AN1304: GPIO Expander Based on EFM8 MCU Product Family

This is a code example running on the EFM8 8bit MCU product family, which can be reused to implement an I^2C/SMBus GPIO Expander. This covers all the EFM8 products via macros.

For example, when concerned about power consumption, customers may choose the EFM8SB devices. When concerned with cost, customers may use the EFM8BB devices.

The following MCU product families are supported:

- EFM8BB1 product family
- EFM8BB2 product family
- EFM8BB3 product family
- EFM8BB51 product family
- EFM8BB52 product family
- EFM8SB1 product family
- EFM8SB2 product family

KEY POINTS

- 8-24 additional GPIOs
- Input and output selectable
- Input and output selectable
- Push/pull/OD selectable
- Interrupt capable
- Sleep mode

1. Introduction

EFM8 devices offer Port 0, Port 1, and Port 2 (for the EFM8BB3 and EFM8BB52 MCU product family) as expanded GPIOs. In this way, usage of port-related SFRs is kept simple.

For communication, EFM8 devices use the SMBus peripheral in slave-mode. Default SMBus address is 0x78 (7bit addressed). SDA and SCL pins are defined out of Port 0/Port 1 by Xbar. Furthermore, an additional GPIO pin has been defined (as input) for address extension bit. If this input is connected to GND, the default SMBus slave address changes from 0x78 to 0x79. This solution allows the use of two EFM8s as IOEXPANDER and doubles the number of available I/O pins.

A further pin, configured as output, serves as interrupt output. Its default value is high and goes low in any cases when any pin of Port 0/Port 1, which is configured as input changes. Acknowledging this signal (rearming the interrupt) can happen through the I^2C/SMBus interface.

Note: I/Os on Port 2 on EFM8BB3 and EFM8BB52 MCU product family cannot be used as an interrupt source.

All functionality has been implemented by I^2C/SMBus commands except items listed above and controlling of reset of EFM8 devices to terminate STOP/SHUTDOWN operating modes.

2. I²C/SMBus Interface Commands

The following table contains all commands that have been implemented.

Table 2.1. I²C/SMBus Interface Commands

CMD	AUTO-INCREMENT	FUNCTION	PROTOCOL
0x00	Disable	Input Port 0	Read
0x80	Enable	Input Port 0	Read
0x01	Disable	Input Port 1	Read
0x81	Enable	Input Port 1	Read
0x04	Disable	Output Port 0	Read/Write
0x84	Enable	Output Port 0	Read/Write
0x05	Disable	Output Port 1	Read/Write
0x85	Enable	Output Port 1	Read/Write
0x08	Disable	Polarity Inversion Port 0	Read/Write
0x88	Enable	Polarity Inversion Port 0	Read/Write
0x09	Disable	Polarity Inversion Port 1	Read/Write
0x89	Enable	Polarity Inversion Port 1	Read/Write
0x0C	Disable	Configuration Port 0	Read/Write
0x8C	Enable	Configuration Port 0	Read/Write
0x0D	Disable	Configuration Port 1	Read/Write
0x8D	Enable	Configuration Port 1	Read/Write
0x10	Disable	Drive Strength Port 0	Read/Write
0x90	Enable	Drive Strength Port 0	Read/Write
0x11	Disable	Drive Strength Port 1	Read/Write
0x91	Enable	Drive Strength Port 1	Read/Write
0x70	Disable	Set SLEEP mode	Write
0xF0	Disable	Set SLEEP mode	Write
0x78	Disable	Device ID Byte 0	Read
0xF8	Enable	Device ID Byte 0	Read
0x79	Disable	Device ID Byte 1	Read
0xF9	Enable	Device ID Byte 1	Read
0x7A	Disable	Device ID Byte 2	Read
0xFA	Enable	Device ID Byte 2	Read
0x7B	Disable	Device ID Byte 3	Read
0xFB	Enable	Device ID Byte 3	Read

Note: Notes (operation instructions):

1. The Polarity Inversion command sets a mask which is XORed with a register value for reading Input or with the output-pattern for writing Output. Reading Output happens without involving the Polarity Inversion Mask and mirrors the natural state of port latch.

