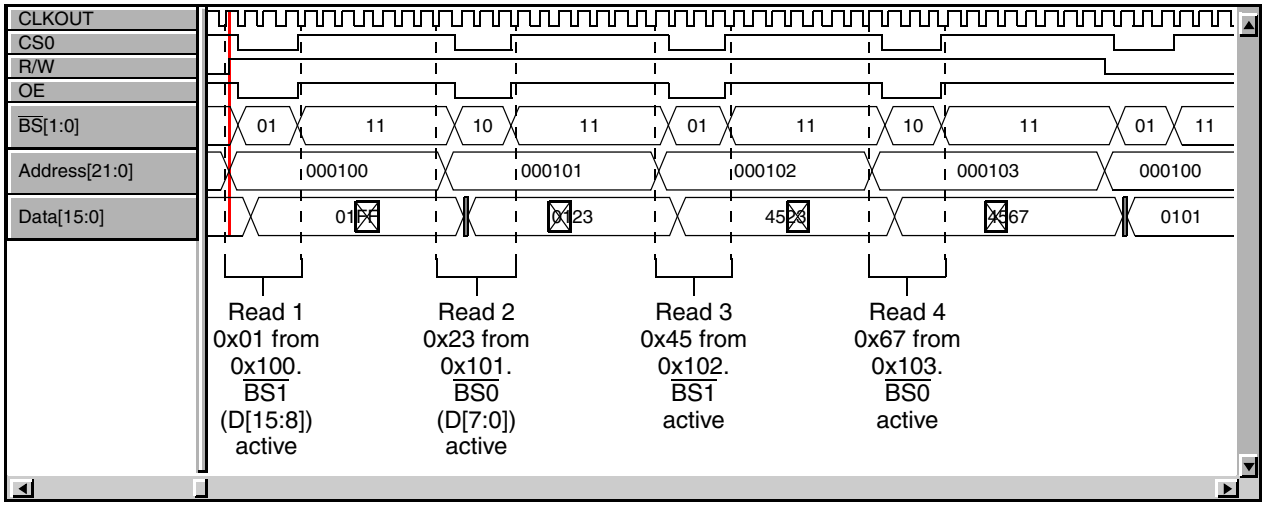


Figure 5-4. Test Case 3 (4 x 8-bit Reads with 1 Wait State)

5.3 16-bit Write/Read

In these examples, the data was written, then read back, as 16-bit values. The EIM port is configured as 16-bit wide.

Test Case 4 (16-bit Write) — Figure 5-5 details two 16-bit writes with a single wait state. Note that, this time, the byte select signals are both low; therefore, all data is valid on the bus.

