**Assignment No.: 9**

**AIM:**

Write assembly language program to generate a square wave of 2 KHz on any port pin. Use Timer with interrupt for delay generation.

**OBJECTIVE:** 1. Understanding the Timers and the required SFRs.

2. Use of Timer to generate the time delays.

S/W & H/W USED: 1. Keil uVision2 Compiler / Assembler

2. SST Flash Flex Software for downloading

3. 8051 Trainer kit.

4. CRO

**THEORY:**

### TIME DELAY GENRATION:

Most used subroutine is one that generates a programmable time delay. Time delays may be done by using software loops that essentially do nothing for some period or by using hardware timers that count internal clock pulses. The key to writing this program is to calculate the exact time each instruction will take at the clock frequency in use. Following terms are very helpful to write a subroutine which generates desired time delay.

**T-state**: T-state is defined as one subdivision of the operation performed in one clock period. The terms T-state and clock period are often used synonymously.

**Machine Cycle:** Machine cycle in 8051 is defined as 12 oscillator periods. The 8051, take one to four machine cycles to execute an instruction. To calculate the machine cycle for the 8051, we take 1/12 of crystal frequency, and then take its inverse.

Assume crystal frequency of 11.0592 MHz

M/C frequency = 11.0592 MHz/12 = 921.6 KHz

Machine Cycle = 1/921.6 KHz = 1.085 us (microseconds)

**Instruction Cycle:** Instruction cycle is defined as the time required for completing the execution of an instruction. One instruction cycle consists of one to four machine cycles.

e.g. 2 Machine cycles are required for instruction DJNZ R2, target to be executed. Then instruction cycle is calculated as follows.

Instruction cycle = No. Machine cycles x Machine cycle period

= 2 x 1.085 us

= 2.17 us

**Simple (single loop) Delay Subroutine**

|  |  |  |
| --- | --- | --- |
| ***Label*** | ***Mnemonics*** | ***Machine cycles*** |
| Delay: | MOV R3, #200 | 1 |
| Here: | DJNZ R3, Here | 2 |
|  | RET | 2 |

### DJNZ instruction executed for 200 times.

### Here loop generates time delay = (200x2) x1.085us = 434 us.

* MOV and RET instruction are executed only once and they are called as overhead of the delay subroutine.
* Overhead = (1+2) x1.085us = 3.255 us.
* Total Time delay = Loop delay + Overhead

= 434 us + 3.255 us = 437.255 us

**Calculation for desired Time delay**

* Suppose we want to generate the delay of 100us.
* Simple delay subroutine can generate the 100us delay since delay is less than maximum delay (566 us) generated by subroutine.

|  |  |  |
| --- | --- | --- |
| ***Label*** | ***Mnemonics*** | ***Machine cycles*** |
| Delay: | MOV R3, #YY | 1 |
| Here: | DJNZ R3, Here | 2 |
|  | RET | 2 |

* Find out the value of YY to get desired time delay.
* Time delay (Td) = Loop Delay + Overhead (as per above discussion)
* Loop delay = YY x 1.085 us, overhead = 3 x 1.085 us = 3.255 us
* Td = YY x 1.085 us + 3.255 us
* Therefore YY = (Td – 3.255us)/1.085us= (100us – 3.255us)/1.085us
* YY = 89.16 = (89)10 = 59H
* In similar manner we can generate the larger delay using nested loops.
* For larger delay, first calculate the count value for inner loop, then calculate count value for outer loop considering delay generated by inner loop.

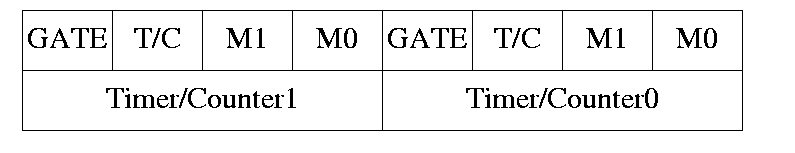
### TIMER/ COUNTER IN 8051:

8051 has 2 timer/ counter. They can be used a timers to generate a time delay or as counter to count events happening outside the microcontroller. Both Timer 0 and Timer 1 are 16 bits wide. Since the 8051 has an 8-bit architecture, each 16-bit timer register is accessed as two separate registers of low byte and high byte.

