Implementation of LCD Panel in a Traffic Light Controller

We will use the traffic light control system designed on "Traffic Light Controller with use of System Timer " lab that has the following hardware:

1) 4 sets of RGB LEDs with the following assignments:
    a. 1 RGB LED for North/South direction
    b. 1 RGB LED for North/South Left Turn direction
    c. 1 RGB LED for East/West direction
    d. 1 RGB LED for East/West Left Turn direction

2) Four DIP switches used as sensors with the following designations:

    a. NS Switch Pedestrian (NSPED_SW) switch acting to indicate pedestrian present in the North/South direction
    b. NS Left Turn (NSLT_SW) switch acting as a sensor for cars making left turn on the North/South direction
    c. EW Switch Pedestrian (EWPED_SW) switch acting to indicate pedestrian present in the East/West direction
    d. EW Left Turn (EWLT_SW) switch acting as a sensor for cars making left turn on the East/West direction

3) Light Sensor to define the mode of operation. There will be two modes: Day Mode and Night Mode.

4) A Mode LED to indicate the mode the traffic controller is running (0 for Night and 1 for Day)

5) A SEC_LED to show 1 second pulse

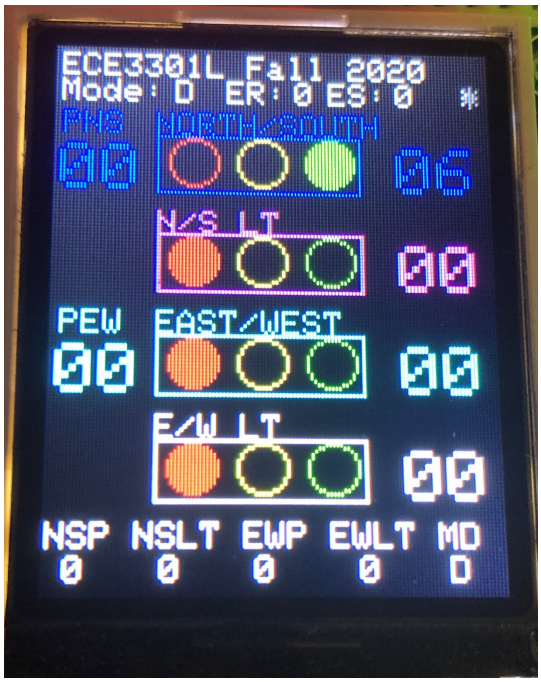
6) Two 7-segment LEDs used to show the number of seconds for pedestrian crossings for each direction

7) A Buzzer circuit

From that circuit, we are going to modify to a new circuit that will have the following:

1) Instead of the two 7-segment displays, a LCD TFT Display is used to show the operation of the controller. A typical screen shot is shown below:

2) Some of the RGB LEDs will have new connections but their functions remain the same



Here are some descriptions on the information on the above TFT screen:

1) First Line: Name of this class and its semester

2) Mode: Actual mode the controller is running – A 'D' will indicate the day mode while a 'N' will specify the night mode.

3) ER and ES fields: These will be defined on Lab #9

4) 'NORTH/SOUTH': The box below this line shows the color of the traffic light in the North/South direction. There are three circles below this box each with the color Red, Yellow, and Green from left to right. When the circle is not filled, this means that the light is off. When it is filled, the corresponding color is on.

5) The two '00' digits at the right of the 'NORTH/SOUTH' are used to display the number of seconds left on this direction when the lights are turned into the green and yellow colors.

6) There are three other labels 'N/S LT', 'EAST/WEST', 'E/W LT'. They have the same descriptions as for the North/South direction but each item is for a different direction.

7) The label 'PNS' and the number right below it show the actual number of seconds left on the Pedestrian Counter in the North/South direction.

8) The label ‘PEW’ and the number right below it show the actual number of seconds left on the Pedestrian Counter in the East/West direction.

