UNIT 3

3.1 I/O Bit Manipulation Programming

There are four ports P0, P1, P2 and P3 each use 8 pins, making them 8-bit ports. All the ports upon RESET are configured as output, ready to be used as output ports. To use any of these ports as an input port, it must be programmed.

Port 0: Port 0 occupies a total of 8 pins (pins 32-39) .It can be used for input or output. To use the pins of port 0 as both input and output ports, each pin must be connected externally to a 10K ohm pull-up resistor. This is due to the fact that P0 is an open drain, unlike P1, P2, and P3.Open drain is a term used for MOS chips in the same way that open collector is used for TTL chips. With external pull-up resistors connected upon reset, port 0 is configured as an output port. For example, the following code will continuously send out to port 0 the alternating values 55H and AAH

```
MOV A, #55H
BACK:  MOV P0, A
ACALL DELAY
CPL A
SJMP BACK
```

Port 0 as Input: With resistors connected to port 0, in order to make it an input, the port must be programmed by writing 1 to all the bits. In the following code, port 0 is configured first as an input port by writing 1's to it, and then data is received from the port and sent to P1.

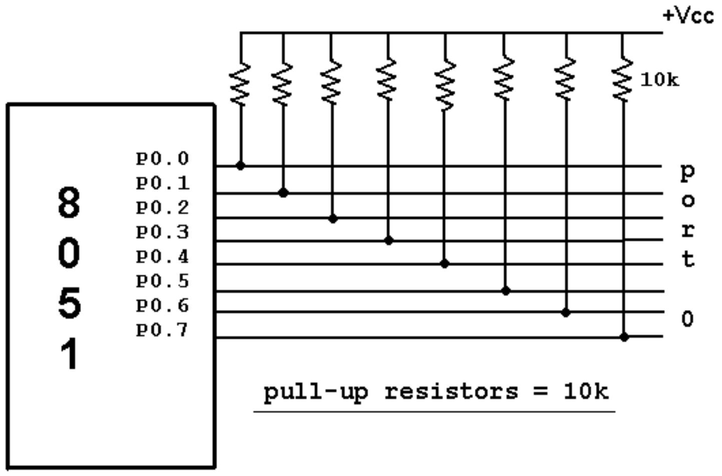


Fig: 8051 I/O Ports

```
MOV A, #0FFH        ; A = FF hex
MOV P0, A           ; make P0 an input port
BACK: MOV A, P0     ; get data from P0
MOV P1, A           ; send it to port 1
SJMP BACK
```

Dual role of port 0: Port 0 is also designated as AD0-AD7, allowing it to be used for both address and data. When connecting an 8051/31 to an external memory, port 0 provides both address and data. The 8051 multiplexes address and data through port 0 to save pins. ALE indicates if P0 has address or data. When ALE = 0, it provides data D0-D7, but when ALE =1 it has address and data with the help of a 74LS373 latch.

Port 1, port 2 and port 3 do not require any pull up registers since they have internal pull up registers. On reset, all ports are configured as input ports.

If the ports are configured as output ports, to make them input ports again, we have to write FFH on these ports.

3.2 8051 Timers

The 8051 has two timers: timer0 and timer1. They can be used either as timers or as counters. Both timers are 16 bits wide. Since the 8051 has an 8-bit architecture, each 16-bit is accessed as two separate registers of low byte and high byte. First we shall discuss about Timer0 registers.

Timer0 registers is a 16 bits register and accessed as low byte and high byte. The low byte is referred as a TL0 and the high byte is referred as TH0. These registers can be accessed like any other registers.

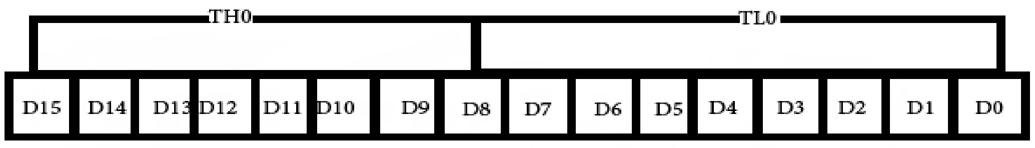


Fig. Timer0

Timer1 registers is also a 16 bits register and is split into two bytes, referred to as TL1 and TH1.

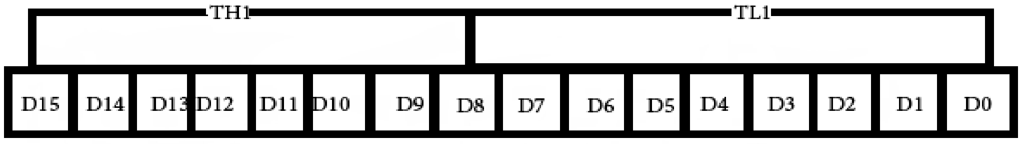


Fig. Timer1

TMOD (timer mode) Register: This is an 8-bit register which is used by both timers 0 and 1 to set the various timer modes. In this TMOD register, lower 4 bits are set aside for timer0 and the upper 4 bits are set aside for timer1. In each case, the lower 2 bits are used to set the timer mode and upper 2 bits to specify the operation.

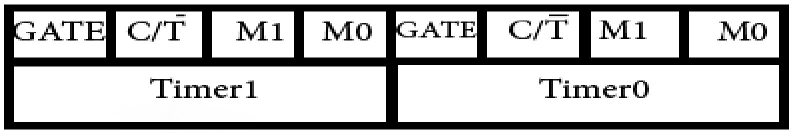


Fig. TMOD Register

TMOD

In upper or lower 4 bits, first bit is a GATE bit. Every timer has a means of starting and stopping. Some timers do this by software, some by hardware, and some have both software and hardware controls. The hardware way of starting and stopping the timer by an external source is achieved by making GATE=1 in the TMOD register. And if we change to GATE=0 then we do no need external hardware to start and stop the timers.

The second bit is C/T bit and is used to decide whether a timer is used as a time delay generator or an event counter. If this bit is 0 then it is used as a timer and if it is 1 then it is used as a counter.

In upper or lower 4 bits, the last bits third and fourth are known as M1 and M0 respectively. These are used to select the timer mode.

M0	M1	Mode	Operating Mode
0	**0**	**0**	13-bit timer mode, 8-bit timer/counter THx and TLx as 5-bit prescalar
0	**1**	**1**	16-bit timer mode, 16-bit timer/counters THx and TLx are cascaded; There are no prescalar.
1	**0**	**2**	8-bit auto reload mode, 8-bit auto reload timer/counter; THx holds a value which is to be reloaded into TLx each time it overflows
1	**2**	**3**	Spilt timer mode.

