EMK 310

Microprocessors

Practical 1

Traffic Light Controller

Student Name	Student Number	Contribution
MT Mokgotho	[redacted]	25%
F Botha	29042951	25%
S Dinath	[redacted]	25%
S Taghizadeh	[redacted]	25%

March 2012

Table of Contents

1. Introduction....3
2. Problem Statement....3
3. Apparatus....3
4. Flowchart of the Software....4
5. State diagrams of the traffic cycle....7
6. Truth Table....8
7. Circuit Design....9
8. Code....11
9. References....21

1. Introduction

The objective of this practical are for students to

- Learn the program coding in assembly language
- Learn to program a microcontroller
- Learn using interrupts and delays
- Get to know the software development tools that accompany the PIC microcontrollers
- Learn to implement a simple application with a PIC microcontroller

2. Problem Statement

Design a microcontroller system to implement a two-way traffic light controller.

3. Apparatus

- PIC18F45K20
- Normally open push-buttons
- LED’s to represent traffic lights
- Various resistors and capacitors

4. Flowchart of the Software

i. Flowchart description

The program starts by going through stages 1-3, setting the appropriate lights on and off by passing binary values into port A and port B respectively.

When the program reaches state 4, the system is tested for rush hour and non-rush hour times by using an increase count procedure. The system then either runs through state 4 for non-rush hour or through state 4a-4c for rush hour, then it goes on to run through states 5-8 concerning interrupts we enable. The global interrupts are only during states 2, 6, 3 and7 so that a button press during any other time is not implemented as an interrupt.

When the button is pressed during state 2 and 6, the program completes the current 2 second delay before going to ISR and implementing the “button press” call to the appropriate LED’s.

For a button press during states 3 and 7, the system is programmed to go immediately to the next state by using a flag check procedure.

The program runs continuously through the different states checking each time whether its rush hour or non-rush hour, whether the button was pressed or not and what actions must be taken accordingly.

ii. System description

One cycle has 8 states (refer to Figure 3)

State 1	2s	N/S and E/W red, all pedestrian lights red.
State 2	4s	E/W red and N/S green, N/S pedestrian green.
State 3	4s	E/W red and N/S green, N/S pedestrian red.
State 4 NRH	4s	E/W red and N/S yellow, all pedestrian lights red.
State 4a RH	2s	E/W red, N green, S yellow, all pedestrian lights red.
State 4b RH	4s	E/W/S red, N green and arrow flashing with 0.5 intervals.
State 4c RH	2s	E/W/S red, N yellow.
State 5	2s	N/S and E/W red, all pedestrian lights red.
State 6	4s	N/S red and E/W green, E/W pedestrian green.
State 7	4s	N/S red and E/W green, E/W pedestrian red.
State 8	4s	N/S red and E/W yellow, all pedestrian lights red.

