# Ngắt trong 8051

Ø Phân biệt cơ chế ngắt với hỏi vòng
Ø Nắm rõ các loại ngắt trong 8051
- Ngắt timer/counter
- Ngắt ngoài
- Ngắt truyền thông nối tiếp

Ø Lập trình các ngắt
- Trình phục vụ ngắt là gì?
- Cho phép ngắt và cấm ngắt
- Thiết lập mức ưu tiên của các ngắt

## Giới thiệu

**Ngắt** (**Interrupt**) - như tên của nó, là một số sự kiện khẩn cấp bên trong hoặc bên ngoài bộ vi điều khiển xảy ra, buộc vi điều khiển tạm dừng thực hiện chương trình hiện tại, phục vụ ngay lập tức nhiệm vụ mà ngắt yêu cầu – nhiệm vụ này gọi là trình phục vụ ngắt (**ISR:** Interrupt Service Routine).

Trong bài này ta tìm hiểu khái niệm ngắt và lập trình các ngắt trong bộ vi điều khiển 8051.

## 1. Các ngắt của 8051

### 1.1 Phân biệt cơ chế ngắt với hỏi vòng

**Lấy ví dụ:** Bộ vi điều khiển đóng vai trò như một vị bác sĩ, các thiết bị kiểm soát bởi vi điều khiển được coi như các bệnh nhân cần được bác sĩ phục vụ.

Bình thường, vị bác sĩ sẽ hỏi thăm lần lượt từng bệnh nhân, đến lượt bệnh nhân nào được hỏi thăm nếu có bệnh thì sẽ được bác sĩ phục vụ, xong lại đến lượt bệnh nhân khác, và tiếp tục đến hết. Điều này tương đương với phương pháp **thăm dò - hỏi vòng**(**Polling**) trong vi điều khiển.

Cứ như thế, nếu chúng ta có 10 bệnh nhân, thì bệnh nhân thứ 10 dù muốn hay không cũng phải xếp hàng chờ đợi 09 bệnh nhân trước đó. Giả sử trường hợp bệnh nhân thứ 10 cần cấp cứu thì sao? Anh ta sẽ gặp nguy cấp trước khi đến lượt hỏi thăm của bác sĩ mất! L Nhưng, nếu anh ta sử dụng phương pháp "ngắt" thì mọi chuyện sẽ ổn ngay. Lúc đó vị bác sĩ sẽ ngừng mọi công việc hiện tại của mình, và tiến hành phục vụ trường hợp khẩn cấp này ngay lập tức, xong việc bác sĩ lại trở về tiếp tục công việc đang dở. Điều này tương đương với phương pháp **ngắt** (**Interrupts**) trong vi điều khiển.

Trở lại với bộ vi điều khiển của chúng ta: 1 bộ vi điều khiển có thể phục vụ cho nhiều thiết bị, có 2 cách để thực hiện điều này đó là sử dụng các **ngắt** (Interrupts) và**thăm dò** (polling):

Ø **Trong phương pháp sử dụng ngắt**: mỗi khi có một thiết bị bất kỳ cần được phục vụ thì nó báo cho bộ vi điều khiển bằng cách gửi một tín hiệu **ngắt**. Khi nhận được tín hiệu **ngắt** thì bộ vi điều khiển ngừng tất cả những gì nó đang thực hiện để chuyển sang phục vụ thiết bị gọi ngắt. Chương trình ngắt được gọi là trình phục vụ ngắt **ISR**(Interrupt Service Routine) hay còn gọi là trình quản lý ngắt (Interrupt handler). Sau khi phục vụ ngắt xong, bộ vi xử lý lại quay trở lại điểm bị ngắt trước đó và tiếp tục thực hiện công việc.

Ø **Trong phương pháp thăm dò**: bộ vi điều khiển kiểm tra liên tục tình trạng của tất cả các thiết bị, nếu thiết bị nào có yêu cầu thì nó dừng lại phục vụ thiết bị đó. Sau đó nó tiếp tục kiểm tra tình trạng của thiết bị kế tiếp cho đến hết. Phương pháp thăm dò rất đơn giản, nhưng nó lại rất lãng phí thời gian để kiểm tra các thiết bị **kể cả khi thiết bị đó không cần phục vụ**. Trong trường hợp có quá nhiều thiết bị thì phương án thăm dò tỏ ra không hiệu quả, gây ra chậm trễ cho các thiết bị cần phục vụ.

Sơ đồ trang web

**Điểm mạnh của phương pháp ngắt là:**

Ø Bộ vi điều khiển có thể phục vụ được rất **nhiều thiết bị** (tất nhiên là không tại cùng một thời điểm). Mỗi thiết bị có thể nhận được sự chú ý của bộ vi điều khiển dựa trên**mức ưu tiên** được gán cho nó. Đối với phương pháp thăm dò thì không thể gán mức ưu tiên cho các thiết bị vì nó kiểm tra tất cả mọi thiết bị theo kiểu hỏi vòng.
Ø Quan trọng hơn, trong phương pháp ngắt thì bộ vi điều khiển còn có thể **che** (làm lơ) một yêu cầu phục vụ của thiết bị. Điều này lại một lần nữa không thể thực hiện được trong phương pháp thăm dò.
Ø Lý do quan trọng nhất mà phương pháp ngắt được ưu chuộng là vì nó không lãng phí thời gian cho các thiết bị không cần phục vụ. Còn phương pháp thăm dò làm lãng phí thời gian của bộ vi điều khiển bằng cách hỏi dò từng thiết bị kể cả khi chúng không cần phục vụ.

Ví dụ trong các bộ định thời được bàn đến ở các bài trước ta đã dùng một vòng lặp kiểm tra và đợi cho đến khi bộ định thời quay trở về 0. Trong ví dụ đó, nếu sử dụng ngắt thì ta không cần bận tâm đến việc kiểm tra cờ bộ định thời, do vậy không lãng phí thời gian để chờ đợi, trong khi đó ta có thể làm việc khác có ích hơn.

### 1.2 Sáu ngắt trong 8051

Thực tế chỉ có **5** ngắt dành cho người dùng trong 8051 nhưng các nhà sản xuất nói rằng có 6 ngắt vì họ tính cả lệnh RESET. Sáu ngắt của 8051 được phân bố như sau:

1. **RESET**: Khi chân RESET được kích hoạt từ 8051, bộ đếm chương trình nhảy về địa chỉ **0000H**. Đây là địa chỉ bật lại nguồn.
2. **2 ngắt dành cho các bộ định thời**: 1 cho **Timer0** và 1 cho **Timer1**. Địa chỉ tương ứng của các ngắt này là **000BH** và **001BH**.
3. **2 ngắt dành cho các ngắt phần cứng bên ngoài:** chân 12 (P3.2) và 13 (P3.3) của cổng P3 là các ngắt phần cứng bên ngoài **INT0** và **INT1** tương ứng. Địa chỉ tương ứng của các ngắt ngoài này là **0003H** và **0013H**.
4. **Truyền thông nối tiếp:** có 1 ngắt chung cho cả nhận và truyền dữ liệu nối tiếp. Địa chỉ của ngắt này trong bảng vector ngắt là **0023H**.