M1, MO

MO and Ml select the timer mode. As shown in above TMOD Register Figure, there are three modes: 0, 1, and 2. Mode 0 is a 13-bit timer, mode 1 is a 16-bit timer, and mode 2 is an 8-bit timer. We will concentrate on modes 1 and 2 since they are the ones used most widely. We will soon describe the characteristics of these modes, after describing the rest of the TMOD register.

C/T (clock/timer)

This bit in the TMOD register is used to decide whether the timer is used as a delay generator or an event counter. If C/T = 0, it is used as a timer for time delay generation. The clock source for the time delay is the crystal frequency of the 8051

GATE

The other bit of the TMOD register is the GATE bit. Notice in the TMOD registers Figure that both Timers 0 and 1 have the GATE bit. What is its purpose? Every timer has a means of starting and stopping. Some timers do this by software, some by hardware, and some have both software and hardware controls. The timers in the 8051 have both. The start and stop of the timer are controlled by way of software by the TR (timer start) bits TRO and TR1. This is achieved by the instructions "SETB TR1″ and "CLR TR1″ for Timer 1, and "SETB TRO" and "CLR TRO" for Timer 0. The SETB instruction starts it, and it is stopped by the CLR instruction. These instructions start and stop the timers as long as GATE = 0 in the TMOD register. The hardware way of starting and stopping the timer by an external source is achieved by making GATE = 1 in the TMOD register. However, to avoid further confusion for now, we will make GATE = 0, meaning that no external hardware is needed to start and stop the timers. In using software to start and stop the timer where GATE = 0. all we need are the instructions "SETB TRx" and "CLR TRx".

Example 1

Indicate which mode and which timer are selected for each of the following.

(a) MOV TMOD,#01H (b) MOV TMOD,#20H (c) MOV TMOD,#12H

Solution:

We convert the values from hex to binary. From Figure 9-3 we have:

1. TMOD = 00000001, mode 1 of Timer 0 *is* selected.
2. TMOD = 00100000, mode 2 of Timer 1 is selected.

3. TMOD = 00010010, mode 2 of Timer 0, and mode 1 of
   Timer 1 are selected.

Clock source for timer

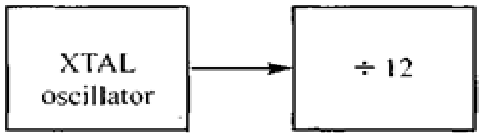
As you know, every timer needs a clock pulse to tick. What is the source of the clock pulse for the 8051 timers? If **$C/\overline{T}$** = 0, the crystal frequency attached to the 8051 is the source of the clock for the timer. This means that the size of the crystal frequency attached to the 8051 also decides the speed at which the 8051 timer ticks. The frequency for the timer is always **1/12th** the frequency of the crystal attached to the 8051.

Example 2

Find the timer's clock frequency and its period for various 8051-based systems. with the following crystal frequencies.

(a) 12 MHz
(b) 16 MHz
(c) 11.0592 MHz

Solution:



```
(a) 1/12 × 12 MHz = 1 MHz and  T = 1/1 MHz = 1 µs

(b) 1/12 × 16 MHz = 1.333 MHz and T = 1/1.333 MHz = .75 µs

(c) 1/12 × 11.0592 MHz = 921.6 kHz;
    T = 1/921.6 kHz = 1.085 µs
```

TCON register- Bits and symbol and functions of every bits of TCON are as follows:

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Fig. TCON Register

Bit Position	Symbol	Functions
TCON.7	TF1	Timer1 over flow flag. Set when timer rolls from all 1s to 0; Cleared When the processor vectors to execute interrupt service routine; Located at program address 001Bh.
TCON.6	TR1	Timer 1 run control bit. Set to 1 by programmer to enable timer to count; Cleared to 0 by program to halt timer
TCON.5	TF0	Timer 0 over flow flag. Same as TF1.
TCON.4	TR0	Timer 0 run control bit. Same as TR1.
TCON.3	IE1	External interrupt 1 Edge flag. Not related to timer operations.
TCON.2	IT1	External interrupt1 signal type control bit. Set to 1 by program to Enable external interrupt 1 to be triggered by a falling edge signal. Set To 0 by program to enable a low level signal on external interrupt1 to generate an interrupt.
TCON.1	IE0	External interrupt 0 Edge flag. Not related to timer operations.
TCON.0	IT0	External interrupt 0 signal type control bit. Same as IT0.

8051 Timer Modes and Programming

Mode 0- Mode 0 is exactly same like mode 1 except that it is a 13-bit timer instead of 16-bit. The 13-bit counter can hold values between 0000 to 1FFFH in TH-TL. Therefore, when the timer reaches its maximum of 1FFH, it rolls over to 0000, and TF is raised.

Mode 1- It is a 16-bit timer; therefore it allows values from 0000 to FFFFH to be loaded into the timer's registers TL and TH. After TH and TL are loaded with a 16-bit initial value, the timer must be started. We can do it by "SETB TR0" for timer 0 and "SETB TR1" for timer 1. After the timer is started. It starts count up until it reaches its limit of FFFFH. When it rolls over from FFFF to 0000H, it sets high a flag bit called TF (timer flag). This timer flag can be monitored. When this timer flag is raised, one option would be stop the timer with the instructions "CLR TR0" or CLR TR1 for timer 0 and timer 1 respectively. Again, it must be noted that each timer flag TF0 for timer 0 and TF1 for timer1. After the timer reaches its limit and rolls over, in order to repeat the process the registers TH and TL must be reloaded with the original value and TF must be reset to 0.

Mode 2- It is an 8 bit timer that allows only values of 00 to FFH to be loaded into the timer's register TH. After TH is loaded with 8 bit value, the 8051 gives a copy of it to TL.

Then the timer must be started. It is done by the instruction "SETB TR0" for timer 0 and "SETB TR1" for timer1. This is like mode 1. After timer is started, it starts to count up by incrementing the TL register. It counts up until it reaches its limit of FFH. When it rolls over from FFH to 00. It sets high the TF (timer flag). If we are using timer 0, TF0 goes high; if using TF1 then TF1 is raised. When Tl register rolls from FFH to 00 and TF is set to 1, TL is reloaded automatically with the original value kept by the TH register. To repeat the process, we must simply clear TF and let it go without any need by the programmer to reload the original value. This makes mode 2 auto reload, in contrast in mode 1 in which programmer has to reload TH and TL.

