University of British Columbia
Electrical and Computer Engineering
Electrical and Biomedical Engineering Design Studio
ELEC291/ELEC292

Laboratory 1 - The EFM8 Microcontroller Board

Introduction

This laboratory module shows how to build a simple microcontroller system using the Silabs EFM8 microcontroller. The EFM8 is a modern, very powerful, derivative of the Industry Standard 8051 microcontroller developed by Intel in the early 1980's. As part of this module, students will compile, download, and run programs into the microcontroller system. Finally in this laboratory module, students will be attaching an LCD to the microcontroller system and display their names and student number on it.

Pre-laboratory

1. Find the datasheet/reference manual for the following components used in this laboratory module. Keep them for reference.
    a. Silabs' EFM8 microcontroller board.
    b. Hitachi's HD44780 LCD controller.
2. Find the 8051 microcontroller assembly instruction set. Describe the major differences between the 8051 microcontroller assembly and the processor assembly you learned in CPEN211 (ARM, 68K, NIOS II, etc.)

Laboratory

The laboratory component of this module consists of three activities:

1. Setting up the EFM8 Microcontroller Board. In previous years the students were required to assemble this board. This year, due to the pandemic, the boards are pre-assembled and available in the kit they order from RP-electronics.
2. Attaching the LCD to the Microcontroller Board.
3. Display your name and student number using the Microcontroller/LCD System.

1) Setting up the Microcontroller Board

Figure 1 shows the circuit schematic of the microcontroller system used in this laboratory assignment. This year, the boards come pre-assembled due to the COVID19 pandemic, so this is here for reference only.

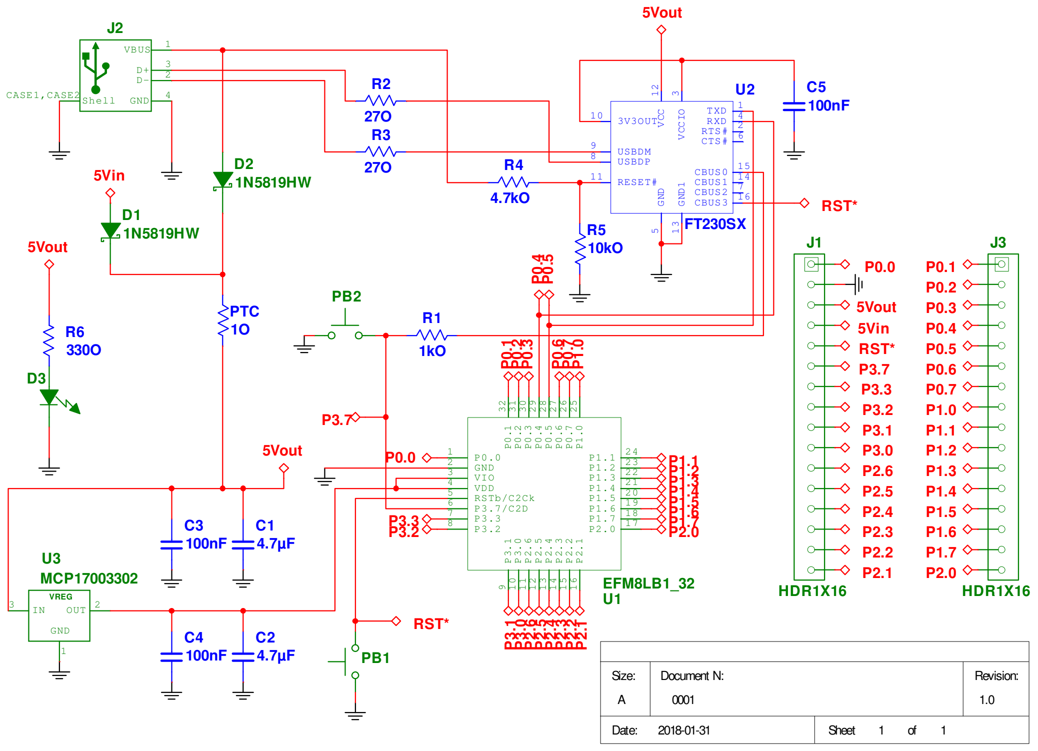


Figure 1. The EFM8 microcontroller Board circuit diagram.

The next picture shows the EFM8 microcontroller system with a red LED attached to pin P2.1 for testing.