* Timer 0 can be accessed as –
  + TL0 – Timer 0 lower byte
  + TH0 – Timer 0 higher byte
* Timer 1 can be accessed as –
  + TL1 - Timer 1 lower byte
  + TH1 – Timer 1 higher byte

Both timer shares the Timer control (TCON) register, which controls the timer/ counter operation and Timer mode (TMOD) register, which is used to configure the timer for different operating modes.

***TMOD (Timer Mode Register):***

******

* Both the timers used the same 8 bit register to set various timer operation mode.
* TMOD is 8-bit register where lower 4 byte are set aside for timer 0 and higher 4 bytes for timer 1. Since, it is not bit addressable; the corresponding bit value is directly loaded into TMOD.

*Gate:*

* 8051 has both hardware and software controls to start and stop the timers.
* By the means of software controlling instruction timers are used to control to start timer or stop.

*C/T:*

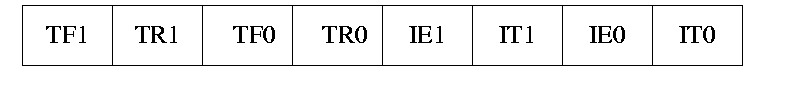
* This bit in TMOD is used to determine whether timer is to be used as delay generator or event counter.
  + If C/T = 0 – used as timer
  + If C/T = 1 – used as counter

*M1 M0:*

M1, M0 selects the timer mode.

|  |  |  |  |
| --- | --- | --- | --- |
| M1 | M0 | Mode | Operation |
| 0 | 0 | 0 | 13 bit counter, 8 bit C/T with THX and TLX as 5 bit Prescalar. |
| 0 | 1 | 1 | 16 bit counter, 8 bit C/T with THX and TLX cascaded with no Prescalar. |
| 1 | 0 | 2 | 8 bit auto reload, THX hold the value which is to be loaded into TLX after each overflow. |
| 1 | 1 | 3 | Split timer mode. |

***TCON (Timer Control Register):***

******

* Timer run control bits TR0 and TR1 and timer overflow flags TF0 and TF1 are the part of 8 bit register called TCON.
* The upper 4 bits are used to store TR and TF flags of both timer- 0 and timer 1 while the lower 4 bits are set aside for interrupt.
* Timer run control bit TR0/TR1 is used to start the corresponding timer / counter.
* Timer overflow flag bit TF0/TF1 is set when corresponding timer/ counter is overflowed i.e. count value FFFF h to 0000 h.

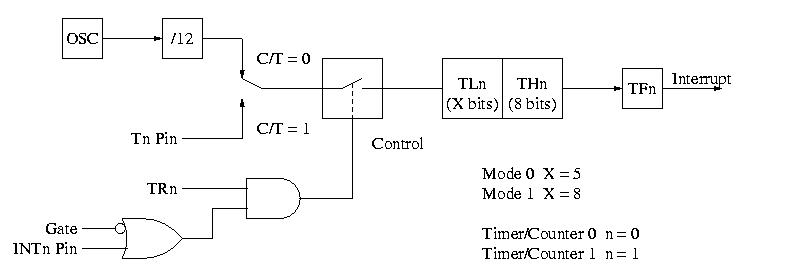
***Different Modes of timer/ counter –***

1. Mode 0 (13 bit timer/counter):

* Mode 0 is exactly like mode 1 except it is 13 bit timer.
* Hence it can hold the values from 0000H to 1FFFH in TL and TH.
* When timer rolls over from 1FFFH to 0000H, the overflow flag i.e. TFX is set.

1. Mode 1 (16 bit Timer / counter):

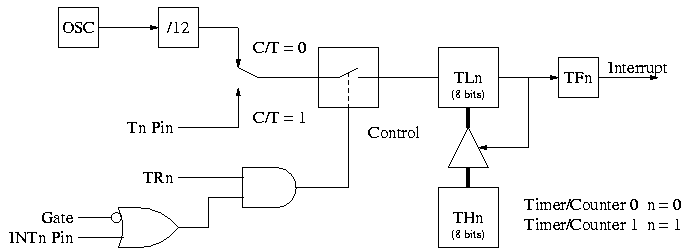
* It is 16 bit timer. Hence allowed values from 0000H to FFFFH to be loaded in TLX and THX.
* After the corresponding 16 bit value is loaded, the timer is started by setting TRX flags.
* After starting, it counts up until it reaches the limit i.e. FFFFH. When it rolls over from FFFFH to 0000H, it sets timer overflow flag i.e. TFX flag.
* After the rolling over process, the operation in mode 1 can be repeated by loading the initial value in TLX & THX and clearing TFX bit.



