2/3/20

8051 SERIAL COMMUNICATION

byte by byte transmission

UART - Universal Asynchronous Receiver Transmitter (8051)

USART - Universal Synchronous " " "

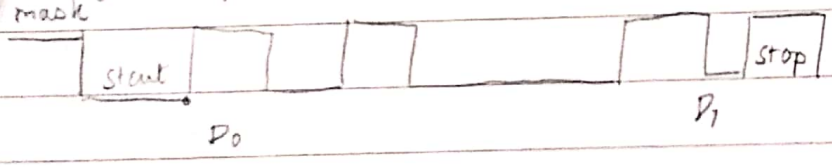
block by block

- 8051 serial communicat" uses asynchronous transfer method.
- Each character is placed bet" start & stop bits. This is called framing.

Consider for example, we have to transfer a byte of data. Draw the pulse pattern.

- start is low, stop is high.

mark - no transfer



Software can be used to write a program to transmit data either synchronously or asynchronously. But these programs can be tedious & long. ∴ Special IC chips are used for the purpose which are called UART or USART

8051 has in-build UART installed

Software - Hardware Requirements for Serial Communication

Pins - RXD, TXD (P3.0, P3.1)

SBUF (Serial Buffer)

SCON (Serial Port Control Special Functional Register)

PCON (Power Mode Control " " ")

* 8051 is ready for transmission anytime but reception depends control reg.

Page No.:
Date:

SBUF Register
- 8 bit register
- Used solely for serial communicat^n of 8051.

For a byte of data to be transfered via the TXD pin, it must be placed onto the SBUF register.
Serial transmission begins anytime data is written to SBUF. The moment it is written onto SBUF, it is framed & then it transfered via the TXD pin.

Receiving data via the RXD.
Recieving data re needs deframing 1st, then placing it on SBUF.

Store a single character into SBUF, move it accumulator
- MOV SBUF, #'A'
  MOV A, SBUF

SBUF has an address 99H
(Both Read & Write use 99H exclusively ie. either it can be read or write)

SCON (Serial Port Control Register)
- Bit Addressable register

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

SM0 & SM1
Serial communicat^n has 4 modes: 0, 1, 2 & 3
Usually mode 1 is used due to various reasons

SM2 - To enable multi-processor communication 0 - No multiprocessor comm 1 - there is
∴ Set to 0

REN (Receiver Enable Bit)

When it is high, it allows 8051 to receive data on RXD.
When it is 0, it is disable i.e. cannot receive anything.

For dual mode of receiving as well as transmitting, turn REN = 1

REN = 1 Dual mode/Receive Only =0 Transmit only

In order to enable/disable REN we can use simple bit-addressable instr's: SETB SCON.4 or CLR SCON.3

TB8 RB8

~~RB8~~ Receive/Transmit Bit 8.

Usually used in mode 2 & 3

TI, RI (Timer Interrupt Flag)

very imp flags for serial communⁿ

When 8051 finishes transfer of 8-bit character, it raises the TI flag to indicate that it is ready to transfer another byte.

TI bit ~~at~~ is raised at the beginning of the stop bit
as transmission is controlled by 8051 itself so it knows where stop is

RI (Receive Interrupt Flag)

When 8051 receives, it's start bits & stop bits are removed & it's data is placed as in SBUF.

Then RI is raised to indicate that a byte has been received & should be removed before it gets lost.

RI is raised half-way through the stop bit

∴ Data is received from outside, so 8051 does not know for sure if it is the end (stop), mistake / false end or part of data.

PCON Register

PCON Register - Not bit addressable

SMOD	-	-	-	GF1	GF0	PD	IDL

SMOD

It is Serial B baud rate modify bit
SMOD = 1 doubles the baud rate eg If first = 2 SMOD makes it 4

6-4 are not implemented.
GF0, GF1 - General Purpose Flags. To be used by user.
PD - Power Down Bit
IDL - Idle mode bit

Baud Rates

For serial communicat", there are various protocols that can be used. Both the communicating devices on ~~a data~~ serial bus are configured to use the exact same protocol. Baud rate specifies how fast data is sent over a serial line & is expressed in ~~unit~~ units of bits per second.
bps - (frequency)

To find the time for single bit transmission, you invert the baud rate.

