Weekly Report: Interactive Wheel of Fortune

Week 4
02/014/07-02/22/07
Written by: Yadverinder Singh

Work Completed:

Last week started with the goal to complete writing the overall program for the game. This goal was accomplished with 95% of the program completed. The program takes into consideration the random time output to run the motor, an automatic text to speech conversion right after the DC motor comes to a complete stop, and the initiation of colorful LEDs to produce a visual enhancement while the motor is running. However, while writing the program few other steps were taken to make sure the program was accurate as possible. Primarily, I started to test the Sp03 to see if it functioned properly and I was able to install the first phrase (“Welcome to Wheel of Fortune”) into the module and test its volume, pitch and speed. After setting the volume at the highest level, I noticed that it was quite difficult to make out the sound. So, I decided to design an amplifying circuit with its own volume control. I have not yet decided the amplification rate, but it will be around 4 to 5 amps. Also, I worked with the motor to see how long does it take for the motor to come to a complete stop after running for a few seconds and then cutting of its voltage supply. Unfortunately, the motor stops as soon as the DC voltage is cut off making the random variable generation a more crucial part of the program. Originally, my program had a delay function to delay the output response of the sp03, but now that the motor stops right after the voltage cut off, I might need to decrease the delay time period or take it out all together. This will be decided after the wheel is constructed and test with the motor to see if we can make the wheel spin down to stop rather than stopping suddenly when the dc motor stops.

These were very basic test and might need to be further tested with the complete circuit. I also met with Dr. Fox to ask for some help regarding I2C function of the microcontroller that I used to write the code for the SP03. Fortunately, Dr. Fox was able to point me to couple of different direction that made the program writing a little bit easy.

Figure1: RS232 connection to the Sp03.

The program is added to the end of the report and described below.

Right after the microcontroller is powered it goes to org 000h. Afterwards, it goes to the main subroutine in which the initial subroutine is called. The initial subroutine
initializes all the input and outputs of all the ports being used and also initializes the INTCON and Option_REG register for the interrupt routine. Then it returns from the initial subroutine and enters the main loop subroutine. Afterwards the check subroutine is called, which checks if any input occurred on pin RB0 or RB7. This subroutine goes into a loop until an input occurs on these pins. When the input occurs, the program automatically moves to the Org H ‘004’ position and initiates the interrupt routine. During an interrupt, all functions of the microcontroller stop and all timers stop. In the interrupt_routine, I store the value of the TMR0 register into my Timer0_temp variable. Since, this variable can have a random value anywhere between 0 and 255, I used this value and initiated a calibrate subroutine, which will help me create my random number. In the calibrate function, the timer0_temp value is checked if it is between 0 and 16 and if it is then a random number one is assigned. If not then it is checked if it is between 16 and 32 and number 2 is assigned as a random number. If the value is not between 16 and 32, then check if is between 32 and 48 and so on. Also in this subroutine, I assigned variable count2 to a value specific to each random number and this value is used to create the random time output for the motor. For instance, random 1 will make count2 equal to 31 and random two will make count2 equal to 33 and random 3 will make count2 equal to 35 and so on. Right after the random variable is generated, I initiated the spin subroutine which outputs a high voltage to the pin connected to the motor and all the pins connected to the LEDs, meaning that all port A is high. Afterwards, I called the delay subroutine. In this subroutine, I make the value of count1 and count3 equal to 255 and count2 is already given value from the calibrate subroutine. And decrement each of these variables in such away that the delay function will have to go through count1*count2*count3 instruction, each instruction taking around 1 microseconds. For example, if the random variable is 1 then count2 is 31 and the number of instructions the delay subroutine goes through is 255*255*31, which equals 2015775 instructions and around 2 seconds of delay. If the random variable was 16, then the count2 will equal to 63 and the number of instructions are 255*255*63, which equals 4096575 instructions and around 4 seconds of delay. So, depending on the time the interrupt occurred, the random time could have 16 different values between 2 and 4 seconds, which makes it quite random if we are using only eight pie pieces. Afterwards, the PORTA is turned low and the Sp03 subroutine is initiated with an additional delay and a text to speech conversion occurs and split out of the speaker. This basically sums up the whole program, leaving behind the variables that were setup in the beginning of the program.

