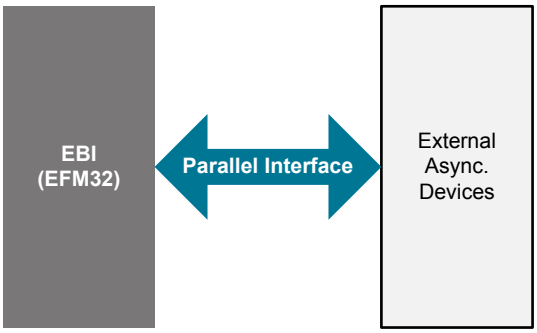
AN0034: External Bus Interface (EBI)

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 EFM32G-DK3550 or the EFM32GG-DK3750 development kit.

For more information on External Bus Interface for the graphical displays on EFM32 devices, see *AN0047: Interfacing Graphical Displays*.

KEY POINTS

- Memory mapping and bus control signals
- EBI multiplexed and non-multiplexed modes operation
- Timing configuration and special features
- an0034-efm32-ebi.zip



1. Device Compatibility

This application note supports multiple device families, and some functionality is different depending on the device.

EFM32 Series 0:

- EFM32 Gecko (EFM32G)
- EFM32 Giant Gecko (EFM32GG)
- EFM32 Wonder Gecko (EFM32WG)
- EFM32 Leopard Gecko (EFM32LG)

EFM32 Series 1:

- EFM32 Giant Gecko GG11 (EFM32GG11)
- EFM32 Giant Gecko GG12 (EFM32GG12)

Note: The EBI capability is only available on packages with 100 pins or above.

2. Parallel Bus Introduction

A parallel bus typically transfers data between devices in a computer system through several electrical connections where 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 where 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.

2.1 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 where 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.

2.2 Bus 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.

2.3 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 taken 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.

3. The EFM32 External Bus Interface (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/M4, 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 EFM32's 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 (EM0) and Energy Mode 1 (EM1) 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, Wonder Gecko, Leopard Gecko, and Giant Gecko GG11/12 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 are not available on EFM32G devices include, but are not limited to: Non-multiplexed operation, individual timing per bank, unaligned access, variable word-size access, NAND flash support.

3.1 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 correspond 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 Table 3.1 Memory Map on page 6 and the reference manual for the different device families.

Table 3.1. Memory Map

Chip Select	Address Ranage	Size	Code Execution	Instruction Cache	DMA Access
Code Space (ALTMAP in EBI_CTRL Register = 0 or 1)					
CS0	0x12000000 - 0x13FFFFFF	32 MB	Yes	Yes	–
CS1	0x14000000 - 0x15FFFFFF	32 MB	Yes	Yes	–
CS2	0x16000000 - 0x17FFFFFF	32 MB	Yes	Yes	–
CS3	0x18000000 - 0x1FFFFFFF	128 MB	Yes	Yes	–
EBI Region (ALTMAP in EBI_CTRL Register = 0)					
CS0	0x80000000 - 0x83FFFFFF	64 MB	Yes	–	Yes
CS1	0x84000000 - 0x87FFFFFF	64 MB	Yes	–	Yes
CS2	0x88000000 - 0x8BFFFFFF	64 MB	Yes	–	Yes
CS3	0x8C000000 - 0x8FFFFFFF	64 MB	Yes	–	Yes
EBI Region (ALTMAP in EBI_CTRL Register = 1)					
CS0	0x80000000 - 0x8FFFFFFF	256 MB	Yes	–	Yes
CS1	0x90000000 - 0x9FFFFFFF	256 MB	Yes	–	Yes
CS2	0xA0000000 - 0xAFFFFFFF	256 MB	Yes	–	Yes
CS3	0xB0000000 - 0xBFFFFFFF	256 MB	Yes	–	Yes

Note:

Address area from 0xA0000000 to 0xC0000000 is marked NX (no-execute) by default.

EBI code execution and instruction cache are not available on EFM32G devices.

3.2 Memory Bank Setting

Depending on the setting of the ITS (Individual Timing Set) bitfield in the `EBI_CTRL` register. The external device behavior, including for example data width, timing definitions, page mode operation, and pin polarities, is either defined for all memory banks at once or individually per bank.

Note: There is no individual setting per memory bank on EFM32G devices.