iii. Flowchart

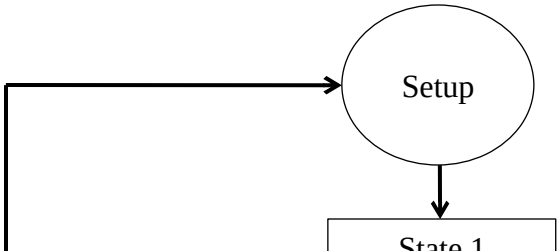


Figure 1 – Flowchart

5. State diagrams of the traffic cycle

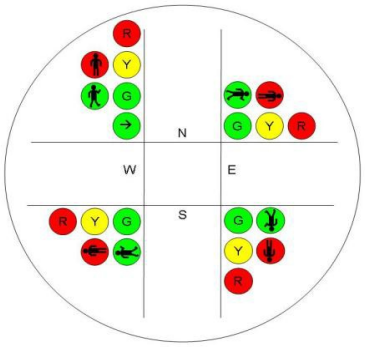


Figure 2 - Traffic light key

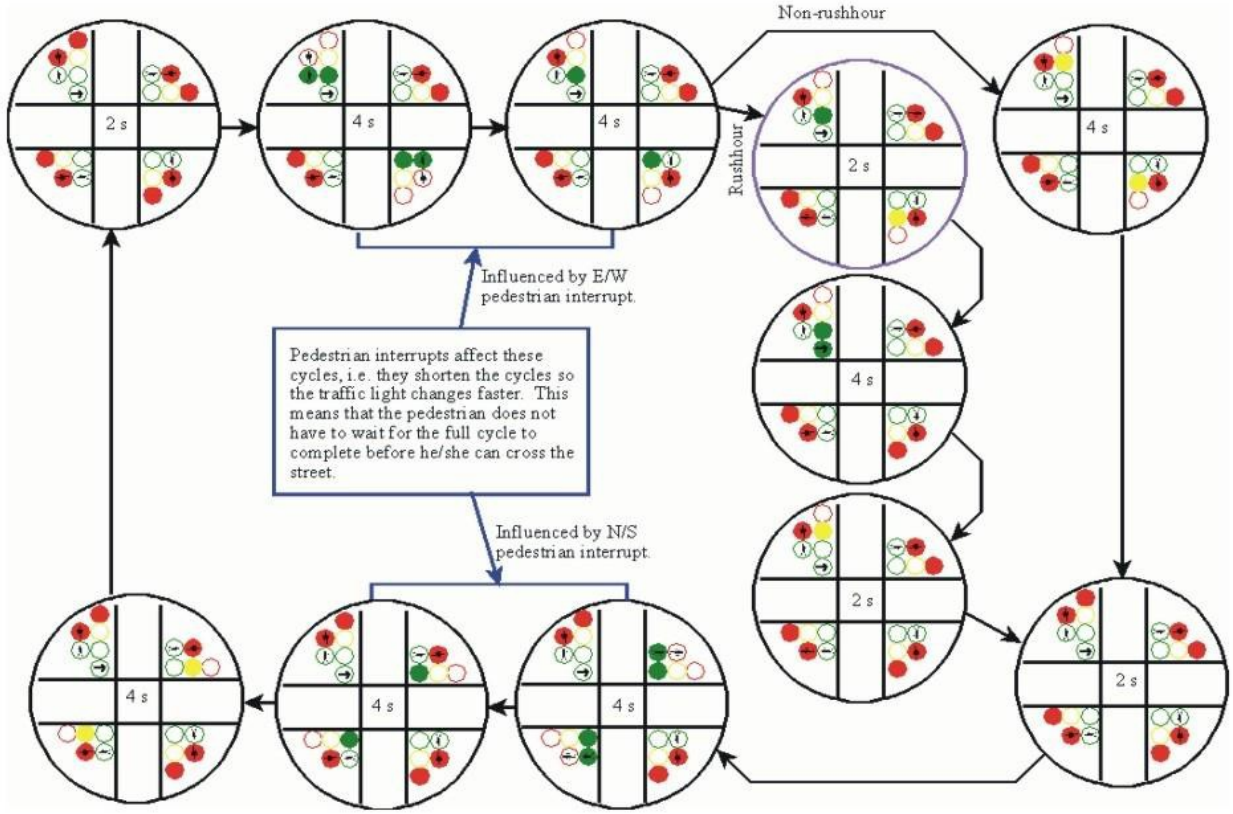


Figure 3 - State diagram. State 1 is top left

6. Truth Table

Pin numbers	8	14	13	12	10	11	1	2	7	5	4	17	20	18	16	19
Port	**R_D**	**R_D**	**R_D**	**R_D**	**R_D**	**R_D**	**R_D**	**R_D**	**R_C**	**R_C**	**R_C**	**R_C**	**R_C**	**R_C**	**R_C**	**R_C**
Port numbers	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
direction	**N**						**S**					**E/ W**				
State		**G**	**Y**	**R**	**P_G**	**P_R**	**G**	**Y**	**R**	**P_G**	**P_R**	**G**	**Y**	**R**	**P_G**	**P_R**
1	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1
2	0	1	0	0	1	0	1	0	0	1	0	0	0	1	0	0
3	0	1	0	0	0	1	1	0	0	0	1	0	0	1	0	0
4	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	1
4a	0	1	0	0	0	1	0	1	0	0	1	0	0	1	0	1
4b	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0
4c	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
5	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1
6	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0
7	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	1
8	0	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

