1. Description:

This project is to make a second timer, using the 4-digit 7 segment LED display, range to 60mins. Increasement per sec.

1. Timer display and control:

IO configuration

* PORTB connect to button 0 and 1. (INT0, INT1)
* PORTC display 7-segment LED number.
* PORTD choose the digit of display.

User input

* INT0 (button 0)used as control start/resume and stop function.
* INT1 (button 1)used as reset function, reset the timer to 00:00

Time count

* TMR0 used for counting 1 second and TMR0 interrupt is used.

Display

* The 1st and 2nd LED use to show mins, range from 00 to 60. Additional dot is add at 2nd digit.
* The 3rd and 4th LED show seconds.

1. Program flow:
2. Keep looping to display four-digit number.
3. When INT0 is detected, toggle TMR0, timer start.
4. If TMR0 overflow, increase the register that store the 4th digit BCD by 1.
5. Check if 4th digit is 10. If 10 then increase the 3rd digit, check 3rd digit if equal 6, and so on.
6. If TMR0 is ON and INT0 is detected, stop TMR0.
7. Whenever TMR0 on or not, INT1 always reset the timer and display.
8. Problem solved:
9. As I want to use 4-digit 7-segment LED, so polling to check button input will be inefficient and input may be missed. Interrupt driven is better choice.
10. As BCD to 7-segment routine is keep calling, so when a interrupt is occur, it will immediately jump to 0x0008. While the stack pointer still pointing to the address of the routine, it increases by 1 as interrupt occurs, and will not pop out address. So, there is a problem that stack will overflow. I decided to sacrifice some program memory and removed all the call and return function, even retfie is also removed. The GIE will all be controlled by program.

Appendix --- code explanation blue: code orange: registers variable green: label

LED display:

Display:

movff in\_1, input

goto bcd\_7seg\_1

D\_1 movwf PORTC

clrf PORTD

movff in\_2, input

goto bcd\_7seg\_2

D\_2 addlw b'10000000'

movwf PORTC

incf PORTD

movff in\_3, input

goto bcd\_7seg\_3

D\_3 movwf PORTC

incf PORTD

movff in\_4, input

goto bcd\_7seg\_4

D\_4 movwf PORTC

incf PORTD

GOTO Display

Similar to the assignment, but I removed the delay. 4-digit BCD in store in in\_1, in\_2, in\_3 and in\_4 registers.

Digit increment function:

|  |  |
| --- | --- |
| inc4: bcf T0CON,TMR0ON  incf in\_4,w  DAW  andlw b'00001111'  movwf in\_4  BZ inc3  bcf INTCON,TMR0IF  bsf INTCON,GIE  goto Again  inc3: incf in\_3  decf cs6  bz inc2  bcf INTCON,TMR0IF  bsf INTCON,GIE  goto Again | inc2: movlw d'6'  movwf cs6  clrf in\_3  incf in\_2,w  DAW  ANDLW b'00001111'  movwf in\_2  BZ inc1  bcf INTCON,TMR0IF  bsf INTCON,GIE  goto Again  inc1: incf in\_1  decf cm6  btfsc STATUS,Z  goto Main  bcf INTCON,TMR0IF  bsf INTCON,GIE  goto Again |

When TMR0 interrupt occur, the 4th digit will increase by 1, after that decimal adjustment is done, next the first nibble is masked. After that, check if is 0. If 0 then increase the 3rd digit. I use cs6 to check if the 3rd digit is 6 by decreasing it by 1 every increasement of 3rd digit.

Same method is used at inc2 and inc1. In inc1, when cm6 is decreased to 0, that means the counter counted to 60 mins, then the timer will start from 00:00.

TMR0, INT0, INT1 configuration:

intcon:

movlw b'00000100'

movwf T0CON

movlw b'01100000'

movwf INTCON2

movlw b'00001000'

movwf INTCON3

movlw b'10110000'

movwf INTCON

goto Restart

Again:

movlw 0x85 ;movlw 0xAA

movwf TMR0H

movlw 0xEE

movwf TMR0L

bsf T0CON,TMR0ON

As I want to generate 1s, I use 1:32 prescaler, and set the TMR0 as 0x85EE (-31250)

Interrupt check:

|  |  |
| --- | --- |
| Checkint:  POP  btfss INTCON,INT0IF  goto inc4  btfss T0CON,TMR0ON  goto Resume  btfsc INTCON3,INT1IF  goto Restart  goto Stop | Stop:  bcf T0CON,TMR0ON  bcf INTCON,INT0IF  bsf INTCON,GIE  goto Display |
| Resume:  bcf INTCON,INT0IF  bsf INTCON,GIE  goto Again |
| Restart:  bcf T0CON,TMR0ON  clrf In\_1  clrf in\_2  clrf in\_3  clrf in\_4  bcf INTCON3,INT1IF  bsf INTCON,GIE  goto Display |

When interrupt occur, the program goto Checkint.

If INT0 flag is 0, that means timer is working, goto inc4.

If TMR0 is not using , that means the timer stop, goto Resume.

If INT1 is interrupting, goto Restart.