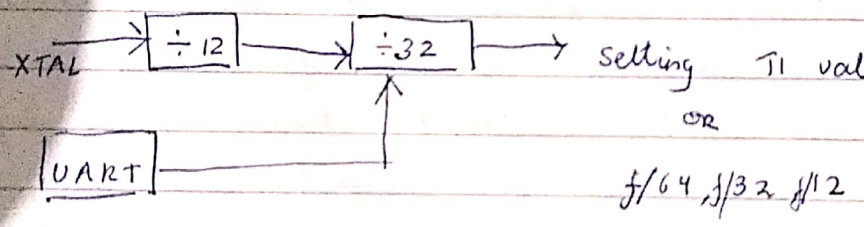
The std baud rates are:-
9600, 200, 2400, 4800, 19200,
38400, 57600, 115200 bps
Most of these baud rates can be obtained if XTAL = 11.0592 MHz as it yields a cycle freq of ~~9+2~~ 921.6 KHz.

3/3/20

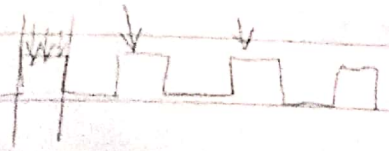
SM0	SM1	Mode	Description
0	0	0	This is called a shift register, 8-bit UART

SM0	SM1	Mode	Description	Source Baud Rate
0	0	0	Shift Register, UART	f/12
0	1	1	8-bit UART	It is variable & determined by Timer 1
1	0	2	9-bit UART	f/32 or f/64
1	1	3	9-bit UART	Variable, determined by Timer 1

The Baud rate is determined by oscillator in Mode 0 & 2.
The Baud rate in Mode 1 & 3 is determined by how frequently Timer 1 overflows.
The more frequently it overflows, the higher is the baud rate.
The most common way is to put Timer 1 in autoreload mode.

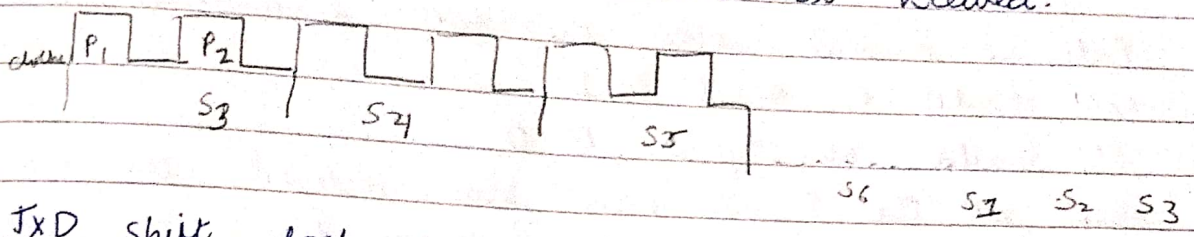


UART needs a clock frequency that is several times faster than the Baud rate that allows UART to support a mismatch bet" the sender & receiver.
UART samples again & again until it detects the start of the start bit
While counting the clocks, UART takes a couple of samples within a window & uses a majority to determine whether the bit was a 0 or 1, thereby improving noise handling.

