AVR1312: Using the XMEGA External Bus Interface

8-bit AVR® Microcontrollers

Application Note

Features

- **Supports SRAM, SDRAM and addressable peripherals**
- **Up to 16 MB address space**
- **Four independent Chip Select lines**
- **1, 2, 3 or 4 ports used for Address and Data lines**
- **SDRAM features:**
  - **Automatic refresh**
  - **4- or 8-bit data**
- **Driver source code included**

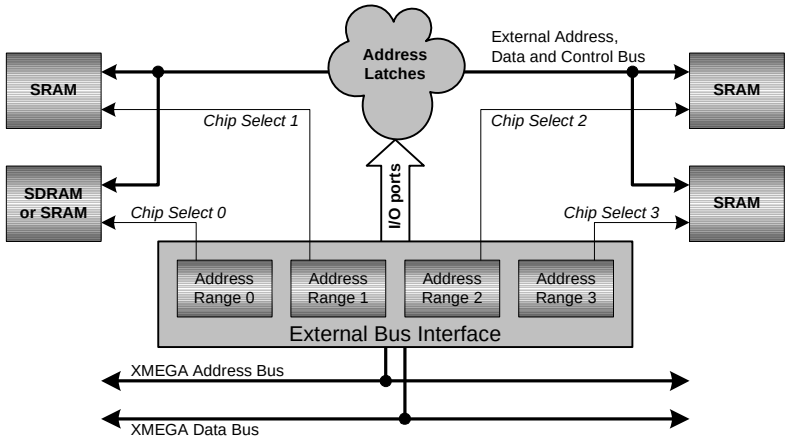
1 Introduction

The XMEGA™ External Bus Interface (EBI) is a highly flexible module for interfacing external memories and memory addressable peripherals such as LCD controllers and advanced communication controllers. The EBI module has four separate Chip Select blocks with individual address ranges and wait state control. Additional Chip Select lines can be decoded externally.

Flexible settings for address multiplexing and external latches, individual settings for the four Chip Select blocks and transparent support for SDRAM with automatic refresh make this module the perfect match for all applications using external memories and addressable peripherals.

This application note describes the basic functionality of the XMEGA EBI with code examples to get up and running quickly. A driver interface written in C is included as well.

Figure 1-1. Module Overview



Rev. 8058A-AVR-02/08

2 Module Overview

This section provides an overview of the functionality and basic configuration options of the EBI. Section 3 then walks you through the basic steps to get you up and running, with register descriptions and configuration details.

2.1 Connecting Memories and Peripherals

The EBI module can be configured to use two, three or four I/O-ports for interfacing external memories and peripherals. The ports used are shown in Table 2-1 below. The following sections describe how to connect to the ports for the different modes.

Table 2-1. EBI Port Mode

Mode	Ports Used by EBI Module
Two-port operation	Port H and J
Three-port operation	Port H, J, and K
Four-port operation	Port H, J, K, and L

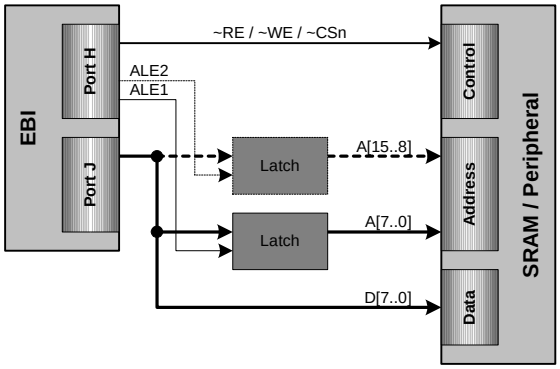
2.1.1 Two-port Interface

The two-port mode only allows for SRAM LPC (Low Pin Count) operation. SRAM LPC usually involves SRAM devices with internal latches. However, ordinary SRAM devices can be used with external latches.

With SRAM LPC, Port J is used for address and data, while Port H is used for control signals. The connections are shown in Figure 2-1 below.

The *LPC Mode* bitfield (LPCMODE) in the *EBI Control* register (CTRL) selects whether ALE2 should be used or not. Enabling ALE2 means that 16-bit addresses are used.