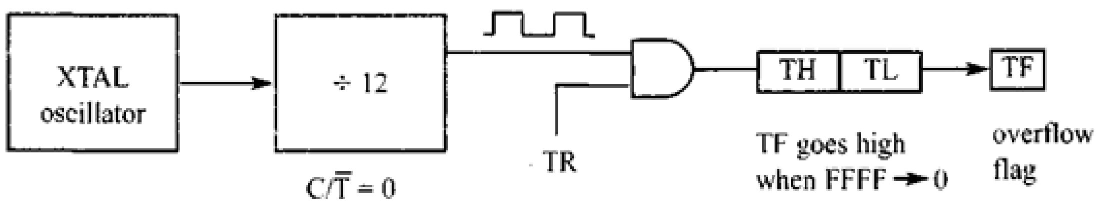
Mode 3- Mode 3 is also known as a split timer mode. Timer 0 and 1 may be programmed to be in mode 0, 1 and 2 independently of similar mode for other timer. This is not true for mode 3; timers do not operate independently if mode 3 is chosen for timer 0. Placing timer 1 in mode 3 causes it to stop counting; the control bit TR1 and the timer 1 flag TF1 are then used by timer0.

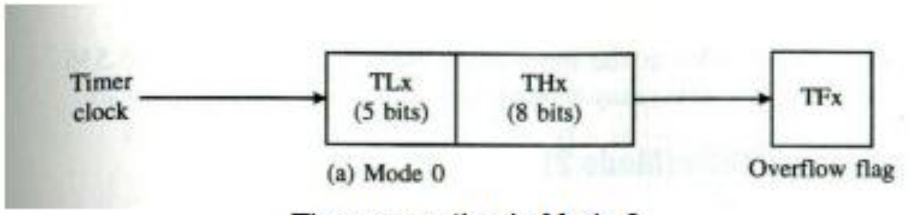
Timer Modes –

- ✓ Mode 0: 13 bit timer
- ✓ Mode 1: 16-bit timer
- ✓ Mode 2: 8-Bit auto reload
- ✓ Mode 3: Split timer mode

Mode 0: 13-Bit Timer

- Lower byte (TL0/TL1) + 5 bits of upper bytes (TH0/TH1).
- Backward compatible to the 8048
- Not generally used

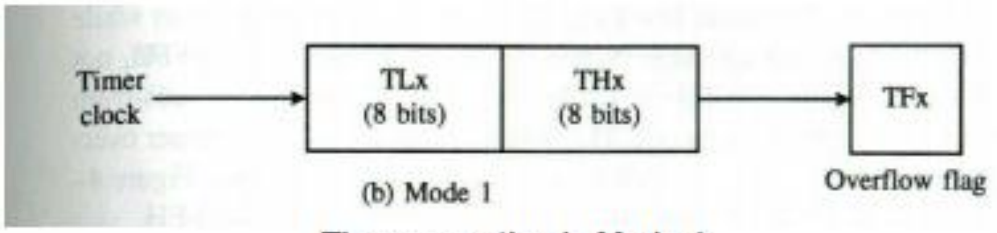




Timer operation in Mode 0

Mode 1: 16-bit

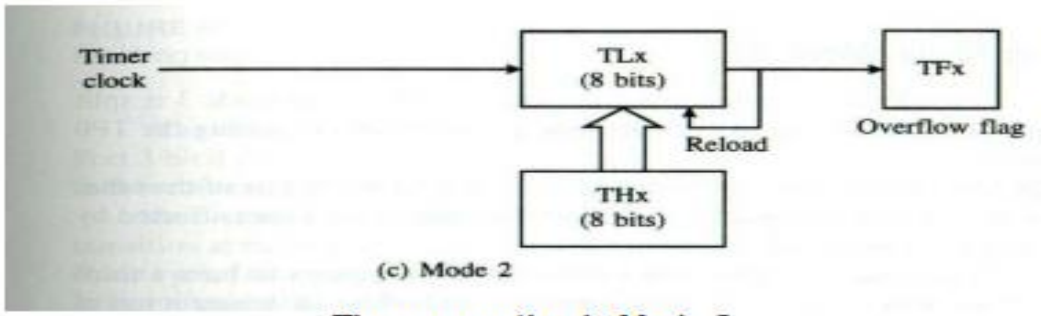
- All 16 bits of the timer (TH0/TL0, TH1,TL1) are used.
- Maximum count is 65,536
- At 12Mhz, maximum interval is 65536 microseconds or 65.536 milliseconds
- TF0 must be reset after each overflow
- THx/TLx must be manually reloaded after each overflow.



Timer operation in Mode 1

Mode 2: 8-bit Auto Reload

- Only the lower byte (TLx) is used for counting.
- Upper byte (THx) holds the value to reload into TLx after an overflow.
- TFx must be manually cleared.
- Maximum count is 256
- Maximum interval is 256 Microseconds or .256 milliseconds

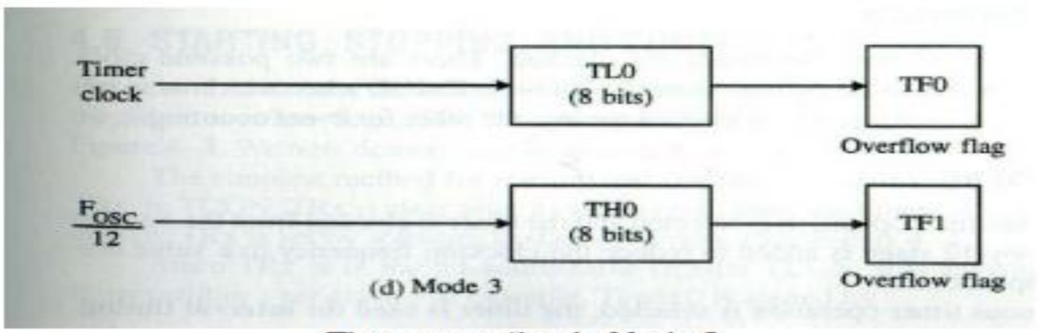


Timer operation in Mode 2

Mode 3- Split Timer

- Splits Timer 0 into two 8-bit timers

- TL0 sets TF0
- TH0 sets TF1
- Timer 1 is available for other 3 modes, but the TF1 is not available.



Timer operation in Mode 3

Mode 1 programming

The following are the characteristics and operations of mode 1:

1. It is a 16-bit timer; therefore, it allows values of 0000 to FFFFH to be loaded into the timer's registers TL and TH.
2. After TH and TL are loaded with a 16-bit initial value, the timer must be started. This is done by "SETB TRO" for Timer 0 and "SETB TR1″ for Timer 1.