Table 3.2. Memory Bank Setting

ITS in EBI_CTRL register = 0	ITS in EBI_CTRL register = 1

Memory Bank 0 to 3 Setting:
• EBI_CTRL
  • MODE
  • NOIDEL
  • ARDYEN
  • ARDYTODIS
  • BL
• EBI_ADDRTIMING
• EBI_RDTIMING
• EBI_WRTIMING
• EBI_POLARITY
	Memory Bank 0 Setting:
	• EBI_CTRL
	  • MODE
	  • NOIDEL
	  • ARDYEN
	  • ARDYTODIS
	  • BL
	• EBI_ADDRTIMING
	• EBI_RDTIMING
	• EBI_WRTIMING
	• EBI_POLARITY

	Memory Bank 1 to 3 Setting (n = 1 - 3 for bank 1 to 3):
	• EBI_CTRL
	  • MODEn
	  • NOIDELn
	  • ARDYnEN
	  • ARDYTOnDIS
	  • BLn
	• EBI_ADDRTIMINGn
	• EBI_RDTIMINGn
	• EBI_WRTIMINGn
	• EBI_POLARITYn

3.3 Data and Address Bus

This section lists the address and data bus signals that are available with the EFM32 external bus interface. 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.

Table 3.3. EBI Data and Address Bus

Signal	Type	Function
EBI_AD[15:0]	I/O	Address Bus: EBI_AD[15:8] is used for non-multiplexed 8-bit address mode and EBI_AD[15:0] is used for multiplexed 16-bit and 24-bit address mode (external latch is required). Data Bus: Would be EBI_AD[7:0] or EBI_AD[15:0] for 8 or 16 bit data buses.
EBI_A[27:0]	O/P	EBI_A[27:0] is used to extend the address range for multiplexed and non-multiplexed address modes. Up to 28 address bits can be individually enabled on the EBI_A address lines providing up to 256 MB of address space per memory bank.
Note: The EBI_A[27:0] is not available on the EFM32G devices.		

3.4 Bus Control Signals

This section lists the bus control signals that are available with the EFM32 external bus interface. 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.

Table 3.4. EBI Bus Control Signals

Signal	Type	Function
EBI_WEn	O/P	Write Enable.
EBI_REn	O/P	Read Enable: This signal can also be called Output Enable (OEn) signal.
EBI_BLn[1:0]	O/P	Byte Lane: 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.
EBI_CSn[3:0]	O/P	Chip Select: This signal can also be called Chip Enable (CEn). 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.
EBI_ARDY	I/P	Address Ready: This signal can also be called Wait (WAIT). Some external devices are able to indicate that they are not finished with either write or read operation by asserting the ARDY line.
EBI_ALE	O/P	Address Latch Enable: This signal goes directly to the address latch in multiplexed operation, not needed for non-multiplexed buses.
EBI_NANDWEn	O/P	NAND Flash Write Enable.
EBI_NANDREn	O/P	NAND Flash Read Enable.

Note:

The n denotes that it is by default an active low signal.

It is possible to individually configure the control signals (except EBI_NANDWEn and EBI_NANDREn) to be active high/low by setting or clearing the appropriate bits in the EBI_POLARITY or EBI_POLARITYn register.

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.

The EBI_BLn[1:0], EBI_NANDWEn, and EBI_NANDREn are not available on the EFM32G devices.

This table does not include the EBI control signals for TFT Direct Drive.

3.5 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.

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.

3.5.1 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 or most 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 least significant 8 bits of the address.

The additional address pins of the EFM32 EBI interface can always be used to extend the address beyond the 16- or 24-bit multiplexed address limitation. See Figure 3.1 EBI Multiplexed Operation on page 9 and Table 3.5 Multiplexed Modes Data and Address Bus on page 9 for an overview of the signals and address latch needed for multiplexed operation.

