EFM32®

... the world's most energy friendly microcontrollers

External Bus Interface

AN0034 - Application Note

Introduction

This application note shows how to use the EFM32's parallel bus interface, the EBI (External Bus Interface), to access an external SRAM or other parallel interface devices. The included software example demonstrates access to the external memory on either the EFM32-GXX-DK or the EFM32GG-DK3750 development kit.

This application note includes:

- This PDF document
- Source files (zip)
  - Example C-code
  - Multiple IDE projects

ARM ZERO TINY GECKO LEOPARD GIANT WONDER

SILICON LABS

1 Parallel Bus Introduction

A parallel bus typically transfers data between devices in a computer system through several electrical connections were the bits are transferred in parallel. A parallel bus most commonly consist of three set of signals; the data lines, called the data bus, an address bus and in addition there are several control signals.

Most often the transfers on the parallel bus are initiated and controlled by a bus master. The master device has full control over the control signals and address bus. In this application note only this type of parallel bus, with one master device, is considered.

In short, the data bus transfers the actual data. The address bus defines were the data belong in an address space. And the control signals define the direction of the data transfer and which devices on the bus the data is transferred between.

1.1 Data and Address Bus

The data bus typically consist of as many electrical signals as there are bits in the transfer word size. For example if the data bus has 8 signals, 8 bits can be transferred at a time and the transfer word size is 1 byte. 16 signals correspond to a 2 byte word size and so on. The control signals take care of signalling when new data are in a valid state on the data bus. This is important since devices reading data from the bus must know when the electrical values are stable and represents valid data.

The address bus consists of several electrical signals that typically represent where in memory the data on the data bus belongs to. In the case were an external memory device is connected on the parallel bus, the address bus directly defines which address in the memory the data on the data bus should be read from or written to.

The width of the address bus, or the number of signals it consists of, directly defines the maximum possible number of words that can be addressed in an external memory device. If other types of devices are attached, for example an ADC or DAC with a parallel bus interface, the address bus is often not needed. It can however be utilized as additional control signals for the external device.

1.2 Control Signals

In addition to the data bus and address bus, a parallel bus interface contains several control signals. If the parallel bus connects just two devices, only two control signals are strictly needed; *Read Enable* and *Write Enable*. These two signals are driven by the bus master, typically the MCU in the system.

Read Enable signals that the bus master wants to read data from the external device. It is typically pulled low when active. This signals to the external device that it should read the address bus and put the corresponding data on the data bus.

Write Enable signals that the bus master wants to write data to the external device. It is typically pulled low when active. This signals to the external device that it should read/decode the address bus and data bus and write the data into its own memory.

If several devices are connected on the bus, additional signals are needed to activate only one device at a time. These are often called *Chip Select* signals. Typically one is needed per external device on the bus in addition to the bus master. In some cases the most significant bits of the address bus can also be used as chip select signals.

The chip select signals can in principle be thought of as an extension to the address bus, but chip select signals often have some special properties that the address bus signals does not have, which are related to timing. This is discussed further in the next section.

If the parallel bus multiplexes the address and data bus, a latch enable signal is needed to control an external address latch. This signal is often called ALE (Address Latch Enable). Multiplexed operation with an external address latch is described later in this document.

Note

This application note only describes asynchronous parallel buses without clock signals. The EFM32 only supports asynchronous parallel bus operation.

1.3 Timing

All electrical signals travelling through an electrical wire have a finite propagation speed. Because of this, the different devices connected to the same parallel bus will not interpret the signal on the bus identically at all times. The signal is typically the voltage level which can be either high or low to denote the binary values of 1 or 0.

When the signal changes from one value to the other, the voltage change will travel through the parallel bus and propagate into each device at a high, but finite speed. The delay from the moment one device puts a new binary value on the bus, to the moment all the other devices interprets the voltage level as the same value must be accounted for when the bus master asserts/deasserts the control signals.

Often, the only difference between several devices connected on the parallel bus is their timing requirements. Some devices require a longer period before and/or after the read/write enable signals are asserted to allow the electrical signals on the address and data bus to propagate and settle within the device itself. These requirements are called setup and hold timings.

Certain setup and hold timing requirements apply for all devices connected to a parallel bus, even the bus master. The propagation delay in the printed circuit board must also be take into account when calculating the necessary timing that the bus master must adhere to when controlling bus accesses.