Steps to program Timer 0 in mode 1

To generate a time delay, using the timer's mode 1, the following steps are taken.

1. Load the TMOD value register indicating which timer (Timer 0 or Timer 1) is to be used and which timer mode (0 or 1) is selected.
2. Load registers TL0 and TH0 with initial count values.
3. Start the timer by setting TR0 bit=1.
4. Keep monitoring the timer flag (TF) with the "JNB TFx, target" instruction to see if it is raised. Get out of the loop when TF becomes high.
5. Stop the timer by clearing TR0 bit=0 with CLR TR0 instruction.
6. Clear the TF0 flag with CLR TF0 instruction for the next round.

Example 3

In the following program we are creating a square wave of 50% duty cycle (with equal portions high and low) on the P1.5 bit. Timer 0 is used to generate the time delay. Analyze the program.

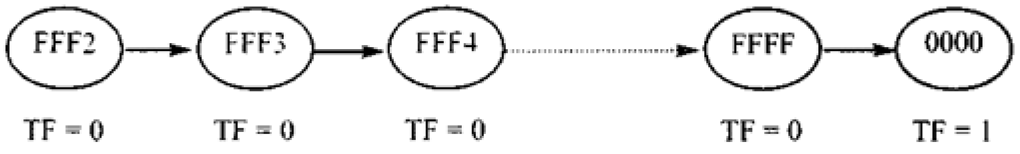
```
          MOV    TMOD,#01        ;Timer 0, mode 1(16-bit mode)
HERE:     MOV    TL0,#0F2H       ;TL0 = F2H, the Low byte
          MOV    TH0,#0FFH       ;TH0 = FFH, the High byte
          CPL    P1.5            ;toggle P1.5
          ACALL  DELAY
          SJMP   HERE            ;load TH, TL again
;————————delay using Timer 0
DELAY:
          SETB   TR0             ;start Timer 0
AGAIN:    JNB    TF0,AGAIN       ;monitor Timer 0 flag until
                                 ;it rolls over
          CLR    TR0             ;stop Timer 0
          CLR    TF0             ;clear Timer 0 flag
          RET
```

Solution:

In the above program notice the following steps.

1. TMOD is loaded.
2. FFF2H is loaded into THO – TLO.
3. P1.5 is toggled for the high and low portions of the pulse.
4. The DELAY subroutine using the timer is called.
5. In the DELAY subroutine, Timer 0 is started by the "SETB TRO" instruction.
6. Timer 0 counts up with the passing of each clock, which is provided by the crystal oscillator. As the timer counts up, it goes through the states of FFF3, FFF4, FFF5, FFF6, FFF7, FFF8, FFF9, FFFA, FFFB, and so on until it reaches FFFFH. One more clock rolls it to 0, raising the timer flag (TFO = 1). At that point, the JNB instruction falls through.
7. Timer 0 is stopped by the instruction "CLR TRO". The DELAY subroutine ends, and the process is repeated.

Notice that, to repeat the process, we must reload the TL and TH registers and start the timer again.



Example 4

In Example 3, calculate the amount of time delay in the DELAY subroutine generated by the timer. Assume that XTAL = 11.0592 MHz

Solution:

The timer works with a clock frequency of 1/12 of the XTAL frequency; therefore, we have 11.0592 MHz / 12 = 921.6 kHz as the timer frequency. As a result, each clock has a period of T = 1 / 921.6 kHz = 1.085 (is. In other words, Timer 0 counts up each 1.085 us resulting in delay = number of counts x 1.085 µs.

The number of counts for the rollover is FFFFH – FFF2H = ODH (13 decimal). However, we add one to 13 because of the extra clock needed when it rolls over from FFFF to 0 and raises the TF flag. This gives 14 x 1.085 µs = 15.19 µs for half the pulse. For the entire period T = 2 x 15.19 µs = 30.38 µs gives us the time delay generated by the timer.

8051 Serial Ports

Baud rate in the 8051

The 8051 transfers and receives data serially at many different baud rates. The baud rate in the 8051 is programmable. This is done with the help of Timer 1. Before we discuss how to do that, we will look at the relationship between the crystal frequency and the baud rate in the 8051.

As discussed earlier, the 8051 divides the crystal frequency by 12 to get the machine cycle frequency. In the case of XTAL = 11.0592 MHz, the machine cycle frequency is 921.6 kHz (11.0592 MHz / 12 = 921.6 kHz). The 8051 's serial communication UART circuitry divides the machine cycle frequency of 921.6 kHz by 32 once more before it is used by Timer 1 to set the baud rate. Therefore, 921.6 kHz divided by 32 gives 28,800 Hz. This is the number we will use throughout this section to find the Timer 1 value to set the baud rate. When Timer 1 is used to set the baud rate it must be programmed in mode 2 that is 8-bit, auto-reload. To get baud rates compatible with the PC, we must load TH1 with the values shown in Table 1. Example 5 shows how to verify the data in Table 1.

Table 1 : Timer 1 TH1 Register Values for Various Baud Rates

Baud Rate	TH1 (Decimal)	TH1 (Hex)
9600	–3	FD
4800	–6	FA
2400	–12	F4
1200	–24	E8

Note: **XTAL = 11.0592 MHz.**

Example 5

With XTAL = 11.0592 MHz, find the TH1 value needed to have the following baud rates.

(a) 9600 (b) 2400 (c) 1200

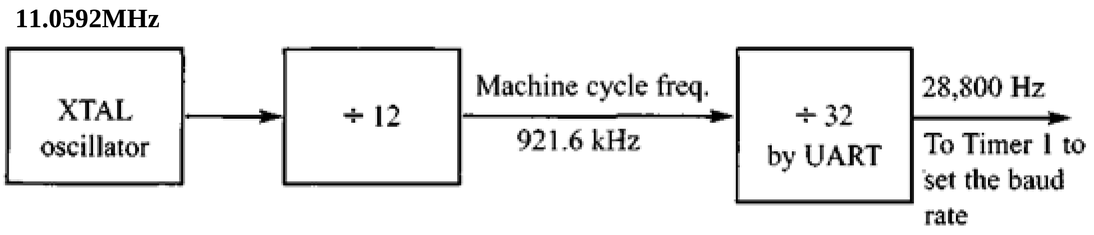
Solution:

With XTAL = 11.0592 MHz, we have:

The machine cycle frequency of the 8051 = 11.0592 MHz / 12 = 921.6 kHz, and 921.6 kHz / 32 = 28,800 Hz is the frequency provided by UART to Timer 1 to set baud rate.