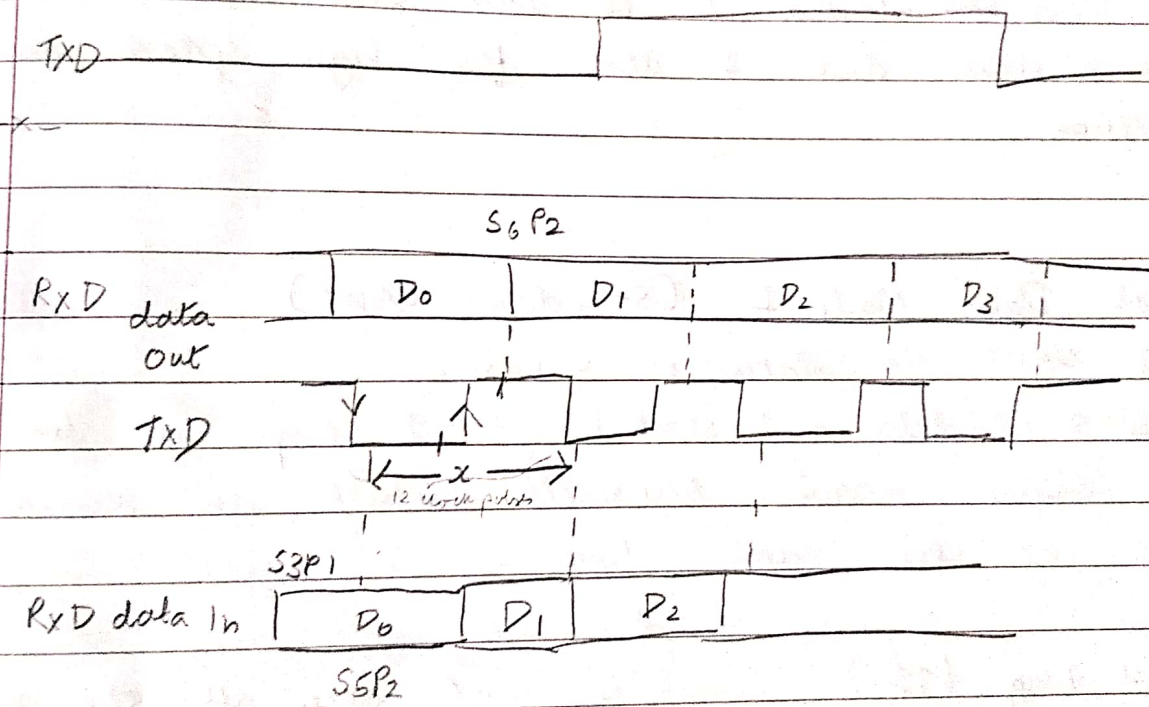


Serial Data Mode # 0 (Shift Register Mode)

In mode 0, SBUF receives or transmits 8 bits of data using only RXD pin for both the functions. Pin TXD provides a pulse source to determine when the data is to be transmitted or received.



TxD shift clock is a square wave that is low for m/c cycle states S_3, S_4, S_5 & high $S_6, S_1 \& S_2$



When transmitting, data is shifted out of RXD. The data changes on the falling edge of S6P2 on 1 clock pulse after the rising edge of the o/p TxD shift clock. Received data comes in on pin RXD & is sampled on the falling edge of S5P2 & shifted into SBUF on the rising edge of the shift clock.

The Transmission And Reception Process

For data to be transmitted, once data is put on SBUF, it will begin transmission & TI_{bit} of SCON reg will is automatically set when all 8 bits have been set.

Rata is received when the fool 2 conditions are satisfied

i) REN should be equal to 1

ii) RI should be equal to 0.

When all the 8 bits have been received, the RI bit of SCON will be automatically set to 1.

Applications of this mode:-

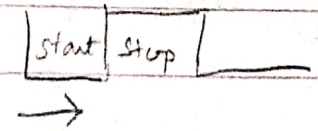
Mode 0 is ~~not~~ intended to be used between devices that have a short dist. & also for high speed serial connection

Serial Data Mode 1 (Standard UART)

- 8-bit UART i.e. data is 8 bits.
- This is 8 bit data + 1 start bit + 1 stop bit becomes a full duplex receiver - transmitter which can receive & transmit data at the same time.

Interrupt Flag (TI) is set once all the bits have been set i.e. all 10 bits have been set.

Data is received in the same order which is triggered by the falling age of start bit & continues if the stop bit is true i.e. level 0 halfway through the start bit interval.



