EFM32®

... the world's most energy friendly microcontrollers

USER MANUAL

Development Kit EFM32WG-DK3850

The EFM32 Wonder Gecko Development Kit is a feature rich development platform for evaluation, prototyping and application development for the EFM32 Wonder Gecko MCU family with the ARM Cortex-M4F CPU core.

Main features:

- **Advanced Energy Monitoring provides real-time information about the energy consumption of an application or prototype design.**
- **Integrated emulator providing full debug and trace capability**
- **Exchangeable prototyping board for custom application circuit development**

ARM ZERO TINY GECKO LEOPARD GIANT WONDER

SILICON LABS

1 Introduction

1.1 Description

The EFM32WG-DK3850 is a highly flexible development platform demonstrating some of the EFM32 Wonder Gecko microcontroller's many capabilities. The rich feature set makes the kit an excellent platform for evaluating the microcontroller as well as a good starting point for application development.

The EFM32WG-DK3850 kit consists of three separate boards:

- 1 x BRD3201A EFM32 Development Kit Motherboard
- 1 x BRD3800A EFM32 Wonder Gecko MCU plugin board
- 1 x BRD3500B EXP32 prototyping board

The EFM32 microcontroller is mounted on the plugin board, which plugs into the Motherboard. All the EFM32 GPIO pins are available through headers on the prototyping board.

Additional kit contents:

- IAR Embedded Workbench ARM Kickstart version CD/DVD
- Atollic TrueSTUDIO for ARM CD
- USB cables

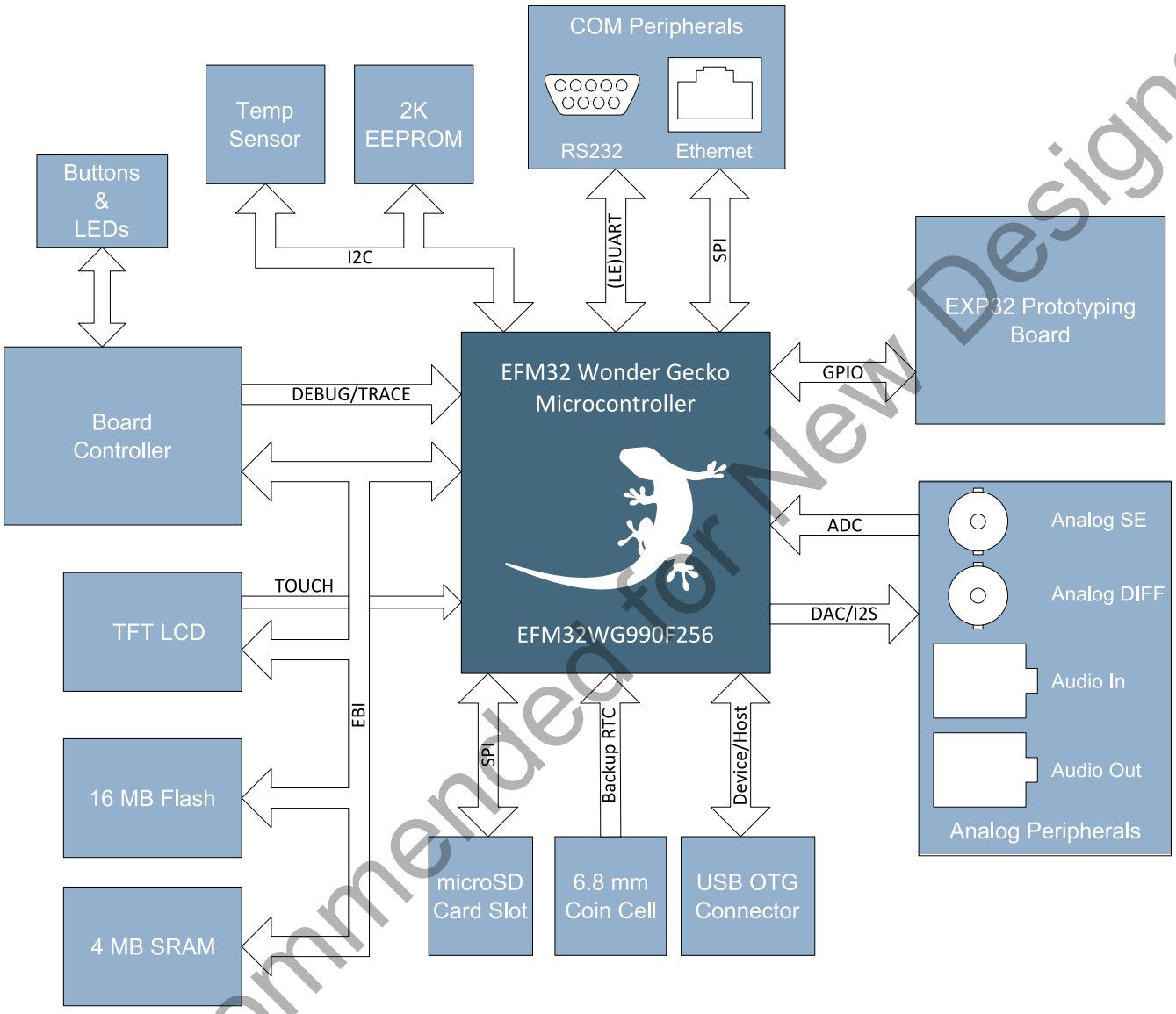
1.2 Features

- EFM32WG990F256 MCU with 256 KB Flash and 32 KB KB SRAM.
- Advanced Energy Monitoring system for precise current tracking.
- Special hardware configuration for isolation of the MCU power domain.
- Replaceable prototyping board for quick custom application development.
- Full feature USB debugger / emulator with trace support and debug out functionality.
- 3.5-inch TFT-LCD 320x240 pixel RGB color display with resistive touch film.
- Smart Board controller handles configuration and signal routing.
- Single ended and differential ADC inputs.
- Line-in stereo audio input amplifier.
- Line-out stereo audio output amplifier and I2S DAC.
- 1 x RS232 Serial Port (DSUB-9).
- 10/100 Mbps Ethernet MAC+PHY with SPI interface
- MicroSD card reader (SPI mode).
- 2Meg x 16 (4MB) PSRAM with 70ns access time.
- 8Meg x 16 (16MB) NOR Flash with 90ns access time.
- 2Kb I²C EEPROM.
- Temperature sensor with I2C interface.
- 5 way joystick.
- 4 User buttons, 4-bit DIP switch and 16 user LEDs.
- USB Micro-AB (OTG) connector
- 6.8mm coin cell holder for backup-RTC
- Crystals for LFXO and HFXO: 32.768kHz and 48.000MHz