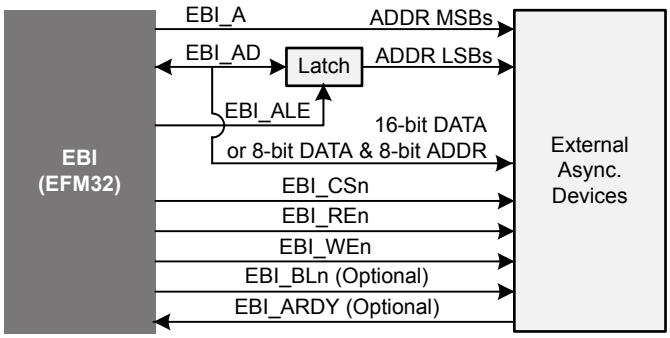


Figure 3.1. EBI Multiplexed Operation

Table 3.5. Multiplexed Modes Data and Address Bus

EBI Data and Address Bus	External Asynchronous Device
Multiplexed 16-bit Data, 16-bit Address Mode	
EBI_AD[15:0]	Address bus (A0-A15) through external address latch.
EBI_AD[15:0]	Data bus (D0-D15).
EBI_A[27:0]	Not in use, can use as peripheral pin or GPIO.
Multiplexed 16-bit Data, N-bit Address Mode (Not support on EFM32G devices)	
EBI_AD[15:0]	Address bus (A0-A15) through external address latch.
EBI_AD[15:0]	Data bus (D0-D15).
EBI_A[15:0]	Not in use, can use as peripheral pin or GPIO.
EBI_A[n:16], where n is 16 to 27	Address bus (A16-An), unused address pins can use as peripheral pin or GPIO.
Multiplexed 8-bit Data, 24-bit Address Mode	
EBI_AD[15:0]	Address bus (A8-A23) through external address latch.
EBI_AD[15:8]	Address bus (A0-A7).
EBI_AD[7:0]	Data bus (D0-D7).
EBI_A[27:0]	Not in use, can use as peripheral pin or GPIO.
Multiplexed 8-bit Data, N-bit Address Mode (Not support on EFM32G devices)	
EBI_AD[15:0]	Address bus (A8-A23) through external address latch.
EBI_AD[15:8]	Address bus (A0-A7).
EBI_AD[7:0]	Data bus (D0-D7).
EBI_A[23:0]	Not in use, can use as peripheral pin or GPIO.

EBI Data and Address Bus	External Asynchronous Device
EBI_A[n:24], where n is 24 to 27	Address bus (A24-An), unused address pins can use as peripheral pin or GPIO.

3.5.2 Non-Multiplexed Modes

For non-multiplexed modes, the data bus is used for transferring 16-bit data or 8-bit data and 8-bit address. This mode can be simpler to implement since it does not require an external address latch. It also offers the fastest operation at the expense of more GPIO pins used for the same addressable space. See Figure 3.2 EBI Non-Multiplexed Operation on page 10 and Table 3.6 Non-Multiplexed Modes Data and Address Bus on page 10 for an overview of the different signals needed for non-multiplexed operation.

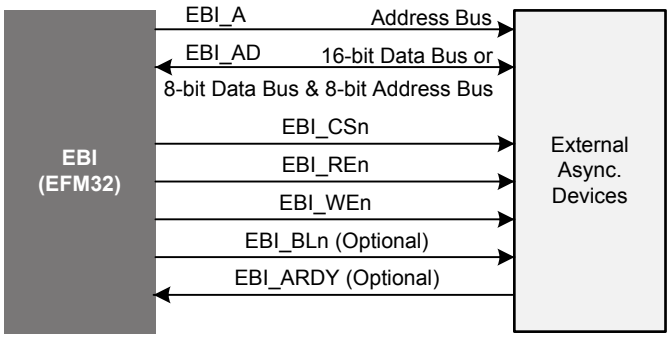


Figure 3.2. EBI Non-Multiplexed Operation

Table 3.6. Non-Multiplexed Modes Data and Address Bus

EBI Data and Address Bus	External Asynchronous Device
Non-Multiplexed 8-Bit Data, 8-Bit Address Mode	
EBI_AD[15:8]	Address bus (A0-A7).
EBI_AD[7:0]	Data bus (D0-D7).
EBI_A[27:0]	Not in use, can use as peripheral pin or GPIO.
Non-Multiplexed 8-Bit Data, N-Bit Address Mode (Not support on EFM32G devices)	
EBI_AD[15:8]	Address bus (A0-A7).
EBI_AD[7:0]	Data bus (D0-D7).
EBI_A[7:0]	Not in use, can use as peripheral pin or GPIO.
EBI_A[n:8], where n is 8 to 27	Address bus (A8-An), unused address pins can use as peripheral pin or GPIO.
Non-Multiplexed 16-Bit Data, N-Bit Address Mode (Not support on EFM32G devices)	
EBI_AD[15:0]	Data bus (D0-D15).
EBI_A[n:0], where n is 0 to 27	Address bus (A0-An), unused address pins can use as peripheral pin or GPIO.

3.6 Timing Configuration

The EBI timing configuration consists of a set of three parameters for both read operations and write operations. In addition, two parameters define 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 (RDSETUP and WRSETUP)*** defines how long the address is available on the bus before the `REn` or `WEn` signal is asserted.
- ***Strobe Time (RDSTRB and WRSTRB)*** defines how long the `REn` or `WEn` signal is asserted.
- ***Hold Time (RDHOLD and WRHOLD)*** 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.

