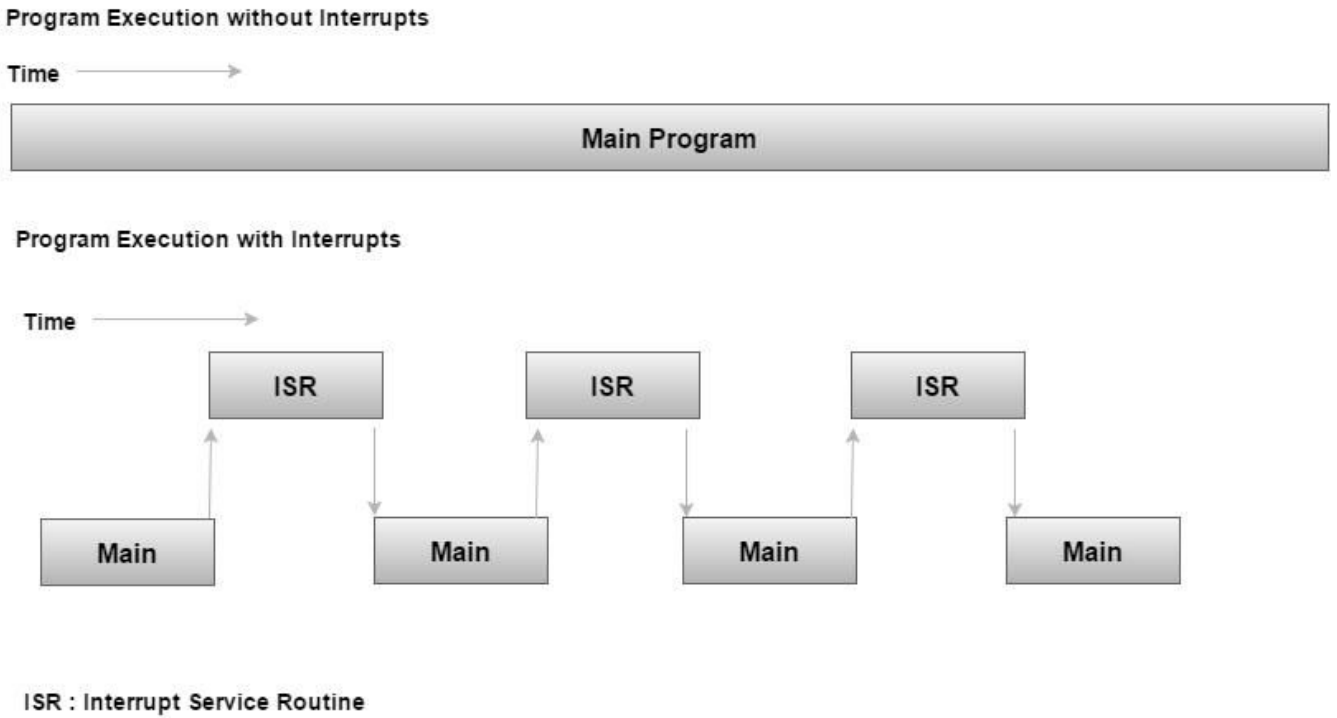
Bài 3: Interrupt - Timer

Võ Thành Danh

1.1 Định nghĩa ngắt

Ngắt là 1 sự kiện khẩn cấp xảy ra **trong hay ngoài** vi điều khiển. Nó yêu cầu MCU phải dừng chương trình chính và thực thi **chương trình ngắt (trình phục vụ ngắt)**.