7. Circuit Design

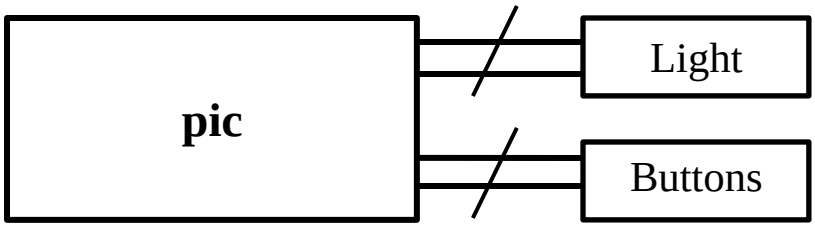


Figure 4 - Circuit diagram

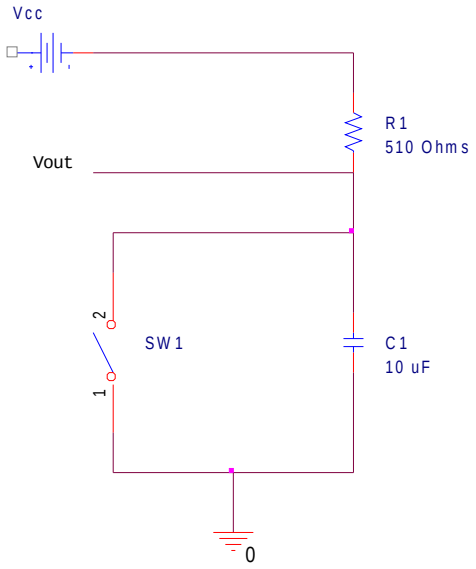


Figure 5 - Buttons diagram

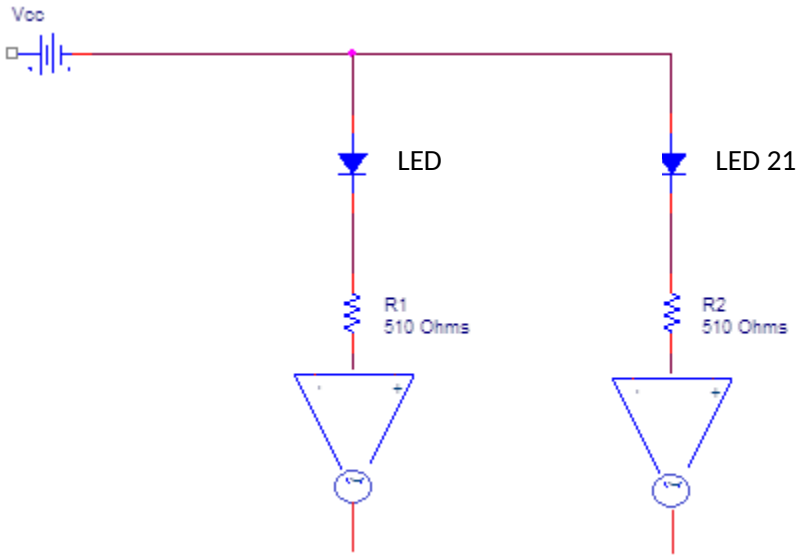


Figure 6 - Lights diagram

Circuit description

The microprocessor drives a block of inverters that serve as current buffers. The inverters sink current through the resistors and LED's turning them on. The interrupt buttons are implemented using a simple capacitor de-bouncing technique. The capacitor discharged rapidly when the button is pressed, it then takes time constant =RC for the pulse to return the high, effectively eliminating bounce pulses.

