University of Jordan

School of Engineering

Department of Mechatronics Engineering

Microprocessor and Microcontroller Laboratory / 0908432

Semester Project

Project Title: Study of Promoting a Better Driving Behavior Through Emojis

Name	ID
Ramez Silawy	0167595
Salah Saleh	0160928
Ibrahim Zaraiqy	0163768
Laith Hadidi	0171498
Zakariya Migdady	0145809

Table of Contents

List of Figures

Abstract

Car accidents harvest and impact a lot of lives on a daily basis, losses which do not only affect the families and wellbeing of individuals, it also affects the effective outcome of individuals and damages the economy of the whole country. This project demonstrates an attempt to limit reckless driving by comparing the speed of the driver to the speed limit of the street; if the driver is driving below the limit then a happy face will be shown to him on an LCD screen, if the driver is driving within the limit a normal face will be showing on the screen, and if the car's speed is above the limit an angry face will be shown.

This report includes the specific steps taken to achieve this project, including drafting a block diagram, then a more sophisticated flowchart, and finally the code itself. Followed by the code is the conclusion of this project and lessons learned.

Block Diagram of the Project

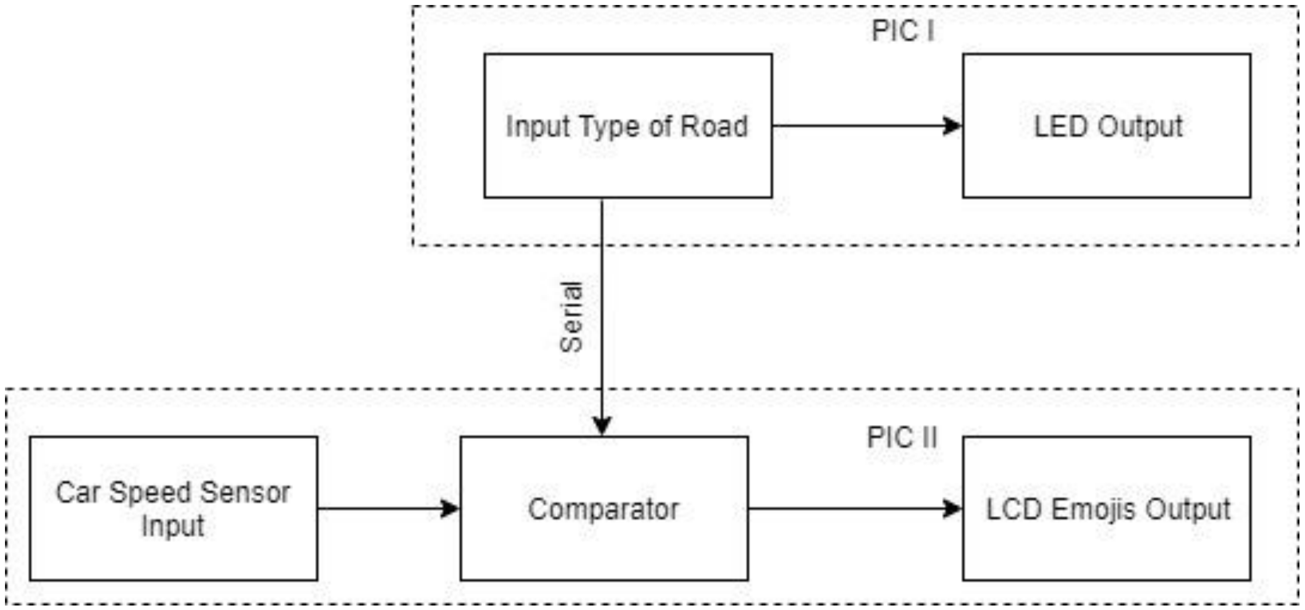

Figure 1. Simplified Block Diagram Illustrating the working of the project

Flow Chart of the Code

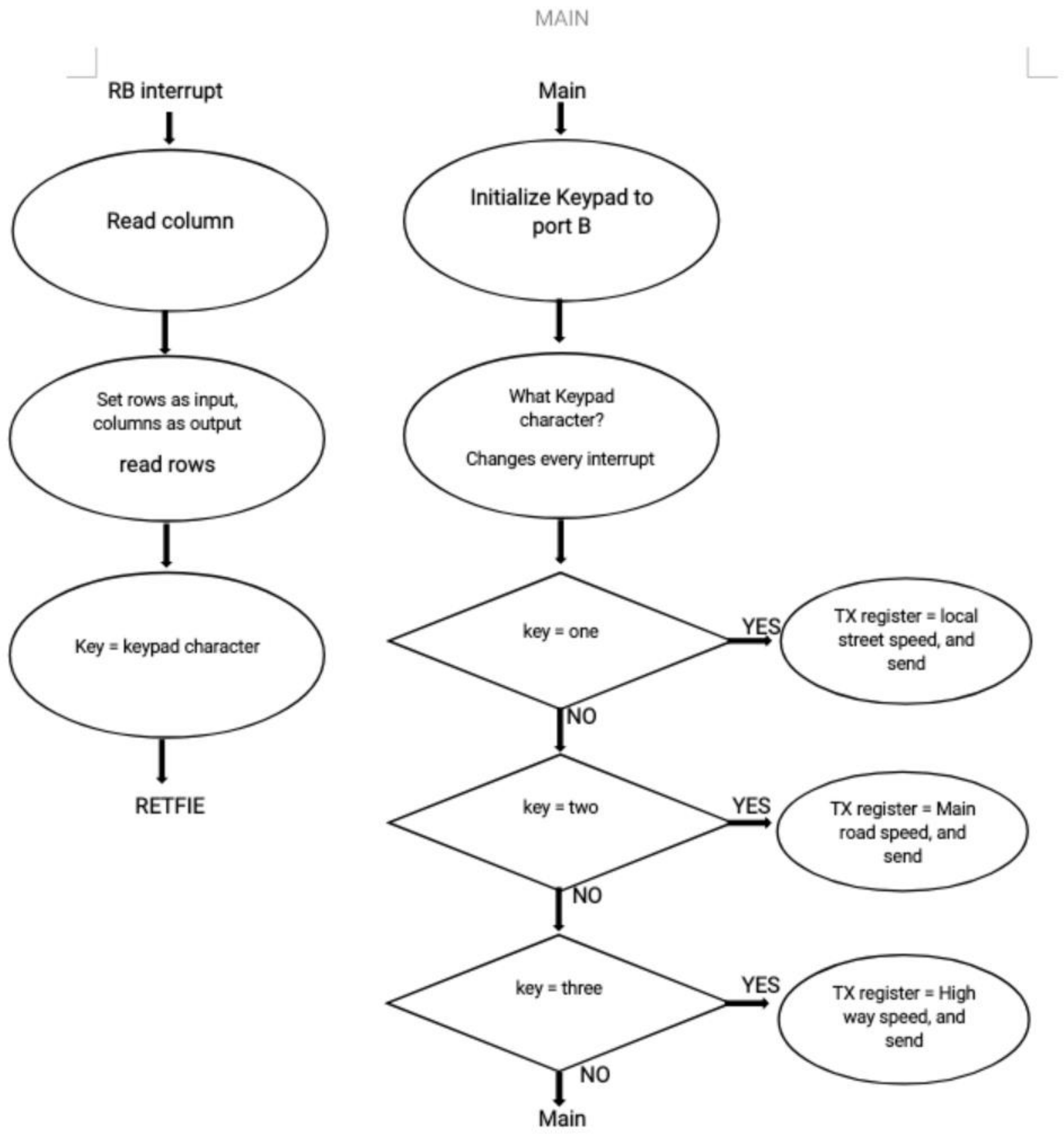
MAIN
RB interrupt
Read column
Set rows as input, columns as output
read rows
Key = keypad character
RETFIE
Main
Initialize Keypad to port B
What Keypad character?
Changes every interrupt
key = one
YES
TX register = local street speed, and send
NO
key = two
YES
TX register = Main road speed, and send
NO
key = three
YES
TX register = High way speed, and send
NO
Main