### 1.3 Trình phục vụ ngắt

Đối với mỗi ngắt thì phải có một **trình phục vụ ngắt** (**ISR**) hay trình quản lý ngắt để đưa ra nhiệm vụ cho bộ vi điều khiển khi được gọi ngắt. Khi một ngắt được gọi thì bộ vi điều khiển sẽ chạy trình phục vụ ngắt. Đối với mỗi ngắt thì có một vị trí cố định trong bộ nhớ để giữ địa chỉ **ISR** của nó. Nhóm vị trí bộ nhớ được dành riêng để lưu giữ địa chỉ của các **ISR** được gọi là **bảng vector ngắt**. Xem **Hình 1**.

| Ngắt | Cờ ngắt | Địa chỉ trình phục vụ ngắt | Số thứ tự ngắt |
|---|---|---|---|
| Reset | - | 0000h | - |
| Ngắt ngoài 0 | IE0 | 0003h | 0 |
| Timer 0 | TF0 | 000Bh | 1 |
| Ngắt ngoài 1 | IE1 | 0013h | 2 |
| Timer 1 | TF1 | 001Bh | 3 |
| Ngắt truyền thông | RI/TI | 0023h | 4 |



**Hình 1:** Bảng vector ngắt của 8051.

Trong lập trình **C** trên **Keil c** cho 8051, chúng ta khai báo **trình phục vụ ngắt**theo cấu trúc sau:

**Void** Name (**void**) **interrupt** X ///( **X**: là **số thứ tự của ngắt** )

```
{
      // chương trình phục vụ ngắt
}
```

Khi đó địa chỉ ngắt sẽ được tự động tính bằng:

Interrupt Address = (X * 8) + 3

### 1.4 Quy trình khi thực hiện một ngắt

Khi kích hoạt một ngắt bộ vi điều khiển thực hiện các bước sau:

Ø Nó hoàn thành nốt lệnh đang thực hiện và lưu địa chỉ của **lệnh kế tiếp** vào ngăn xếp.
Ø Nó cũng **lưu tình trạng hiện tại** của tất cả các ngắt.
Ø Nó nhảy đến một vị trí cố định trong bộ nhớ được gọi là **bảng vector ngắt**, nơi lưu giữ địa chỉ của một **trình phục vụ ngắt**.
Ø Bộ vi điều khiển nhận địa chỉ **ISR** từ bảng vector ngắt và nhảy tới đó. Nó bắt đầu thực hiện trình phục vụ ngắt cho đến lệnh cuối cùng của **ISR** và trở về chương trình chính từ ngắt.
Ø Khi bộ vi điều khiển quay trở về nơi nó đã bị ngắt. Trước hết nó nhận địa chỉ của bộ đếm chương trình PC từ ngăn xếp bằng cách kéo 02 byte trên đỉnh của ngăn xếp vào PC. Sau đó bắt đầu thực hiện tiếp các lệnh từ địa chỉ đó.

### 1.5 Các bước cho phép và cấm ngắt

**Khi bật lại nguồn thì tất cả mọi ngắt đều bị cấm** (bị che), có nghĩa là không có ngắt nào được bộ vi điều khiển đáp ứng trừ khi chúng được kích hoạt.

Các ngắt phải được kích hoạt bằng phần mềm để bộ vi điều khiển đáp ứng chúng. Có một thanh ghi được gọi là thanh ghi cho phép ngắt **IE** (Interrupt Enable) – ở địa chỉ A8H chịu trách nhiệm về việc cho phép và cấm các ngắt. **Hình 2** trình bày chi tiết về thanh ghi **IE**.

| Bit | Tên | Địa chỉ | Chức năng |
|---|---|---|---|
| 7 | **EA** | AFh | Cho phép/cấm hoạt động của cả thanh ghi |
| 6 | - | AEh | Chưa sử dụng |
| 5 | - | ADh | Chưa sử dụng |
| 4 | **ES** | ACh | Cho phép ngắt cổng truyền thông nối tiếp |
| 3 | **ET1** | ABh | Cho phép ngắt Timer 1 |
| 2 | **EX1** | AAh | Cho phép ngắt ngoài 1 |
| 1 | **ET0** | A9h | Cho phép ngắt Timer 0 |
| 0 | **EX0** | A8h | Cho phép ngắt ngoài 0 |

**Hình 2:** Thanh ghi cho phép ngắt **IE**.

Để cho phép một ngắt ta phải thực hiện các bước sau:

Ø Nếu **EA = 0** thì **không** có ngắt nào được đáp ứng cho dù bit tương ứng của nó trong **IE**có giá trị cao. **Bit D7 - EA** của thanh ghi **IE** phải được bật lên cao để cho phép các bit còn lại của thanh ghi hoạt động được.
Ø Nếu **EA = 1** thì tất cả mọi ngắt đều được phép và sẽ được đáp ứng nếu các bit tương ứng của chúng trong **IE** có **mức cao**.

Để hiểu rõ điểm quan trọng này ta hãy xét **ví dụ 1.**

**Ví dụ 1:**
Hãy lập trình cho 8051:
a) cho phép **ngắt nối tiếp**, **ngắt Timer0** và **ngắt phần cứng ngoài 1** (EX1)

b) cấm **ngắt Timer0**
c) sau đó trình bày cách **cấm tất cả mọi ngắt** chỉ bằng một lệnh duy nhất.
**Lời giải:**

```
#include<at89x51.h>
main()
{
        //a)
        IE=0x96;      //1001 0110: lệnh này tương đương với 4 lệnh phía dưới

        EA=1;         //Cho phép sử dụng ngắt
        ES=1;         //Cho phép ngắt cổng nối tiếp
        ET0=1;        //Cho phép ngắt timer0
        EX1=1;        //Cho phép ngắt ngoài 1

        //b)
        ET0=0;        //Cấm ngắt timer0

        //c)
       EA=0;          //Cấm tất cả các ngắt

        while(1)
        {
                //Chương trình chính
                //…
        }
}
```

**2. Lập trình các ngắt bộ định thời**

Trong các bài trước ta đã biết cách sử dụng các bộ định thời **Timer0** và **Timer1**bằng phương pháp **thăm dò**. Trong phần này ta sẽ sử dụng các **ngắt** để lập trình cho các bộ định thời của 8051.