8. Code

```
;============== Device setup ==========================
            title             "Practical 1"
            LIST              P=PIC18F45K20 ; processor type
            #include    "C:\Program Files\Microchip\MPASM
Suite\p18f45K20.inc"

; -------------
; PROGRAM START
; -------------

;============== Definition of variables ===============
      cblock 0x00             ;start at first address in general
purpose memory (part of Access Bank)
            Delay1                  ; Define two file registers for the
            Delay2                  ; delay loop
            count1                  ; counter register address
            count2                  ; counter register address
            count3                  ; counter register address
            interrupt_flag          ; flag to test if interrupt occurred
            stateflag               ; flag to show which state
            state3_flag             ;flag to show state 3
            state7_flag             ;flag to show state 7

      endc

;============== Reset vector ===============
            org         0h                ; startup address = 0000
            goto        Setup

;============== Interrupt vector ==========
            org         08h
            goto        ISR


;================ Setup ====================

Setup

            ; Set oscillator speed at 4MHz
            bsf               OSCCON,IRCF0
            bcf               OSCCON,IRCF1
            bsf               OSCCON,IRCF2


      ; Setup Port B for external interrupt

      clrf        PORTB             ; Initialise PORTB by clearing port
```

```
            clrf        ANSELH
            bsf         TRISB,0           ; Set IO pin B0 as an input
            bsf         TRISB,1           ; Set IO pin B1 as an input

      ; Initialize Port C

            CLRF        PORTC             ; Initialize PORTC by clearing port

            CLRF        TRISC             ; All digital outputs

      ; Initialize Port D
            CLRF        PORTD             ; Initialize PORTD by clearing port
            CLRF        TRISD             ; All digital outputs

      ; Setup external interrupt
            clrf        INTCON            ; clear all interrupt bits
            bsf         INTCON,INT0IE     ; enable RB0 interrupt


      ;============= main program ===============

      Main

      ;State 1
                  call        state1
                  call        two_second_delay_loop

      ;State 2

                  bcf         INTCON,INT0IF        ;clear interrupt at other
time

                  call        state2
                  call        two_second_delay_loop
                  bsf         INTCON,GIE           ;enable global interrupt
                  btfss       interrupt_flag,0     ;test flag to see if
      interrupt occurred in first 2 seconds
                  call        two_second_delay_loop
                  bcf         INTCON,GIE           ;disable global interrupt

                  bcf         interrupt_flag,0     ;reset flag

      ;State 3
                  bcf         INTCON,INT0IF        ;clear interrupt at other
      time
                  bsf         state3_flag,0        ;test to show interrupt
      occured in state 3
                  call        state3
                  bsf         INTCON,GIE        ;enable global interrupt
                  call        two_second_delay_loop
```

```
            call        two_second_delay_loop
            bcf         INTCON,GIE        ;disable global interrupt
            bcf         state3_flag,0     ;reset flag to show end of
state 3

carry_on1   ; to continue after an interrupt in state 3

            bcf         state3_flag,0      ;clear flag to show end of
state 3
            bcf         INTCON,INT0IF

;State 4
            movlw       0x02              ;move value 2 to working
register
            cpfslt      count2            ;compare count2 to wreg skip
if less than
            clrf        count2            ;reset count 2 to show end of
morning rushour
            movlw       0x04              ;move number 4 into working
register
            cpfslt      count1            ;compare count to working
register
            goto        morning_rushhour  ;call rush hour subroutine
            call        state4
            call        two_second_delay_loop
            call        two_second_delay_loop

nonrushhour ; to continue after rushhour period
;State 5

            movlw       0x02                ; move value 2 to working
register
            cpfslt      count2              ; compare value in count2
to wreg skip if less than
            clrf        count1              ; clear count1 to show end
of morning rushhour
            movlw       0x0C                ; move the value 12 to wreg

            cpfslt      count3
            goto        evening_rushhour    ; provide for extra cycle
in evening rushhour
            call        state5
            call        two_second_delay_loop

;State 6
            bcf         INTCON,INT0IF        ;clear interrupt at other
time
            call        state6
            call        two_second_delay_loop
```

```
            bsf         INTCON,GIE              ;enable global interrupt
            btfss       interrupt_flag,0        ;test flag to see if
interrupt occurred in first 2 seconds
            call        two_second_delay_loop
            bcf         INTCON,GIE              ;disable global
interrupt
            bcf         interrupt_flag,0        ;reset flag

;State 7
            bcf         INTCON,INT0IF           ;clear interrupt at
other time
            bsf         state7_flag,0           ;test to show interrupt
occured in state 7
            call        state7
            bsf         INTCON,GIE              ;enable global interrupt
            call        two_second_delay_loop
            call        two_second_delay_loop
            bcf         INTCON,GIE              ;disable global
interrupt
            bcf         state7_flag,0           ;reset flag to show end
of state 7

carry_on2   ; to continue after an interrupt in state 7

            bcf         state7_flag,0           ;clear flag to show end
of state 7
            bcf         INTCON,INT0IF
;State 8

            call        state8
            call        two_second_delay_loop
            call        two_second_delay_loop

            incf        count1                      ; count to keep
track of no. of cycles (to test for rushhour)
            incf        count3                      ; count to keep
track of no. of cycles (to test for evening rushhour)
            goto        Main                        ; do this loop
forever


;=========== end of main program ==================



state1
            movlw       b'10100101'       ; move binary to W register
            movwf       PORTC             ; move W to port C
```

```
            movlw       b'00101000'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return


state2
            movlw       b'10100010'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'01010010'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

state3
            movlw       b'10100100'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'01100010'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

state4