Figure 2. PIC I Code Flow Chart

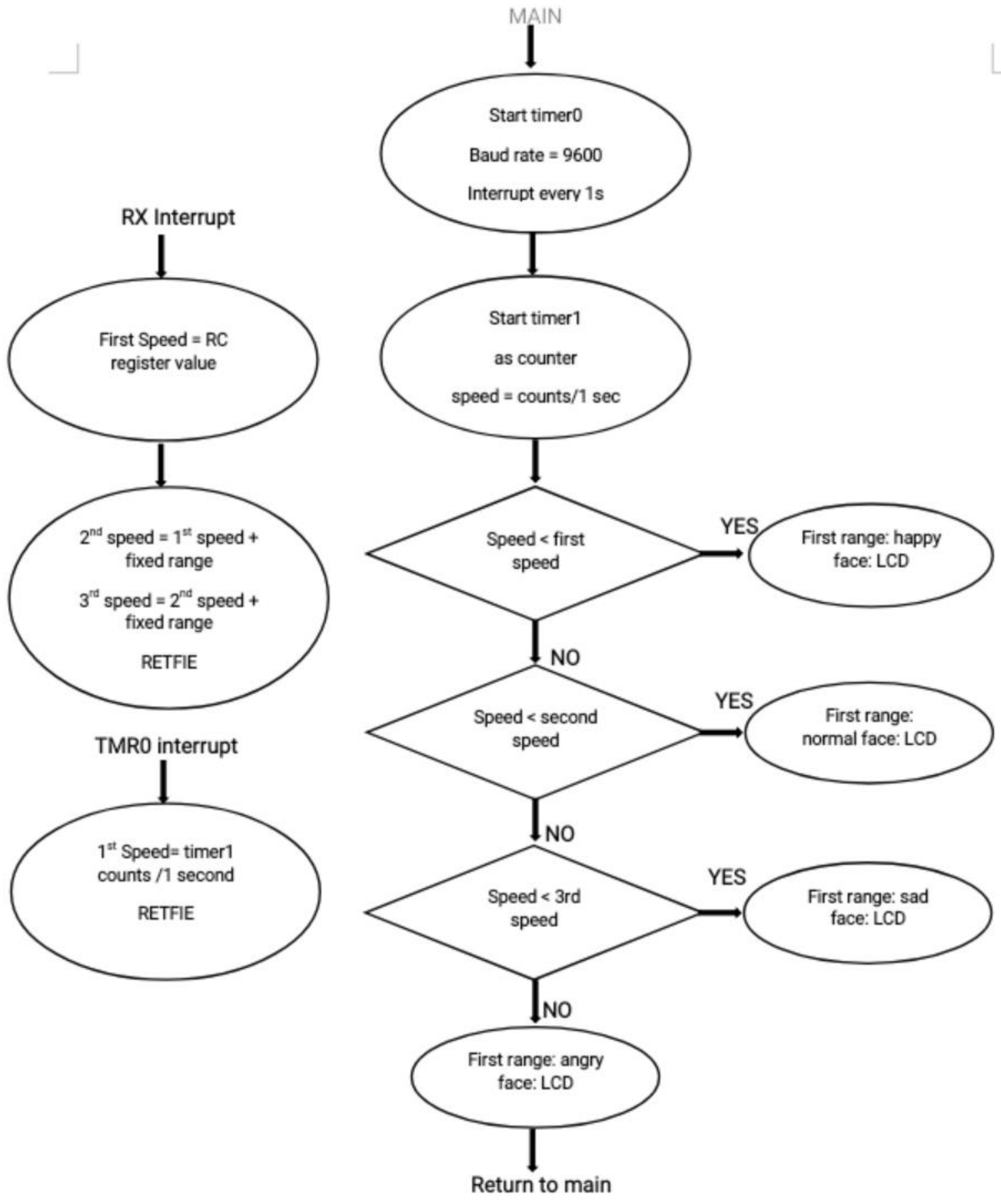
MAIN
Start timer0
Baud rate = 9600
Interrupt every 1s
Start timer1
as counter
speed = counts/1 sec
Speed < first speed
YES
First range: happy face: LCD
NO
Speed < second speed
YES
First range: normal face: LCD
NO
Speed < 3rd speed
YES
First range: sad face: LCD
NO
First range: angry face: LCD
Return to main
RX Interrupt
First Speed = RC register value
2nd speed = 1st speed + fixed range
3rd speed = 2nd speed + fixed range
RETFIE
TMR0 interrupt
1st Speed= timer1 counts /1 second
RETFIE

Figure 3. PIC II Code Flow Chart

Conclusion

The challenging part of working in this project has been more of an organizational and methodological challenge rather than project itself. It was evident that working our way up from a block diagram conceptualization to flow chart of the code was more critical than the code itself and enabled us to effectively divide the load of the project among ourselves.

