Lab Experiments with Pt51 Card: Timers

In this set of experiments, we familiarize ourselves with the 8051 timers and interrupt system. Please review the tutorial on timers appneded to this document.

The 8051 family provides a minimum of 2 timers, which can be used for timing/counting applications without tying up the processor. In fact most of the newer chips in the family (including 89C5131A), provide 3 timers.

In default conditions, the timer count advances every 12 clock cycles. This means that the Pt51 board, which uses a 24 MHz crystal, provides 500 ns delay for each count. Thus a maximum of about 32 ms of delay can be provided by a timer in the usual 16 bit mode (mode 1).

For longer delays, we can program the timer to generate an interrupt when it rolls over. The interrupt service routine can then increment a count on every interrupt. About a second of delay can be generated for every 30 to 31 interrupts. When the count reaches a pre-set value depending on the delay required, the interrupt routine can reset the counter and start counting up again.

Homework Assignment

Program timer T0 to generate the maximum delay and cause an interrupt on time out. The interrupt service routine should increment a counter till it reaches 40. Every time the count reaches 40, an LED should be toggled and the count reset to 0.
By measuring the time taken by the LED to toggle 20 times, adjust the timer count, such that the toggle time is as close to 1 second as possible. (20 toggles should take 20 seconds).

Lab work

1. Using the 1 second timer developed as homework, implement a clock which displays the time on the LCD unit. (This will involve separate memory locations for seconds, minutes and hours). At the end of 1 second interval, advance the seconds count. When

it reaches 60, reset it to zero and advance the minutes count. when that reaches 60, reset it to zero and advance the hours count. Reset the hours count to 01 when it reaches 13. Initialize the hours, minutes and seconds count to 12, 59 and 30 respectively. Display the time on the LCD in hh:mm:ss format.

2. Use timer T1 to debounce the switches on Pt51 board. Program T1 for a delay of 20 ms. Read the switch twice at intervals of 20 ms. Reject the new value if it is different and re-read after another 20 ms. If two values are the same, accept it and report it as the switch status.

3. Use the debounced switches to implement a facility for setting the clock. If Switch 0 is on, the program should enter the time set mode. In the time set mode, every toggle on switch 1 should advance minutes by 1 while every toggle on switch2 should advance the hours by 1. Returning switch 0 to off should start showing time from the set value onwards and the usual clock display as implemented in the earlier experiment should run. In this mode, check the status of switch 0 every minute and go to the set mode if switch 0 is ON.

8051 Interrupts

Interrupt Sources

The 8051 architecture can handle interrupts from 5 sources. These are: the two external interrupt lines, two timers and the serial interface. Each one of these is assigned an interrupt vector address. This is quite similar to the RST interrupt vectors in the case of 8085.

External Interrupts

Port P3 of 8051 is a multi-function port. Different lines of this port carry out functions which are additional to data input-output on the port.

Additional functions of Port 3 lines

Port Line	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
Function	$\overline{\text{RD}}$	$\overline{\text{WR}}$	T1 in	T0 in	$\overline{\text{INT1}}$	$\overline{\text{INT0}}$	TxD	RxD

Lines P3.2 and P3.3 can be used as interrupt inputs. Interrupts will be caused by a 'LOW' level, or a negative edge on these lines. Half of the special function register TCON is used for setting the conditions for causing interrupts from external sources. This register is bit addressable.

SFR TCON at byte address 88H

Bit No.	7	6	5	4	3	2	1	0
Bit Name	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Bit Addr	8F	8E	8D	8C	8B	8A	89	88

IT1 and IT0 are the "Interrupt Type" flags for external sources 1 and 0 respecively. These decide whether a negative going edge or a 'LOW' level will cause an interrupt. If the bit is set, the corresponding interrupt is edge sensitive. If it is cleared, the interrupt is level sensitive. IE1 and IE0 are the status flags for the two external interrupt lines. If the flag is 1, the selected type of event (edge or level) has occured on the corresponding interrupt line.

Internal Interrupts

Internally generated interrupts can be from either timer, or from the serial interface. The serial interface causes interrupts due to a receive event (RI) or due to a transmit event (TI). The receive event occurs when the input buffer of the serial line (sbuf in) is full and a byte needs to be read from it. The transmit event indicates that a byte has been sent a new byte can be written to output buffer of the serial line (sbuf out).

8051 timers always count up. When their count rolls over from the maximum count to 0000, they set the corresponding timer flag TF1 or TF0 in TCON. Counters run only while their run flag (TR1 or TR0) is set by the user program. When the run flag is cleared, the count stops incrementing. The 8051 can be setup so that an interrupt occurs whenever TF1 or TF0 is set.

Enabling Interrupts

At power-up, all interrupts are disabled. Suppose Timer 0 is started. When it times out, TF0 in the special function register TCON will be set. However, this will not cause an interrupt. To enable interrupts, a number of steps need to be taken.

Interrupts are enabled in a manner which is quite similar to the 8085. There is an interrupt enable special function register IE at byte address A8H. This register is bit addressable. (The assembler gives special mnemonics to each bit address.)