            movlw       b'10100100'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'10100100'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

state4a
            movlw       b'10100100'             ; move binary to W
register
            movwf       PORTC                   ; move W to port C
            movlw       b'10100010'             ;move binary to W
register
            movwf       PORTD                   ;move W to port D

            return

state4b1
            movlw       b'10100101'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'00100011'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

state4b2
```

```
            movlw       b'10100101'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'00100010'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

state4c
            movlw       b'10100101'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'00100100'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

state5
            movlw       b'10100101'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'00101000'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

state6
            movlw       b'01001101'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'00101000'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

state7
            movlw       b'10001101'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'00101000'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

state8
            movlw       b'10010101'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'00101000'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

;========= Rush hour routines ===================
```

```
morning_rushhour                                ;Rushhour routine in
place of state 4
            call  state4a                       ;start rushhour routine
            call  two_second_delay_loop

            call  state4b1
            call  half_second_delay
            call  state4b2
            call  half_second_delay
            call  state4b1
            call  half_second_delay
            call  state4b2
            call  half_second_delay
            call  state4b1
            call  half_second_delay
            call  state4b2
            call  half_second_delay
            call  state4b1
            call  half_second_delay
            call  state4b2
            call  half_second_delay

            call  state4c
            call  two_second_delay_loop

            incf  count2                ; increase count to keep
track of no. of rush hour cycles

            goto  nonrushhour

evening_rushhour                                ;Rushhour routine in
place of state 4
            call  state4a                             ;start rushhour
routine
            call  two_second_delay_loop

            call  state4b1
            call  half_second_delay
            call  state4b2
            call  half_second_delay
            call  state4b1
            call  half_second_delay
            call  state4b2
            call  half_second_delay
            call  state4b1
            call  half_second_delay
            call  state4b2
            call  half_second_delay
            call  state4b1
```

```
            call  half_second_delay
            call  state4b2
            call  half_second_delay

            call  state4c
            call  two_second_delay_loop

            clrf  count3

            goto nonrushhour

;========= Button press routine ================

button_press1
            btfsc       state3_flag,0     ; flag to test if in state 3
            goto        carry_on1         ;go immediately to next state
            btfsc             state7_flag,0     ; flag to test if in
state 7
            goto        carry_on2         ;go immediately to next state
            movlw       b'10100010'       ; move binary to W register
            movwf       PORTC             ; move W to port C
            movlw       b'01100010'       ;move binary to W Register
            movwf       PORTD             ;move W to port D

            return

button_press2
            btfsc       state3_flag,0           ; flag to test if in
state 3
            goto        carry_on1         ;go immediately to next state
            btfsc       state7_flag,0           ; flag to test if in
state 7
            goto        carry_on2         ;go immediately to next state
            movlw       b'01100010'             ; move binary to W
register
            movwf       PORTC             ; move W to port C
            movlw       b'01010010'       ;move binary to W register
            movwf       PORTD             ;move W to port D

            return

;========= Delay loop subroutine: 2 second delay

two_second_delay_loop

      call  delay_loop
      call  delay_loop
      call  delay_loop
      call  delay_loop
```

```
      call  delay_loop
      call  delay_loop
      call  delay_loop
      call  delay_loop
      call  delay_loop
      call  delay_loop
      call  delay_loop

      return
;======== 0.5s delay ========================

half_second_delay

      call delay2_loop
      call delay2_loop
      call delay2_loop
      call delay2_loop

      return


;======== Delay loop subroutine: 182 ms delay

delay_loop                    ; Subroutine delay_loop
      movlw 0xF5
      movwf Delay2
Go_on1                        ; Label to facilitate reloading of Delay1
initial value
      movlw 0xF6
      movwf Delay1
Go_on2
      decfsz      Delay1,f    ; Waste time
      goto  Go_on2            ; The Inner loop takes (1418)
instructions per loop * 256 loopss = (363008) instructions
      decfsz      Delay2,f    ; The outer loop takes an additional
(1420) instructions per loop + 2 instructions to reload Delay 1) * 256
loops
      goto  Go_on1            ; (2838.75+5) * 256 = 728000 instructions
/ 4M instructions per second = 0.182 sec.

      return


delay2_loop                   ; Subroutine delay_loop
      movlw 0xC4
      movwf Delay2
Go_on3                             ; Label to facilitate reloading of
Delay1 initial value
      movlw 0xD3
```

```
      movwf Delay1
Go_on4
      decfsz      Delay1,f    ; Waste time
      goto  Go_on4            ; The Inner loop takes (1418)
instructions per loop * 256 loops = (363008) instructions
      decfsz      Delay2,f    ; The outer loop takes an additional
(1420) instructions per loop + 2 instructions to reload Delay 1) * 256
loops
      goto  Go_on3            ; (2838.75+5) * 256 = 728000 instructions
/ 4M instructions per second = 0.182 sec.

      return

;======== Interrupt Service Routine =========

ISR

      btfss PORTB,0                       ;test whether button E/W was
pressed
      call  button_press1                 ;set pedestrian lights to Red
      btfss PORTB,1
      call  button_press2                 ;test whether button N/S was
pressed...set pedestrian lights to Red
      call  two_second_delay_loop
      bsf         interrupt_flag,0        ;set flag to show interrupt
occured
      bcf   INTCON,INT0IF                 ;clear interrupt flag

      retfie

;============ End of code ====================
            end
```

9. References

- EMK 310 Study guide, Prof T Hanekom, 30 January 2012

- PICKIT3 data sheet
- http://www.ee.up.ac.za/main/en/undergrad/subjects/emk310/index , Prof T Hanekom, 29 February 2012
- PICKIT45K20 data sheet