Freescale Semiconductor
Application Note

AN3773
Rev. 0, 02/2009

Using the External Bus Interface (EBI) on the MPC5510

by: Alasdair Robertson
Applications Engineering
Microcontroller Solutions Group, East Kilbride

1 Introduction

The Freescale Semiconductor MPC5510 family of microcontrollers has an optional EBI (External Bus Interface) that can be used to interface to external memory or peripherals such as memory mapped displays.

This application note details the operation of the EBI. It should provide a fundamental understanding of the EBI operation as well as how to configure and interface the EBI to your chosen peripheral. An example configuration is given for external asynchronous SRAM.

This application note is not intended to replace the MPC5510 reference manual and should be read alongside the reference manual. The latest reference manual and other device information is available at www.freescale.com/mpc55xx.

Contents

2 EBI Description

The MPC5510 EBI module is located on the slave side of the crossbar and can be accessed by either of the cores or by the eDMA controller. The MPC5510 implementation of the EBI supports the following key features:

- Up to 32 data lines and 24 address lines giving a maximum addressable block size of 16 MBytes
- Multiplexed address and data pins to minimize the number of MCU pins required for an external bus implementation
- Configurable data port size of 32-bits or 16-bits
- Full support for asynchronous memories (limited burst support)
- External termination and error acknowledge with bus monitor timeout
- Automated internal termination with configurable wait states to interface with slower memories or peripherals
- Up to four chip selects (giving a total addressable space of up to 64 Mbytes)
- Four write enable signals, configurable as byte enables, to control memory laning
- Individual clock group with configurable bus speeds of system clock speed, system clock speed divided by two, and system clock speed divided by four, up to a maximum of 25 MHz
- Optional automatic clock gating to allow the CLKOUT signal to be turned off when EBI accesses are not being performed. This helps with EMI reduction as well as power saving
- Module disable mode to help with overall power saving

2.1 MPC5510 EBI Restrictions

Some limitations in the MPC5510 EBI implementation are detailed below:

- The EBI does operate in non multiplexed mode, and the EBI control registers can be configured for this mode. However, the MPC5510 only supports multiplexed address/data mode. The one exception to this is when running the EBI in 16-bit data port mode, the upper eight address lines are available non multiplexed. See Section 3.3, “Multiplexed Address/Data bus”, for details.
- There is no external boot mode so an external flash cannot be used to boot the 5510 from reset.
- The maximum operating speed of the EBI is 25 MHz.
- The EBI supports burst mode on the MPC5510. However, only the DMA can instigate an EBI burst cycle. Accesses from the core cannot be bursted because there is no cache. This application note focuses on non-burst access mode.

2.2 Conceptual Overview

Before delving into the mechanics of the EBI, it is important to have a fundamental understanding of what an EBI provides.

Essentially, the EBI offers a user-definable external memory space. Multiple chip selects allow multiple external memory areas to be configured independently for different peripherals. When an access is performed to a pre-defined address space, by reading to or writing from an address within that memory range, the EBI automatically asserts the relevant chip selects for that range. By connecting a peripheral such as SRAM to the appropriate EBI signals, this device is then mapped to the defined external memory space.

On the MPC5510, there are up to four independent chip selects that allow up to four memory-mapped peripherals to be connected to the EBI as shown in Figure 1.

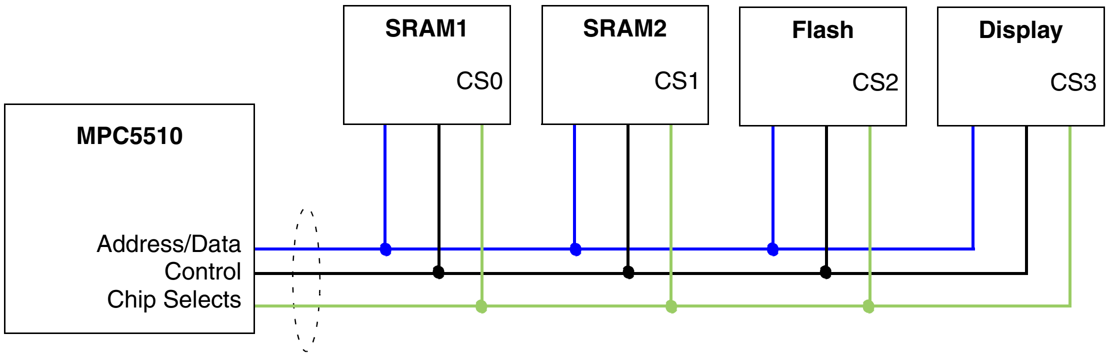


Figure 1. External Bus Interface

NOTE

As mentioned above, the MPC5510 does not provide the facility to boot from external memory. This means that instruction fetches over the EBI are not an intended mode of operation. However, external flash memory can be used for data storage.

3 Signal Description

Before looking at the configuration and setup of the EBI, you need to understand the physical EBI signals and their purpose. See Section 4, "Configuring the EBI," for actual configuration information that relates the signals to register configuration.

Table 1. MPC5510 EBI Signal Descriptions

EBI Signal	Port Position	Description	Comments
AD[0:31]	AD[0:7] = PortJ[0:7] AD[8:15] = PortF[2:9] AD[16:31] = PortG[0:15]	Multiplexed Address and Databus	AD[0] is the MSB, AD[31] is the LSB. At the start of the EBI access, the multiplexed bus initially contains address information then it switches to data.
CLKOUT	PortE[6]	MCU Clockout Signal	Derived from the system clock with default /2 ratio
$\overline{\text{CS[0]}}$	PortF[11]	EBI Chip Select 0	Used to control multiple external peripherals. The EBI triggers the relevant chip select when an external access is performed to the relevant address. Not all chip selects are available on all packages.
$\overline{\text{CS[1]}}$	PortF[10]	EBI Chip Select 1	
$\overline{\text{CS[2]}}$[1]	PortH[3]	EBI Chip Select 2	
$\overline{\text{CS[3]}}$[1]	PortH[2]	EBI Chip Select 3	
$\overline{\text{OE}}$	PortF[13]	Output Enable	Indicates that a peripheral may drive data onto bus
R/$\overline{\text{W}}$	PortF[0]	Read/Write	Indicates current cycle is read (1) or write (0)
$\overline{\text{TA}}$	PortF[1]	Transfer Acknowledge	Indicates that a peripheral has completed transfer

Table 1. MPC5510 EBI Signal Descriptions (continued)

EBI Signal	Port Position	Description	Comments
$\overline{\text{TS}}$	PortF[12]	Transfer Start	Indicates that the Address on AD[8:31] is valid
$\overline{\text{ALE}}$[1]		Address Latch Enable	Used to control an external address latch
$\overline{\text{WE}[0]}$	PortF[14]	Write/Byte enable 0	Enables external memory byte block 0
$\overline{\text{BDIP}}$[1]		Burst Data In Progress	Indicates that current transfer is a burst
$\overline{\text{WE}[1]}$	PortF[15]	Write/Byte enable 1	Enables external memory byte block 1
$\overline{\text{TEA}}$[1]		Transmit Error Acknowledge	Allows peripheral to terminate current transfer
$\overline{\text{WE}[2]}$	PortH[14]	Write/Byte enable 2	Enables external memory byte block 2
$\overline{\text{WE}[3]}$	PortH[15]	Write/Byte enable 3	Enables external memory byte block 3

[1] EBI functions are based on the port first alternate function unless the function name is denoted in which case the second or third alternate function is used. These pins must be manually configured to the appropriate function from the relevant SIU_PCR (Pad Configuration Register).

All the signals above are available in the 208BGA package. If you are using a 144QFP package the functionality is limited as follows:

- The address/data bus is limited to AD[8:31] giving 16-bit data and 24-bit address.
- $\overline{\text{CS}[2]}$ and $\overline{\text{CS}[3]}$ are not available
- $\overline{\text{WE}[2]}$ and $\overline{\text{WE}[3]}$ are not available.

If you are using a 176QFP package, the top four address/data lines are not present, which means you are limited to 16-bit data mode.