2. Configuration command example: 0x0D 0x70 sequence configures P1.4, P1.5, and P1.6 port pins as output (push-pull) while the others are kept in input (open-drain) state;
3. DRIVE strength command examples: 0x10 0x01 sequence sets HIGH-DRIVE-STRENGTH on P0; 0x90 0x00 0x00 sequence sets LOW-DRIVE-STRENGTH both on P0 and P1. Driver strength cannot be set for the P2 port on the EFM8BB3 and EFM8BB52 family devices.
4. SLEEP command stops the SoC, which remains in this state until reset occurs. This command has only one parameter: 0x01-cause STOP state of the SoC. Future improvement of this command can extend its capability to reach other low energy modes as well. Although 0xF0 has kept for command format compatibility reason, AUTO-INCREMENT functionality is meaningless for SLEEP command: 0x70 0x01 and 0xF0 0x01 sequences have the same result.
5. Logic behind AUTO-INCREMENT: AUTO-INCREMENT functionality makes it possible to read or write multiple bytes without needing to set the address byte every case.
6. Device ID usage example: 0x78, 0x79, 0x7A, 0x7B commands read back the hardcoded 'I', 'O', 'X', 'P' characters respectively;
7. The Device ID command supports AUTO-INCREMENT functionality as well, but 0xF8 is unique in this point of view. With the help of AUTO-INCREMENT, Device ID command can read out the value of DEVICEID, DERIVID, REVID, SMB0ADR registers, and UID from FLASH (BB2, BB3: 0xFFCF-0xFFC0), or from XRAM (BB1, SB1: 0xFF-0xFC; SB2: 0x0FFF-0x0FFC) after the 'IOXP' signature string; Sequence can be: 0xF8 0x18; Though the number of bytes to read is not maximized, there is no possibility of over read. If the number of bytes to read is bigger than the length of Device ID, the remaining place is filled with the part of repeated Device ID.
8. Command codes not listed are reserved.

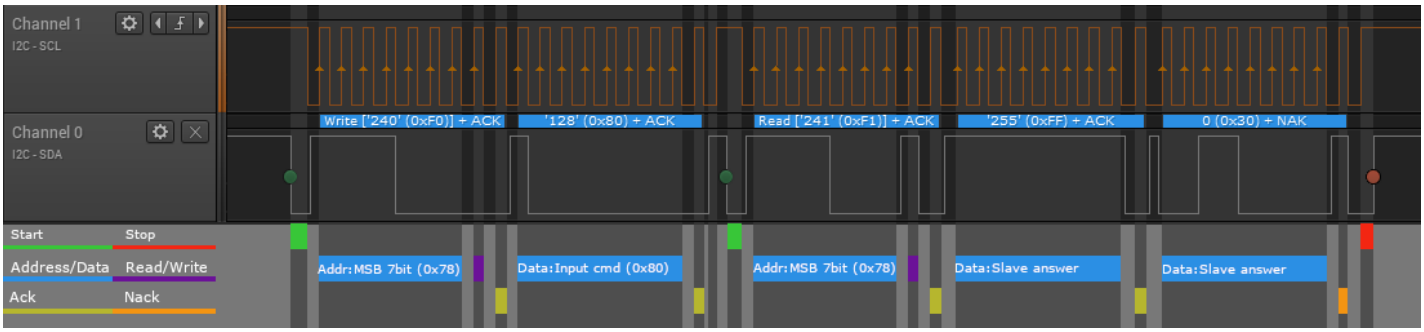


Figure 2.1. Example of SMBus Transition

3. Sensing of Input Changes by Interrupt

1. P0 and P1 are configured as a source of matching interrupt. At initialization state, all pins are involved to a matching mask. This mask(s) can change each time pin configuration changes (PnMDOUT).
2. When any of the pins, which are configured as input, have changed, the matching ISR is called.
3. Matching ISR sets (pulls to low) the output pin and disables Port Matching Interrupt (EMAT).
4. The SW keeps this state until P0 and P1 have read through I^2C/SMBus command.
5. When P0 and P1 are reading, their current state is saved to PnMAT register as the new expected value.
6. Rearm Port Matching Interrupt (EMAT).