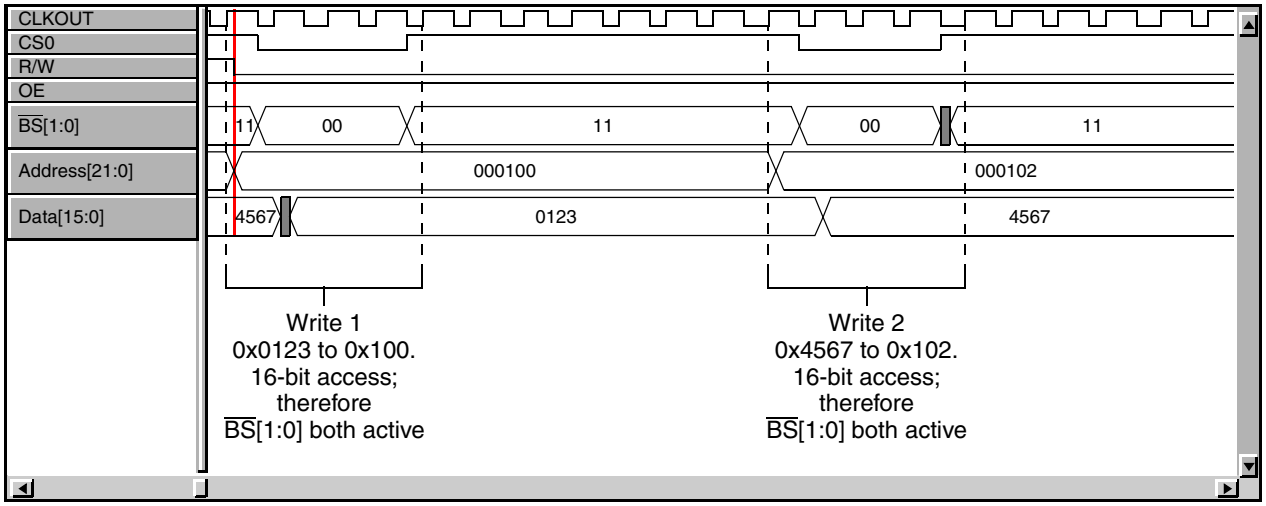


Figure 5-5. Test Case 4 (2 x 16-bit Writes with 1 Wait State)

Test Case 5 (16-bit Read) — Figure 5-6 details two 16-bit reads with a single wait state.

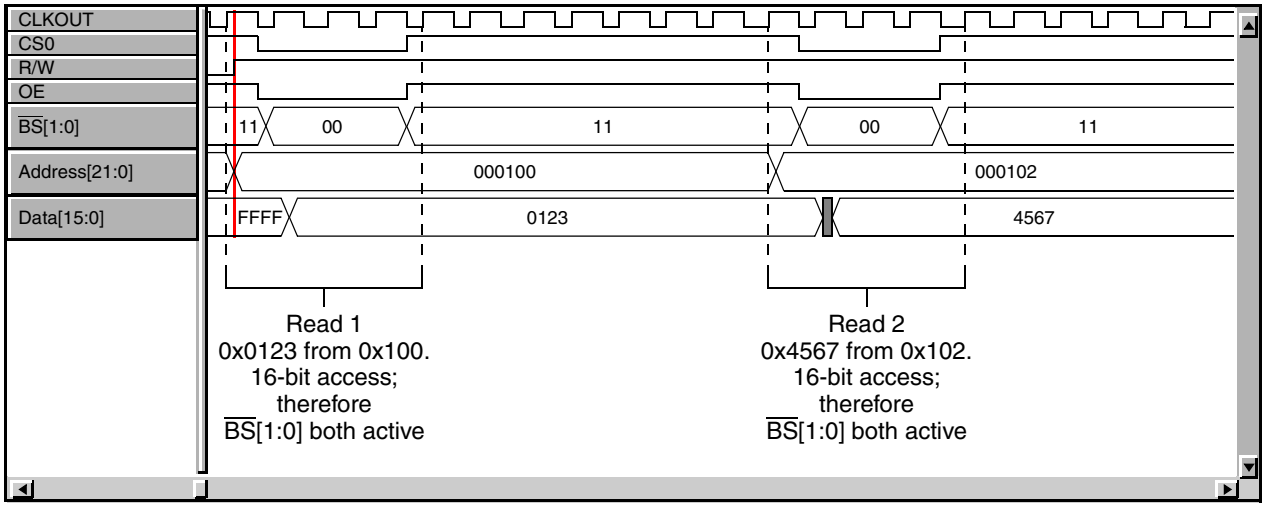


Figure 5-6. Test Case 5 (2 x 16-bit Reads with 1 Wait State)

5.4 32-bit Write/Read

In these examples, the data was written then read-back as 32-bit values, with the EIM configured as a 16-bit port. In this case, a 32-bit operation is performed as two back-to-back 16-bit accesses. This is noticeably different from two 16-bit transfers in that there is no delay between transfers and the read/write line, R/$\overline{W}$, is asserted for the complete 32-bit transfer.

Test Case 6 (32-bit Write) — Figure 5-7 shows a 32-bit write with a single wait state. Note this is completed in 8 clock cycles.

