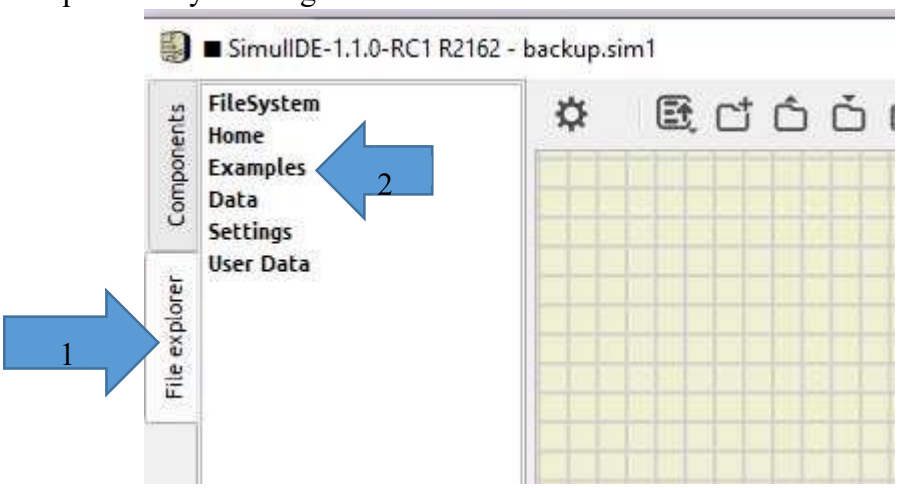
CENG482 Embedded Systems Final Project
Spring 2023-2024
Final Project

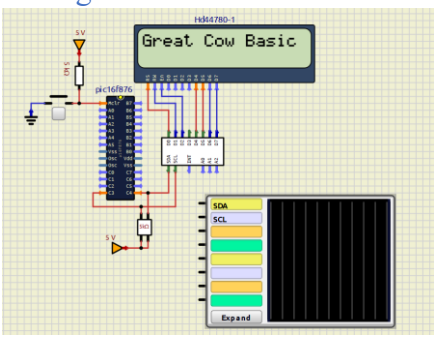
1. Download SimulIDE version 1.1. The program is also on the Olearn site.
2. Run the program
3. Access the example files by clicking here:



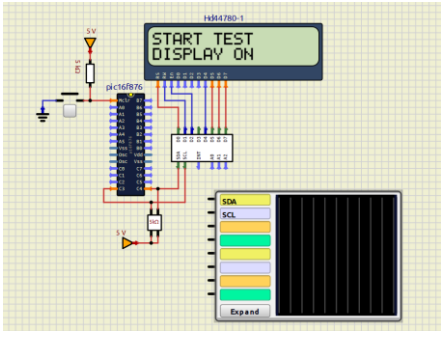
4. Open the example for I2C simulation using the PIC16F876. It is in file:
SimulIDE_1.1.0-RC1_Win32/examples/Micro/Pic/16F876_I2C-Lcd/i2c-lcd.sim1

5. Run the simulation and press the button. Explains what happens. Include a screenshot.

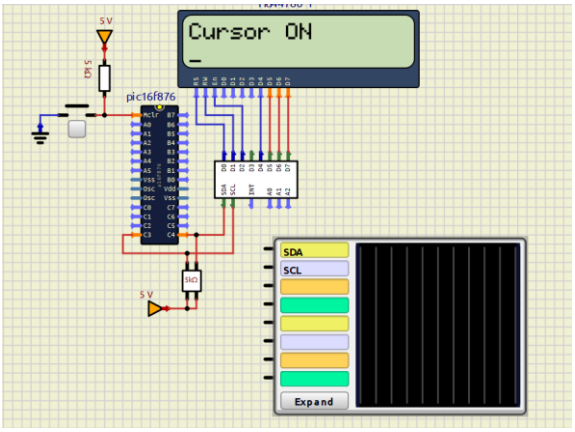
When I started the simulation, the microcontroller initialized the I2C communication and set up the LCD screen. First, the LCD displayed the message "Great Cow Basic" for about 1 second.

