Rajdhani

DATE / /

Experiment no: 5.

Aim :-

Theory :- The 8051 microcontroller can recognize five different events that cause the main program to interrupt from the normal execution these five sources of interrupts in 8051 are:

1) Timer 0 overflow interrupt - TF0
2) Timer 1 overflow interrupt - TF1
3) External hardware interrupt interrupt INT0
4) External hardware interrupt INT1
5) Serial communication interrupt - RI/TI

The timer & serial interrupts are internally generated by the microcontroller whereas the external interrupts are generated by additional interfacing devices or switches that are externally connected to the controller when an interrupts occurs the controller executes the interrupt service routine so that memory location ~~interrupt~~ corresponds to the interrupt that enables if the interrupt corresponding to that memory location is given in the interrupt vector table below.

Interrupt NO.	Interrupt Discription	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.

o Registers used for 8051 microcontroller Interrupt

1) Interrupt Enable (IE) register:
This register is responsible for enabling disabling the interrupt.

EA	—	—	ES	ET1	EX2	ET0	EX0

EA - Disables all interrupts.
0 - no interrupt will be acknowleged.
1 - Interrupt sources is enabled.
ES - Enable or disable the serial port interrupt
ET1 - Enable or disable the timer 1 overflow interrupt.
EV1 - Enable or disable External interrupt 1.
ET0 - Enable or disable timer 0 overflow Interrupt.
EX0 - Enable or disable External Interrupt 0.

2) Interrupt priority register (IP):
It is also possible to change the priority levels of interrupts by Setting or clearing the corresponding bit in the interrupt priority (IP) register.

—	—	PT2	PS	PT1	PX1	PT0	PX0

bit 7 bit 0

- 2p.7 - Reserved for future use.
- 2p.6 - Reserved for future use.
PS - It defines the serial port interrupt priority level.
PT1 - It defines the timer interrupt of 1 priority.
PX1 - It defines the external interrupt 1 priority level.
PT0 - It defines the timer 0 interrupt priority level.
PX0 - It defines the external interrupt of 0 priority level.

Rajdhani
DATE / /

3) TCON (Timer / counter control Register):

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 - Timer 1 overflow flag set by hardware on timer/counter overflow cleared when interrupt processed.

TR1 - Timer 1 Run control bit
1 - Timer 1 Start.
0 - Timer 1 off (stop).

TF0 - Timer 0 overflow flag set by hardware on timer / counter overflow cleared when interrupt processed.

IE1 - Interrupt 1 Edge flag set by hardware when external interrupt Edge detected cleared when interrupt processed.

IT1 - Interrupt 0 Edge flag set by hardware when external interrupt edge detected cleared when interrupt processed.

IT0 - Interrupt 0 type control bit set/ cleared by software to specify falling edge / low level triggered external interrupt.

0 Interrupt programming steps in 8051 microcontroller
1) selecting the timer by configuring TMOD register. first mode of operation
2) choosing & loading the Initial values of TLx & THx for appropriate modes.
3) Enabling the IE registers & corresponding timer bit in it.

4) Setting the timer run bit to start the tim
5) write the subroutine for the timer for tim required & clear timer value TRx at the end of subroutine.

• program:

1) write a c program that continuously gets a single bit of data from p1.7 & sends to p1.0 while simultaneously creating square wave of 200 µs period on pin p2.5 Use timer0 to create the square wave Assume that XTAL = 11.0592 MHZ.

⇒ We will use timer 0 in mode 2 (auto reload) one half of the period is 100 µs.

100/1.085 µs = 92, & TH0 = 256 − 92 = 164 or A4H

```
#include <reg51.h>
sbit SW = P1^7;
sbit IND = P1^0;
sbit wave = P2^5;
  void timer0(void) interrupt1
{
  wave = ~wave //Toggle pin
}
void main(){
   SW = 1; // Make switch input
TMod = 0x02
TH0 = 0xA4; //TH0 = -92.
```

Rajdhani
DATE / /

```
IE = 0x81 ; // enable interrupts for
                // Timer 0
TR0 = 1          // Start Timer 0
while (1) {
    IND = SW; // send switch to LED
}
```

200 µs / 2 = 100 µs
100 µs / 1085 µs = 92.

2)
```
    ORG 0000H
    LJMP MAIN
    ORG 0003H
    MOV P2, #00H
    RETI
MAIN: MOV IE, #81H
LOOP: ADD A, #01H
    MOV P2, #OFFH
    SJMP LOOP
    END.
```

Conclusion :-
By using the concept of interrupt we can produce desired output by changing the direction of µC according to the user.

C

13/5/22