In this work, we had to read further and understand some concepts of working with the MPLAB IDE, the resource is shown in [1]

Assembly Code

```
__CONFIG
_DEBUG_OFF&_CP_OFF&_WRT_HALF&_CPD_OFF&_LVP_OFF&_BODEN_OFF&_PW
RTE_OFF&_WDT_OFF&_XT_OSC
;*****************************************************************************
***
#INCLUDE "P16F877A.INC"
CBLOCK      0x20
DELCNTR1            ; Used in generating 10 ms delay
DELCNTR2
KPAD_PAT            ; Holds the pattern retrieved from keypad
KPAD_ADD            ; Holds keypad address to lookup table (generated from
                        ; KPAD_PAT to get KPAD_CHAR)
KPAD_CHAR               ; Holds the 7-segment representation of the most recent
                        ; character pressed on keypad
Counter
;tempChar
;charCount
lsd                     ;lsd and msd are used in delay loop calculation
msd
        WTemp           ; WTemp must be reserved in all banks
                    StatusTemp
                    Timer1Counts
                    TMR0_Counter
Counter
speed
speed1
```

```
speed2
ENDC
        cblock      0x0A0       ; bank 1 assignnments
                    WTemp1              ; bank 1 WTemp
        endc

        cblock      0x120       ; bank 2 assignnments
                    WTemp2              ; bank 2 WTemp
        endc

        cblock      0x1A0       ; bank 3 assignnments
                    WTemp3              ; bank 3 WTemp
        endc

;*********************************************************************************
***
Zero    equu    B'00111111' ; 7-Segment Code for Zero
One             equ     B'00000110'     ; 7-Segment Code for One
Two             equ  B'01011011' ; 7-Segment Code for Two
Three           equ     B'01001111' ; 7-Segment Code for Three
Four            equ  B'01100110'    ; 7-Segment Code for Four
Five            equ  B'01101101'    ; 7-Segment Code for Five
Six     equ  B'01111101' ; 7-Segment Code for Six
Seven           equ     B'00000111' ; 7-Segment Code for Seven
Eight           equ  B'01111111'    ; 7-Segment Code for Eight
Nine            equ  B'01101111'    ; 7-Segment Code for Nine
LetterA                 equ         B'01110111'   ; 7-Segment Code for A
LetterB                 equ         B'01111100'   ; 7-Segment Code for B
LetterC                 equ         B'01011000'   ; 7-Segment Code for C
LetterD                 equ         B'01011110'   ; 7-Segment Code for D
LetterE         equ         B'01111001'   ; 7-Segment Code for E
```

```
LetterF        equu           B'01110001'    ; 7-Segment  Code for F
;***************************************************************************
; START  OF EXECUTABLE  CODE
;***************************************************************************
        ORG    0x00
        GOTO INITIAL
;***************************************************************************
; INTERRUPT  VECTOR
;***************************************************************************
        ORG    0x04
        GOTO ISR
;***************************************************************************
INITIAL
movlw  D'25'           ; This  sets the baud  rate to 9600
        banksel        SPBRG                ; assuming  BRGH=1  and  Fosc=4.000 MHz
        movwf  SPBRG

        banksel        RCSTA
        bsf            RCSTA,SPEN ; Enable  the serial  port
        bcf            RCSTA,RX9   ; Disable9-bit  Receive
        bsf            RCSTA,CREN          ; Enable  continuous  receive

        banksel        TXSTA
        bcf            TXSTA,SYNC          ; Set up the port for asynchronous  operation
        bsf            TXSTA,TXEN          ; Transmit  enabled
        bsf            TXSTA,BRGH          ; High  baud  rate
        bcf            TXSTA,TX9   ; Disable9-bit  send

        banksel        PIE1            ; Enable  the Receive  interrupt
        bsf     PIE1, RCIE
        bcf     TRISC,RC6           ; Set RC6 to output Send Pin
        bsf     TRISC,RC7           ; Set RC7 to input Receive Pin


movlw 0x07
        movwf OPTION_REG                        ;Prescaler is assigned to the Timer0 module
                                                        ;Prescaler TMR0 (1:256)
        bsf             INTCON,GIE              ;Enable Global Interrupt
        bsf             INTCON,PEIE                     ;Enable peripheral interrupts
        bsf             INTCON,TMR0IE
        movlw 0x02
        movwf T1CON                             ;TMR1 Prescale (1:1),TMR1 Oscillator is
shut-off
                                                        ;External clock from pin
RC0/T1OSO/T1CKI (on the rising edge)
        banksel         T1CON
        bsf             T1CON,TMR1ON    ;Enables Timer1
        bsf             T1CON,2         ;disable syncronize external clock
        bsf T1CON,TMR1CS        ; enable external clock *********
        movlw Zero
        movwf PORTB                             ;initialize 7-Segments to Zero
        clrf    TMR0_Counter
        clrf    Timer1Counts
        movlw d'5' ; value of initial pulses per interrupt
        movwf speed
        movlw d'5'  ; value of range between speeds
        addwf speed,w
        movwf speed1
        movlw d'5'
        addwf speed1,w
        movwf speed2
        banksel         Counter
        clrf    Counter
;***************************************************************************
                BANKSEL TRISA
                CLRF    TRISA
                CLRF  TRISD
                CLRF  TRISC
                MOVLW       B'11110000'                    ; PORTB initially row bits as
output, column
                                                                   ; as input
                MOVWF       TRISB
                BCF              OPTION_REG, NOT_RBPU    ; Turn on all internal pull-ups of
PORTB

                BANKSEL ADCON1
                MOVLW   0X06
                MOVWF   ADCON1                    ;set PORTA as general Digital I/O PORT

                BANKSEL PORTB
                CLRF  PORTB
                movlw            Zero                                  ; Send Zero to output
                movwf PORTC
                BSF              PORTA,RA0
                BCF              INTCON, RBIF                          ; Initialize and enable port-
on-change
                BSF              INTCON, RBIE                          ; interrupt
                BSF              INTCON, GIE
;Initialize LCD
                Movlw 0x38              ;8-bit mode, 2-line display, 5x7 dot format
                Call    send_cmd
                Movlw 0x0e              ;Display on, Cursor Underline on, Blink off