**2.1 Cờ quay về 0 của bộ định thời và ngắt**

Chúng ta đã biết rằng cờ bộ định thời **TF** được bật lên cao khi bộ định thời đạt giá trị cực đại và quay về **0** (Roll - over). Trong các bài trước chúng ta cũng chỉ ra cách kiểm tra cờ **TF** bằng một vòng lặp. Trong khi thăm dò cờ **TF** thì ta phải đợi cho đến khi cờ **TF**được bật lên. Vấn đề với phương pháp này là bộ vi điều khiển bị trói buộc trong khi chờ cờ **TF** được bật và không thể làm được bất kỳ việc gì khác.

Sử dụng các **ngắt** sẽ giải quyết được vấn đề này và tránh được sự trói buộc bộ vi điều khiển. Nếu bộ ngắt định thời trong thanh ghi **IE** được phép thì mỗi khi nó quay trở về 0 bộ vi điều khiển sẽ bị ngắt, bất chấp nó đang thực hiện việc gì và nhảy tới bảng vector ngắt để phục vụ **ISR**. Bằng cách này thì bộ vi điều khiển có thể làm những công việc khác cho đến khi nó được thông báo rằng bộ định thời đã quay về 0. Xem **hình 3** và **ví dụ 2**.


**Hình 3:** Ngắt bộ định thời **TF0** và **TF1**.

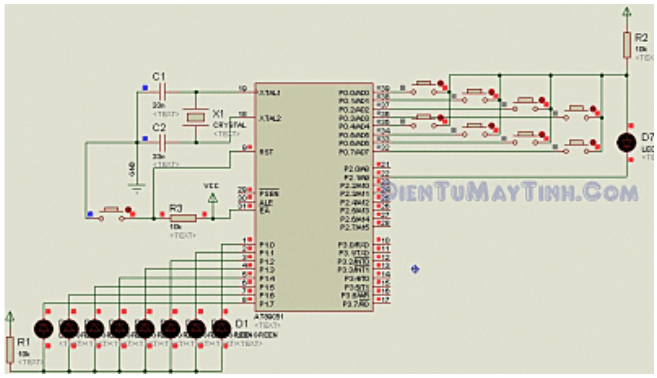
**Ví dụ 2:**

Hãy viết chương trình nhận liên tục dữ liệu 8 Bit ở cổng P0 và gửi nó đến cổng P1 trong khi nó cùng lúc tạo ra một sóng vuông chu kỳ 200ms trên chân P2.1. Hãy sử dụng bộ**Timer0** để tạo ra sóng vuông, tần số của 8051 là XTAL = 11.0592MHz.
**Lời giải:**

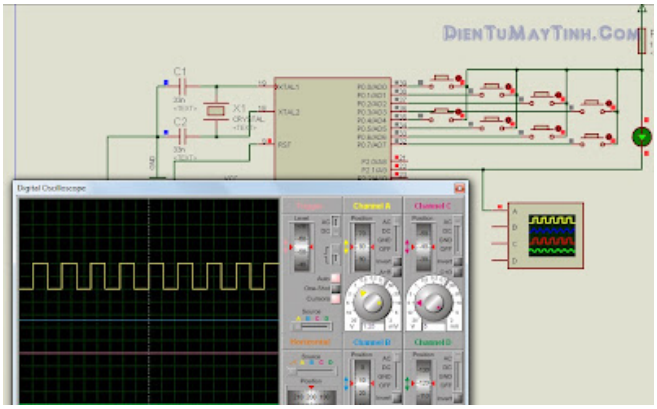
Chu kỳ 200ms, vậy nửa chu kỳ là 100ms.
Ta có: **100ms/1,085ms=92**.
Suy ra giá trị cần nạp cho timer0 là: **-92** <=> **A4H**. Ta sử dụng timer0 8 bit.

```
#include<at89x51.h>                    //khai báo thu viện cho VĐK 89x51
main()
{
        TMOD=0x02;          //chọn timer0, chế độ 2, 8Bit tự nạp lại
        TL0=0xA4;           //nạp giá trị cho TL0
        TH0=0xA4;           //nạp giá trị cho TH0
        TR0=1;                //khởi động timer0
        IE=0x82;              //cho phép ngắt timer0

        while(1)              //vòng lặp vô hạn
        {
                P1=~P0;       //Cập nhật giá trị cho cổng P1 từ P0.
        }
}
void songvuong(void) interrupt 1   //Khai báo trình phục vụ ngắt cho timer0
{
        TR0=0;                //Ngừng timer0
        P2_1=~P2_1;                 //Đảo trạng thái chân P2_1.
        TR0=1;                //Khởi động timer0
                               //Không cần xóa cờ TF0, 8051 tự động xóa.
}
```




**Hình 4:** Mô phỏng trên proteus: cập nhật liên tục cổng P1 từ P0, trong khi tạo xung ở chân P2.1




**Hình 5:** Sóng vuông hiển thị trên Oscilloscope

Hãy để ý những điểm dưới đây của chương trình trong **ví dụ 2**:

1. Chúng ta cho phép ngắt bộ **Timer0** với lệnh IE=0x82; trong chương trình chínhmain().

2. Trong khi dữ liệu ở cổng **P0** được nhận vào và chuyển liên tục sang cổng **P1** thì mỗi khi bộ **Timer0** trở về 0, cờ **TF0** được bật lên và bộ vi điều khiển thoát ra khỏi hàmmain() và đi đến địa chỉ **000BH** để thực hiện **ISR** gắn liền với bộ **Timer0**.
3. Trong trình phục vụ ngắt **ISR** của **Timer0** ta thấy rằng không cần đến lệnh xóa cờ **TF0**của **timer0**. Lý do này là vì **8051 đã tự xoá cờ TF0 ngay khi thoát khỏi ISR.**

**Ví dụ 3:**

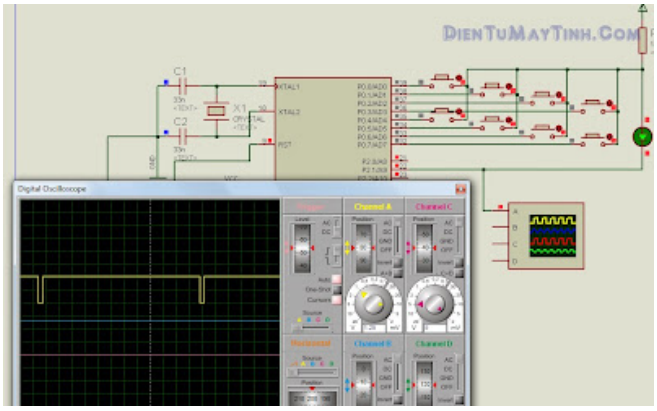
Hãy viết lại chương trình ở **ví dụ 2** để tạo sóng vuông với mức cao kéo dài 1085ms và mức thấp dài 15ms với giả thiết tần số XTAL = 11.0592MHz. Hãy sử dụng bộ định thời**Timer1**.

