Embedded Systems Interfacing Assignment 02

Due: 12 September 2012

Problem 01:Bargraph

You will use 8 LEDs to display an integer between 0 and 256. Create two functions, one to initialize the bargraph, and one that takes a value and displays it to the bargraph.

initBargraph

Write a routine called *initBargraph* that takes no arguments and returns no values. The declaration of this function should be:

void initBargraph(void);

The function should follow the steps given in the sequence below:

1. Clear $PORTA$.
2. Set $PORTA < 6 : 0 >$ to output.

Why is $RA7$ not also set to output?

setBargraph

Write a routine called *setBargraph* that takes one integer argument and returns no values. The declaration of this function should be

void setBargraph(unsigned int display);

where *display* is the integer that will be displayed on the bargraph.The function should follow the steps given in the sequence below:

1. Set TRISAbits.TRISA7 to output.
2. Transfer *dislpay* to the bargraph.

Hint: To transfer *display* to the bargraph, set the appropriate register to $0x00FF\&display$.

Problem 02: Buttons

initButtons

Explain the following routine

```
void initButtons(unsigned int mask)
{
    if(mask&0x0008)
        TRISDbits.TRISD6=1;
    if(mask&0x0004)
        TRISDbits.TRISD7=1;
    if(mask&0x0002)
        TRISAbits.TRISA7=1;
    if(mask&0x0001)
        TRISDbits.TRISD13=1;
}
```

getButton

Explain the following code:

```
unsigned int getButton(unsigned int mask)
{
    unsigned int button;
    switch(mask)
    {
        case 0x0008:
            button=!PORTDbits.RD6;
            break;
        case 0x0004:
            button=!PORTDbits.RD7;
            break;
        case 0x0002:
            TRISAbits.TRISA7=1;
            button=!PORTAbits.RA7;
            break;
        case 0x0001:
            button=!PORTDbits.RD13;
            break;
        default:
        button=0;
    }
    return (button);
}
```

Problem 03: Counter

Write a program that outputs a counter to the bargraph that changes about once every 15 seconds. The counter will do the following:

- If $S3$ is pressed, the counter will reset.
- If $S4$ is pressed, the counter will decrement.
- If no button is pressed, the counter will increment.

Use the functions that you wrote and were given in the previous two problems to modularize your code.

Problem 04: A/D Conversion

Configuration

Find the value to which $AD1CON1$ should be configured using the following information.

1. A/D Converter module is disabled (de-energized)
2. Continue module operation in idle mode
3. Data output format is integer (0000 00dd dddd dddd)
4. Internal counter ends sampling and starts conversion (auto convert)
5. Sampling begins when SAMP bit is set.

Find the value to which $AD1CON2$ should be configured using the following information.

1. $VR+ = AVDD$ and $VR- = AVSS$
2. Do not scan inputs
3. Buffer configured as one 16-word buffer ($ADC1BUF0$ to $ADC1BUFF$)
4. Always uses MUX A input Multiplexer settings.

Find the value to which $AD1CON3$ should be configured using the following information.

1. Clock derived from system clock
2. Auto-sampling time set to 31 TAD ($31 * 10 * 125ns = 387.5\mu s$)
3. A/D conversion clock set to 5 $TCYC$ or 10 $TOSC$ ($10 * 125ns = 1250ns$)

initADC

Write a routine called initADC that takes one augments and returns no data. The declaration of this function should be:

void initADC(unsigned int initChannel);

where *initiChannel* specifies which channel is selected for sampling. The function should follow the steps given in the sequence below:

1. Initialize $AD1CON1$ with ADC de-energized using the above values.
2. Initialize $AD1CON2$ with ADC using the above values
3. Initialize $AD1CON3$ with ADC using the above values.
4. Select initial channel using initChannel
5. Ignore all scan select channels
6. Set AN3, AN4 and AN5 as analog inputs
7. Enable A/D converter (energize).

getADC

Write a routine called *getADC* that takes one augments and returns one data. The declaration of this function should be:

unsigned int getADC(unsigned int channel);

where *channel* specifies which channel is selected for sampling and the returned value is the converted voltage. The function should follow the steps given in the sequence below:

1. Select channel using input argument.
2. Start sample by setting $AD1CON1bits.SAMP$.
3. Wait for $AD1CON1bits.DONE$ to be set.
4. Clear $AD1CON1bits.DONE$
5. Return the value $ADC1BUF0$.

Main

Write a main function that initializes the ADC using $initADC(4)$ and in the endless loop gets the ADC results using $getADC()$. Dont forget to add $\_CONFIG1$ and $\_CONFIG2$ macros.

Compile and debug this code. Run a MPLAB simulation with register injection for $ADC1BUF0$ from a file called ADC.txt. Place the hex values of 0042, 0063, and others in ADC.txt. Verify that the code and simulations work by watching the value returned by getADC function.

Problem 05: Temperature Sensor

Background

The temperature sensor on the Explore-16 Development Board is the TC1047A. In the units of volts VOUT for the TC1047A is given by

$$V_{OUT} = 0.01T + 0.5 \tag{1}$$

where T is the temperature in $^{\circ}C$. Therefore the temperature in $^{\circ}C$ is given by

$$T = 100V_{OUT} - 50 \tag{2}$$

The scaled voltage derived from the PIC25FJ128GA010s ADC is given by

$$V_{SCALED} = \frac{3.3N}{1024}, \tag{3}$$

where N is the value obtained from the ADC. Therefore the temperature in $^{\circ}C$ as a function of the ADC value is given by

$$T = \frac{330N - 51200}{1024}. \tag{4}$$

Sometimes the temperature in degrees Fahrenheit, $^{\circ}F$, is desired. The conversion equation is

$$F = \frac{9}{5}T + 32. \tag{5}$$

where T is the temperature in $^{\circ}C$ and F is the temperature in $^{\circ}F$.

Preliminary

Create a spreadsheet with the following 6 columns:

1. N as whole number from 32 to 312
2. Voltage output of TC1047A to nearest mV
3. Temperature in $^{\circ}C$ to the nearest degree as a function of N
4. Temperature in previous column in binary [use DEC2BIN(value,bits)]
5. Temperature in $^{\circ}F$ to the nearest degree as a function of N
6. Temperature in previous column in binary [use DEC2BIN(value,bits)]

You will find repeated column labels helpful at the top of each page.

Since V_{OUT} can range from 0.1 to 1.75 for the TC1047A, what are the possible values for N? What data type would be needed to perform calculations involving Equation 4?

Temperature Sensor

Write a main function that does the following initialization:

1. Initialize the ADC with channel $AN4$ set as the initial channel using $initADC(0x0004)$.
2. Initialize the buttons so S4 press can be detected using initButtons(0x0001).
3. Initialize the bargraph using initBargraph( ).

Write an endless loop for the main function that does the following and then repeats:

1. Read the ADCs channel AN4 using $getADC(0x0004)$.
2. Convert this reading to a binary representation of the temperature in $^{\circ}C$
3. Display temperature in $^{\circ}C$ on the bargraph using $setBargraph(\cdots)$.
4. Wait for one second using $msDelay(1000)$.

Dont forget to add $\_CONFIG1$ and $\_CONFIG2$ macros.