(a) 28,800 / 3 = 9600	where –3 = FD (hex) is loaded into TH1
(b) 28,800 / 12 = 2400	where –12 = F4 (hex) is loaded into TH1
(c) 28,800 / 24 = 1200	where –24 = E8 (hex) is loaded into TH1

Notice that 1/12th of the crystal frequency divided by 32 is the default value upon activation of the 8051 RESET pin.



SBUF register

SBUF is an 8-bit register used solely for serial communication in the 8051. For a byte of data to be transferred via the TxD line, it must be placed in the SBUF register. Similarly, SBUF holds the byte of data when it is received by the 8051's RxD line. SBUF can be accessed like any other register in the 8051. Look at the following examples of how this register is accessed:

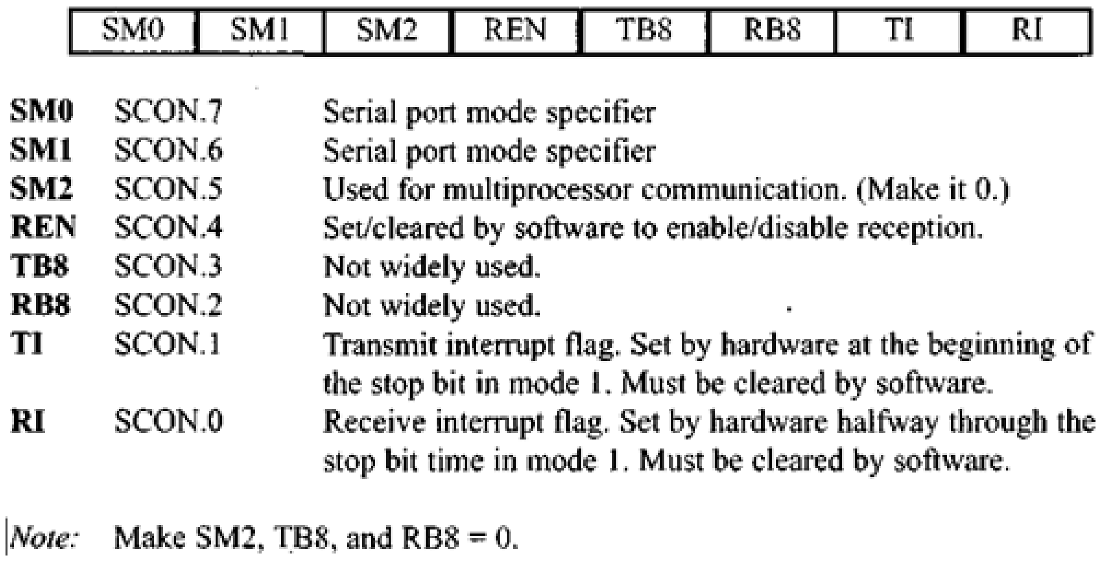
```
MOV SBUF,#'D'        ;load SBUF=44H, ASCII for 'D'
MOV SBUF,A           ;copy accumulator into SBUF
MOV A,SBUF           ;copy SBUF into accumulator
```

The moment a byte is written into SBUF, it is framed with the start and stop bits and transferred serially via the TxD pin. Similarly, when the bits are received serially via RxD, the 8051 deframes it by eliminating the stop and start bits, making a byte out of the data received, and then placing it in the SBUF.

SCON (serial control) register

The SCON register is an 8-bit register used to program the start bit, stop bit, and data bits of data framing, among other things.

The following figure describes various bits of the SCON register.



SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier
SM2	SCON.5	Used for multiprocessor communication. (Make it 0.)
REN	SCON.4	Set/cleared by software to enable/disable reception.
TB8	SCON.3	Not widely used.
RB8	SCON.2	Not widely used.
TI	SCON.1	Transmit interrupt flag. Set by hardware at the beginning of the stop bit in mode 1. Must be cleared by software.
RI	SCON.0	Receive interrupt flag. Set by hardware halfway through the stop bit time in mode 1. Must be cleared by software.

Note: Make SM2, TB8, and RB8 = 0.

Figure: SCON Serial Port Control Register (Bit-Addressable)

SMO, *SM1*

SMO and SMI are D7 and D6 of the SCON register, respectively. These two bits determine the framing of data by specifying the number of bits per character, and the start and stop bits. They take the following combinations.

SM0	*SM1*	
0	0	Serial Mode 0
0	1	Serial Mode 1, 8-bit data, 1 stop bit, 1 start bit
1	0	Serial Mode 2
1	1	Serial Mode 3

Of the 4 serial modes, only mode 1 is of interest to us. In the SCON register, when serial mode 1 is chosen, the data framing is 8 bits, 1 stop bit, and 1 start bit, which makes it compatible with the COM port of IBM/compatible PCs. More importantly, serial mode 1 allows the baud rate to be variable and is set by Timer 1 of the 8051. In serial mode 1, for each character a total of 10 bits are transferred, where the first bit is the start bit, followed by 8 bits of data, and finally 1 stop bit.

SM2

SM2 is the D5 bit of the SCON register. This bit enables the multiprocessing capability of the 8051. For our applications, we will make SM2 = 0 since we are not using the 8051 in a multiprocessor environment.

REN

The REN (receive enable), bit is D4 of the SCON register. The REN bit is also referred to as SCON.4 since SCON is a bit-addressable register. When the REN bit is high, it allows the 8051 to receive data on the RxD pin of the 8051. As a result if we want the 8051 to both transfer and receive data, REN must be set to 1. By making REN = 0, the receiver is disabled. Making REN — 1 or REN = 0 can be achieved by the instructions “SETB SCON.4″ and “CLR SCON.4″, respectively. Notice that these instructions use the bit-addressable features of register SCON. This bit can be used to block any serial data reception and is an extremely important bit in the SCON register.

TBS

TBS (transfer bit 8) is bit D3 of SCON. It is used for serial modes 2 and 3. We make TBS = 0 since it is not used in our applications.

RB8

RB8 (receive bit 8) is bit D2 of the SCON register. In serial mode 1, this bit gets a copy of the stop bit when an 8-bit data is received. This bit (as is the case for TBS) is rarely used anymore. In all our applications we will make RB8 = 0. Like TB8, the RB8 bit is also used in serial modes 2 and 3.

TI

TI (transmit interrupt) is bit Dl of the SCON register. This is an extremely important flag bit in the SCON register. When the 8051 finishes the transfer of the 8-bit character, it raises the TI flag to indicate that it is ready to transfer another byte. The TI bit is raised at the beginning of the stop bit.