**Lời giải:**

Vì 1085ms/1.085ms=1000 nên ta cần sử dụng chế độ 1 của bộ định thời Timer1.
Các giá trị cần nạp cho timer1 là:

1085/1.085=1000 , **-1000óFC18H**
15/1.085=14 , **-14óFFF2H**

```
#include<at89x51.h>
bit a=0;
main()
{
        TMOD=0x10;        //chọn timer1, chế độ 1, 16Bit
        TL1=0x18;         //nạp giá trị cho TL1
        TH1=0xFC;         //nạp giá trị cho TH1
        TR1=1;            //khoi dong timer1
        IE=0x88;          //cho phép ngat timer1
        while(1)          //vòng lặp vô hạn
        {
                P1=~P0;   //Cập nhật cổng P1
        }
}
void songvuong(void) interrupt 3      //Khai báo trình phục vụ ngắt timer1
{
        TR1=0;                        //Dừng timer1
        if(a==0)                      //Nếu Xung vuông đang ở mức thấp
        {
                P2_1=1;               //Bật xung vuông lên cao
                a=1;                  //Đặt lại bit kiểm tra
                TL1=0x18;             //Nạp lại TL1: Ứng với mức trễ phần cao
                TH1=0xFC;             //Nạp lại TH1
        }
        Else                          //Nếu Xung vuông đang ở mức cao
        {
                P2_1=0;               //Lật xung xuống thấp
                a=0;                  //Đặt lại bit kiểm tra
                TL1=0xF2;             //Nạp lại TL1: Ứng với mức trễ phần thấp
TH1=0xFF;             //Nạp lại TH1
        }
        TR1=1;                        //Khởi động lại timer1
                                      //Không cần xóa cờ TF1, 8051 tự động xóa
}
```



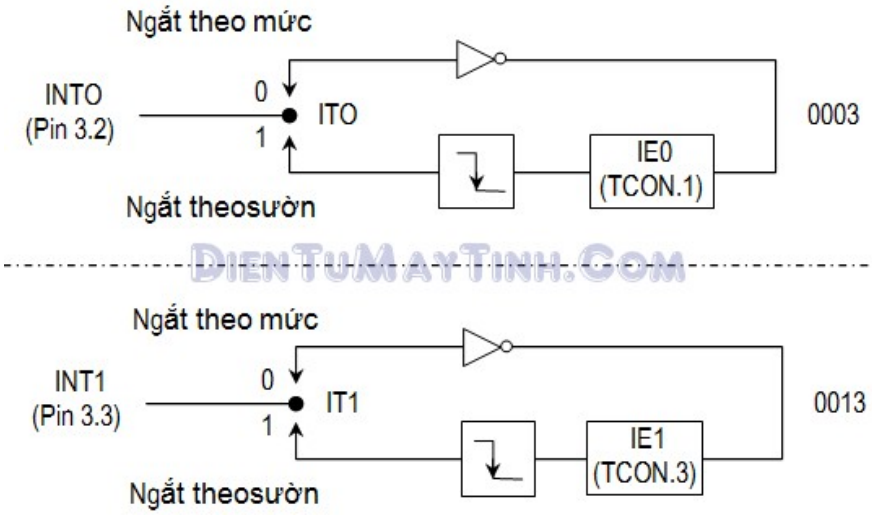

**Hình 6:** Sóng vuông hiển thị trên Oscilloscope

**Lưu ý:** Các xung được tạo ra ở các ví dụ trên không thật sự chính xác, vì chưa tính đến hao phí của các lệnh cài đặt.

### 3 Lập trình các ngắt phần cứng bên ngoài

Bộ vi điều khiển 8051 có 2 ngắt phần cứng bên ngoài ở chân 12 (**P3.2**) và chân 13 (**P3.3**) gọi là ngắt **INT0** và **INT1**.

Như đã nói ở trên thì chúng được phép và bị cấm bằng việc sử dụng thanh ghi **IE**. Nhưng cấu hình cho ngắt ngoài có phần phức tạp hơn.Có hai mức kích hoạt cho các ngắt phần cứng ngoài: **Ngắt theo mức** và **ngắt theo sườn**.




**Hình 7:** Ngắt ngoài **INT0** và **INT1**

Dưới đây là mô tả hoạt động của mỗi loại.

### 3.1 Ngắt theo mức

Ở chế độ ngắt theo mức thì các chân **INT0** và **INT1** bình thường ở **mức cao** và nếu một tín hiệu ở **mức thấp** được cấp tới thì chúng ghi nhãn ngắt. Sau đó bộ vi điều khiển dừng tất cả mọi công việc nó đang thực hiện và nhảy đến bảng vector ngắt để phục vụ ngắt. Đây là chế độ ngắt **mặc định** khi cấp nguồn cho 8051.

**Tín hiệu mức thấp tại chân INTx phải được lấy đi trước khi thực hiện lệnh cuối cùng của trình phục vụ ngắt, nếu không một ngắt khác sẽ lại được tạo ra, và vi điều khiển sẽ thực hiện ngắt liên tục.**

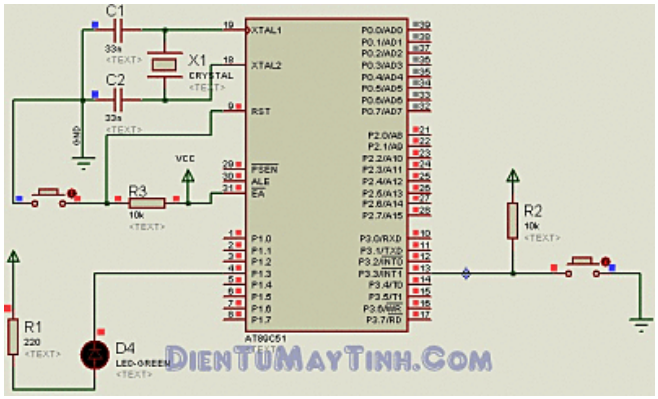
Để rõ hơn chúng ta hãy xem **ví dụ 4**.

**Ví dụ 4:**

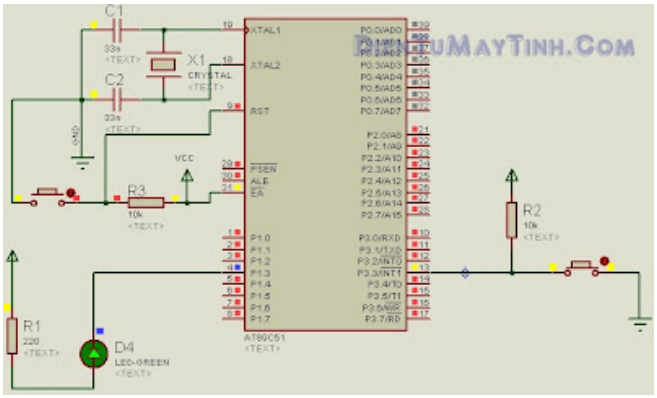
Giả sử chân **INT1** được nối đến công tắc bình thường ở mức cao. Mỗi khi nó **ấn xuống thấp** phải bật một đèn **LED** ở chân P1.3 (bình thường Led tắt), khi nó được bật lên nó phải sáng vài giây. Chừng nào công tắc được **giữ ở trạng thái thấp** đèn LED phải sáng liên tục.