3.1 CLKOUT

The EBI CLKOUT signal provides a useful timing reference for the EBI control signals and indicates the operation speed of the EBI and system. This is also often used as a system alive signal in system debug situations. By default, the CLKOUT signal is disabled for EMI considerations.

The EBI can be clocked at the system clock frequency /1, /2 (default), or /4 with a maximum EBI speed of 25 MHz.

3.2 Chip Selects

There are up to four active low chip selects on the MPC5510, $\overline{\text{CS}[0:3]}$. These allow up to four peripherals to be connected to the same external bus with independent control. When a chip select is mapped to an address range, the EBI automatically asserts that chip select when an access is made within the specified address range.

Chip selects areas should not be overlapped as this results in a data conflict (for data reads where multiple external peripherals drive the bus) or chip select attribute conflicts based on the configuration of the chip selects.

3.3 Multiplexed Address/Data bus

The 5510 EBI provides up to 24 address lines and up to 32 data lines giving an addressable data space of 16 Mbytes per chip select. The address and data signals are multiplexed together to form the address/data bus AD[0:31]. At the start of the EBI access cycle, address information is driven onto the bus for one clock cycle. During this phase the transfer start ($\overline{\text{TS}}$) signal is driven low. When the address is valid, ALE (Address Latch Enable) is asserted and is used to control an external address latch which captures the address and makes it available all through the bus cycle. After $\overline{\text{TS}}$ is negated, the multiplexed bus then enters the data phase for the remainder of the bus cycle. See Figure 1 for details.

The EBI can be configured to use a 16-bit or 32-bit wide data port. Due to pin availability, the full 32-bit wide port is only available on the 208BGA package. If a 16-bit data port is selected, the data can be mapped to the upper or lower half of the 32-bit address/data bus (on AD[0:15] or AD[16:31]).

If the data-port width is set to 16-bits and the data is mapped to the lower half of the address/data bus, then non multiplexed address lines are available for the upper eight address lines, Address[23:16]. Using these non-multiplexed address lines saves an additional address latch. This feature is not available in 32-bit data port mode.

Irrespective of the data port size, 8, 16, and 32-bit accesses can be performed over the EBI. In the case of a 32-bit access to a 16-bit port, the EBI automatically performs two back-to-back 16-bit transfers. Misaligned accesses are supported, although these take more cycles to complete because they require multiple EBI accesses.

CAUTION

Power Architecture naming convention is used on the EBI signals. The Most Significant Bit (MSB) is Address/Data [0] and the Least Significant Bit (LSB) is Address/Data [31]. This can cause some confusion when interfacing the EBI to external peripherals with conventional MSB/LSB.

The following plots show the EBI performing a data write access in non burst mode. Important things to note are:

- $\overline{\text{CS0}}$ (chip select 0), $\overline{\text{TS}}$ (Transfer Start), and $\overline{\text{ALE}}$ (Address Latch Enable) are active low
- The chip select is always asserted at the start of the cycle on the rising edge of CLKOUT.
- When $\overline{\text{TS}}$ is low, the multiplexed address/data bus is in the address bus phase so in this case, the sampled address/data line has an address value of 0. $\overline{\text{TS}}$ changes state on the rising CLKOUT edge
- Looking at the second plot where MCU PF12 has been re-configured from $\overline{\text{TS}}$ to $\overline{\text{ALE}}$, the address is latched using an external latch on the falling edge of $\overline{\text{ALE}}$ (see Section 3.8, "Address Latch Enable")
- After the address phase, the Addr/Data line changes to data. In this case a data value of 1 is then driven to the observed addr/data pin.

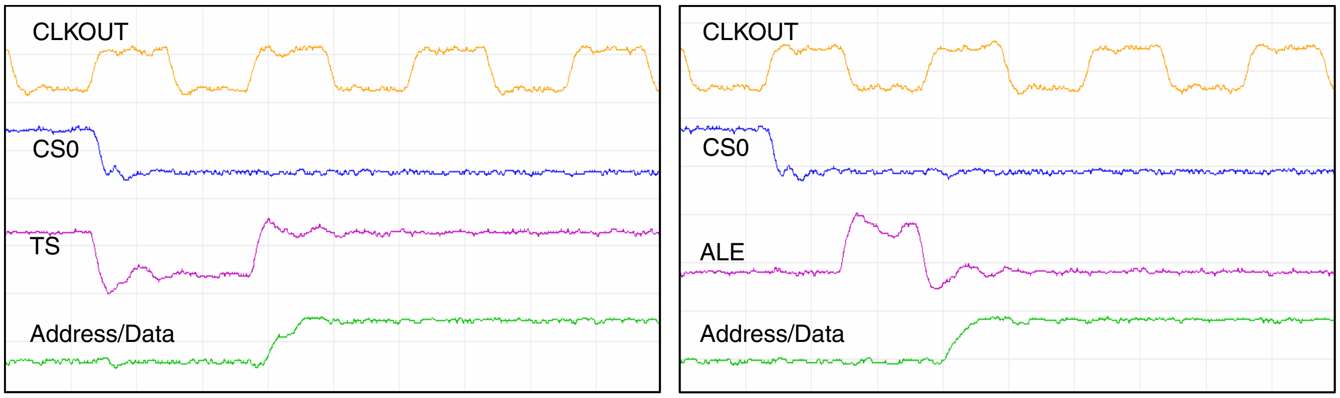


Figure 2. EBI Data Write (Clkout, CS, TS/ALE, and A/D)

3.4 Output Enable

The active low Output Enable signal, $\overline{\text{OE}}$, is driven by the EBI during a read cycle to allow the external peripheral to drive data onto the AD bus. Any peripheral connected to the EBI capable of driving data onto the bus must have a corresponding output enable or bus enable signal input.

3.5 Read/Write

The Read/Write signal, R/$\overline{\text{W}}$, is driven by the EBI to determine if the current EBI access is a read or a write. If R/$\overline{\text{W}}$ is high, the current transfer is a read. If R/$\overline{\text{W}}$ is low the current transfer is a write. The R/$\overline{\text{W}}$ is set and valid for the complete bus cycle.

3.6 Transfer Acknowledge

The active low Transfer Acknowledge signal, $\overline{\text{TA}}$, is used to terminate the current EBI cycle after the peripheral has completed it's current data read or write. This signal can be driven from an external peripheral or from the EBI itself if auto $\overline{\text{TA}}$ generation is enabled.

Most standard SRAMs are asynchronous with no $\overline{\text{TA}}$ generation capability so require the EBI to be configured for automatic $\overline{\text{TA}}$ generation.

If the $\overline{\text{TA}}$ signal is configured to be driven externally, the EBI module waits indefinitely for $\overline{\text{TA}}$ assertion. There are, however, four ways to terminate the cycle:

- Normal $\overline{\text{TA}}$ assertion
- Assertion of $\overline{\text{TEA}}$ (Transfer Error Acknowledge) signal by external hardware
- Assertion of MCU Reset
- Expiration of the EBI bus monitor. This allows a $\overline{\text{TA}}$ timeout period to be set. If the external peripheral has not asserted $\overline{\text{TA}}$ within that period, the cycle is terminated. This is discussed in detail in Section 4.2.2, "Bus Monitor Configuration".

3.7 Transfer Start

The Transfer Start signal, $\overline{\text{TS}}$, is asserted by the EBI to indicate the start of the current transfer. The address information on the multiplexed address/data bus is valid when $\overline{\text{TS}}$ is asserted and CLKOUT goes high.

3.8 Address Latch Enable

The active low Address Latch Enable signal, $\overline{\text{ALE}}$, is asserted by the EBI when the address information on the multiplexed address/data bus is valid. $\overline{\text{ALE}}$ is maintained throughout the cycle and is de-asserted at the start of the next EBI cycle before the new address information is valid.

This signal can be connected directly to the latch input on a suitable fast latch to demultiplex the address information from the address/data bus and make it available throughout the complete bus cycle.

