

8051 Microcontrollers

I. Các khái niệm

Khi nói đến cơ bản của vi điều khiển, chúng ta phải biết về các thành phần khác nhau của vi điều khiển bao gồm: bộ xử lý trung tâm (CPU), bộ nhớ truy cập ngẫu nhiên (RAM), bộ nhớ chỉ đọc (ROM), đầu vào / đầu ra bộ định thời cổng, bộ điều khiển ngắn bộ đếm, bộ chuyển đổi analog sang kỹ thuật số, bộ chuyển đổi kỹ thuật số analog, cổng giao tiếp nối tiếp và mạch dao động.

- CPU: được gọi là bộ não với chức năng chính là tìm nạp và giải mã các lệnh để các chức năng khác được thực hiện trơn tru.
- Bộ nhớ: khi nói đến bộ nhớ của bộ vi điều khiển chúng ta sẽ hình dung ra bộ vi xử lý và các bộ nhớ khác nhau được cài đặt bên trong bộ vi điều khiển là RAM và ROM (EEPROM, EPROM, v.v.) hoặc bộ nhớ flash để lưu trữ mã nguồn chương trình.
- Cổng đầu ra và cổng đầu vào song song: mục tiêu chính của các cổng này bên trong vi điều khiển là điều khiển các giao diện khác nhau giữa các thiết bị được kết nối.
- Các cổng nối tiếp: đây cũng là một bộ phận quan trọng của vi điều khiển.
- Bộ định thời và bộ đếm: số bộ định thời và bộ đếm bên trong vi điều khiển khác nhau và chủ yếu chúng được sử dụng cho mục đích khóa chức năng, điều chế, tạo xung, đo tần số và tạo dao động để tác vụ có thể được thực hiện trong khoảng thời gian quy định.
- Bộ chuyển đổi analog sang kỹ thuật số và bộ chuyển đổi kỹ thuật số sang analog: là bộ chuyển đổi được sử dụng bên trong vi điều khiển để chuyển đổi tín hiệu từ analog sang kỹ thuật số và ngược lại.
- Kiểm soát ngắn: bản thân tên gọi này đã tự giải thích và nó giúp thực hiện chương trình mà không bị gián đoạn.
- Khối chức năng đặc biệt: Đây là phần bổ sung đặc biệt và bổ sung cho vi điều khiển để thực hiện một số nhiệm vụ đặc biệt.

II. Giới thiệu

Vào năm 1981, hãng Intel giới thiệu một số bộ vi điều khiển được gọi là 8051. Bộ vi điều khiển này có 128B RAM, 4KB ROM trên chip, hai bộ định thời, một cổng nối tiếp và 4 cổng (đều rộng 8 bit) vào ra tất cả được đặt trên một chíp. Lúc ấy nó được coi là một “hệ thống trên chíp”. 8051 là một bộ xử lý 8 bit có nghĩa là **CPU chỉ có thể làm việc với 8 bit dữ liệu tại một thời điểm. Dữ liệu lớn hơn 8 bit được chia ra thành các dữ liệu 8 bit để xử lý.**

Đặc tính	8051
ROM trên chíp	4K byte
RAM	128 byte
Bộ định thời	2
Chân vào - ra	32
Cổng nối tiếp	1
Nguồn ngắt	6

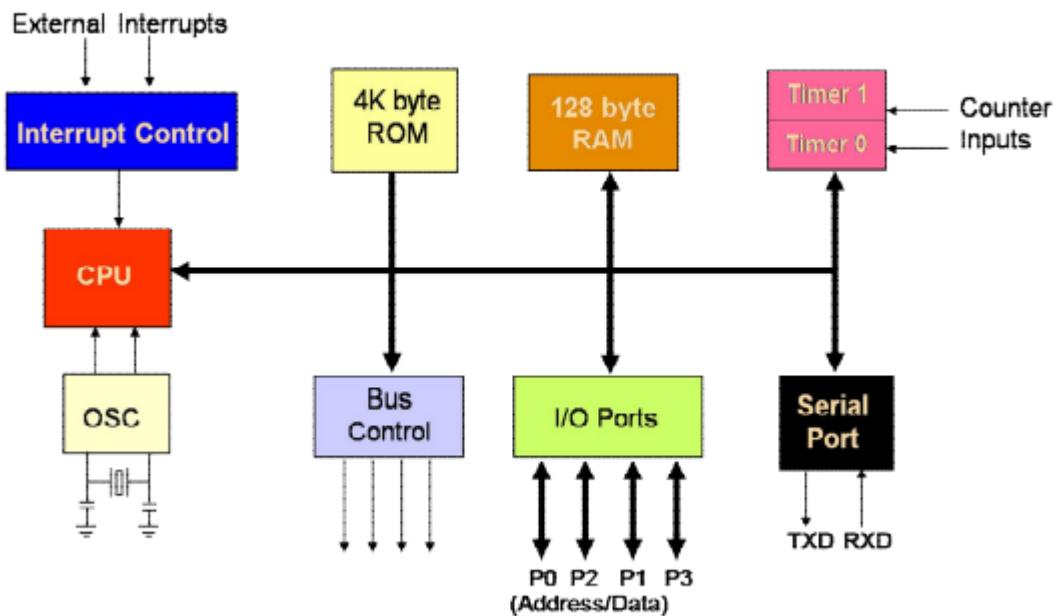
III. Kiến trúc phần cứng

1. Sơ đồ khối

Kiến trúc cơ bản bên trong 8051 bao gồm các khối chức năng sau

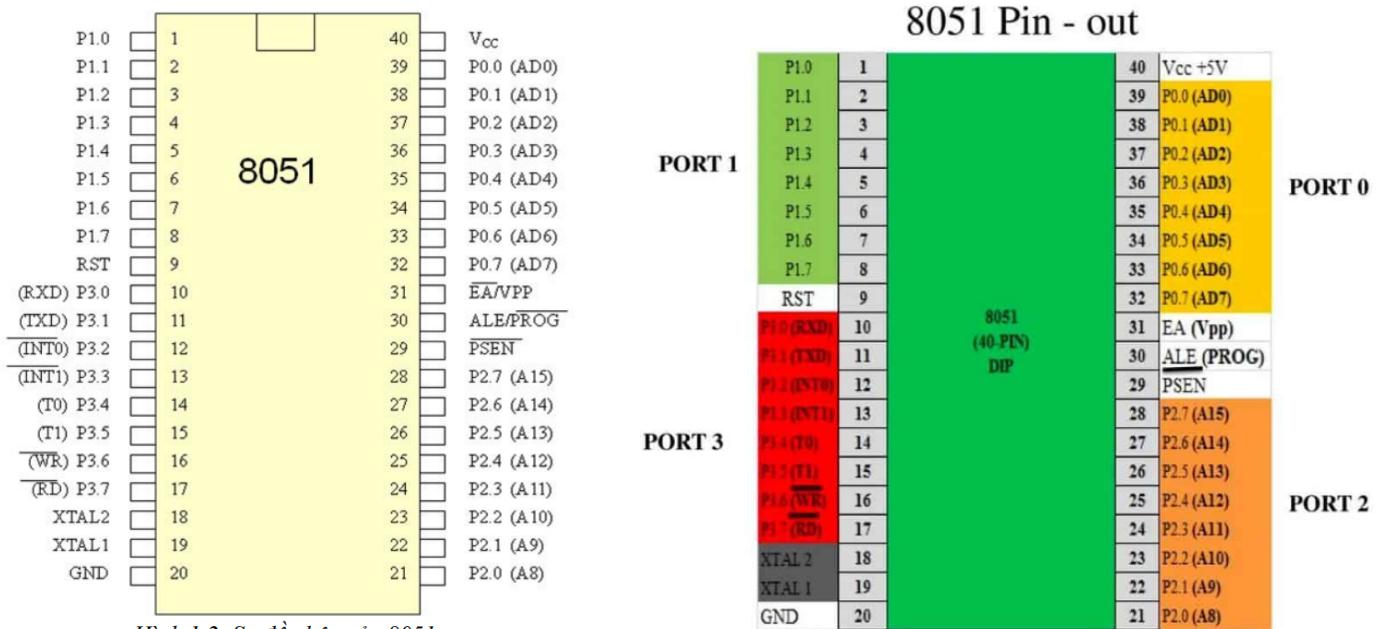
- CPU (Central Processing Unit): đơn vị điều khiển trung tâm
- Bộ nhớ chương trình ROM bao gồm 4 Kbyte
- Bộ nhớ dữ liệu RAM bao gồm 128 byte
- Bốn cổng xuất nhập tương ứng với 32 chân I/O
- Hai bộ định thời/bộ đếm 16 bit thực hiện chức năng định thời và đếm sự kiện
- Bộ giao diện nối tiếp (1 cổng nối tiếp)
- Khối điều khiển ngắt với hai nguồn ngắt ngoài
- Bộ chia tần số

The 8051 Block Diagram



Hình 1.1: Sơ đồ khối của bộ vi điều khiển 8051

2. Sơ đồ chân và chức năng các chân



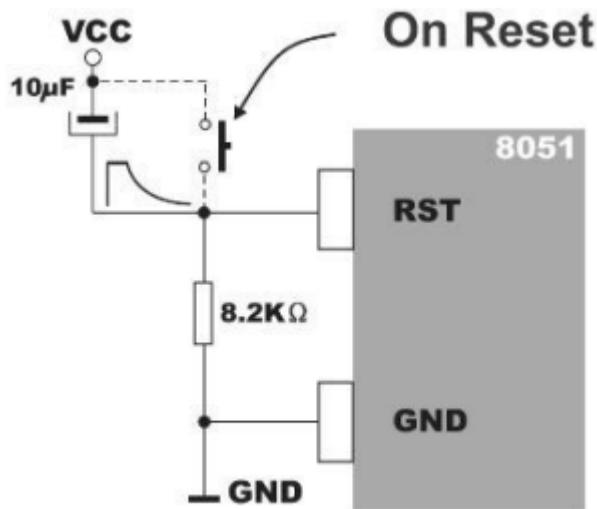
Hình 1.2: Sơ đồ chân của 8051

Chân 1 đến 8: được gọi là Cổng 1 (Port 1)

Tám chân này có duy nhất 1 chức năng là xuất và nhập. Cổng 1 có thể xuất và nhập theo bit hoặc byte. Ta đánh tên cho mỗi chân của Port 1 là P1.X (X = 0 đến 7)

Chân 9: là chân vào reset của 8051

Khi tín hiệu này được đưa lên mức cao trong ít nhất là 2 chu kỳ máy, các thanh ghi trong bộ vi điều khiển được tải những giá trị thích hợp để khởi động hệ thống. Hay nói cách khác là vi điều khiển sẽ bị reset nếu chân này được kích hoạt mức cao.