Often the master device can be configured to use different timing delays for the control signals for each chip select signals. This is useful if two devices on the bus require different timing. This relieves the software running in the MCU or bus master of the job of changing timing when different devices are accessed on the bus.

2 The EFM32 EBI

The parallel bus interface present on EFM32 microcontrollers is called; EBI or External Bus Interface. It is a versatile asynchronous parallel address/data bus that provides access to common external parallel interface devices such as SRAM, FLASH, ADCs and LCDs. The interface is memory mapped into the address bus of the Cortex-M3, which enables seamless software access without the need for IO-level access each time a read or write is performed.

Since the devices connected through the EBI appear as a part of the EFM32s internal memory map, they are simple to use. When the processor performs read or writes to the address range of the EBI, the EBI handles data transfer to and from the external device.

The EBI is available in Energy Mode 0 and Energy Mode 1 and may be interfaced by the DMA, thus enabling autonomous operation in EM1.

The data and address lines can be multiplexed in order to reduce the number of pins required to interface the external devices. The timing is adjustable and individual per chip select bank to meet specifications of the external devices. The interface is limited to asynchronous devices (no clock signal is available).

There are differences in functionality of the EBI interface between EFM32 device families. The features discussed in this document are present in the EBI interface of EFM32 Giant Gecko and Leopard Gecko devices, but some are absent in the EFM32 Gecko EBI interface. Please refer to the reference manual for your device for an accurate overview of the EBI features available.

Note

Some EBI features only available on giant, wonder and leopard devices include, but is not limited to: Non-multiplexed operation, individual timing per bank, unaligned access, variable word-size access.

2.1 Memory Mapping

The EFM32 EBI interface is memory mapped. This means that the external devices connected are accessed by software in the EFM32 through certain address ranges in memory. For example reading data from an external device is done simply by reading data from a certain memory address. Likewise, writing data to the same address in the same external device is done by writing to the same memory address in the EFM32.

The address map is divided into 4 banks which corresponds to the 4 chip select signals that are available. An accurate description of the addressable area of each memory bank, the location of the banks and the division between code space and data space can be found in the reference manual for the different device families.

2.2 Bus Control Signals

This section lists the most important control signals that are available with the EFM32 external bus interface in addition to the address and data bus. The naming convention will be used throughout the rest of this document, and it is the same as the naming convention used in the EFM32 reference manuals. Note that the EBI on the EFM32 Gecko series MCU only has a subset of the functionality described below.

- Data Bus, typically denoted: D[xx:0] where xx is the most significant data bit available. Would be D[7:0] or D[15:0] for 8 or 16 bit data buses.
- Address Bus, typically denoted: A[xx:0] where xx is the most significant address bit available.
- Read Enable, REn. This signal can also be called Output Enable, OEn. The n (can be bar or #) denotes that it is an active low signal.
- Write Enable, WEn. The n (can be bar or #) denotes that it is an active low signal.
- Chip Select, CSn. This signal can also be called Chip Enable, CEn. The n (can be bar or #) denotes that it is an active low signal. The EFM32 EBI has 4 of these, for connecting up to 4 different devices

on the external bus interface. Each of the chip select signals are related to its own memory range, and can be configured with individual timing.

- Byte Lane, BLn. This signal typically consists of two bits, it can be thought of as enable signal for the high and low byte part of the data bus. Often denoted as two separate signals, LBn, Lower Byte and UBn, Upper Byte. These signals typically affect both read and write operations. Often the byte select signals are not needed, remember to define the signals on the external device in the correct state if they are not connected to the EFM32.
- Address Latch Enable, ALE. This signal goes directly to the address latch in multiplexed operation, not needed for non-multiplexed buses.

See Figure 2.2 (p. 6) for how an asynchronous SRAM would be connected with respect to control signals to the EFM32 EBI interface. Notice that the connection in Figure 2.2 (p. 6) is non-multiplexed with 16 bit data bus and 20 address bits, it has separate address lines for all the bits in the address.

Note

Pull-resistors are recommended on the control signals to have a defined bus state when the EFM32 is in reset and before the EBI interface is configured.

