BME 580 Microcontrollers I

Lab 3: OLED – Organic LED Display Modules *Ross Rucho*

In this lab we will be using an OLED display**.** These are very bright, high-contrast displays that can be purchased for about $25 each, or substantially less if you purchase them in bulk from the manufacturer (usually in China). These displays are small enough that they can easily be incorporated into portable devices and wearable technologies. OLED displays are becoming very common in medical devices.

OLED displays do not necessarily have touch-screen interfaces. This is because most embedded applications (devices and circuits that use microcontrollers) do not need very sophisticated user interfaces. You see this sort of thing all the time in simple consumer appliances, home temperature controllers, gas station pumps, even simple electronic toys. Just a few hardware buttons and a display are all you need to get the job done for most embedded systems. Unlike a smart phone, most embedded applications only perform one function with just a few inputs, such as controlling a microwave oven or setting the temperature for an air conditioner. IoT will change the way we interface remotely with our appliances, but this will tend to make the physical human interface on those devices even simpler, if they change at all.

Before you begin this lab you should read the documents listed below. These will help you understand what you are doing in the lab, and this material will be covered on the final examination as well.

Reading assignment:

Overview of OLED displays

https://en.wikipedia.org/wiki/OLED

I2C interface that we will be using to interface the microcontroller with the OLED display:

https://en.wikipedia.org/wiki/I%C2%B2C

Find the CCS C Compiler Manual on the lab workstation, make sure you place a copy of this manual on your laptop. Read the following section:

“printf()” or it might be titled “printf() fprintf()”

BME 580 Microcontrollers I

OTHER DOCUMENTS AND FILES YOU WILL NEED FOR THIS LAB:

oled_2015.c This is the source code that you will need to compile and load on the microcontroller

a copy of this source code is given in Appendix 1 of this lab template document

https://www.dropbox.com/s/hd95u8ayz01a6d9/oled_2015_a.c?dl=0

here is a link to the compiled (*.hex) code:

https://www.dropbox.com/s/vajgddvxd68xtrj/oled_2015_a.hex?dl=0

oled_16F1829_i2c.h This is a header file you need to include when you compile the source code

a copy of this header file is given in Appendix 2 of this lab template document

https://www.dropbox.com/s/7nvywtyv8gf9bx6/oled_16F1829_i2c.h?dl=0

OLED Module data sheet

https://www.dropbox.com/s/yawnfghxahxzc3n/LM096-128064--OLED-MODULE-DATASHEET.pdf?dl=0

OLED Driver data sheet

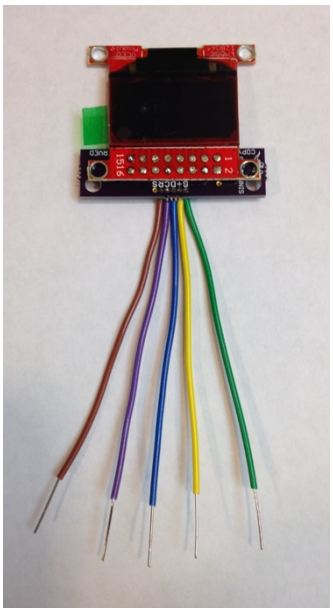
https://www.dropbox.com/s/kqxek26ykh6hjkd/SSD1306-1.2--OLED-DRIVER-DATASHEET.pdf?dl=0

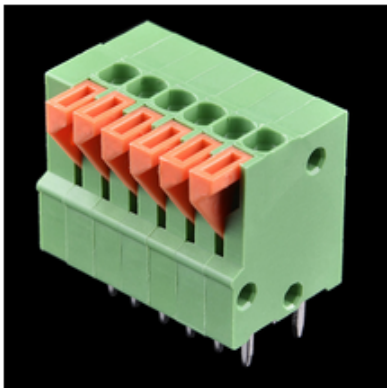
BME 580 Microcontrollers I

OLED DISPLAY MODULE

For this laboratory you will need to get an OLED display module from the instructor.

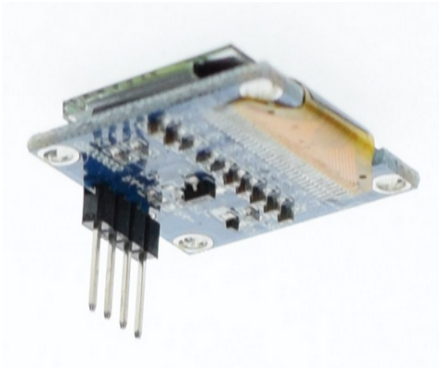
There are two options: one with 5 wires, one with 4 pins:





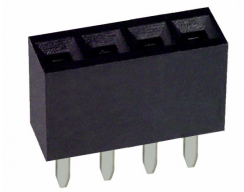
You will also use a 6-position spring terminal, shown above.

Alternatively, your kit may have a 4-pin OLED. Functionally these two different OLED modules are identical.



If you get an OLED with 4 rigid pins instead of 5 wires, then you should use the 4-position header (both shown to the left)

When you build the OLED display into your circuit board I want you to use the spring terminal (above) or the header (left) so that the OLED display can be easily removed and replaced. Please do not cut or change the wires on the OLED display. They have been tested and they should work perfectly without needing to be modified.



The wires on the OLED module use the same color code that was used in the earlier lab (Hello World):

Yellow = +5V
Green = ground
Brown = reset
Blue = digital data
Violet = digital clock

The OLED module with 4 pins does not have the "RESET" function, which is OK since you do not really need that function. They are otherwise identical. Just be sure to check the 4 pin functions, which are labeled on the OLED PCB for the OLED module with 4 rigid pins.

BME 580 Microcontrollers I

OLED Module setup

As you can see in the OLED documentation, these OLED displays can be configured for several different interface protocols. We will use I2C, which you read about on Wikipedia (in the Assigned Readings).

