Ga-wun Kim

ENGR 270

Scott Koss

Lab Partners: Rebecca, Kantas

Computer and Organization

&

Microprocessors

Lab #5

Introduction

In this lab, we were required to experiment the application of timers to schedule tasks and use of pulse width modulation to control average power delivered. Based on the given code, we modified the code.

Experiment 1

In this experiment, we are required to write an assembly code that controls the power delivered to EDbot's left motor using PWM functionality of PICmicro. We performed the following five steps.:

1. Drive the motor at minimum power level(0% duty cycle)
2. Increase the percent of power delivered to the motor by 10% every tree seconds until 40% of maximum power is achieved.
3. Decrease the percent of power delivered to the motor by 10% every tree seconds until minimum power achieved.
4. Reverse the motor direction.
5. Go to step 1

Pseudo Code:
Step 1: Initialize input/output pins on the PICmicro (18F1220)
Step 2: Enable interrupts; Enable timers
Step 3: Set timer to 3 seconds
Step 4: Wait; start of the timer: set to 0% power by default
Step 5: 3 second timer
Step 6: Increment to 10% power
Step 7: 3 second timer;Increment to 20% power
Step 8: 3 second timer;Increment to 30% power
Step 9: 3 second timer
Step 10: Increment to 40% power
Step 11: 3 second timer;Decrement to 30% power
Step 12: 3 second timer
Step 13: Decrement to 20% power
Step 14: 3 second timer
Step 15: Decrement to 10% power
Step 16: 3 second timer
Step 17: Decrement to 0% power
Step 18: 3 second timer;change wheel direction
Step 19: Go back to Step 6

The below codes are what we did.