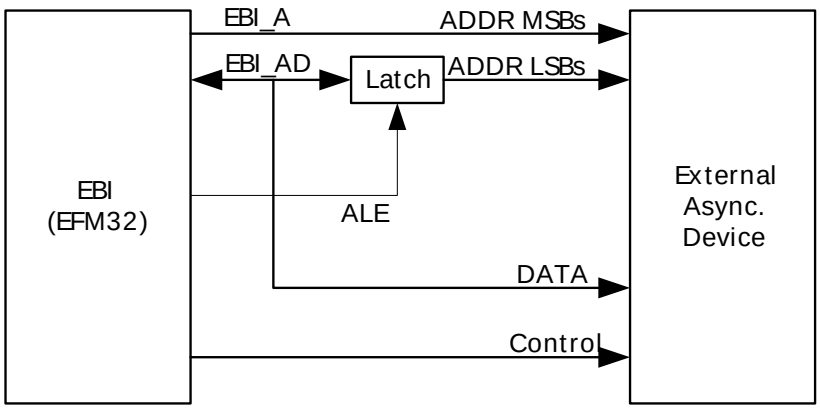
2.3 EBI Operating Modes

The EFM32 EBI peripheral can operate in several different modes which mainly differs in the data word size and if the address bus is multiplexed or not. Multiplexing the address and data bus is useful for increasing the amount of addressable external memory without using too many GPIO pins. Multiplexing comes at the expense of a slight decrease in performance and the need for an external address latch.

2.3.1 Multiplexed Modes

The data bus width of the EFM32 EBI is 16 bits. When multiplexing the address and data bus, these 16 signal lines are first used for putting out the least significant 16 bits of the address, which are then held by the external address latch. Then these 16 signal lines are either used for 16 bits of data, or 8 bits of data and the remaining 8 bits of the address. See Figure 2.1 (p. 5) for an overview of the signals and address latch needed for multiplexed operation. The rest of the control signals are the same as in the non-multiplexed operation example, see Figure 2.2 (p. 6) .

Figure 2.1. EBI Address Latch Setup

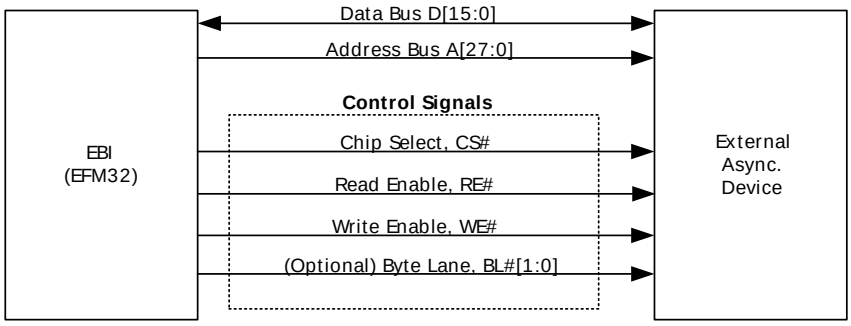


2.3.2 Non-Multiplexed Modes

For non-multiplexed modes, the data bus is only used for transferring data and the external address latch is not required. This mode can be simpler to implement since it does not require an external latch. It also offers the fastest operation at the expense of more GPIO pins used for the same addressable space.

The different multiplexed or non-multiplexed modes differ mainly in how wide the data bus is. It can either be 8 bit or 16 bit wide. One important thing to remember when selecting data bus width is that for 16 bit wide data bus the least significant address bit represents 16 bit increments instead of 8 bit increments. Which is equivalent to the statement that the address bus is shifted one place to the right as it is described in the reference manual. See Figure 2.2 (p. 6) for an overview of the different signals needed for non-multiplexed operation.

Figure 2.2. EBI Non-Multiplexed Operation



Note

The additional address pins of the EFM32 EBI interface used in non-multiplexed operation can always be used to extend the address beyond the 16 or 24 bit multiplexed address limitation . Please see the reference manual for more information on the different multiplexing modes and extended addressing possibilities.

2.4 Timing Configuration

The EBI timing configuration consist of a set of three parameters for both read operations and write operations. In addition two parameters defines the timing of the multiplexed address latch operation.

The three main parameters for read and write operations are *Setup Time*, *Strobe Time* and *Hold Time*.

- *Setup Time* defines how long the address is available on the bus before the REn or WEn signal is asserted.
- *Strobe Time* defines how long the REn or WEn signal is asserted.
- *Hold Time* defines how long the address and data lines are held after the REn or WEn signal is deasserted, before a new transfers starts or the chip select signal is deasserted.

The following figures illustrates the three different timing periods for the read and write operations respectively.