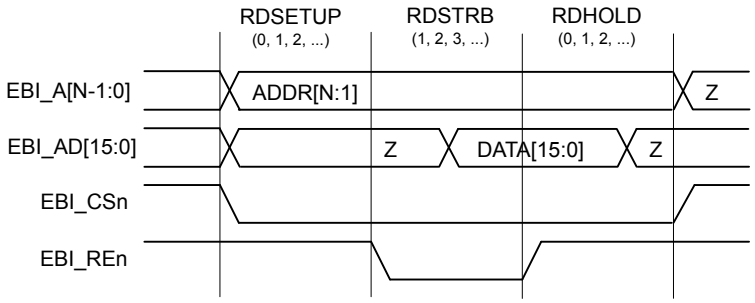


Figure 3.3. EBI Non-Multiplexed Read Timing

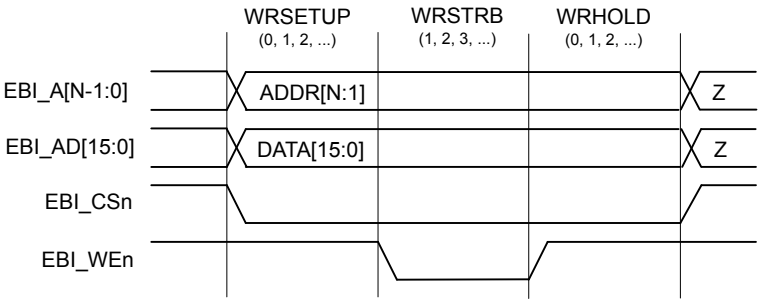


Figure 3.4. EBI Non-Multiplexed Write Timing

For multiplexed operation, configuration of ***Address Setup (ADDRSETUP)*** and ***Address Hold (ADDRHOLD)*** 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 requires the designer of the system to look at the worst-case timing parameters for each device connected on the bus. Additionally, 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.

Note:

The ***ADDRSETUP***, ***RDSTRB***, and ***WRSTRB*** all have a minimum duration of 1 cycle, which is set by hardware in case the bitfield is programmed to 0.

The EBI is clocked by HFCORECLK (maximum 32 MHz or 48 MHz) on EFM32 Series 0 and HFCLK (maximum 72 MHz) on EFM32 Series 1.

3.7 Pin Configuration

The EBI pins can be enabled individually or in group by the bitfields of the `EBI_ROUTE` register (EFM32 Series 0) or `EBI_ROUTEPEN` register (EFM32 Series 1). See the following Table 3.7 EBI Pin Configuration on page 12 for details on EBI pin configuration.

Table 3.7. EBI Pin Configuration

Bitfield of EBI_ROUTE or EBI_ROUTEPEN Register	Enabled Pin
EBIPEN	EBI_AD[15:0], EBI_WEn and EBI_REn
CS0PEN	EBI_CS0
CS1PEN	EBI_CS1
CS2PEN	EBI_CS2
CS3PEN	EBI_CS3
ALEPEN	EBI_ALE
ARDYPEN	EBI_ARDY
BLPEN	EBI_BL[1:0]
NANDPEN	EBI_NANDREn and EBI_NANDWEn
ALB	Lower and Upper bound of EBI_A[U:L]
APEN	
TFTPEN	EBI_DCLK, EBI_VSYNC and EBI_HSYNC
DATAENPEN	EBI_DATAEN
CSTFTPEN	EBI_CSTFT
Note: The BLPEN, NANDPEN, ALB, APEN, TFTPEN, DATAENPEN, and CSTFTPEN bitfields are not available on EFM32G devices. The EBI_AD[15:8] cannot use as GPIO even these pins do not use in 8-bit NAND Flash mode.	

The alternative pin locations can be chosen by the bitfields of the EBI_ROUTE register (EFM32 Series 0) or EBI_ROUTELOC0 and EBI_ROUTELOC1 registers (EFM32 Series 1). See the following Table 3.8 EBI Alternative Location on page 13 for details on EBI pin location setting.