Configured Code:

```
list p=18F1220 ; processor type
radix hex ; default radix for data
config WDT=OFF, LVP=OFF, OSC = INTIO2 ; Disable Watchdog timer, Low V. Prog, and
RA6 as a clock
STATE equ 0x80
#include p18f1220.inc ; This header file includes address and bit definitions for all SFRs
org 0x000
GOTO StartL ; Executes after reset
org 0x008 ; Executes after high priority interrupt
GOTO HPRIO
org 0x20 ; Start of the code
HPRIO: ; high priority service code
;BCF T1CON, TMR1ON ; Disable Timer 1
; Start of code to be executed during Timer1 interrupts
;0% power at the start of the code
Percent10: ; set PWM to 10%
MOVLW .0 ;test to check STATE's value
CPFSEQ STATE ;if STATE is = 0
GOTO Percent20 ;it will SKIP this
MOVLW .10 ;since that was skipped it makes PWM 10% power
MOVWF CCPR1L
INCF STATE ;increment STATE, which means it WILL go to Percent20 next time
GOTO TIdone
Percent20: ; Increment PWM to 20%
MOVLW .1 ;from percent 10 STATE was changed to be 1
CPFSEQ STATE ;so it will be equal to this value
GOTO Percent30
MOVLW .20
MOVWF CCPR1L
INCF STATE ;but next time around STATE will be 2, which moves it to Percent30
GOTO TIdone
Percent30: ; Increment PWM to 30%
MOVLW .2 ;and this keeps happening until DPercent0
CPFSEQ STATE
GOTO Percent40
MOVLW .30
MOVWF CCPR1L
INCF STATE
GOTO TIdone
Percent40: ; Increment PWM to 40%
MOVLW .3
CPFSEQ STATE
GOTO DPercent30
MOVLW .40
MOVWF CCPR1L
INCF STATE
GOTO TIdone
DPercent30: ; Decrement PWM to 30%
MOVLW .4
CPFSEQ STATE
GOTO DPercent20
MOVLW .30
```

```
MOVWF CCPR1L
INCF STATE
GOTO TIdone
DPercent20: ; Decrement PWM to 20%
MOVLW .5
CPFSEQ STATE
GOTO DPercent10
MOVLW .20
MOVWF CCPR1L
INCF STATE
GOTO TIdone
DPercent10: ;Decrement PWM to 10%
MOVLW .6
CPFSEQ STATE
GOTO DPercent0
MOVLW .10
MOVWF CCPR1L
INCF STATE
GOTO TIdone
DPercent0: ;Decrement PWM to 0% power
CLRF STATE ;Clear the state so it goes back to Percent10 next time around.
MOVLW .0 ;0% power
MOVWF CCPR1L
BTG PORTA,7 ;Every time the code reaches this point it will change wheel directions
GOTO TIdone
TIdone: ; get ready to return from interrupt
; Reset Timer 1 so next timer interrupt is in approximately 3 seconds
MOVLW 0x39
MOVWF TMR1L
MOVLW 0xD2
MOVWF TMR1H
BCF PIR1, TMR1IF ; Clear Timer 1 Interrupt Flag
BSF T1CON, TMR1ON ; Enable Timer 1;
RETFIE ; Return from interrupt
StartL: ; entry point from reset
; Initialize all I/O ports
;BTFSS STATE ;filler for now, here if my code doesn't work how I want it to
CLRF STATE ; Initialize STATE
CLRF PORTA ; Initialize PORTA
CLRF PORTB ; Initialize PORTB
MOVLW 0x7F ; Set all A\D Converter Pins as
MOVWF ADCON1 ; digital I/O pins
MOVLW 0x0D ; Value used to initialize data direction
MOVWF TRISA ; Set Port A direction
MOVLW 0xC7 ; Value used to initialize data direction
MOVWF TRISB ; Set Port B direction
MOVLW 0x00 ; clear Wreg
; Timer 1 Initialization + interrupt enable/disable
BSF INTCON, GIE ; enable interrupts
BSF INTCON, PEIE ; enable all interrupts
BSF PIE1, TMR1IE ; enable Timer1 Interrupt
BSF IPR1, TMR1IP ; Set Timer 1 Interrupt to High priority
MOVLW 0x58 ; Timer 1: "8&8-bit, osc.clock, 1:2 pre-scale, enabled, internal clk"
MOVWF T1CON ; "0 1 01 1 0 0 0"
```

```
; Set Timer 1 so next timer interrupt is in approximately 2seconds
; 3sec/(4*2*32*10^(-6)) = 11718 (really 11719 but I calculated for 11718) desired ticks
;so where do these values come from? The formula is "desired-second/tick." Or 3/tick.
;We have a pre-scale of 2 which means we multiply our tick by 2 (*1 for prescale 1:1, *4
for 1:4, etc)
;and number of ticks is 4 * Hz (in this case 32*10^-6) along with the prescale 1:2.
;Our desired second is 3. Which is where all the numbers for line 122 come from.
;Once we have the desired tick number (11718) we subtract from the machine's bit value.
;T1con is 2^16, subtract with the desired tick value we got, we put that number in TRM1H
& TMR1L in hex
; so we set (TRM1H & TMRL) to { (2^16) ? 11,719 = 53817} or (D239)H
MOVLW 0x39
MOVWF TMR1L ;I tested using an online counter, D239 is definitely approximately 3
seconds. Definitely approximately.
MOVLW 0xD2
MOVWF TMR1H
BSF T1CON, TMR1ON ; Enable Timer 1
; Following 6 steps configure PWM power level based on the PICmicro Data Sheet
starting at page 131
; 1) PWM will be delivered on P1A (pin 18) which controls Left Motor; for this code, use
TOSC = 32 usec.
MOVLW 0x00C ; "0000 1100
MOVWF CCP1CON ; PWM output on P1A (Pin 18)
; 2)PWM Requires Timer 2 and must be enabled for(PWM requires Timer 2)
CLRF TMR2 ; Timer 2 Register
MOVLW 0x05 ; Enable timer and set pre-scale to 4
MOVWF T2CON
BCF PIR1, TMR2IF ; Clear Timer 2 flag
; 3) Initialize PWM Period to PWM Period = (PR2 + 1) * 4 * TOSC * (TMR2 Pre-scale) =
(99 + 1) * 4 * 32 usec * 4 = 51 msec
MOVLW .99
MOVWF PR2
;4) Set PWM On-time to(CCPR1L:CCP1CON<5:4>)*TOSC*(TMR2 Pre-scale) =
(CCPR1L:00)* 32 * 4 usec
; With this configuration, value stored in CCPR1L defines the duty cycle and therefore
the % power leve
MOVLW .0
MOVWF CCPR1L ; Set the power level to 10%
;5) Need to wait until timer2 has overflowed once and set PWM Pin 18 to output
WAITL:
BTFSS PIR1, TMR2IF
BRA WAITL
BCF TRISB,3 ; Set P1A/RB3/CCP1 as an output pin
BSF PORTB,5 ; turn on LED just to indicate EDbot is on
MainL: ; waiting in a loop
; Add main (non-interrupt) code that should be executed here.
BRA MainL
end ; end of code
```

Experiment 2

In this experiment, we were required to write an assembly code that drives EDbot forward in

circles. At first, the robot circled clockwise at 50% power for 5 seconds, after that, clockwise at 20% for 5 seconds before stopping. Also, we used a program that modulated(PWM) left and right motor drive pins.