2 Kit Block Diagram

An overview of the EFM32 Wonder Gecko Development Kit is shown in Figure 2.1

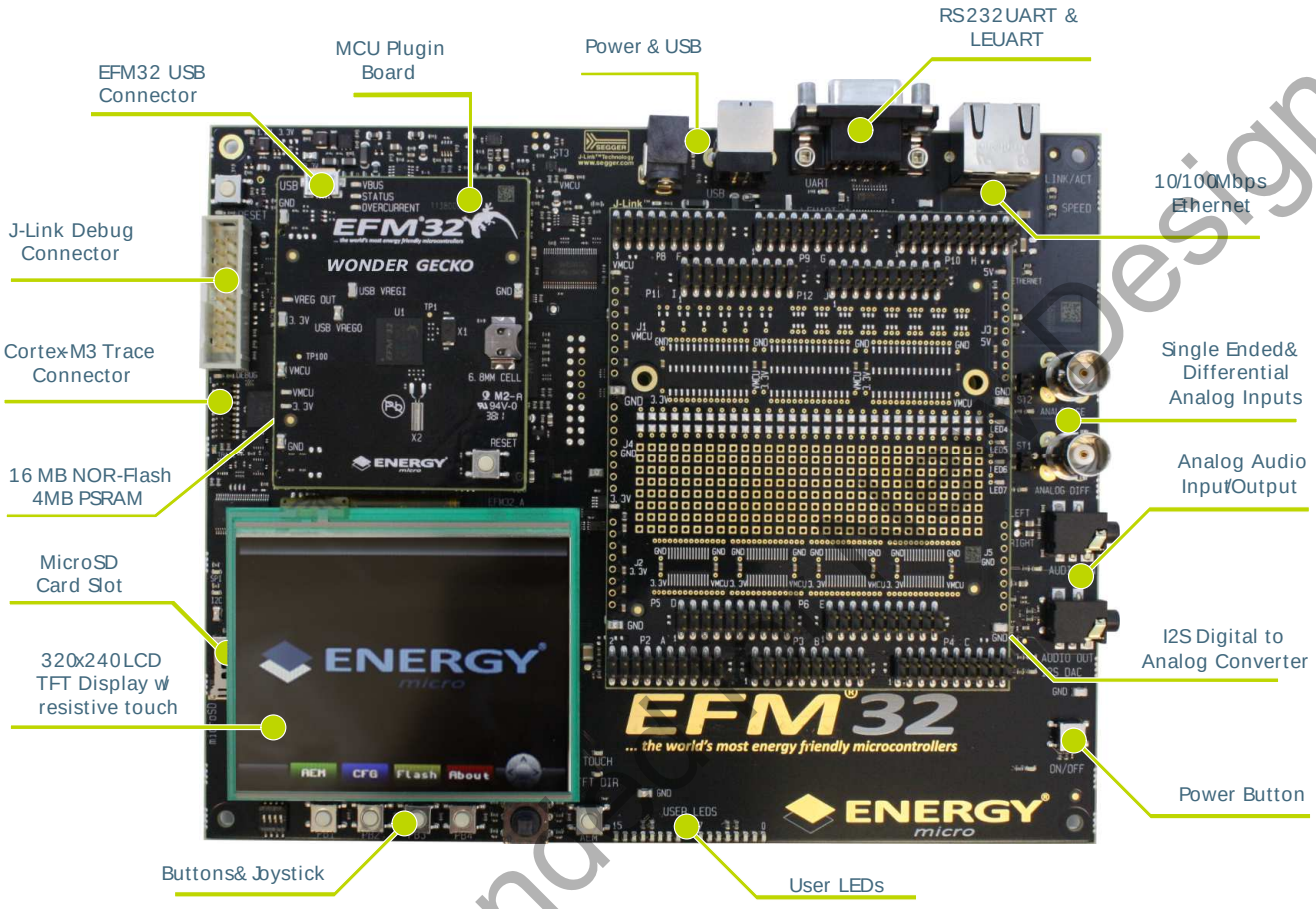
Figure 2.1. EFM32WG-DK3850 Block Diagram



3 Kit Hardware Layout

The layout of the EFM32 Wonder Gecko Development Kit is shown below.

Figure 3.1. EFM32WG-DK3850 hardware layout



4 Using the EFM32WG-DK3850

The EFM32 Wonder Gecko Development Kit is intended to be a complete platform for developing applications for the EFM32 microcontroller. The embedded debugger allows applications to be downloaded and debugged directly. A set of useful peripherals is provided, and custom hardware can be developed on the prototyping area, where all the microcontroller's IO pins are made available.

By default the peripherals on the board are *not* connected to the MCU. Interfacing the peripherals is done entirely without using jumpers, but instead through the kit's board controller. Two main approaches exist to configuring the board for an application: from within the application itself using the Board Support Packge, or by using the kit's user interface.

4.1 Board Support Package

The kit Board Support Package, or BSP, is provided to allow an application to control various aspects of the EFM32WG-DK3850 kit. Peripherals can be connected or disconnected with simple calls to the API. The user buttons and LEDs are also accessed through the BSP.

The easiest way to obtain the latest version of the BSP is through Simplicity Studio. It can also be downloaded at: http://www.energymicro.com/downloads/software.

The BSP can be configured to use two different methods of communication toward the board controller: *SPI mode* or *EBI mode*. In SPI mode the EFM32 communicates with the board controller using a simple 4-wire SPI bus, and in EBI mode the board controller becomes a memory-mapped peripheral in the EFM32's address space using the EFM32's External Bus Interface module.

SPI mode uses fewer pins to communicate with the board controller, but the external memory devices and TFT-display are not available in this mode. Use this mode when the IO taken up by the EBI are needed for other purposes. To enable the board controller in SPI-mode use the BSP function from within the application code:

```
BSP_Init ( BSP_INIT_DK_SPI )
```

EBI mode is the preferred mode of interfacing to the board controller. This gives access to all the board functions as well as the external memory devices (PSRAM and Flash) and the TFT-LCD display. To enable the board controller in EBI-mode use the BSP function from within the application code:

```
BSP_Init ( BSP_INIT_DK_EBI )
```

In order to configure the BSP, some dedicated GPIO pins are used. These pins are listed in Table 4.1, and are normally controlled by the BSP. No manual configuration of these pins are necessary.

Table 4.1. GPIO's used for BSP functions

MCU Pin	Description
PB15	Board controller configuration line 1.
PD13	Board controller configuration line 2.
PE0	Interrupt request from board controller.

Once the BSP has been configured, the different peripherals and kit functions can be accessed through the BSP API. Please refer to Chapter 6 for detailed information about the different kit peripherals and how to access them with the BSP. It can also be useful to take a look at some of the software examples found in Simplicity Studio.

Note

Full documentation and source code for the BSP can be found in Simplicity Studio.

4.2 User Interface

In addition to using the API provided by the BSP, the kit can also be configured through the graphical user interface consisting of the TFT-LCD display together with the buttons PB1 to PB4 and the 5-way joystick located below. The board controller provides a simple menu system, allowing most aspects of the kit to be configured directly.

The user is encouraged to explore the menu system and the different functions provided. Some useful functions that can be performed using the menu system are:

- Enabling or disabling access for individual peripherals.
- Displaying information about the different boards on the kit.
- Getting and displaying information about the EFM32 MCU part mounted on the MCU board
- Displaying real-time current consumption of the EFM32 MCU.
- Uploading and running example applications stored in the kit.
- Adjusting the MCU voltage (VMCU).
- Selecting the debugging mode (IN/OUT/MCU/OFF)

Since the TFT display and keys are shared between the board controller and the EFM32, a separate button labeled "AEM" is present to switch control of the buttons and display. By default, when the kit has been started up, control is given to the board controller, and pressing the buttons interracts with the graphical user interface. Pressing the "AEM" button once switches control over to the EFM32, and pressing it again switches control back. The current state is shown in the top right corner of the display: "KEYS:AEM" means that the board controller has control, and "KEYS:EFM" indicates that the EFM32 has control.