**Lời giải:**

```
#include<at89x51.h>        //Khai báo thư viện cho VĐK 89x51
main()                              //Chương trình chính
{
        IE=0x84;              //cho phép ngắt ngoài 1
        while(1)              //vòng lặp vô hạn
        {
                              //không làm gì
        }
}
void nutan(void) interrupt 2        //Khai báo trình phục vụ ngắt ngoài 1
{                                    //(mặc định là ngắt theo mức)
        int a=50000;                //Biến đếm trễ
        P1_3=0;                     //Cho Led sáng
        while(a--){}                //Trễ cho Led sáng vài giây
        P1_3=1;                     //Tắt Led
                                    //Không cần xóa cờ ngắt
}
```




**Hình 8:** Ấn công tắc xuống sẽ làm cho đèn LED sáng một thời gian.




**Hình 9:** Nhưng nếu công tắc được giữ ở trạng thái ấn thì đèn LED sáng liên tục.

**Lưu ý:**

- Trong chương trình trên bộ vi điều khiển quay vòng liên tục trong vòng lặp while(1)của chương trình chính. Mỗi khi công tắc trên chân **P3.3** (**INT1**) được kích hoạt thì bộ vi điều khiển thoát khỏi vòng lặp và nhảy đến bảng vector ngắt tại địa chỉ 0013H. Trình **ISR** cho **INT1** bật đèn LED lên giữ nó một lúc và tắt nó trước khi trở về. Nếu trong lúc nó thực hiện lệnh cuối cùng để quay trở về từ **ISR** mà chân **INT1** vẫn còn ở**mức thấp** thì bộ vi điều khiển khởi tạo lại ngắt, ngắt lại xảy ra 1 lần nữa.

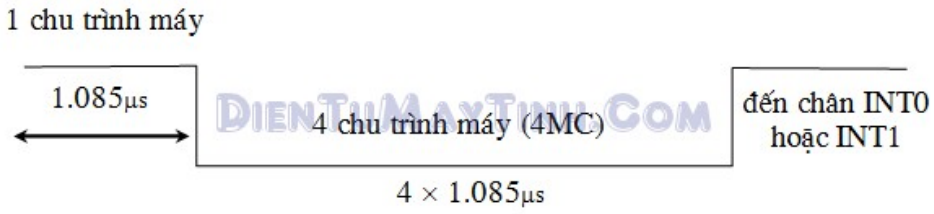
· Do vậy, để giải quyết vấn đề này thì chân **INT1** phải được đưa lên cao trước thời điểm lệnh cuối cùng của ngắt được thực hiện.
· Có một cách khác để giải quyết triệt để vấn đề trên: đó là sử dụng **ngắt theo sườn.**Khi đó với mỗi 1 lần ấn phím, dù thế nào ngắt cũng chỉ thực hiện 1 lần.
· Trước khi tìm hiểu **ngắt theo sườn** là gì? Ta hãy xem qua **ngắt theo mức** hoạt động như thế nào.

**Ø Trích mẫu ngắt theo mức**

Các chân **P3.2** và **P3.3** bình thường được dùng cho **vào/ra** nếu các Bit **INT0** và**INT1** trong thanh ghi **IE** không được kích hoạt. Sau khi các ngắt phần cứng trong thanh gi**IE** được kích hoạt thì bộ vi điều khiển **duy trì trích mẫu** trên chân **INTx** đối với tín hiệu mức thấp **1 lần trong 1 chu trình máy**.

Theo bảng dữ liệu từ nhà sản xuất của bộ vi điều khiển thì "**chân ngắt phải được giữ ở mức thấp cho đến khi bắt đầu thực hiện trình phục vụ ngắt ISR. Nếu chân INTx được đưa trở lại mức cao trước khi bắt đầu thực hiện ISR thì sẽ chẳng có ngắt nào xảy ra**". Do vậy, để bảo đảm việc kích hoạt ngắt phần cứng tại các chân **INTx** phải đảm bảo rằng thời gian tồn tại tín hiệu **mức thấp** là khoảng **4 chu trình máy** và không được bé hơn, nếu không đủ lâu thì ngắt không được thực hiện.

Tuy nhiên trong quá trình kích hoạt **ngắt theo mức thấp** nên nó lại phải đưa lên**mức cao** trước khi **ISR** thực hiện lệnh cuối cùng và lại theo bảng dữ liệu từ nhà sản xuất thì "**nếu chân INTx vẫn ở mức thấp sau lệnh cuối cùng của trình phục vụ ngắt thì một ngắt khác lại sẽ được kích hoạt**". Điều này do một thực tế là **ngắt theo mứckhông được chốt**.

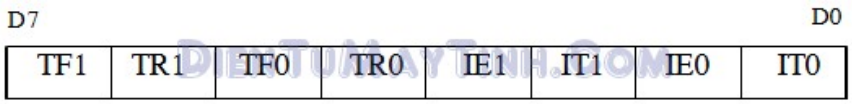



**Hình 10:** Thời gian tối thiểu của xung ngắt theo mức thấp (XTAL = 11.0592MHz)

**3.2 Ngắt theo sườn**

**Ngắt theo sườn** là ngắt sẽ xảy ra khi có một **sườn âm** xuất hiện trên các chân ngắt của vi điều khiển. Điều này làm cho ngắt theo sườn khắc phục được nhược điểm của **ngắt theo mức** như ta đã thấy ở trên.

Để kích hoạt chế độ **ngắt theo sườn** thì chúng ta phải viết chương trình cài đặt cho các bit của thanh ghi **TCON**:




**Hình 11:** Thanh ghi **TCON**.

**Ø Các Bit IT0 và IT1**

Các bit **TCON.0** và **TCON.2** được coi như là các bit **IT0** và **IT1** tương ứng. Đây là các bit xác định kiểu ngắt theo sườn xung hay theo mức xung của các ngắt phần cứng trên chân **INT0** và **INT1** tương ứng. Khi bật lại nguồn cả 2 bit này đều có mức **0** để biến chúng thành ngắt theo tín hiệu **mức thấp**. Lập trình viên có thể điều khiển một trong số chúng lên cao để chuyển ngắt phần cứng bên ngoài thành **ngắt theo sườn**.