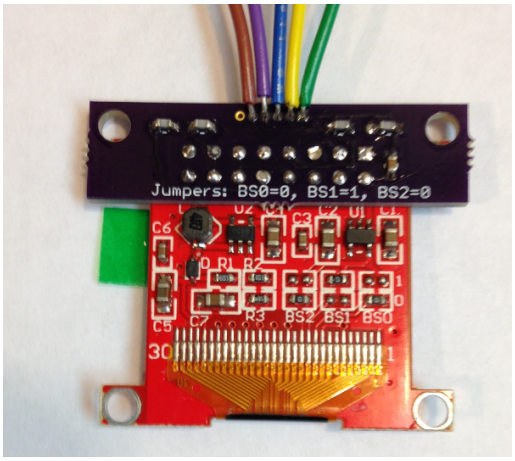
You do not have to do anything to configure the OLED module for this lab; it has been done for you. To understand how it was modified, look at the picture below of the back of the OLED module. First you should note that the OLED module uses Surface Mount Device (SMD) components. In the lower right area you can find three jumpers: BS2 BS1 BS0. Each of these can be set to one of two values: 1 or 0

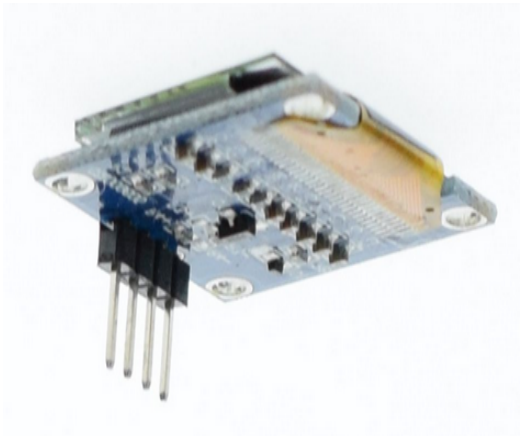
To set the OLED Display for the I2C interface, these jumpers are soldered into the following positions:

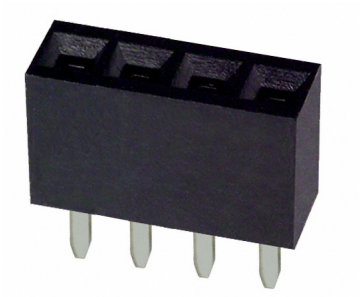
BS2 = 0 BS1 = 1 BS0 = 0

The SMD jumpers are little black resistors with a resistance of 0 ohms.

Also note that the jumper values for I2C are indicated on the silkscreen (white text) on the adapter PCB.



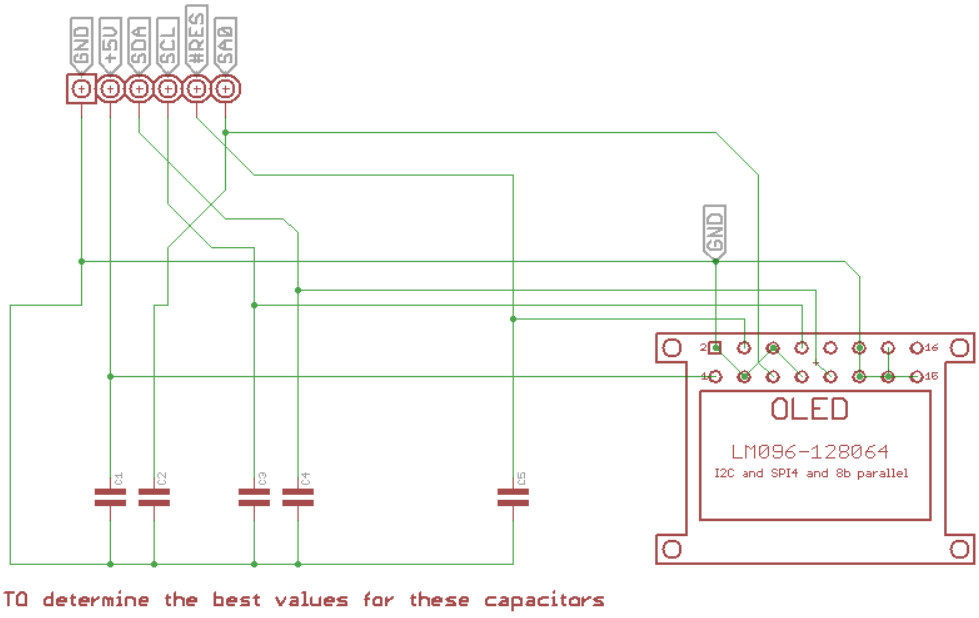




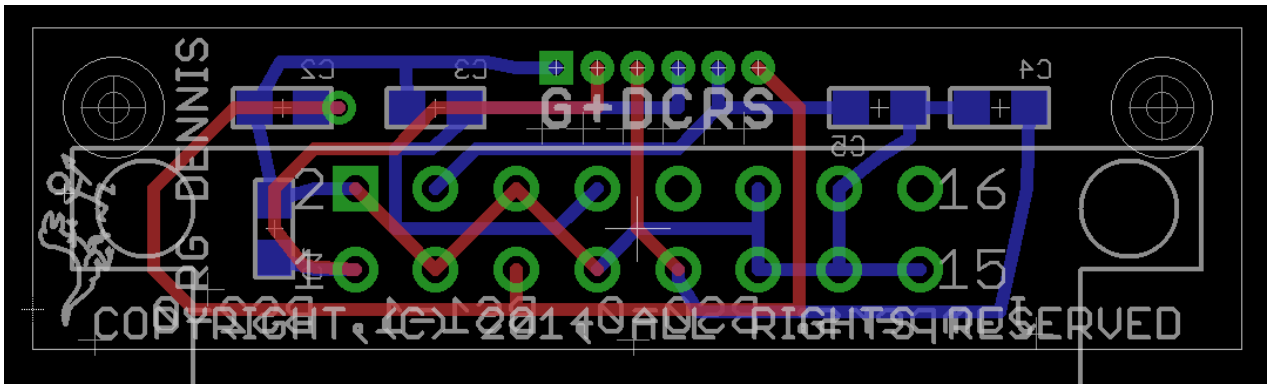
BME 580 Microcontrollers I

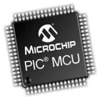
ADAPTER PCB (only for the OLED with 5 wires soldered on)

You will note that the 5-wired OLED Display Module has a small blue PCB soldered to it, with 5 colored wires sticking out. The small blue PCB is an adapter PCB that I designed to make the OLED module easier to use and more stable (less glitchy if you touch it). By adding the adapter PCB, the number of connections you have to solder is reduced from 16 down to 5. And the adapter PCB also includes 5 additional capacitors which tend to filter out the glitch problems you would see if you accidently touch the OLED module or connection wires: very light contact with these wires can make the display malfunction temporarily. So, adding this adapter PCB makes the OLED module more reliable as well as simpler to use for I2C. The newer OLED modules released in 2017 (with 4 rigid pins) have this design improvement built-in, so we do not need an adapter PCB. The schematic for the PCB adapter and the PCB layout were both designed in EagleCAD, and are shown below:



NEED TO determine the best values for these capacitors





Connecting the OLED to the microcontroller

You will build upon the circuit that you built for the Hello World lab. The connections you will use for the OLED display module are shown below:

```
                                   PIC16F1829  8SOIC Pinout:
                                  ___________    ___________
                                 |           [___]           |
         +5V power            1- | +5V                   GND |-20 - Ground
                                 |                           |
                              2- | RA5       RA0,AN0,PGD,DAC |-19 - Programming Port “PGD”
                                 |                           |
                              3- | RA4, AN3      RA1,AN1,PGC |-18 - Programming Port “PGC”
                                 |                           |
Programming Port “/MCLR”      4- | /MCLR, RA3  RA2,AN2,CCP3  |-17 - GREEN LED
                                 |                           |
 SD-Card module  TX-O  >>>    5-| RC5, CCP1, Rx          RC0 |-16 – potentiometer (analog INPUT)
                                 |                           |
  SD-Card module RX-I <<<     6- | RC4, Tx                RC1 |-15
                                 |                           |
                              7- | RC3, CCP2              RC2 |-14 – OLED - SA0 (no connection)
                                 |                           |
                              8- | RC6, CCP4              RB4 |-13 – OLED – SDA (blue wire)
                                 |                           |
                              9- | RC7                    RB5 |-12 – OLED – RES (brown wire)
                                 |                           |
                             10- | RB7                    RB6 |-11 – OLED – SCL (violet wire)
                                 |___________________________|
```

I have reserved the OLED-SA0 connection just in case you want to use a different communication interface for the OLED module in the future, such as SPI4. For the OLED with 4 rigid pins instead of 5 wires, you do not need to connect OLED-RES to pin 12 of the microcontroller. Leave this unconnected.

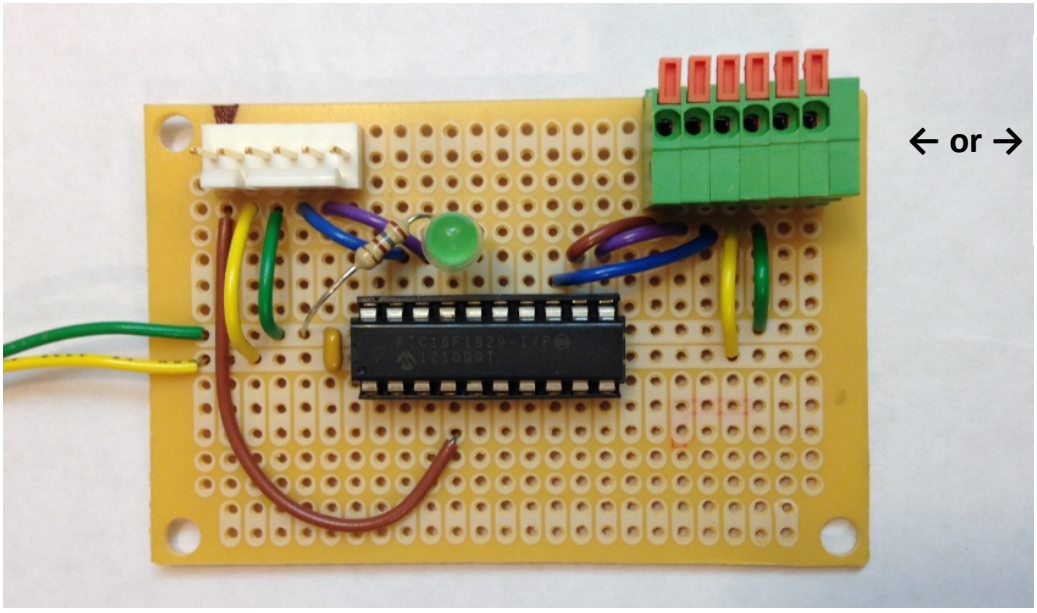
For I2C operation you only need the following 5 wires (or 4 pins) to be properly connected:

OLED module	**microcontroller**	**Function**
Green wire	ground on the PCB	ground
Yellow wire	+5V on the PCB	power
Blue wire	Pin #13 on the microcontroller (RB4)	data
Violet wire	Pin #11 on the microcontroller (RB6)	clock
Brown wire	Pin #12 on the microcontroller (RB5)	reset **(not used on 4-pin OLED)**

BME 580 Microcontrollers I

Step-by-step instructions for the OLED lab

1- Use EagleCAD to design a component for the OLED module with the adapter PCB, so your component will only have 5 connections (or 4), not all 16 on the original OLED display module.
2- Use EagleCAD to build a component for the 6-pin spring terminal or 4-pin header. The spring terminals are available from sparkfun.com https://www.sparkfun.com/products/8077
3- Use your new components to update your EagleCAD schematic of your circuit board.
4- Make sure to take screenshots and paste them into this lab template document (below).
5- Solder the 6-pin spring terminal onto your PCB as shown in the picture below:

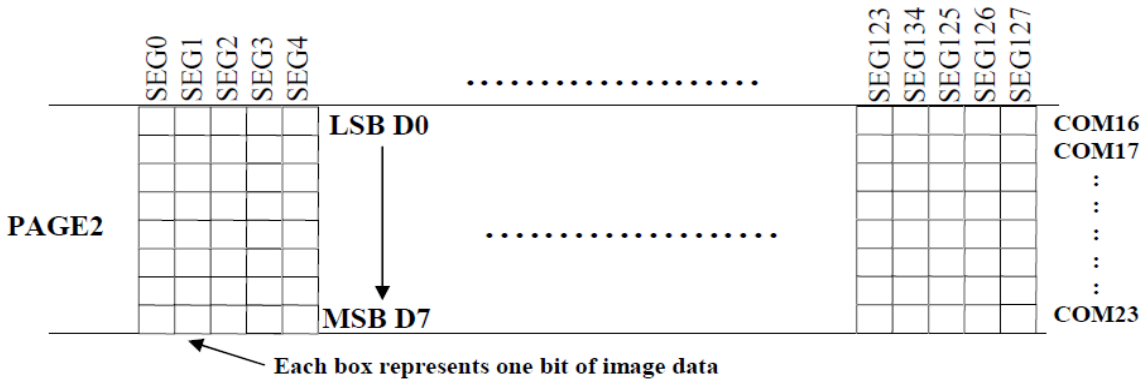


6- Connect the 5 or 4 interface wires (see previous page and above photo) to the terminal or header
7- Open a new source code project using the PICC compiler
8- Cut-and paste the source code from Appendix 1 into the project
9- Place the OLED header file () into the same folder as your source code
10- Compile the source code. There will be several warnings for unused variables, but no errors (hopefully). That is OK. These variables are used for graphics, you may use them later.
11- Locate the *.hex code that results from compiling your source code (should be in same folder)
12- Burn the *.hex code onto your microcontroller using the PICKit3 programmer and MPLab
13- Insert the 5 wires or 4 pins from your OLED display into the 6-pin spring terminal or 4-pin header
14- Test to make sure your OLED display functions properly. Be sure to apply only +5.0 volts
15- You will see a generic message that I have included in the source code. The “voltages” are just numbers, they do not represent actual voltages yet.
16- Once everything is working, change the source code to personalize it. Put your name on the display.
17- Study the OLED header file (oled_16F1829_i2c.h, open it as a text document). Find where I changed the “$” character to a much more useful “ ° ” character for displaying temperature.
18- Create your own oled_16F1829_i2c.h file: change one of the useless characters to a useful one (see the next page for a description of how to do this)
19- Fill in each location in this lab template where indicated, and email me the final copy for credit

BME 580 Microcontrollers I

About OLED Characters

It's not obvious how the OLED display generates characters, so here is some more information on it. Although different display manufacturers will have differences in the details of how this is done, basically each character is a bit map of an array of pixels. The OLED displays we use in this class use characters that are each 5 columns x 8 rows. You can see this by looking at one of the documents I sent you for this lab, SSD1306-1.2—OLED-DRIVER-DATASHEET.pdf On page 26 of the PDF document you will find this figure (Fig 8-14):

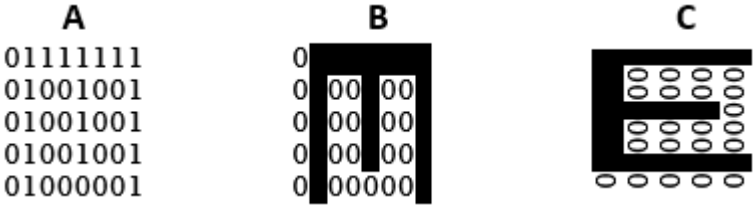


Look at the OLED header file I sent you (oled_16f1829_i2c.h), and scroll down to the character table. I'll just pick out a random character "E". The pixel map for the character "E" is defined in that file by this line:

0x7f,0x49,0x49,0x49,0x41, // 45 E

Normally to transmit the ASCII character "E", for example, using a serial protocol, you would only have to print its decimal ASCII code: 69 (hex code 0x45), but to drive the OLED, you have to print out a series of five hexadecimal numbers: 0x7f, 0x49, 0x49, 0x49, 0x41 Each bit in each hex number represents one pixel.

In binary format, lined up in rows, these five hexadecimal numbers look like figure **A** below
If you highlight the "1"s in black it looks like figure **B** below
and then if you rotate it 90° counter clockwise looks like figure **C**



You can do this process by hand, in reverse order, to create new characters in a 5 x 8 array of pixels, or you can create a spread sheet to make new character generation easy. Try experimenting with this.

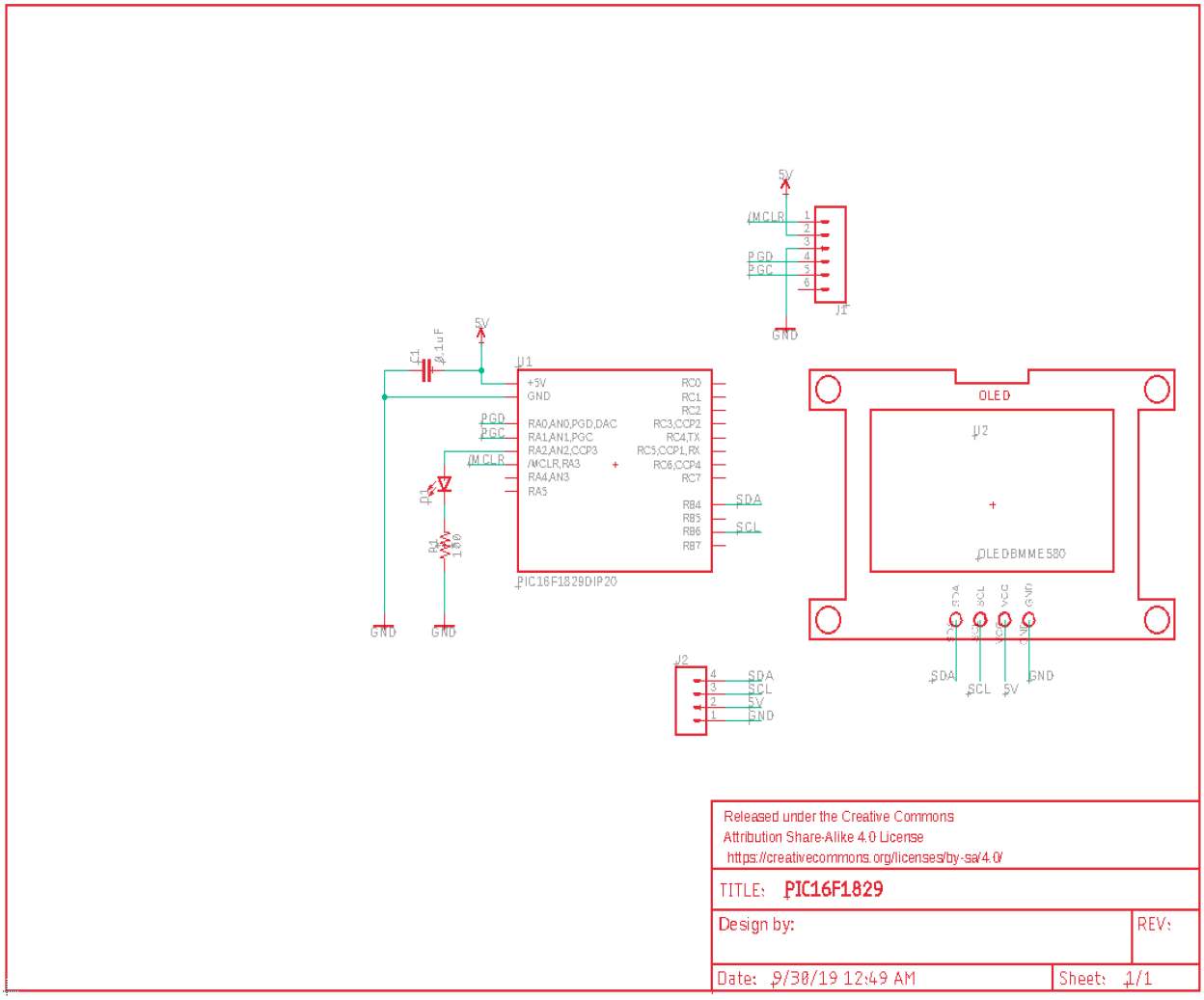
The OLED driver chip does the above steps A-B-C automatically every time you send a character using 5 hexadecimal numbers. You need to send all five hexadecimal numbers to the OLED display to generate one character, so you use the printf() function in your source code. Here is an example line of C code:

printf(oled_printchar,"BMME580");