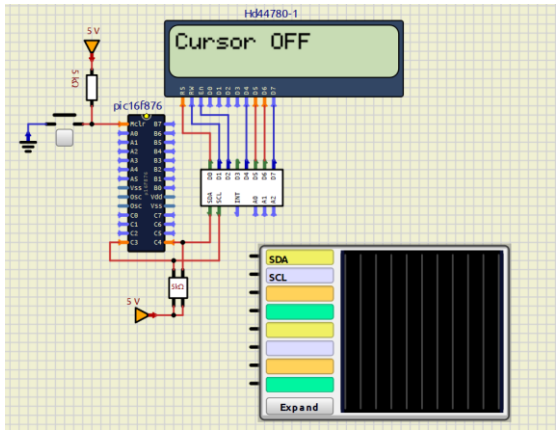


Next, the screen cleared, and then the message "START TEST" appeared followed by "DISPLAY ON" on the second line of the LCD, and it stayed visible for about 3 seconds.

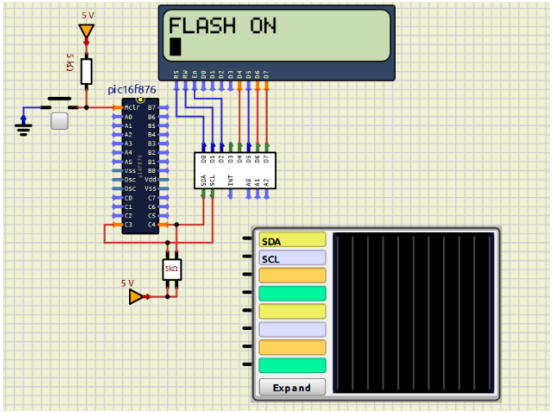


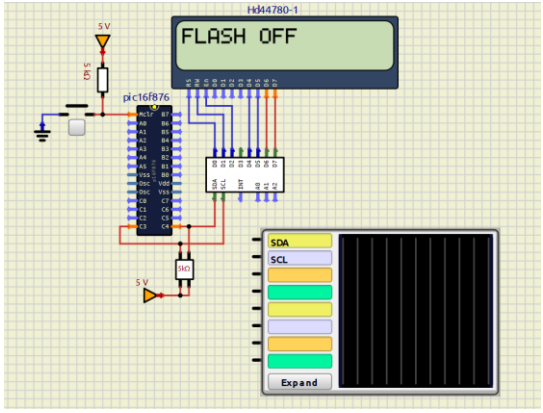
After that, the screen cleared again, and the message "Cursor ON" showed up and the cursor on the LCD was visible and blinking, which stayed like that for another 3 seconds. Then, the screen cleared, and the message changed to "Cursor OFF" and the cursor was no longer visible.



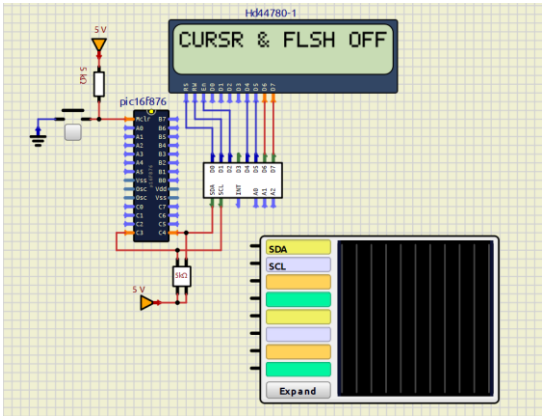


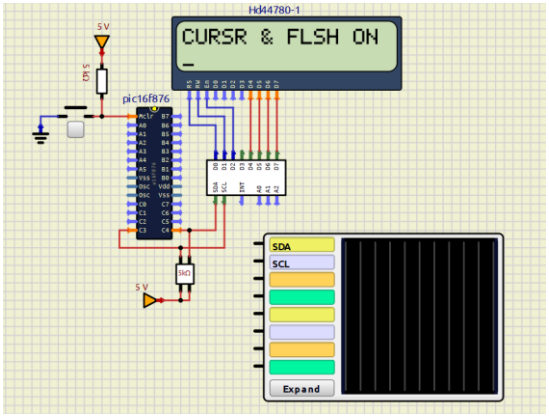
The screen cleared once more and displayed "FLASH ON", which made the cursor start flashing on and off, this lasted for 3 seconds. Then, when the screen displayed "FLASH OFF" the cursor stopped flashing and turned off staying like this for another 3 seconds.



