Chương 4: NGẮT

Để hiểu về ngắt chúng ta hãy xem 1 ví dụ sau:
Bộ vi điều khiển đóng vai trò như một vị bác sĩ, các thiết bị được điều khiển 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! 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):
* **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.
*** 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.

Điểm mạnh của phương pháp ngắt là:
- 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.
- 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 chương 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.
Dưới đây chúng ta sẽ tìm hiểu kỹ về ngắt

4.1. Các ngắt của 8051

Theo các nhà sản xuất thí vi điều khiển 8051 có 6 ngắt được phân bố như sau:

- **RESET**: Khi chân RESET được kích hoạt từ 8051, bộ đếm chương trình nhảy về địa chỉ **0000H**.
- **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**.
- **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**.
- **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**.

4.1.1 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 **Bảng 4.1**.

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

Bảng 1: Bảng vector ngắt của 8051.

4.124 Quy trình khi thực hiện một ngắt

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

- Vi điều khiển sẽ hoàn thành nốt lệnh đang thực hiện và lưu địa chỉ của **lệnh kế tiếp** vào ngăn xếp.
- Nó cũng **lưu tình trạng hiện tại** của tất cả các ngắt.
- 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**.
- 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.

4.1.3 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. Bảng 4.2 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

Bảng 4.2: Thanh ghi cho phép ngắt **IE**.

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

- 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.
- 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**.

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:

a) Mov IE, #96h; //1001 0110: lệnh này tương đương với 4 lệnh phía dưới

Hoặc

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

b)

```
Mov ET0,#00h;        //Cấm ngắt timer0
```

c)

Mov EA,#00h; //Cấm tất cả các ngắt

4.2. Lập trình các ngắt bộ định thời

Trong các chương 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.

4.2.1 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 chương trình 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 4.1: Ngắt bộ định thời **TF0** và **TF1**.

4.2.2. Các bước lập trình ngắt định thời

Để hiểu trình tự lập trình ngắt định thời, ta xem các ví dụ sau:

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:

Bước 1: tìm giá trị cần nạp cho thanh ghi timer và xác định chế độ hoạt động của timer

Theo yêu cầu Chu kỳ 200ms, vậy nửa chu kỳ là 100ms.

Ta có: **100ms/1,085ms=92**. (Đối với mạch hoạt động với tần số thạch anh 11.0592 Mhz)

Suy ra giá trị cần nạp cho timer0 là: **-92** <=> **A4H**. Ta sử dụng timer0 chế độ 2, 8 bit tự nạp lại.

Chương trình viết bằng Assembly

```
Org 0000h
Ljmp Main       ; Nhảy đến chương trình chính, tránh dùng không gian bộ nhớ
                ; của ngắt
Org 000Bh       ; Bảng vectơ ngắt của timer0
Cpl P2.1        ; Đảo mức cho chân P2.1
Reti            ; Kết thúc chương trình con, về chương trình chính
```

```
        Org 0030h
Main:   Mov TMOD,#02h   ; Chọn timer 0, chế độ 2 – tự nạp lai
        Mov TH0,#-92    ; Nạp giá trị cho thanh ghi TH0
        Mov TL0,#-92    ; Nạp giá trị ban đầu cho thanh ghi TL0
        Mov P0,#0FFh    ; Thiết lập chân P0 làm cổng vào
        Mov IE,#82h     ; IE = 1000.0010 – cho phép ngắt toàn cục và timer 0
        SetB TR0        ; Khởi động timer 0
Back:   Mov A,P0        ; Lấy giá trị cổng vào P0
        Mov P1,A        ; Xuất giá trị ra lại cổng 1
        Sjmp Back       ; Lặp lại quá trình
        End             ; Không cần xóa cờ TF0, 8051 tự động xóa.
```

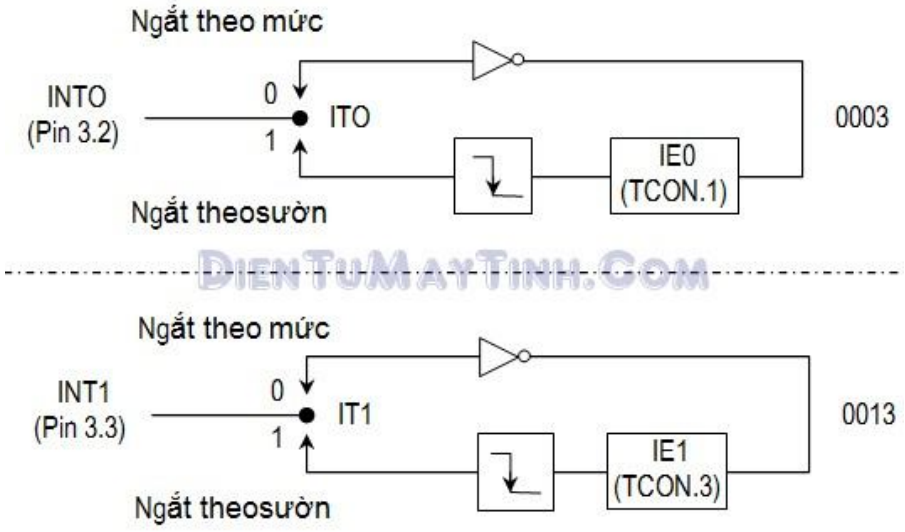
Lưu ý: Các xung được tạo ra ở các ví dụ trên không thật sự chính xác, vì chưa tính đến hao phí của các lệnh.

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

1. Trình ứng dụng không được sử dụng vùng không gian bộ nhớ của ngắt, do đó có lệnh Ljmp Main
2. Trình phục ngắt có không gian bộ nhớ nhỏ
3. Để cho phép ngắt chúng ta phải thực hiện lệnh cho phép ngắt bộ **Timer0** với lệnh Mov IE, #82h; trong chương trình chính main.
4. 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 vòng lặp BACK và nhảy đến địa chỉ **000BH** để thực hiện trình phục vụ **ISR** của bộ **Timer0**.
5. Trong trình phục vụ ngắt **ISR** của **Timer0** ta thấy rằng không cần đến lệnh xóa cờ **TF0** của **timer0** trước lệnh **RETI**. Lý do này là vì **8051 đã tự xoá cờ TF0 ngay khi nhảy đến ISR.**

4.3 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 4.2: Ngắt ngoài **INT0** và **INT1**

Dưới đây là mô tả hoạt động của mỗi loại.

4.3.1 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:

```
Org 0000h
Ljmp Main
Org 0013h
Setb P1.3
Mov R7,#250d
Here: Djne R7, Here
Clr P1.3
Reti
Org 0030h
Main:
Mov IE,#84h
```

```
Jmp $
End
```

Lưu ý: 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.

4.3.2 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**:

D7							D0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Hình 4.3: Thanh ghi **TCON**.

a)**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**.

b) **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.

c)**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 phần trước, chúng được dùng để khởi động và dừng các bộ định thời Timer0 và Timer1 tương ứng.

d) **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.