The data bits are shifted into the receiver at the program baud rate & data will be loaded into SBUF if the following is true. RI=0, REN=1 & stop bit should be high.

SM2=0 also from SCON is alternate to stop bit.

13/20

Mode 1 Baud Rates

$$f_{baud} = \frac{\square}{32/64/12}$$

SMOD = 0 (bit) - X
= 1 - doubles it

$$\frac{2^{SMOD=0}}{32} = \frac{1}{32}$$

$$\frac{2^{SMOD=1}}{32} = \frac{2}{32}$$

Typically, Timer 1 is used to generate the baud rate for mode 1. by using the Overflow flag of the timer to determine the baud frequency. The calculation will be as follows.

$$f_{baud} = \frac{2^{SMOD}}{32} \times \text{timer1 overflow freq.}$$

If timer 1 is used in Mode 0, then the calculation will be $\frac{2^{SMOD}}{32} \times \frac{\text{oscillator freq}}{12 \times [256 - TH1]}$

Sa

Page No.:
Date:

Serial Data Mode 2 (Multiprocessor Mode)

- 9 bit data + 1 stop bit + 1 start bit
  8 bit data + 1 parity bit
- The 9th bit is copied from bit TB8 in SCON during transmit & is stored in RB8 of SCON when data is received.
- The baud rate calculation will be

$$\frac{2^{SMOD}}{64} \times \text{oscillator freq.}$$

Serial Data Mode 3

- 9bit UART
- working is similar to Mode 2.
  @ Timer not osc. freq as in Mode 1. for baud rate

Programming in 8051 / Serial Communication.

1. Assuming that we are programming the timers for mode 2. Find the values in Hex loaded into TH for the following cases.

i) MOV TH1, #-3

FFH - 3 ∴ FDH

ii) MOV TH0, #-12

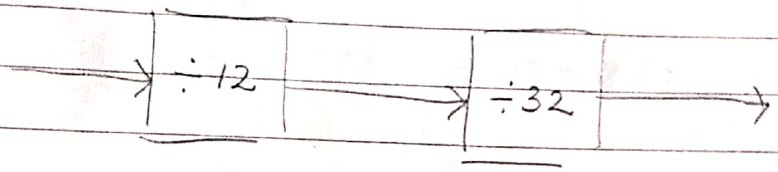
F4H

2. With XTAL = @ 11.0592 MHz, find the value of TH1 needed to have the following Baud rates.

a) 9600

b) 2400

c) 1200



The machine cycle freq. of 8051 = $\frac{\text{XTAL freq}}{12}$

= 921.6 kHz

$\frac{921.6 \text{ kHz}}{32}$ = 28.8 kHz = 28800 Hz

a) $\frac{28800}{3}$ = 9600

∴ 3

∴ #-3H

The freq XTAL = 11.0592 MHz gives us most of the std baud rates needed for communication.

∴ The value to be loaded in TH1 is -3 which is FD in Hexadecimal.

b) -12

c) -24

3. WAP for 8051 to transfer the letter A, serially at a baud rate of 9800 continuously. uses Timer 1 Mode 2.

TMOD:

TMOD = #20H

0010 0000

SCON

; Timer → Mode 1 or 3

Transmitting — TREN

SCON:

We will use Mode 1

SCON #40H

TH1:

28800 − 4800

TH1 =#-6

Program:

```
MOV   TMOD, #20H
MOV   TH1 , #-6
MOV   SCON, #40H

SET   TR1
;
AGAIN:
MOV   SBUF, #"A"

HERE: JNB   TI, HERE

      CLR   TI
      JMP   AGAIN:
```

A → SBUF → transmit

TI=1 when bit transmitted

To send next bit

Write a program to continuously transfer the letter B at the rate 2400 bits per second.