The header file (oled_16f1829_i2c.h) that you included in your source code defines the function "oled_printchar" (way down toward the bottom of the header file), so that every time you use the function printf(oled_printchar, "*something*"); the microcontroller knows to send out the correct stream of five hexadecimal numbers for each character in the string *something*.

BME 580 Microcontrollers I

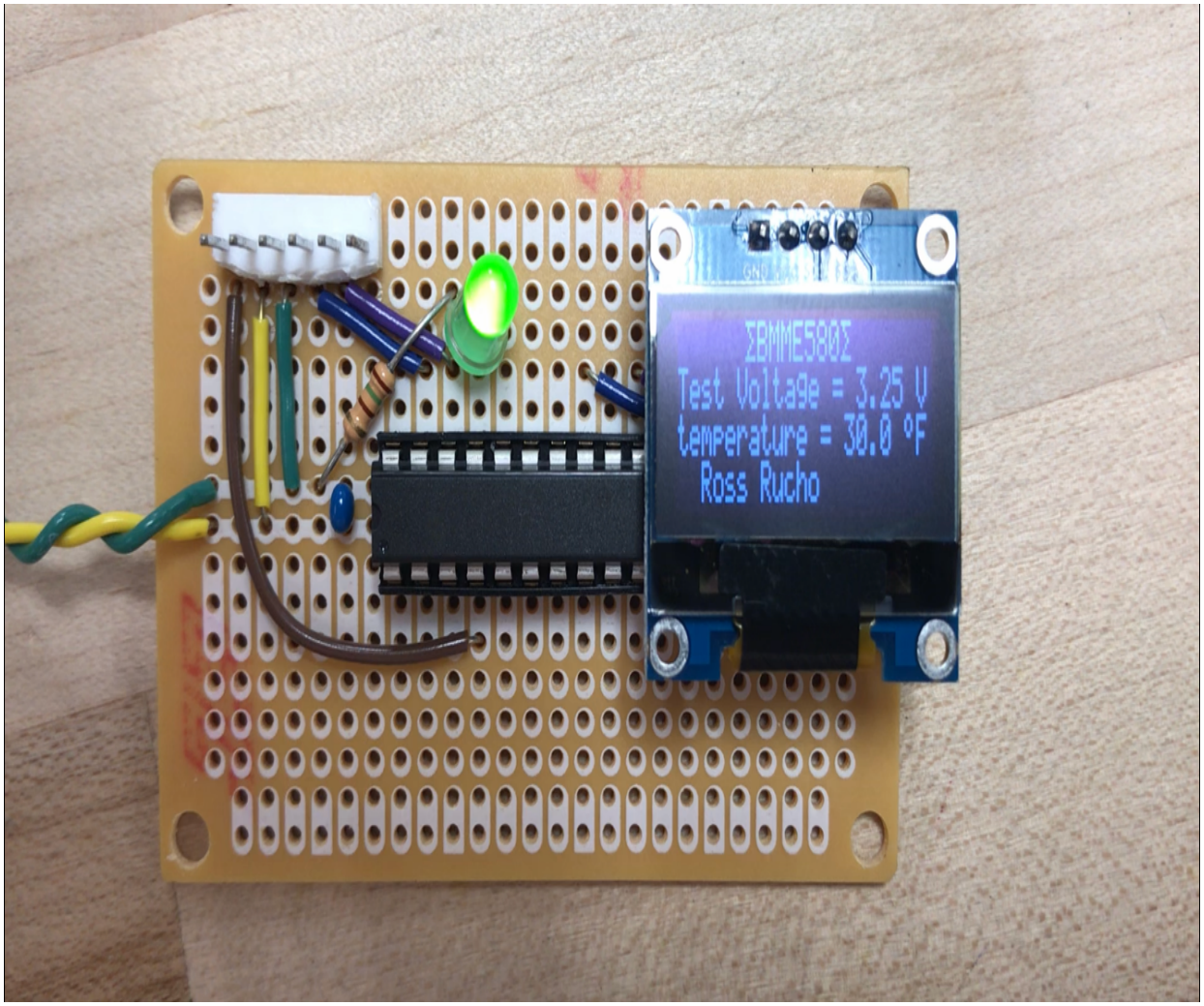
Your EagleCAD version of the circuit schematic:



BME 580 Microcontrollers I

BME 580 Microcontrollers I

Photograph of your completed circuit:



BME 580 Microcontrollers I

Paste a copy of your modified source code here

Include only the lines of code that you changed to display your name on the OLED:

Changed printf(oled_printchar,"moving text"); to printf(oled_printchar,"Ross Rucho");

Paste a copy of your modified OLED header file here (oled_16F1829_i2c.h)

Include only the line of code that you changed to display your new symbol:

Changes to header file:

// 0x02,0x01,0x51,0x09,0x06, // 3f ? // original pixels for "?" symbol

0x41,0x63,0x55,0x49,0x41, // 3f ?