Figure 2-1. 8- and 16-bit address bus in two-port SRAM LPC mode



2.1.2 Three-port Interface

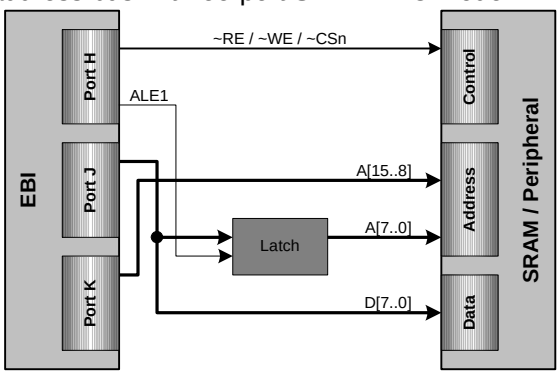
The three-port mode allows for ordinary SRAM and SDRAM connections in addition to SRAM LPC. SRAM and SRAM LPC can be used together, but SDRAM can only be used alone in three-port mode. In order to use SDRAM together with other memory types, the EBI module must be in four-port mode.

2.1.2.1 Three-port SRAM LPC

For SRAM LPC, the connections are similar to two-port mode with the addition of Port K used for bit 8 to 15 of the address. The connections for three-port SRAM LPC are shown in Figure 2-2 below.

If the ALE2 line is enabled with the *LPC Mode* bitfield (LPCMODE), Port K is not used and the connections are equal to two-port SRAM LPC, as shown in Figure 2-1 above.

Figure 2-2. 16-bit address bus in three-port SRAM LPC mode



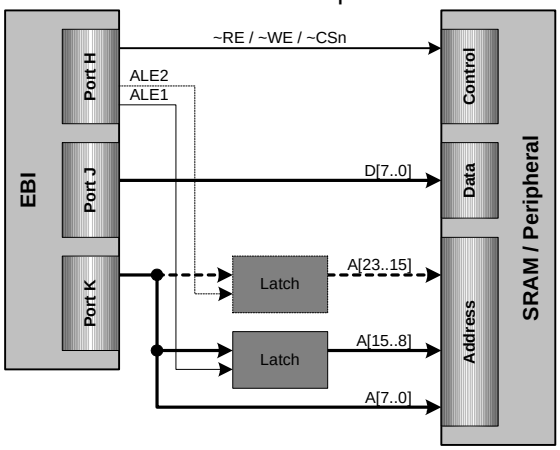
Note that three-port SRAM LPC with only ALE1 enabled is similar to the XRAM interface available on many devices in the megaAVR® family.

2.1.2.2 Three-port SRAM

Instead of multiplexing address and data on one I/O port as in SRAM LPC mode, ordinary SRAM mode uses one dedicated port for data lines and one or more ports for multiplexing address lines. For three-port SRAM mode, Port H is used for control signals as usual, Port J is used for data only, and Port K is used for multiplexing address lines.

The *SRAM Mode* bitfield (SRMODE) in the *EBI Control* register (CTRL) select whether ALE2 should be used or not. Enabling ALE2 means that 24-bit addresses are used.

Figure 2-3. 16- and 24-bit address bus in three-port SRAM mode



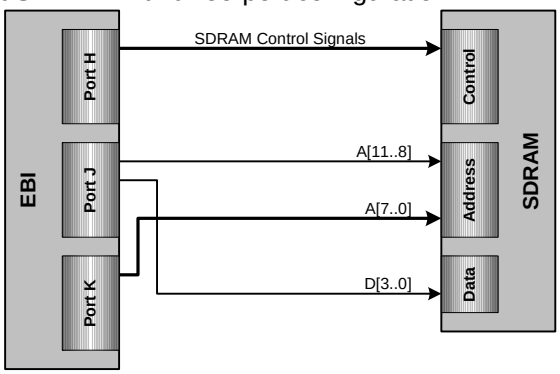
2.1.2.3 Three-port SDRAM

When SDRAM is enabled in three-port mode, neither SRAM nor SRAM LPC is supported. There are no spare pins for the Chip Select control signals on the EBI ports.

The *SDRAM Data Width* bitfield (SDDATAW) in the *EBI Control* register (CTRL) selects between 4-bit and 8-bit SDRAM data width, however only 4-bit SDRAM data width is supported in three-port mode.