4. Test Application on EFR32 MCU Family

For test reasons and as an operating example, an application example has been implemented on the EFR32 family. More information on this example application can be found on the Silicon Labs GitHub page under the application_examples/platform/platform_applications/platform_i2c_test_for_efm8_ioexpander folder.

- Its basic functionality: The application works as an I^2C master and can sense the interrupt signal from IOEXPANDER as a simple input.
- It offers a serial console and a command interpreter to ensure a test interface for IOEXPANDER.
- The application checks the interrupt input in a polling process and lights an LED when it receives a signal from the IOEXPANDER.

Table 4.1. Available Commands of the Test Application

COMMAND	DESCRIPTION
loop	Execute commands repeatedly
addr	Get/Set I^2C slave address
read	Read register from IOEXP
write	Write register of IOEXP
mod	Read Modify Write a register of IOEXP
sleep	Set a waiting time period (msec)
time	Get current system time
ver	Get build information

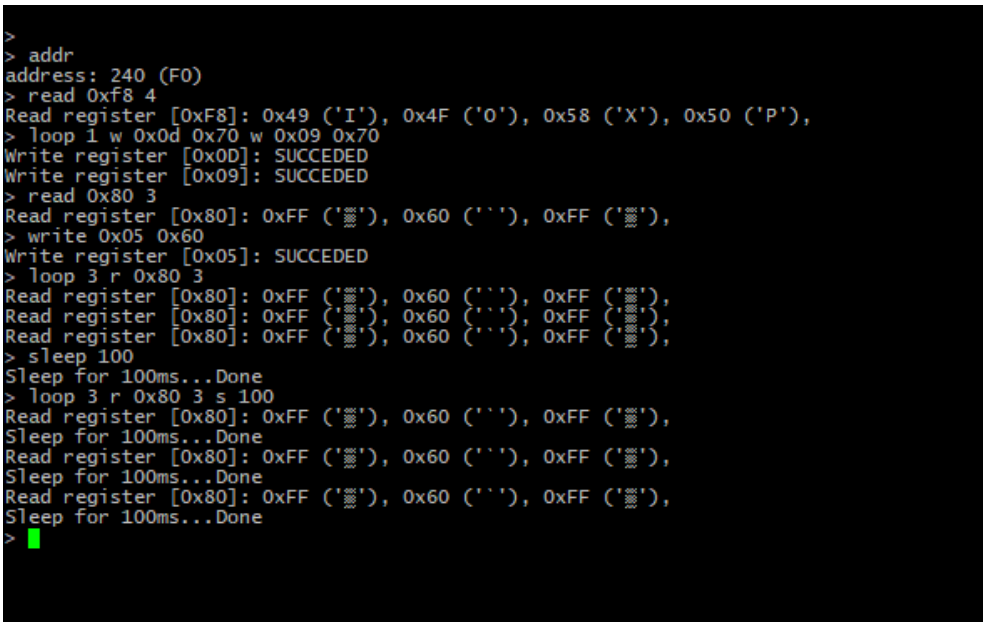


Figure 4.1. Example of Test Application Console

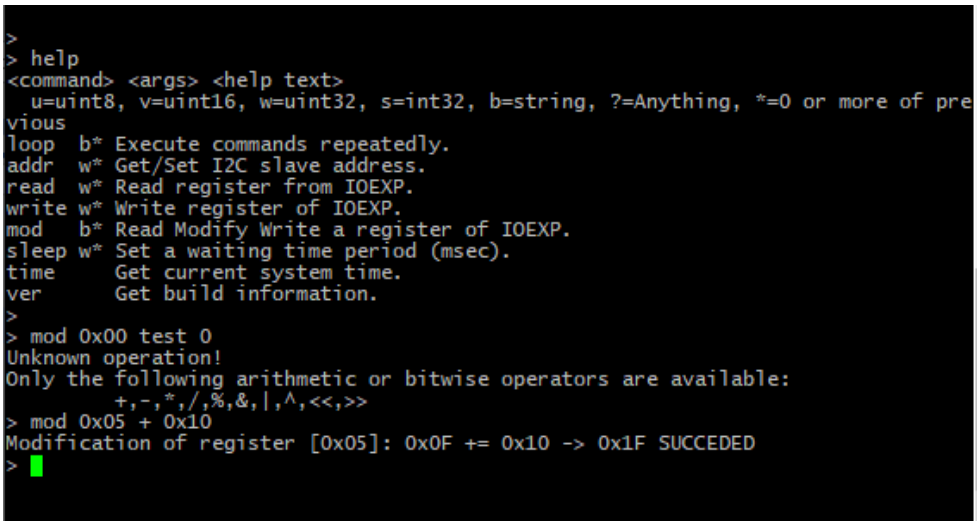


Figure 4.2. Using the “mod” Command Correctly

5. Hardware Configuration Guide

Every device has a different default pin configuration to fit the appropriate SLSTK. However these pins, which are used for SCL, SDA, interrupt or address extension are configurable through .hwconf and EFM8_IOEXP_common.h files with the following limitations:

- SCL and SDA pins are controlled by crossbar so must be in its range.
- Port0 (0.0-0.7) and Port1(1.0-1.7) should be keep free, as far as it possible.