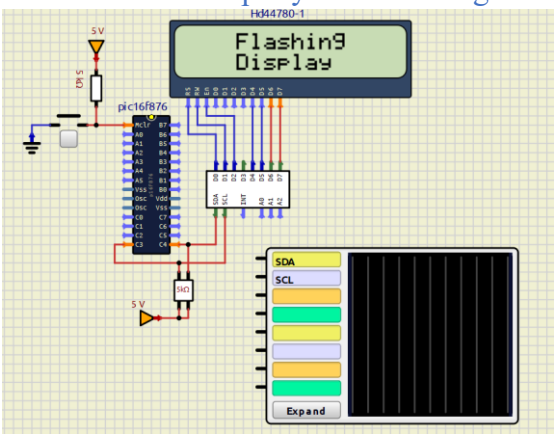


Next, the LCD showed "CURSR & FLSH ON," and both the cursor and the flashing were turned on simultaneously, remaining visible for 3 seconds. Then, "CURSR & FLSH OFF" appeared on the screen, turning both the cursor and the flashing off. This sequence demonstrated the control over both cursor visibility and flashing simultaneously.

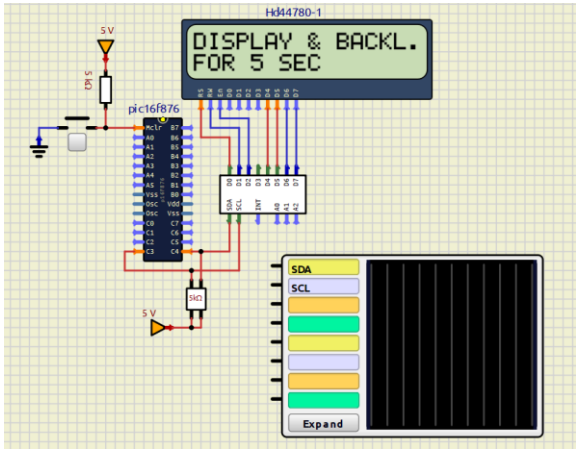




After that, the screen cleared and displayed "Flashing" on the first line and "Display" on the second line. The entire display then began to flash on and off rapidly. This flashing continued for about 5 seconds.

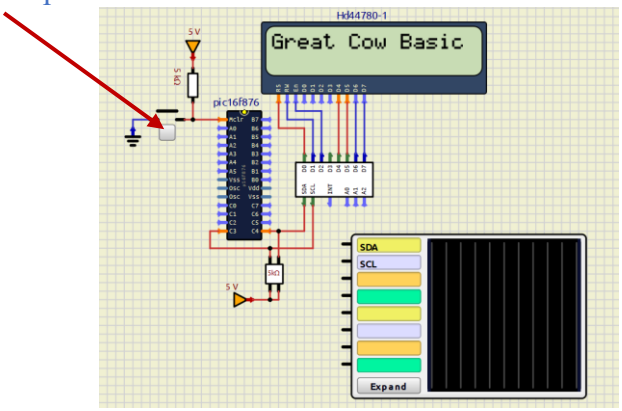


Finally, the screen showed "DISPLAY & BACKL. FOR 5 SEC" for 2 seconds. Then, both the display and the backlight turned off for about 5 seconds. After this, the backlight came back on, the screen cleared, and the message "END TEST" appeared, staying visible for 3 seconds before the loop started over again.



Button Functionality:

At any point during the loop execution, pressing the button triggers a reset of the entire process because it is connected to the MCLR pin. This means the loop restarts from the beginning, reinitializing the display and repeating the sequence of operations.



6. Below is a section of Assembly Language code. Look up what each of the commands bcf, movlw, movwf, call, clrf, decfsz, goto and nop mean in assembly language from the manual for the microcontroller, which is on the website. Try to guess what this code does.

```
CLS
```

(Clear Screen or equivalent) a macro or label indicating the start of a screen clear routine.

```
bcf SYSLCDTEMP,1
```

Clear bit 1 of the SYSLCDTEMP register. This might be used to reset a specific status flag related to the LCD.

```
movlw       1
movwf       LCDBYTE
call LCDNORMALWRITEBYTE
movlw       4
```

1. movlw 1:Load the literal value 1 into the W register.
2. movwf LCDBYTE: Move the value from the W register into the LCDBYTE register. This likely prepares the LCDBYTE register for an LCD operation.
3. call LCDNORMALWRITEBYTE: Call the subroutine LCDNORMALWRITEBYTE. This likely writes the byte stored in LCDBYTE to the LCD in normal mode.
4. movlw 4: Load the literal value 4 into the W register.

```
movwf   SysWaitTempMS
clrf SysWaitTempMS_H
call Delay_MS
```