Hình 1.3: Sơ đồ mạch reset ngoài của 8051

Chân 10 đến 17: được gọi là Cổng 3 (Port 3)

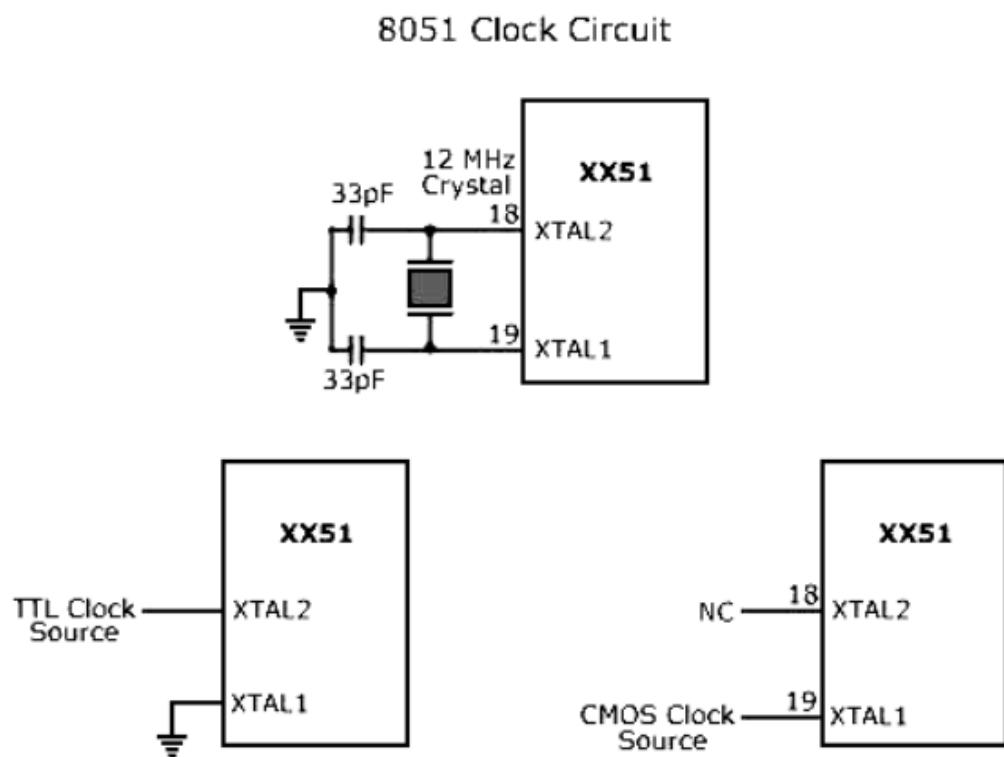
Tám chân này ngoài chức năng là xuất và nhập như các chân ở cổng 1 (chân 1 đến 8) thì mỗi chân này còn có chức năng riêng nữa, cụ thể như sau:

Bit	Tên	Chức năng
P3.0	RxD	Chân nhận dữ liệu cho cổng nối tiếp
P3.1	TxD	Chân truyền dữ liệu cho cổng nối tiếp
P3.2	INT0	Chân ngắt bên ngoài 0
P3.3	INT1	Chân ngắt bên ngoài 1
P3.4	T0	Ngõ vào của Timer/counter 0
P3.5	T1	Ngõ vào của Timer/counter 1
P3.6	WR	Xung ghi bộ nhớ dữ liệu ngoài
P3.7	RD	Xung đọc bộ nhớ dữ liệu ngoài

Bảng 1.3: Bảng mô tả chức năng riêng của cổng 3

Chân 18 và 19 (XTAL1 & XTAL2)

Hai chân này được sử dụng để nối với bộ dao động ngoài



Hình 1.4: Mạch dao động cấp cho 8051

Thông thường một bộ dao động thạch anh sẽ được nối tới các chân đầu vào XTAL1(chân 19) và XTAL2 (chân 18) cùng với hai tụ gốm giá trị khoảng 30pF. Một phía của tụ điện được nối xuống đất như hình trên.

Các hệ thống xây dựng trên 8051 thường có tần số thạch anh từ 10 đến 40 MHz, thông thường ta dùng thạch anh 12 MHz

Chân 20: được nối vào chân 0V của nguồn cấp

Chân 21 đến chân 28: được gọi là cổng 2 (Port 2)

Tám chân của cổng 2 có 2 công dụng, ngoài chức năng là cổng xuất và nhập như cổng 1 thì cổng 2 này còn là byte cao của bus địa chỉ khi sử dụng bộ nhớ ngoài.

Chân 29 (PSEN):

Chân PSEN là chân điều khiển đọc chương trình ở bộ nhớ ngoài, nó được nối với chân OE của ROM ngoài để cho phép đọc các byte mã lệnh trên ROM ngoài. PSEN ở mức

thấp trong thời gian đọc mã lệnh. Khi thực hiện chương trình trong ROM nội thì PSEN được duy trì ở mức cao

Chân 30 (ALE):

Chân ALE cho phép tách các đường dữ liệu và các đường địa chỉ tại Port 0 và Port 2.

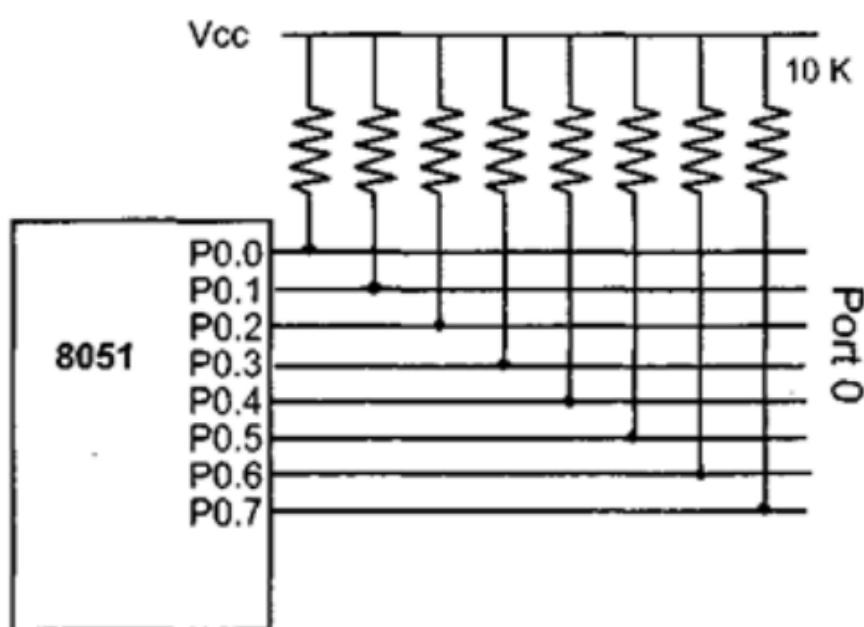
Chân 31 (EA):

Tín hiệu chân EA cho phép chọn bộ nhớ chương trình là bộ nhớ trong hay ngoài vi điều khiển. **Nếu chân EA được nối ở mức cao (nối nguồn Vcc), thì vi điều khiển thi hành chương trình trong ROM nội. Nếu chân EA ở mức thấp (được nối GND) thì vi điều khiển thi hành chương trình từ bộ nhớ ngoài.**

Chân 32 đến 39: được gọi là cổng 0 (Port 0)

Cổng 0 gồm 8 chân cung có 2 công dụng, ngoài chức năng xuất nhập, cổng 0 còn là bus đa hợp dữ liệu và địa chỉ, chức năng này sẽ được sử dụng khi 8051 giao tiếp với các thiết bị ngoài có kiến trúc Bus như các vi mạch nhớ...

Vì cổng P0 là một máng mở khác so với các cổng P1, P2 và P3 nên các chân ở cổng 0 phải được nối với điện trở kéo khi sử dụng các chân này như chân vào/ra. Điện trở này tùy thuộc vào đặc tính ngõ vào của thành phần ghép nối với chân của port 0. Thường ta dùng điện trở kéo khoảng 4K7 đến 10K



Hình 1.5: Nối điện trở kéo cho cổng 0 của 8051

Chân 40: chân nguồn của vi điều khiển, được nối vào chân Vcc của nguồn

IV. Tổ chức bộ nhớ

Trên vi điều khiển 8051/8052 đều có cả bộ nhớ chương trình (ROM) và bộ nhớ dữ liệu (RAM). Tuy nhiên dung lượng của các bộ nhớ trên chip là hạn chế. Khi thiết kế các ứng dụng đòi hỏi bộ nhớ lớn người ta có thể dùng bộ nhớ ngoài.

1. Bộ nhớ chương trình

Bộ nhớ chương trình là bộ nhớ chỉ đọc, là nơi lưu trữ chương trình của vi điều khiển. Bộ nhớ chương trình của họ 8051 có thể thuộc một trong các loại sau ROM, EPROM, FLASH hoặc không có bộ nhớ chương trình trên chip. Với họ vi điều khiển 89xx, bộ nhớ chương trình được tích hợp sẵn trong chip có kích thước nhỏ nhất là 4kByte. Với các vi điều khiển không tích hợp sẵn bộ nhớ chương trình trên chip, buộc phải thiết kế bộ nhớ chương trình bên ngoài.

Địa chỉ đầu tiên của bộ nhớ chương trình là 0000H, chính là địa chỉ reset của vi điều khiển. Ngay khi bật nguồn hoặc reset vi điều khiển, thì CPU sẽ nhảy đến thực hiện lệnh ở địa chỉ 0000H này.

Khi sử dụng bộ nhớ trên chip thì chân EA phải được nối lên mức logic cao (+5V). Nếu bạn muốn mở rộng bộ nhớ chương trình thì chúng ta phải dùng bộ nhớ ngoài với dung lượng tối đa là 64Kbyte.

2. Bộ nhớ dữ liệu

Bộ nhớ dữ liệu tồn tại độc lập so với bộ nhớ chương trình. Họ vi điều khiển 8051 có bộ nhớ dữ liệu tích hợp trên chip nhỏ nhất là 128byte và có thể mở rộng với bộ nhớ dữ liệu ngoài lên tới 64kByte.

Bộ nhớ dữ liệu được phân chia như sau:

* Các băng thanh ghi có địa chỉ từ 00H đến 1FH

32 byte thấp của bộ nhớ nội được dùng cho các băng thanh ghi (dãy thanh ghi). Bộ lệnh 8051 hỗ trợ 8 thanh ghi R0 đến R7 và theo mặc định sau khi reset hệ thống, các thanh ghi này có các địa chỉ từ 00H đến 07H.

Do có 4 băng thanh ghi nên tại một thời điểm chỉ có duy nhất 1 băng thanh ghi được truy suất bởi các thanh ghi R0 – R7, để thay đổi các băng thanh ghi thì ta thay đổi các bit chọn băng trong thanh ghi trạng thái PSW.