                Call     send_cmd
                Movlw 0x02              ;Display  and cursor home
                Call     send_cmd
                Movlw 0x01              ;clear  display
                Call     send_cmd
                call     DrawStick1
            Call    DrawStick2
                Movlw 0x01              ;clear  display
                Call     send_cmd
;*****************************************************************************
*****
call delay
Movlw a'A'
  Call        send_char


Main
  call   Range_Test
        goto     Main                   ;Do it again
;*****************************************************************************
*****
DrawStick1                              ; Setting  the CGRAM address at which  we draw the stick
man
                Movlw 0x40              ; Here it is address 0x00
                Call     send_cmd
                Movlw 0X0E              ; Sending  data that implements  the Stick man
                Call     send_char
                Movlw 0X11
                Call     send_char
      Movlw     0X0E
            Call send_char
      Movlw     0X04
```

```
        Call        send_char
        Movlw       0X1F
        Call        send_char
        Movlw       0X04
        Call        send_char
                    Movlw 0X0A
                    Call      send_char
                    Movlw 0X11
                    Call      send_char
                    Return
;*****************************************************************************
*****
DrawStick2                            ; Setting the CGRAM address at which we draw the stick
man
                    Movlw 0x48        ; Here it is address 0x01
                    Call      send_cmd
                    Movlw 0X0E        ; Sending data that implements the Stick man
                    Call      send_char
                    Movlw 0X0A
                    Call      send_char
        Movlw       0X04
               Call send_char
        Movlw       0X15
        Call        send_char
        Movlw       0X0E
        Call        send_char
        Movlw       0X04
        Call        send_char
                    Movlw 0X0A
                    Call      send_char
                    Movlw 0X0A
                    Call      send_char
                    Return
;*****************************************************************************
*****
Page1
             Movlw         0x88                 ;Set display  address
                    Call      send_cmd
                    Movlw 0x00
                    Call    send_char
                    Movlw 0x88                  ;Set display  address
                    Call      send_cmd
                    call       delay
                    call       delay
                    Movlw    0x01
          Call    send_char
          return
Page2
                    Movlw 0x01                  ;clear  display
                    Call      send_cmd
Movlw 0x02                    ;Display  and cursor home
                    Call      send_cmd
return
Page3
                    Movlw 0x01
                    Call    send_char
                    call       delay
                    call       delay
                    Movlw    0x00
          Call    send_char
          return
send_cmd
```

```
		movwf	PORTD			; Refer to table 1 on Page 5 for review of this
subroutine
		bcf		PORTA,RA1
		bsf		PORTA,RA3
		nop
		bcf		PORTA,RA3
		bcf		PORTA,RA2
		call	delay
		return
;*****************************************************************************
*****
send_char
		movwf	PORTD			; Refer to table 1 on Page 5 for review of this
subroutine
		bsf		PORTA,RA1
		bsf		PORTA,RA3
		nop
		bcf		PORTA,RA3
		bcf		PORTA,RA2
		call	delay
		return
;*****************************************************************************
*****
delay
		movlw	0x80
		movwf	msd
		clrf	lsd
loop2
		decfsz	lsd,f
		goto	loop2
		decfsz	msd,f
endLcd
```

```
            goto     loop2
            return

;*****************************************************************************
; INTERRUPT SERVICE ROUTINE.
;*****************************************************************************
; Keypad press has been detected. Does the following:
;1.      Solves de-bouncing through software delay
;2.      Gets keypad pattern (location of the pressed button)
;3.      Converts location to table index
;4.      Access look-up table with index, gets 7-seg. Code, display on 7-segment.
;5.      Solves the port-on-change interrupt when button is released (2nd call delay):
;The code will enter the interrupt service routine two times for the same ;button,
;once when the button is pressed (change from 1 to 0), another when ;the button is released
(change from 0 to 1),
;therefore we insert the second delay such that the action of pressing/releasing the button will
occur inside ;the interrupt routine,
;and when it happens we will clear the flag only once ;and not enter the subroutine again for the
release action.
;*****************************************************************************
Range_Test
speed_range ; tests the range of the speed


t1       movfw Timer1Counts
         subwf speed,w
         btfss STATUS,C ; skip if speed < tmr1
         goto display1
         bcf STATUS,C


t2       movfw Timer1Counts
         subwf speed1,w
```

```
        btfss STATUS,C ; skip if speed1 < tmr1
        goto display2
        bcf STATUS,C

t3      movfw Timer1Counts
        subwf speed2,w
        btfss STATUS,C
        goto display3
        bcf STATUS,C
        goto display4

display1 ; displays the first emoji   ----------------------------------------------------------
call DrawStick1
        goto Finish
display2
call DrawStick1
        goto Finish

display3
call DrawStick2
        goto Finish

display4
call DrawStick2
        goto Finish



Finish
        Return
;*********************************************************
```

```
; TIMER0  RE-Initialize  and reset

RECEIVE

            movf    RCREG,w
            movf    RCREG,w
            movwf speed
    movlw  d'5'
    addwf speed,w
    movwf  speed1
    movlw  d'5'
    addwf speed1,w
    movwf  speed2





return

T0    ; the  code of the  timer  0 , it  restarts  the timer  until  it counts  1 second
            movlw  D'0'                                 ;..
            movwf TMR0                                  ;initialize  TMR0  with  60 counts to get interrupt
every  47.64ms

            incf      TMR0_Counter,F
            movf     TMR0_Counter,w
            sublw   D'21'
    btfss STATUS,Z
            goto     Continue
            movf     TMR1L,w