9) The line at the bottom shows the following: ’NSP NSLT EWP EWLT MD’. The numbers below that line show the status of the switches for the North/South PED, NSLT sensor, East/West PED, EWLT sensor and the logic state of the light sensor for MODE (This is the instant sensing of the light sensor and not the actual mode of operation).

The operation of the traffic light was already implemented on Lab #7. The purpose of this lab is to display additional information on the TFT panel and to use the push-button switches to cause edge interrupts to the controller activating the request for pedestrian access associated with the direction of the switches.

The schematic of the design is shown on the attached file.

The TFT Panel uses the hardware interface called SPI Bus in order for the microcontroller to control the panel to display data on its screen. The basic understanding of the SPI bus will be visited on a later lab and the hardware explanation will be handled at that time.

In addition, since it would take a good amount of time to implement all the software routines in order to allow the user to display texts or to generate different types of shapes, a library software has been compiled and put together into the file "ST7735_TFT.c". This file should be included at the beginning of the ‘.c’ program to provide all library functions used in the program.

To develop the basic screen shown on the above screenshot, it will take a good amount of time to do so. We will skip this step. A basic routine is provided to the student that will create the basic framework of the screen. The student should look at the provided routine ‘Initialize_TFT()’ to get the basic ideas to implement the rest of the requirements of this lab.

Here are the definitions for the various values that the variable ‘direction’ can have:

```
#define NS      0       // Number definition of North/South
#define NSLT    1       // Number definition of North/South Left Turn
#define EW      2       // Number definition of East/West
#define EWLT    3       // Number definition of East/West Left Turn
```

Here are the definitions for the various values that the variable ‘color’ can have:

```
#define Color_Off     0     // Number definition of Off Color
#define Color_Red     1     // Number definition of Red Color
#define Color_Green   2     // Number definition of Green Color
#define Color_Yellow  3     // Number definition of Yellow Color
```

We will need to implement the codes for the following functions:

A) void update_color(char direction, char color)

This routine will change the color of the traffic light on the TFT panel for the specified ‘direction’ with the specified ‘Color’. Fox example, if this function is called with direction=NS and Color=Color_Red, then the RED light in the North/South field should be filled. The other two colors (Yellow and Green) should have only its outline drawn but inside the circle should not be filled.

.

Here is a portion of that routine providing a starting example:

```
void update_color(char direction, char color)
{
   char Circle_Y;
   Circle_Y = NS_Cir_Y + direction * 30;

   if (color == Color_Off)     //if Color off make all circles black but leave outline
   {
         fillCircle(XRED, Circle_Y, Circle_Size, ST7735_BLACK);
         fillCircle(XYEL, Circle_Y, Circle_Size, ST7735_BLACK);
         fillCircle(XGRN, Circle_Y, Circle_Size, ST7735_BLACK);
         drawCircle(XRED, Circle_Y, Circle_Size, ST7735_RED);
         drawCircle(XYEL, Circle_Y, Circle_Size, ST7735_YELLOW);
         drawCircle(XGRN, Circle_Y, Circle_Size, ST7735_GREEN);
   }

   if (color == Color_Red)     //if the color is red only fill the red circle with red
   {
         fillCircle(XRED, Circle_Y, Circle_Size, ST7735_RED);
         fillCircle(XYEL, Circle_Y, Circle_Size, ST7735_BLACK);
         fillCircle(XGRN, Circle_Y, Circle_Size, ST7735_BLACK);
         drawCircle(XRED, Circle_Y, Circle_Size, ST7735_RED);
         drawCircle(XYEL, Circle_Y, Circle_Size, ST7735_YELLOW);
         drawCircle(XGRN, Circle_Y, Circle_Size, ST7735_GREEN);
   }

  // Add code for the remaining colors
}
```

B) void update_count(char direction, char count)

This routine will change the value of the Second Counter (right side of the traffic light) for the specified ’direction’ with the value indicated by ‘count’.