Table 3.8. EBI Alternative Location

Bitfield	Pin Location
EBI_ROUTE Register	
LOCATION	All EBI pins
EBI_ROUTELOC0 Register	
EBILOC	EBI_WEn, EBI_REn, EBI_BL[1:0] and EBI_ALE
CSLOC	EBI_CS0, EBI_CS1, EBI_CS2 and EBI_CS3
NANDLOC	EBI_NANDREn and EBI_NANDWEn
TFTLOC	EBI_DCLK, EBI_VSYNC, EBI_HSYNC, EBI_DATAEN and EBI_CSTFT
EBI_ROUTELOC1 Register	
ADLOC	EBI_AD[15:0]
ALOC	EBI_A[27:0]
RDYLOC	EBI_ARDY
Note: The LOCATION bitfield in EBI_ROUTE register is not available on EFM32G devices.	

3.8 Special Features

The EBI interface supports some special features (not available on EFM32G devices) that are outside the scope of this document:

- 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/M4 or DMA.
- TFT Direct Drive with support for masking and alpha blending.

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

4. 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 EFM32G-DK3550 and the EFM32GG-DK3750 development kits.

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-section describes relevant hardware information for the development kits.

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.

4.1 Development Kit Hardware Description

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, which 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.

The EFM32G-DK3550 and EFM32GG-DK3750 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 development kit is MT45W2MW16PGA-70 IT from Micron.

On the development kit, the address latch is a 16-bit 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 access 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 development kit. The status is indicated in the top right hand corner of the TFT display.

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

5. Further Reading and Examples

The EBI chapter in the EFM32 reference manuals include description of the different modes of the EBI, timing, and additional features, such as prefetch, NAND-flash, and TFT-Direct Drive mode.

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 Table 5.1 EBI Examples and Application Notes in Simplicity Studio on page 15 includes several application notes and examples which use the external bus interface in some way. The examples and their description are a good source for more information on this topic. Please note that the list is not exhaustive, because we update and release new examples continuously.

Table 5.1. EBI Examples and Application Notes in Simplicity Studio

Example/Application Note	Description	Required Hardware
norflash	Demonstrates use of the NOR Flash driver and executes code from NOR flash sector.	EFM32xG_DK3x50
scroller	Demonstrates TFT Direct Drive with external frame buffer in 4 MB PSRAM.	EFM32xG_DK3x50
tft and tftprintf	Demonstrates how to communicate with the external TFT controller over the EBI without direct drive (mode 8080).	EFM32G_DK3550 and EFM32xG_DK3x50
usbdcomposite and usbdmsd	Implements a Mass Storage Class device (MSD) with 4 MB external PSRAM or 16 MB external NOR Flash.	EFM32xG_DK3x50
nandflash	Demonstrates use of the NAND Flash driver.	EFM32xG_STK3x00
AN0047	Demonstrates both TFT Direct Drive and 8080-mode communication with the TFT display.	EFM32GG_DK3750
AN1011	Uses NAND Flash to store binary file for firmware upgrade.	EFM32GG_STK3700

Note:

Requires Gecko SDK Suite MCU to run the examples and application notes.

The EFM32xG_DK3x50 development kits include EFM32LG_DK3650, EFM32GG_DK3750, and EFM32WG_DK3850.

The EFM32xG_STK3x00 starter kits include EFM32LG_STK3600, EFM32GG_STK3700, and EFM32WG_STK3800.

The 4 MB PSRAM on EFM32xG_DK3x50 development kits is MT45W2MW16PGA-70 IT from Micron.

The 16 MB NOR Flash on EFM32xG_DK3x50 development kits is S29GL128P90FFIR13 from Spansion.

The 32 MB NAND Flash on EFM32GG_STK3700 starter kit is NAND256W3A2BZA6E from Numonyx.

6. Revision History

Revision 1.10

December, 2018

- Updated formatting.
- Updated content for EFM32 Series 1.
- Added Device Compatibility chapter 1. Device Compatibility.
- Added Memory Bank Setting chapter 3.2 Memory Bank Setting.
- Added Data and Address Bus chapter 3.3 Data and Address Bus.
- Added Pin Configuration chapter 3.7 Pin Configuration.
- Removed all information for obsolete EFM32G development kit.

Revision 1.09

May, 2014

- Updated example code to CMSIS 3.20.5.
- Changed to Silicon Labs license on code examples.
- Added project files for Simplicity IDE.
- Removed makefiles for Sourcery CodeBench Lite.

Revision 1.08

September, 2013

- New cover layout.

Revision 1.07

May, 2013

- Added software projects for ARM-GCC and Atollic TrueStudio.

Revision 1.06

January, 2013

- Added information about Giant Gecko EBI and restructured document.
- Modified software example to work with several development kit revisions.