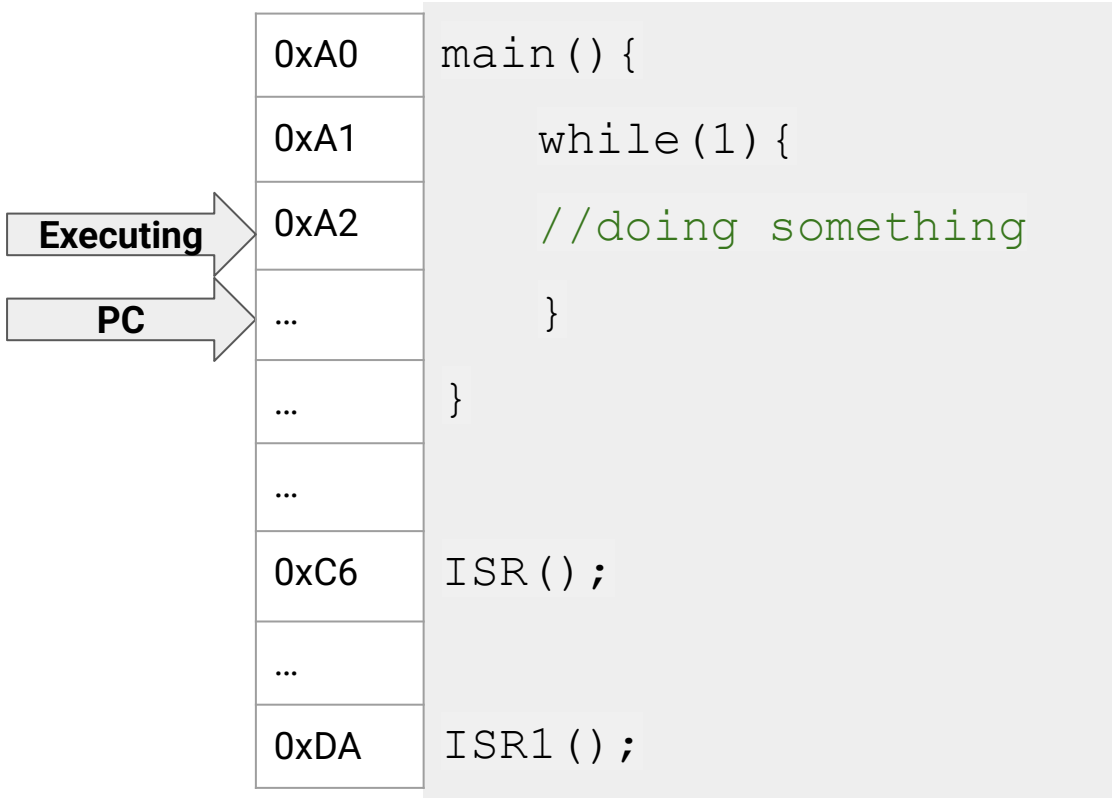
1.2 Các ngắt thông dụng

- Mỗi ngắt sẽ có 1 trình phục vụ ngắt riêng
- **Trình phục vụ ngắt (Interrupt Service Routine - ISR)** là một đoạn chương trình được thực hiện khi ngắt xảy ra.
- Địa chỉ trong bộ nhớ của ISR được gọi là **vector ngắt**

Ngắt	Cờ ngắt	Vector ngắt	Độ ưu tiên ngắt
Reset	-	0000h	-
Ngắt ngoài	IE0	0003h	Lập trình được
Timer1	TF1	001Bh	Lập trình được
Ngắt truyền thông			

1.2 Các ngắt thông dụng

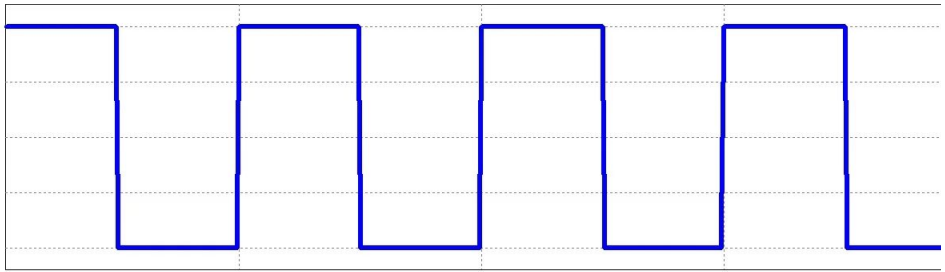
PC (Program counter) là thanh ghi luôn chỉ đến lệnh tiếp theo trong chương trình.`

