Date: 28/02/2021

| | |
|---|---|
| Student's name | Anuj Shah |
| Roll Number | 18104B0024 |
| Name of Professor | Professor Mohit Gujar |

| | |
|---|---|
| Experiment number | 4 |
| Experiment title | Delay using timer registers |
| Hardware requirement | - |
| Software requirement | Keil uVision5 |

| | |
|---|---|
| Aim | To write a program that will cause 8 LEDs connected to port-0 of our 8051 microcontroller to blink on-and-off, in an infinite loop, with a 1 second delay between each on-state and off-state. |

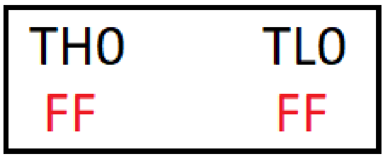
Theory

**Timer register calculations**

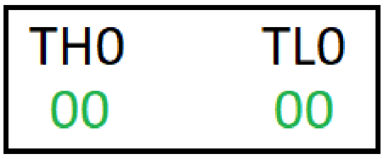
8051 consists of two pairs of timer registers:

- timer zero: TL0 and TH0
  this pair is selected by the instruction "MOV TMOD, #01H"
- timer one: TL1 and TH1
  this pair is selected by the instruction "MOV TMOD, #10H"

For simplicity, let's assume that we only work with timer zero (TL0, TH0). Now, we can generate any delay from 1.085us to 71ms, just by changing the values stored in the TL0 and TH0 registers.



1.085 microseconds (minimum delay)



71 milliseconds (maximum delay)

In order to generate a delay of *Y* microseconds, we must first calculate *X*, using the formula:

$$X = 65536 - \frac{Y\ \mu s}{1.085\ \mu s}$$

Then, we must convert *X* from decimal to hexadecimal, and store the lower byte in TL0, and the higher byte in TH0.

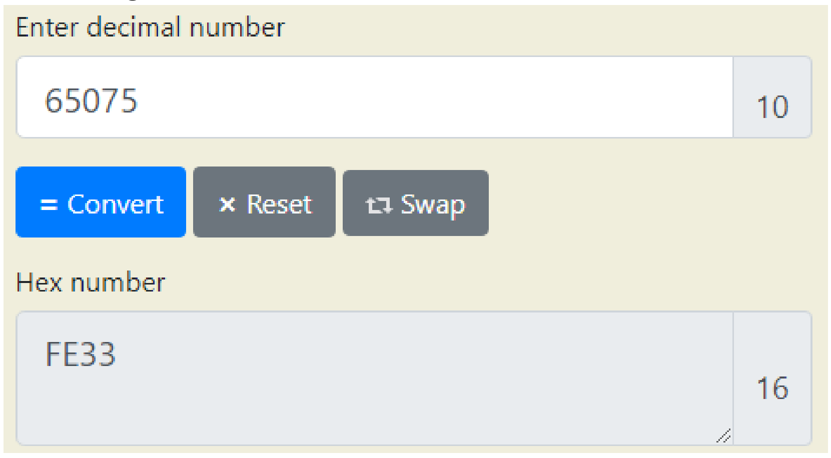
For example, suppose we want to generate a delay of 0.5 milliseconds. Well, 1 millisecond = $10^3$ microseconds. Thus,

$Y = 0.5 * 10^3$

and

$$X = 65536 - (0.5 * 10^3)/(1.085) \approx 65075$$

Now, using the online Decimal to Hexadecimal converter, we find that:



Enter decimal number

65075 | 10

Hex number

FE33 | 16

Thus, to generate a 0.5 ms delay, we need to set timer zero to:

- TH0 = FE
- TL0 = 33

**What about longer delays?**
Using the timer registers alone, we can only generate the following range of delays:

- minimum delay = 1.085us (T0 = 0xFFFF)
- maximum delay = 71ms (T0 = 0x0000)

Now, the minimum delay (1.085us) is an absolute lower limit; it is based on the fact that the 8051 has a machine cycle which lasts for 1.085us. Thus, we cannot generate delays shorter than this.