**Figure 1: Timer/ Counter in Mode 0 and Mode 1**

1. Mode 2 (8 bit auto reload):

* It is 8 bit timer. Hence it allows only values from 00H to FFH to be loaded in THX.
* When THX is loaded into 8 bit value, it sends a copy of it to corresponding TLX. Then timer must be started which is done by SETB TR1 for T1.
* After rolling over of TLX from FFH to 00H, overflow flag for corresponding timer i.e. TFX is set. TLX auto loaded by value present in THX.



**Figure 2: Timer/ Counter in Mode 2**

1. Mode 3 (Split timer mode):

* In mode 3, timer0 worked as split timer i.e. two independent timer.
* TL0 uses the TR0 and TF0 bits of timer 0.
* TH0 uses the TR1 and TF1 bits of timer 1.

### 

**Figure 3.: Timer/ Counter in Mode 3**

### TIME DELAY GENERATION USING TIMER:

Mode 1 and Mode 2 are widely used for most of the applications. Mode 1 is used for time delay generation and mode 2 is used to generate the baud rate in serial communication.

The following steps are taken to generate a time delay using the mode 1 and polling method.

1. Load the TMOD value indicating which timer is to be used and timer mode 1 is selected.
2. Load the registers TL and TH with initial count values. Delay generated is Depends upon the initial count value.
3. Start the timer. (SETB TR0 or SETB TR1)
4. Keep the monitoring the timer flag (TF) with the “JNB TF0, target” or “JNB TF1, target” instruction.
5. Get out of the loop when TF becomes high.
6. Stop the timer. (CLR TR0 or CLR TR1)
7. Clear the TF flag for the next round. (CLR TF0 or CLR TF1)
8. Go back to the step 2 to load TH and TL values.

The size of the time delay depends on two factors, (a) the crystal frequency and (b) the timer’s 16-bit register in mode 1. The largest delay is achieved by the making both TH and TL zero.

Formula for delay calculations using mode 1 of the timer for crystal frequency of XTAL = 11.0592 MHz, (TH, TL) = (NNNNN)10 is as follows.

Time Delay (Td) = (65536 - NNNNN) x 1.085 us.

Therefore Maximum delay = (65536 – 0000) x 1.085 us = 71 ms.

***Finding the Values to be loaded into timer for desired delay***

Assume the 11.0592 MHz as crystal frequency for 8051.

* Divide the desired time delay by 1.085us. ( n = Td/1.085us)
* Perform 65536 – n. where n is the decimal value from step 1.
* Convert the result of step 2 to hexadecimal, where yyxx is the initial hex value to be loaded into the timer’s registers.
* Set TL = xx and TH = yy.

**­­­­­­­­­­­­­­­­­­­­­­­­­­­Calculations:**

Frequency = 2 KHz

Machine Cycle = 1.085 us (microseconds)

Time period = TP = 1/2x103 = 0.5 ms

Required duty cycle is 50%

Therefore, Ton = Toff = 1/TP = 0.25 ms

Desired time delay is TD = 0.25 ms

Divide TD by 1.085x10-6 = n = 0.25x10-3/1.085x10-6 = 230

Subtract n from 65536 = 65536 – 230 = 65306

Convert above decimal value in to Hex value = FF1Ah

Load this value into Timer Register. (TH = FFh , TL = 1Ah)

**ALGORITHM: To generate the 2 KHz square wave on port pin**

1. **Main Program**
2. Load the value “10h” in TMOD register indicating Timer-1 is to be used and timer mode 1 is selected.
3. Load the registers TL and TH with initial count values i.e. FF1Ah.

(TH = FFh, TL = 1Ah)

1. Enable the Timer-1 interrupt by loading the value “88h” in IE register.
2. Start the timer by setting TR1 bit in TCON register.
3. Halt the program.
4. **ISR Routine**
5. Complement the port bit on which square wave is to be monitored.
6. Reload the registers TL and TH with initial count values i.e. FF1Ah.

(TH = FFh, TL = 1Ah)

1. Return from ISR.

**OUTPUT:** Square waveform on C.R.O

**CONCLUSION:**