            movwf Timer1Counts
```

```
        clrf    TMR1L
        clrf    TMR1H
        clrf    TMR0_Counter
Continue
        bcf             INTCON,TMR0IF
        goto    POLL                    ;Check for another interrupt

;*********************************
ISR
        ;       CALL  DELAY
push
POLL
        btfsc   INTCON,TMR0IF                   ; Check for an TMR0 Interrupt
        goto    T0
   btfsc PIR1, RCIF     ; Check for a Receive interrupt
        call    RECEIVE



        ;       CALL  KPAD_RD
        ;       CALL  KP_CODE_CONV
        ;       CALL  DELAY

                ;MOVF           PORTB, W        ; READ PORTB VALUE.
                ;BCF            INTCON, RBIF    ; CLEAR INTERRUPT FLAG
                pop
        RETFIE

;**************************************************************************
; Function:
;       Gets the coordinates of the pressed keypad button and stores it
; Input:
;       Nibble Values from PORTB: initially high nibble of PORTB then low nibble
; Output:
;       Pressed button coordinates in the matrix in the form {Column, row} store in
;       KPAD_PAT
; REFER TO THE FLOWCHART ON PAGE 3 TO FULLY UNDERSTAND HOW THIS
; SUBROUTINE WORKS
;************************************************************************
KPAD_RD
        MOVF PORTB,W                ; Read Column
        ANDLW   B'11110000'         ; Ensure unwanted bits are suppressed
        MOVWF   KPAD_PAT
        BSF  STATUS, RP0            ; Set row as input, column as output.
        MOVLW   B'00001111'
        MOVWF   TRISB
        BCF  STATUS, RP0
        CLRF PORTB            ; Send Zero to output
        MOVF PORTB, W         ; Read Row
        ANDLW   B'00001111'         ; Ensure unwanted bits are suppressed
        IORWF   KPAD_PAT, 1
        BSF  STATUS, RP0
        MOVLW   B'11110000'   ; Restore row as output, column as input
        MOVWF   TRISB
        BCF     STATUS, RP0
        CLRF  PORTB                 ; Send Zero to output
        RETURN
;************************************************************************
; Function:
;       Converts keypad pattern held in KPAD_PAT to an index KPAD_ADD which we use
;       to access the look-up table.
; INPUT:
;       KPAD_PAT which holds the location in terms of {Column,Row} of the pressed
;       key
; OUTPUT:
;       A value in KPAD_ADD in the range of 0 to 15 which is the index to be used in the
;       look-up table
; REFER TO THE FLOWCHART ON PAGE 5 TO FULLY UNDERSTAND HOW THIS
; SUBROUTINE
;***************************************************************************
KP_CODE_CONV
            CLRF  KPAD_ADD            ; Initially Base index is 0
KP0         BTFSC       KPAD_PAT, 4               ; Is Column1?
            GOTO KP1
            GOTO ROW_FIND
KP1         BTFSC       KPAD_PAT, 5               ; Is Column2?
            GOTO KP2
            MOVLW       B'00000100'         ; Base index is 4
            ADDWF       KPAD_ADD, 1
            GOTO ROW_FIND
KP2         BTFSC       KPAD_PAT, 6               ; Is Column3?
            GOTO KP3
            MOVLW       B'00001000'         ; Base index is 8
            ADDWF       KPAD_ADD, 1
            GOTO ROW_FIND
KP3
            BTFSC       KPAD_PAT, 7               ; Is Column4?
            GOTO KEEP                 ; If no button was pressed, display last
                                                  ; character on 7-segment display
            MOVLW       B'00001100'         ; Base index is 12
            ADDWF       KPAD_ADD, 1
```

```
ROW_FIND
          BTFSC     KPAD_PAT, 0          ; Is Row1?
          GOTO RF1
          GOTO      KEYPAD_OP
RF1       BTFSC     KPAD_PAT, 1          ; Is Row2?
          GOTO RF2
          MOVLW     B'00000001'        ; Add 1 to the base index
          ADDWF     KPAD_ADD, 1
          GOTO      KEYPAD_OP

RF2       BTFSC     KPAD_PAT, 2          ; Is Row3?
          GOTO RF3
          MOVLW     B'00000010'        ; Add 2 to the base index
          ADDWF     KPAD_ADD, 1
          GOTO      KEYPAD_OP

RF3       BTFSC     KPAD_PAT, 3          ; Is Row4?
          GOTO KEEP           ; If no button was pressed, display last
                                         ; character on 7-segment display
          MOVLW     B'00000011'        ; Add 3 to the base index
          ADDWF     KPAD_ADD, 1

KEYPAD_OP
          MOVF KPAD_ADD, 0
          CALL  KP_TABLE      ; Access table with index
          MOVWF     KPAD_CHAR            ; Save character
          MOVWF     PORTC                ; Display Character
          GOTO FIN
KEEP
```

```
            MOVF KPAD_CHAR, W
            MOVWF      PORTC                    ; If no button was pressed, display
last
                                                ; character on 7-segment display
FIN         RETURN
;******************************************************************************
;This table contains the common cathode 7-segment representations of the numbers 0
;to 9 and the characters 'A' to 'f' found on the keypad.
;As seen below, the table lists the numbers in the order Col1, Col2 and son on.
;******************************************************************************
KP_TABLE
            ADDWF       PCL,  1
            RETLW      One                      ;'1'      COLUMN1
            RETLW      Four               ;'4'
            RETLW      Seven              ;'7'
            RETLW   LetterA               ;'A'
            RETLW      Two                ;'2'      COLUMN2
            RETLW      Five               ;'5'
            RETLW      Eight              ;'8'
            RETLW      Zero               ;'0'
            RETLW      Three              ;'3'      COLUMN3
            RETLW      Six                      ;'6'
            RETLW      Nine               ;'9'
            RETLW      LetterB            ;'B'
            RETLW      LetterF            ;'F'      COLUMN4
            RETLW      LetterE            ;'E'
            RETLW      LetterD                  ;'D'
            RETLW      LetterC                  ;'C'
;******************************************************************************
; Delay subroutine
; Delay of approx. 10ms which is more than enough for de-bouncing
```

```
;**************************************************************************
DELAY
            MOVLW       0X20
            MOVWF       DELCNTR1
            CLRF  DELCNTR2
LOOP2
            DECFSZ      DELCNTR2,F
            GOTO LOOP2
            DECFSZ      DELCNTR1,F
ENDLCD
            GOTO        LOOP2
      RETURN
            END
;**************************************************************************
```