RI

RI (receive interrupt) is the DO bit of the SCON register. This is another extremely important flag bit in the SCON register. When the 8051 receives data serially via RxD, it gets rid of the start and stop bits and places the byte in the SBUF register. Then it raises the RI flag bit to indicate that a byte has been received and should be picked up before it is lost. RI is raised halfway through the stop bit.

Programming the 8051 to transfer data serially

In programming the 8051 to transfer character bytes serially, the following steps must be taken.

1. The TMOD register is loaded with the value 20H, indicating the use of Timer 1 in mode 2 (8-bit auto-reload) to set the baud rate.
2. The TH1 is loaded with one of the values in Table 1 to set the baud rate for serial data transfer (assuming XTAL = 11.0592 MHz).
3. The SCON register is loaded with the value 50H, indicating serial mode 1, where an 8-bit data is framed with start and stop bits.
4. TR1 is set to 1 to start Timer 1.
5. TI is cleared by the "CLR TI" instruction.
6. The character byte to be transferred serially is written into the SBUF register.
7. The TI flag bit is monitored with the use of the instruction" JNB TI, xx" to see if the character has been transferred completely.
8. To transfer the next character, go to Step 5.

Importance of the TI flag

To understand the importance of the role of TI, look at the following sequence of steps that the 8051 goes through in transmitting a character via TxD.

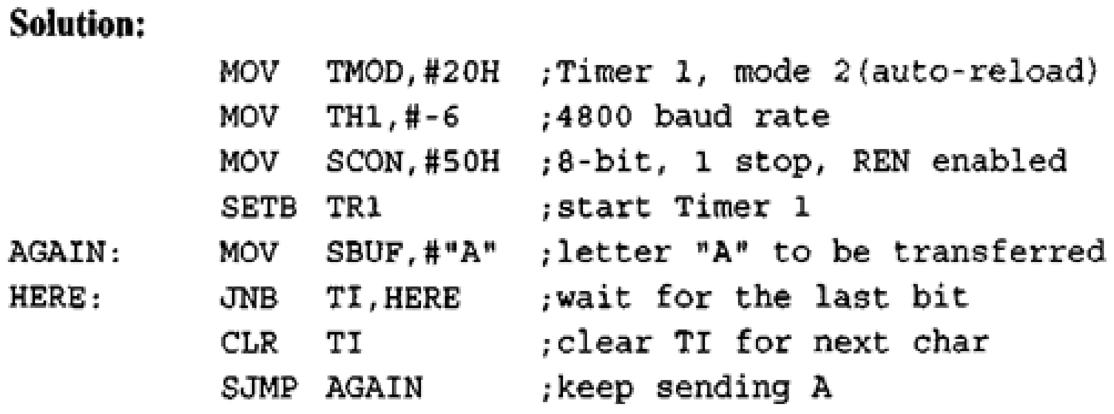
1. The byte character to be transmitted is written into the SBUF register.
2. The start bit is transferred.
3. The 8-bit character is transferred one bit at a time.
4. The stop bit is transferred. It is during the transfer of the stop bit that the 8051 raises the TI flag (TI =1), indicating that the last character was transmitted and it is ready to transfer the next character.

5. By monitoring the TI flag, we make sure that we are not overloading the SBUF register. If we write another byte into the SBUF register before TI is raised, the un-transmitted portion of the previous byte will be lost. In other words, when the 8051 finishes transferring a byte, it raises the TI flag to indicate it is ready for the next character.
6. After SBUF is loaded with a new byte, the TI flag bit must be forced to 0 by the "CLR TI" instruction in order for this new byte to be transferred.

From the above discussion we conclude that by checking the TI flag bit, we know whether or not the 8051 is ready to transfer another byte. More importantly, it must be noted that the TI flag bit is raised by the 8051 itself when it finishes the transfer of data, whereas it must be cleared by the programmer with an instruction such as "CLR TI". It also must be noted that if we write a byte into SBUF before the TI flag bit is raised, we risk the loss of a portion of the byte being transferred. The TI flag bit can be checked by the instruction "JNB TI, ..." or we can use an interrupt.

Example 6:

Write a program for 8051 to transfer letter "A" serially at 4800 baud rate, continuously.



Solution:

```
          MOV   TMOD,#20H  ;Timer 1, mode 2(auto-reload)
          MOV   TH1,#-6    ;4800 baud rate
          MOV   SCON,#50H  ;8-bit, 1 stop, REN enabled
          SETB  TR1        ;start Timer 1
AGAIN:    MOV   SBUF,#"A"  ;letter "A" to be transferred
HERE:     JNB   TI,HERE    ;wait for the last bit
          CLR   TI         ;clear TI for next char
          SJMP  AGAIN      ;keep sending A
```

Example 7:

Write a program to transfer the message "YES" serially at 9600 baud, 8-bit data, 1 stop bit. Do this continuously.

Solution:

```
            MOV    TMOD,#20H ;Timer 1, mode 2
            MOV    TH1,#-3   ;9600 baud
            MOV    SCON,#50H ;8-bit, 1 stop bit, REN enabled
            SETB   TR1       ;start Timer 1
AGAIN:      MOV    A,#"Y"    ;transfer "Y"
            ACALL  TRANS
            MOV    A,#"E"    ;transfer "E"
            ACALL  TRANS
            MOV    A,#"S"    ;transfer "S"
            ACALL  TRANS
            SJMP   AGAIN     ;keep doing it
;-----serial data transfer subroutine
TRANS:      MOV    SBUF,A    ;load SBUF
HERE:       JNB    TI,HERE   ;wait for last bit to transfer
            CLR    TI        ;get ready for next byte
            RET
```

Programming the 8051 to receive data serially

In the programming of the 8051 to receive character bytes serially, the following steps must be taken.

1. The TMOD register is loaded with the value 20H, indicating the use of Timer 1 in mode 2 (8bit auto reload) to set the baud rate.

2. TH1 is loaded with one of the values in Table 104 to set the baud rate (assuming XTAL = 11.0592MHz).

3. The SCON register is loaded with the value 50H, indicating serial mode 1, where 8bit data is framed with start and stop bits and receive enable is turned on.

4. TR1 is set to 1 to start Timer 1.

5. RI is cleared with the “CLR RI” instruction.

6. The RI flag bit is monitored with the use of the instruction “JNB RI, xx” to see if an entire character has been received yet.