4.3 Simplicity Studio

The first step in getting started with the EFM32 Wonder Gecko Development Kit is to download Simplicity Studio from: http://www.energymicro.com/simplicity

The Simplicity Studio software package contains tools, software examples and documentation relevant to developing applications with the development kit. Some important tools included in Simplicity Studio are:

- *energyAware Commander*
- *energyAware Profiler*

The *energyAware Commander* is a tool for updating the kit's firmware, programming the MCU and launching demos.

The *energyAware Profiler* is the PC-side interface to the Advanced Energy Monitor. It provides the possibility to do energy-debugging and profiling of application code.

5 Power and Reset

5.1 USB Power

The EFM32WG-DK3850 can get its power from the standard USB 2.0 Type B port located on the motherboard. The USB hub the kit is connected to needs to be able to deliver 500 mA (5 unit loads).

5.2 External Power Supply

By using the DC jack plug located on the motherboard, the EFM32WG-DK3850 can be powered by an external power supply. The voltage must be 5 V and the supply must be able to deliver 500 mA. This is mainly intended as a supplement to the USB power, for example when a custom circuit on the prototyping board needs more power.

The power jack dimensions should be a standard 5.5 mm outer diameter and 2.1 mm inner diameter. The tip is 5V and the sleeve is GND.

5.3 ON/OFF Button

A power button is situated on the lower right corner of the motherboard. Press once to turn on the kit, and press once again to turn off.

5.4 MCU Reset

The primary user reset for the MCU is the reset button on the MCU board. This will only reset the MCU. The MCU can also be reset by an emulator, either by the on-board debugger, or an externally connected emulator.

5.5 Board Controller Reset

The board controller can be reset by pushing the reset button on the main board.

6 Peripherals

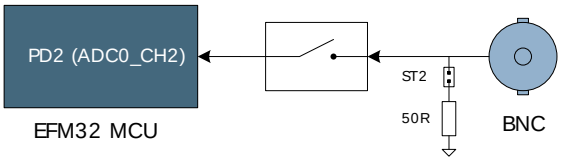
The peripherals on the EFM32 Wonder Gecko Development Kit are all isolated from the EFM32's IO pins by default. Peripherals are isolated to prevent excess current leakage into unused peripherals. The different peripherals can be connected using simple functions in the kit's board control software package.

This chapter describes the different peripherals that can be connected to the EFM32, together with the BSP functions required to do this. Before any of the described functions can be called, the board must first be configured in either EBI or SPI mode, as described above.

6.1 Single-ended Analog Input

A BNC connector is available for directly connecting an analog signal to the ADC of the EFM32. The input can also be used for digital I/O. If required, 50 ohm termination can be added by soldering in a jumper, ST2.

Figure 6.1. ANALOG SE



The single-ended analog input can be connected by calling:

```
BSP_PeripheralAccess ( BSP_ANALOG_SE, true )
```

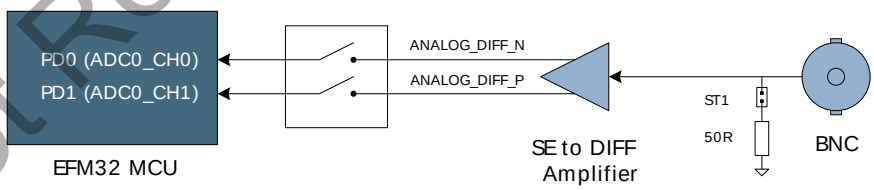
Note

The pin PD2 is shared between the Analog SE, the I2S DAC, and the Ethernet Controller peripherals. As a consequence, these kit features cannot be used simultaneously.

6.2 Differential Analog Input

The ANALOG DIFF input consists of a BNC connector and a differential operational amplifier with ground as reference. The op-amp output common mode voltage is 1.65V, and also implements a low-pass active filter with a cut-off frequency of 4MHz.

Figure 6.2. ANALOG Diff



This peripheral can be connected by calling:

```
BSP_PeripheralAccess ( BSP_ANALOG_DIFF, true )
```

Note

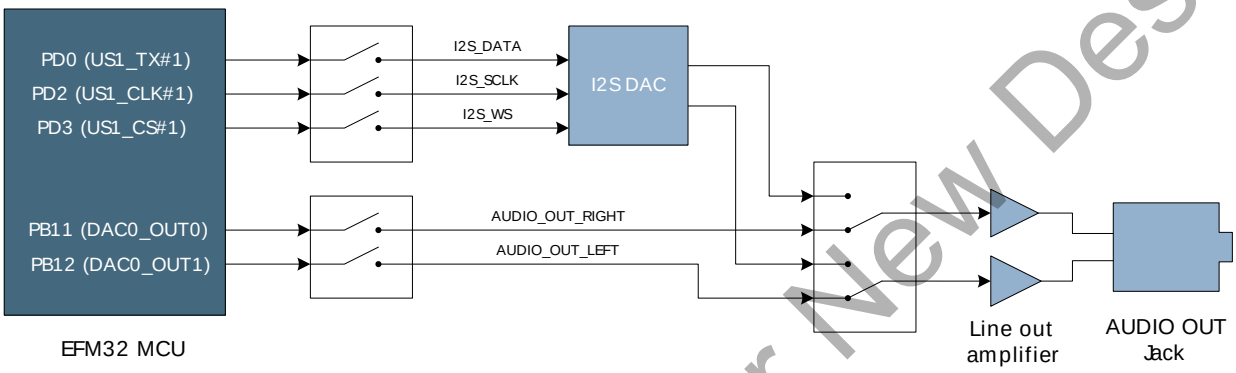
The pins PD0 and PD1 are shared between the Analog Diff, the I2S DAC, the Ethernet Controller. As a consequence, these kit features cannot be used simultaneously.

6.3 Audio Out

The kit contains an audio output amplifier with filter connected to a 3.5 mm jack. The gain of the amplifier is fixed to 6 dB and is referenced to ground. The filter is a 3-pole linear phase MFB filter with a cutoff frequency (at -3 dB) of 27 kHz. Two possibilities exist to drive the audio output amplifier:

- Using the internal DAC of the EFM32
- Using the external I2S DAC on the motherboard

Figure 6.3. Audio Out Block Diagram



As shown in the block diagram above, a multiplexer is used to select between the two possible audio sources. The multiplexer and isolation switches are controlled by the board controller, and can be enabled by calling the appropriate function in the BSP API:

- The audio output amplifier can be connected to the EFM32's internal DAC by calling

```
BSP_PeripheralAccess ( BSP_AUDIO_OUT, true )
```

- The audio output amplifier can be connected to I2S DAC by calling

```
BSP_PeripheralAccess ( BSP_I2S, true )
```

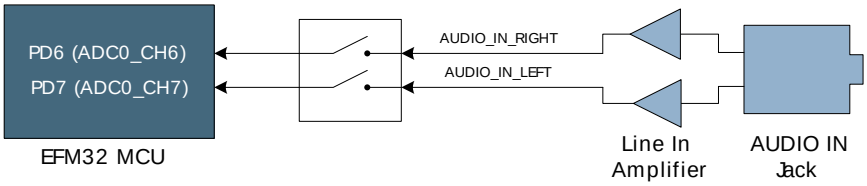
Note

The pins PD0, PD2 and PD3 are shared between the I2S DAC, Analog Input and Ethernet Controller peripherals. As a consequence, these kit features cannot be used simultaneously.