Otherwise, goto Stop.

Stop:

Stop TMR0, just display number.

Resume:

Start TMR0, re-calculate 1 sec.

Restart:

Clear all the digit and stop the timer, wait for INT0 to start the timer.

Whole code:

LIST P=18F4520

#include <P18F4520.INC>

CONFIG OSC = XT

CONFIG WDT = OFF

CONFIG LVP = OFF

cblock 0x10

input, DELAY\_L, DELAY\_H, in\_1, in\_2, in\_3, in\_4,Eint,Wint,cs6,cm6,stopf

endc

ORG 0x00

goto Main

ORG 0x0008

goto Checkint

ORG 0x50

Main: movlw 0x0F

movwf ADCON1

clrf TRISD

clrf PORTD

clrf TRISC

clrf PORTC

setf TRISB

clrf in\_1

clrf in\_2

clrf in\_3

clrf in\_4

movlw d'6'

movwf cs6

movwf cm6

intcon:

movlw b'00000100'

movwf T0CON

movlw b'01100000'

movwf INTCON2

movlw b'00001000'

movwf INTCON3

movlw b'10110000'

movwf INTCON

goto Restart

Again:

movlw 0x85 ;movlw 0xAA

movwf TMR0H

movlw 0xEE

movwf TMR0L

bsf T0CON,TMR0ON

Display:

movff in\_1, input

goto bcd\_7seg\_1

D\_1 movwf PORTC

clrf PORTD

movff in\_2, input

goto bcd\_7seg\_2

D\_2 addlw b'10000000'

movwf PORTC

incf PORTD

movff in\_3, input

goto bcd\_7seg\_3

D\_3 movwf PORTC

incf PORTD

movff in\_4, input

goto bcd\_7seg\_4

D\_4 movwf PORTC

incf PORTD

GOTO Display

Checkint:

POP

btfss INTCON,INT0IF

goto inc4

btfss T0CON,TMR0ON

goto Resume

btfsc INTCON3,INT1IF

goto Restart

goto Stop

Stop:

bcf T0CON,TMR0ON

bcf INTCON,INT0IF

bsf INTCON,GIE

goto Display

Resume:

bcf INTCON,INT0IF

bsf INTCON,GIE

goto Again

Restart:

bcf T0CON,TMR0ON

clrf in\_1

clrf in\_2

clrf in\_3

clrf in\_4

bcf INTCON3,INT1IF

bsf INTCON,GIE

goto Display

ORG 0x300

inc4: bcf T0CON,TMR0ON

incf in\_4,w

DAW

andlw b'00001111'

movwf in\_4

BZ inc3

bcf INTCON,TMR0IF

bsf INTCON,GIE

goto Again

inc3: incf in\_3

decf cs6

bz inc2

bcf INTCON,TMR0IF

bsf INTCON,GIE

goto Again

inc2: movlw d'6'

movwf cs6

clrf in\_3

incf in\_2,w

DAW

ANDLW b'00001111'

movwf in\_2

BZ inc1

bcf INTCON,TMR0IF

bsf INTCON,GIE

goto Again

inc1: incf in\_1

decf cm6

btfsc STATUS,Z

goto Main

bcf INTCON,TMR0IF

bsf INTCON,GIE

goto Again

bcd\_7seg\_1:

MOVLW low bcd\_table

MOVWF TBLPTRL

MOVLW high bcd\_table

MOVWF TBLPTRH

MOVLW upper bcd\_table

MOVWF TBLPTRU

MOVF input, W

ADDWF TBLPTRL, F

MOVLW 0

ADDWFC TBLPTRH

ADDWFC TBLPTRU

TBLRD\*

MOVF TABLAT, W

goto D\_1

bcd\_7seg\_2:

MOVLW low bcd\_table

MOVWF TBLPTRL

MOVLW high bcd\_table

MOVWF TBLPTRH

MOVLW upper bcd\_table

MOVWF TBLPTRU

MOVF input, W

ADDWF TBLPTRL, F

MOVLW 0

ADDWFC TBLPTRH

ADDWFC TBLPTRU

TBLRD\*

MOVF TABLAT, W

goto D\_2

bcd\_7seg\_3:

MOVLW low bcd\_table

MOVWF TBLPTRL

MOVLW high bcd\_table

MOVWF TBLPTRH

MOVLW upper bcd\_table

MOVWF TBLPTRU

MOVF input, W

ADDWF TBLPTRL, F

MOVLW 0

ADDWFC TBLPTRH

ADDWFC TBLPTRU

TBLRD\*

MOVF TABLAT, W

goto D\_3

bcd\_7seg\_4:

MOVLW low bcd\_table

MOVWF TBLPTRL

MOVLW high bcd\_table

MOVWF TBLPTRH

MOVLW upper bcd\_table

MOVWF TBLPTRU

MOVF input, W

ADDWF TBLPTRL, F

MOVLW 0

ADDWFC TBLPTRH

ADDWFC TBLPTRU

TBLRD\*

MOVF TABLAT, W

goto D\_4

bcd\_table ORG 0x500

db 0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F

End