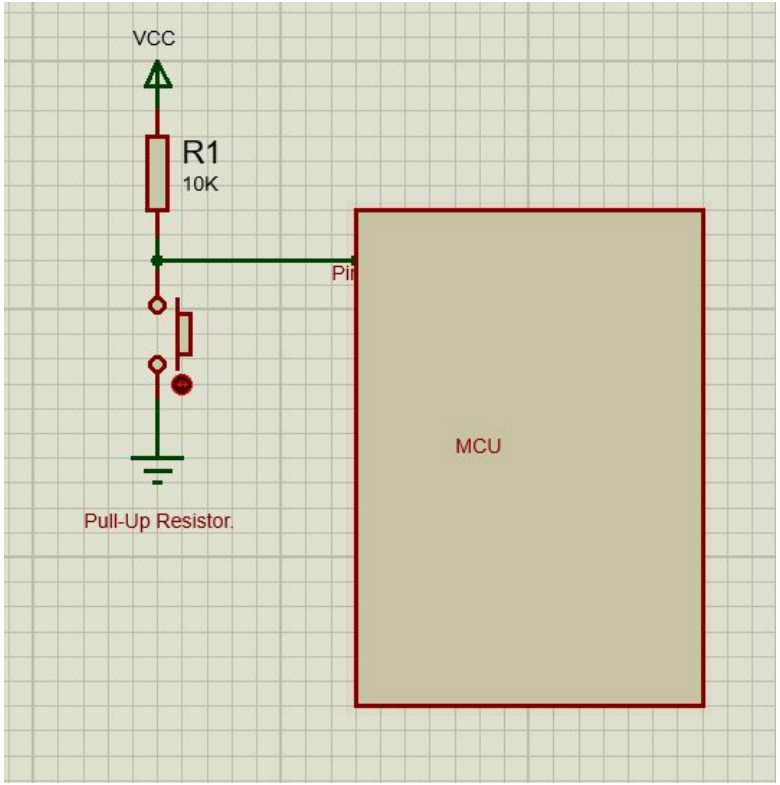


1.2.1 Ngắt ngoài

Xảy ra khi có thay đổi điện áp trên các chân GPIO được cấu hình làm **ngõ vào ngắt.** Có 4 dạng:

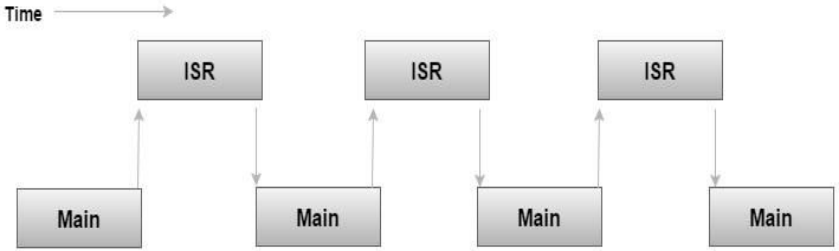
- **LOW**: kích hoạt ngắt liên tục khi chân ở mức thấp.
- **HIGH**: Kích hoạt liên tục khi chân ở mức cao.
- **RISING**: Kích hoạt khi trạng thái trên chân chuyển từ thấp lên cao.
- **FALLING**: Kích hoạt khi trạng thái trên chân chuyển từ cao xuống thấp.





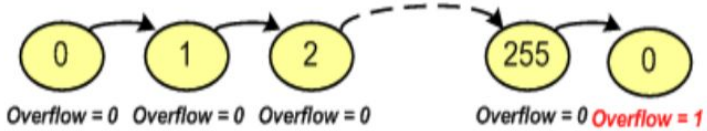
1.2.2 Ngắt timer

Xảy ra khi giá trị trong thanh ghi đếm của **timer bị tràn**. Sau mỗi lần tràn, cần phải reset giá trị thanh ghi để có thể tạo ngắt tiếp theo.