Code of the Keypad part

```
__CONFIG
_DEBUG_OFF&_CP_OFF&_WRT_HALF&_CPD_OFF&_LVP_OFF&_BODEN_OFF&_PW
RTE_OFF&_WDT_OFF&_XT_OSC
;****************************************************************************
***
#INCLUDE "P16F877A.INC"

CBLOCK      0x20
DELCNTR1          ; Used in generating 10 ms delay
DELCNTR2
KPAD_PAT          ; Holds the pattern retrieved from keypad
KPAD_ADD          ; Holds keypad address to lookup table (generated from
                        ; KPAD_PAT to get KPAD_CHAR)
```

```
KPAD_CHAR                   ; Holds the 7-segment representation of the most recent
                            ; character pressed on keypad
;tempChar
;charCount
lsd                         ;lsd and msd are used in delay loop calculation
msd
ENDC
;*****************************************************************************
***
Zero    equ       B'00111111' ; 7-Segment Code for Zero
One             equ       B'00000110'        ; 7-Segment Code for One
Two             equ   B'01011011' ; 7-Segment Code for Two
Three           equ       B'01001111' ; 7-Segment Code for Three
Four            equ   B'01100110'    ; 7-Segment Code for Four
Five            equ   B'01101101'    ; 7-Segment Code for Five
Six     equ   B'01111101' ; 7-Segment Code for Six
Seven           equ       B'00000111' ; 7-Segment Code for Seven
Eight           equ   B'01111111'    ; 7-Segment Code for Eight
Nine            equ   B'01101111'    ; 7-Segment Code for Nine
LetterA                 equ             B'01110111'   ; 7-Segment Code for A
LetterB                 equ             B'01111100'   ; 7-Segment Code for B
LetterC                 equ             B'01011000'   ; 7-Segment Code for C
LetterD                 equ             B'01011110'   ; 7-Segment Code for D
LetterE         equ             B'01111001'   ; 7-Segment Code for E
LetterF         equ             B'01110001'   ; 7-Segment Code for F
;****************************************************************************
; START OF EXECUTABLE CODE
;****************************************************************************
        ORG   0x00
        GOTO INITIAL
;****************************************************************************
```

```
; INTERRUPT  VECTOR
;***************************************************************************
        ORG   0x04
        GOTO KPAD_TO_7SEG
;***************************************************************************
INITIAL

        movlw  D'25'            ; This  sets the baud rate to 9600
        banksel         SPBRG                   ; assuming  BRGH=1 and  Fosc=4.000 MHz
        movwf  SPBRG
        banksel         RCSTA
        bsf             RCSTA,SPEN ; Enable  the serial port
        bcf             RCSTA,RX9   ; Disable9-bit  Receive
        bsf             RCSTA,CREN          ; Enable  continuous  receive
        banksel         TXSTA
        bcf             TXSTA,SYNC          ; Set up the port for asynchronous  operation
        bsf             TXSTA,TXEN          ; Transmit  enabled
        bsf             TXSTA,BRGH          ; High  baud rate
        bcf             TXSTA,TX9   ; Disable9-bit  send

;***************************************************************************
                BANKSEL TRISA
                CLRF    TRISA
                CLRF  TRISD
                CLRF  TRISC
                MOVLW       B'11110000'                         ; PORTB initially  row bits as
output,  column
                                                                        ; as input
                MOVWF       TRISB
                BCF             OPTION_REG, NOT_RBPU    ; Turn on all internal  pull-ups  of
PORTB
```

```
          BANKSEL ADCON1
          MOVLW   0X06
          MOVWF   ADCON1                 ;set PORTA as general Digital I/O PORT

          BANKSEL PORTB
          CLRF  PORTB
          movlw         Zero                          ; Send Zero to output
          movwf PORTC
          BSF           PORTA,RA0
          BCF           INTCON, RBIF                  ; Initialize and enable port-
on-change
          BSF           INTCON, RBIE                  ; interrupt
          BSF           INTCON, GIE
;Initialize LCD
          Movlw 0x38          ;8-bit mode, 2-line display, 5x7 dot format
          Call    send_cmd
          Movlw 0x0e          ;Display on, Cursor Underline on, Blink off
          Call    send_cmd
          Movlw 0x02          ;Display and cursor home
          Call    send_cmd
          Movlw 0x01          ;clear display
          Call    send_cmd
          call    DrawStick1
      Call    DrawStick2
          Movlw 0x01          ;clear display
          Call    send_cmd
;*****************************************************************************
*****
call delay
Movlw a'A'
```

```
  Call        send_char

Main


            movf    KPAD_CHAR,w
            sublw   One
            btfsc   STATUS,Z
            call    Page1
            movf    KPAD_CHAR,w
            sublw   Two
            btfsc   STATUS,Z
            call    Page2
            movf    KPAD_CHAR,w
            sublw   Three
            btfsc   STATUS,Z
            call    Page3
;           movf    KPAD_CHAR,w
;           sublw   Four
;           btfsc   STATUS,Z
;           call    Page4
;           movf    KPAD_CHAR,w
;           sublw   Five
;           btfsc   STATUS,Z
;           call    Page5
;           movf    KPAD_CHAR,w
;           sublw   Six
;           btfsc   STATUS,Z
;           call    Page6
;           movf    KPAD_CHAR,w
```

```
;	sublw	Seven
;	btfsc	STATUS,Z
;	call	Page7
;	movf	KPAD_CHAR,w
;	sublw	Eight
;	btfsc	STATUS,Z
;	call	Page8
;	movf	KPAD_CHAR,w
;	sublw	Nine
;	btfsc	STATUS,Z
;	call	Page9
;	movf	KPAD_CHAR,w
;	sublw	LetterA
;	btfsc	STATUS,Z
;	call	Page10
;	movf	KPAD_CHAR,w
;	sublw	LetterB
;	btfsc	STATUS,Z
;	call	Page11
;	movf	KPAD_CHAR,w
;	sublw	LetterC
;	btfsc	STATUS,Z
;	call	Page12
;	movf	KPAD_CHAR,w
;	sublw	LetterD
;	btfsc	STATUS,Z
;	call	Page13
;	movf	KPAD_CHAR,w
;	sublw	LetterE
;	btfsc	STATUS,Z
```

```
;           call    Page14
;           movf    KPAD_CHAR,w
;           sublw   LetterF
;           btfsc   STATUS,Z
;           call    Page15
;           movf    KPAD_CHAR,w
;           sublw   Zero
;           btfsc   STATUS,Z
;           call    Page16

            goto    Main                    ;Do it again
;******************************************************************************
*****
DrawStick1                                  ; Setting the CGRAM address at which we draw the stick
man
                    Movlw 0x40              ; Here it is address 0x00
                    Call    send_cmd
