Embedded Systems
(Lec_ 13_ Interrupts)

Chamika Bandara
BEng(Hons)Digital Communication & Electronics ,
MSc in Information Security,
MIET(London),AMIIESL.
0714446072

An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. Whenever an interrupt occurs, the controller completes the execution of the current instruction and starts the execution of an **Interrupt Service Routine** (ISR) or **Interrupt Handler. ISR tells the processor or controller what to do when the interrupt** occurs. The interrupts can be either hardware interrupts or software interrupts.

Hardware Interrupt

A hardware interrupt is an electronic alerting signal sent to the processor from an external device, like a disk controller or an external peripheral. For example, when we press a key on the keyboard or move the mouse, they trigger hardware interrupts which cause the processor to read the keystroke or mouse position.

Software Interrupt

A software interrupt is caused either by an exceptional condition or a special instruction in the instruction set which causes an interrupt when it is executed by the processor. For example, if the processor's arithmetic logic unit runs a command to divide a number by zero, to cause a divide-by-zero exception, thus causing the computer to abandon the calculation or display anerror message. Software interrupt instructions work similar to subroutine calls.

What is Polling?

The state of continuous monitoring is known as **polling. The microcontroller keeps checking the** status of other devices; and while doing so, it does no other operation and consumes all its processing time for monitoring. This problem can be addressed by using interrupts. In the interrupt method, the controller responds only when an interruption occurs. Thus, the controller is not required to regularly monitor the status (flags, signals etc.) of interfaced and inbuilt devices.

Interrupts v/s Polling

Here is an analogy that differentiates an interrupt from polling:

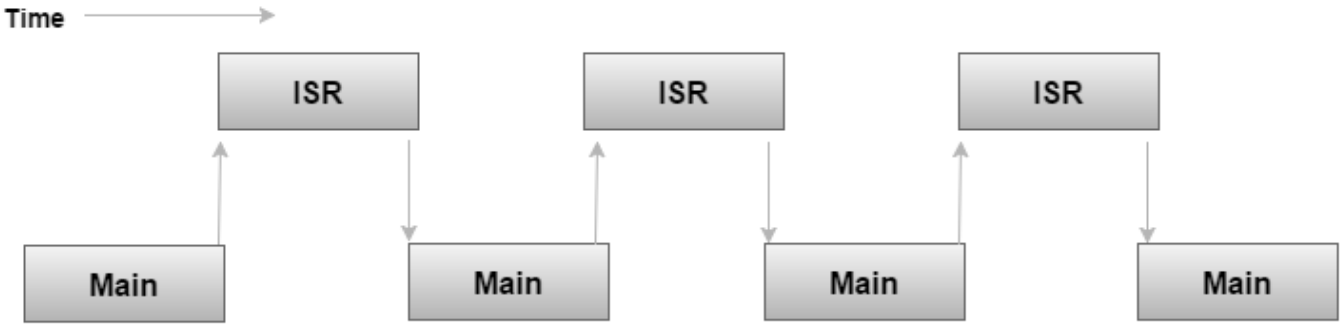
Interrupt	Polling
An interrupt is like a **shopkeeper. If one** needs a service or product, he goes to him and apprises him of his needs. In case of interrupts, when the flags or signals are received, they notify the controller that they need to be serviced.	The polling method is like a **salesperson. The** salesman goes from door to door while requesting to buy a product or service. Similarly, the controller keeps monitoring the flags or signals one by one for all devices and provides service to whichever component that needs its service

Interrupt Service Routine

For every interrupt, there must be an interrupt service routine (ISR), or **interrupt handler.** When an interrupt occurs, the microcontroller runs the interrupt service routine. For every interrupt, there is a fixed location in memory that holds the address of its interrupt service routine, ISR. The table of memory locations set aside to hold the addresses of ISRs is called as the Interrupt Vector Table

Program Execution without Interrupts

Program Execution with Interrupts



ISR : Interrupt Service Routine

Interrupt Vector Table

There are six interrupts including RESET in 8051

Interrupts	ROM Location (Hex)	Pin
Interrupts	ROM Location (HEX)	
Serial COM (RI and TI)	0023	
Timer 1 interrupts(TF1)	001B	
External HW interrupt 1 (INT1)	0013	P3.3 (13)
External HW interrupt 0 (INT0)	0003	P3.2 (12)
Timer 0 (TF0)	000B	
Reset	0000	9

1. When the reset pin is activated, the 8051 jumps to the address location 0000. This is power-up reset.

2. Two interrupts are set aside for the timers: one for timer 0 and one for timer 1. Memory locations are 000BH and 001BH respectively in the interrupt vector table.

3. Two interrupts are set aside for hardware external interrupts. Pin no. 12 and Pin no. 13 in Port 3 are for the external hardware interrupts INT0 and INT1, respectively. Memory locations are 0003H and 0013H respectively in the interrupt vector table.

4. Serial communication has a single interrupt that belongs to both receive and transmit. Memory location 0023H belongs to this interrupt.

Steps to Execute an Interrupt

When an interrupt gets active, the microcontroller goes through the following steps:

1. The microcontroller closes the currently executing instruction and saves the address of the next instruction (PC) on the stack.

2. It also saves the current status of all the interrupts internally (i.e., not on the stack).

3. It jumps to the memory location of the interrupt vector table that holds the address of the interrupts service routine.

4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it. It starts to execute the interrupt service subroutine, which is RETI (return from interrupt).

5. Upon executing the RETI instruction, the microcontroller returns to the location where it was interrupted. First, it gets the program counter (PC) address from the stack by popping the top bytes of the stack into the PC. Then, it start to execute from that address.