6.4 Audio In

An audio input amplifier with filter is present, and can be connected to the ADC of the EFM32. The gain of the amplifier is 0 dB and the bias point is 1.65 V. The filter is a 3-pole linear phase MFB filter with a cutoff frequency of 20 kHz. In addition to the input amplifier and filter, the line in is equipped with a voltage divider resulting in 6 dB attenuation.

Figure 6.4. Audio In Block Diagram



The line in amplifier can be connected directly to the EFM32 by calling

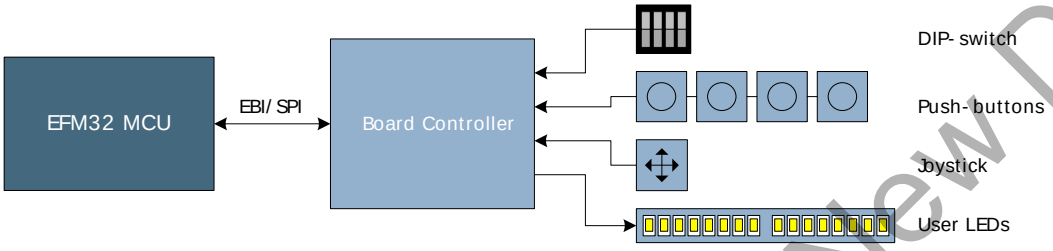
```
BSP_PeripheralAccess ( BSP_AUDIO_IN, true )
```

6.5 User Interface Peripherals

A set of buttons and LEDs are provided as a simple way of interfacing to applications. These peripherals include:

- A 4-way DIP Switch
- 4 Push-Buttons
- A 5-way Joystick
- 16 User LEDs

Figure 6.5. User interface



The buttons and LEDs are not directly connected to the MCU, instead the board controller is used to read button states and set the LED outputs. This can be done with a set of BSP API functions

- `uint16_t BSP_PushButtonsGet ( void )`
- `uint16_t BSP_JoystickGet ( void )`
- `uint32_t BSP_DipSwitchGet ( void )`
- `void BSP_LedsSet ( uint32_t leds )`
- `BSP_LedSet ( int ledNo ) / void BSP_LedClear ( int ledNo )`
- `uint32_t BSP_LedsGet ( void )`
- `int BSP_LedGet ( int ledNo )`

The various buttons on the kit can also be configured to generate an interrupt to the EFM32 when their state changes. Please refer to Section 6.14 for information on how to enable interrupts for these peripherals.

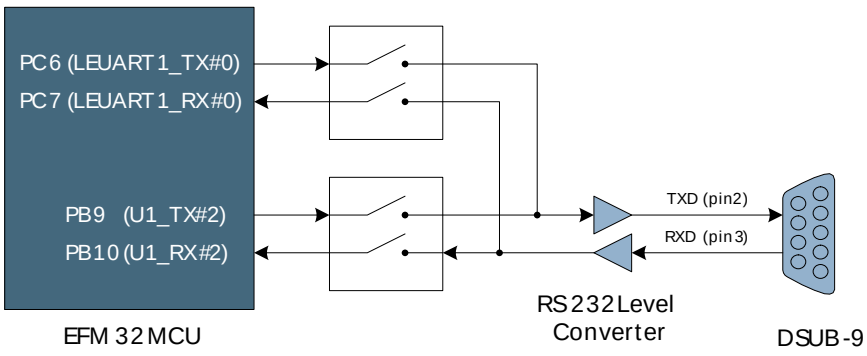
Note

The push-buttons are also used to control the Advanced Energy Monitor (AEM) application. A separate button, labeled "AEM" is used to switch the role of the push-buttons.

6.6 RS232

An RS232 level converter together with a DSUB-9 connector is provided for serial communication between the EFM32 and an external device. The pinout is such that the kit is the DCE (Data Circuit-terminating Equipment). Hardware flow-control signals are not used.

Figure 6.6. RS232



The RS232 peripheral can be connected either to a standard UART peripheral, or to the Low Energy UART (LEUART) of the EFM32.

- The audio output amplifier can be connected to the EFM32's UART peripheral by calling

```
BSP_PeripheralAccess ( BSP_RS232_UART, true )
```

- The audio output amplifier can be connected to the EFM32's LEUART peripheral by calling

```
BSP_PeripheralAccess ( BSP_RS232_LEUART, true )
```

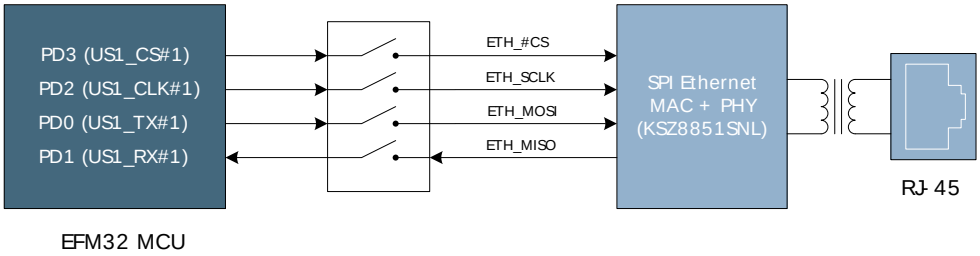
The RS232 transceiver can also be shut down to prevent excess current leakage when the UART or LEUART is not in use, without disconnecting the switches. This can be done with:

```
BSP_PeripheralAccess ( BSP_RS232_SHUTDOWN, true )
```

6.7 Ethernet

The kit contains a single-chip Fast Ethernet controller consisting of a 10/100 physical layer transceiver (PHY), a MAC and an SPI interface. Also present are the required magnetics and RJ-45 connector to provide network connectivity to an application.

Figure 6.7. SPI Ethernet MAC+PHY



The Ethernet controller has 12KB buffer memory on the receive queue and 6KB on the transmit queue, and supports Auto-MDIX. Two LEDs are placed next to the RJ-45 connector to indicate link speed and activity.

The Ethernet interface can be enabled and connected to the EFM32 by calling

```
BSP_PeripheralAccess ( BSP_ETH, true )
```

The Ethernet controller also has an interrupt pin which can be routed through the board controller to the EFM32. Please refer to Section 6.14 for details on how to enable the Ethernet controller interrupt.

Note

The pins PD0 to PD3 are shared between the Ethernet Controller, the I2S DAC and the Analog Input peripherals. As a consequence, these kit features cannot be used simultaneously.

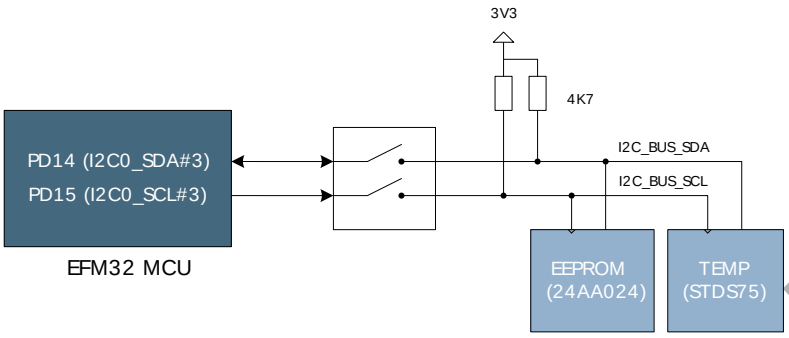
6.8 I²C EEPROM and Temperature Sensor

Two devices are attached to an I²C bus which can be connected to the EFM32. These devices are:

- Temperature Sensor
- 2Kb EEPROM

Both devices support a maximum I²C bus speed of 400 kHz.