Revision 1.05

November, 2012

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

Revision 1.04

April, 2012

- Adapted software projects to new peripheral library naming and CMSIS_V3.

Revision 1.03

March, 2012

- Fixed compilation error in CodeSourcery projects.

Revision 1.02

October, 2011

- Updated IDE project paths with new kits directory.

Revision 1.01

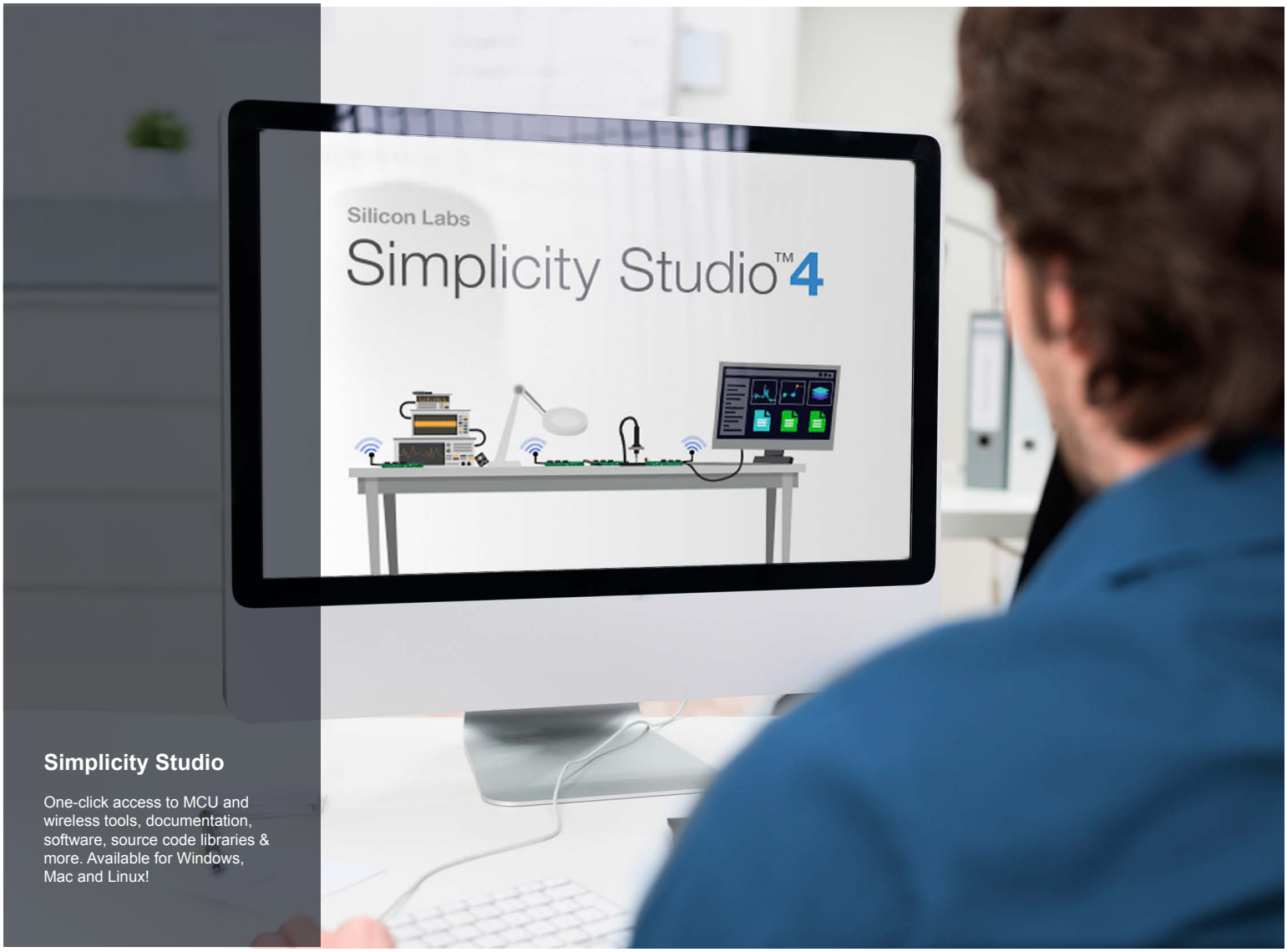
July, 2011

- Fixed errors in main.c and added line to wait for AEM state.
- Added note in document that AEM state must be waited for.

Revision 1.00

March, 2011

- Initial revision.



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com