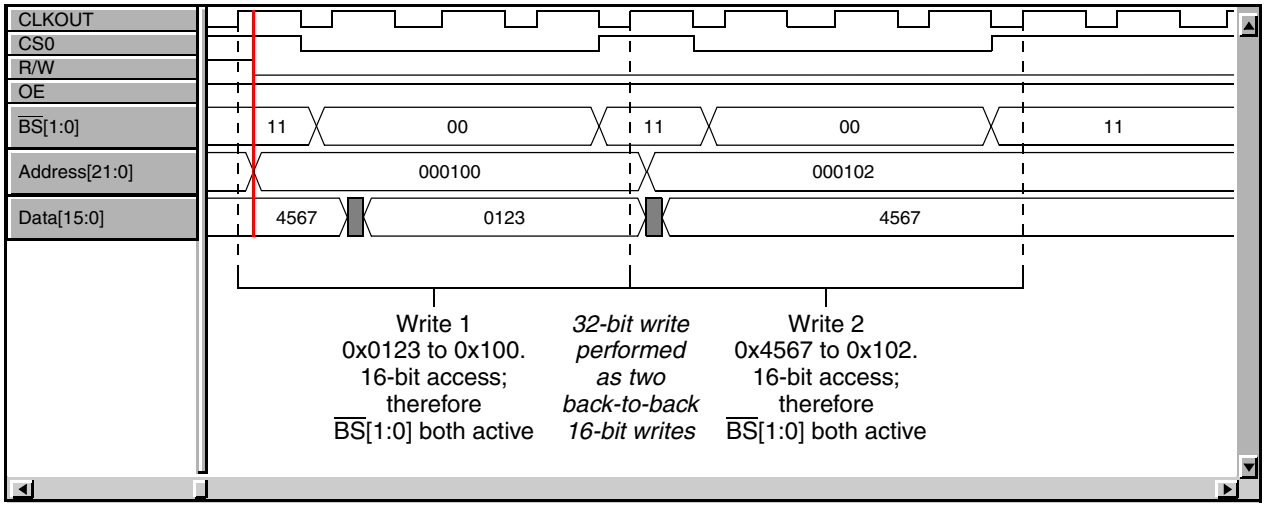


Figure 5-7. Test Case 6 (32-bit Write to 16-bit Port with 1 Wait State)

Test Case 7 (32-bit Read) — Figure 5-8 shows a 32-bit write with a single wait state.

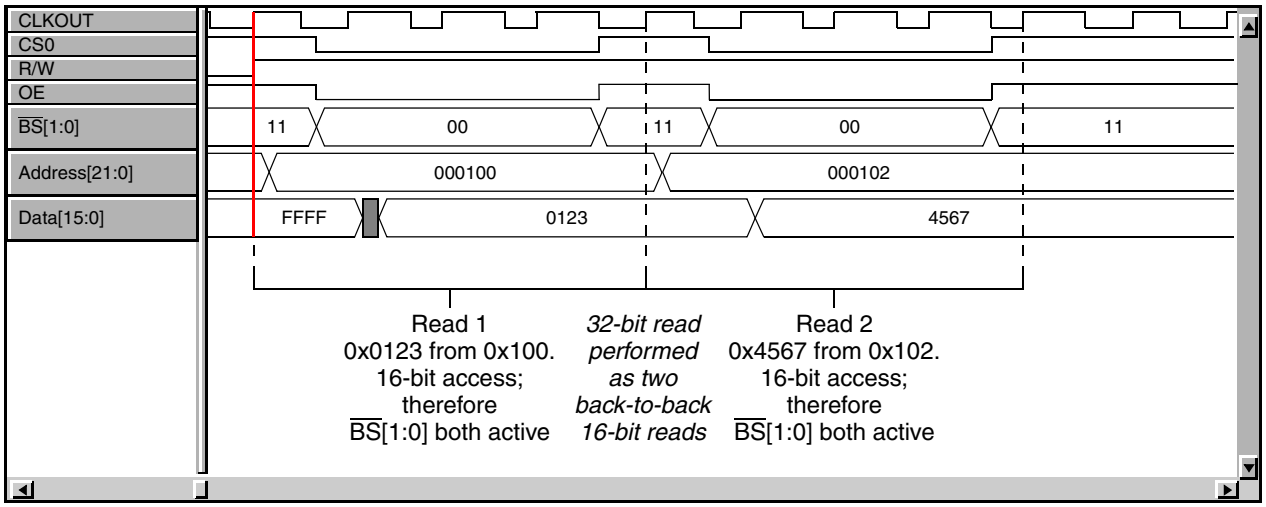


Figure 5-8. Test Case 7 (32-bit Read to 16-bit Port with 1 Wait State)

6 Additional Configuration

This section details some additional information for configuring the EIM in advanced modes.

