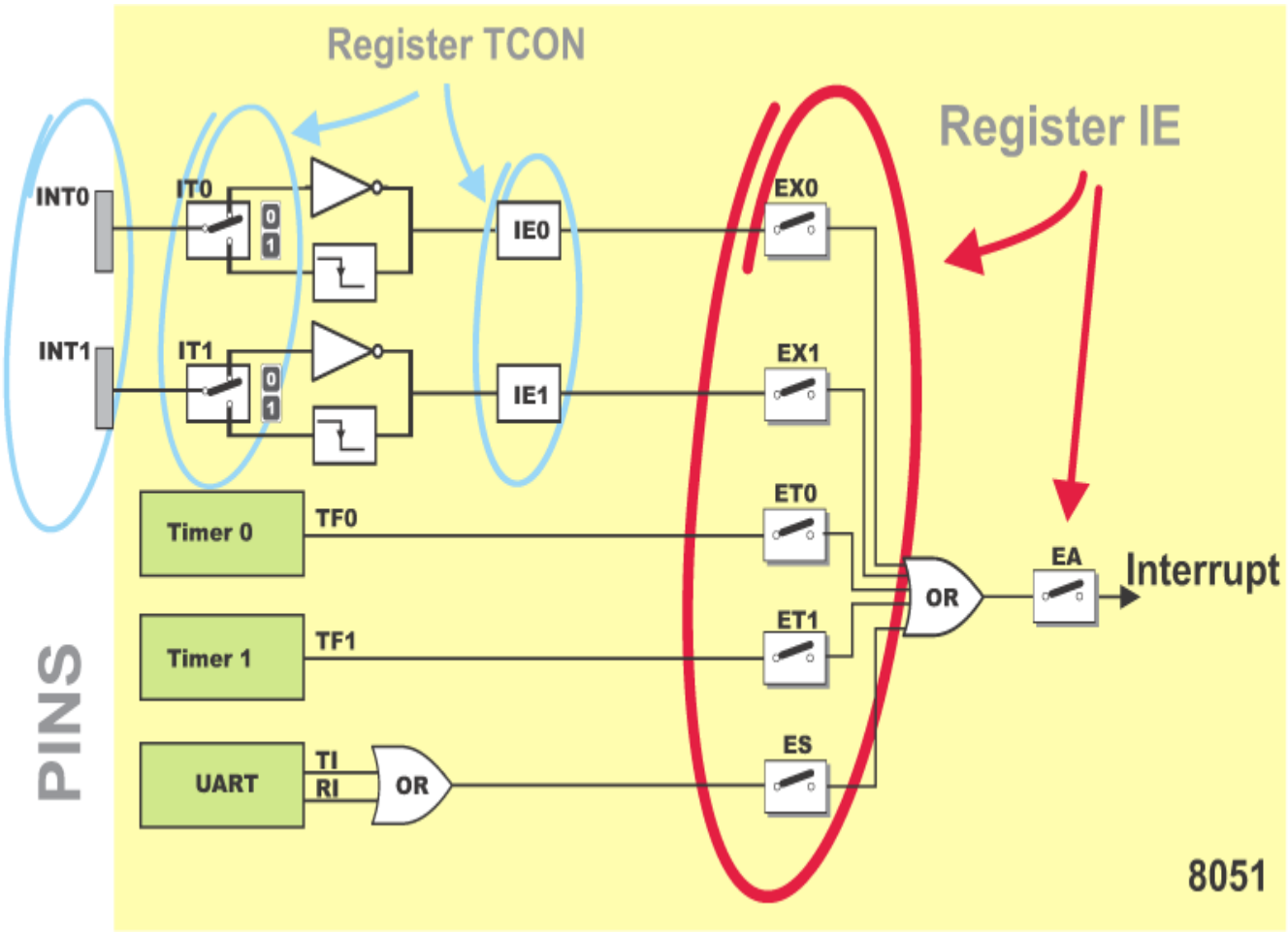
8051 Microcontroller Interrupts

There are five interrupt sources for the 8051, which means that they can recognize 5 different events that can interrupt regular program execution. 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. Refer to figure below.

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, i.e. on the falling pulse edge (only in that moment). If these bits are cleared, an interrupt will be continuously executed as far as the pins are held low.

- EA - global interrupt enable/disable:
  - 0 - disables all interrupt requests.
  - 1 - enables all individual interrupt requests.
- ES - enables or disables serial interrupt:
  - 0 - UART system cannot generate an interrupt.
  - 1 - UART system enables an interrupt.
- ET1 - bit enables or disables Timer 1 interrupt:
  - 0 - Timer 1 cannot generate an interrupt.
  - 1 - Timer 1 enables an interrupt.
- EX1 - bit enables or disables external 1 interrupt:
  - 0 - change of the pin INT0 logic state cannot generate an interrupt.
  - 1 - enables an external interrupt on the pin INT0 state change.