// I have changed the display value for the ASCII symbol "?" to the summation symbol "Σ" Ross Rucho

Changes to source code:

Changed printf(oled_printchar,"BMME580"); to printf(oled_printchar,"?BMME580?");

BME 580 Microcontrollers I

APPENDIX 1: SOURCE CODE for Lab 3: OLED

```
//      oled_2015_a.c     RGD 9/1/2015
//
//
//
//
// Use PCM 14 bit compiler
// Circuit configuration:
// Internal Oscillator with I/O on RA6 and RA7, osc speed = 4 MHz, 1 us instruction cycle

// NOTE: the ASCII representation of the PIC16F1829 will look correct in the PICC text editor

//                    ^ = ENABLE INTERNAL PULL-UP RESISTOR
/*                                  PIC16F1829  20SOIC Pinout:
                                  _____________  _____________
                                 |        [___]         |
                              +++  1-| +5V Regulated        GND |-20 ---
                                 |                  |
                             nc <<<  2-| RA5       DAC,PGD,RA0,AN0^|-19 >>> PGD
                                 |                  |
                             nc <<<  3-| RA4,AN3      PGC,RA1,AN1^|-18 >>> PGC
                                 |                  |
                          /MCLR  >>>  4-| /MCLR,RA3        RA2,AN2 |-17 >>> GREEN LED OUT
                                 |                  |
                             nc <<<  5-| RC5              RC0 |-16 >>> nc
                                 |                  |
                             nc <<<  6-| RC4              RC1 |-15 >>> nc
                                 |                  |
                             nc <<<  7-| RC3              RC2 |-14 >>> OLED SA0 -- I2C address for OLED,set
output = HIGH (= 1) or IGNORE
                                 |                  |
                             nc <<<  8-| RC6              RB4 |-13 >>> OLED I2C SDA
                                 |                  |
                             nc <<<  9-| RC7              RB5 |-12 >>> OLED RES#
                                 |                  |
                             nc <<< 10-| RB7               RB6 |-11 >>> OLED I2C SCK
                                 |_____________________________|
```

OLED Module LM096-128064 (OLED = organic LED)

Pinout using the OLED adapter PCB:

BME 580 Microcontrollers I

green wire = ground on PCB - connect to ground on PCB
yellow wire = +5V on PCB - connect to +5volts on PCB
blue wire = OLED SDA (serial data) - connect to microcontroller pin 13 (RB4)
violet wire = OLED SCK (serial clock) - connect to microcontroller pin 11 (RB6)
brown wire = OLED RES# (reset) - connect to microcontroller pin 12 (RB5)

headers: USE SPRING PIN TERMINAL ON pcb

OLD configuration:

Pinout and connections for the ORIGINAL OLED MODULE
Pin # - Connection
1 - +5 volts
2 - Ground
3 - Ground
4 - RB5 (OLED RES#)
5 - RC2 (OLED SA0)
6 - Ground
7 - Ground
8 - RB6 (OLED I2C SCL)
9 - RB4 (OLED I2C SDA)
10 - no connection (nc)
11 - Ground
12 - Ground
13 - Ground
14 - Ground
15 - Ground
16 - no connection (nc)

HEADERS: (use double row 16 pin headers: 2 x 8 pins)
use the MALE 16-pin header on the OLED display with the pins pointing downward, on the back of the OLED display panel
Place the FEMALE header on your PC board and connect to your uC as shown above, and be sure to mark Pin #1

*/

```
#include <16F1829.H>
// #include <stdlib.h>


//#device adc=8
```

BME 580 Microcontrollers I

```
#use delay (clock=8000000)    //  8 MHz, 0.5 us instruction cycle, this MUST come before the #include
<oled_16F1829_i2c.h> to define delays in header file such as "delay_us(1)"
//   #use fast_io (A)        //

#include <oled_16F1829_i2c.h>

#fuses WDT, INTRC_IO, PUT     // internal oscillator operation with RA6 and RA7 as digital I/O, ref
16F1829.h, enable Power Up Timer
#fuses NOMCLR               // disable /MCLR pin
#fuses NOPROTECT             // disable code protection
// #fuses NOLVP

// Bob's pin definitions for PIC16F1829 for I2C communication with OLED are in oled_16F1829_i2c.h

#define led pin_a2         // BE SURE TO REDEFINE THE LED PIN TO MATCH YOUR CIRCUIT

unsigned int i = 0;
unsigned int j = 0;

float t = 0;              // floating point variable for "temperature"
float v = 0;              // floating point variable for "voltage"


////////////////////////////////////
// SUBROUTINES

void init_ports(void) {

  SETUP_OSCILLATOR(OSC_8MHZ|OSC_INTRC);
  SETUP_WDT(WDT_8S);                //  WDT set to 8 second period

}



////////////////////////////////////
// PIC12F1829 goes here at RESET

void main() {

  init_ports();              //  Initialize ports
  restart_wdt();
  delay_ms(50);              //  insert a short delay before starting the system up
```

BME 580 Microcontrollers I

```
//  the "bit bang" code below and in the *.h file to emulate I2C
//  to run the OLED Display was developed by Kenny Donnelly

output_low(res);
  delay_ms(10);
  output_high(res);
  delay_ms(10);

initialise_screen();
  delay_ms(10);
  clear_screen();
  delay_ms(10);
  fill_screen();
  delay_ms(10);
  clear_screen();
  delay_ms(10);


CYCLE:

  i = 0;            //   counter for testing "moving text"
  clear_screen();     //   clear OLED screen

  while(true){
    restart_wdt();

 oled_write_command(0xb0);
 oled_write_command(0x00);
 oled_write_command(0x10);

//  uncomment the following line to zoom in to text (top four lines of text only)
  oled_zoom();


//  the following command places the next character on line 0 (the 8 text lines are numbered 0 to 7)
//  and column 42 (there are 128 pixels from left to right on this 128 x 64 pixel display)
//  note that text character placement on the OLED display is therefore:
//      128 columns (individual pixels) numbered 0 to 127
//       8 lines, numbered 0 to 7, where each line is one character tall
//         each character is 8 pixels tall
//         8 x 8 = 64 pixels = the full display height
//  By placing the next character at this location the text "BMME580" is centered at the top
```

BME 580 Microcontrollers I

```
//  question: how many characters fit into the 128 characters across the OLED display
  oled_gotoxy(0,42);
 printf(oled_printchar,"BMME580");


   oled_gotoxy(1,0);
 printf(oled_printchar,"Test Voltage = %2.2f ", v);   //  see the PIC_C User Manual under "printf()" to
format numbers in the display

 printf(oled_printchar,"V");     // notice how the "V" displays directly after the last printed character

    oled_gotoxy(2,0);
 printf(oled_printchar,"temperature = %1.1f $F", t);
 //  look in the oled_16F1829_i2c.h file to see why the "$" character actually
 //  displays a the temperature degree symbol "°"
 //  answer: I "remapped" the pixels in the BYTE const TABLE5 for "$".
 //  So I changed the ASCII Character Table for the OLED Display in the *.h file

   oled_gotoxy(3,0);

   for (j=0; j<i; j++) printf(oled_printchar," ");


 printf(oled_printchar,"moving text");


     OUTPUT_HIGH(LED);   //  RA5 - turn ON green LED
       delay_ms(100);
     OUTPUT_LOW(LED);    //  RA5 - turn OFF green LED
       delay_ms(100);

  t = t + 1.2;      //  increment the "temperature" variable
  v = v + .13;      //  increase the "voltage" variable
  i = i + 1;        //  nudge test to the right one character
  if(i>22) {
     i = 0;   // prevent moving text from scrolling down to next row on OLED display
     clear_screen();      // clear OLED screen
  }
 }
     restart_wdt();

goto CYCLE;  //  cycle indefinitely until battery is removed or battery runs out of power
} // main
```