Figure 2.3. EBI Non-Multiplexed Read Timing

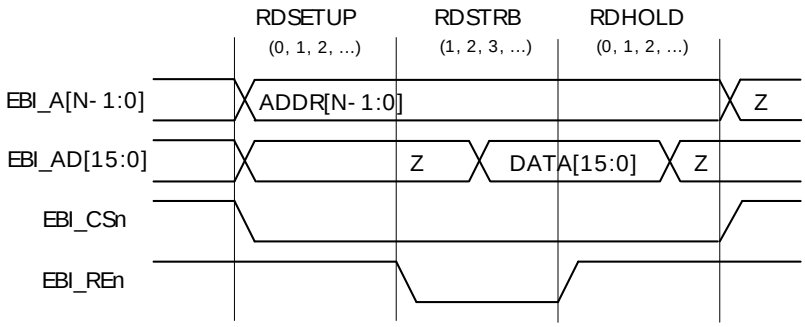
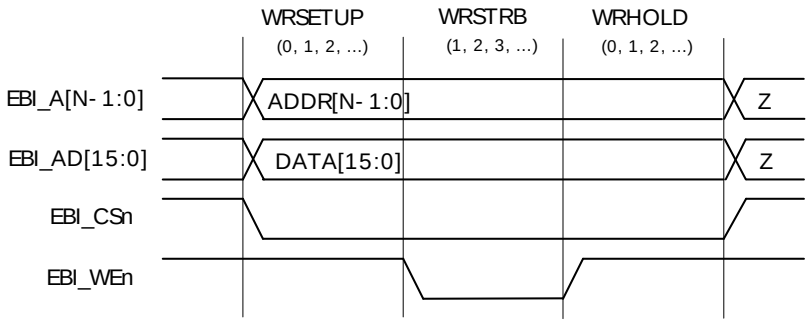


Figure 2.4. EBI Non-Multiplexed Write Timing



For multiplexed operation, configuration of *Address Setup* and *Address Hold* timings are also provided to control the timing of the external address latch operation. Please see the reference manual for more detailed timing diagrams of multiplexed operation with address latch.

Configuring the different timing parameters correctly require the designer of the system to look at the worst case timing parameters for each device connected on the bus, in addition the propagation delay in the printed circuit board must be taken into account. Just trying out how short the timing intervals can be will almost certainly result in an unstable system. A set of parameters that work at room temperature and a high supply voltage might not work if the system is heated up or at lower supply voltage.

2.5 Special Features

The EBI interface supports some special features that are outside the scope of this document:

- Individual active high / active low setting of interface control signals per memory bank.
- Slave read/write cycle extension per memory bank.
- Page mode read.
- NAND Flash support.
- Automatic translation when AHB transaction width and memory width differ.
- Configurable prefetch from external device.
- Write buffer to limit stalling of the Cortex-M3 or DMA.
- TFT Direct Drive with 2D DMA copy and support for masking and alpha blending.

The reference manual describes each of these additional features in detail.

3 Software Example

The supplied software example demonstrates how to access an external address mapped memory device with the EBI interface. The example will run on both the EFM32-G290-DK and the DK3750/DK3650 development kits. Since the external memory device is attached differently on the two development kit versions, please make sure to select the correct example project for your kit.

The software example simply writes a test-array with some random data to the external memory and reads it back again. Upon finish, the software is stuck in one of two while(1)-loops indicating success or failure. The following sub-sections describe relevant hardware information for the two different development kit versions.

When developing a custom design with EBI and external parallel bus components, the EBI_Init() function in emlib can be used directly without the BSP (Board Support Package)-library. The BSP library can still be used as a reference on how to configure both the EBI and corresponding GPIO pins correctly for EBI operation.

3.1 Development Kit Hardware Description

The external bus interface is connected a bit differently on the different development kits that have been released for the EFM32 devices. This section describes the most important details related to the two included software examples.

3.1.1 EFM32-GXX-DK Hardware Description

The EFM32-GXX-DK Gecko development kit include a single chip 4 Mbit parallel bus SRAM. This is a 16-bit wide data path memory (as such it is organized as 256K words of 16 bits each). The exact part number is CY62147EV30LL-45BVXI from Cypress.

The EFM32G290 MCU board (without an LCD) is needed for this application, in conjunction with the EFM32-G2xx-DK. (Because of the presence of the LCD on the EFM32G890 MCU board, this board with its corresponding DK has limited EBI support.)