- ET0 - bit enables or disables timer 0 interrupt:
  - 0 - Timer 0 cannot generate an interrupt.
  - 1 - enables timer 0 interrupt.
- EX0 - bit enables or disables external 0 interrupt:
  - 0 - change of the INT1 pin logic state cannot generate an interrupt.
  - 1 - enables an external interrupt on the pin INT1 state change.



IE Register (Interrupt Enable)

	0	X	0	0	0	0	0	0	Value after Reset
IE	EA		ET2	ES	ET1	EX1	ET0	EX0	Bit name
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	

Interrupt Priorities

It is not possible to forseen when an interrupt request will arrive. If several interrupts are enabled, it may happen that while one of them is in progress, another one is requested. In order that the microcontroller knows whether to continue operation or meet a new interrupt request, there is a priority list instructing it what to do.

The priority list offers 3 levels of interrupt priority:

1. Reset! The apsolute master. When a reset request arrives, everything is stopped and the microcontroller restarts.
2. Interrupt priority 1 can be disabled by Reset only.
3. Interrupt priority 0 can be disabled by both Reset and interrupt priority 1.

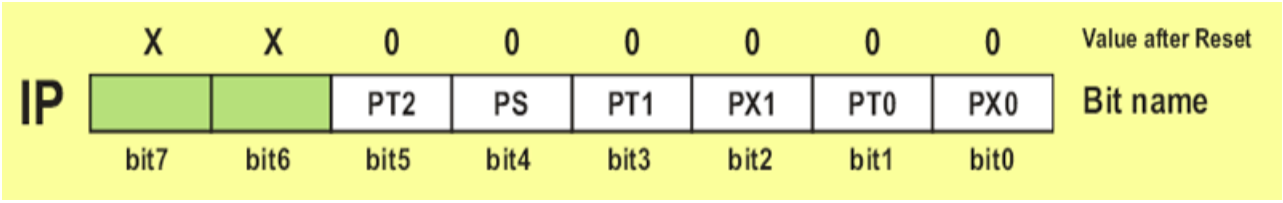
The IP Register (Interrupt Priority Register) specifies which one of existing interrupt sources have higher and which one has lower priority. Interrupt priority is usually specified at the beginning of the program. According to that, there are several possibilities:

- If an interrupt of higher priority arrives while an interrupt is in progress, it will be immediately stopped and the higher priority interrupt will be executed first.
- If two interrupt requests, at different priority levels, arrive at the same time then the higher priority interrupt is serviced first.
- If the both interrupt requests, at the same priority level, occur one after another, the one which came later has to wait until routine being in progress ends.
- If two interrupt requests of equal priority arrive at the same time then the interrupt to be serviced is selected according to the following priority list:

1. External interrupt INT0
2. Timer 0 interrupt
3. External Interrupt INT1
4. Timer 1 interrupt
5. Serial Communication Interrupt

IP Register (Interrupt Priority)

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



- PS - Serial Port Interrupt priority bit
  - Priority 0
  - Priority 1
- PT1 - Timer 1 interrupt priority
  - Priority 0
  - Priority 1
- PX1 - External Interrupt INT1 priority
  - Priority 0
  - Priority 1
- PT0 - Timer 0 Interrupt Priority
  - Priority 0
  - Priority 1
- PX0 - External Interrupt INT0 Priority
  - Priority 0
  - Priority 1

Handling Interrupt

When an interrupt request arrives the following occurs:

1. Instruction in progress is ended.
2. The address of the next instruction to execute is pushed on the stack.
3. Depending on which interrupt is requested, one of 5 vectors (addresses) is written to the program counter in accordance to the table below:
4.