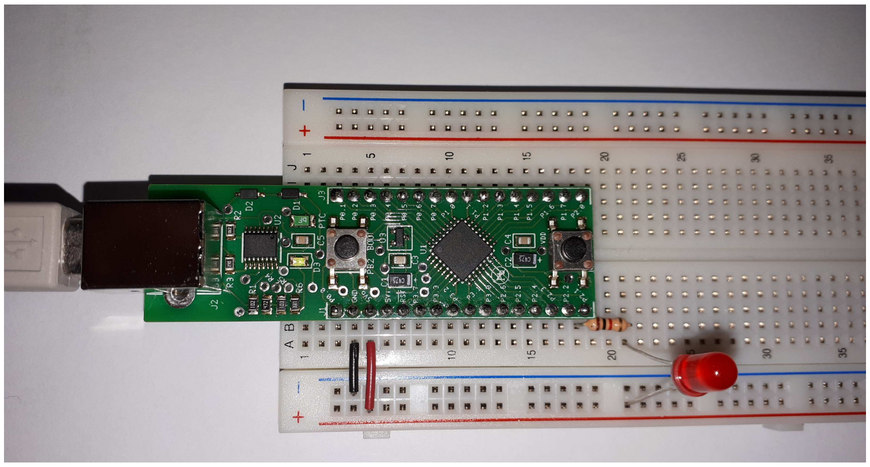


Figure 2. The EFM8 microcontroller board in the breadboard with a red LED attached to pin P2.1.

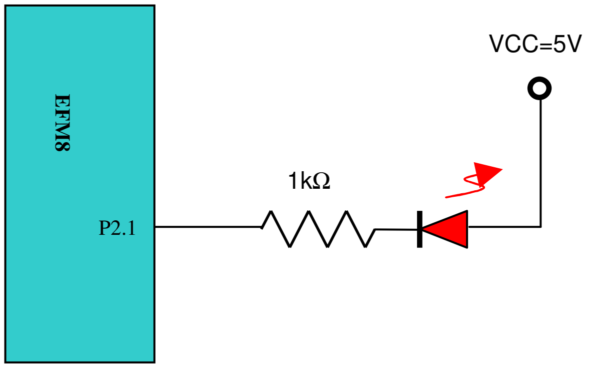
Writing, Compiling, and Downloading Code to the EFM8 Microcontroller Board.

a) Download and install CrossIDE from the course webpage.

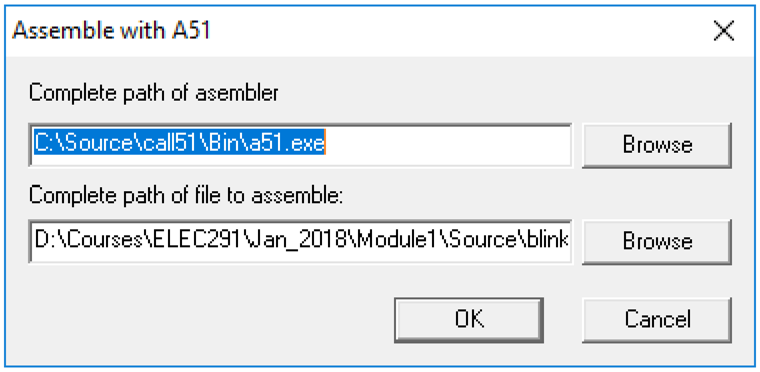
b) Run CrossIDE, and create a new 'Asm' file. Add the code listed below. Tip 1: copy and paste from this document! Tip 2: Sometimes this code is available from the course web page.

```
; Blinky.asm: toggles an LED attached to pin 2.1

$MODEFM8LB1

CSEG at 0H
    ljmp main

Wait_half_second:
    ;For a 6MHz clock one machine cycle takes 1/6.0000MHz=166.666ns
    mov R2, #25
L3: mov R1, #250
L2: mov R0, #120
L1: djnz R0, L1 ; 4 machine cycles-> 4*166.666ns*120=80us
    djnz R1, L2 ; 80us*250=0.02s
    djnz R2, L3 ; 0.02s*25=0.5s
    ret

main:
    ; DISABLE WDT: provide Watchdog disable keys
    mov WDTCN, #0xDE ; First key
    mov WDTCN, #0xAD ; Second key

    mov SP, #7FH
    ; Enable crossbar and weak pull-ups
    mov XBR0, #0x00
    mov XBR1, #0x00
    mov XBR2, #0x40
    mov P2MDOUT, #0x02 ; make LED pin (P2.1) output push-pull
M0:
    cpl P2.1 ; Led off/on
    lcall Wait_half_second
    sjmp M0
end
```

c) Attach a red LED in series with a 1k Ohm resistor to P2.1 of the EFM8 microcontroller. The LED circuit is shown below.