1. movwf SysWaitTempMS: Move the value from the W register into the SysWaitTempMS register. This sets up a delay value in milliseconds.
2. clrf SysWaitTempMS_H: Clear the SysWaitTempMS_H register. This might be used to reset the higher byte of the delay value.
3. call Delay_MS: Call the subroutine Delay_MS. This causes a delay for the number of milliseconds specified in SysWaitTempMS.

```
movlw   128
movwf       LCDBYTE
call LCDNORMALWRITEBYTE
movlw       66
movwf       DELAYTEMP
```

1. movlw 128: Load the literal value 128 into the W register.
2. movwf LCDBYTE: Move the value from the W register into the LCDBYTE register. This prepares the LCDBYTE register for another LCD operation.
3. call LCDNORMALWRITEBYTE: Call the subroutine LCDNORMALWRITEBYTE again to write the byte 128 to the LCD in normal mode.
4. movlw 66: Load the literal value 66 into the W register.
5. movwf DELAYTEMP: Move the value from the W register into the DELAYTEMP register. This sets up a delay value for a microsecond delay loop.

```
DelayUS1
decfsz      DELAYTEMP,F
goto DelayUS1
nop
return
```

1. DelayUS1: This is a label for the start of a delay loop.
2. decfsz DELAYTEMP,F: Decrement the DELAYTEMP register. If the result is not zero, continue to the next instruction. If it is zero, skip the next instruction.
3. goto DelayUS1: If the DELAYTEMP register is not zero, jump back to the DelayUS1 label, creating a delay loop.
4. nop: No operation. This instruction does nothing and is often used for timing adjustments or to wait for hardware to stabilize.
5. return: Return from the subroutine to the calling code.

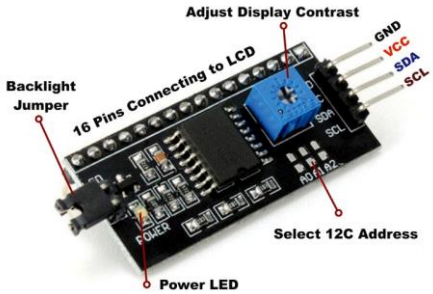
Therefore this section of code performs the following:

The code appears to initialize and configure an LCD display. It sends specific commands to the LCD, introduces delays to ensure proper operation, and then returns to the main program.

ASSEMBLY COMMANDS	DEFINITION
BCF	Clears a specific bit in a register.
MOVLW	Loads an 8-bit constant (literal value) into the W register.
MOVWF	Moves the value from the W register into a specified file register.
CALL	Calls a subroutine at the specified address.
CLRF	Clears the contents of a specified file register.
DECFSZ	Decrements the value in a file register and skips the next instruction if the result is zero.
GOTO	Jumps to a specified address.
NOP	Does nothing for one instruction cycle.

7. Research online: what is I2C? Write 1 paragraph to explain. Give two real world examples.

As per my research I2C (Inter-Integrated Circuit) is a serial communication protocol developed by Philips Semiconductor that allows multiple integrated circuits (ICs) to communicate over two wires: a data line (SDA) and a clock line (SCL). It supports multiple master and slave devices on the same bus, each with a unique address, allowing for efficient communication with minimal wiring. I2C is popular for its simplicity and versatility in connecting various peripherals.



Real World Examples:

1. Temperature Sensors: In smart home devices, I2C is used to connect temperature sensors like the DHT22 to a central microcontroller, enabling efficient data collection and processing.
2. LCD Displays: I2C is commonly used in embedded systems to connect microcontrollers with LCD displays, such as the YwRobot LCD1602, reducing the number of required GPIO pins and simplifying circuit design.

8. If you do not have one, open an account on github. Upload the report with the answers to 5-7 in an archive on the website.
   Your report should be called CENG_482_Final_200218322.
   Where you replace the numbers 123456789 with your student id number. Your repository can be called anything you want.

   Don't make the repository public, instead share it privately as explained here: https://stackoverflow.com/questions/71911601/share-the-github-repo-with-someonewothout-making-it-public
   You can use the desktop github app from here: https://desktop.github.com/
   If you need to learn how to use the desktop app, there is a tutorial.
   You do not have to use desktop, if you followed along at the beginning of the semester you can run git from command line.
   You can also use the website directly.