On the EFM32-G2xx-DK kits, the role of the "latch" is implemented on the FPGA known as the "board controller". Software support is available to configure the board controller in order to access the SRAM via EBI (and disable SPI) in either 8-bit or 16-bit data width mode. The AEM STATE must be set to EFM. This state can be toggled by pushing the AEM button on the DK. The status is indicated in the top right hand corner of the TFT display.

The SRAM memory on the EFM32-GXX-DK is mapped starting at address 0x84000000 (When accessing this bank, the EBI_CS1 chip select signal is automatically pulled low to select the SRAM).

3.1.2 DK3750/3650/3550 Hardware Description

The newer EFM32 DK3x50 development kits include a single chip 32 Mbit parallel bus PSRAM (Pseudo SRAM). This is a 16-bit wide data path memory (as such it is organized as 2048K words of 16 bits each). The exact part number on the current DKs is MT45W2MW16PGA-70 IT from Micron.

On the DK3750/DK3650/DK3550, the address latch is a 16bit D-Type Latch (74LVCH16373). All the connections also pass through analog switches to reduce leakage when the EBI is not in use. This means that to accessed the external devices through the EBI interface, the board controller must be instructed to turn on these switches. The software example demonstrates how to use the board support package functions to do this. Also, the AEM STATE must be set to EFM. This state can be toggled by pushing the AEM button on the DK. The status is indicated in the top right hand corner of the TFT display.

The SRAM memory on the DK3750/DK3650/DK3550 is mapped starting at address 0x88000000 (When accessing this bank, the EBI_CS2 chip select signal is automatically pulled low to select the SRAM).

In the supplied software example, all necessary board controller configuration, except the AEM state switch, is done from software. The BSP (Board Support Package) library is used for configuring the board, this makes for a completely jumper-free development kit. To understand what actually happens, the user must look at the *BSP_Init(BSP_INIT_DEFAULT)*-function.

4 Further Reading and Examples

Many of the examples included with Simplicity Studio use the external bus interface in some way without explicitly stating that they do in their description. The following list includes several application notes and examples which use the external bus interface in some way. The examples and their description is a good source for more information on this topic. Please note that the list is not exhaustive as we update and release new examples continuously.

- Application Notes
  - AN0047 Interfacing Graphical Displays, demonstrates both TFT direct drive and 8080-mode communication with the TFT display on the DK3750/3650.
- DK3750 Kit Examples
  - NOR-Flash Example, demonstrates communication with the NOR-Flash on the Development Kit, connected to the EFM32 through the EBI-interface.
  - Scroller Example, demonstrates TFT Direct Drive with external frame buffer in SRAM.
  - TFT Example, demonstrates how to communicate with the external TFT controller over the EBI without direct drive.
  - Usbdmsd, USB Device, Mass Storage Device demo, utilizes both the external NOR-Flash and External PSRAM over EBI to store data.

The EBI chapter in the Giant Gecko reference manual includes description of the different modes of the EBI, timing and additional features such as prefetch, NAND-flash and TFT-Direct Drive mode.

5 Revision History

5.1 Revision 1.08

2013-09-03

New cover layout

5.2 Revision 1.07

2013-05-08

Added software projects for ARM-GCC and Atollic TrueStudio.

5.3 Revision 1.06

2013-01-07

Added information about Giant Gecko EBI and restructured document.

Modified software example to work with several development kit revisions.

5.4 Revision 1.05

2012-11-12

Adapted software projects to new kit-driver and bsp structure.

5.5 Revision 1.04

2012-04-20

Adapted software projects to new peripheral library naming and CMSIS_V3.

5.6 Revision 1.03

2012-03-14

Fixed compilation error in CodeSourcery projects.

5.7 Revision 1.02

2011-10-21

Updated IDE project paths with new kits directory.

5.8 Revision 1.01

2011-07-18

Fixed errors in main.c and added line to wait for AEM state.

Added note in document that AEM state must be waited for.

5.9 Revision 1.00

2011-03-28

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

A.2 Trademark Information

Silicon Laboratories Inc., Silicon Laboratories, the Silicon Labs logo, Energy Micro, EFM, EFM32, EFR, logo and combinations thereof, and others are the registered trademarks or trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

B Contact Information

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701

Please visit the Silicon Labs Technical Support web page:
http://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Table of Contents

List of Figures

silabs.com


ZERO
ARM Cortex-M0+
TINY
ARM Cortex-M3
GECKO
ARM Cortex-M3
LEOPARD
ARM Cortex-M3
GIANT
ARM Cortex-M3
WONDER
ARM Cortex-M4