NOTE

On RevA devices, the $\overline{\text{ALE}}$ signal is asserted a bit late and the address/data bus has already started to change from address to data by the time $\overline{\text{ALE}}$ gets to a typical latch VIL of 0.8 V. The workaround for this is to configure the address/data pads as medium drive so that when then address information is latched, the bus values reflect the address state.

The following plots shows the EBI performing a data read in non burst mode. Important observations:

- $\overline{\text{CS0}}$ (chip select 0), $\overline{\text{OE}}$ (output enable), and $\overline{\text{TA}}$ (transfer acknowledge) are all active low
- $\overline{\text{OE}}$ is asserted on the second CLKOUT period after chip select goes low. Recalling the plots showing $\overline{\text{TS}}$, this is the point where the address/data bus has been assigned to the data phase
- After $\overline{\text{OE}}$ goes low, the peripheral is then free to drive data on the bus. In this example, the monitored address/data line is driven high.
- The time taken for the data to become valid on the bus depends on the peripheral. Looking at the $\overline{\text{TA}}$ plot in the second diagram, when $\overline{\text{TA}}$ is asserted, the data is latched into the MCU on the next rising CLKOUT edge

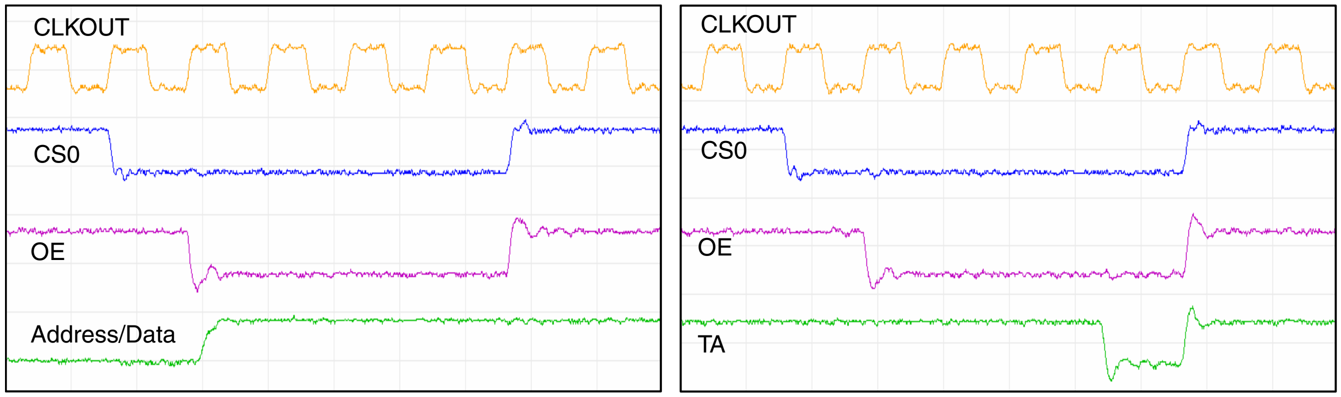


Figure 3. EBI Data Read (Clkout, CS, OE, A/D, and $\overline{\text{TA}}$)

3.9 Write/Byte Enables

The EBI has four active low Write Enable signals, $\overline{\text{WE}[0:3]}$, that can also be configured as byte enables (active for reads and writes). The write/byte enables are used to control byte wide access to external memory, sometimes called the laning. Figure 4 shows how a 32-bit data word 0x0102_0304 is addressed in 32-bit and 16-bit data port modes.

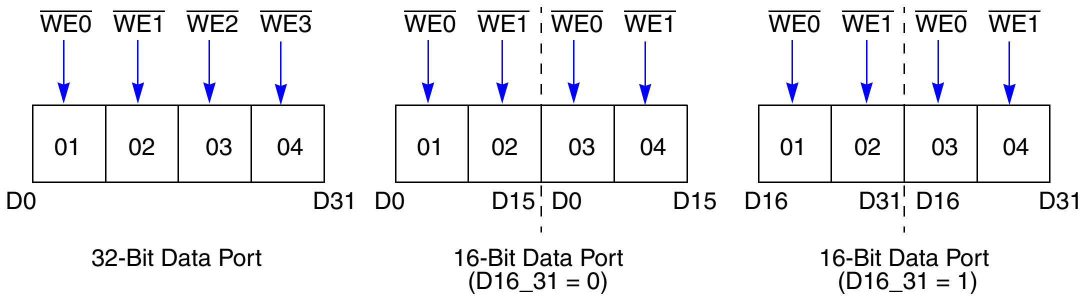


Figure 4. EBI Write/Byte Enables

As previously mentioned, the EBI can be configured for a 16-bit or 32-bit data port. In each of these configurations, an external memory connected to the EBI must match the data port configuration so would be configured as a 16-bit or 32-bit memory system. This introduces a problem when trying to write a single byte to the memory (or writing 16-bits to a 32-bit memory) as the address lines that provide that resolution are not connected to the memory. The write/byte enables are used for this purpose.

Table 2 shows all possible access types over the EBI and details which of the write/byte enable signals are valid for each access.

Most SRAM devices have individual enables for byte blocks to provide the necessary laning. With this configuration, the WEx signals must be configured as byte enables to enable the SRAM for reads and writes. With this in mind, all scope plots show the signals as $\overline{\text{BEx}}$ (Byte Enable) although the pins are normally referenced as write enables in the reference manual.

Table 2. MPC5510 EBI Write/Byte Enables

EBI Port Size	Transfer Type	Active Data Pins	WE/BE[0]	WE/BE[1]	WE/BE[2]	WE/BE[3]
32-bit	32-bit	D[0:31]	0	0	0	0
	16-bit (upper)	D[0:15]	0	0	1	1
	16-bit (lower)	D[16:31]	1	1	0	0
	8-bit (01[1])	D[0:7]	0	1	1	1
	8-bit (02[1])	D[8:15]	1	0	1	1
	8-bit (03[1])	D[16:23]	1	1	0	1
	8-bit (04[1])	D[24:31]	1	1	1	0
16-bit (D16_31=0)	16-bit	D[0:15]	0	0	X	X
	8-bit (01[1])	D[0:7]	0	1	X	X
	8-bit (02[1])	D[8:15]	1	0	X	X
16-bit (D16_31=1)	16-bit	D[16:31]	0	0	X	X
	8-bit (01[1])	D[16:23]	0	1	X	X
	8-bit (02[1])	D[24:31]	1	0	X	X

[1] The 01, 02, 03, and 04 refer to the bytes shown in Figure 4.

The plots in Figure 5 show a normal 16-bit aligned access (to an EBI configured with a 16-bit data port) and a misaligned 16-bit access. The write/byte enables are configured as byte enables. Observations:

- During the 16-bit aligned access, the Byte enables both triggers as expected
- The Byte enable signals are only asserted for the data phase of the EBI cycle.
- During the 16-bit non-aligned access (doing 16-bit access to an odd address), this is performed as two back-to-back 8-bit accesses by the EBI. The chip select is not de-asserted between the accesses (shown as shaded area on plot). Because two 8-bit accesses are performed, only one $\overline{\text{BE}}$ signal triggers in each access. $\overline{\text{TS}}$ and $\overline{\text{TA}}$ have been superimposed onto the second plot so it's obvious where the 8-bit EBI accesses start and stop.