* Vùng RAM địa chỉ hóa từng bit có địa chỉ từ 20h đến 2Fh

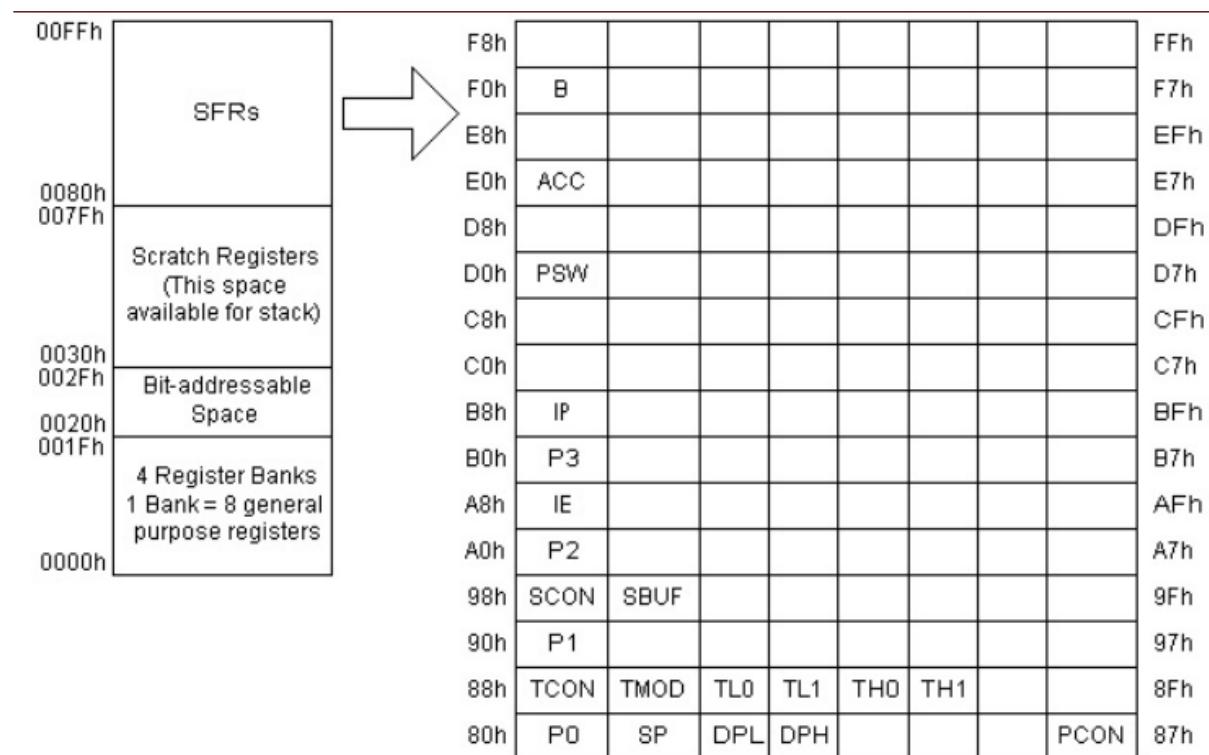
8051 chứa 210 vị trí bit được định địa chỉ trong đó 128 bit chứa trong các byte ở địa chỉ từ 20H đến 2FH (16 byte x 8 bit = 128 bit) và phần còn lại chứa trong các thanh ghi đặc biệt. Ngoài ra 8051 còn có các cổng xuất/nhập có thể định địa chỉ từng bit, điều này làm đơn giản việc giao tiếp bằng phần mềm với các thiết bị xuất/nhập đơn bit.

* Vùng RAM đa dụng có địa chỉ từ 30h đến 7Fh

Bất kỳ vị trí nhớ nào trong vùng RAM đa mục đích đều có thể được truy xuất tự do bằng cách sử dụng các kiểu định địa chỉ trực tiếp hoặc gián tiếp

* Các thanh ghi chức năng đặc biệt có địa chỉ từ 80h đến FFh

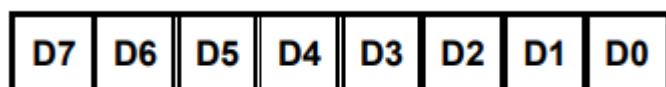
Cũng như các thanh ghi từ R0 đến R7, ta có 21 thanh ghi chức năng đặc biệt SFR chiếm phần trên của Ram nội từ địa chỉ 80H đến FFH. Cần lưu ý là không phải tất cả 128 địa chỉ từ 80H đến FFH đều được định nghĩa mà chỉ có 21 địa chỉ được định nghĩa.



Hình 1.6: Cấu trúc bộ nhớ dữ liệu của 8051

3. Các thanh ghi chức năng đặc biệt (SFR)

Thanh ghi của 8051 được dùng để lưu trữ tạm thời dữ liệu hoặc địa chỉ. Các thanh ghi này chủ yếu có kích thước 8 bit, 8 bit của các thanh ghi được sắp xếp như hình dưới trong đó bit D7 là bit có trọng số cao nhất, còn bit D0 là bit có trọng số thấp nhất.



Hình 1.7: Thanh ghi 8 bit

a. Thanh ghi chính A

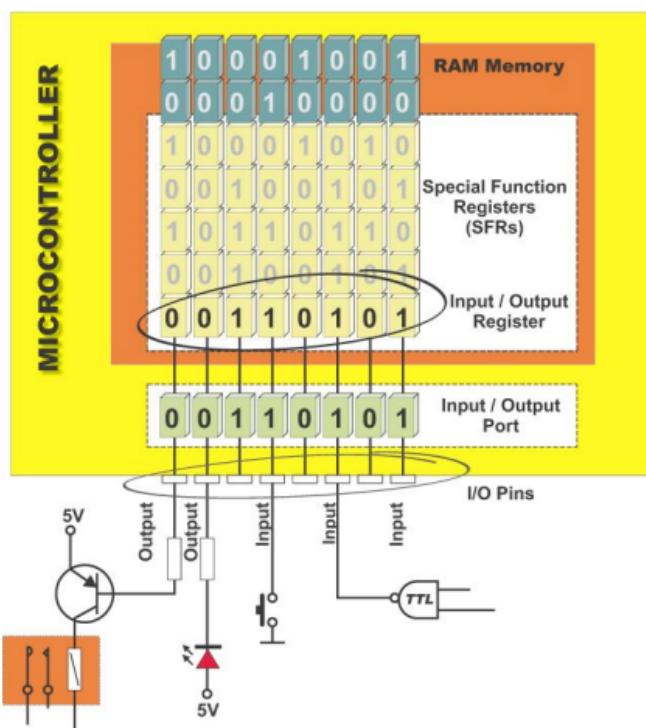
Là thanh ghi đặc biệt của 8051 dùng để thực hiện các phép toán của CPU, thường kí hiệu là A (Accumulator).

b. Thanh ghi phụ B

Là thanh ghi tính toán phụ của vi điều khiển 8051, ở địa chỉ F0H được dùng chung với thanh ghi chính A trong các phép toán nhân, chia. Thanh ghi B cũng được sử dụng như một thanh ghi trung gian

c. Thanh ghi cổng P0 - P3

Các port xuất/nhập của 8051 bao gồm Port 0 tại địa chỉ 80H, Port 1 tại địa chỉ 90H, Port 2 tại địa chỉ A0H và Port 3 tại địa chỉ B0H. Tất cả các port này đều có thể truy suất theo bit hoặc theo byte.



Hình 1.8: Mô tả chức năng xuất/nhập của thanh ghi cổng

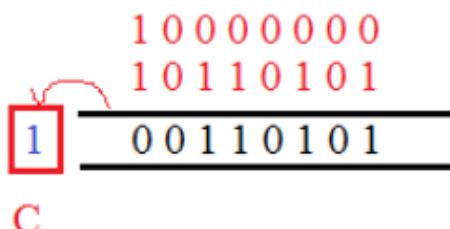
d. Thanh ghi trạng thái chương trình PSW

Thanh ghi trạng thái chương trình PSW (địa chỉ: D0H) là thanh ghi mô tả toàn bộ trạng thái chương trình đang hoạt động của hệ thống.

PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
CY	AC	F0	RS1	RS0	OV	-	P

Bảng 1.4: Thanh ghi trạng thái chương trình PSW

* **Cờ nhớ CY (Carry Flag) - địa chỉ D7h:** cờ nhớ có tác dụng kép (dùng trong các phép toán số học và logic). Thông thường nó được dùng cho các lệnh toán học: Cờ C được Set lên 1 nếu phép cộng có Bit nhớ từ Bit 7 (bị tràn) hoặc phép trừ có Bit mượn cho Bit 7. Ngược lại, cờ nhớ C = 0 nếu phép cộng không có nhớ và phép trừ không có mượn. Để dễ hình dung, ta xem ví dụ ở hình dưới đây



Hình 1.9: ví dụ về cờ nhớ CY

* **Cờ nhớ phụ AC (Auxiliary Carry Flag) - địa chỉ D6h:** khi cộng những giá trị BCD, cờ nhớ phụ được Set lên 1 nếu có Bit nhớ từ Bit 3 (bit thứ 4) sang Bit 4 hoặc kết quả trong 4 Bit thấp nằm trong khoảng 0AH→0FH.

* **Cờ 0 (Flag 0) - địa chỉ D5h:** cờ 0 là một bit cờ đa dụng dùng cho các ứng dụng của người dùng

* **Bit chọn băng thanh ghi RS1 và RS0 - địa chỉ D4h & D3h:** hai bit này được dùng để chọn băng thanh ghi R0, R1, R2 hay R3. Chúng được xóa về 0 khi chip bị reset và được thiết lập mức 1 hay 0 bằng phần mềm. Tùy theo RS1, RS0 có giá trị bằng 00,01,10 hay 11 sẽ chọn được băng thanh ghi tương ứng Băng 0, Băng 1, Băng 2, Băng 3

RS1	RS0	Register Bank
0	0	0
0	1	1
1	0	2
1	1	3

* **Cờ tràn OV (Overflow flag) - địa chỉ D2h:** cờ tràn được thiết lập (OV = 1) sau một hoạt động cộng hoặc trừ nếu có sự tràn toán học. - Khi các số có dấu được cộng hoặc trừ với nhau, ta có thể dùng cờ này để kiểm tra xem kết quả có nằm trong giới hạn xác định không (-127, +128). - Khi cộng hoặc trừ các số không dấu thì cờ này được bỏ qua.

* **Bit PWS.1 - địa chỉ D1h:** bit được dùng cho người dùng định nghĩa

* **Cờ chẵn lẻ P - địa chỉ D0h:** phản ánh số bit 1 trong thanh ghi A là chẵn hay lẻ. Nếu thanh ghi A chứa một số chẵn các bít 1 thì P = 0 còn chứa một số lẻ bit 1 thì P = 1.

e. Con trỏ ngăn xếp SP (Stack Point)

Ngăn xếp chính là vùng bộ nhớ RAM được CPU sử dụng để lưu thông tin tạm thời. Thông tin này có thể là dữ liệu hoặc địa chỉ. CPU cần các thanh ghi này vì số các thanh ghi bị hạn chế.