**Future Work:**

This week, I plan on debugging the program and trying to install it onto the PIC 16F874 chip to see if it functions. Also, during this week we will decide on which phrases to install into the sp03 and install them on. So by the end of the week, we should be close to completing the majority electron work, with exception of the batteries that will have to wait until the whole game structure is completed.

**Project Review:**

Currently, we are slightly head in our electrical part of the project and on task with the mechanical part of the project. We believe that majority of the project will be completed
by the due date, with small things left over like writing the question or putting them on the tape.

**Time Worked:**

14 hours.

**Appendix: The program**

```plaintext
#include P16F874.inc
__config(_CP_OFF & _PWRTE_ON & _XT_OSC & _WDT_OFF & _BODEN_OFF & _LVP_OFF)
errorlevel -302 ;remove bank warnings

Equates

Bank0RAM equ H'20' ;Start of Bank 0 RAM area
TRISC equ 87h
TRISA equ 85h
PortA eqU 05h

Variables

cblock Bank0RAM
W_TEMP ; Temporary Storage for W during
interrupts
STATUS_TEMP ; Temporary storage for STATUS during
interrupts
Timer0_temp ; temporary storage for TMR0 register at
interrupt
endc

Macro definitions;

movlf macro literal,dest
movlw literal
movwf dest
endm

movff macro source,dest
movf source,w
movwf dest
endm

Bank1 macro
bsf STATUS,RP0
endm
```
Bank0         macro
    bcf   STATUS,RP0
endm

org H'000'       ;Needed to begin the program
nop
goto Main        ;goes to the begining of the program
org H'004'       ;the program returns here when ever
                 ;there is an interrupt
goto Interrupt_routine

Main
    call Initial
    call Mainloop
Mainloop
    call Check

Initial
    Bank0
    movlf b'00101000', SSPCON

    Bank1
    movlf b'01011000', OPTION_REG  ;sets the option_reg
    movlf b'11110001', TRISB       ;set I/O for Port B
    clrf TRISA    ;set I/O for Port A
    movlf b'00011000', TRISC       ;set I/O for the SP03PIN 3AND4)
    movlf b'10000000', SSPSTAT
    movlf b'11011000', INTCON
    movlf b'000101000',, SSPPCON
    return

Check
    btfss INTCON,1
    btfss INTCON,0
    bcf INTCON,1
    bcf INTCON,0
    call Check

Interrupt_routine
    Movff TMR0,Timer0_temp        ;Places the value of Timer0 into
                                  ;the temporary storage
    call Calibrate                ;Calibrate starts the process of
                                  ;generating the random number
Random number
Subroutines

Calibrate

; Move Timer0 into the w register
movf Timer0_temp,0
Sublw b'00010000'
BTFSS STATUS,0
Call Random1

; Subtract w register from 16
movf Timer0_temp,0
Sublw b'01000000'
BTFSS STATUS,0
Call Random2

; Check if carry flag has occurred
movf Timer0_temp,0
Sublw b'01110000'
BTFSS STATUS,0
Call Random3

; Repeat for Random4 to Random12
Call Random13
movf Timer0_temp,0
Sublw b'11100000'
BTFSS STATUS,0
Call Random14
movf Timer0_temp,0
Sublw b'11110000'
BTFSS STATUS,0
Call Random15
movf Timer0_temp,0
Sublw b'11111111'
BTFSS STATUS,0
Call Random16

Random1
movlf b'00000001', Random
movlf b'00011111', Count2
movff Count2, Restore
call Spin

Random2
movlf b'00000010', Random
movlf b'00100001', Count2
movff Count2, Restore
call Spin

Random3
movlf b'00000011', Random
movlf b'00100011', Count2
movff Count2, Restore
call Spin