**Ø Các Bit IE0 và IE1**

Các bit **TCON.1** và **TCON.3** còn được gọi là **IE0** và **IE1** tương ứng. Các bit này được 8051 dùng để bám kiểu ngắt theo sườn xung, nếu các bit **IT0** và **IT1** bằng **0** thì có nghĩa là các ngắt phần cứng là ngắt theo **mức thấp** và các bit **IE0** và **IE1** sẽ không dùng đến. Các Bit **IE0** và **IE1** chỉ được 8051 dùng để **chốt sườn xung** từ cao xuống thấp trên các chân **INT0** và **INT1**. Khi có chuyển dịch sườn xung trên chân **INT0** (hay **INT1**) thì 8051 đánh dấu (bật lên cao) các bit **IEx** trên thanh ghi **TCON** và nhảy đến bảng vector ngắt và bắt đầu thực hiện trình phục vụ ngắt **ISR**. Trong khi 8051 thực hiện **ISR** thì không có một sườn xung nào được ghi nhận trên chân **INT0** (hay **INT1**) để ngăn mọi ngắt trong ngắt. Chỉ trong khi thực hiện lệnh cuối của trình phục vụ ngắt **ISR** thì các bit **IEx** mới được 8051 tự động xóa, và các chân ngắt lại hoạt động bình thường.

Ta thấy rằng các bit **IE0** và **IE1** được 8051 sử dụng bên trong để báo có một ngắt đang được xử lý hay không. Hay nói cách khác là lập trình viên không phải quan tâm đến các bit này.

Ø **Các Bit TR0 và TR1**

Đây là những bit D4 và D6 (hay TCON.4 và TCON.6) của thanh ghi TCON. Các bit này đã được giới thiệu ở các bài trước, chúng được dùng để khởi động và dừng các bộ định thời Timer0 và Timer1 tương ứng.

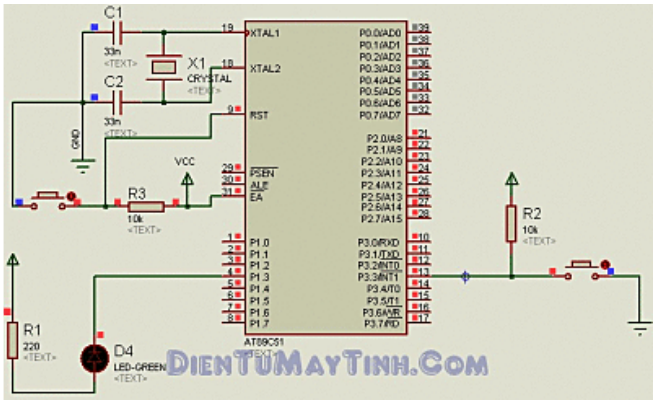
Ø **Các Bit TF0 và TF1**

Các bit này là D5 (TCON.5) và D7 (TCON.7) của thanh ghi TCON mà đã được giới thiệu ở các bài trước. Chúng được sử dụng bởi các bộ Timer0 và Timer1 tương ứng để báo rằng các bộ định thời bị tràn hay quay về không.

Để hiểu rõ sự khác biệt của ngắt theo sườn âm, ta xét **ví dụ 5**. Chú ý rằng sự khác nhau duy nhất giữa **ví dụ 5** và **ví dụ 4** là ở lệnh chuyển ngắt **INT1** về kiểu **ngắt theo sườn**. Khi **sườn âm** của tín hiệu được cấp đến chân **INT1** thì đèn LED sẽ bật lên một lúc. Đèn LED có thời gian sáng phụ thuộc vào độ trễ bên trong **ISR** của **INT1**. Trong **ví dụ 4** do bản chất **ngắt theo mức** của ngắt thì đèn LED còn sáng chừng nào tín hiệu ở chân **INT1**vẫn còn ở **mức thấp**. Nhưng trong **ví dụ 5** này để bật lại đèn LED thì xung ở chân **INT1**phải được đưa lên cao rồi sau đó bị hạ xuống thấp để tạo ra một **sườn âm** làm kích hoạt ngắt.

**Ví dụ 5:**

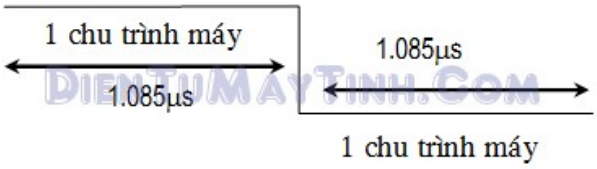
```
#include<at89x51.h>        //Khai báo thư viện cho VĐK 89x51
main()                              //Chương trình chính
{
        IE=0x84;             //cho phép ngắt ngoài 1
        IT1=1;               //Thiết lập ngắt ngoài 1 theo sườn âm
        while(1)             //vòng lặp vô hạn
        {
                             //không làm gì
        }
}
void nutan(void) interrupt 2        //Khai báo trình phục vụ ngắt ngoài 1
{                                    //(mặc định là ngắt theo mức)
        int a=50000;                //Biến đếm trễ
        P1_3=0;                      //Cho Led sáng
        while(a--){}                //Trễ cho Led sáng vài giây
        P1_3=1;                      //Tắt Led
                                     //Không cần xóa cờ ngắt
}
```




**Hình 12:**mô phỏng ngắt ngoài 1 theo **sườn âm:**Dù công tắc được giữ, cũng chỉ có 1 ngắt xảy ra.

Ø **Trình mẫu ngắt theo sườn**

Trước khi kết thúc phần này ta cần trả lời câu hỏi: vậy thì ngắt theo sườn được trích mẫu thường xuyên như thế nào? Trong các ngắt theo sườn**, nguồn xung phải giữ ở mức cao tối thiểu là 1 chu kỳ máy, và xung thấp cũng phải kéo dài 1 chu kỳ máy nữa**để đảm bảo bộ vi điều khiển nhìn thấy được sự chuyển dịch từ cao xuống thấp của **sườn âm**.




**Hình 13:** Thời hạn xung tối thiểu để phát hiện ra các ngắt theo sườn âm với tần số XTAL = 11.0592MHz

**Sườn âm** của xung được chốt bởi 8051 và được giữ bởi thanh ghi **TCON**. Các bit**TCON.1 (IE0)** và **TCON.3 (IE1)** giữ các sườn được chốt của chân **INT0** và **INT1** tương ứng như chỉ ra trên **hình 11**. Chúng hoạt động như các cờ "**ngắt đang được phục vụ**" (Interrupt-in-server). Khi một cờ "**ngắt đang được phục vụ**" bật lên thì nó báo rằng ngắt hiện nay đang được xử lý và trên chân **INTx** này sẽ không có ngắt nào được đáp ứng chừng nào ngắt này chưa được phục vụ xong. Đây giống như tín hiệu báo bận ở máy điện thoại.