Như vậy, để có thể truy cập vào vùng nhớ ngăn xếp thì cần phải có thanh ghi trong CPU trỏ đến. Thanh ghi SP này sẽ được dùng để trỏ đến ngăn xếp, nên được gọi là thanh ghi con trỏ ngăn xếp. Thanh ghi này có độ rộng là 8 bit, tức là chỉ có thể trỏ được các địa chỉ từ 00h đến FFh.

f. Con trỏ dữ liệu DPTR (Data pointer)

Con trỏ dữ liệu được dùng để truy xuất bộ nhớ chương trình ngoài hoặc bộ nhớ dữ liệu ngoài. Con trỏ dữ liệu là một thanh ghi 16 bit ở địa chỉ 82H (DPL - byte thấp) và 83H (DPH - byte cao).

g. Thanh ghi bộ đệm truyền thông nối tiếp SBUF (Serial Data Buffer)

Bộ đệm truyền thông được chia thành hai bộ đệm, bộ đệm truyền dữ liệu và bộ đệm nhận dữ liệu. Khi dữ liệu được chuyển vào thanh ghi SBUF, dữ liệu sẽ được chuyển vào bộ đệm truyền dữ liệu và sẽ được lưu giữ ở đó cho đến khi quá trình truyền dữ liệu qua truyền thông nối tiếp kết thúc. Khi thực hiện việc chuyển dữ liệu từ SBUF ra ngoài, dữ liệu sẽ được lấy từ bộ đệm nhận dữ liệu của truyền thông nối tiếp.

h. Thanh ghi của bộ định thời/bộ đếm

8051 có 2 bộ đếm/định thời (counter/timer) 16 bit để định các khoảng thời gian hoặc để đếm các sự kiện. Các cặp thanh ghi (TH0, TL0) và (TH1, TL1) là các thanh ghi của bộ đếm thời gian. Bộ định thời 0 có địa chỉ 8AH (TL0, byte thấp) và 8CH (TH0, byte cao). Bộ định thời 1 có địa chỉ 8BH (TL1, byte thấp) và 8DH (TH1, byte cao). Hoạt động của bộ định thời được thiết lập bởi thanh ghi chế độ định thời TMOD (Timer Mode Register) ở địa chỉ 88H. Chỉ có TCON được định địa chỉ từng bit.

i. Thanh ghi ngắt (Interrupt register)

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).

Để thiết lập ngắt, ta sẽ thiết lập các giá trị trong thanh ghi cho phép ngắt IE ở địa chỉ A8H, thứ tự ưu tiên ngắt được đặt bằng cách set các bit ở thanh ghi ưu tiên ngắt IP ở địa chỉ B8h. Cả hai thanh ghi này được định địa chỉ theo bit

V. Ứng dụng và các đồ án đã có dựa trên vi điều khiển 8051

1. Ứng dụng

Quản lý năng lượng: Vi điều khiển 8051 được trang bị hệ thống đo lường hiệu quả và nó giúp vi điều khiển tiết kiệm năng lượng ở mức độ lớn.

Màn hình cảm ứng: những vi điều khiển hiện đại có tính năng màn hình cảm ứng và bộ vi điều khiển 8051 cũng đi kèm với tính năng màn hình cảm ứng. Do đó nó có một ứng dụng rộng rãi trong điện thoại di động, máy nghe nhạc và game.

Lĩnh vực ô tô: vi điều khiển 8051 có một ứng dụng rộng rãi trong lĩnh vực ô tô và đặc biệt là trong quản lý xe hybrid. Ngoài ra hệ thống kiểm soát hành trình và chống phanh là lĩnh vực khác mà nó có công dụng rất lớn.

2. Đồ án dựa trên vi điều khiển 8051

Giao tiếp dữ liệu không dây an toàn (at89s52)

Trình tạo số ngẫu nhiên sử dụng 8051

Hệ thống chấm công dựa trên RFID (at89s52 + rf)

Giao diện bàn phím Hex đến 8051,

Đồng hồ kỹ thuật số được điều khiển từ xa với DS1307 & AT89c2051

Hệ thống theo dõi năng lượng mặt trời (at89c2051)

Công cụ tìm phạm vi siêu âm sử dụng 8051

Hệ thống bảo mật dựa trên RFID (at89s52 + rfid)

Mạch cồn kẽ sử dụng 8051,

SMS qua điện thoại (at89s8252),

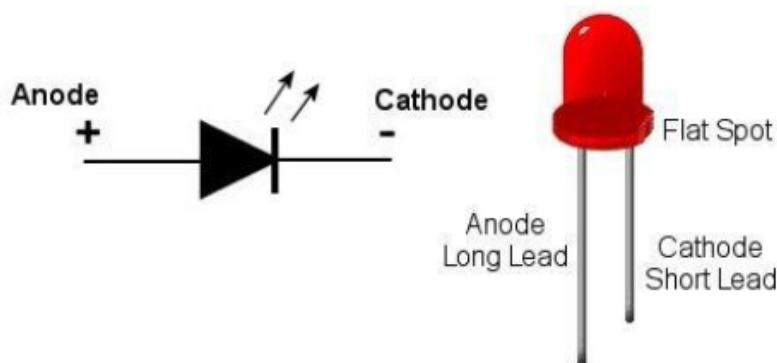
Robot dò đường sử dụng vi điều khiển 8051

Điều khiển từ xa dựa trên RF (at89c2051)

Đọc máy đo tự động dựa trên RF và nhiều đồ án khác nữa

LÝ THUYẾT

I. Giao tiếp với LED đơn



- + Chân anode nối với nguồn dương.
- + Chân cathode nối với nguồn âm.

- Cách tính trở hạn dòng:

$$R_{han\ dong} = \frac{U_{nguon} - U_{led}}{I_{qua\ led}}$$

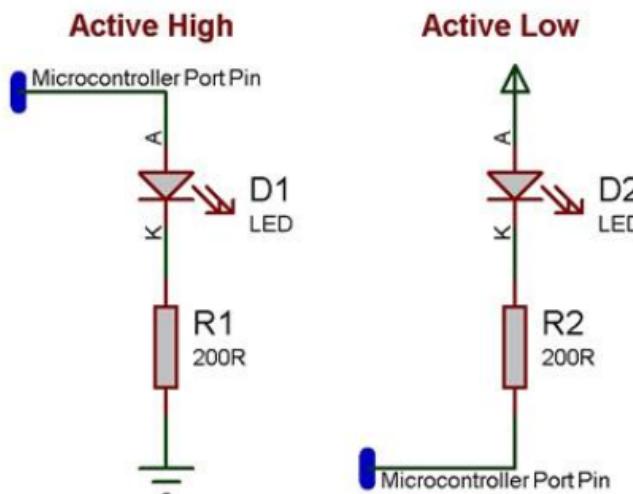
- + $R_{han\ dong}$: giá trị trở mắc nối tiếp với LED, hạn dòng cho LED.
- + U_{nguon} : giá trị điện áp nguồn cung cấp(cấp qua trở và qua LED).
- + U_{led} : giá trị điện áp rơi trên LED, nhà sản xuất sẽ cho thông số này, thường LED đơn tầm 2V đến 3.3V(led màu đỏ 2V).
- + $I_{qua\ led}$: dòng điện cho phép qua LED, nhà sản xuất cho thông số này, thường LED đơn tầm 15mA đến 30mA, dòng nhỏ hơn giá trị cho phép LED sẽ sáng mờ, vượt quá dòng điện cho phép LED sẽ chết.

Ví dụ: Nguồn cấp 5V, Led đỏ có $U=2V$, dòng $I=15mA=0.015A$. Tính giá trị trở hạn dòng,

$$\Rightarrow R_{han\ dong} = \frac{U_{nguon} - U_{led}}{I_{qua\ led}} = \frac{5-2}{0.015} = 200\Omega.$$

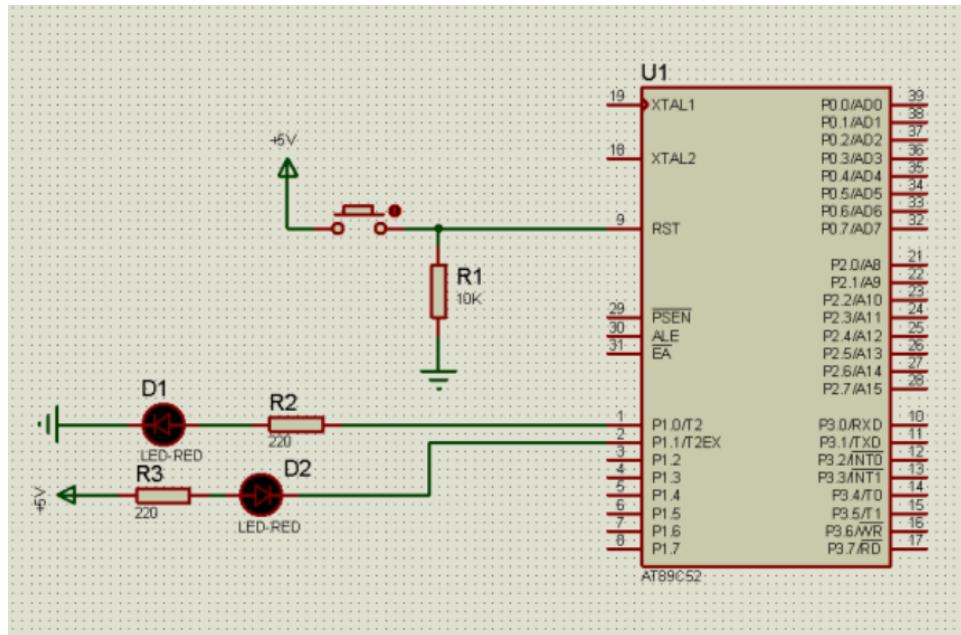
Giá trị trở nhỏ nhất phải dùng là 200Ω , nếu dùng trở nhỏ hơn LED thì tuổi thọ LED sẽ không cao, nếu dùng trở lớn hơn thì LED sẽ sáng mờ đi.

