Interrupt
Interrupt Programming

• An interrupt is an external or internal event that interrupts the microcontroller to inform it that a device needs its service.

• A single 8051 can serve several devices. There are 2 ways to do that
  – Interrupt: interrupt service routine (ISR) or interrupt handler
  – polling
Interrupt

• Whenever any device needs its service, the device notifies the microcontroller by sending it an interrupt signal.
• Upon receiving an interrupt signal, the microcontroller interrupts whatever it is doing and serves the devices.
Polling

- The microcontroller continuously monitors the status of a given device; when the condition is met, it performs the service.
- After that, it moves on the monitor the next device until everyone is serviced.

```
8051
Device 1
Device 2
...
```

This is NOT efficient!!
Interrupt vs. Polling

- The interrupt method is preferable because the polling method wastes much of the microcontroller’s time by polling devices that do not need service.
- For interrupt method, during the waiting, you could do other tasks and when the TF is raised, it will interrupt the microcontroller in what it is doing.
- Monitoring by polling method

\[
\text{Target: } JNB \ TF, \ Target
\]

Doing waiting, you could not do anything!!
Interrupt Service Routine (ISR)

- For every interrupt, there is a fixed location in memory that holds the address of its ISR.
- The group of memory location set aside to hold the addresses of ISRs is called “Interrupt Vector Table”
- **Interrupt Vector Table**

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Description</th>
<th>ROM locations</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>Reset interrupt</td>
<td>0000 ~ 0002 (3B)</td>
<td>9</td>
</tr>
<tr>
<td>INT0</td>
<td>External H/W Interrupt 0</td>
<td>0003 ~ 000A (8B)</td>
<td>12  (P3.2)</td>
</tr>
<tr>
<td>TF0</td>
<td>Timer 0 Interrupt</td>
<td>000B ~ 00012(8B)</td>
<td></td>
</tr>
<tr>
<td>INT1</td>
<td>External H/W Interrupt 1</td>
<td>0013 ~ 001A (8B)</td>
<td>13  (P3.3)</td>
</tr>
<tr>
<td>TF1</td>
<td>Timer 1 Interrupt</td>
<td>001B ~ 0022(8B)</td>
<td></td>
</tr>
<tr>
<td>RI and TI</td>
<td>Serial COM interrupt</td>
<td>0023 ~ 002A(8B)</td>
<td></td>
</tr>
</tbody>
</table>
Reset Interrupt

- There are 3 bytes of ROM spaces assigned to the reset pin.

\[
\begin{array}{c}
0000 \\
0001 \\
0002 \\
\end{array}
\]

- For this reason, in our program, we put LJMP as the first instruction and redirect the processor away from the interrupt vector table.

\[
\text{ORG 0} \\
\text{LJMP Main} \Rightarrow \text{3-byte instruction located to} \\
0000, 0001, \text{and} 0002
\]

\[
\text{ORG 30H} \\
\text{Main:} \quad \text{.........}
\]
## Interrupt Enable (IE) Register

- **Enabling and disabling an interrupt**

<table>
<thead>
<tr>
<th>EA</th>
<th>ET2</th>
<th>ES</th>
<th>ET1</th>
<th>EX1</th>
<th>ET0</th>
<th>EX0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  **IE**

- **EA:**
  - EA=0 ➞ No interrupt
  - EA=1 ➞ Interrupt

- **ET2** is for Timer 2 (in 8052)

- **ES**: enable/disable the serial port interrupt

- **ET1** (Timer 1 interrupt) and **ET0** (Timer 0 interrupt)

- **EX1** (External interrupt: INT1) and **EX0** (External interrupt INT0)
Example

• Q1: Show the instructions to (a) enable the serial interrupt, timer 0 interrupt, and external H/W interrupt 1, and (b) disable the timer 0 interrupt, and (c) show how to disable all the interrupt with a single instruction.
Timer Interrupts

- TF0 interrupt (for example of Timer 0)

\[
\text{TF0}
\]

- Steps
  1. IE register is enable for ET0.

\[
\begin{array}{cccccccc}
\text{EA} & \text{-} & \text{ET2} & \text{ES} & \text{ET1} & \text{EX1} & \text{ET0} & \text{EX0} \\
=1 & & & & & & =1 & \\
\end{array}
\]

  2. When the timer rolls over, TF0 is raised.

  3. Microcontroller is interrupted and jump to the interrupt vector table to serve the ISR. In this way, the microcontroller can do other things until it is notified that the timer 0 has rolled over.
Timer Interrupts

- **Step 1:**
  
  ORG 0
  LJMP Main
  ORG 30H
  
  Main: ……………

- **Step 2:**
  
  Timer 0 ➔ 000BH

- **Step 3:** (EA=1, and ET0 =1 for Timer 0)
  
  MOV IE, #10000010B
Example

- Q2: Write a program that continuously gets 8-bit data from P0 and sends it to P1 while simultaneously creating a square-wave of 200us period on Pin P2.1 (Use timer 0 to create the square wave. Assume XTAL=11.0592MHz)