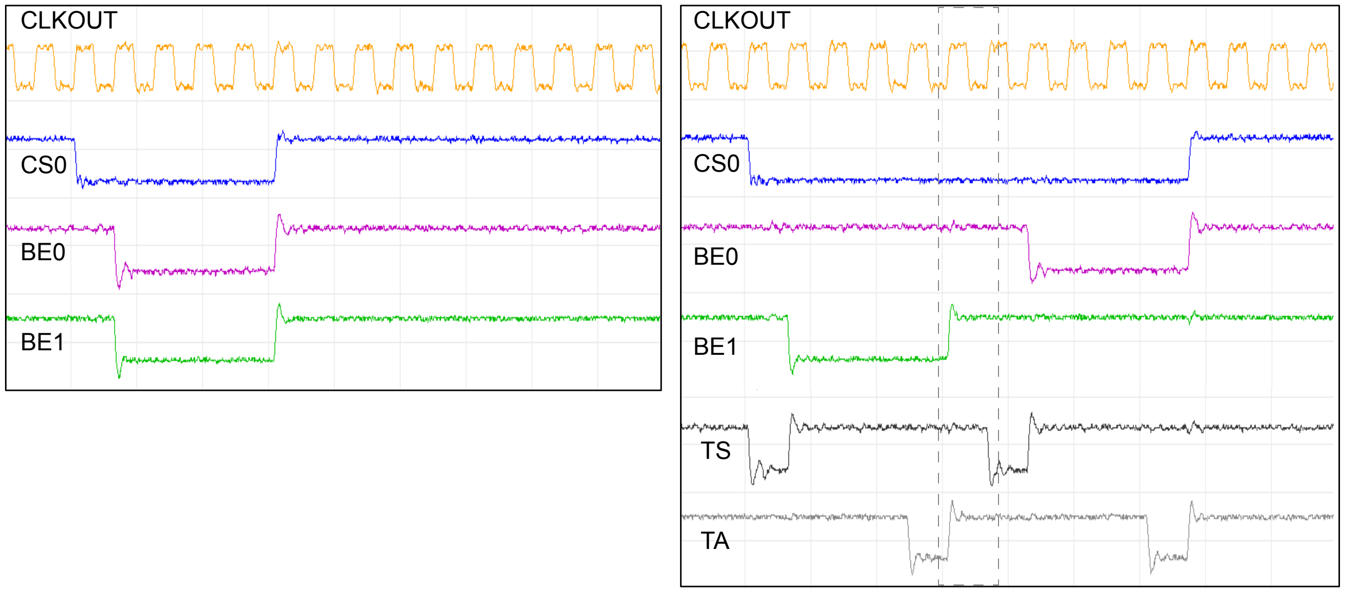


Figure 5. EBI Aligned and Misaligned Access (Clkout, CS, BEx, TS, and $\overline{\text{TA}}$)

3.9.1 Burst Data In Progress

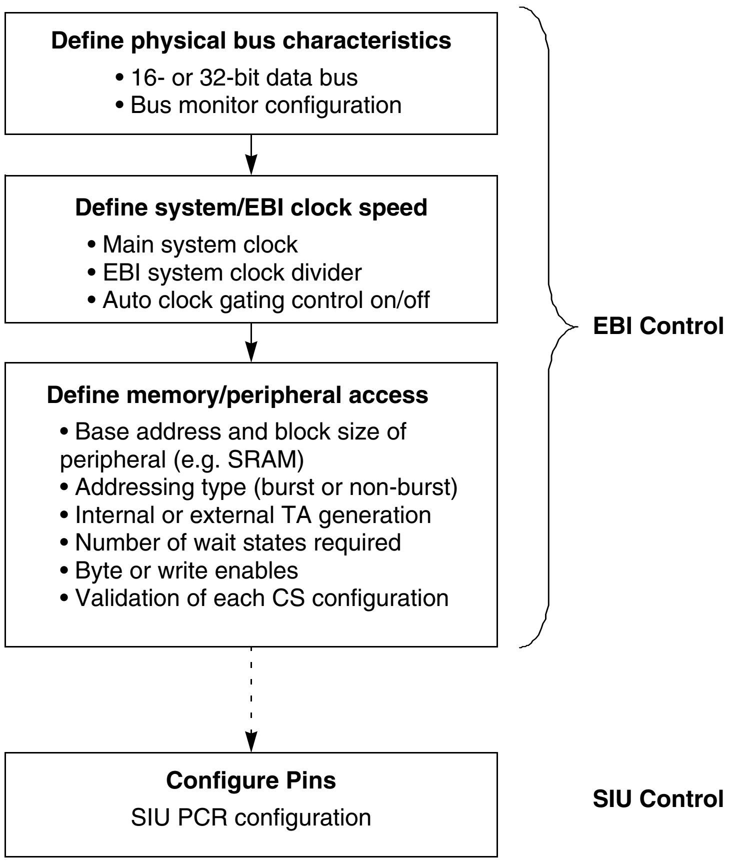
The active low Burst Data In Progress signal, $\overline{\text{BDIP}}$, is asserted by the EBI during a burst access to indicate that there is another data beat in the burst.

3.9.2 Transfer Error Acknowledge

If supported by an external device, $\overline{\text{TEA}}$ can be asserted to indicate that there was an error with the current transfer (e.g. the EBI has attempted to access a protected area of an external peripheral).

4 Configuring the EBI

This section guides you through the actual configuration options for the EBI. There are four main areas to consider when looking at the EBI configuration as described by Figure 6.



Define physical bus characteristics
• 16- or 32-bit data bus
• Bus monitor configuration
Define system/EBI clock speed
• Main system clock
• EBI system clock divider
• Auto clock gating control on/off
Define memory/peripheral access
• Base address and block size of peripheral (e.g. SRAM)
• Addressing type (burst or non-burst)
• Internal or external TA generation
• Number of wait states required
• Byte or write enables
• Validation of each CS configuration
EBI Control
Configure Pins
SIU PCR configuration
SIU Control

Figure 6. EBI Configuration

4.1 EBI Registers

There are 11 registers in the EBI. These are referred to in the following sections so are detailed here for reference. Please consult the MPC5510 reference manual for information on register address maps and other details.

There are three generic control/status registers

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ACGE	EXTM	EARB	0	1	0	0	0	0	MDIS	0	0	0	D16_31	AD_MUX	DBM
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0

Figure 7. Module configuration register (EBI_MCR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TEAF	BMTF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8. Transfer Error Status Register (EBI_TESR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BMT								BME	0	0	0	0	0	0	0
W																
Reset	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0

Figure 9. Bus Monitor Control Register (EBI_BMCR)

There are two registers per chip select (eight registers in total).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BA			BA												
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BA	0	0	0	PS	0	0	0	AD_MUX	BL	WEBS	TBDIP	0	SETA	BI	V
W																
Reset	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0

Figure 10. Base Register (EBI_BRx)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	AM												
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AM								SCY				0	BSCY		0
W																
Reset	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0

Figure 11. Option Register (EBI_ORx)

4.2 Define Physical Bus Characteristics

When looking at an EBI design, it is important to consider how the external bus is configured in terms of physical address/data configuration as well as determining the base address and size of the external peripheral(s).

4.2.1 Data Bus Configuration

Affected register bits: EBI_MCR (DBM, D16_31, ADMUX),   EBI_BRx (PS, AD_MUX)

The first thing you need to consider is the width of the databus. This is a fundamental decision because the hardware configuration for a 16-bit and 32-bit bus are different and not interchangeable. You can, however, perform a 16-bit data access to a 32-bit bus and vice-versa. If multiple accesses are required (32-bit access to a 16-bit bus), this is handled automatically by the EBI.

The bus width is configured by writing the DBM bit within the EBI_MCR (Module Configuration Register. Due to pin availability, the 32-bit data port is only available on the 208BGA package. The data port size also needs to be configured in the Chip select Base register (EBI_BRx) via the PS field.

If the data port width is set to 16-bits, there is an option to locate the data on the lower (AD[16:31]) or upper (AD[0:15]) half of the bus. The normal place to locate this is on the lower address space so that address pins on the upper half of the bus can be configured as non-multiplexed address lines or used as GPIO. The location of the data-bus is configured by the D16_31 bit in the EBI_MCR. Non-Multiplexed address lines are available as the second alternate SIU function where available.

As mentioned previously, the EBI can be configured for non multiplexed mode but the only configuration acceptable for the MPC5510 is multiplexed mode. This is configured via the AD_MUX bit which must be configured in the EBI_MCR and the EBI_BRx.

4.2.2 Bus Monitor Configuration

Affected register bits: EBI_BMCR (BME, BMT),   EBI_TESR (BMTF)

If the bus monitor is enabled, the system recovers from the situation where an external $\overline{\text{TA}}$ has not been received within a user defined time period.