7. When RI is raised, SBUF has the byte. Its contents are moved into a safe place.

8. To receive the next character, go to Step 5.

Example 8:

Program the 8051 to receive bytes of data serially, and put them in PI. Set the baud rate at 4800, 8bit data, and 1 stop bit.

```
              MOV   TMOD,#20H  ;Timer 1, mode 2(auto-reload)
              MOV   TH1,#-6    ;4800 baud
              MOV   SCON,#50H  ;8-bit, 1 stop, REN enabled
              SETB  TR1        ;start Timer 1
HERE:         JNB   RI,HERE    ;wait for char to come in
              MOV   A,SBUF     ;save incoming byte in A
              MOV   P1,A       ;send to port 1
              CLR   RI         ;get ready to receive next byte
              SJMP  HERE       ;keep getting data
```

Importance of the Rl flag bit

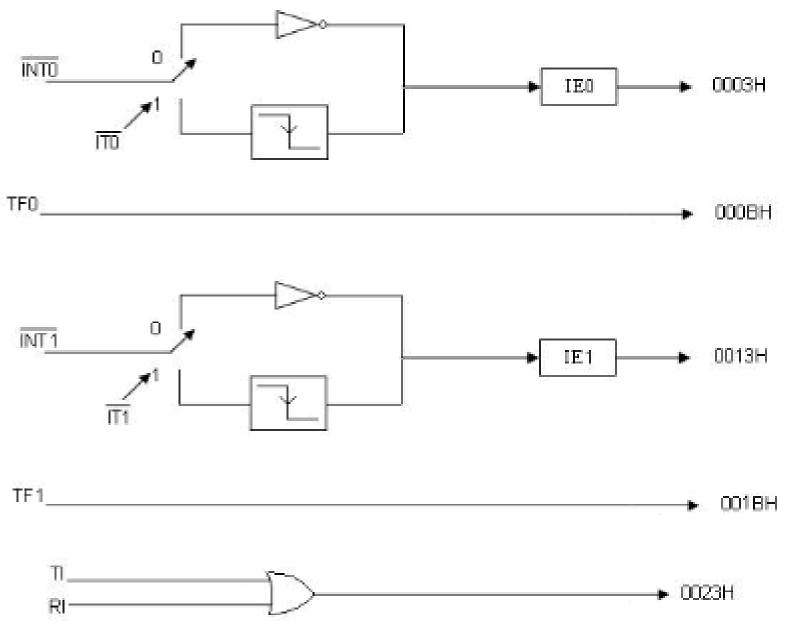
In receiving bits via its RxD pin, the 8051 goes through the following steps.

1. It receives the start bit indicating that the next bit is the first bit of the character byte it is about to receive.
2. The 8bit character is received one bit at time. When the last bit is received, a byte is formed and placed in SBUF.
3. The stop bit is received. When receiving the stop bit the 8051 makes RI = 1, indicating that an entire character byte has been received and must be picked up before it gets overwritten by an incoming character.
4. By checking the RI flag bit when it is raised, we know that a character has been received and is sitting in the SBUF register. We copy the SBUF contents to a safe place in some other register or memory before it is lost.
5. After the SBUF contents are copied into a safe place, the RI flag bit must be forced to 0 by the "CLR RI" instruction in order to allow the next received character byte to be placed in SBUF. Failure to do this causes loss of the received character.

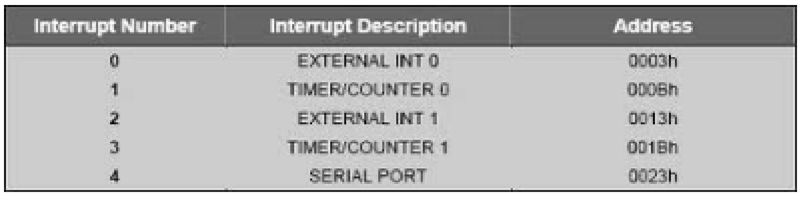
From the above discussion we conclude that by checking the RI flag bit we know whether or not the 8051 has received a character byte. If we fail to copy SBUF into a safe place, we risk the loss of the received byte. More importantly, it must be noted that the RI flag bit is raised by the 8051, but it must be cleared by the programmer with an instruction such as "CLR RI". It also must be noted that if we copy SBUF into a safe place before the RI flag bit is raised, we risk copying garbage. The RI flag bit can be checked by the instruction “JNB RI, xx” or by using an interrupt.

8051 Microcontroller Interrupt

- There are five interrupt sources for the 8051, which means that they can recognize 5 different events that can interrupt regular program execution.
- there are five interrupts : $\overline{\text{INT0}}$, TF0, $\overline{\text{INT1}}$, TF1, RI/TI
- Each interrupt can be enabled or disabled by setting bits of the IE register. Likewise, the whole interrupt system can be disabled by clearing the EA bit of the same register.
- Now, it is necessary to explain a few details referring to external interrupts- INT0 and INT1. If the IT0 and IT1 bits of the TCON register are set, an interrupt will be generated on high to low transition.

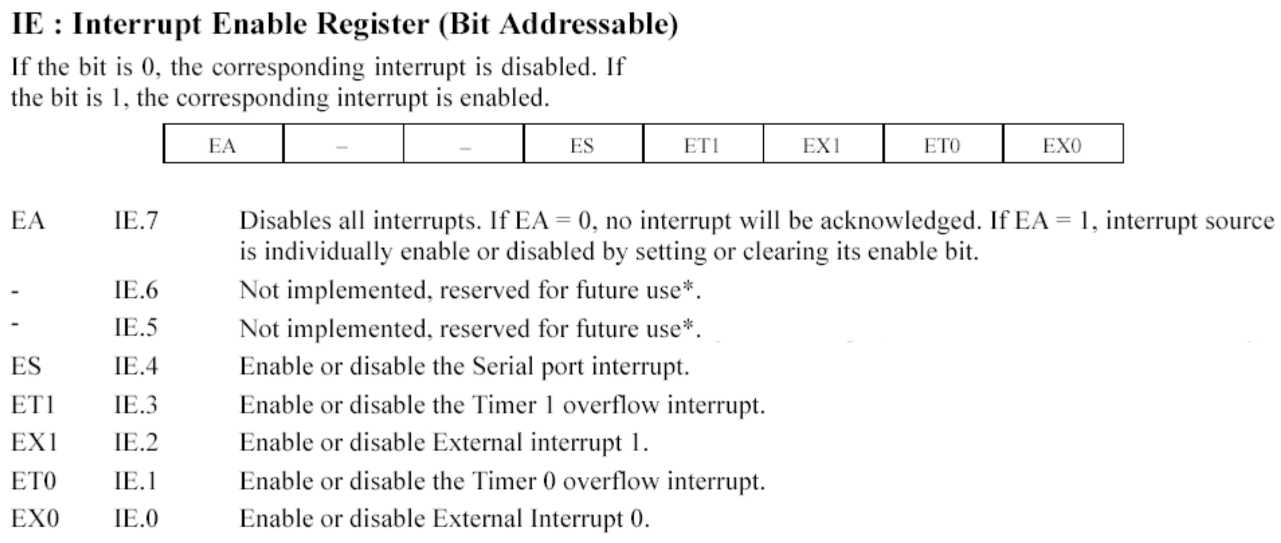


Here we are mention vector address of interrupt.