Circuit divide by 12 then by 32 } NOTE
Value in TH1
TMOD
SCON

FF - 12 = F4

```
MOV  TMOD , # 20H
MOV  TH1  , # -12
MOV  SCON , # 40 H
Set b TR1
AGAIN : MOV  SBUF , " # B"
HERE  : JNB , TI , HERE
        CLR  TI
        JMP  AGAIN
```

(#)

XTAL = 11.0592
Always
Timer 1 mod 2

(Any letter)

TI - Timer Input

TH1 Decimal	SMOD = 0	SMOD = 1
-3	9600	19200
-6	4800	9600
-12	2400	4800
-24	1200	2400

Write a program to transfer the message "YES" serially at Baud rate 8 bit data 1 STOP bit continuously. Using SMOD = 1 (Timer 1 mod 2).

Since we are using SMOD = 1 ∴ TH1 = -6

The original baud rate is 4800 and we are doubling it 9600.

```
mov TMOD, # 20H
mov TH1 , # -6   → assumed already set to SMOD = 1
mov SCON, # 50H
```

You can transmit YES together or separately Y, E, S using loop. This example is apparately ⇒ YES

Scanned with CamScanner

```
        set b TRI
AGAIN:  mov a, # "Y"
        call TRANS
TRANS:  mov SBUF, a
HERE:   JNB TI, HERE   → wait for timer flag
        clr TI
        RET            → Return to Main
                         call timer (before).

        mov a, # "E"        SMOD = 0
        call TRANS             = 1
        mov a, # "S"        is in PCON.
        call TRANS
        JMP AGAIN          → To keep on have YES, YES ...
```

✓ Receiving Data

Write a program to receive bytes of Data serially and put them in P1. Set the baud rate 4800 8 bit data and 1 stop bit (Normal)

```
        mov TMOD, # 20H        REN - 1
        mov TH1, # -6          ↳ Receive Enable
        mov SCON, # 50 H       Checking for Reception
        Set b, TR1             Continuous monitor
HERE:   JMP RI, HERE           ↳ 1st goes in SBUF
        mov a, SBUF → Receiving    and then anywhere
        mov P1, a      data
        clr: RI                clear RI
        SJMP HERE          → Continuously Monitoring
```

Doubling the Baud Rate

We can increase the baud rate by using higher frequency crystal or by SMOD = 1

The PCON register holds the 8th bit as SMOD which resists whether the baud rate could be doubled or not.

SMOD = 0 → ÷ 32

SMOD = 1 → ÷ 16

and PCON is not bit addressable (limitation).
Set PCON register to double the band rate while leaving all the other values to 0.
[new PCON is bit addressable register so that you can use Set b, clr like in a register A]

We could use this procedure to address the limitation of PCON not being addressable.

```
mov A, PCON
Set b ACC.7
mov PCON, A
```

Assuming XTAL = 11.0592 MHz → ÷ 12 → 921.6 KHz → ÷ 16 (SMOD=1, SMOD=2)

Write a program assuming XTAL = 11.0592 MHz to transfer the letter B receive continuously at a Band rate 192.00 Bits per sec

```
mov a, PCON
Set b ACC.7
mov PCON, a
```

Since we are using SMOD = 1

```
mov TMOD, # 20 H
mov TH1, # -3
mov SCON, # 40 H
Set b TR1
HERE: mov a, SBUF
LOOP: JMP, RI, LOOP
      Clr RI
      JMP HERE
```

SCON = 0, 40
1, 50

Find the Band rate TH1 = -2, SMOD = 1 and XTAL 11.0592 MHz

28800 / 2 = 14400

SMOD1 = Double 14400 = 28800

This Band rate is not supported by IBM a

9/2/20

8051 INTERRUPT PROGRAMMING

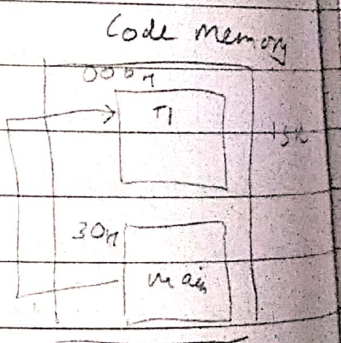
There are 2 ways of determining the conditions that exist in internal or external circuits

i Software Method / Polling.
Wherein jumps are made to subroutines on the states of flags & port pin.

ii Hardware Signals / Interrupts.
Wherein no flags need to be checked & the program is forced to call a subroutine
Most real time systems use h/w interrupts.

