Introduction to Interrupts

By: Yehia M. Abu Eita

Outlines

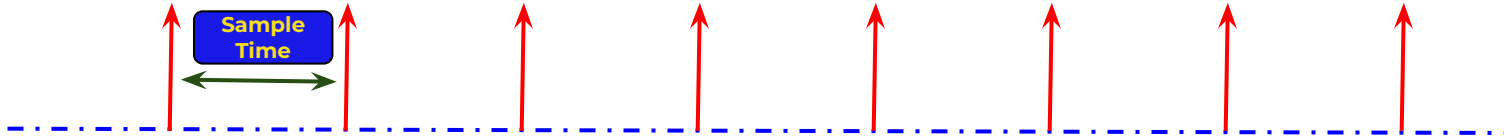
- Introduction
- Polling
- Interrupt Service Routine (ISR)
- Interrupt Vector Table (IVT)
- Interrupt Cycle
- Interrupt latency and response

Introduction

- An interrupt is a **signal** that is generated by **hardware or software** when a process or an event needs **immediate attention**.
- **Sources of Interrupts**:
  - **Hardware**:
    - It is an **electronic signal** sent from an **external device**, or from an **internal peripheral** (Timer, ADC, ...).
    - **Types**:
      - **Maskable:** These interrupts **can** be **enabled** or **disabled by software**.
      - **Non-Maskable:** These interrupts **can not be disabled**.
  - **Software**:
    - It is caused either by an **exceptional condition** or a **special instruction** in the instruction set.
    - **Types**:
      - **Exceptions**.

Polling

- It refers to actively **sampling** the **status** of an **external device** by a program as a synchronous activity.
- **Polling rate**: It is the **number of samples per second**.
- **Example**:
  - **Keyboard polling rate 125Hz.**
  - **This means that the keyboard is being scanned 125 times per second.**



Interrupt Service Routine (ISR)

- It is a **function** that **takes void** and **returns void**.
- It **must** contain **small logic**.
- It is **not called by the software**.
- It **mustn't** be **optimized** by the compiler.

```
/* ISR to handle the External Interrupt Request 0 */
void __vector_1(void) __attribute__ ((signal,used));
```

```
/* ISR to handle the External Interrupt Request 0 */
void __vector_1(void)
{
        //write your ISR here
}
```

```
/* Vectors */

/* External Interrupt Request 0 */
__vector_1
/* External Interrupt Request 1 */
__vector_2
.....
.....
```

Interrupt Service Routine (ISR)

- To implement the interrupt library.

```
/* Vectors in interrupts.h*/

/* External Interrupt Request 0 */
#define EXT_INT_0 __vector_1
/* External Interrupt Request 1 */
#define EXT_INT_1 __vector_2
/* External Interrupt Request 2 */
#define EXT_INT_2 __vector_3
/* Macro defines the ISR */
#define ISR(INT_VECT)void INT_VECT(void) __attribute__ ((signal,used));\
void INT_VECT(void)
```

```
/* dio.c*/

/* functions */


/* Implement ISR for external
interrupt 0 */
ISR(EXT_INT_0)
{
      /* Write the ISR here */
}
```

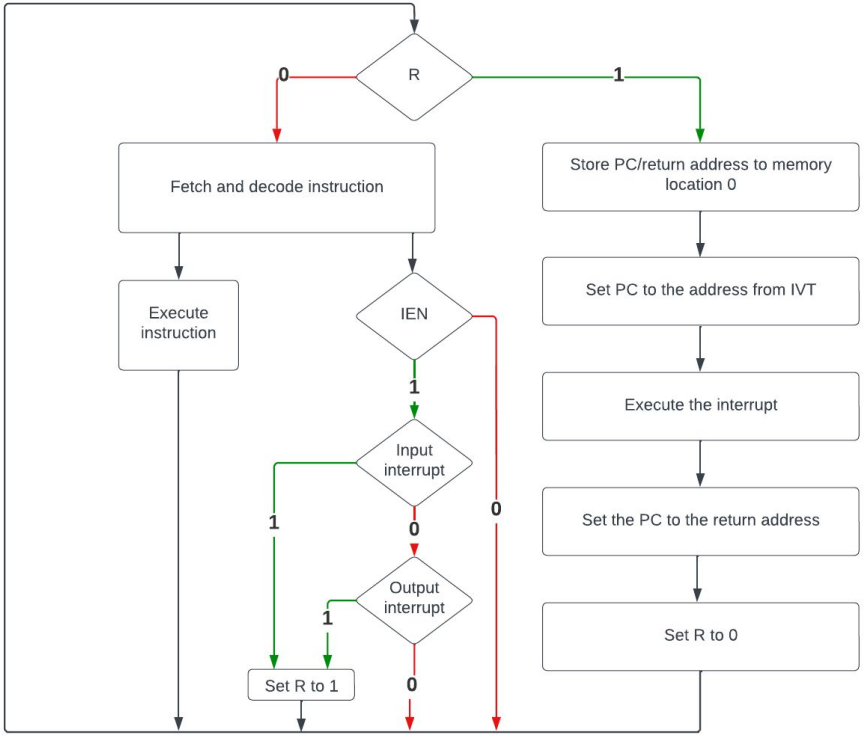
Interrupt Vector Table (IVT)