Pseudo Code:

Step 1: Initialize input/output pins on the PICmicro (18F1220)
Step 2: Enable interrupts; Enable timers
Step 3: Set timer to 5 seconds
Step 4: Wait; start of the timer: set to 50% power
Step 5: Change wheel direction
Step 6: Set timer to 5 seconds
Step 7: Wait; start of the timer: set to 20% power

```
list p=18F1220
radix hex ; default radix for data
config WDT=OFF, LVP=OFF, OSC = INTIO2 ; Disable Watchdog timer, Low V. Prog, and
RA6 as a clock

#include p18f1220.inc
  org 0x000
  GOTO StartL ; Executes after reset
  org 0x008 ; Executes after high priority interrupt
  GOTO HPRIO
  org 0x20 ; Start of the code

Counter equ 0x80
Direction equ 0x81
DelayCounter equ 0x82
TurnCounter equ 0x83

HPRIO: ; high priority service code
  BTG Direction, 0
  INCF TurnCounter

TIdone: ; get ready to return from interrupt; Reset Timer 1 so next timer interrupt is in
approximately 2 seconds
  MOVLW .3
  CPFSLT TurnCounter
  BRA Done

  INCF TurnCounter

  BTG PORTB, 5 ; toggle LED

  MOVLW 0xB3 ; Reset timer to 5 seconds
  MOVWF TMR1H
  MOVLW 0xB5
  MOVWF TMR1L

  BCF PIR1, TMR1IF ; Clear Timer 1 Interrupt Flag
  BSF T1CON, TMR1ON ; Enable Timer 1;
  RETFIE ; Return from interrupt
```

```
StartL: ; entry point from reset
   MOVLW .10
   MOVWF Counter
   MOVLW .1
   MOVWF Direction
   CLRF TurnCounter

   ; Initialize all I/O ports
   CLRF PORTA ; Initialize PORTA
   CLRF PORTB ; Initialize PORTB
   MOVLW 0x7F ; Set all A\D Converter Pins as
   MOVWF ADCON1 ; digital I/O pins
   MOVLW 0x0D ; Value used to initialize data direction
   MOVWF TRISA ; Set Port A direction
   MOVLW 0xC7 ; Value used to initialize data direction
   MOVWF TRISB ; Set Port B direction
   MOVLW 0x00 ; clear Wreg

   ; Timer 1 Initialization + interrupt enable/disable
   BSF INTCON, PEIE ; enable all peripheral interrupts
   BSF PIE1, TMR1IE ; enable Timer1 Interrupt
   BSF IPR1, TMR1IP ; Set Timer 1 Interrupt to High priority
   MOVLW 0x58 ; Timer 1: "8&8-bit, osc. clock, 1:2 pre-scale, enabled, internal clk"
   MOVWF T1CON ; "0 1 01 1 0 0 0"

   MOVLW 0xB3 ; Set timer to 5 seconds
   MOVWF TMR1H
   MOVLW 0xB5
   MOVWF TMR1L

   BSF T1CON, TMR1ON ; Enable Timer 1
   BSF INTCON, GIE ; enable interrupts globally
   BRA MainL

GoClockwise: ; Handle left motor to 50% power
   MOVLW .1
   CPFSEQ Direction
   RETURN
   MOVLW .6
   CPFSLT Counter
   RETURN
   BSF PORTB,3 ; Enable left motor
   RETURN

GoCounterClockwise: ; Handle right motor 20% power
   MOVLW .0
   CPFSEQ Direction
   RETURN
   MOVLW .3
   CPFSLT Counter
   RETURN
   BSF PORTB,4 ; Enable right motor
```

```
   RETURN

DisableMotors:
   BCF PORTB,3 ; Disable left motor
   BCF PORTB,4 ; Disable right motor
   RETURN

ResetCounter:
   MOVLW .10
   MOVWF Counter
   RETURN

MainL: ; waiting in a loop
   ; Add main (non-interrupt) code that should be executed here.
   CALL DisableMotors
   CALL GoClockwise
   CALL GoCounterClockwise

   DECF Counter
   MOVLW .0
   CPFSGT Counter
   CALL ResetCounter

   MOVLW .250
   MOVWF DelayCounter

Loop:
   INCF DelayCounter
   BZ MainL
   BRA Loop

Done:
   CALL DisableMotors
   end ; end of code
```

What we learned from this lab.

- Application of timer to schedule tasks.
- Use of PWM to control average power delivered.

Conclusion (New experiment)

Before this lab, we learned regarding the PWM period; calculating PWM period. From this lab, we actually used PWM period so that I could make more sense about this conception. Also, we could finish the experiment 1 early rather than experiment 2. At first, we though that we could combine the lab4 and lab5(ex1), and then, we expected that we could get a right code for the experiment 2 but it did not. Because of that, we should spend much more time than the experiment 1.

Signature for certificate.