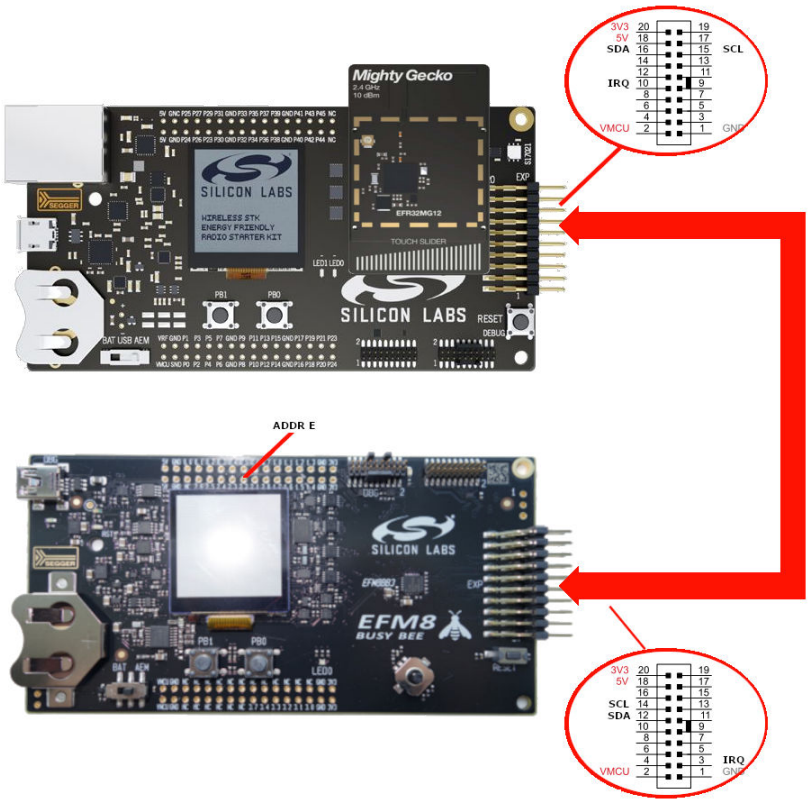


Figure 5.1. Example of Connection Between EFR32 and EFM8BB3

Configuration in EFM8_IOEXP_common.h

```
// P0.0 - P0.7 delegated as port 0;
// P1.0 - P1.7 delegated as port 1;
// P2.4 - P2.6 delegated as port 2 (port2.5-port2.7);
// P3.0 - P3.4 delegated as port 2 (port2.0-port2.4);
// MCU STK2022A Function
// port port description
// P2.0 EXP12 SMB SDA;
// P2.1 EXP14 SMB SCL;
// P2.2 ----- ADDR_EXT;
// P2.3 EXP03 INP_CHANGED_IRQ;