Figure 6.8. I2C Bus



The EEPROM device consists of 256 bytes (256 x 8) and has a lifetime of 1,000,000 erase/write cycles. The EEPROM's I²C address is 0xA0.

The temperature sensor can measure temperatures from -55 to +125°C, with selectable resolution between 9 and 12 bits. The temperature sensor's I²C address is 0x90.

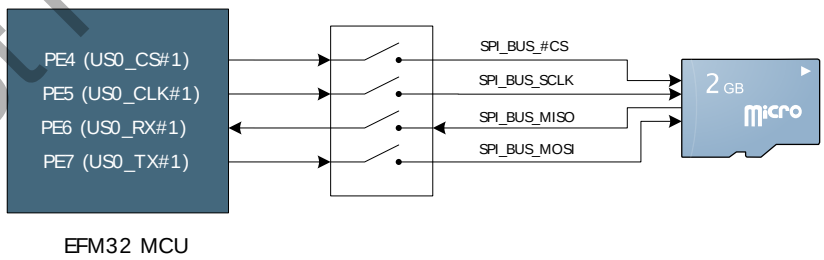
The I²C bus can be connected to the EFM32 with the BSP function:

```
BSP_PeripheralAccess ( BSP_I2C, true )
```

6.9 microSD

A microSD card can be connected to the EFM32 through the serial peripheral bus. The card slot is situated under the LCD display. This allows for applications with large storage and/or file system requirements.

Figure 6.9. microSD



The microSD card slot can be connected to the EFM32 with the BSP function:

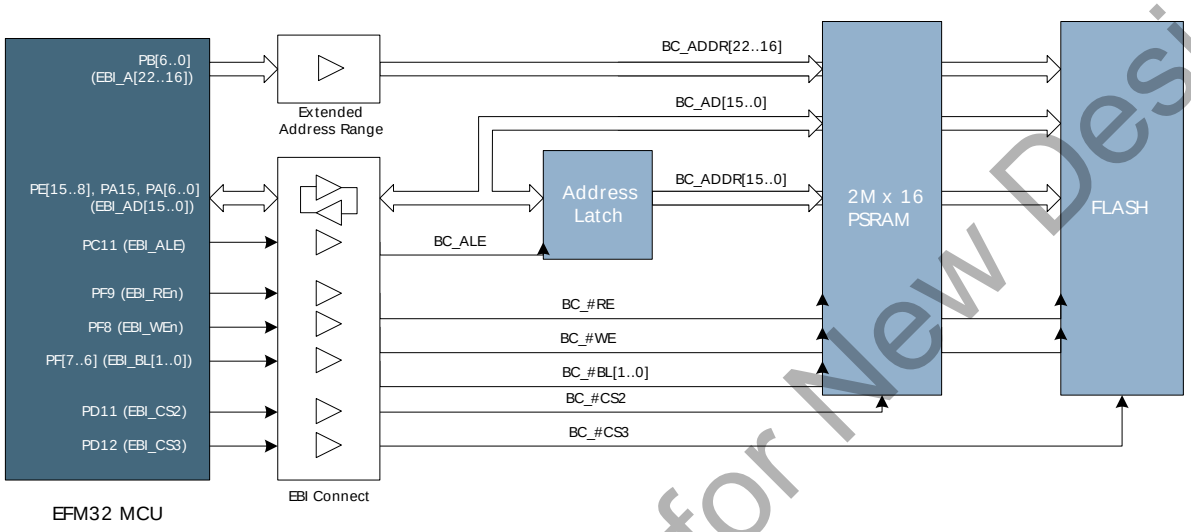
```
BSP_PeripheralAccess ( BSP_MICROSD, true )
```

6.10 Flash and PSRAM

Two memory devices are available through the EFM32's external bus interface:

- A 4MB (2M x 16) PSRAM
- A 16MB (8M x 16) NOR Flash

Figure 6.10. EBI peripherals



As shown in figure Figure 6.10, the PSRAM and FLASH devices are selected by the EBI_CS2 and EBI_CS3 signals, respectively. The PSRAM and Flash devices map to the EFM32's address space as following:

- PSRAM: 0x88000000 to 0x883FFFFF
- Flash: 0x8C000000 to 0x8CFFFFFF

The EBI is automatically configured by the BSP for all external memory devices when the board is configured in EBI mode with:

```
BSP_Init ( BSP_INIT_DK_EBI )
```

By default, extended addressing mode is enabled, allowing access to the full capacity of the external memory devices. This consumes 7 IO pins (PB0 to PB6) in addition to the other EBI pins. If desired, these pins can be freed up and used for other purposes by disabling extended addressing mode in the EBI, and calling the BSP function:

```
BSP_EbiExtendedAddressRange ( False )
```

Note

With extended addressing mode *disabled*, only 128 KB of PSRAM and only 128 KB of flash is available.

6.11 TFT-LCD Display

The EFM32 Wonder Gecko Development Kit contains a 320x240 pixel TFT-LCD display, which is used both as a graphical user interface toward the kit itself, as well as a possible output device for the EFM32 MCU. The "AEM" button is used to switch control of the TFT-LCD display between the board controller and the EFM32 MCU.

Two different methods exist to drive the display:

- As a memory mapped peripheral using the display's built in SSD2119 controller
- Using the TFT direct drive mode of the EFM32

In both cases the data is sent as 16-bit RGB data,

6.11.1 TFT Address Mapped Mode

In address mapped mode, the memory of the integrated SSD2119 controller is used to hold display data. The peripheral is mapped in the EFM32's address space from address 0x84000000 to 0x87FFFFFF.

Please refer to the "TFT" software example on how to set up and use the TFT-LCD in this mode.

6.11.2 TFT Direct Drive Mode

In TFT direct drive mode, the EBI peripheral of the EFM32 is used together with the external PSRAM to drive the TFT-LCD. Data is placed in the PSRAM, and clocked directly into the display using dedicated lines.

Table 6.1. Additional GPIOs used for TFT Direct Drive

MCU Pin	Signal Name	Description
PA8	EBI_DCLK	Display Dotclock
PA9	EBI_DTEN	Display Enable
PA10	EBI_VSYNC	Display Vertical Synchronization
PA11	EBI_HSYNC	Display Horizontal Synchronization

6.12 Resistive Touch Screen

The TFT-LCD display is covered by a resistive touch panel, which is connected to some ADC pins of the EFM32 microcontroller.