The connections for SDRAM in three-port mode are shown in Figure 2-4 below.

Figure 2-4. Four-bit SDRAM with three-port configuration



2.1.3 Four-port Interface

The four-port mode allows for 8-bit SDRAM operation and SDRAM together with SRAM and SRAM LPC. Of course, SRAM and SRAM LPC operation without SDRAM is also supported in four-port mode.

Note that the connections for SRAM LPC and SRAM are slightly different if SDRAM is enabled.

2.1.3.1 Four-port SRAM LPC

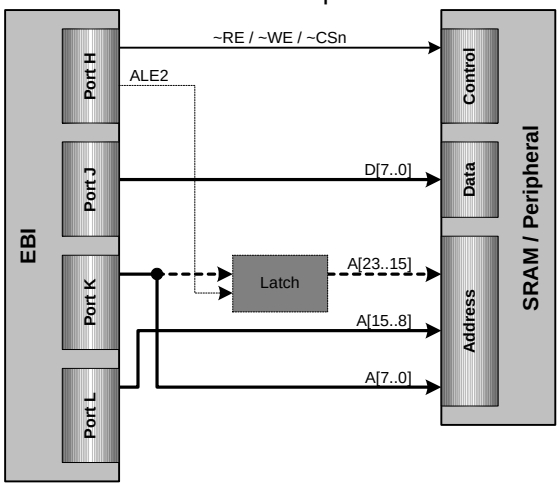
For SRAM LPC, the connections are similar to three-port mode. SRAM LPC does not make use of the fourth port.

2.1.3.2 Four-port SRAM

Compared to three-port SRAM operation, four-port SRAM removes the need for the ALE1 line. Instead, the second byte of the address is moved to the fourth port, Port L. If only 16-bit addresses are used, there is no need for an address latch. However, the ALE2 line can be used to multiplex the third address byte on Port K, similar to three-port operation.

The connections, with optional use of ALE2, are shown in Figure 2-5 below. What is not shown in the figure is that if the ALE2 line is not used, address lines A16 and A17 are available from Port H. This allows for 18-bit addressing without using latches.

Figure 2-5. 16- and 24-bit address bus in four-port SRAM mode

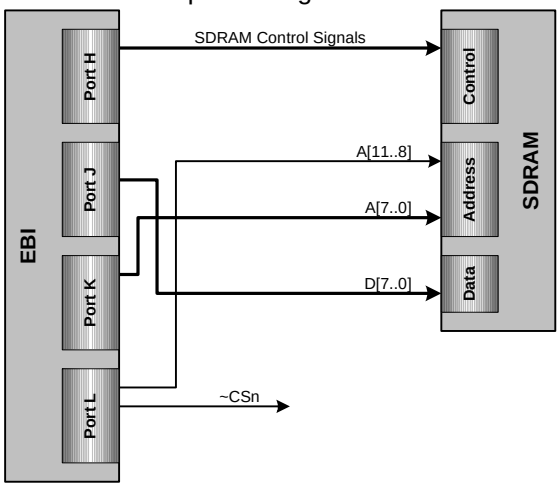


2.1.3.3 Four-port SDRAM

Compared to three-port SDRAM operation, four-port SDRAM supports simultaneous operation of SRAM and SRAM LPC. When configured in four-port SDRAM mode, the EBI interface places the Chip Select lines on the upper four bits of Port L.

The connections for eight-bit SDRAM are shown in Figure 2-6 below. Compared to four-bit SDRAM, Port J now holds all eight bits of data while the upper four address bits are moved to Port L together with the Chip Select lines.

Figure 2-6. 8-bit SDRAM with four-port configuration



If SRAM and SRAM LPC is used together with SDRAM, consider that address lines A16 to A19 are moved to the lower four bits of Port L. This applies to all SRAM and SRAM LPC configurations that share address lines A16 to A19 with the Chip Select lines. Refer to Section 4.1 below for details.

Note that four-port SRAM is not supported when SDRAM is used.