- It **stores the addresses of ISR**, **if implemented**.
- It is **stored** in the **lowest address** into the **Flash** memory.
- **Linker Script** or **startup code**.

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	INT2	External Interrupt Request 2
5	$008	TIMER2 COMP	Timer/Counter2 Compare Match
6	$00A	TIMER2 OVF	Timer/Counter2 Overflow
7	$00C	TIMER1 CAPT	Timer/Counter1 Capture Event
8	$00E	TIMER1 COMPA	Timer/Counter1 Compare Match A
9	$010	TIMER1 COMPB	Timer/Counter1 Compare Match B
10	$012	TIMER1 OVF	Timer/Counter1 Overflow
11	$014	TIMER0 COMP	Timer/Counter0 Compare Match
12	$016	TIMER0 OVF	Timer/Counter0 Overflow
13	$018	SPI, STC	Serial Transfer Complete
14	$01A	USART, RXC	USART, Rx Complete
15	$01C	USART, UDRE	USART Data Register Empty
16	$01E	USART, TXC	USART, Tx Complete
17	$020	ADC	ADC Conversion Complete
18	$022	EE_RDY	EEPROM Ready
19	$024	ANA_COMP	Analog Comparator
20	$026	TWI	Two-wire Serial Interface
21	$028	SPM_RDY	Store Program Memory Ready

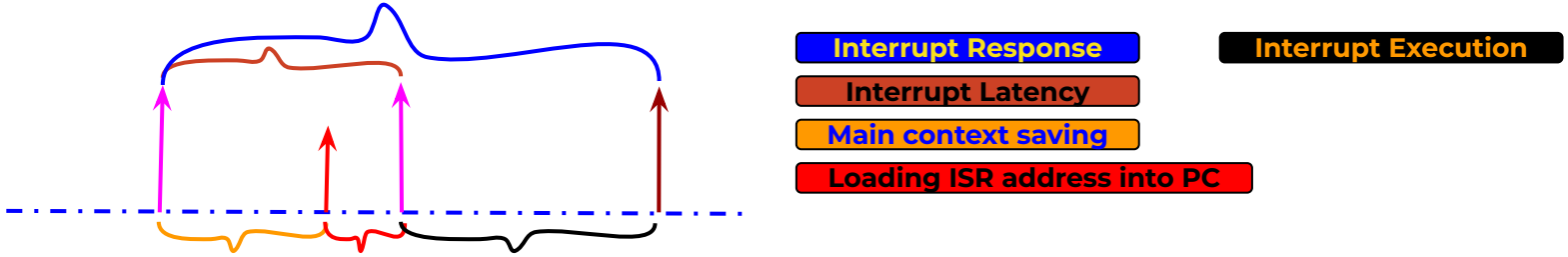
Interrupt Cycle

- **R**: is a flag refers to interrupt event.
- **IEN**: Global Interrupt Enabled.



Interrupt latency and response

- **Interrupt latency**:
  - It refers to the delay from the **start of the interrupt request to the start of the interrupt** execution.
- **Interrupt response**:
  - It refers to the time taken from the **start of the interrupt request to the end of the interrupt** execution.



Summary

- Now you are familiar with interrupts, its sources and types
- Remember to build your interrupts.h file so you can use the interrupts
- Remember that ISR is a function that has no input and no output and can’t be called in your code