Interrupt Number	Interrupt Description	Address
0	EXTERNAL INT 0	0003h
1	TIMER/COUNTER 0	000Bh
2	EXTERNAL INT 1	0013h
3	TIMER/COUNTER 1	001Bh
4	SERIAL PORT	0023h

IE Register (Interrupt Enable)



IE : Interrupt Enable Register (Bit Addressable)

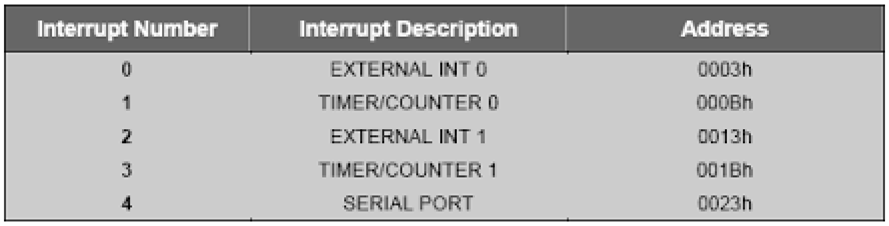
If the bit is 0, the corresponding interrupt is disabled. If
the bit is 1, the corresponding interrupt is enabled.

EA	–	–	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, interrupt source is individually enable or disabled by setting or clearing its enable bit.
-	IE.6	Not implemented, reserved for future use*.
-	IE.5	Not implemented, reserved for future use*.
ES	IE.4	Enable or disable the Serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

Priority level structure

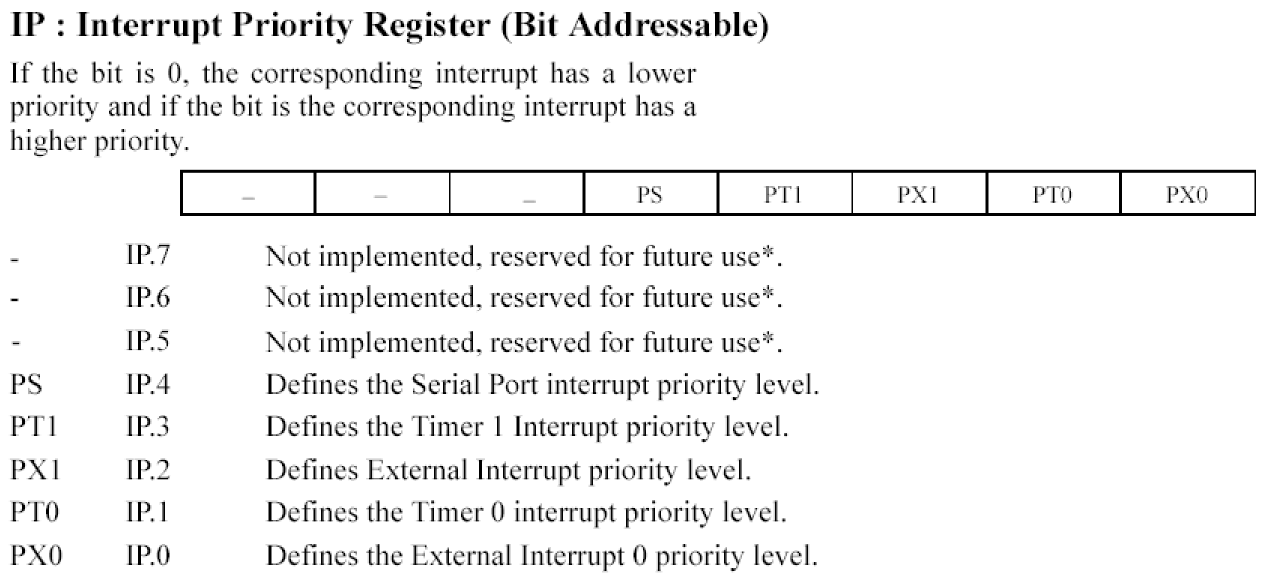
- Each interrupt source can be programmed to have one of the two priority levels by setting (high priority) or clearing (low priority) a bit in the IP (Interrupt Priority) Register.
- A low priority interrupt can itself be interrupted by a high priority interrupt, but not by another low priority interrupt.
- If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served.
- If the requests of the same priority level are received simultaneously, an internal polling sequence determines which request is to be serviced. Thus, within each priority level, there is a second priority level determined by the polling sequence, as follows.



Interrupt Number	Interrupt Description	Address
0	EXTERNAL INT 0	0003h
1	TIMER/COUNTER 0	000Bh
2	EXTERNAL INT 1	0013h
3	TIMER/COUNTER 1	001Bh
4	SERIAL PORT	0023h

IP Register (Interrupt Priority)

The IP register bits specify the priority level of each interrupt (high or low priority).



IP : Interrupt Priority Register (Bit Addressable)

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is the corresponding interrupt has a higher priority.

–	–	–	PS	PT1	PX1	PT0	PX0

-	IP.7	Not implemented, reserved for future use*.
-	IP.6	Not implemented, reserved for future use*.
-	IP.5	Not implemented, reserved for future use*.
PS	IP.4	Defines the Serial Port interrupt priority level.
PT1	IP.3	Defines the Timer 1 Interrupt priority level.
PX1	IP.2	Defines External Interrupt priority level.
PT0	IP.1	Defines the Timer 0 interrupt priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level.

Interrupt handling

- The interrupt flags are sampled at P2 of S5 of every instruction cycle (Note that every instruction cycle has six states each consisting of P1 and P2 pulses).
- The samples are polled during the next machine cycle (or instruction cycle). If one of the flags was set at S5P2 of the preceding instruction cycle, the polling detects it and the interrupt process generates a long call (LCALL) to the appropriate vector location of the interrupt.
- The LCALL is generated provided this hardware generated LCALL is not blocked by any one of the following conditions.
- An interrupt of equal or higher priority level is already in progress.
- The current polling cycle is not the final cycle in the execution of the instruction in progress.
- The instruction in progress is RETI or any write to IE or IP registers.