Steps in executing an interrupt :-

1) Upon activation of an interrupt.
(1) Saves the address of next instruction i.e. Program Counter on the stack.
(2) Saves the current status of all the interrupts internally
(3) Jumps to the fixed memory called interrupt vector table that holds addresses of ISR (Interrupt Service Routines)
(4) Executes the ISR.
(5) Returns to where it was interrupted.



Requirements for Timer Programmer
TCON - lower 4 bits of TCON
IE - (Interrupt Enable)
IP - & Interrupt Priority Register
6 interrupts :-
RST, T0, T1, $\overline{INT0}$, $\overline{INT1}$, TI & RI - user same addr space
(Input)

EA bit of Interrupt Enable (EA) Register decides whether all the interrupts are enabled or disabled.

Interrupt Enable (IE) Register

- 8 bit register.

EA	-	ET_2	ES	ET_1	EX_1	ET_0	EX_0
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

T_1 INT0 T_0 INT0

ET_2 – Enables/Disables Timer 2 overflow.
ES – Enables/Disables ~~use of~~ serial port interrupt
ET1 – External Timer 1 overflow
EX1 – Enables/Disables external interrupt 1

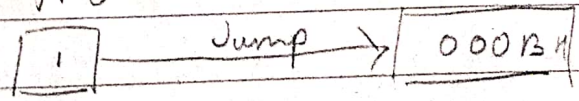
- This register is bit-addressable.

Interrupt Vector Table

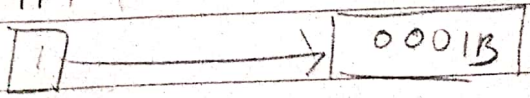
Interrupt	Location	Pin	Size in Bytes
RST	0000	9	3 Bytes
INT0	0003	P3.2	
TF0	000B		8 Bytes
INT1	0013		
TF1	001B		
RITI	0023		

Programming Timer Interrupts.

TF0



TF1



1. WAP that continuously gets 8-bit data from P0 & sends it to P1 while simultaneously creating a square wave of 200 μs time period on pin P2.1. Use timer 0 to create the square wave. XTAL = 11.0592 MHz.

- We can use Timer 0 Mode 2 ∵ 256 is sufficient.

∵ Whole wave = 200 μs

∴ for a portion of the wave, we require 100 μs

$$\frac{100}{1.085} = 92$$

TMOD = 02H

TH0 = -92

There are 2 operations taking place & we need an ISR to generate the square wave.

```
        ORG   000H
        JMP   MAIN      - To indicate main program is somewhere
        ORG 0003H
        CPL  P2.1
        RETI            ; Return from interrupt
```

Small code in ISR

```
        ORG  30H
MAIN:   MOV  TMOD, #02H
        MOV  P0, #0FFH
        MOV  TH0, #-92      ; Draw high & low portion of wave
        MOV  IE, #82H
```

Let us set the values of IE register

1 0 0 0 0 0 1 0 = 82H

(EA ↑, ET0)

```
SETB    TR0
BACK:   MOV A, P0       ; transmitting to data
        MOV P1, A
        JMP BACK
```

We do not req. to clear TR0 (ie use CLR TR0) ∵ it is automatically cleared after leaving the Interrupt Service routine.
We also do not req. to write JNB TF0, Target. as we have enabled the respective interrupts.
The ISR here, is short enough to fit in the memory space allocated. Otherwise it would have to be written in a larger space of memory