2.2 The Chip Select Blocks

The EBI module has four Chip Select lines (CS0 to CS3) that can be associated with separate address ranges. The control registers and functionality associated with each Chip Select line are logically grouped into Chip Select Blocks. The four Chip Select Blocks are shown in Figure 1-1 on page 1.

Depending on the memory types you are connecting to the EBI, the Chip Select Blocks can be configured for SRAM LPC (Low Pin Count), ordinary SRAM or SDRAM. The blocks can be configured independently, but only the CS3 block supports SDRAM.

The base address associated with each Chip Select Block must be on a 4 Kbyte boundary, and decides the location in data memory space where the connected memory hardware can be accessed. The *Base Address* register (BASEADDR) of each Chip Select Block holds this address.

The size of the memory space for each Chip Select Block is selected by the *Address Space* bitfield (ASPACE) in *Control Register A* (CTRLA). The size can be from 256 bytes up to 16M bytes.

Note that if the address space is set to anything larger than 4K bytes, the base address must be on a boundary equal to the address space. For instance, with 1 Mbyte address space for a Chip Select Block, the base address must be on a 1 Mbyte boundary.

If the address spaces overlap, the internal memory spaces have priority, followed by CS0, CS1, CS2 and then CS3 with least priority.

2.3 I/O-port configuration

The EBI module does not override I/O-port directions when enabled. The user must configure all interface pins as outputs, except the pins used for the data bus.

Control signals that are active-low, the pin output value should be set to logic one, while active-high signals should have the corresponding pin output value set to logic zero. Address lines do not care whether the output value is one or zero.

Chip Select lines should have pull-up resistors to ensure that these are kept high during start-up. This ensures that floating control signals during start-up are ignored by external modules. In modes where Chip Select lines are not controlled by the EBI module (e.g. 3-port SDRAM), General Purpose IO pins should be used to set the Chip Select lines low when needed.

For more information, please refer to the device datasheet or the application note "AVR1313: Using the XMEGA I/O-pins and External Interrupts".

2.4 Selecting External Latches

The EBI module was designed to meet the spec for 74AHC series of address latches. Refer to datasheet for details for timing requirements.

2.5 SDRAM Refresh Considerations

The EBI takes care of refreshing the SDRAM module for us. Since the only Chip Select block capable of interfacing SDRAM have the lowest priority, it could happen that other memories are being accessed when its time for a new SDRAM refresh. In that case, the EBI remembers the missed refresh and refreshes the SDRAM when other memory accesses are finished.

When in sleep mode and the clock to the EBI module is stopped, it is possible to enter Self-refresh mode for the SDRAM module. To enable Self-refresh, set the *SDRAM Self-refresh Enable* bit (SDSREN) in *Control Register B* (CTRLB) for Chip Select block 3. Note that it is not possible to access the SDRAM when in Self-refresh mode.

3 Getting Started

This section walks you through the basic steps for getting up and running with the XMEGA EBI. The necessary registers are described along with relevant bit settings.

Setting up the EBI for SRAM and SDRAM operation is an easy task, and the reader is advised to study the code example for details. SDRAM setup requires some more though, and a walkthrough is provided below.

3.1 SDRAM Setup and Initialization

Task: Setup EBI for SDRAM operation and initialize SDRAM controller.

- Select three-port or four-port interface with the *Interface Mode* bitfield (IFMODE) in the *EBI Control* register (CTRL).
- Configure IO-port directions and values according to Section 2.3 above.
- Use the *SDRAM Data Width* bitfield (SDDATAW) in the *EBI Control* register (CTRL) to select 4-bit or 8-bit data.