SFR IE at byte address A8H

Bit No.	7	6	5	4	3	2	1	0
Function	IE	U	U	SI	TF1	Ex1	TF0	Ex0
Bit Addr	AF	AE	AD	AC	AB	AA	A9	A8
Bit Name	EA	-	-	ES	ET1	EX1	ET0	EX0

The most significant bit of the register is a global interrupt enable flag. This bit must be set in order to enable any interrupt. Bits 6 and 5 are undefined for 8051. (Bit 5 is used by 8052 for the third timer available in 8052). Bit 4, when set, enables interrupts from the serial port. Bit 3 should be set to enable interrupts from Timer 1 overflow. Bit 2 is set to enable interrupts from external interrupt 1 (pin P3.3 on Port 3). Bit 1 enables interrupts from Timer 0 when it overflows. Bit 0, when set, will enable interrupts from external interrupt 0 (pin P3.2 on Port 3).

Interrupt Vectors

When an interrupt occurs, the updated PC is pushed on the stack and is loaded with the vector address corresponding to the interrupt. The following table gives the vector addresses. The order of entries in the table is also the order in which the 8051 will poll these in case of multiple interrupts.

Interrupt Source	Vector address
External Interrupt 0	0003H
Timer 0 Overflow	000BH
External Interrupt 1	0013H
Timer 1 Overflow	001BH
Serial Interface	0023H

8051 starts executing from address 0000H at power-up or reset. The first 3 bytes are typically used for placing a long jump instruction to start of the code area. The interrupt vectors start from 0003 and are separated by 8 bytes from each other. Many simple interrupt handlers can be accommodated in this space. Otherwise, jump instructions (to handler locations) need to be placed at the vector addresses. This is quite similar to the RST handlers for 8085.

Thus, to enable interrupts from T0, we have to do

```
SetB   EA    ;(or SetB IE.7) to enable interrupts
SetB   ET0   ;(or SetB IE.1) to enable interrupts from T0
```

After this, whenever T0 overflows, TF0 will be set (in SFR TCON), the currently running program will be interrupted, its PC value will be put on the stack (PC-L first, PC-H after – because the stack grows upwards in 8051), and PC will be loaded with 000B H. The interrupt handler for T0 should be placed here, and it should end with the instruction:
RETI

Interrupt Priorities

8051 has two levels of interrupt priorities: high or low. By assigning priorities, we can control the order in which multiple interrupts will be serviced. Priorities are set by bits in a special function register called IP, which is at the byte address B8H. This register is also bit addressable. The assembler defines special names for bits of this register.

SFR IP at byte address B8H

Bit No.	7	6	5	4	3	2	1	0
Bit Addr	BF	BE	BD	BC	BB	BA	B9	B8
Bit Name	U	U	U	PS	PT1	PX1	PT0	PX0

Notice that the bits are in the polling order of interrupts. A 1 in a bit position assigns a high priority to the corresponding source of interrupts – a 0 gives it a low priority. In case of multiple interrupts, the following rules apply:

- While a low priority interrupt handler is running, if a high priority interrupt arrives, the handler will be interrupted and the high priority handler will run. When the high priority handler does 'RETI', the low priority handler will resume. When this handler does 'RETI', control is passed back to the main program.
- If a high priority interrupt is running, it cannot be interrupted by any other source – even if it is a high priority interrupt which is higher in polling order.
- A low-priority interrupt handler will be invoked only if no other interrupt is already executing. Again, the low priority interrupt cannot preempt another low priority interrupt, even if the later one is higher in polling order.
- If two interrupts occur at the same time, the interrupt with higher priority will execute first. If both interrupts are of the same priority, the interrupt which is higher in polling sequence will be executed first. This is the only context in which the polling sequence matters.

Serial Interrupts

Serial interrupts are handled somewhat differently from the timers. There are independent interrupt flags for reception and transmission of serial data, called RI and TI. RI indicates that a byte has been received and is available for reading in the input buffer. TI indicates that the previous byte has been sent serially and a new byte can be written to the serial port. A serial interrupt occurs if *either* of these flags is set. (Of course the serial interrupt must be enabled for this to occur). The interrupt service routine should check which of these events caused the interrupt. This can be done by examining the flags. Either or both of the flag might be set, requiring a read from or write to the serial buffer sbuf (or both). Recall that the input and output buffers are distinct but are located at the same address. A read from this address reads the input buffer while a write to the same address writes to the output buffer. The RI and TI flags are *not* automatically cleared when an interrupt is serviced. Therefore, the interrupt service routine must clear them before returning. Here is an example handler for serial interrupts:

```
Serial_ISR:
        PUSH PSW      ; Save flags and context
        PUSH ACC      ; and accumulator
        JNB  RI,output; If RI not set, check for TI
        MOV  A, SBUF
        MOV  inchar, A; Save this character
        CLR  RI       ; clear receive interrupt flag
output:               ; Check if output is required
```

```
        JNB TI, done  ; If no transmit interrupt, leave
        MOV outchar, A; Else get the char to send
        MOV sbuf, A   ; Write to serial buffer
        CLR TI        ; Clear Transmit interrupt flag
done:   POP ACC       ; Restore Accumulator
        POP PSW       ; and flags
        RETI          ; and return
```

Sequence of Events after an interrupt

When an enabled interrupt occurs,

1. The PC is saved on the stack, low byte first. Notice that this order is different from 8085. This is because the stack grows upwards in 8051.

2. Other interrupts of lower priority and same priority are disabled.

3. Except for the serial interrupt, the corresponding interrupt flag is cleared.

4. PC is loaded with the vector address corresponding to the interrupt.

When the handler executes 'RETI"

1. PC is restored by popping the stack.

2. Interrupt status is restored to its original value. (Same and lower priority interrupts restored to original status).