12/3/20

2. WAP to generate a square wave of 50 Hz freq using interrupts.

$$\frac{1}{50} = 20\ ms$$

whole wave = 20 ms
½

For ½ wave = 10 ms

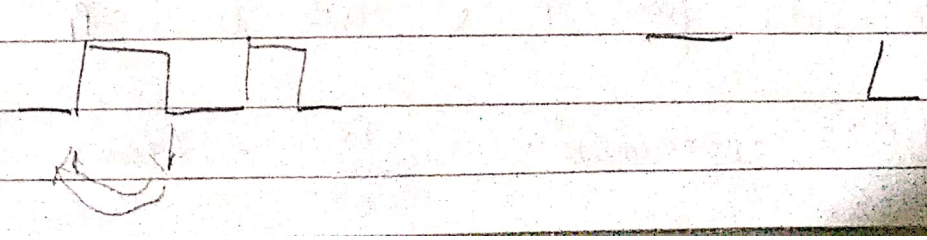
$$\frac{10}{1.085} = 9216 \text{ counts.}$$

FFFF

Timer0 Mode 1 will be used.

000BH

IE = 82H ∵ Timer 0 & D7 bit of IE.



```
        ORG   0H
        JMP   MAIN
        ORG   000BH
        CPL   P1.2
        MOV   TL0,#00H
        MOV   TH0,#0DCH
        RETI

        ORG   30H
MAIN:   MOV   TMOD,#01H
        MOV   TL0,#00H
        MOV   TH0,#0DCH
        MOV   IE,#82H

        SETB  TR0

HERE:   SJMP  HERE
```

```
        CPL   P1.2
        MOV   TL0,00H
        MOV   TH0,#0DCH
        RETI
```

FFFF
0000
TF=0
ISR:000H

In the ISR, we do not start the timer again because we did not stop it. The process will continue until you clear the Timer Run flag.

3. WAP to create a square wave that has a high portion of 1.085 ms & a low portion of 15ms & also continuously get 8 bit data from P0 & sends it to P1.

$\frac{1.085}{1.085}$ = 1000 counts

$\frac{15}{1.085}$ = 14 counts

15 ms
14 m/c
14 counts

Date:

∴ The max counts we req here is 1000 ∴ we will use mode 1 programming.

This ISR code will req a larger code space & hence we will have to service this interrupt by jumping to a larger code space.

TL1 = 018H TH1 = 0FCH IE = 88H ∴ Timer 1 interrupt

```
        ORG 0000H
        LJMP MAIN
        ORG 001BH
        LJMP ISR_T1

        ORG 30
MAIN:   MOV TMOD, #10H
        MOV P0, #0FFH
        MOV TL01, #018H
        MOV TH1, #0FCH
        MOV IE, #88H

        SETB TR1
BACK:   MOV A, P0
        MOV P1, A
        SJMP BACK

ISR_T1: CLR TR1                    1000ms
        CPL P1.2
        MOV TL1, #18H      ; To move TL1 we use 2 m/c
        MOV TH1, #0FCH     ; To move TH1 we use 2 m/c

        MOV R2, #4
HERE:   DJNZ R2, HERE
```

```
SETB   TR1     ; uses 1 m/c
SETB   P2.1    ; 1 m/c

MOV    R2, #4  ; uses 2 m/c

RETI
```

Notice that the lower portion of the pulse is created using 14 m/c where each m/c = 1.085 μs & 14 x 1.085 μs = 15.19 μs.

Programming External Hardware Interrupts.

INT0 - P3.2

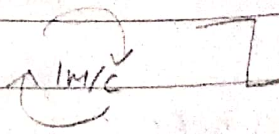
INT1 - P3.3

INT0 is serviced at 0003H & INT1 is serviced at 0013H.

Level Triggered Interrupts.

1. ~~WAP~~ Assume that INT1 pin is connected to a switch that is normally high, whenever it goes low, it should turn on an LED. The LED is connected to P1.3 & is normally off.

When it is turned on, it should turn on for a fraction of a sec, as long as the switch is pressed low, the LED should stay on.



We assume that 1 fraction of a sec = 1 m/c.
∴ We can load 255 or any other value or use the looping method to generate a time delay.