Figure 6.11. Resistive Touch Film

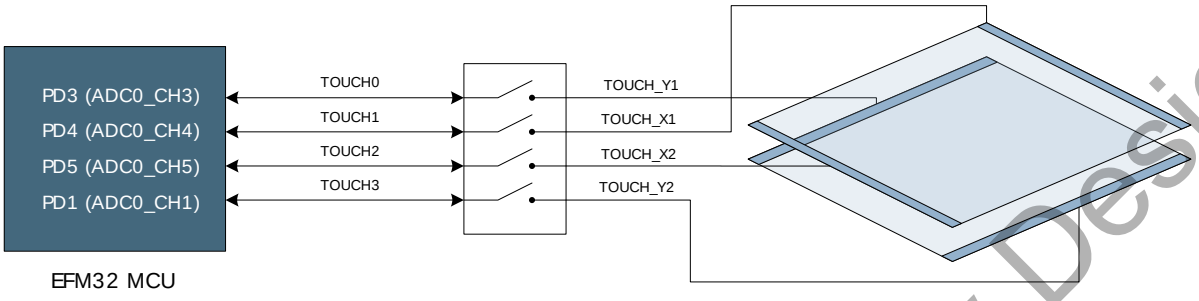


Figure shows how the resistive touch film is connected. When touched, the X-position can be read out by applying a voltage between the X1 and X2 electrodes and measuring the voltage on the Y1 or Y2 electrodes. Likewise, the Y-position can be read out by applying a voltage accross the Y1 and Y2 electrodes and measuring the X1 or X2 electrodes.

The resistive touch screen can be accessed with the BSP command:

```
BSP_PeripheralAccess ( BSP_TOUCH, true )
```

Note

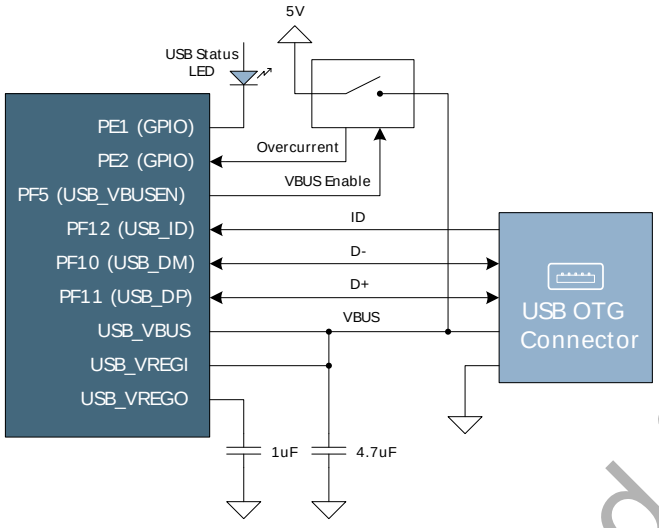
The pins PD1 and PD3 are shared between the Resistive Touch, the Ethernet Controller, the I2S DAC and the Analog Diff peripherals. As a consequence, these peripherals cannot be used simultaneously.

6.13 USB Micro-AB Connector

The MCU plugin board is equipped with a USB Micro-AB connector supporting USB On-The-Go. The figure below shows how the USB lines are connected to the EFM32.

The USB_VBUSEN line is connected to a current limited switch which supplies the VBUS line with 5 V when operating as a USB Host. The current limited switch also has a flag signal connected to the EFM32 which can notify it in case excessive current is drawn by the attached device. The current limit of the switch is set at 0.8 A.

Figure 6.12. EFM32 USB



6.14 Peripheral Interrupts

Some of the peripherals on the development kit can generate interrupts. The interrupts from these peripherals are routed through the board controller, which in turn signals pin PE0 on the EFM32 MCU to indicate that an interrupt has occurred. In order for the board controller to signal interrupts, the interrupts must first be enabled. The BSP provides functions for enabling and disabling interrupts:

- `int BSP_InterruptEnable ( uint16_t flags )`
- `int BSP_InterruptDisable ( uint16_t flags )`

When a GPIO interrupt occurs, and the interrupt is caused by a falling edge of PE0, the interrupt flag register in the board controller should be read to determine which peripheral caused the interrupt. The flag should also be cleared after processing the interrupt. This can be done with the functions:

- `uint16_t BSP_InterruptFlagsGet ( void )`
- `int BSP_InterruptFlagsClear ( uint16_t flags )`

The parameter *flags* is indicates which bits in the corresponding interrupt enable or flag registers should be set or cleared. This parameter should be a combination of the bit masks shown in Table 6.2.

Table 6.2. Interrupt sources

Number	Interrupt Source	Interrupt Enable Mask	Interrupt Flag Mask
0	Push Buttons	BC_INTEN_PB	BC_INTFLAG_PB
1	Dip Switch	BC_INTEN_DIP	BC_INTFLAG_DIP

Number	Interrupt Source	Interrupt Enable Mask	Interrupt Flag Mask
2	Joystick	BC_INTEN_JOYSTICK	BC_INTFLAG_JOYSTICK
3	AEM Button	BC_INTEN_AEM	BC_INTFLAG_AEM
4	Ethernet Controller	BC_INTEN_ETH	BC_INTFLAG_ETH

Example 6.1. Interrupt enable example

For example, to enable interrupts from both the push buttons and the joystick:

```
/* Disable all BSP interrupts */
BSP_InterruptDisable ( 0xffff );

/* Clear all interrupt flags */
BSP_InterruptClear ( 0xffff );

/* Enable interrupts in the BSP */
BSP_InterruptEnable ( BC_INTEN_PB | BC_INTEN_JOYSTICK );
```

In addition to enabling the interrupts in the BSP, the EFM32 must also be configured to allow interrupts from pin PE0:

```
/* Configure interrupt pin as input with pull-up */
GPIO_PinModeSet ( gpioPortE, 0, gpioModeInputPull, 1 );

/* Set falling edge interrupt and clear/enable it */
GPIO_IntConfig ( gpioPortE, 0, false, true, true );

/* Enable even GPIO interrupts */
NVIC_ClearPendingIRQ(GPIO_EVEN_IRQn);
NVIC_EnableIRQ(GPIO_EVEN_IRQn);
```