The bus monitor is enabled by writing the BME field in the EBI_BMCR (Bus Monitor Control Register). The timeout period is set as described in the reference manual by writing the BMT field in the EBI_BMCR. When a timeout occurs, the BMTF flag is set in the EBI_TESR (Transfer Error Status Register) register

NOTE

The bus monitor is only valid for chip select accesses where $\overline{\text{TA}}$ is driven by the external peripheral.

4.3 Define System/EBI Clock Speed

4.3.1 EBI Clock Control

Affected register bits: SIU_ECCR (EBDF)

The EBI is clocked from the main system clock via a clock divider with possible ratios of /1, /2 and /4. The maximum permitted speed of the EBI is 25 MHz. With the maximum MCU system speed of between 66 and 80 MHz, there is an obvious trade off between overall system performance and maximum EBI clock speed.

The EBI clock divider is configured by writing to the EBDF field in the SIU_ECCR (External Clock Control Register).

CAUTION

The default EBI clock divider is 2. If you are running with a system clock greater than 50 MHz, you need to change the divider ratio.

4.3.2 EBI Clock Gating

Affected register bits: EBI_MCR (ACGE)

The EBI CLKOUT signal (multiplexed on PE6) provides a timing reference for EBI signals. It also shows a representation of the system clock although caution should be exercised when looking at this clock to determine system frequency due to the EBI clock dividers.

There is an option to only drive CLKOUT during a chip select access. This provides a good method of minimizing EMI and power consumption while maintaining the CLKOUT signal for timing references during an EBI bus cycle. Most external memory does not require the use of CLKOUT. However, some external peripherals need this.

Some peripherals do not work correctly without a continuous clock.

The CLKOUT gating is controlled via the ACGE bit in the EBI_MCR. The following plots show the EBI operation with clock gating disabled and enabled.

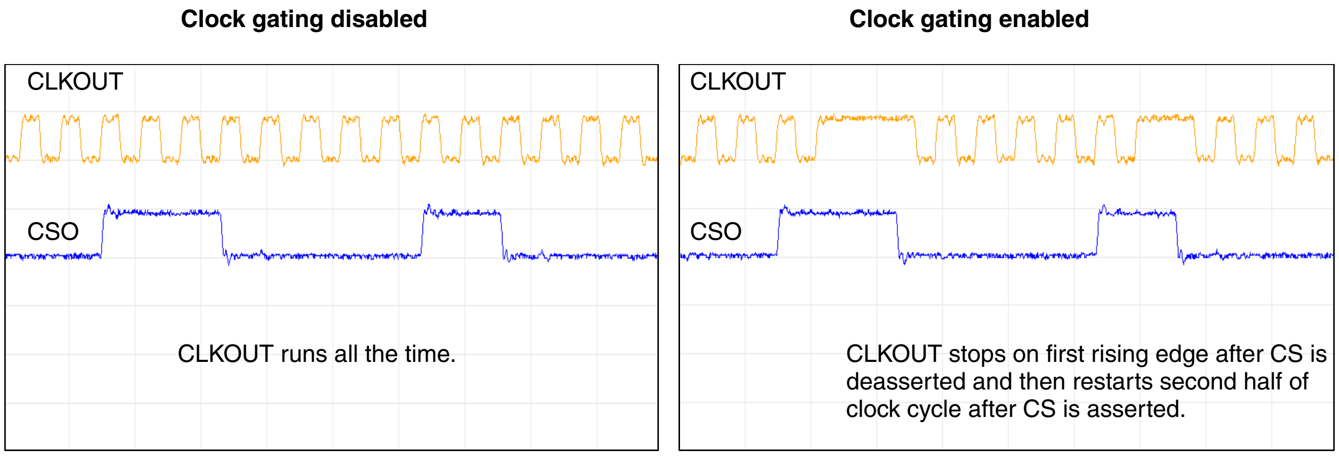


Figure 12. EBI Automatic Clock Gating (Clkout, CS)

4.4 Define Memory/Peripheral Access

4.4.1 Base Address and Block Size of Peripherals

Affected register bits: EBI_BRx (BA, AM)

The chip select base registers (EBI_BRx) and option registers (EBI_ORx) are configured to define the base or starting address attributed to each chip select as well as the physical size of the memory block defined. When an address is accessed within the chip select range, the EBI triggers the relevant chip select accordingly.

The BA field in the EBI_BRx register defines the chip select base address. The BA field is 17-bits wide and the upper 3 bits of BA are set to 001b. When configuring the base address you are defining the upper part of a 32-bit address so the resolution/alignment is 32 bytes. This gives an effective base address range between 0x2000_0000 and 0x3FFF_8000.

The AM field in the EBI_ORx register defines the size of the external address block. As with the base address, the address mask field is 17-bit long and the upper 3 bits of the address mask are set. Writing a 1 to the address mask masks that bit so it is not used in the comparison with the address. Clearing the bit validates it and that bit is used as a valid entry in the address mask comparison.

Example 1. Configuring an External Bus Peripheral

To configure an external bus peripheral to sit at base address 0x2010_0000 with block size of 64 bytes, then configure the BA and AM fields as follows:

BA = 0x2010_0000 (Write 0x20100 to BA field)

AM = 0xFFFF_0000 (Write 0xFFFF0 to AM field)

NOTE

If you have a physically smaller memory than the memory size defined in the address mask, the memory space wraps around so it may appear more memory exists than actually does. Ensure that the AM maps correctly to the size of an external memory to avoid this.

4.4.2 Addressing Type (Burst or Non-Burst)

Affected register bits: EBI_MCR (BI, TBDIP), EBI_ORx (BSCY)

As mentioned earlier, the EBI can perform burst accesses. However, only the DMA in the MPC5510 has the capability of instigating a burst. If you wish to use burst memories, consult the MPC5510 reference manual.

For non burst memory access, you need to ensure that the BI (Burst Inhibit) bit is set in the EBI_MCR. When this bit is set, the other burst control fields (BL, TBDIP in EBI_MCR and BSCY in EBI_ORx) have no effect on the operation of the EBI

4.4.3 Internal or External $\overline{\text{TA}}$ Generation (Including Wait States)

Affected register bits: EBI_BRx (SETA), EBI_ORx (SCY)

Every EBI cycle needs to be terminated by a $\overline{\text{TA}}$ (Transfer Acknowledge) signal. On some peripherals, especially those with non deterministic response times, they provide the $\overline{\text{TA}}$ signal which is fed into the EBI. Other peripherals such as SRAMs, respond in a very deterministic time frame and do not provide a $\overline{\text{TA}}$ signal. The EBI has the ability to generate a $\overline{\text{TA}}$ signal after a set number of clock cycles, known as automatic $\overline{\text{TA}}$ generation.

The SETA bit (Select External TA) in the EBI_BRx determines whether the $\overline{\text{TA}}$ signal is generated by the EBI (default state) or if an external peripheral provides the $\overline{\text{TA}}$ signal.

If SETA is cleared so the EBI generates the $\overline{\text{TA}}$, then the SCY field in the EBI_ORx determines the number of wait states or extra clock cycles in the EBI bus cycle before the cycle is terminated. Up to 15 wait states can be added. This allows accurate tailoring of your external peripheral with the EBI timing. The reference manual gives some useful information on calculating the number of wait states required for your application.

Figure 13 shows an EBI access with zero wait states compared to one with five wait states. The address phase in each case is maintained at one cycle and is the only data access time extended.