- Use the *SDRAM CAS Latency* bit (SDCAS) in *SDRAM Control Register A* (SDRAMCTRLA) to select 2 cycles (logic zero) or 3 cycles (logic one) CAS latency.
- Use the *SDRAM Row Bits* bit (SDROW) in *SDRAM Control Register A* (SDRAMCTRLA) to select 11-bit (logic zero) or 12-bit (logic one) row addressing.
- Use the *SDRAM Column Bits* bitfield (SDCOL) in *SDRAM Control Register A* (SDRAMCTRLA) to select 8-, 9-, 10- or 11-bit column addressing.
- Use the bitfields MRDLY, ROWCYCDLY and RPDLY bitfields in *SDRAM Control Register B* (SDRAMCTRLB) and WRDLY, ESRDLY and ROWCOLDLY in *SDRAM Control Register C* (SDRAMCTRLC) to configure various SDRAM interface delays given in peripheral clock cycles.
- Use the *SDRAM Refresh Period* register (REFRESH) to select SDRAM refresh period in peripheral clock cycles.
- Use the *SDRAM Initialization Delay* register (INITDLY) to select SDRAM initialization delay in peripheral clock cycles.
- Set the desired SDRAM base address in the *Base Address* register (BASEADDR) for Chip Select Block 3.
- Set the desired address space using the *Address Space* bitfield in *Control Register A* (CTRLA) for Chip Select Block 3.
- Enable SDRAM and start initialization sequence by setting the *Memory Mode* bitfield (MODE) to SDRAM in *Control Register A* (CTRLA) for Chip Select Block 3.

When initializing SDRAM, it is important to configure all parameters *before* enabling the Chip Select Block in SDRAM mode. The order of setting the parameters themselves are not important.

4 Advanced Features

This section introduces more advanced features and possibilities with the EBI. In-depth treatment is outside the scope of this application note and the user is advised to study the device datasheet and relevant application notes.

4.1 Chip Select Lines as Address Lines

In certain configurations, the Chip Select lines can be used as address lines instead. Say, if only Chip Select Block 2 and 3 is used, the Chip Select lines CS0 and CS1 are used as address lines A16 and A17 instead. Figure 4-1 below shows the possible configurations.

This feature only applies to configuration where address lines A16 to A19 are not already multiplexed on other ports.

Figure 4-1. Possible combinations of Chip Select lines and address lines

CS3	CS3	CS3	A19
CS2	CS2	CS2	A18
CS1	CS1	A17	A17
CS0	A16	A16	A16

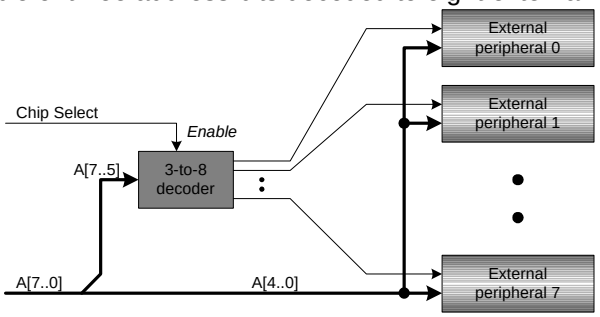
The figure shows that if only CS3 is enabled, all four CS lines are used as address lines.

4.2 Additional Chip Select Lines

If the four Chip Select lines should not be enough for your application, further address decoding can be done externally.

Say you want to access eight small external peripheral devices, all having 32 internal registers or less. Configure the Chip Select Block you want to use with 256-byte address space. The resulting 8-bit address is divided into two parts, the upper three bits selects one of the eight external peripherals, while the lower five bits addresses one out of 32 internal registers. Figure 4-2 below shows an example for such an implementation.

Figure 4-2. Example of three address bits decoded to eight external Chip Select lines



5 Driver Implementation

This application note includes a source code package with a basic EBI driver implemented in C. It is written for the IAR Embedded Workbench® compiler.

Note that this EBI driver is not intended for use with high-performance code. It is designed as a library to get started with the EBI. For timing and code space critical application development, you should access the EBI registers directly. Please refer to the driver source code and device datasheet for more details.

5.1 Files

The source code package consists of four files:

- *ebi_driver.c* – EBI driver source file
- *ebi_driver.h* – EBI driver header file
- *ebi_sram_example.c* – Example code using the driver with SRAM
  ebi_sdram_example.c – Example code using the driver with SDRAM

For a complete overview of the available driver interface functions and their use, please refer to the source code documentation.

5.2 Doxygen Documentation

All source code is prepared for automatic documentation generation using Doxygen. Doxygen is a tool for generating documentation from source code by analyzing the source code and using special keywords. For more details about Doxygen please visit http://sourceforge.net/projects/doxygen. Precompiled Doxygen documentation is also supplied with the source code accompanying this application note, available from the *readme.html* file in the source code folder.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

8058A-AVR-02/08