Interrupt Source	Vector (address)
IE0	3 h
TF0	B h
TF1	1B h
RI, TI	23 h

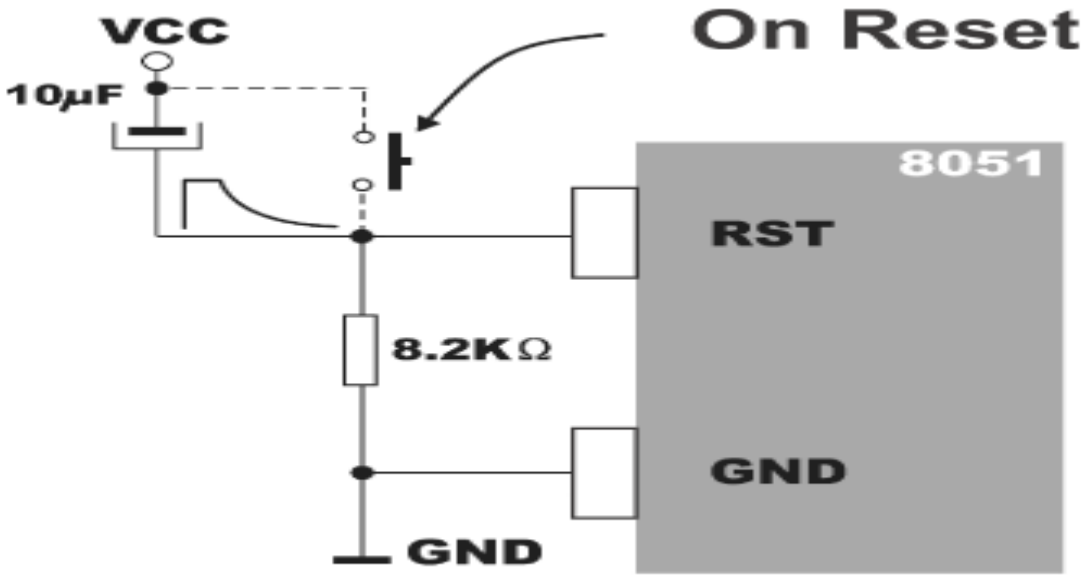
All addresses are in hexadecimal format

5. These addresses store appropriate subroutines processing interrupts. Instead of them, there are usually jump instructions specifying locations on which these subroutines reside.
6. When an interrupt routine is executed, the address of the next instruction to execute is poped from the stack to the program counter and interrupted program resumes operation from where it left off.

From the moment an interrupt is enabled, the microcontroller is on alert all the time. When an interrupt request arrives, the program execution is stopped, electronics recognizes the source and the program "jumps" to the appropriate address (see the table above). This address usually stores a jump instruction specifying the start of appropriate subroutine. Upon its execution, the program resumes operation from where it left off.

Reset

Reset occurs when the RS pin is supplied with a positive pulse in duration of at least 2 machine cycles (24 clock cycles of crystal oscillator). After that, the microcontroller generates an internal reset signal which clears all SFRs, except SBUF registers, Stack Pointer and ports (the state of the first two ports is not defined, while FF value is written to the ports configuring all their pins as inputs). Depending on surrounding and purpose of device, the RS pin is usually connected to a power-on reset push button or circuit or to both of them. Figure below illustrates one of the simplest circuit providing safe power-on reset.



Basically, everything is very simple: after turning the power on, electrical capacitor is being charged for several milliseconds through a resistor connected to the ground. The pin is driven high during this process. When the capacitor is charged, power supply voltage is already stable and the pin remains connected to the ground, thus providing normal operation of the microcontroller. Pressing the reset button causes the capacitor to be temporarily discharged and the microcontroller is reset. When released, the whole process is repeated…

Through the program- step by step...

Microcontrollers normally operate at very high speed. The use of 12 Mhz quartz crystal enables 1.000.000 instructions to be executed per second. Basically, there is no need for higher operating rate. In case it is needed, it is easy to built in a crystal for high frequency. The problem arises when it is necessary to slow down the operation of the microcontroller. For example during testing in real environment when it is necessary to execute several instructions step by step in order to check I/O pins' logic state.

Interrupt system of the 8051 microcontroller practically stops operation of the microcontroller and enables instructions to be executed one after another by pressing the button. Two interrupt features enable that:

- Interrupt request is ignored if an interrupt of the same priority level is in progress.
- Upon interrupt routine execution, a new interrupt is not executed until at least one instruction from the main program is executed.

In order to use this in practice, the following steps should be done:

1. External interrupt sensitive to the signal level should be enabled (for example INT0).
2. Three following instructions should be inserted into the program (at the 03hex. address):

```
JNB     P3.2$   ← Means: wait here until the pin P3.2 (INT0) is set to "1".
JB      P3.2$   ← Means: wait here until the pin P3.2 (INT0) is set to "0".
RETI            ← Means: go back to the main program
```

What is going on? As soon as the P3.2 pin is cleared (for example, by pressing the button), the microcontroller will stop program execution and jump to the 03hex address will be executed. This address stores a short interrupt routine consisting of 3 instructions.

The first instruction is executed until the push button is realised (logic one (1) on the P3.2 pin). The second instruction is executed until the push button is pressed again. Immediately after that, the RETI instruction is executed and the processor resumes operation of the main program. Upon execution of any program instruction, the interrupt INT0 is generated and the whole procedure is repeated (push button is still pressed). In other words, one button press - one instruction.