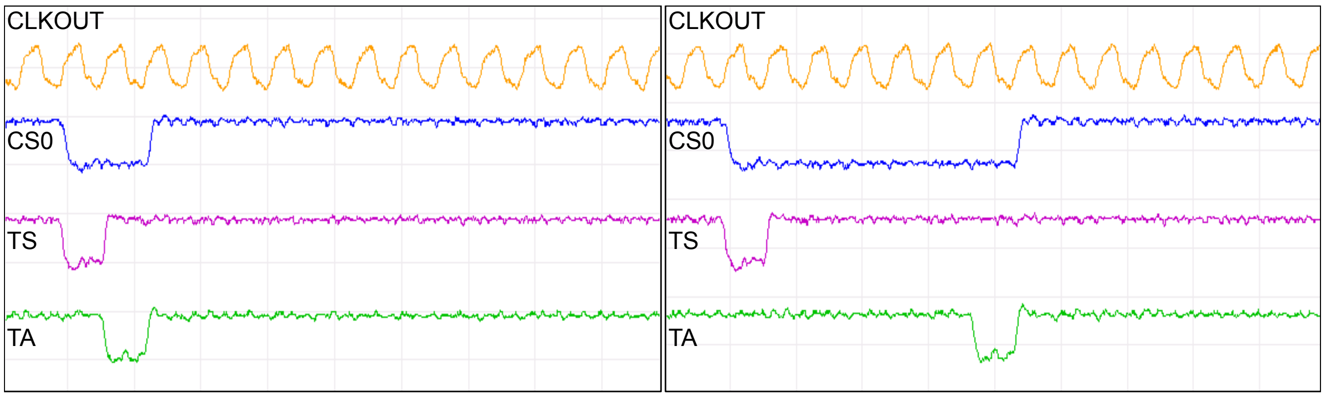


Figure 13. EBI Wait States (Clkout, CS, TS, and TA)

4.4.4 Write Enable/Byte Enable Configuration

Affected register bits: EBI_BRx (WEBS)

As previously mentioned, most external 16/32-bit memories (possibly with the exception of flash) have individual byte enables that need to be controlled irrespective of whether the access is a read or a write. In this instance, the laning signals need to be enabled for read and write. Therefore, the write enable/byte enable signals must be configured as byte enables.

The WEBS (Write enable byte select) bit within the EBI_BRx controls this functionality.

4.4.5 Validating the EBI Configuration

Affected register bits: EBI_MCR (MDIS), EBI_BRx (V)

Like a lot of the MPC5500/MPC5510 family peripherals, there is an MDIS control bit in the EBI_MCR. This bit allows the complete EBI to be disabled if required and is useful for power saving if the EBI is not used. For operation of the EBI, the MDIS bit must be set to its default state of 0.

Each chip select configuration needs to be validated before it is used by the EBI. This is done by writing the V (valid) bit in the EBI_BRx.

4.5 Other Register Bits

Affected register bits: EBI_MCR (EXTM, EARB), EBI_TESR (TEAF)

The EXTM and EARB bits in the EBI_MCR should not be changed from their default value of 0. The EBI cannot be used as an external master on the MPC5510 so EXTM (External Master Mode) must remain at 0. The EARB (External Arbitration) bit has no effect when EXTM is 0.

The TEAF (Transfer Error Acknowledge Flag) in the EBI_TESR indicates that an external system has terminated an EBI cycle by asserting the $\overline{\text{TEA}}$ line. This flag can be cleared by writing a 1 to the bit.

4.6 SIU Pad Configuration

Table 3 shows how each of the pins used on the EBI should be configured for 16 and 32-bit access.

Table 3. EBI SIU Pad Configuration

EBI Signal	MCU Ports	SIU.PCR Configuration		Comments
		16-bit data Port	32-bit Data Port	
AD[16:31]	PortG[0:15]	First alternate function, MEDIUM slew rate	First Alternate function MEDIUM slew rate	—
AD[0:15]	PortJ[0:7], PortF[2:9]	Address pins second alternate MEDIUM slew rate.	First Alternate function, MEDIUM slew rate	In 16-bit mode, pins not used for address can be used for GPIO.
CLKOUT	PortE[6]	First Alternate, Output Custom Slew Rate	First Alternate, Output Custom Slew Rate	Slew rate according to requirements but likely high/medium.
$\overline{\text{CS[0]}}$	PortF[11]	First Alternate, Output HIGH Slew Rate	First Alternate, Output HIGH Slew Rate	Only need to define one chip select per external peripheral. Unused CS can be configured as GPIO.
$\overline{\text{CS[1]}}$	PortF[10]	First Alternate, Output HIGH Slew Rate	First Alternate, Output HIGH Slew Rate	
$\overline{\text{CS[2]}}$	PortH[3]	Third Alternate, Output HIGH slew rate	Third Alternate, Output HIGH slew rate	
$\overline{\text{CS[3]}}$	PortH[2]	Third Alternate, Output HIGH slew rate	Third Alternate, Output HIGH slew rate	
$\overline{\text{OE}}$	PortF[13]	First Alternate, Output HIGH Slew Rate	First Alternate, Output HIGH Slew Rate	—
R/$\overline{\text{W}}$	PortF[0]	First Alternate, Output HIGH slew Rate	First Alternate, Output HIGH Slew Rate	—

Table 3. EBI SIU Pad Configuration (continued)

EBI Signal	MCU Ports	SIU.PCR Configuration		Comments
		16-bit data Port	32-bit Data Port	
$\overline{\text{ALE}}$	PortF[12]	Third Alternate, Output HIGH slew rate	Third Alternate, Output HIGH slew rate	—
$\overline{\text{WE[0]}}$	PortF[14]	First Alternate, Output HIGH slew Rate	First Alternate, Output HIGH slew Rate	Multiplexed with $\overline{\text{BDIP}}$
$\overline{\text{WE[1]}}$	PortF[15]	First Alternate, Output HIGH slew Rate	First Alternate, Output HIGH slew Rate	Multiplexed with $\overline{\text{TEA}}$
$\overline{\text{WE[2]}}$	PortH[14]	Not Used. GPIO	First Alternate, Output HIGH slew Rate	—
$\overline{\text{WE[3]}}$	PortH[15]	Not Used. GPIO	First Alternate, Output HIGH slew Rate	—
$\overline{\text{TA}}$	PortF[1]	First Alternate, Output HIGH slew Rate	First Alternate, Output HIGH slew Rate	Assumes Auto TA generation is set.

NOTE

When the EBI is enabled and the pads in the SIU.PCR are configured for an EBI function, the pad input/output state is automatically controlled by the EBI for bidirectional (data) pins. This means that you do not need to set the OBE or IBE bits in the PCR for the data pins.

The general rule for configuration is to use medium slew rate control for all address/data lines and maximum slew rate control on all of the control signals. CLKOUT slew rate is a matter of preference and depends on what you are wanting to use it for. If it is being used as a timing reference, slew rate probably needs to be set to maximum.

5 Interface Examples

This section shows how to connect a 16-bit and 32-bit external synchronous SRAM (non burst) to the EBI. This is the same hardware configuration used on the MPC5510EVB with the exception that a GAL is used on the EVB to generate the ALE signal not present on Rev0 devices. The GAL also latches 2 address lines to save multiple latches on the EVB.

5.1 16-Bit Memory

In this example, a 16-bit asynchronous memory is connected to the EBI. The 16-bit SRAMs are readily available from various companies in various sizes. The EBI must be configured to use a 16-bit port and it is strongly recommended to select the lower address lines for the data multiplexing.