- Kết nối với MCU:

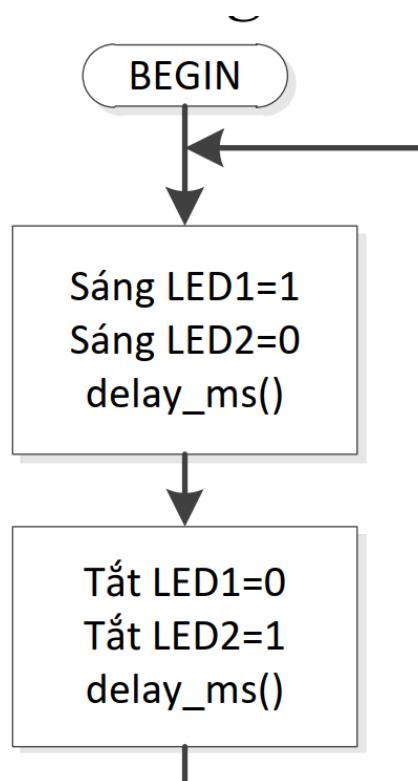


Kết nối trực tiếp với chân IO của MCU

1. Chớp tắt LED đơn



- Sơ đồ khối cho chớp tắt 2 LED đơn liên tiếp

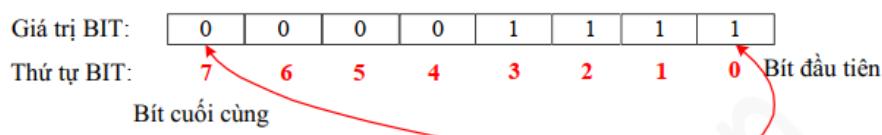


Bước 3- Viết Code

Dựa vào sơ đồ nguyên lý và lưu đồ giải thuật viết code

1- Giá trị BIT và vị trí PIN(chân) của một PORT

POR1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
------	------	------	------	------	------	------	------	------

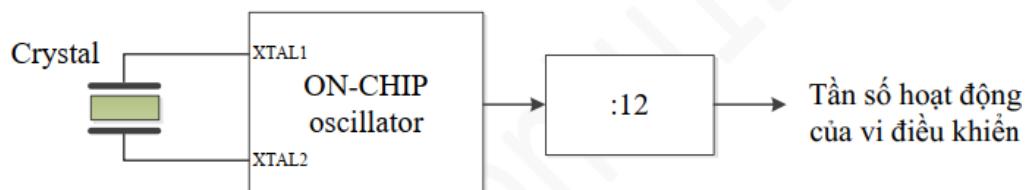


=> Ghi theo số nhị phân(chỉ có bit 0 và 1 hay gọi là cơ số 2): **PORT1=0b00001111**

=> Ghi theo số thập phân(Cơ số 10): **PORT1=15**

=> Ghi theo số thập lục phân(Cơ số 16): **PORT1=0x0F**

2- Tính và tạo hàm delay_us() và delay_ms()



- Tần số hoạt động của vi điều khiển=Fosc(tần số dao động thạch anh)/12
=> chu kỳ máy: $T_{osc} = 12/F_{osc}$.
- Nếu chọn tần số thạch anh là 12 MHz thì:
=> chu kỳ máy (Chu kỳ hoạt động của vi điều khiển) là $12/(12 \times 10^6 \text{ Hz}) = 1\text{us}$

```
/*=====
*Chuc nang: tao tre ms
*Tham so: Time la gia tri can tan tre, gia tri 16 bit
*Gia tri tra ve: Khong co
=====*/
void delay_ms(unsigned int Time)
{
    unsigned int i,j;
    for(i=0;i<Time;i++)
    {
        for(j=0;j<125;j++);
    }
}
/*=====
*Chuc nang: tao tre us
*Tham so: Time la gia tri can tan tre, gia tri 16 bit
*Gia tri tra ve: Khong co
=====*/
void delay_us(unsigned int Time)
{
    while((--Time)!=0);
}
```

```
#ifndef __MAIN_H
#define __MAIN_H
//=====Khai bao cac thu vien can su dung=====
#include <stdio.h>
#include <math.h>
#include <reg52.h>
//=====Khai bao cac thu vien nguoi dung viet=====
#include <delay.h>
#endif

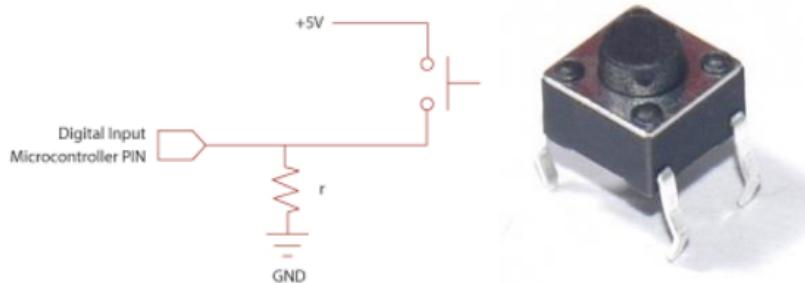
//=====khai bao cac thu vien can su dung=====
#include<main.h>
//=====khai bao cac chan vao ra=====
sbit LED1 = P1^0; //khai bao LED1 o chan P1.0
sbit LED2 = P1^1; //khai bao LED2 o chan P1.1

//=====khai bao bien va hang=====
//=====HAM MAIN=====
int main()
{
    while(1)//vong lap vo han
    {
        LED1=1; //goi muc 1 s?ng LED1
        LED2=1; //goi muc 0 s?ng LED2
        delay_ms(100); //delay de nhin thay LED sang

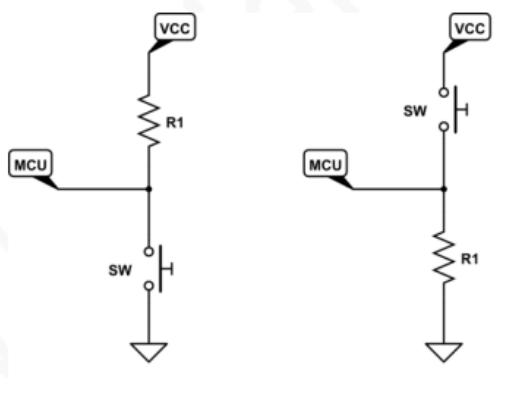
        LED1=0; //goi muc 0 t at LED1
        LED2=0;//goi muc 1 tat LED2
        delay_ms(100);//delay de nhin thay LED tat
    }
}
```

II. Giao tiếp nút nhấn và LED

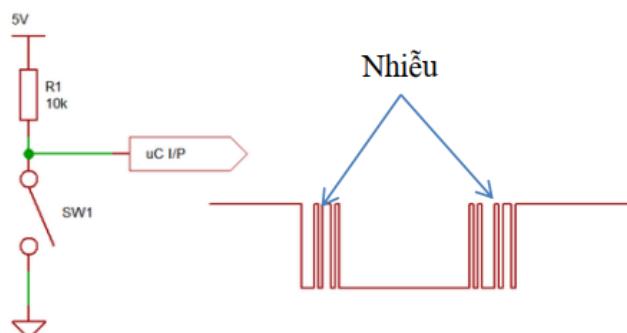
- Nếu bạn đã biết đến cái công tắc đóng / mở thì nút nhấn cũng hoạt động tương tự như vậy. Nút nhấn có 2 hoặc 4 chân, làm thành 1 cặp khi bạn nhấn nút các cặp đó sẽ được nối với nhau, cho phép dòng điện từ một chân bất kì có thể tới các chân còn lại.



- Button (nút nhấn) là 1 linh kiện cơ khí, sử dụng kim loại có tính đàn hồi cao để làm tiếp điểm, do đó khi các tiếp điểm tiếp xúc với nhau, sẽ xảy ra hiện tượng các tiếp điểm dao động trước khi ổn định.



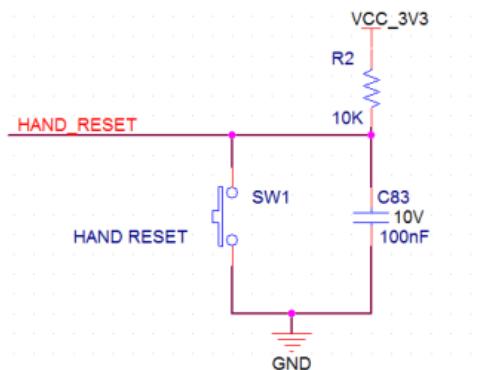
Có hai cách kết nối nút nhấn với MCU



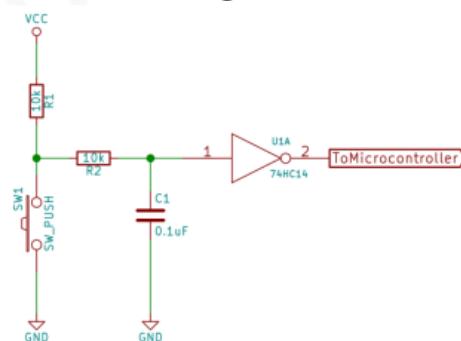
Nút nhấn bị nhiễu (dội phím)

- Cách khắc phục:

- + Dùng phần mềm: delay vài us hay ms để cho nút nhấn ổn định.
- + Dùng phần cứng:



Dùng tụ lọc

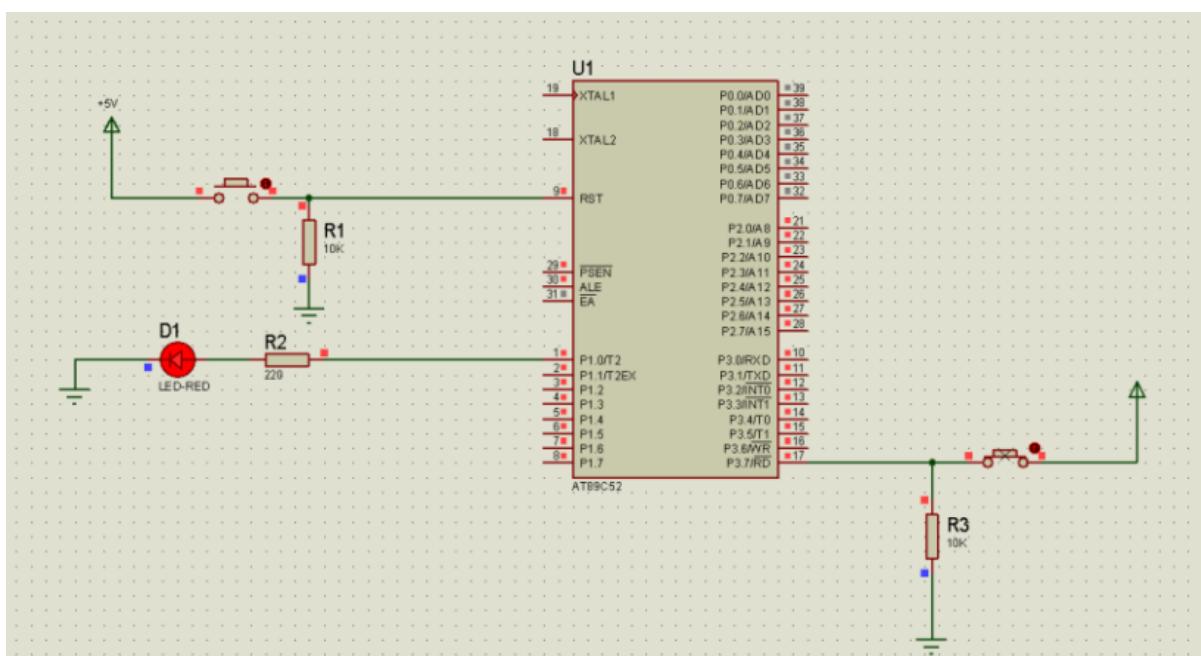


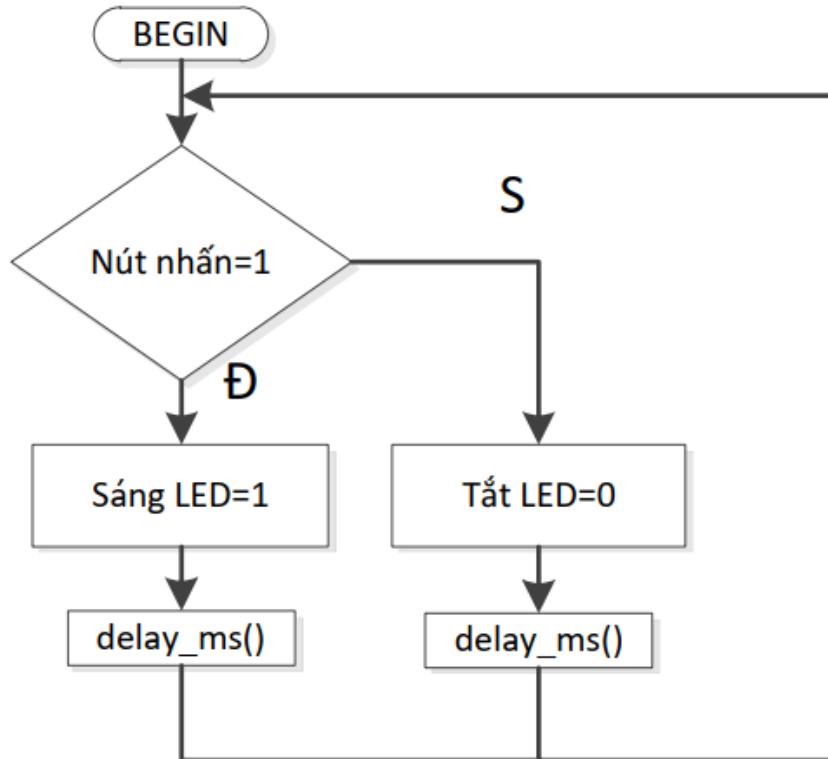
Dùng các cổng logic

Bài 2: Đọc giá trị nút nhấn ở chân P3.7 của PORT3 vào:

- Nếu là mức 1 thì sáng LED ở chân P1.0 của PORT1.
- Nếu là mức 0 thì tắt LED ở chân P1.0 của PORT1.

Chú ý: Đối với vi điều khiển 8051 thì không có lệnh quy định ngõ vào hay ngõ ra tại một chân hay một PORT như các dòng vi điều khiển khác. Nên khi đọc nút nhấn hay đọc tín hiệu (mức 0 hoặc mức 1) vào thì nó cứ đem chân đó hay PORT đó so sánh với giá trị nào đó và thực hiện lệnh.





```

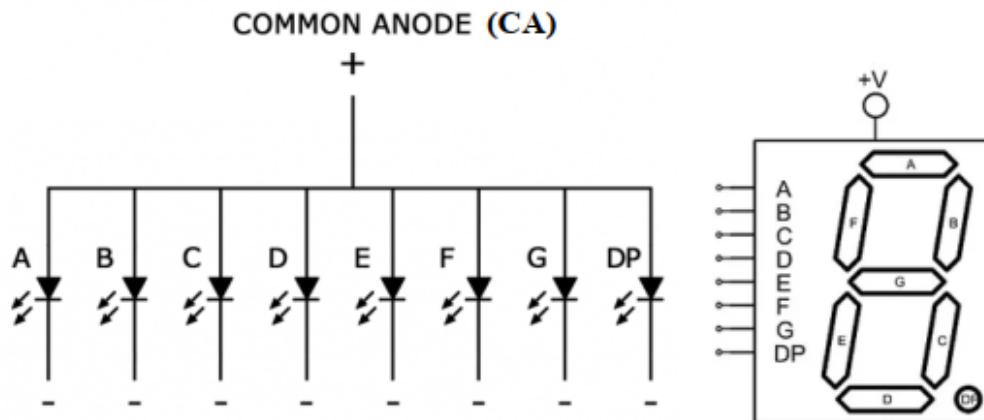
//=====khai bao cac thu vien can su dung=====
#include<main.h>
//=====dinh nghia cac chan vao ra=====
sbit LED1 = P1^0; //khai bao LED1 o chan P1.0
sbit BUT = P3^7; //khai bao nut nhan o chan P3.7
//=====khai bao bien va hang=====

//=====HAM MAIN=====
int main()
{
    BUT=0; //cho NUT NHAN =0, xem nhu ban dau la ngo vao=0
    while(1)
    {
        if(BUT==1)//co nhan nut nhan =1=>sang LED
        {
            LED1=1; //goi muc 1 sang LED1
            delay_ms(500); //delay de nhin thay LED sang
        }
        else //khong co nhan =>tat LED
        {
            LED1=0; //goi muc 0 tat LED1
        }
    }
}
  
```

III. Giao tiếp với LED 7 đoạn

1. LED Anode chung:

- Hình dạng và kiểu chân Led 7 đoạn Anode chung: Với Led Anode chung thì một chân cấp nguồn 5V (chân CA), các chân còn lại vi điều khiển sẽ xuất mức 0 làm cho Led sáng, xuất mức 1 cho tắt Led.



- Mã HEX xuất ra các chân của Led để Led sáng theo các số không sáng dấu chấm

	Số nghị phân								Số HEX
Thứ tự BIT	7	6	5	4	3	2	1	0	
Vị trí LED (cạnh)	DP	G	F	E	D	C	B	A	
Số 0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	0	0	0	0	90

- Mã HEX xuất ra các chân của Led để Led sáng theo các số sáng dấu chấm

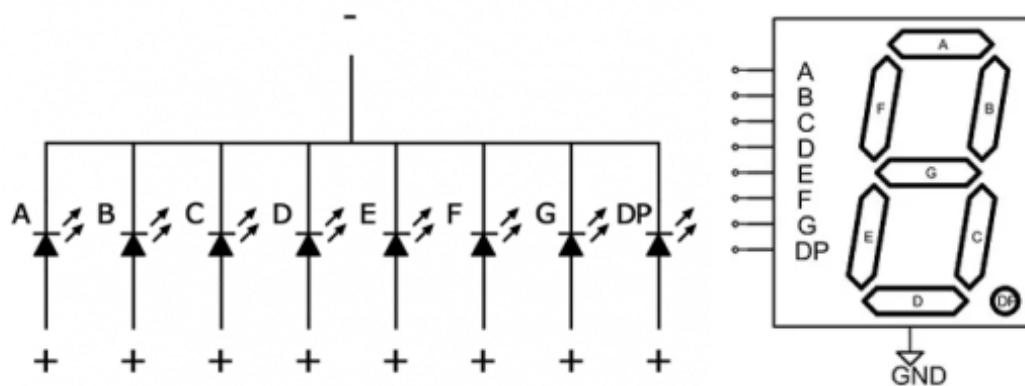
	Số nghị phân								Số HEX
Thứ tự BIT	7	6	5	4	3	2	1	0	
Vị trí LED (cạnh)	DP	G	F	E	D	C	B	A	
Số 0	0	1	0	0	0	0	0	0	40
1	0	1	1	1	1	0	0	1	79
2	0	0	1	0	0	1	0	0	24

3	0	0	1	1	0	0	0	0	30
4	0	0	0	1	1	0	0	1	19
5	0	0	0	1	0	0	1	0	12
6	0	0	0	0	0	0	1	0	02
7	0	1	1	1	1	0	0	0	78
8	0	0	0	0	0	0	0	0	00
9	0	0	0	1	0	0	0	0	10

2. LED Cathode chung:

- Hình dạng và kiểu chân Led 7 đoạn Cathode chung: Với Led Cathode chung thì một chân cấp nguồn GND (chân CA), các chân còn lại vi điều khiển sẽ xuất mức 1 làm cho Led sáng, xuất mức 0 cho tắt Led.

COMMON CATHODE (CA)

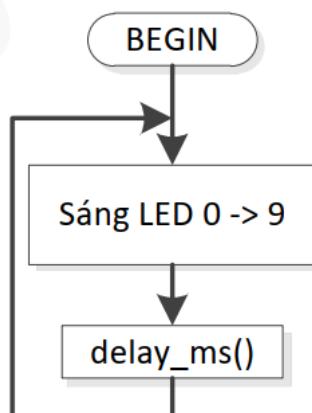
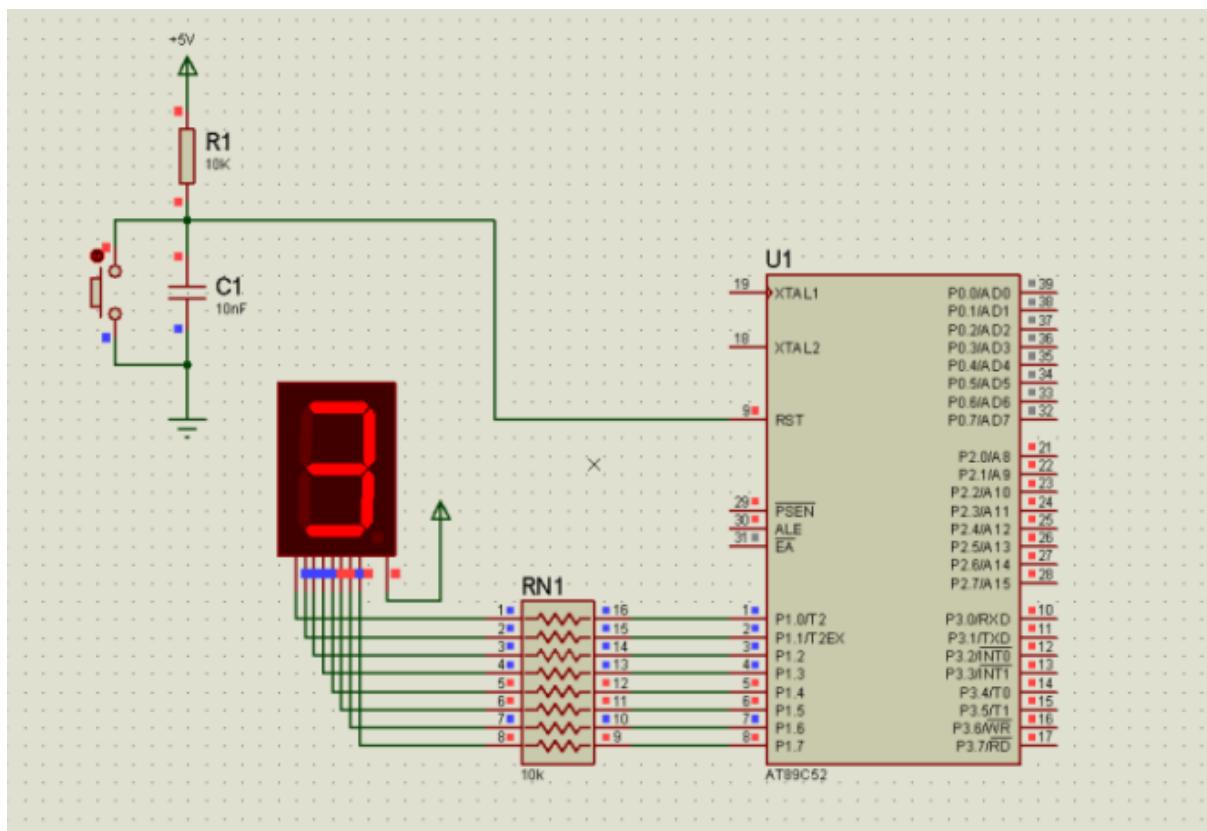