SI_SBIT(IOEXP_INP_CHANGED_IRQ, SFR_P2, 3);
SI_SBIT(IOEXP_ADDR_EXT, SFR_P2, 2);
#define P0DEVMASK 0xFF
#define P1DEVMASK 0xFF
#define P2DEVMASK 0x70
#define P3DEVMASK 0x1F
```

6. Revision History

Revision 0.2

March, 2021

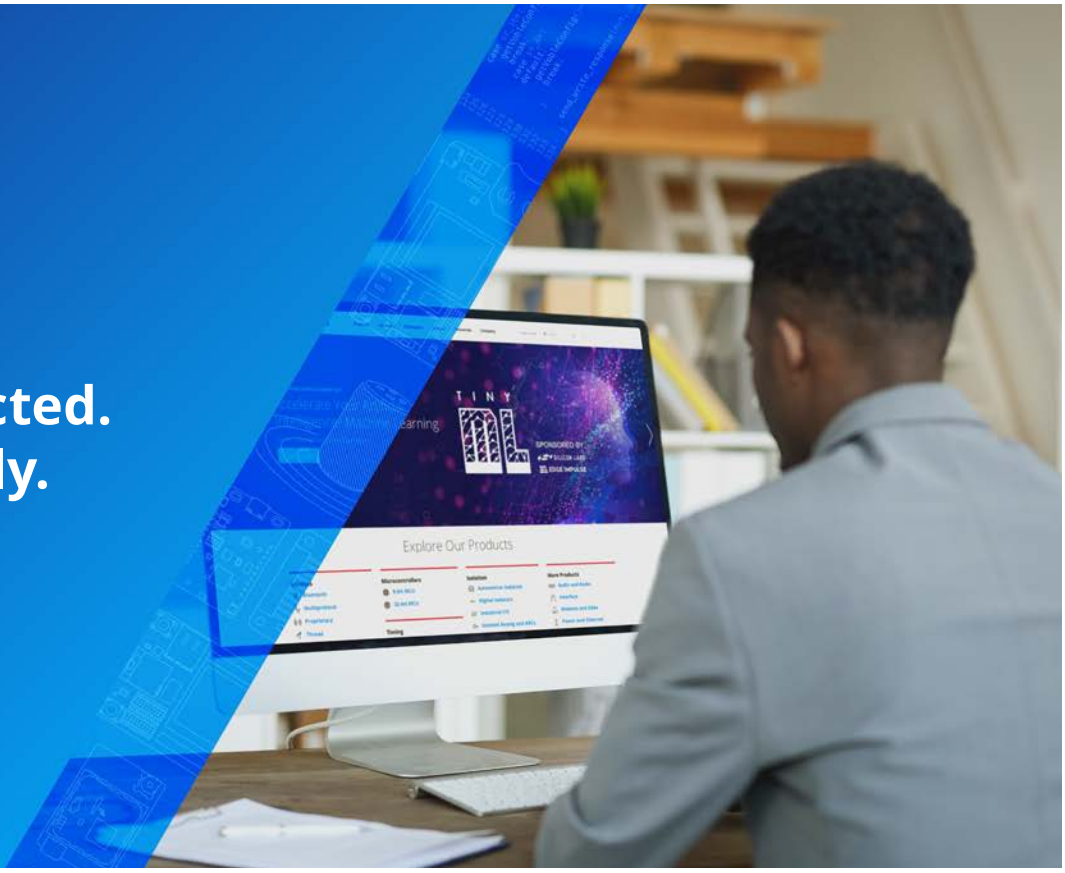
- Added support for the EFM8BB51 product family

Revision 0.1

December, 2020

- Initial release.

Smart. Connected. Energy-Friendly.



IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com