Below is a beginning example of the routine:

```
void update_count(char direction, char count)
{
  switch (direction)                  //update traffic light no ped time
  {
    case NS:
      NS_Count[0] = count/10  + '0';
      NS_Count[1] = count%10  + '0';
      drawtext(XCNT, NS_Count_Y, NS_Count, NS_Color, ST7735_BLACK, TS_2);
      break;
  }
        // Add code for the remaining directions

}
```

C) void update_PED_Count(char direction, char count)

This routine will change the value of the Pedestrian Counter (left side of the traffic light) for the specified 'direction' with the value indicated by 'count'.

Below is a beginning example of the routine:

```
void update_PED_Count(char direction, char count)
{
  switch (direction)
  {
    case NS:
      PED_NS_Count[0] = count/10  + '0';            // PED count upper digit
      PED_NS_Count[1] = count%10  + '0';          // PED Lower
      drawtext(PED_Count_X, PED_NS_Count_Y, PED_NS_Count, NS_Color,
ST7735_BLACK, TS_2);        //Put counter digit on screen
      break;
   }

        // Add code for the remaining direction

}
```

D) void update_misc()

This routine will update the various states of the switches and the photo sensor. Below is an example of that routine:

```
void update_misc()
{
    char  ch = 0;
    ADCON0 = ch*4+1;                    // channel AN0
   int nStep = get_full_ADC();          // calculates the # of steps for analog conversion
   volt = nStep * 5 /1024.0;            // gets the voltage in Volts, using 5V as reference

   SW_MODE = volt < 3.3 ? 1:0;       // SW_MODE = 1, Day_mode, SW_MODE = 0 Night
   SW_NSPED = NS_PED_SW;
   SW_NSLT = NS_LT_SW;
   SW_EWPED = EW_PED_SW;
   SW_EWLT = EW_LT_SW;


   if (SW_MODE == 0) SW_MODE_Txt[0]= 'N'; else SW_NSPED_Txt[0] = 'D';

  //  put code here to fill the fields for:
  //    SW_NSPED_Txt[0]  <- use '0' if the switch is 0 and '1' is the switch is 1
  //     SW_NSLT_Txt[0]
  //     SW_EWPED_Txt[0]
  //     SW_EWLT_Txt[0]

   drawtext(35,10, Act_Mode_Txt, ST7735_WHITE, ST7735_BLACK, TS_1);
   drawtext(6,  Switch_Txt_Y+9, SW_NSPED_Txt, ST7735_WHITE, ST7735_BLACK, TS_1);
drawtext(32,  Switch_Txt_Y+9, SW_NSLT_Txt, ST7735_WHITE, ST7735_BLACK, TS_1);
    drawtext(58,  Switch_Txt_Y+9, SW_EWPED_Txt, ST7735_WHITE, ST7735_BLACK, TS_1);
   drawtext(87,  Switch_Txt_Y+9, SW_EWLT_Txt, ST7735_WHITE, ST7735_BLACK, TS_1);
   drawtext(112,  Switch_Txt_Y+9, SW_MODE_Txt, ST7735_WHITE, ST7735_BLACK, TS_1);

   }
```

E) When complete doing the creations of the new routines, perform the following modifications:

a. In the 'void Set_NS(char color)' routine, add at the beginning of the routine the following lines:

```
direction = NS;
update_color(direction, color);
```

b. repeat the same modifications for the other routines Set_NSLT(), Set_EW() and Set_EWLT()

c. In the 'PED_Control(char direction, char count)' routine, remove all the instructions to output to the PORTC and PORTD and replace them with the use of the newly created routine 'void update_PED_Count(char direction, char count)'. Note, the value of the direction for NS is now 0 while for EW it becomes 2 and not 1 as in lab #7.

d. In the 'void Wait_N_Seconds (char seconds)' routine, replace with the following:

```
void Wait_N_Seconds (char seconds)
{
   char I;
   for (I = seconds; I> 0; I--)
   {
      Wait_One_Second();          // calls Wait_One_Second for x number of times
      update_count(direction, I);
   }
   Wait_One_Second();             // calls Wait_One_Second for x number of times
   update_count(direction, 0);
}
```