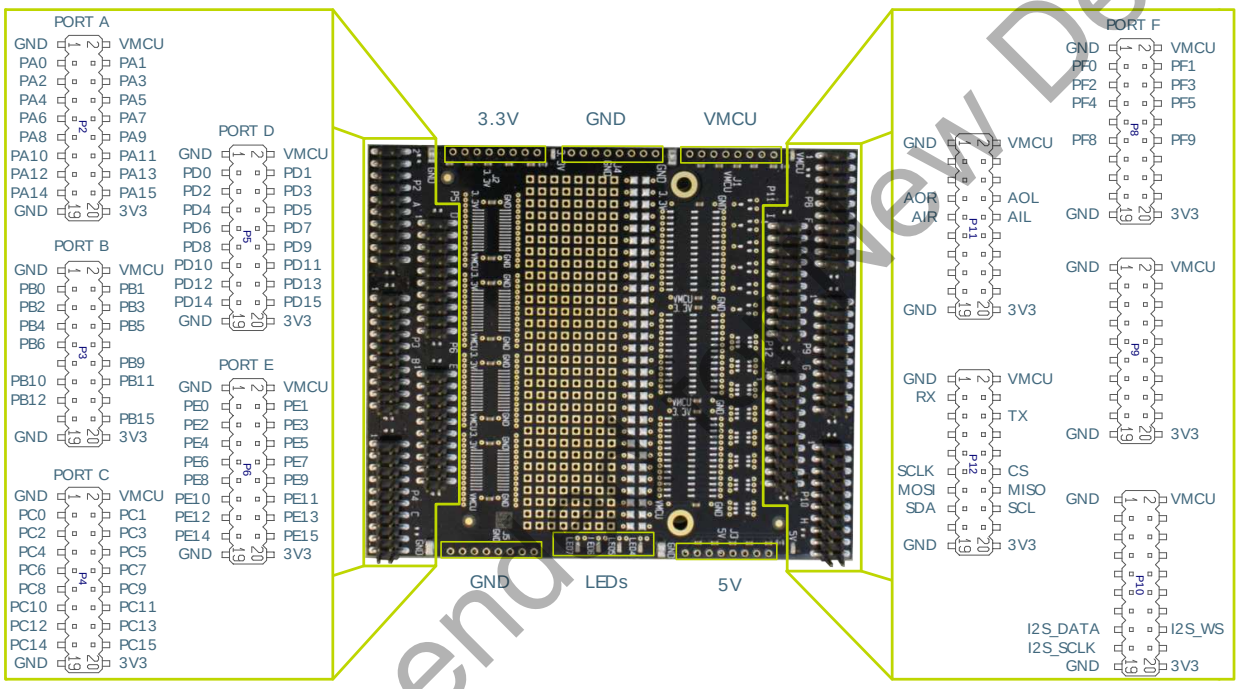
7 Prototyping Board

7.1 Description

The Prototyping Board is a plugin board that contains a large area for constructing custom circuits. It contains a "veroboard" area and many unpopulated footprints which can be used for different SMT parts. Each TSSOP and SSOP site has decoupling capacitors close by.

All the EFM32 GPIO pins are made available on pin headers. Figure 7.1 is an illustration which shows how the MCU GPIO pins are mapped to the Prototyping Board.

Figure 7.1. Prototyping Board



Additionally, the Prototyping Board also contains connection points for different voltages: 3.3V, 5V, GND, and VMCU. Any current drawn from the VMCU pins will also be measurable with the Advanced Energy Monitor, allowing the whole circuit to be evaluated.

7.2 Dedicated Signals

In order to ensure the best possible signal integrity during certain modes of operation, some signals become disconnected from the Prototyping Board in these modes. These modes of operation are:

- When the BSP is configured in EBI mode, all the EBI pins are disconnected from the protoboard.
- When Trace is enabled, all the high speed Trace signals are routed directly to the board controller, and are not available on the Prototyping Board.

Table 7.1 summarizes the different signals that become unavailable on the Prototyping Board during certain operating modes.

Table 7.1. GPIO pins made unavailable in certain operating modes

MCU pins	Signal name	Unavailable in mode
PA[6..0]	EBI_AD[15..9]	EBI

MCU pins	Signal name	Unavailable in mode
PA8	EBI_DCLK	EBI
PA9	EBI_DTEN	EBI
PA10	EBI_VSYNC	EBI
PA11	EBI_HSYNC	EBI
PA15	EBI_AD8	EBI
PE[15..8]	EBI_AD[7..0]	EBI
PC11	EBI_ALE	EBI
PF8	EBI_WEn	EBI
PF9	EBI_REn	EBI
PF[7..6]	EBI_BL[1..0]	EBI
PD[12..9]	EBI_CS[3..0]	EBI
PD[6..3]	DBG_TD[3..0]	Trace
PD7	DBG_TCLK	Trace

In the default configuration, pins PB7, PB8, PB13 and PB14 are used for the LFXTAL and HFXTAL, and are by default not available on the Prototyping Board. It is however possible to make them available if necessary by modifying the MCU plugin board. Please refer to the BRD3600A schematics for more details.

7.3 Peripheral Signals

In addition to the mapping of MCU pins, some of the kit's peripherals are also mapped directly to the Prototyping Board. In Figure 7.1 the pins labeled "Xn" are extra peripheral functions. Note that these pins are connected to the peripherals "after" the isolation switches, so calls to the BSP are not necesarry to enable/connect them. The table below shows which peripheral function signals are mapped to which pins on the Prototyping board

This can be useful when a peripheral cannot be used normally because the required pins on the EFM32 are already used for another purpose. Custom connections between EFM32 pins and some kit peripherals can then be made on the Prototyping Board.

Table 7.2. Peripheral functions mapped directly to the Prototyping Board

Prototyping Board pin	Signal Name	Description
P11.7	AUDIO_OUT_RIGHT	Audio out right channel (before audio out mux)
P11.8	AUDIO_OUT_LEFT	Audio out left channel (before audio out mux)
P11.9	AUDIO_IN_RIGHT	Audio in right channel
P11.10	AUDIO_IN_LEFT	Audio in left channel
P12.3	IF_RS232_RX	RS232 receive signal
P12.6	IF_RS232_TX	RS232 transmit signal
P12.11	SPI_BUS_SCLK	MicroSD serial clock
P12.12	SPI_BUS_#CS	MicroSD chip select

Prototyping Board pin	Signal Name	Description
P12.13	SPI_BUS_MOSI	MicroSD data in
P12.14	SPI_BUS_MISO	MicroSD data out
P12.15	I2C_BUS_SDA	I²C EEPROM and Temperature sensor serial data
P12.16	I2C_BUS_SCL	I²C EEPROM and Temperature sensor serial clock
P10.15	I2S_DATA	I2S DAC serial data
P10.16	I2S_WS	I2S DAC word select
P10.17	I2S_SCLK	I2S DAC serial clock

8 Advanced Energy Monitor

8.1 Usage

The AEM (Advanced Energy Monitor) data is collected by the board controller and can be displayed by the energyAware Profiler, available through Simplicity Studio. By using the energyAware Profiler, current consumption and voltage can be measured and linked to the actual code running on the EFM32 in real time.

The current consumption data can also be viewed directly on the TFT-LCD display of the kit, by selecting the "AEM" menu function. The scale is logarithmic, and the time scale of the graph can be adjusted (AEM > CFG > Graph x scale).

8.2 AEM theory of operation

In order to be able to measure currents ranging from 100 nA to 50 mA (114 dB dynamic range), two current sense amplifiers are utilized. The amplifiers measure voltage drop over a small series resistor and translates this into a current. Each amplifier is adjusted for current measurement in a specific range. The ranges for the amplifiers overlap and a change between the two occurs when the current is 200 uA. To reduce noise, averaging of the samples is performed before the current measurement is presented in the AEM GUI.

During start-up of the kit, and when VMCU is changed, an automatic calibration of the AEM is performed. This calibration compensates for the offset error in the sense amplifiers.

8.3 AEM accuracy and performance

The Advanced Energy Monitor is capable of measuring currents in the range of 100 nA to 50 mA. For currents above 200 uA, the AEM is accurate within 100 uA. When measuring currents below 200 uA, the accuracy increases to 1 uA. Even though the absolute accuracy is 1 uA in the sub 200 uA range, the AEM is able to detect changes in the current consumption as small as 100 nA The measurement bandwidth of the AEM is 60 Hz when measuring currents below 200 uA and 120 Hz when measuring currents above 200 uA. The table below summarizes the accuracy of the two current sense amplifiers in different ranges.

Table 8.1. AEM accuracy

Current range	Low gain amplifier accuracy	High gain amplifier accuracy
50 mA	0.1 mA	-
1 mA	0.1 mA	-
200 uA	0.01 mA	1 uA
10 uA	-	0.1 uA
1 uA	-	0.1 uA

Note

In order for the AEM to work correctly, VMCU should be 3.0V or higher.

9 Debugging

The EFM32 Wonder Gecko Development Kit contains a built-in J-Trace for Cortex-M3 from Segger. It is a fully functional debugger capable of both serial wire debugging and trace (ETM). The embedded debugger can also be used to download flash and debug external targets. In addition to the internal debugger, using an external debugger is also supported.