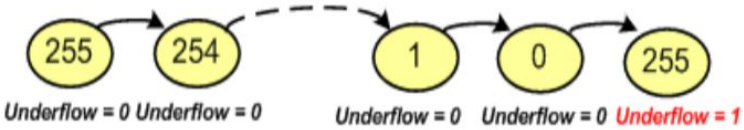


ISR : Interrupt Service Routine

- Up-counter

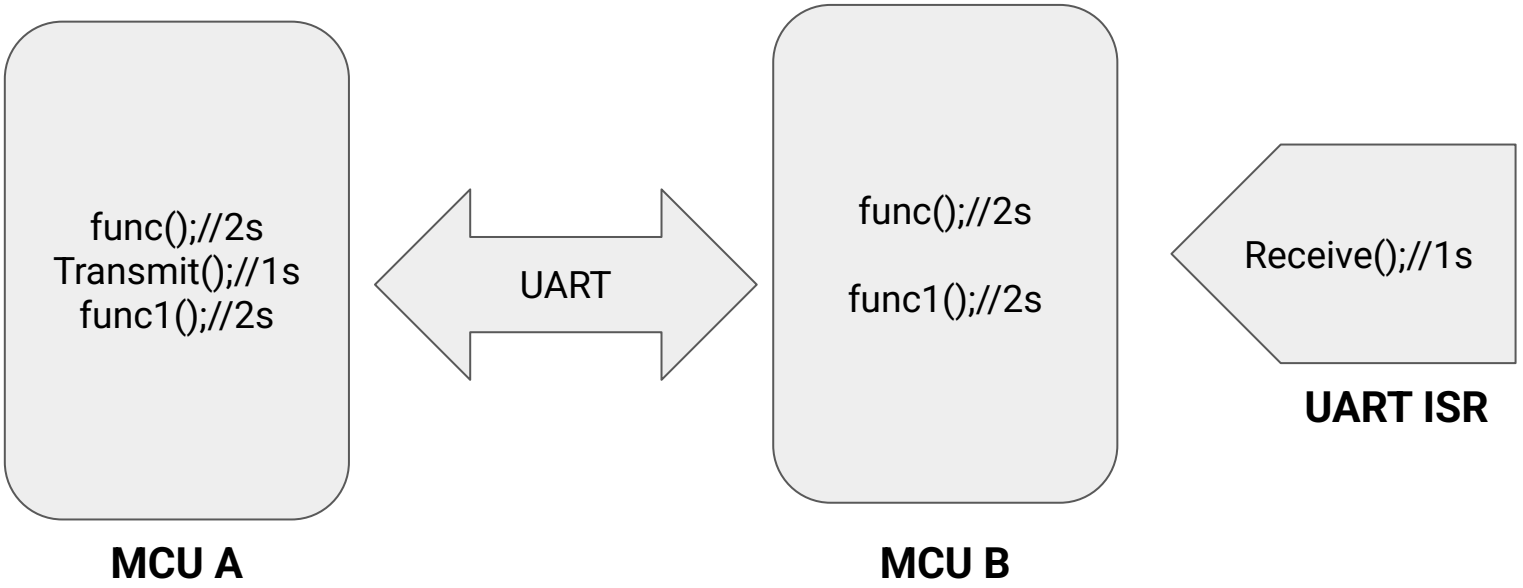


- Down counter



1.2.3 Ngắt truyền thông

Xảy ra khi có sự kiện truyền/nhận dữ liệu giữa MCU và các thiết bị khác, thường sử dụng cho các giao thức như UART, SPI, I2C để đảm bảo việc truyền/nhận được chính xác.



1.3 Độ ưu tiên ngắt

- Các ngắt có độ ưu tiên khác nhau, quyết định ngắt nào được thực thi khi nhiều ngắt xảy ra đồng thời.
- Trên STM32, ngắt có số ưu tiên càng thấp thì có quyền càng cao.
- Độ ưu tiên ngắt có thể lập trình được
- **Stack Pointer** là thanh ghi trỏ tới đỉnh của vùng stack chứa các địa chỉ trả về của các hàm

Stack Pointer

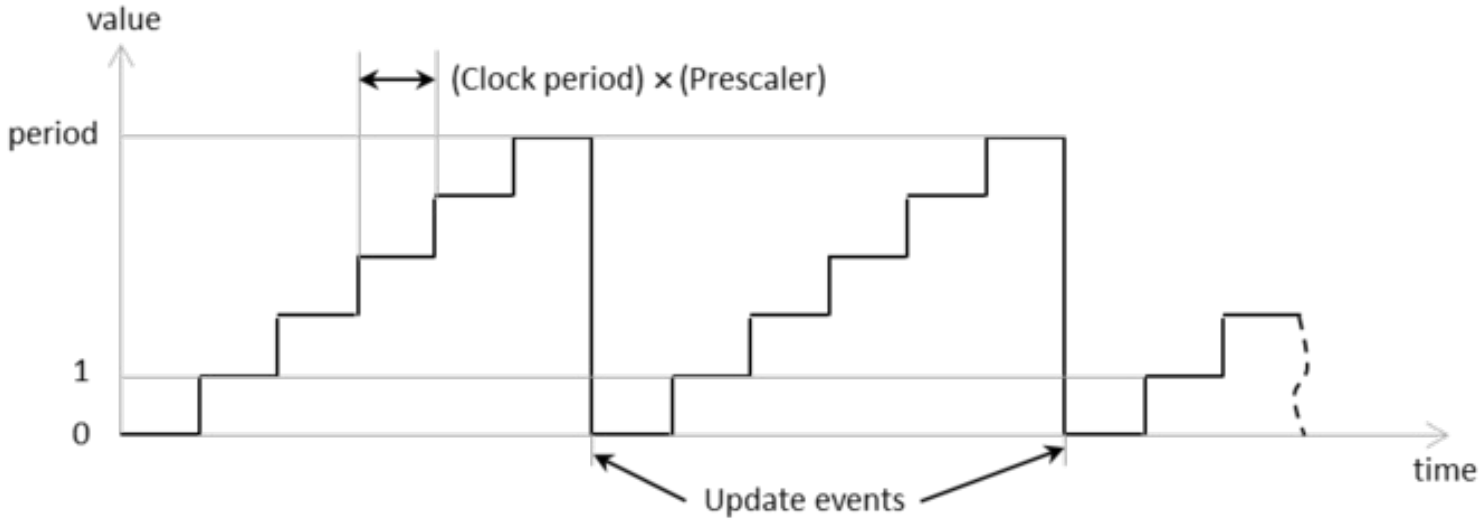
0x01 ... 0xA1	main(){}	
0xB2 ... 0xB9	Timer_ISR();	Ưu tiên cao 0
0xD4 ... 0xE2	UART_ISR();	Ưu tiên thấp 1