Random4
movlf b'00000100', Random
movlf b'00100101', Count2
movff Count2, Restore
call Spin

Random5
movlf b'00000101', Random
movlf b'00100111', Count2
movff Count2, Restore
call Spin

Random6
movlf b'00000110', Random
movlf b'00101001', Count2
movff Count2, Restore
call Spin

Random7
movlf b'00000111', Random
movlf b'00101011', Count2
movff Count2, Restore
call Spin

Random8
movlf b'00001000', Random
movlf b'00101011', Count2
movff Count2, Restore
call Spin

Random9
movlf b'00001001', Random
CALCULATE COUNT2
movl b'00101111', Count2
movff Count2, Restore
Call SPIN

RANDOM 10
movl b'00001010', Random
movl b'00110001', Count2
movff Count2, Restore
Call SPIN

RANDOM 11
movl b'00001011', Random
movl b'00110011', Count2
movff Count2, Restore
Call SPIN

RANDOM 12
movl b'00001100', Random
movl b'00110101', Count2
movff Count2, Restore
Call SPIN

RANDOM 13
movl b'00001101', Random
movl b'00110111', Count2
movff Count2, Restore
Call SPIN

RANDOM 14
movl b'00001110', Random
movl b'00111001', Count2
movff Count2, Restore
Call SPIN

RANDOM 15
movl b'00001111', Random
movl b'00111011', Count2
movff Count2, Restore
Call SPIN

RANDOM 16
movl b'00010000', Random
movl b'00111101', Count2
movff Count2, Restore
Call SPIN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Spin;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Spin
movl b'00000001', PortA ; high output to pin RA0
movl b'11111111', Count1
Call Delay ;Delay
movl b'00000000', PortA ;low output to pin RA0
Call Delay3 ;Delay for Sp03
Call SP03 ;initiates the Sp03 subroutine
retlw 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Delay subroutine;;;;;;;;;;
Delay
movl b'11111111', Count3 ;stores 255 into the count3
decfsz Count1,1 ;decrements count1 by 1 call

Return
Delay1
decfsz Count3,1
call Delay2

Delay2
  decfsz Count2,1
  goto Delay2
  movff Restore, Count2
  call Delay1

; THIS SUBROUTINE IS USED TO DELAY THE RESPONSE FOR THE SP03 UNTIL THE WHEEL
; COMES TO A COMPLETE STOP

Delay3
  decfsz

; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Sp03 subroutine;;;;;;;;;;
; Configure Baud rate

;*** Begin I2C data transfer sequences***

  bsf  SSPCON2, SEN ;generates a start condition
  call WAITMSSP ;wait if the start condition finished

  bsf  SSPCON2, SEN ;generate START condition
  call WaitMSSP ;wait for I2C operation
                  ;send and check Address byte, wait for it to complete

  movlw 0XC4
  call Send_I2C_Byte
  call WaitMSSP

  btfsc SSPCON2, ACKSTAT ;Check ACK status bit
  goto I2CFail

; Send and check DATA byte, wait for it to complete

  movf  spin1
  call send_I2C_byte
  call WaitMSSP

  btfsc SSPCON2, ACKSTAT
  goto I2CFail

; Send and check the stop condition, wait for it to complete

  bsf  SSPCON2, PEN
  call WaitMSSP

; I2C operation failed code sequence. A stop is sent and the entire code is tried again

I2CFail

  bsf  SSPCON2, PEN
  call WaitMSSP

; This routine sends the W register to SSPBUF, thus transmitting a byte. The SSPIF flag is
; checked to ensure the byte was sent. On completion, the routine exits.
Send_I2C_Byte

Bank0
  movwf SSPBUF ; get value to send put in SSPBUF
  retlw 0      ; Done, Return 0

; This routine waits for the last I2C operation to complete
; It does this by polling the SSPIF flag in PIR1

waitMSSP
  btfss PIR1, SSPIF
  goto ???????? ; I2C module is not ready yet
  bcf PIR1, SSPIF ; I2C ready, clear flag
  retlw 0