- Mã HEX xuất ra các chân của Led để Led sáng theo các số không sáng dấu chấm

Thứ tự BIT	Số nghị phân								Số HEX
	7	6	5	4	3	2	1	0	
Vị trí LED (cạnh)	DP	G	F	E	D	C	B	A	
Số 0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	1	1	1	1	6F

- Mã HEX xuất ra các chân của Led để Led sáng theo các số sáng dấu chấm

	Số nghị phân								Số HEX
Thứ tự BIT	7	6	5	4	3	2	1	0	
Vị trí LED (cạnh)	DP	G	F	E	D	C	B	A	
Số 0	1	0	1	1	1	1	1	1	BF
1	1	0	0	0	0	1	1	0	86
2	1	1	0	1	1	0	1	1	DB
3	1	1	0	0	1	1	1	1	CF
4	1	1	1	0	0	1	1	0	E6
5	1	1	1	0	1	1	0	1	ED
6	1	1	1	1	1	1	0	1	FD
7	1	0	0	0	0	1	1	1	A7
8	1	1	1	1	1	1	1	1	FF
9	1	1	1	0	1	1	1	1	EF



```
//=====khai bao cac thu vien can su dung=====//
#include<main.h>
//=====khai bao cac chan vao ra=====//

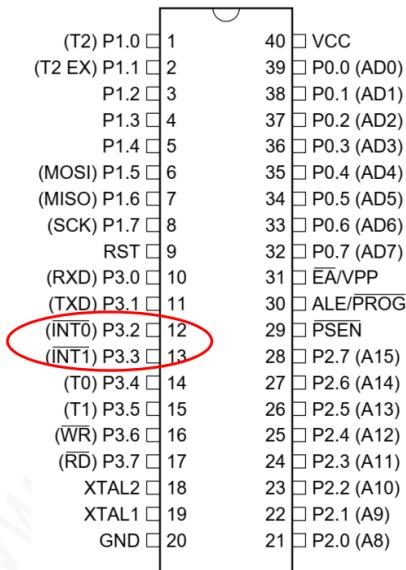
//=====khai bao bien va hang=====//
//khai bao ma sang LED 7 doan Anode chung
unsigned char seg[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8, 0x80, 0x90};
unsigned char i=0;
//=====HAM MAIN=====//
int main()
{
    P1=0xFF;      //ban dau tat het LED
    while(1)
    {
        // vong lap for chay den vi tri nao cuoi mang seg[] thi lay gia tri do ra gan cho PORTD1
        for(i=0;i<10;i++)//so dem tu 0->9 roi quay ve 0
        {
            P1=seg[i]; //xuat cac gia tri ra PORT1, tai vi tri i co trong mang seg[]
            delay_ms(300);    //delay nhin thay LED sang
        }
    }
}
```

IV. Ngắt ngoài

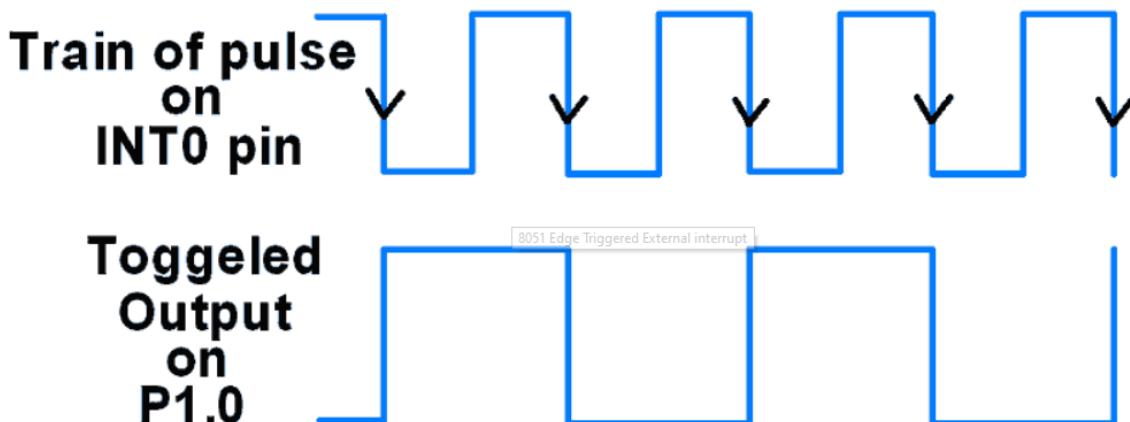
1. Ngắt là gì?

- AT89C52 luôn thực hiện một chương trình gọi là chương trình chính, khi có tác động từ bên ngoài phần cứng hay từ bên trong làm cho AT89C52 ngừng thực hiện chương trình chính mà nhảy vào thực hiện một chương trình khác, sau khi thực hiện xong chương trình đó thì quay lại chương trình chính. Quá trình gián đoạn chương trình chính nhảy vào chương trình khác gọi là ngắt.

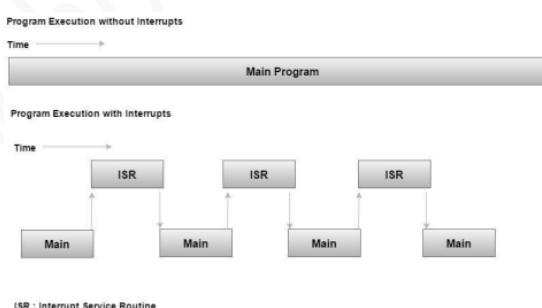
- Trong AT89C52 có 2 ngắt vào ngắt ngoài tại chân INT0 (P3.2) và INT1(P3.3). Khi có một sự kiện trên chân INT (cạnh xuống hoặc mức 0) sẽ sinh ra một ngắt (nếu được cho phép).



- Trong AT89C52 cho phép ngắt cạnh xuống hoặc ngắt mức 0:



- Sơ đồ minh họa cho ngắt ngoài INT:



+ Chú thích: Khi đang hiện trong main có ISR (ngắt) thì nhảy vào chương trình phục vụ ngắt, sau khi thực hiện xong nhảy ra main tiếp, chu trình cứ tiếp tục như vậy.

+ Chú ý: vì chương trình main là chương trình chính, còn ISR là chương trình phục vụ ngắt (nhất thời), do đó khi viết code trong chương trình phục vụ ngắt nên ngắn gọn, tránh delay quá lâu làm ảnh hưởng đến main.

2. Các thanh ghi liên quan đến ngắt ngoài:

2.1. Thanh ghi điều khiển ngắt IE:

D7	EA	--	ET2	ES	ET1	EX1	ET0	EX0	D0
----	----	----	-----	----	-----	-----	-----	-----	----

- Bit EX0: cho phép ngắt ngoài INT0 (=1 cho phép ngắt INT0,=0 cấm ngắt INT0)
- Bit ET0: cho phép ngắt TIMER0 (=1 cho phép ngắt TIMER0,=0 cấm ngắt TIMER0)
- Bit EX1: cho phép ngắt ngoài INT1 (=1 cho phép ngắt INT1,=0 cấm ngắt INT1)
- Bit ET1: cho phép ngắt TIMER1 (=1 cho phép ngắt TIMER1,=0 cấm ngắt TIMER1)
- Bit ES: cho phép ngắt truyền thông nối tiếp.
- Bit ET2: cho phép ngắt TIMER2 (=1 cho phép ngắt TIMER2,=0 cấm ngắt TIMER2)
- Bit EA: cho phép ngắt toàn cục.

2.1. Thanh ghi IP: thanh ghi chọn mức ưu tiên ngắt

---	---	PT2	PS	PT1	PX1	PT0	PX0
-----	-----	-----	----	-----	-----	-----	-----

- Bit: PX0: Ưu tiên ngắt INT0.
- Bit: PT0: Ưu tiên ngắt TIMER0
- Bit: PX1: Ưu tiên ngắt INT1.
- Bit: PT1: Ưu tiên ngắt TIMER1
- Bit: PS: Ưu tiên ngắt truyền thông nối tiếp
- Bit: PT1: Ưu tiên ngắt TIMER2

2.3. Thanh ghi TCON: thanh ghi thiếp lặp trạng thái ngắt

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

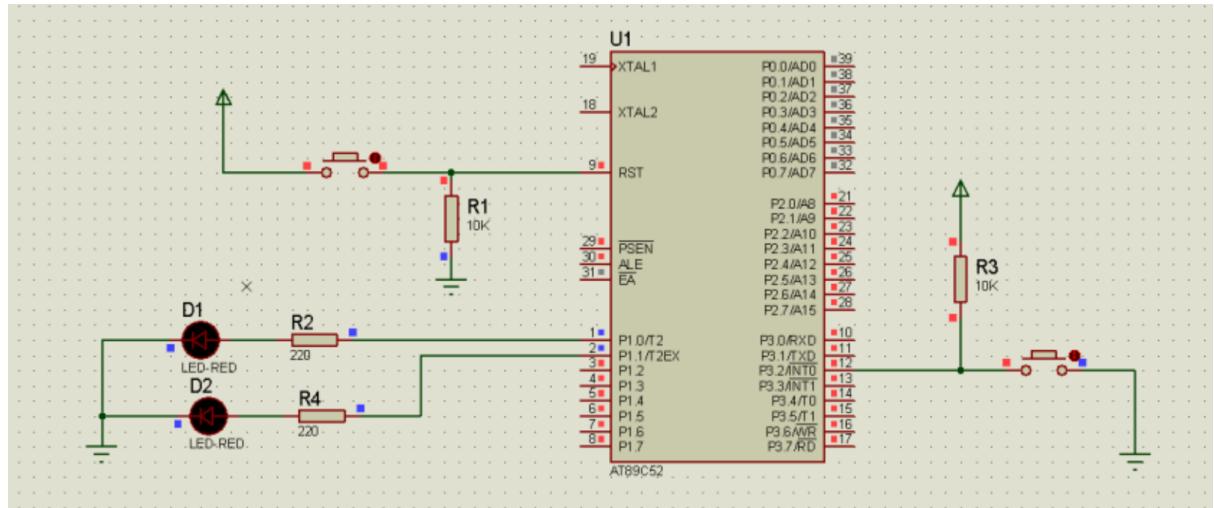
- Bit TF1: Cờ báo tràn bộ T/C1.
- Bit TR1: Bit điều khiển khởi động hoặc dừng bộ T/C1.
- Bit TF0: Cờ báo tràn bộ T/C0.
- Bit TR0: Bit điều khiển khởi động hoặc dừng bộ T/C0.
- Bit IE1: IE1=1 chọn ngắt INT1.
- Bit IT1: + IT1=1 chọn ngắt INT1 cạnh xuống
+ IT1=0 chọn ngắt INT1 mức 0.
- Bit IE0: IE0=1 chọn ngắt INT0.
- Bit IT0: + IT0=1 chọn ngắt INT0 cạnh xuống
+ IT0=0 chọn ngắt INT0 mức 0.