However, there is an ingenious way to generate delays longer than 71ms - we need to use looping! For example, to generate a delay of 1 second (= 1000ms), all we need to do is generate a delay of 50ms, and repeat this delay 20 times:

50 msec * 20 = 1000 msec = 1 second

Algorithm / Flowchart

***MAIN routine:***

1. The instruction "MOV P0,#0FFH" causes all the pins of port 0 to become activated. Physically, this means that all the LEDs will turn on.
2. "ACALL DELAY" calls the DELAY subroutine, producing a delay of around 1 second.
3. The instruction "MOV P0,#00H" causes all the pins of port 0 to become disactivated. Physically, this means that all the LEDs will turn off.
4. "ACALL DELAY" calls the DELAY subroutine, again producing a delay of around 1 second.
5. "JMP MAIN" causes the whole MAIN routine to be executed again.

Thus, we see that the MAIN routine produces an infinite loop, in which all the LEDs are on for 1 seconds, and then they are off for 1 second. We thus get a blinking effect.

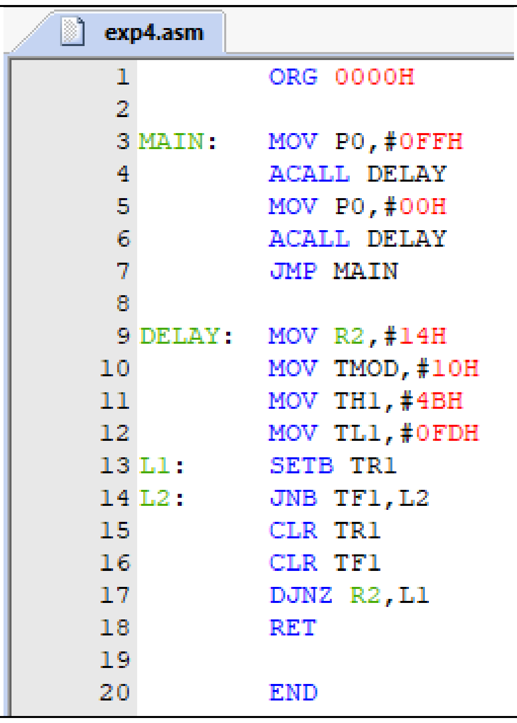
***DELAY subroutine:***

1. The instruction "MOV R2,#14H" causes the hex number 0x14 (decimal = 20) to be loaded into register 2.
2. The instruction "MOV TMOD,#10H" selects timer1.
3. The instructions "MOV TH1,#4BH" and "MOV TL1,#0FDH" together load the hexadecimal number 0x4BFD (decimal = 19453).
4. The instruction "SETB TR1" makes TR1=1. This starts the timer, causing the timer flag TF1 to become high for a certain duration (TF1=1).
5. The instruction "JNB TF1,L2" will keep executing, as long as the timer flag (TF1) is high.
6. The instructions "CLR TR1" and "CLR TF1" will cause TR1 = TF1 = 0.
7. The instruction "DJNZ R2,L1" will decrement the value of R2 by 1, and will jump back to step 4 if R2 is not zero.
8. The "RET" statement lets us leave the DELAY subroutine, and go back to the MAIN routine.

The DELAY subroutine can be explained in terms of the 50 msec delay produced by the timer registers (loaded with 0x4BFD); and the fact that we repeat this 50 msec delay 20 times using register R2 (loaded with 0x14, which is equal to 20 in decimal). Thus, we get a total delay of:

50 msec * 20 = 1000 msec = 1 second

Program

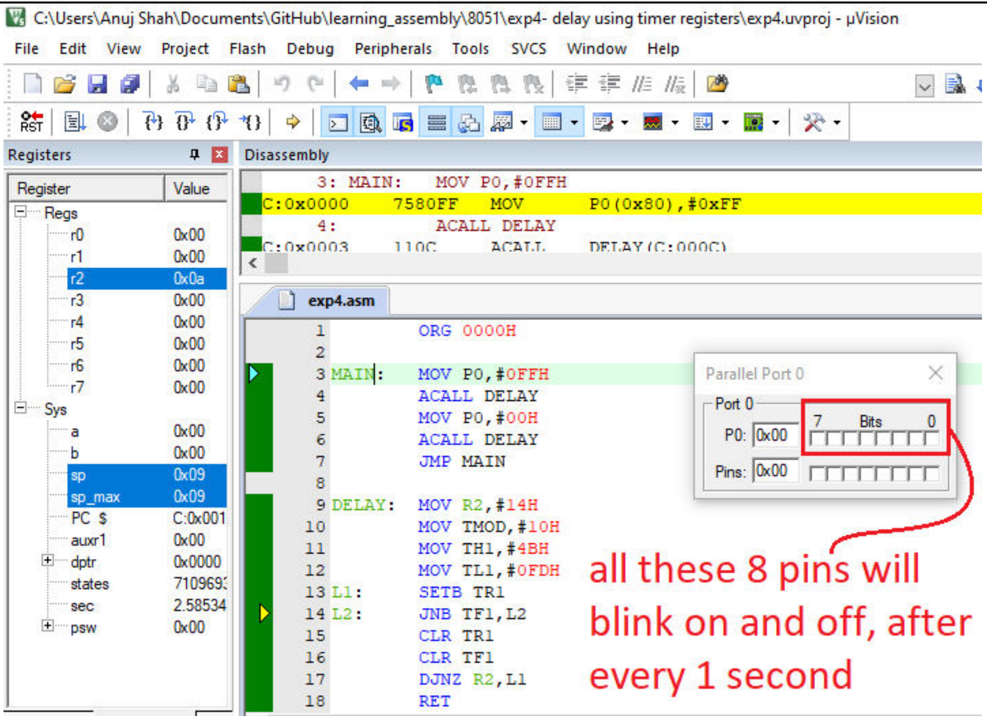


```
        ORG 0000H

MAIN:   MOV P0,#0FFH
        ACALL DELAY
        MOV P0,#00H
        ACALL DELAY
        JMP MAIN

DELAY:  MOV R2,#14H
        MOV TMOD,#10H
        MOV TH1,#4BH
        MOV TL1,#0FDH
L1:     SETB TR1
L2:     JNB TF1,L2
        CLR TR1
        CLR TF1
        DJNZ R2,L1
        RET

        END
```

| | |
|---|---|
| Results / Output |  |
| Conclusion | In this experiment, we learned how to use the Timer registers (either timer0 or timer1) to generate arbitrarily long delays.<br><br>We saw the formula used to calculate the Hex value used to generate a delay; and we also saw how to use looping/repetition to generate longer delays. |

C:\Users\Anuj Shah\Documents\GitHub\learning_assembly\8051\exp4- delay using timer registers\exp4.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
exp4.asm

| Register | Value |
|---|---|
| Regs | |
| r0 | 0x00 |
| r1 | 0x00 |
| r2 | 0x0a |
| r3 | 0x00 |
| r4 | 0x00 |
| r5 | 0x00 |
| r6 | 0x00 |
| r7 | 0x00 |
| Sys | |
| a | 0x00 |
| b | 0x00 |
| sp | 0x09 |
| sp_max | 0x09 |
| PC $ | C:0x001 |
| auxr1 | 0x00 |
| dptr | 0x0000 |
| states | 710969: |
| sec | 2.58534 |
| psw | 0x00 |

```
     3: MAIN:    MOV  P0,#0FFH
C:0x0000     7580FF   MOV      P0(0x80),#0xFF
     4:          ACALL DELAY
C:0x0003     110C     ACALL    DELAY(C:000C)
```

```
          ORG 0000H

MAIN:     MOV P0,#0FFH
          ACALL DELAY
          MOV P0,#00H
          ACALL DELAY
          JMP MAIN

DELAY:    MOV R2,#14H
          MOV TMOD,#10H
          MOV TH1,#4BH
          MOV TL1,#0FDH
L1:       SETB TR1
L2:       JNB TF1,L2
          CLR TR1
          CLR TF1
          DJNZ R2,L1
          RET
```

Parallel Port 0

Port 0
P0: 0x00 | 7 Bits 0
Pins: 0x00

all these 8 pins will blink on and off, after every 1 second

Faculty Sign

Grade received