9.1 Debug Modes

The different debug modes are referred to as Debug IN, Debug OUT, Debug MCU and Debug OFF, and are summarized in Table 9.1. Switching between the different debugging modes can either be done with the User Interface (CFG > Debug Control), or through the *energyAware Commander* tool.

Table 9.1. Debug modes

Mode	Description
Debug MCU	In this mode the built-in debugger is connected to EFM32 on the MCU plugin board. The debug connector on the kit is not used.
Debug IN	In this mode the built-in debugger is disconnected, and an external debugger can be connected to debug the EFM32 on the MCU plugin board.
Debug OUT	In this mode the EFM32 on the MCU plugin board is disconnected, and the built-in debugger can be used to debug an EFM32 in an external application.
Debug OFF	In this mode both the debug connector and the built-in debugger is disconnected.

9.2 Trace

Additional debugging modes are provided for Trace functionality. The Trace modes are similar to the Debug modes, but have Trace enabled as well as SWD.

Table 9.2. Trace modes

Mode	Description
Trace MCU	In this mode the built-in J-Trace is connected to EFM32 on the MCU plugin board, and Trace is enabled. The debug connector on the kit is not used.
Trace IN	In this mode the built-in debugger is disconnected, and an external Trace emulator can be connected to debug the EFM32 on the MCU plugin board.
Trace OUT	In this mode the EFM32 on the MCU plugin board is disconnected, and the built-in J-Trace can be used to run Trace on an EFM32 in an external application.

9.3 Debug Connectors

9.3.1 J-Link Debug Connector

This connector is situated on the top left side of the kit, and is used for Debug IN and Debug OUT. The pinout is described in Table 9.3

Figure 9.1. Debug Connector

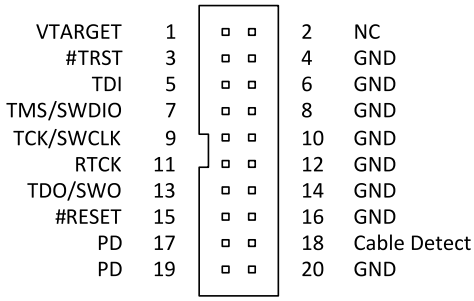


Table 9.3. Debug connector pinout

Pin number	Function	Note
1	VTARGET	Target voltage on the debugged application.
2	NC	Not Connected
3	#TRST	JTAG test reset
5	TDI	JTAG data in
7	TMS/SWDIO	JTAG TMS or Serial Wire data I/O
9	TCK/SWCLK	JTAG TCK or Serial Wire clock
11	RTCK	JTAG RTCK
13	TDO/SWO	JTAG TDO or Serial Wire Output
15	#RESET	Target MCU reset
17	PD	This pin has a 100k pulldown.
18	Cable detect	This signal must be pulled to ground by the external debugger or application for cable insertion detection.
19	PD	This pin has a 100k pulldown.
4, 6, 8, 10, 12, 14, 16, 20	GND	

9.3.2 Trace Connector

The Trace Connector is situated on the left side of the kit, below the Debug Connector. It is used for the "Trace IN" and "Trace OUT" debug modes. The pinout is described in Table 9.4

Figure 9.2. Debug Connector

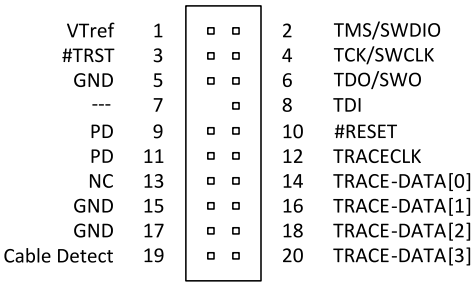


Table 9.4. Trace header pinout

Pin number	Function	Note
1	VTref	Target reference voltage.
2	TMS/SWDIO	Serial Wire Data Input/Output
3	#TRST	JTAG test reset
4	TCK/SWCLK	JTAG TCK / Serial Wire Clock
6	TDO/SWO	JTAG TDO / Serial Wire Output
8	TDI	JTAG TDI
10	#RESET	Target MCU reset
12	TRACECLK	Trace Clock
14	TRACE-DATA[0]	Trace Data pin 0.
16	TRACE-DATA[1]	Trace Data pin 1.
18	TRACE-DATA[2]	Trace Data pin 2.
20	TRACE-DATA[3]	Trace Data pin 3.
7, 9	NC	Not Connected
11, 13	PD	These pins have a 100k pull-down.
19	Cable Detect	This signal must be pulled low externally for the kit to detect cable insertion.
3, 5, 15, 17, 19	GND	

10 Integrated Development Environments

The Energy Micro software packages contains various examples in source form to use with the Starter Kit. The following IDEs are supported.

10.1 IAR Embedded Workbench for ARM

An evaluation version of IAR Embedded Workbench for ARM is included on a CD in the EFM32WG-DK3850 package. Check the quick start guide for where to find updates, and IAR's own documentation on how to use it. You will find the IAR project file in the

```
iar
```

subfolder of each project

10.2 Atollic TrueSTUDIO for ARM

An evaluation version of Atollic TrueSTUDIO for ARM is also included on a CD in the EFM32WG-DK3850 kit.

10.3 Rowley Associates - CrossWorks for ARM

See the quick start guide for download details for CrossWorks for ARM. You will find CrossWorks project files in the

```
rowley
```

subfolder of each project.

10.4 CodeSourcery - Sourcery G++

See the quick start guide for download details for Sourcery G++. The

```
codesourcery
```

subfolder contains Makefiles for use with the Sourcery G++ development environment.

10.5 Keil - MDK-ARM

See the quick start guide for download details for evaluation versions of Keil MDK-ARM. The

```
arm
```

subfolder in each project contains project files for MDK-ARM. Please see the MDK-ARM documentation for usage details.

11 Schematics, Assembly Drawings and BOM

The schematics, assembly drawings and bill of materials for the three different boards included in the EFM32 Wonder Gecko Development Kit are available through Simplicity Studio when the correct kit documentation package has been installed.

12 Kit Revision History and Errata

12.1 Kit Revision History

The kit revision can be found printed on the box label of the kit, as outlined in the figure below.

Figure 12.1. Revision info

Table 12.1. Change log

Kit Revision	Released	Description
A00	20.12.2012	Initial kit release.

12.2 Kit Errata

Table 12.2. Errata

Kit Revision	Problem	Description
All	Trace does not work with current kit firmware (0v9p10).	Due to problems with the current kit firmware, embedded trace does not work either to the MCU or to external devices. This issue will be resolved in a future firmware update. Using an external trace emulator with "Trace In" mode still works.
All	Ethernet Interrupt is not currently working.	Due to issues with the current kit firmware (0v9p10), interrupts from the ethernet controller are not currently working. This will be fixed in a future firmware update.

13 Document Revision History

Table 13.1. Revision history

Revision Number	Effective Date	Change Description
0.11	10.10.2013	Updated document template and Silicon Labs contact/legal information.
0.10	07.01.2012	Initial document release.

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, Silicon Labs, SiLabs and the Silicon Labs logo, CMEMS®, EFM, EFM32, EFR, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZMac®, EZRadio®, EZRadioPRO®, DSPLL®, ISOmodem®, Precision32®, ProSLIC®, SiPHY®, USBXpress® and others are trademarks or registered 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

List of Tables

List of Examples

silabs.com