2. Timer

Timer là 1 mạch digital logic có vai trò đếm mỗi chu kỳ clock (đếm lên hoặc đếm xuống).

Timer còn có thể hoạt động ở chế độ nhận xung clock từ các tín hiệu ngoài. Ngoài ra còn các chế độ khác như PWM, định thời …vv.

STM32F103 có 7 timer



2. Timer

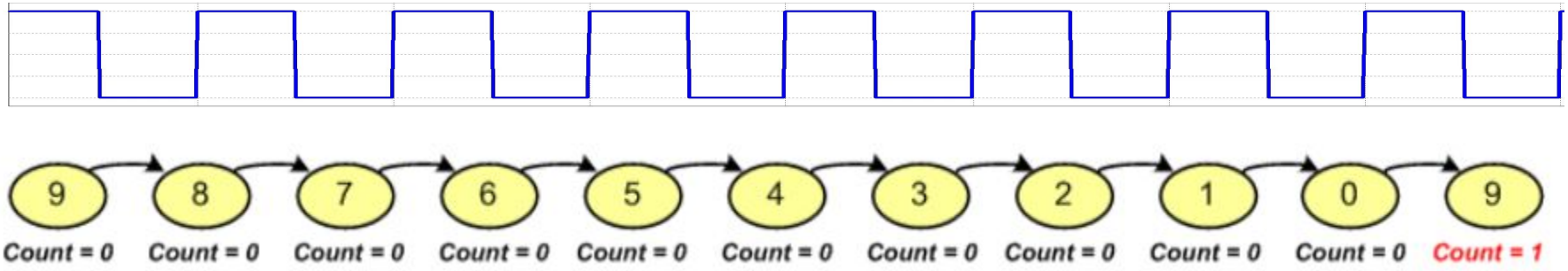
```
typedef struct
{
  uint16_t TIM_Prescaler;          /*!< Specifies the prescaler value used to divide the TIM clock.
                                       This parameter can be a number between 0x0000 and 0xFFFF */

  uint16_t TIM_CounterMode;        /*!< Specifies the counter mode.
                                       This parameter can be a value of @ref TIM_Counter_Mode */

  uint16_t TIM_Period;             /*!< Specifies the period value to be loaded into the active
                                       Auto-Reload Register at the next update event.
                                       This parameter must be a number between 0x0000 and 0xFFFF.  */

  uint16_t TIM_ClockDivision;      /*!< Specifies the clock division.
                                      This parameter can be a value of @ref TIM_Clock_Division_CKD */

} TIM_TimeBaseInitTypeDef;
```



2. Timer

```
void TIM_SetCounter(TIM_TypeDef* TIMx, uint16_t Counter);
// Đặt giá trị ban đầu cho timer
uint16_t TIM_GetCounter(TIM_TypeDef* TIMx);
// Lấy giá trị đếm hiện tại của timer
```

```
// Delay function
void delay_ms(uint8_t timedelay)
{
    TIM_SetCounter(TIM2,0);
    while(TIM_GetCounter(TIM2) < timedelay * 10){}
}
```