- Q3: Rewrite Q2 to create a square wave as below

![Square Wave Diagram]

- Q4: Write a program for Timer 0 to generate a square wave of 50Hz frequency on P1.2 (a) using polling method, and (b) using interrupt method.
External H/W Interrupt

- !INT0 (0003 ~ 000AH) and !INT1 (0013 ~ 001AH)

- Level-Triggered Interrupt
  - Normally high (!INT0 and !INT1) ➔ when a low signal applied to them, it trigger the interrupt.
  - The low signal must be removed before RETI
Example

• Q5: Assume that the 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 and is normally off when it is turned on it should stay on for a fraction of a second. As long as the switch is pressed low, the LED should stay ON.
Sampling of the low level-triggered interrupt

- The INTx pin must be held in a low state until the start of the execution of ISR – This duration is around 4 machine cycle but no more.
- The INTx must be brought back to high before the execution of RETI
- IT0 (TCON.0) and IT1 (TCON.2) are both ‘0’, making external interrupts level-triggered.
Edge-Triggered Interrupt

- SETB TCON.0 (IT0)
- SETB TCON.2 (IT1)

Diagram:
- Pulse generator (edge-triggered)
- INT1
- P1.3 LED
Example

• Q6: Assume that P3.3 (INT1) is connected to a pulse generator, write a program in which the falling edge of the pulse will send a high to P1.3, which connected to a LED (or buzzer). In other words, the LED is turned ON and OFF at the same rate as the pulses are applied to the INT1 pin (edge-triggered).
Sampling of the edge-triggered interrupt

- The falling edge is located by 8051 and is held by the TCON register.
- Interrupt-in-service flags
  - TCON.1 (IE0) ➔ hold the latch falling edge for INT0
  - TCON.3 (IE1) ➔ hold the latch falling edge for INT1
- IE0=1 or IE1=1 ➔ in Interrupt service. They are only cleared by RETI, so
  - CLR TCON.1
  - CLR TCON.3
- are NOT necessary!!!
Serial Communication Interrupt

- Remind the SCON register

<table>
<thead>
<tr>
<th>SM0</th>
<th>SM1</th>
<th>SM2</th>
<th>REN</th>
<th>TB8</th>
<th>RB8</th>
<th>TI</th>
<th>RI</th>
</tr>
</thead>
</table>

- TI=1  SBUF is ready to transfer the next byte.
- RI=1  The received byte needs to be picked up before it is lost.
- Enable Serial comm. Interrupt  SETB IE.4

<table>
<thead>
<tr>
<th>EA</th>
<th>-</th>
<th>ET2</th>
<th>ES</th>
<th>ET1</th>
<th>EX1</th>
<th>ET0</th>
<th>EX0</th>
</tr>
</thead>
<tbody>
<tr>
<td>=1</td>
<td></td>
<td></td>
<td>=1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Serial Communication Interrupt

- The serial interrupt is used mainly for receiving data and is never used for sending data serially.
Example

- Q7: Write a program in which the 8051 reads data from P1 and writes it to P2 continuously while giving a copy of it to the serial com port to be transferred serially. (XTAL=11.0592MHz, and baud rate = 9600)
- Q8: Write a program in which the 8051 gets data from P1 and send it to P2 continuously while incoming data from the serial port is sent to P0. (XTAL=11.0592MHz, and baud rate = 9600)
- Q9: Write a program using interrupts to do the following (a) receive data serially and sent it to P0, (b) have P1 port read and transmitted serially, and a copy given to P2, and (c) make timer 0 generate a square wave of 5KHz frequency on P0.1 (XTAL=11.0592MHz, and baud rate = 4800)
Interrupt Flags

- For Serial Communication Interrupt
  - RI (SCON.0)
  - TI (SCON.1)

- For Timer 0 and Timer 1 interrupt
  - TF0 (TCON.5 for Timer 0)
  - TF1 (TCON.7 for Timer 1)

- For external HW interrupt
  - IE0 (TCON.1 for INT0)
  - IE1 (TCON.3 for INT1)
Interrupt Priority

- High to Low Priority
  - INT0
  - TF0
  - INT1
  - TF1
  - RI + TI
Interrupt Priority (IP) Register

- IP register

<table>
<thead>
<tr>
<th>-</th>
<th>-</th>
<th>PT2</th>
<th>PS</th>
<th>PT1</th>
<th>PX1</th>
<th>PT0</th>
<th>PX0</th>
</tr>
</thead>
</table>

- PT2 (timer 2 in 8052)
- PS (for serial Port)
- PT1 (for timer 1)
- PX1 (for external interrupt 1)
- PT0 (for timer 0)
- PX0 (for external interrupt 0)
Example

• Q10: (a) program the IP register to assign the highest priority to INT1 then (b) discuss what happen if INT0 and TF0 are activated at the same time. Assume the interrupts are both edge-triggered.

• Q11: Assume that after reset, the interrupt priority is set by the instruction “MOV IP, #00001100B”. Discuss the sequence in which the interrupts are serviced.