6.1 Disabling EIM Signals in Expanded Mode

It is likely that not all of the EIM signals will be required for your external bus configuration. For example, the data bus width may be only 8-bit, or it may be required to disable the CLKOUT signal to reduce EMI. In these cases, the unused EIM ports can be reclaimed easily for use as GPIO.

This is achieved by clearing the MODE bit of the respective port pin configuration register within the PIM (Port Integration Module). See the MAC7100 Reference Manual for more details.

6.2 Single Chip Mode Operation

The EIM can be used when the MCU is in single chip mode. This is particularly useful if you wish to boot from internal FLASH but still gain access to an external peripheral.

There are three main differences between single chip mode and expanded mode operation with respect to the EIM configuration:

1. In single chip mode, the ports used for the EIM signals (ports A, C and D) default to GPIO mode and not peripheral (EIM) mode.
2. In single chip mode, the EIM clock system is disabled by default. This power saving feature allows the EIM module to be effectively disabled when not required.
3. In single chip mode, the EIM and FLASH are re-mapped so the EIM is located at 0x2000_0000 (base address) and the FLASH is at 0x0000_0000.

Therefore, to use the EIM in single chip mode you must:

1. Configure all required EIM signals to peripheral mode, by setting the relevant MODE bit high in the port pin configuration register (in the PIM).
2. Enable the EIM module, by setting the EIMCLKEN bit in the PIM configuration register (in the PIM).
3. Set the chip select base address as 0x2000_0000 + required offset.

The following example code (Figure 6-1) uses a loop to configure all pins of ports A, C, and D to peripheral mode and enable the EIM clock. Note that PortD[4..3] are actually interrupt signals, which are not relevant for EIM configuration. These are set as a consequence of using a loop to set ports A, C, and D to peripheral mode. This must be taken into account, if using this code as a template.

```
typedef unsigned shortuint16_t;
typedef unsigned charuint8_t;

#define PIMCONFIG 0xFC0E83C2                    // PIM Global Configuration Register
#define CONFIG0_A 0xFC0E8000                    // PortA[0] Config Register
#define CONFIG0_C 0xFC0E8080                    // PortC[0] Config Register
#define CONFIG0_D 0xFC0E80C0                    // PortD[0] Config Register

int main(void)
 {
   uint8_t bit;

   *((uint16_t *) PIMCONFIG)= 0x0002;          // Enable EIM Clock (PIMCONFIG Register)

   for (bit=0; bit<0x16; bit++)
    {
       *((uint16_t *) CONFIG0_A+bit)= 0x0080;  // Set PortA[0..15] to peripheral mode
       *((uint16_t *) CONFIG0_C+bit)= 0x0080;  // Set PortC[0..15] to peripheral mode
       *((uint16_t *) CONFIG0_D+bit)= 0x0080;  // Set PortD[0..15] to peripheral mode
    }

   while(1);
} // End Of Main
```

Figure 6-1. Enabling EIM in Single Chip Mode

6.3 Single Chip Mode Default Chip Select

The “Boot Chip Select” configuration is still valid for single chip operation as described in Section 4, “Boot Chip Select Operation,” for expanded mode, however in single chip mode, the base address is 0x2000_0000. In this respect, the “boot chip select” is more of a “default chip select”, as the MCU chip select has no boot functionality.

As with expanded mode operation, the single chip mode default chip select remains active until the Valid bit is set. $\overline{CS0}$ must be configured and validated before $\overline{CS1}$ and $\overline{CS2}$ can be used.