Ngoài ra cần phải nhấn mạnh 2 điểm dưới đây khi quan tâm đến các bit **IE0** và **IE1**của thanh ghi **TCON**:

- Khi các trình phục vụ ngắt **ISR** kết thúc: Các Bit **IE0** và **IE1** được tự động xoá để báo rằng ngắt được hoàn tất và 8051 sẵn sàng đáp ứng ngắt khác trên chân đó. Để ngắt khác được nhận và thì tín hiệu trên chân đó phải trở lại mức cao và sau đó nhảy xuống thấp để được phát hiện như một ngắt theo sườn âm.
- Trong thời gian trình phục vụ ngắt đang được thực hiện thì chân **INTx** bị làm ngơ, 8051 không quan tâm đến nó có bao nhiêu lần chuyển dịch từ cao xuống thấp. Trong thực tế điều này có được là do các bit **IEx**. Vì lý do này mà các bit **IEx** được gọi là các cờ báo "**ngắt đang được phục vụ**", cờ này sẽ lên cao khi 1 **sườn âm** được phát hiện trên chân **INTx** và **giữ ở mức cao** trong toàn bộ quá trình thực hiện **ISR**. Nó chỉ bị xoá sau lệnh cuối cùng của **ISR**. **Do vậy, ta cũng sẽ không bao giờ cần đến các lệnh xoá cờ này trong trình phục vụ ngắt đối với các ngắt cứng INT0 và INT1.**

## 4 Lập trình ngắt truyền thông nối tiếp

Trong các bài trước chúng ta đã nghiên cứu về truyền thông nối tiếp của 8051. Tất cả các ví dụ trong ấy đều sử dụng phương pháp thăm dò (polling). Ở chương này chúng ta sẽ khám phá phương pháp truyền thông nối tiếp dựa trên ngắt.

### 4.1 Các cờ RI và TI và các ngắt

Như đã nói ở bài trước thì cờ ngắt truyền **TI** (Transfer interrupt) được bật lên khi bit cuối cùng của khung dữ liệu - **bit stop** được truyền đi, báo rằng thanh ghi **SBUF** sẵn sàng truyền byte kế tiếp. Trong trường hợp cờ **RI** (Receive Interrupt) thì nó được bật lên khi toàn bộ khung dữ liệu kể cả **bit stop** đã được nhận.

Chừng nào còn nói về truyền thông nối tiếp thì tất cả mọi khái niệm trên đây đều áp dụng giống như nhau cho dù sử dụng phương pháp thăm dò hay sử dụng phương pháp ngắt. Sự khác nhau duy nhất giữa hai phương pháp này là ở cách phục vụ quá trình truyền thông nối tiếp như thế nào:

Ø Trong phương pháp thăm dò thì chúng ta phải đợi cho cờ (**TI** hay **RI**) bật lên và trong lúc chờ đợi thì ta không thể làm gì được cả.
Ø Còn trong phương pháp ngắt thì ta được báo khi 8051 đã nhận được một byte hoặc nó sẵn sàng truyền byte kế tiếp và ta có thể làm các công việc khác trong khi chờ truyền thông nối tiếp được thực hiện.

Trong 8051 chỉ có **một ngắt dành riêng cho truyền thông nối tiếp**. Ngắt này được dùng cho **cả truyền và nhận dữ liệu**. Nếu bit ngắt truyền thông **ES** - **IE.4** trong thanh ghi **IE** được phép, thì khi 1 trong 2 cờ **RI** hoặc **TI** bật lên, 8051 sẽ nhận được ngắt và nhảy đến địa chỉ trình phục vụ ngắt dành cho truyền thông nối tiếp 0023H trong bảng vector ngắt để thực hiện nó. Trong trình **ISR** này chúng ta phải kiểm tra các cờ **TI** và **RI**để xem cờ nào gây ra ngắt để đáp ứng một cách phù hợp (xem **ví dụ 6**).


**Hình 14:** Ngắt truyền thông có thể do hai cờ TI và RI gọi.

### 4.2 Sử dụng cổng COM nối tiếp trong 8051

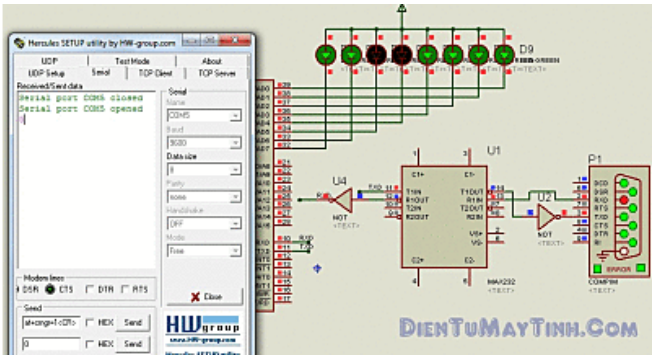
**Trong các ứng dụng, ngắt nối tiếp chủ yếu được sử dụng để nhận dữ liệu và không bao giờ được sử dụng để truyền dữ liệu nối tiếp.** Điều này giống như việc báo chuông để ta biết và nhận điện thoại vì ta không thể biết trước được lúc nào có điện thoại, còn nếu muốn gọi điện thoại thì ta không cần đổ chuông để báo trước.

**Ví dụ 6:**

Hãy viết chương trình ngắt để 8051 nhận dữ liệu từ cổng nối tiếp **COM** và gửi đến cổng **P0**. Giả thiết tần số XTAL là 11.0592MHz và tốc độ baud 9600.
**Lời giải:**

```
#include<at89x51.h>        //Khai báo thư viện cho 89c51
main()                      //Chương trình chính
{
        TMOD=0x20;          //Chọn Timer1, chế độ 2
        TH1=0xFD;           //Cài đặt tốc độ baud 9600
        SCON=0x50;          //0101 0000: Chọn chế độ 1, Cho phép nhận
        TR1=1;              //Khởi động Timer1
        IE=0x90;            //cho phép ngắt truyền thông nối tiếp
        while(1)            //Vòng lặp vô hạn
        {
        }
}
void nhandulieu(void) interrupt 4      //Khai báo ISR truyền thông nối tiếp
{
        if(RI==1)           //Kiểm tra có phải là ngắt nhận dữ liệu không
        {
                P0=SBUF;    //Gửi dữ liệu đến cổng P0
                RI=0;       //Xóa cờ nhận dữ liệu nối tiếp RI
```

```
        }
    }
```




**Hình 15:** Mô phỏng nhận các ký tự **0**,**1**,**2**,**3**,**4**, từ máy tính, gửi đến **Port0**.

Trong ví dụ trên ta chú ý đến vai trò của cờ **RI**. Trong trình phục vụ ngắt nối tiếp, ta phải kiểm tra cả cờ **TI** và cờ **RI** vì cả hai đều có thể gọi ngắt truyền thông nối tiếp, hay nói cách khác là chỉ có một ngắt cho cả truyền và nhận.