BME 580 Microcontrollers I

APPENDIX 2: HEADER FILE for Lab 3: OLED (using a bit-bang I2C interface)

```
// Bob's pin definitions for PIC16F1829 for I2C communication:
#define res pin_b5
#define sa0 pin_c2=1
#define scl pin_b6
#define sda pin_b4



/*  Kennys pin assignments for PIC18F4550 for I2C communication:
#define res pin_b3
#define sa0 pin_b2=1
#define scl pin_b1
#define sda pin_b0
*/


// Number of pixels in LCD
#define LCD_X     128
#define LCD_Y     64


int char_row,charsel,charpos,chardata;       // for nokia_5110 lcd

int16 ddram;

//char plot_value;

int32 plot_value32;

int32 plot_umsb,plot_lmsb,plot_ulsb,plot_llsb;

// ASCII table for NOKIA LCD and OLED: 96 rows * 5 bytes= 480 bytes

BYTE const TABLE5[240]=          {0x00,0x00,0x00,0x00,0x00,     //  20 space

  0x00,0x00,0x5f,0x00,0x00,   // 21 !
  0x00,0x07,0x00,0x07,0x00,   // 22 "
  0x14,0x7f,0x14,0x7f,0x14,   // 23 #
//  0x24,0x2a,0x7f,0x2a,0x12,   // 24 $   // original pixels for "$" symbol
  0x06,0x09,0x09,0x06,0x00,   // 24 $
```