6.4 External Bus Buffering

The MAC7100 has a maximum drive capacitance of approximately 30 pF. This means that for each of the address, data, and EIM control signals, the total capacitance loading per pin must not exceed 30 pF. If this loading is exceeded, the EIM will operate out of spec, and bus timings cannot be guaranteed. Most external memory devices have input capacitances of between 7 pF and 15 pF; so, if there are multiple peripheral devices connected to the EIM, it may be necessary to use buffers to reduce the loading on the bus.

There is a vast array of different buffers available, with varying voltages, propagation delays, data widths, single or bidirectional data flow, etc. One interesting point is that buffers can also be used for voltage translation, allowing a 3.3V memory to interface to a 5V MCU and vice versa. This, therefore, opens up a new array of peripheral choice.

Figure 6-2 details the connection of a set of buffers in a typical external memory system. Note that for the data bus, the buffers are bidirectional. This is controlled simply by the MCU read/write line, R/$\overline{W}$.

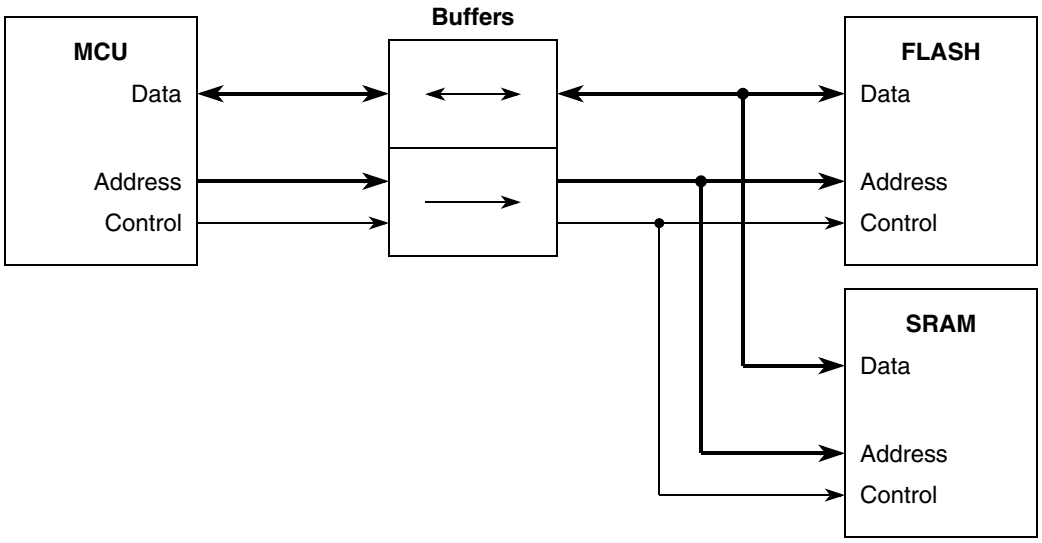


Figure 6-2. EIM Buffering

NOTE

When calculating the total capacitance of a signal, you must take the PCB layout into consideration, particularly if connectors are present in the signal.

30 pF is stated as a reference point for the MAC7100 EIM maximum capacitance loading. Consult the *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for accurate figures. This should be consulted before starting any design work.

Table 6-1 lists some example buffers.

Table 6-1. Example Buffers

Buffer	Direction	Size	Voltage
On Semi MC74LCX16244	Unidirectional	16-bit	3.3 V with 5V tolerant inputs
On Semi MC74LCX16245	Bidirectional	16-bit	3.3 V with 5 V tolerant inputs

6.5 Real Time Trace

MAC7100 real time trace information is supplied via the on-chip Nexus 2+ debug module. The MAC7100 EIM does not support show cycles; so, internal CPU cycles are not visible over the external bus.

7 Conclusion

Having read this application note, you should now have a good understanding of the MAC7100 EIM. In particular, you should be able to create your own tailored hardware and software configurations to perform the required functions. With screen shots of successful EIM transfers, this provides a good reference point, should any debug work be required.

For more information on the Freescale MAC7100 family, see www.freescale.com/mac7100.

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

AN2934
Rev. 0
8/2005