### 4.3 Xoá cờ RI và TI trước khi thoát khỏi ngắt truyền thông nối tiếp

Để ý rằng lệnh cuối cùng trước khi trở về từ **ISR** là lệnh xoá các cờ **RI** và **TI**. **Điều này tương phản với ngắt ngoài và ngắt bộ định thời là đều được 8051 xoá các cờ**.

## 5. Các mức ưu tiên ngắt trong 8051

### 5.1 Các mức ưu tiên trong quá trình bật lại nguồn

Khi 8051 được cấp nguồn thì các mức ưu tiên ngắt được gán theo **Hình 16**. Từ hình này ta thấy ví dụ nếu các ngắt phần cứng ngoài 0 và 1 được kích hoạt cùng một lúc thì ngắt ngoài 0 sẽ được đáp ứng trước. Chỉ sau khi ngắt INT0 đã được phục vụ xong thì INT1 mới được phục vụ vì INT1 có mức ưu tiên thấp hơn. Trong thực tế sơ đồ mức ưu tiên ngắt trong bảng chỉ là một quy trình thăm dò, trong đó 8051 thăm dò các ngắt theo trình tự cho trong **hình 16** và đáp ứng chúng một cách phù hợp.

| Mức ưu tiên từ cao xuống thấp | | | |
|---|---|---|---|
| **Ngắt** | **Cờ ngắt** | **Địa chỉ trình phục vụ ngắt** | **Số thứ tự ngắt** |
| **Ngắt ngoài 0** | **IE0** | 0003h | **0** |
| **Timer 0** | **TF0** | 000Bh | **1** |
| **Ngắt ngoài 1** | **IE1** | 0013h | **2** |
| **Timer 1** | **TF1** | 001Bh | **3** |
| **Ngắt truyền thông** | **RI/TI** | 0023h | **4** |



**Hình 16:** Mức ưu tiên các ngắt trong khi cấp lại nguồn.

| D7 | | | | | | | D0 |
|---|---|---|---|---|---|---|---|
| -- | -- | PT2 | PS | PT1 | PX1 | PT0 | PX0 |



**Hình 17:** Thanh ghi mức ưu tiên ngắt **IP**: Bit ưu tiên = 1 là mức ưu tiên cao, Bit ưu tiên = 0 là mức ưu tiên thấp.

- Bit D7 và D6 -- chưa dùng.
- Bit D5 hay **PT2** là Bit ưu tiên ngắt Timer2 (dùng cho 8052)
- Bit D4 hay **PS** là Bit ưu tiên ngắt cổng nối tiếp
- Bit D3 hay **PT1** là Bit ưu tiên ngắt Timer1
- Bit D2 hay **PX1** là mức ưu tiên ngắt ngoài 1

- Bit D1 hay **PT0** là mức ưu tiên ngắt Timer 0
- Bit D0 hay **PX0** là mức ưu tiên ngắt ngoài 0

### 5.2 Thiết lập mức ưu tiên ngắt với thanh ghi IP

Chúng ta có thể thay đổi trình tự trong **hình 16** bằng cách gán mức ưu tiên cao hơn cho bất kỳ ngắt nào. Điều này được thực hiện bằng cách lập trình một thanh ghi gọi là thanh ghi mức ưu tiên ngắt **IP** (**Interrupt Priority**). Trên **hình 17** là các bit của thanh ghi này. Khi bật lại nguồn thanh thi **IP** chứa hoàn toàn các số **0** để tạo ra trình tự ưu tiên ngắt theo **Hình 16**. Để một ngắt nào đó có mức ưu tiên cao hơn ta thực hiện đưa bit tương ứng lên cao.

Một điểm khác nữa cần được làm sáng tỏ là mức ưu tiên ngắt **khi 2 hoặc nhiều bit ngắt trong thanh ghi IP được đặt lên cao**: Trong trường hợp này thì trong khi các ngắt này có mức ưu tiên cao hơn các ngắt khác, chúng sẽ được phục vụ theo trình tự cho trong **Hình 16**.

### 5.3 Ngắt trong ngắt

Điều gì xảy ra nếu 8051 đang thực hiện một trình phục vụ ngắt thuộc một ngắt nào đó thì lại có một ngắt khác được kích hoạt? Trong những trường hợp như vậy thì **1 ngắt có mức ưu tiên cao hơn có thể ngắt 1 ngắt có mức ưu tiên thấp hơn**. Đây gọi là ngắt trong ngắt. Trong 8051 một ngắt ưu tiên thấp có thể bị ngắt bởi một ngắt có mức ưu tiên cao hơn chứ không bị ngắt bởi một ngắt có mức ưu tiên thấp hơn. Mặc dù tất cả mọi ngắt đều được chốt và giữ bên trong nhưng không có ngắt mức thấp nào được CPU quan tâm ngay tức khắc, nếu 8051 chưa kết thúc phục vụ các ngắt mức cao.

### 5.4 Thu chộp ngắt bằng phần mềm (Triggering)

Có nhiều lúc ta cần kiểm tra một trình phục vụ ngắt bằng con đường mô phỏng. Điều này có thể được thực hiện bằng các lệnh đơn giản để thiết lập các ngắt lên cao và bằng cách đó buộc 8051 nhảy đến bảng vector ngắt. Ví dụ, nếu bit cho phép ngắt **Timer1**trong thanh ghi **IE** được bật lên 1 thì một lệnh như **TF1=1;** sẽ ngắt 8051 ngừng thực hiện công việc đang làm bất kỳ và buộc nó nhảy đến bảng vector ngắt timer1. Hay nói cách khác, ta không cần đợi cho Timer1 quay trở về 0 mới tạo ra ngắt. Chúng ta có thể gây ra một ngắt bằng các lệnh đưa các bit của ngắt tương ứng lên cao.

Như vậy qua bài này chúng ta đã biết ngắt là một sự kiện bên trong hoặc bên ngoài gây ra ngắt bộ vi điều khiển để báo cho nó biết rằng thiết bị cần được phục vụ. Mỗi một ngắt có một chương trình đi kèm với nó được gọi là trình phục vụ ngắt **ISR**. Bộ vi điều khiển 8051 có 6 ngắt, trong đó có **5 ngắt** người dùng có thể truy cập được. Đó là: 2 ngắt cho các thiết bị phần cứng bên ngoài **INT0** và **INT1**, 2 ngắt cho các bộ định thời là **TF0** và**TF1** và 1 ngắt dành cho **truyền thông nối tiếp**.

8051 có thể được lập trình cho phép hoặc cấm một ngắt bất kỳ cũng như thiết lập mức ưu tiên cho nó theo yêu cầu của thuật toán ứng dụng.

### Nhận xét

Bạn không có quyền thêm nhận xét.