d) Compile and link the program using A51. In CrossIDE click "Build", then "Compile/Link with A51", a pop-up like this one is displayed:


Assemble with A51

Complete path of asembler

C:\Source\call51\Bin\a51.exe

Complete path of file to assemble:

D:\Courses\ELEC291\Jan_2018\Module1\Source\blink

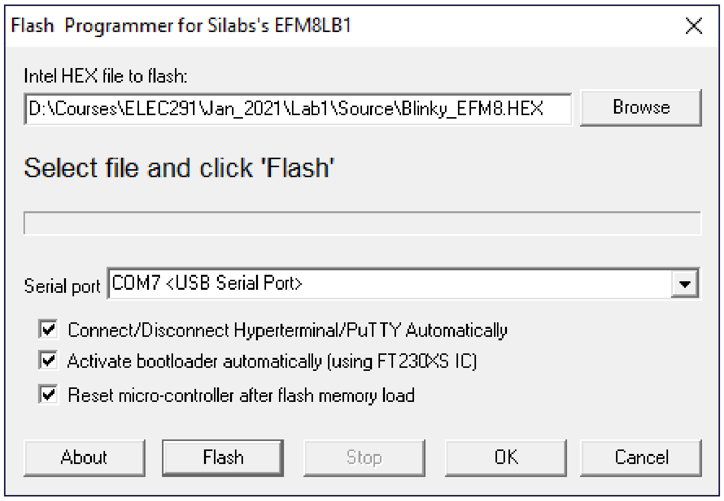
Make sure the complete path of a51.exe is correct and then click ok. a51.exe is located in the same folder where Crosside is installed, in a sub-folder called "call51\bin". If everything went well and the program has no errors, the compiler creates the file 'blinky.hex':

```
------- CrossIde - Assembling -------
D:\Courses\ELEC291\Jan_2018\Module1\Source\blinky.asm:1: Assembling...
No errors found
```

The HEX file contains the binary program that is downloaded to the microcontroller system. If the program has errors, they will be reported directly into CrossIDE. The errors need to be fixed and the compilation process should be repeated again.

To download the code to the microcontroller system use CrossIDE's built-in flash loader. Before proceeding with this step, attach an USB cable to the EFM8 board and the computer.

On CrossIDE click "fLash" followed by "Silabs EFM8LB1". A pop-up window like this one is displayed:



Flash Programmer for Silabs's EFM8LB1

Intel HEX file to flash: D:\Courses\ELEC291\Jan_2021\Lab1\Source\Blinky_EFM8.HEX

Select file and click 'Flash'

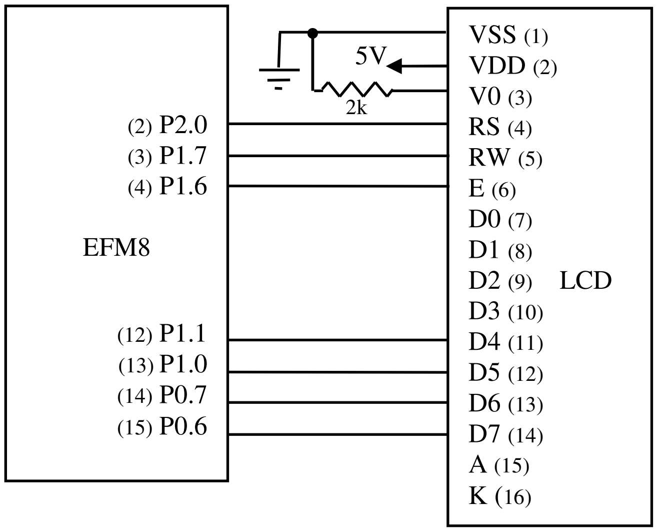
Serial port COM7 <USB Serial Port>

- [x] Connect/Disconnect Hyperterminal/PuTTY Automatically
- [x] Activate bootloader automatically (using FT230XS IC)
- [x] Reset micro-controller after flash memory load