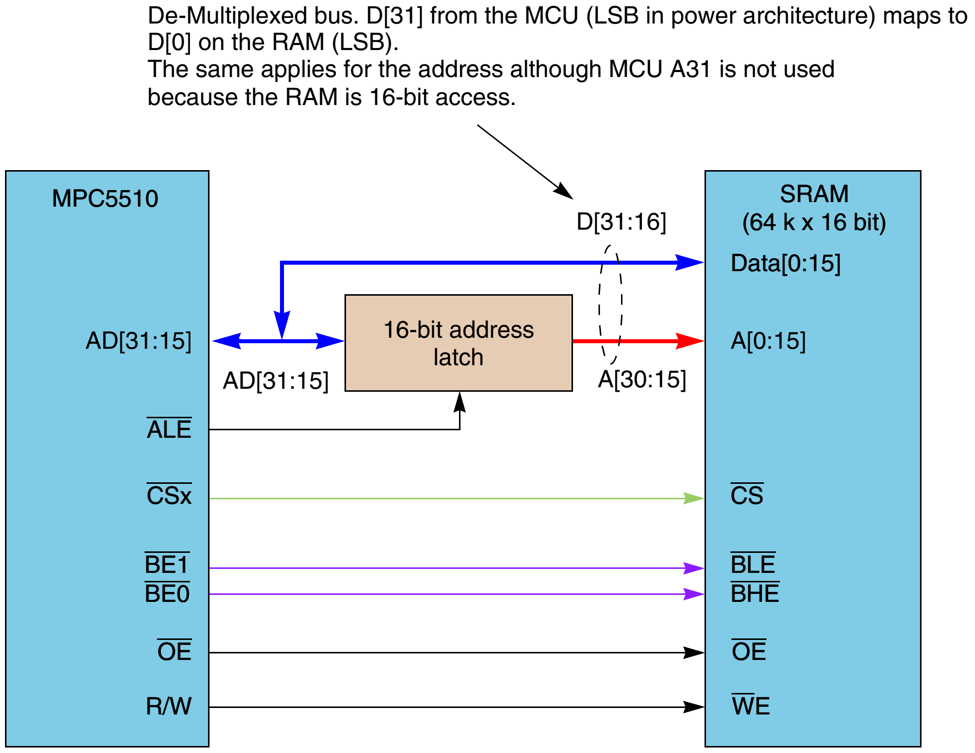

De-Multiplexed bus. D[31] from the MCU (LSB in power architecture) maps to D[0] on the RAM (LSB).
The same applies for the address although MCU A31 is not used because the RAM is 16-bit access.
MPC5510
AD[31:15]
ALE
CSx
BE1
BE0
OE
R/W
AD[31:15]
16-bit address latch
D[31:16]
A[30:15]
SRAM (64 k x 16 bit)
Data[0:15]
A[0:15]
CS
BLE
BHE
OE
WE

Figure 14. 16-Bit Asynchronous Memory Connected to EBI

NOTE

The MCU is addressed in Power Architecture terminology with ADDR[0] being the MSB (Most Significant Bit) whereas the memory is configured so that ADDR[0] is the LSB (Least Significant Bit). This is inherently confusing when connecting the MCU and memory together.

Because the RAM is being addressed as a 16-bit block, the least significant address from the MCU, Addr[31] is not required so MCU Addr[30] is connected to SRAM Addr[0].

Recall that in a 16-bit memory system, address lines A[15:8] are available in non multiplexed form (Si RevA and above only) on the same pin, SIU.PCR second alternate function.

5.1.1 16-Bit Initialisation Code

The following example code shows how to initialize a typical 16-bit EBI configuration with non-burst memory and 128 KB data port size. This uses the standard Freescale MPC5510 header file found at www.freescale.com.

Example 2. 16-Bit Initialisation Code

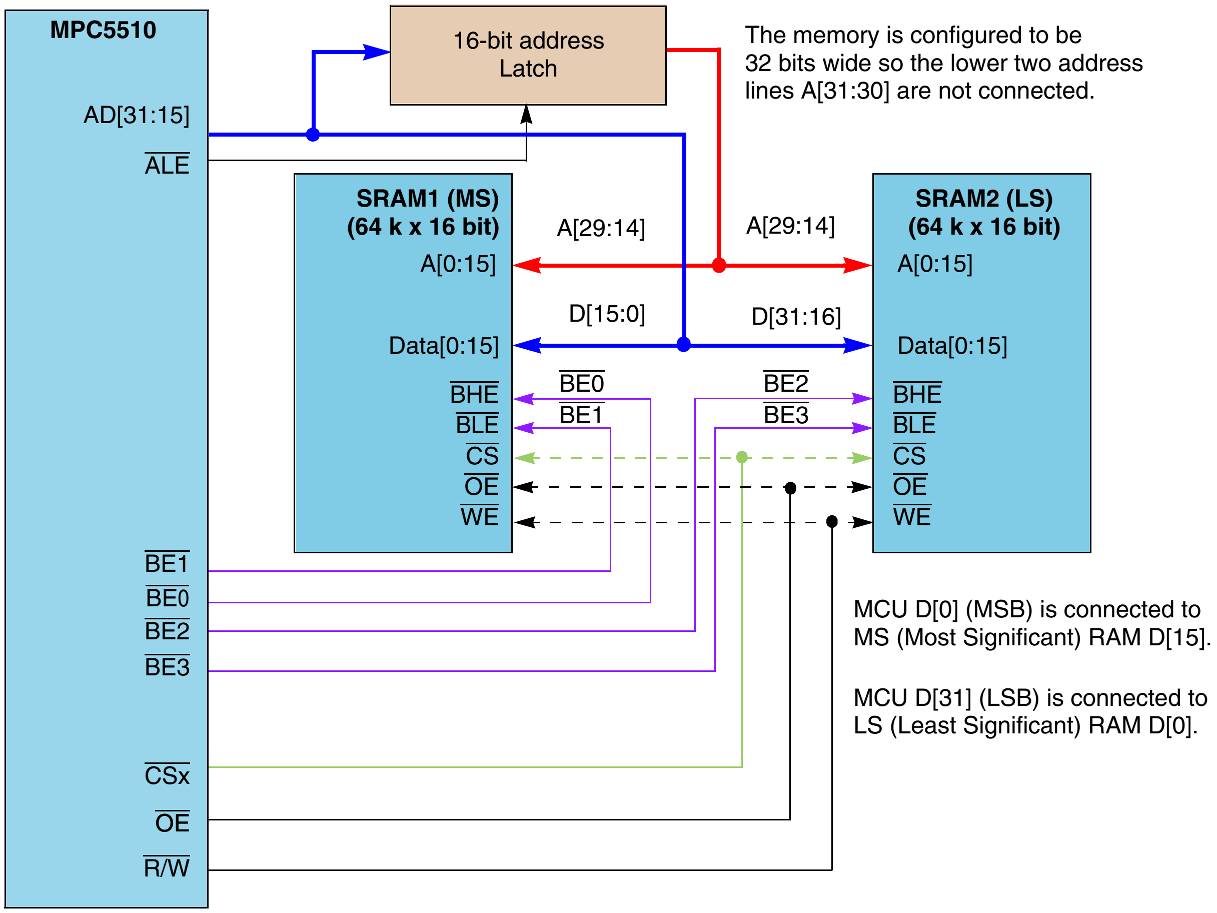
```
/* Set EBI clock divider to sysclck /4  */
   SIU.ECCR.B.EBDF = 0x3;

/* EBI General Config. Data on AD[16..31], Multiplexed A/D, 16-bit Data port   */
    EBI.MCR.R = 0x00000807;

/* CS0 Configuration. -  Base Address 0x2000_0000, 16-bit port, Mux AD, Byte Enable, Auto TA, no burst   */
   EBI.CS[CHIPSEL].BR.R  = 0x200008F3;
   EBI.CS[CHIPSEL].OR.R  = 0xFFFE0030;   /* 128K block size, 3 wait states */

/* Now setup SIU for each pin used in EBI. */

  /* CLKOUT = PE6 */
    SIU.PCR[70].R = 0x060C;

    /* Configure PortF[0..15] = PCR[80..95]  */
    SIU.PCR[80].R = 0x060F;                    /* PF0 = R/W.  Fast Slew, Output */
    SIU.PCR[81].R = 0x060F;                    /* PF1 = TA.   Fast Slew, Output */

    for (count=82; count<90; count++)          /* PF[2..9] = Address[8..15] (non multiplexed)  */
      {                                        /*  Medium Slew, Input / Output  */
          SIU.PCR[count].R = 0x0807;           /*  Set as 2nd alternate function */
      };

    SIU.PCR[90].R = 0x060F;                    /* PF10 = CS1. Fast Slew, Output */
    SIU.PCR[91].R = 0x060F;                    /* PF11 = CS0. Fast Slew, Output */
    SIU.PCR[92].R = 0x0C0F;                    /* PF12 = ALE. Fast Slew, Output */
    SIU.PCR[93].R = 0x060F;                    /* PF13 = OEx. Fast Slew, Output */
    SIU.PCR[94].R = 0x060F;                    /* PF14 = WE0. Fast Slew, Output */
    SIU.PCR[95].R = 0x060F;                    /* PF15 = WE1. Fast Slew, Output */

    /* Configure PortG[0..15] = PCR[96..111]. Address/Data [16..31]          */
    /* Med slew rate on these pads!   */
    for (count=96; count < 112; count++)
      {
        SIU.PCR[count].R = 0x0407;
      };
```

5.2 32-Bit Memory

In this example, a 32-bit memory subsystem is constructed from 2x 16-bit asynchronous memories. For this configuration, the EBI must be configured to use a 32-bit data port.