                    Movlw 0X0E              ; Sending data that implements the Stick man
                    Call    send_char
                    Movlw 0X11
                    Call    send_char
        Movlw       0X0E
            Call    send_char
        Movlw       0X04
        Call        send_char
        Movlw       0X1F
        Call        send_char
        Movlw       0X04
        Call        send_char
                    Movlw 0X0A
                    Call    send_char
                    Movlw 0X11
                    Call     send_char
                    Return

;*****************************************************************************
*****
DrawStick2                       ; Setting  the CGRAM address at which  we draw the stick
man
                    Movlw 0x48           ; Here  it  is  address 0x01
                    Call     send_cmd
                    Movlw 0X0E           ; Sending  data that implements  the Stick  man
                    Call     send_char
                    Movlw 0X0A
                    Call     send_char
        Movlw   0X04
            Call send_char
        Movlw   0X15
        Call    send_char
        Movlw   0X0E
        Call    send_char
        Movlw   0X04
        Call    send_char
                    Movlw 0X0A
                    Call     send_char
                    Movlw 0X0A
                    Call     send_char
                    Return
;*****************************************************************************
*****
Page1
 ;          Movlw      0x88            ;Set display  address
;                   Call     send_cmd
;                   Movlw 0x00
;                   Call    send_char
;                   Movlw 0x88              ;Set display address
;                   Call     send_cmd
;                   call     delay
;                   call     delay
;                   Movlw   0x01
 ;        Call    send_char
          movlw  d'5'
          banksel           TXREG
          movwf  TXREG                      ; Send a next character out the serial port
          banksel           TXSTA
L1        btfss    TXSTA,TRMT
          goto     L1

          return
Page2
movlw  d'10'
          banksel           TXREG
          movwf  TXREG                      ; Send a next character out the serial port
          banksel           TXSTA
L2        btfss    TXSTA,TRMT
          goto     L2

          ;         Movlw 0x01              ;clear display
          ;;        Call     send_cmd
;Movlw              0x02              ;Display and cursor home
          ;         Call     send_cmd
return
```

```
Page3
        movlw  d'15'
   banksel     TXREG
        movwf  TXREG                ; Send a next character out the serial port
        banksel        TXSTA
L3      btfss  TXSTA,TRMT
        goto   L3

        ;      Movlw 0x01
        ;      Call   send_char
        ;      call      delay
        ;      call      delay
        ;      Movlw    0x00
    ;   Call   send_char
        return
;Page4
;Movlw         0x02          ;Display  and cursor home
;              Call     send_cmd
;              Movlw 0x01           ;clear  display
;              Call     send_cmd
;return
;Page5
 ;      Movlw    0x00
  ;    Call        send_char
;return
;Page6
;Movlw         0x02          ;Display  and cursor home
;              Call     send_cmd
;              Movlw 0x01           ;clear  display
;              Call     send_cmd
```

```
;return
;Page7
 ;         Movlw    0x00
;         Call         send_char
;return
;Page8
;Movlw            0x02                ;Display  and cursor home
;                     Call      send_cmd
;                     Movlw 0x01                 ;clear  display
;                     Call      send_cmd
;return
;Page9
 ;         Movlw    0x00
 ;        Call         send_char
;return
;Page10
;Movlw            0x02                ;Display  and cursor home
;                     Call      send_cmd
;                     Movlw 0x01                 ;clear  display
;                     Call      send_cmd
;return
;Page11
;          Movlw    0x00
;         Call         send_char
;return
;Page12
;Movlw            0x02                ;Display  and cursor home
;                     Call      send_cmd
;                     Movlw 0x01                 ;clear  display
;                     Call      send_cmd
```

```
;return
;Page13
;         Movlw    0x00
;        Call         send_char
;return
;Page14
;Movlw           0x02               ;Display  and cursor home
;                     Call      send_cmd
;                     Movlw 0x01               ;clear  display
;                     Call      send_cmd
;return
;Page15
  ;        Movlw    0x00
 ;        Call         send_char
;return
;Page16
;Movlw           0x02               ;Display  and cursor home
;                     Call      send_cmd
;                     Movlw 0x01               ;clear  display
;                     Call      send_cmd
;return
;****************************************************************************
*****
send_cmd
                     movwf PORTD                       ; Refer  to table  1 on Page 5 for review  of this
subroutine
                     bcf                  PORTA,RA1
                     bsf                  PORTA,RA3
                     nop
                     bcf                  PORTA,RA3
                     bcf                  PORTA,RA2
```

```
                call    delay
                return
;*****************************************************************************
*****
send_char
                movwf   PORTD                   ; Refer to table 1 on Page 5 for review of this
subroutine
                bsf             PORTA,RA1
                bsf             PORTA,RA3
                nop
                bcf             PORTA,RA3
                bcf             PORTA,RA2
                call    delay
                return
;*****************************************************************************
*****
delay
                movlw   0x80
                movwf   msd
                clrf    lsd
loop2
                decfsz  lsd,f
                goto    loop2
                decfsz  msd,f
endLcd
                goto    loop2
                return
KPAD_TO_7SEG
                CALL    DELAY
                CALL    KPAD_RD
                CALL    KP_CODE_CONV
                CALL  DELAY
                MOVF PORTB, W        ; READ PORTB VALUE.
                BCF         INTCON, RBIF   ; CLEAR INTERRUPT FLAG
                RETFIE

;***************************************************************************
; Function:
;       Gets the coordinates of the pressed keypad button and stores it
; Input:
;       Nibble Values from PORTB: initially high nibble of PORTB then low nibble
; Output:
;       Pressed button coordinates in the matrix in the form {Column, row} store in
;       KPAD_PAT
; REFER TO THE FLOWCHART ON PAGE 3 TO FULLY UNDERSTAND HOW THIS
; SUBROUTINE WORKS
;***************************************************************************
KPAD_RD
                MOVF PORTB,W                    ; Read Column
                ANDLW       B'11110000'         ; Ensure unwanted bits are suppressed
                MOVWF       KPAD_PAT
                BSF    STATUS, RP0              ; Set row as input, column as output.
                MOVLW       B'00001111'
                MOVWF       TRISB
                BCF    STATUS, RP0
                CLRF PORTB               ; Send Zero to output
                MOVF PORTB, W            ; Read Row
                ANDLW       B'00001111'         ; Ensure unwanted bits are suppressed
                IORWF       KPAD_PAT, 1
                BSF    STATUS, RP0
                MOVLW       B'11110000'    ; Restore row as output, column as input
                MOVWF       TRISB
                BCF         STATUS, RP0
                CLRF  PORTB                     ; Send Zero to output
                RETURN
;***************************************************************************
; Function:
;       Converts keypad pattern held in KPAD_PAT to an index KPAD_ADD which we use
;       to access the look-up table.
; INPUT:
;       KPAD_PAT which holds the location in terms of {Column,Row} of the pressed
;       key
; OUTPUT:
;       A value in KPAD_ADD in the range of 0 to 15 which is the index to be used in the
;       look-up table
; REFER TO THE FLOWCHART ON PAGE 5 TO FULLY UNDERSTAND HOW THIS
; SUBROUTINE
;***************************************************************************
KP_CODE_CONV
                CLRF  KPAD_ADD          ; Initially Base index is 0
KP0             BTFSC       KPAD_PAT, 4             ; Is Column1?
                GOTO KP1
                GOTO ROW_FIND
KP1             BTFSC       KPAD_PAT, 5             ; Is Column2?
                GOTO KP2
                MOVLW       B'00000100'         ; Base index is 4
                ADDWF       KPAD_ADD, 1
                GOTO ROW_FIND
KP2             BTFSC       KPAD_PAT, 6             ; Is Column3?
                GOTO KP3
                MOVLW       B'00001000'         ; Base index is 8
```

```
                    ADDWF         KPAD_ADD, 1
                    GOTO ROW_FIND
KP3
                    BTFSC         KPAD_PAT, 7                   ; Is Column4?
                    GOTO KEEP                   ; If no button was pressed, display last
                                                                ; character on 7-segment display
                    MOVLW         B'00001100'             ; Base index is 12
                    ADDWF         KPAD_ADD, 1

ROW_FIND
                    BTFSC         KPAD_PAT, 0                   ; Is Row1?
                    GOTO RF1
                    GOTO          KEYPAD_OP
RF1                 BTFSC         KPAD_PAT, 1                   ; Is Row2?
                    GOTO RF2
                    MOVLW         B'00000001'             ; Add 1 to the base index
                    ADDWF         KPAD_ADD, 1
                    GOTO          KEYPAD_OP

RF2                 BTFSC         KPAD_PAT, 2                   ; Is Row3?
                    GOTO RF3
                    MOVLW         B'00000010'             ; Add 2 to the base index
                    ADDWF         KPAD_ADD, 1
                    GOTO          KEYPAD_OP

RF3                 BTFSC         KPAD_PAT, 3                   ; Is Row4?
                    GOTO KEEP                   ; If no button was pressed, display last
                                                                ; character on 7-segment display
                    MOVLW         B'00000011'             ; Add 3 to the base index
                    ADDWF         KPAD_ADD, 1
```

```
KEYPAD_OP
                    MOVF KPAD_ADD, 0
                    CALL  KP_TABLE              ; Access table with index
                    MOVWF         KPAD_CHAR                    ; Save character
                    MOVWF         PORTC                        ; Display Character
                    GOTO FIN
KEEP
                    MOVF KPAD_CHAR, W
                    MOVWF         PORTC                        ; If no button was pressed, display
last
                                                               ; character on 7-segment display
FIN                 RETURN
;***************************************************************************
;This table contains the common cathode 7-segment representations of the numbers 0
;to 9 and the characters 'A' to 'f' found on the keypad.
;As seen below, the table lists the numbers in the order Col1, Col2 and son on.
;***************************************************************************
KP_TABLE
                    ADDWF          PCL,  1
                    RETLW          One                         ;'1'      COLUMN1
                    RETLW          Four               ;'4'
                    RETLW          Seven              ;'7'
                    RETLW    LetterA                  ;'A'
                    RETLW          Two                ;'2'      COLUMN2
                    RETLW          Five               ;'5'
                    RETLW          Eight              ;'8'
                    RETLW          Zero               ;'0'
                    RETLW          Three              ;'3'      COLUMN3
                    RETLW          Six                         ;'6'
                    RETLW          Nine               ;'9'
```

```
        RETLW   LetterB         ;'B'
        RETLW   LetterF         ;'F'    COLUMN4
        RETLW   LetterE         ;'E'
        RETLW   LetterD                 ;'D'
        RETLW   LetterC                 ;'C'
;***************************************************************************
; Delay  subroutine
; Delay  of approx. 10ms  which  is  more  than  enough  for de-bouncing
;***************************************************************************
DELAY
        MOVLW   0X20
        MOVWF   DELCNTR1
        CLRF  DELCNTR2
LOOP2
        DECFSZ  DELCNTR2,F
        GOTO LOOP2
        DECFSZ  DELCNTR1,F
ENDLCD
        GOTO    LOOP2
    RETURN
        END
;***************************************************************************
```

References

[1] Microchip, "PIC16F87XA Data Sheet".