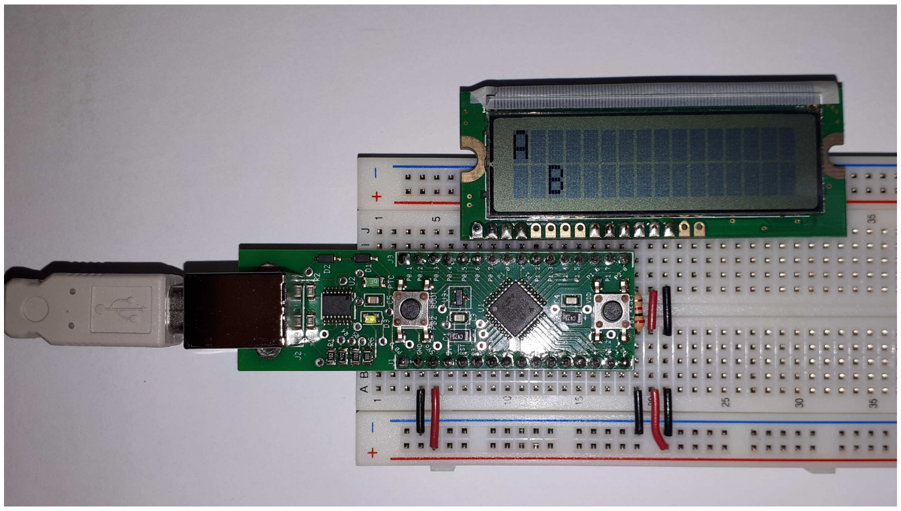
Select the HEX file created by the compiler, and click "Flash". Wait for the process to complete. The LED should start blinking.

2) Attaching the LCD to the Microcontroller System

Included in the Microcontroller parts kit is the LCM-S01602DTR/M Liquid Crystal Display Module (LCD). This LCD module uses the industry standard HD44780 controller. The LCD can be configured either in 4-bit or 8-bit mode. For this laboratory experiment examples the LCD will be configured in 4-bit mode in order to minimize the number of I/O pins and pins used. The figure below shows the connections between the microcontroller system and the LCD. The pin assignments in the microcontroller are arbitrary. The pins used in the microcontroller can be changed, but the example source code provided should be modified to match the wiring.



The next figure shows the microcontroller board with the LCD attached. In order to verify the functionality of the microcontroller/LCD system, a test program called 'LCD_test_4bit.asm' is provided in the course web page. Compile and load the program into the microcontroller system. After reset the LCD should display letter ‘A’ in the first column of row one and letter ‘B’ in the third column of row two.



3) Display your name and student name using the Microcontroller/LCD System

Write a program to display your name and student number using the LCD. The first row should display your name, while the second row should display your student number. You can use the test program as a reference and/or starting point for your program. **DO NOT** use an inefficient ‘brute force’ approach like this one:

```
mov a, #0x80 ; Move cursor to line 1 column 1
lcall WriteCommand
mov a, #'J'
lcall WriteData
mov a, #0x81 ; Move cursor to line 1 column 2
lcall WriteCommand
mov a, #'e'
lcall WriteData
mov a, #0x82 ; Move cursor to line 1 column 3
lcall WriteCommand
mov a, #'s'
lcall WriteData
mov a, #0x83 ; Move cursor to line 1 column 4
lcall WriteCommand
mov a, #'u'
lcall WriteData
mov a, #0x84 ; Move cursor to line 1 column 5
lcall WriteCommand
mov a, #'s'
etc.
```

← NO!

INSTEAD, use a pre-defined constant string and the ‘`movc a, @dptr+a`’ instruction. To define constant strings with the 8051 assembler, the syntax looks like:

```
MyName: DB ‘Jesus Calvino-Fraga’, 0
MyID: DB ‘12345678’, 0
```

The ‘0’ at the end of the string is used to indicate the end of the string, so the program you write knows when to stop sending characters to the LCD. Remember to place the strings outside the loops of executable code.

Upload your program to Canvas together with a picture or video of your working LCD. Don’t forget to add extra functionality and/or features for bonus marks.

A few tips:

a) The LCD can be configured to scroll left/right.
b) You can create and use up to 8 custom characters.
c) Write a function to display a constant string pointed by register dptr.