The lower two address lines are not used because the memory is 4 bytes wide. Resolution for individual bytes is provided by the byte/write enable signals.

A single 16-bit address latch is sufficient for memory sizes up to 256 KBs (2 off 64 k x 16-bit SRAMS). An additional latch is required if more than 16 address lines are required.



MPC5510
AD[31:15]
ALE
16-bit address Latch
The memory is configured to be 32 bits wide so the lower two address lines A[31:30] are not connected.
SRAM1 (MS) (64 k x 16 bit)
A[0:15]
Data[0:15]
BHE
BLE
CS
OE
WE
A[29:14]
A[29:14]
D[15:0]
D[31:16]
BE0
BE1
BE2
BE3
SRAM2 (LS) (64 k x 16 bit)
A[0:15]
Data[0:15]
BHE
BLE
CS
OE
WE
BE1
BE0
BE2
BE3
CSx
OE
R/W
MCU D[0] (MSB) is connected to MS (Most Significant) RAM D[15].
MCU D[31] (LSB) is connected to LS (Least Significant) RAM D[0].

Figure 15. 32-Bit Asynchronous Memory Connected to EBI

5.2.1 32-Bit Initialisation Code

The following example code shows how to initialize a typical 32-bit EBI configuration with non-burst memory and 256 KB data port size (2x 128K SRAMS). This uses the standard Freescale MPC5510 header file found at www.freescale.com.

Example 3. 32-Bit Memory

```
/* Set EBI clock divider to sysclck /4  */
    SIU.ECCR.B.EBDF = 0x3;

 /* EBI General Config.  Multiplexed A/D, 32-bit Data port    */
    EBI.MCR.R            = 0x00000806;

 /* CS0 Configuration. Base Address 0x2000_0000, 32-bit port, Mux AD, Byte Enable, Auto TA, no burst */
     EBI.CS[CHIPSEL].BR.R  = 0x200000F3;
     EBI.CS[CHIPSEL].OR.R  = 0xFFFC0030;   /* 256K block size, 3 wait states */

     SIU.PCR[70].R = 0x060C;   /* CLKOUT = PE6 */

     /* Configure PortF[0..15] = PCR[80..95]  */
      SIU.PCR[80].R = 0x060F;                    /* PF0 = R/W.  Fast Slew, Output */
      SIU.PCR[81].R = 0x060F;                    /* PF1 = TA.   Fast Slew, Output */
      for (count=82; count<90; count++)          /* PF[2..9] = A/D [8..15].       */
        {
           SIU.PCR[count].R = 0x0407;            /*   Medium Slew, Input / Output*/
        };
       SIU.PCR[90].R = 0x060F;                   /* PF10 = CS1. Fast Slew, Output */
      SIU.PCR[91].R = 0x060F;                    /* PF11 = CS0. Fast Slew, Output */
      SIU.PCR[92].R = 0x0C0F;                    /* PF12 = ALE. Fast Slew, Output */
      SIU.PCR[93].R = 0x060F;                    /* PF13 = OEx. Fast Slew, Output */
      SIU.PCR[94].R = 0x060F;                    /* PF14 = WE0. Fast Slew, Output */
      SIU.PCR[95].R = 0x060F;                    /* PF15 = WE1. Fast Slew, Output */


     /* Configure PortG[0..15] = PCR[96..111]. Address/Data [16..31] - Med slew rate on these pads!  */
      for (count=96; count < 112; count++)
        {
         SIU.PCR[count].R = 0x0407;
        };

     /* Configure PortJ[0..7] = PCR[128..135]. Data[0..7]   - Med slew rate on these pads!          */
      for (count=128; count < 136; count++)
        {
         SIU.PCR[count].R = 0x0407;
        };

     /* Configure PortH[14..15] = PCR[126..127]. Additional WEx signals        */
       SIU.PCR[126].R = 0x060F;                  /* PH14 = WE2. Fast Slew, Output */
       SIU.PCR[127].R = 0x060F;                  /* PH15 = WE2. Fast Slew, Output */
```

6 Using the EBI with Nexus

It is possible to use the EBI and Nexus concurrently, but there are certain constraints that must be met for this to function. Some of these constraints are hardware related and may require changes to be made to your hardware configuration.

1. The EBI external hardware and configuration must be set to use a 16-bit data port size.
   - Set the DBM bit in the EBI_MCR register to 1 to enable a 16-bit data port. This also configures the PS bits within the EBI_BRn registers as 1 for 16-bit port size.
2. The data pins must be multiplexed to the least significant address lines ADDR[16:31]
   - Set the D16_31 bit in the EBI_MCR register to 1.
3. The most significant address line available to be used with the EBI is ADDR[14], giving an addressable space of 2^{18} or 256 KBs. The remaining address lines are multiplexed with signals that are required for Nexus.
4. The Nexus must be run in reduced data port mode using 4MDO lines rather than 8MDO. The upper MDO lines are multiplexed with signals required for the EBI.
   - This is configured by the debugger when it establishes the debug session. Please see the debugger manufacturer instructions for details on how to change this.
5. The Nexus must be run without using EVTI/EVTO. These signals are multiplexed with signals that are required for the EBI (R/$\overline{W}$ and $\overline{TA}$).
   - EVTI and EVTO are released from the Nexus by clearing the EVT_EN bit within the Nexus PCR (Port Configuration Register). This has to be done by the debug tool because it is a Nexus register. Please consult your debug manufacturer for details on how to write to this bit
   - After EVT_EN has been cleared, the ports (PF0, PF1) need to be re-assigned to the EBI functions R/W and TA within the SIU. This is not done automatically.
6. Port PF0 must be physically disconnected from the Nexus connector as the debug probe drives EVTI high after EVT_EN is cleared. If this pin is not disconnected, there is contention with the MCU R/W signal and the EBI does not function correctly.

Table 4 details the signals that are shared between the Nexus and the EBI and where there are potential conflicts between the EBI and Nexus (as listed above).

Table 4. MPC5510 EBI /Nexus Signal Descriptions

Port	Nexus Function	EBI function	Setting for Shared Nexus/EBI
TDI	TDI	—	JTAG only pins, not shared with any other function
TDO	TDO	—	
TCLK	TCLK	—	
TMS	TMS	—	
JCOMP	JCOMP	—	
PE[6]	CLKOUT	CLKOUT	Used for CLKOUT in EBI and Nexus modes
PF[0]	EVTI	R/$\overline{W}$	Automatically configured for Nexus functions. Disable EVTI/EVTO via EVT_EN bit and then re-configure SIU for respective EBI functions.
PF[1]	EVTO	TA	

Table 4. MPC5510 EBI /Nexus Signal Descriptions (continued)

Port	Nexus Function	EBI function	Setting for Shared Nexus/EBI
PF[2]	MSEO[0]	AD8	Automatically configured for Nexus functions
PF[3]	MCKO	AD9	
PF[4]	MDO[0]	AD10	
PF[5]	MDO[1]	AD11	
PF[6]	MDO[2]	AD12	
PF[7]	MDO[3]	AD13	
PF[8]	MDO[4]	AD14	With MDO width set to 4, these pins can be configured to respective EBI functions via SIU configuration.
PF[9]	MDO[5]	AD15	
PF[10]	MDO[6]	$\overline{CS}1$	
PF[11]	MDO[7]	$\overline{CS}0$	

THIS PAGE IS INTENTIONALLY BLANK

Document Number: AN3773
Rev. 0
02/2009