BME 580 Microcontrollers I

```
// I have changed the display value for the ASCII symbol "$" to the degree symbol "°"  RG Dennis
  0x23,0x13,0x08,0x64,0x62,   // 25 %
  0x36,0x49,0x55,0x22,0x50,   // 26 &
  0x00,0x05,0x03,0x00,0x00,   // 27 '
  0x00,0x1c,0x22,0x41,0x00,   // 28 (
  0x00,0x41,0x22,0x1c,0x00,   // 29 )
  0x14,0x08,0x3e,0x08,0x14,   // 2a *
  0x08,0x08,0x3e,0x08,0x08,   // 2b +
  0x00,0x50,0x30,0x00,0x00,   // 2c ,
  0x08,0x08,0x08,0x08,0x08,   // 2d -
  0x00,0x60,0x60,0x00,0x00,   // 2e .
  0x20,0x10,0x08,0x04,0x02,   // 2f /
  0x3e,0x51,0x49,0x45,0x3e,   // 30 0
  0x00,0x42,0x7f,0x40,0x00,   // 31 1
  0x42,0x61,0x51,0x49,0x46,   // 32 2
  0x21,0x41,0x45,0x4b,0x31,   // 33 3
  0x18,0x14,0x12,0x7f,0x10,   // 34 4
  0x27,0x45,0x45,0x45,0x39,   // 35 5
  0x3c,0x4a,0x49,0x49,0x30,   // 36 6
  0x01,0x71,0x09,0x05,0x03,   // 37 7
  0x36,0x49,0x49,0x49,0x36,   // 38 8
  0x06,0x49,0x49,0x29,0x1e,   // 39 9
  0x00,0x36,0x36,0x00,0x00,   // 3a :
  0x00,0x56,0x36,0x00,0x00,   // 3b ;
  0x08,0x14,0x22,0x41,0x00,   // 3c <
  0x14,0x14,0x14,0x14,0x14,   // 3d =
  0x00,0x41,0x22,0x14,0x08,   // 3e >
  0x02,0x01,0x51,0x09,0x06,   // 3f ?
  0x32,0x49,0x79,0x41,0x3e,   // 40 @
  0x7e,0x11,0x11,0x11,0x7e,   // 41 A
  0x7f,0x49,0x49,0x49,0x36,   // 42 B
  0x3e,0x41,0x41,0x41,0x22,   // 43 C
  0x7f,0x41,0x41,0x22,0x1c,   // 44 D
  0x7f,0x49,0x49,0x49,0x41,   // 45 E
  0x7f,0x09,0x09,0x09,0x01,   // 46 F
  0x3e,0x41,0x49,0x49,0x7a,   // 47 G
  0x7f,0x08,0x08,0x08,0x7f,   // 48 H
  0x00,0x41,0x7f,0x41,0x00,   // 49 I
  0x20,0x40,0x41,0x3f,0x01,   // 4a J
  0x7f,0x08,0x14,0x22,0x41,   // 4b K
  0x7f,0x40,0x40,0x40,0x40,   // 4c L
  0x7f,0x02,0x0c,0x02,0x7f,   // 4d M
  0x7f,0x04,0x08,0x10,0x7f,   // 4e N
  0x3e,0x41,0x41,0x41,0x3e    // 4f O
```

```
};


BYTE const TABLE6[240]=          {0x7f,0x09,0x09,0x09,0x06,   // 50 P
 0x3e,0x41,0x51,0x21,0x5e,   // 51 Q
 0x7f,0x09,0x19,0x29,0x46,   // 52 R
 0x46,0x49,0x49,0x49,0x31,   // 53 S
 0x01,0x01,0x7f,0x01,0x01,   // 54 T
 0x3f,0x40,0x40,0x40,0x3f,   // 55 U
 0x1f,0x20,0x40,0x20,0x1f,   // 56 V
 0x3f,0x40,0x38,0x40,0x3f,   // 57 W
 0x63,0x14,0x08,0x14,0x63,   // 58 X
 0x07,0x08,0x70,0x08,0x07,   // 59 Y
 0x61,0x51,0x49,0x45,0x43,   // 5a Z
 0x00,0x7f,0x41,0x41,0x00,   // 5b [
 0x02,0x04,0x08,0x10,0x20,   // 5c
 0x00,0x41,0x41,0x7f,0x00,   // 5d
 0x04,0x02,0x01,0x02,0x04,   // 5e
 0x40,0x40,0x40,0x40,0x40,   // 5f
 0x00,0x01,0x02,0x04,0x00,   // 60
 0x20,0x54,0x54,0x54,0x78,   // 61 a
 0x7f,0x48,0x44,0x44,0x38,   // 62 b
 0x38,0x44,0x44,0x44,0x20,   // 63 c
 0x38,0x44,0x44,0x48,0x7f,   // 64 d
 0x38,0x54,0x54,0x54,0x18,   // 65 e
 0x08,0x7e,0x09,0x01,0x02,   // 66 f
 0x0c,0x52,0x52,0x52,0x3e,   // 67 g
 0x7f,0x08,0x04,0x04,0x78,   // 68 h
 0x00,0x44,0x7d,0x40,0x00,   // 69 i
 0x20,0x40,0x44,0x3d,0x00,   // 6a j
 0x7f,0x10,0x28,0x44,0x00,   // 6b k
 0x00,0x41,0x7f,0x40,0x00,   // 6c l
 0x7c,0x04,0x18,0x04,0x78,   // 6d m
 0x7c,0x08,0x04,0x04,0x78,   // 6e n
 0x38,0x44,0x44,0x44,0x38,   // 6f o
 0x7c,0x14,0x14,0x14,0x08,   // 70 p
 0x08,0x14,0x14,0x18,0x7c,   // 71 q
 0x7c,0x08,0x04,0x04,0x08,   // 72 r
 0x48,0x54,0x54,0x54,0x20,   // 73 s
 0x04,0x3f,0x44,0x40,0x20,   // 74 t
 0x3c,0x40,0x40,0x20,0x7c,   // 75 u
 0x1c,0x20,0x40,0x20,0x1c,   // 76 v
 0x3c,0x40,0x30,0x40,0x3c,   // 77 w
 0x44,0x28,0x10,0x28,0x44,   // 78 x
```

```
 0x0c,0x50,0x50,0x50,0x3c,   // 79 y
 0x44,0x64,0x54,0x4c,0x44,   // 7a z
 0x00,0x08,0x36,0x41,0x00,   // 7b
 0x00,0x00,0x7f,0x00,0x00,   // 7c
 0x00,0x41,0x36,0x08,0x00,   // 7d
 0x10,0x08,0x08,0x10,0x08,   // 7e
 0x78,0x46,0x41,0x46,0x78    // 7f
};


void iic_start(){
output_high(scl);
delay_us(1);
output_high(sda);
delay_us(1);
output_low(sda);
delay_us(1);
output_low(scl);
delay_us(1);
}

void iic_stop(){
output_low(scl);
delay_us(1);
output_low(sda);
delay_us(1);
output_high(scl);
delay_us(1);
output_high(sda);
delay_us(1);
}

void iic_write(byte iic_byte){
int i;
for (i=0;i<8;i++) {
output_bit(sda, shift_left(&iic_byte,1,0));
output_high(scl) ;
delay_us(1);
output_low(scl) ;
delay_us(1);
}
output_high(sda);
output_high(scl);
delay_us(1);
```

```
output_low(scl);
delay_us(1);
}

void oled_write_data(byte iic_data){
iic_start();
iic_write(0x78);
iic_write(0x40);
iic_write(iic_data);
iic_stop();
}

void oled_write_command(byte iic_command){
iic_start();
iic_write(0x78);
iic_write(0x00);
iic_write(iic_command);
iic_stop();
}

void initialise_screen(){
oled_write_command(0xae);
oled_write_command(0x20);
oled_write_command(0x10);
oled_write_command(0xb0);
oled_write_command(0xc8);
oled_write_command(0x00);
oled_write_command(0x10);
oled_write_command(0x40);
oled_write_command(0x81);
oled_write_command(0xaf);
oled_write_command(0xa1);
oled_write_command(0xa6);
oled_write_command(0xa8);
oled_write_command(0x3f);
oled_write_command(0xa4);
oled_write_command(0xd3);
oled_write_command(0x00);
oled_write_command(0xd5);
oled_write_command(0xf0);
oled_write_command(0xd9);
oled_write_command(0x22);
oled_write_command(0xda);
oled_write_command(0x12);
```

```
oled_write_command(0xdb);
oled_write_command(0x20);
oled_write_command(0x8d);
oled_write_command(0x14);
oled_write_command(0xaf);
}

void table_to_oled(void){
if (charsel<0x20)return;
 if (charsel>0x7f)return;

 for (char_row=0;char_row<5;char_row++) {     // 5 bytes

   if (charsel<0x50){charpos=(((charsel&0xff)-0x20)*5);chardata=TABLE5[(charpos+char_row)];}          //
use TABLE5
   else if (charsel>0x4f){charpos=(((charsel&0xff)-0x50)*5);chardata=TABLE6[(charpos+char_row)];}
// use TABLE6


   oled_write_data(chardata);     // send data to oled
 }

 oled_write_data(0x00);     //    1 byte (always blank)

}

void oled_printchar(int cvar){
charsel=cvar;
table_to_oled();
}



void clear_screen(void){
int m,n;
for(m=0;m<8;m++){
oled_write_command(0xb0+m);
oled_write_command(0x00);
oled_write_command(0x10);
for(n=0;n<128;n++) {
oled_write_data(0x00);
}
}
}
```

```
void fill_screen(void){
int m,n;
for(m=0;m<8;m++){
oled_write_command(0xb0+m);
oled_write_command(0x00);
oled_write_command(0x10);
for(n=0;n<128;n++) {
oled_write_data(0xff);
}
}
}

void oled_gotoxy(int page, int column){
byte column_lower=0x00;
byte column_upper=0x10;
byte page_start=0xb0;
page_start=page_start+page;
column_upper=(column_upper+column/16);
column_lower=(column_lower+column%16);
oled_write_command(page_start);
oled_write_command(column_lower);
oled_write_command(column_upper);
}

void oled_zoom(void){
//  These four lines (two 2-byte commands) set ZOOM IN mode (characters are twice as high)
//  see SSD1306-1.2 user manual for OLED driver, pages 33, 34, and 49
  oled_write_command(0xda);
  oled_write_command(0x12);
  oled_write_command(0xd6);
  oled_write_command(0x01);
}

void vertical_line(int length){

}

void oled_drawbox(int width, int height){

}
```