3. Các vector ngắt

Table 11–4 8051/52 Interrupt Numbers in C

Interrupt	Name	Numbers used by 8051 C
External Interrupt 0	(INT0)	0
Timer Interrupt 0	(TF0)	1
External Interrupt 1	(INT1)	2
Timer Interrupt 1	(TF1)	3
Serial Communication	(RI + TI)	4
Timer 2 (8052 only)	(TF2)	5

Ví dụ: Ngắt INT0 chớp tắt LED, main.c chớp tắt LED1, nếu có ngắt đảo trạng thái LED2

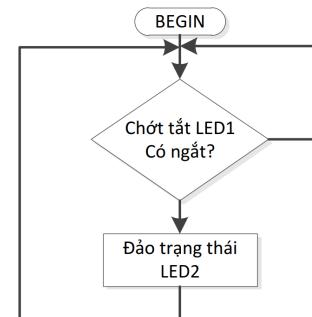


```

//=====khai bao cac thu vien can su dung=====
#include<main.h>
//=====dinh nghia cac chan vao ra=====
sbit LED1 = P1^0; //khai bao LED1 o chan P1.0
sbit LED2 = P1^1; //khai bao LED1 o chan P1.0
//=====khai bao bien va hang=====

//=====HAM MAIN=====
int main()
{
    EA = 1;           //cho phep ngat toan cuc
    EX0 = 1;          //chon ngat ngoai INT0
    IT0 = 1;          //chon ngat canh xuong
    LED2=0;           //ban dau cho LED2 tat
    while(1)
    {
        LED1=1;
        delay_ms(200);
        LED1=0;
        delay_ms(200);
    }
/*
 *Chuc nang: phuc vu ngat INT0
 */
void External0_ISR() interrupt 0
{
    LED2 = ~LED2;    //dao trang thai LED2
}

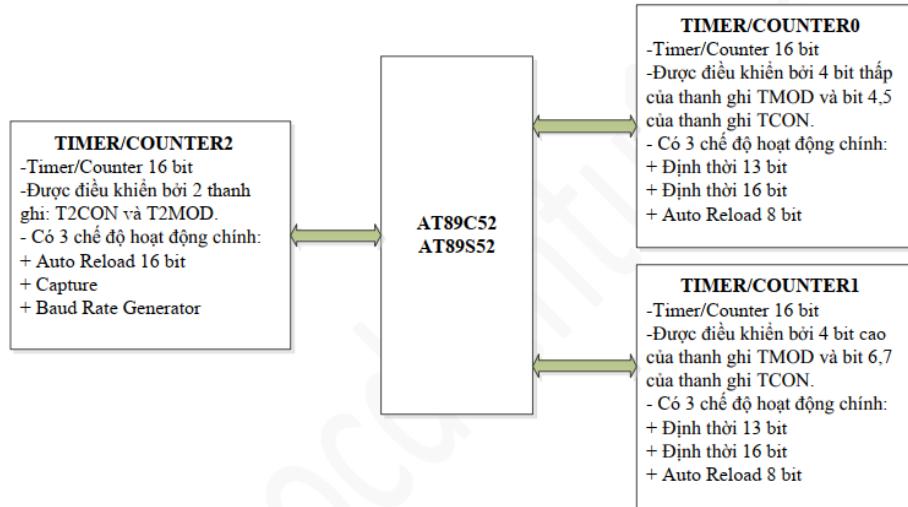
```



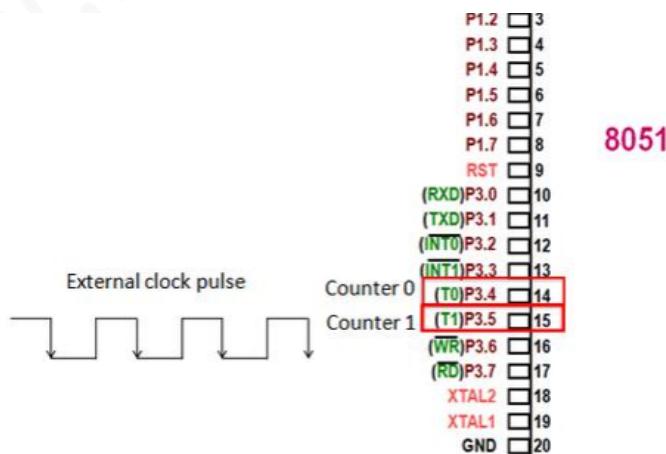
V. Timer

1. Giới thiệu

- Bộ Timer/Counter trong 8051 có thể được sử dụng trong rất nhiều ứng dụng như Counter (đếm sự kiện cạnh lên hay cạnh xuống) trên chân T0(P3.4), T1(P3.5), T2(P1.0), Capture (đếm sự kiện tính ra tần số của xung) trên chân T2EX (P1.1), sử dụng tạo hàm delay với độ chính xác cao, sử dụng tạo tín hiệu PWM, sử dụng ngắt timer...
- Trong AT89C52 hoặc AT89S52 có 3 bộ TIMER/COUNTER: Timer/Counter 0 và Timer/Counter1, Timer/Counter 2.



Các Timer/Counter trong AT89C52

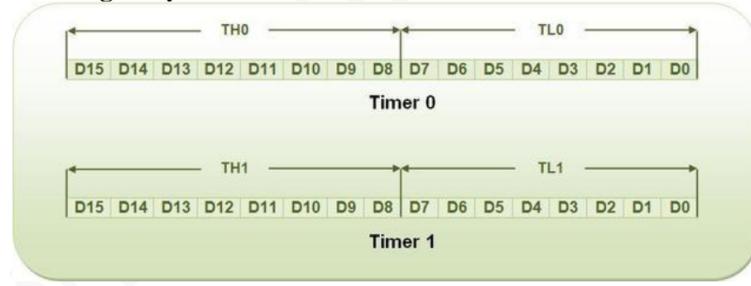


Chân Counter của Timer0(T0) và Timer1(T1)

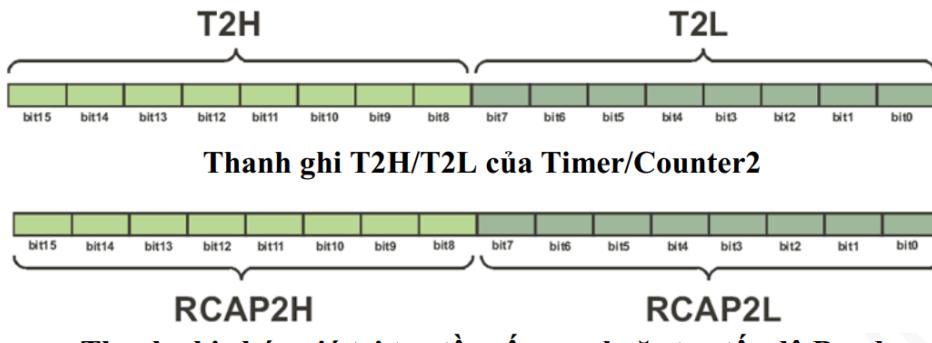
Existing	Alternate	Function
P1.0	T2	Timer/counter 2 External Count input
P1.1	T2 EX	Timer/counter 2 Trigger input

Chân Counter của Timer0(T0) và Timer1(T1)

2. Thanh ghi chứa giá trị đếm Timer/Counter



Thanh ghi TH0/TL0 của Timer/Counter0 và thanh ghi TH1/TL1 của Timer/Counter1



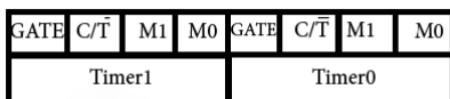
Thanh ghi chứa giá trị tạo tần số xung hoặc tạo tốc độ Baud

3. Thanh ghi điều khiển và trạng thái TCON(dùng cho TIMER0 và TIMER1)



- **TF1:** Cờ báo tràn bộ T1/C1 (Timer1/Counter1)
- **TR1:** Bit điều khiển khởi động hoặc dừng bộ T1/C1.
- **TF0:** Cờ báo tràn bộ T1/C0.
- **TR0:** Bit điều khiển khởi động hoặc dừng bộ T1/C0.
- 4 bit thấp (IE1, IT1, IE0, IT0) phục vụ ngắt ngoài.

4. Thanh ghi cài đặt chế độ hoạt động TMOD (dùng cho TIMER0 và TIMER1)



- GATE:

+ GATE = 1: Bộ T/C được cho phép hoạt động khi chân INTx ở mức cao và bit TRx trong thanh ghi TCON được set lên 1. (x: 0 hoặc 1).

+ GATE = 0: Bộ T/C được cho phép hoạt động khi bit TRx trong thanh ghi TCON được set lên 1.

- CT:

+ CT = 1: Chức năng Counter được chọn.

+ CT = 0: Chức năng Timer được chọn.

- M1, M0: 2 bit chọn các chế độ hoạt động của T/C.

M1	M0	Mode	Mô tả
0	0	0	Chế độ định thời 13 bit, 8 bit THx, 5 bit TLx
0	1	1	Chế độ định thời 16 bit.
1	0	2	Chế độ 8 bit tự nạp lại.
1	1	3	Chế độ bộ định thời

7. Các bước thiết lập Timer/Counter

Các bước cấu hình cho bộ T/C0 hoạt động cở chế độ timer (định thời) như sau.

- **B1:** Ghi vào thanh ghi TMOD để chọn chế độ timer, chọn mode: Ví dụ ở đây mình chọn chế độ timer (C/T = 0) và mode 16 bit (M1=0, M0=1), mình sẽ ghi giá trị 0x01 vào TMOD (TMOD = 0x01).
- **B2:** Ghi vào thanh ghi TL, TH các giá trị để tạo ra giá trị định thời mong muốn.
- **B3:** Set bit TR để cho phép bộ T/C hoạt động.
- **B4:** Kiểm tra cờ TF để biết bộ T/C đã tràn hay chưa.
- **B5:** Khởi tạo lại giá trị cho các thanh ghi TH,TL để chuẩn bị cho lần chạy tiếp theo.

8. Tính giá trị nạp cho thanh ghi TH và TL

1 chu kỳ của 8051 bằng 12 lần chu kỳ thạch.

- Nếu tần số thạch anh 12 MHz => chu kỳ thạch anh $T_{osc} = 1/12 = 0.0833$ (us)

=> **Chu kỳ máy: $T = T_{osc} * 12 = 0.0833 * 12 = 1$ us.**

- Nếu tần số thạch anh 11.0592 MHz => chu kỳ thạch anh $T_{osc} = 1/11.0592 = 0.0904$ (us)

=> **Chu kỳ máy: $T = T_{osc} * 12 = 0.0904 * 12 = 1.085$ us.**

Muốn tạo thời gian $T_c = 100$ us, dùng thạch anh 11.0592Mhz:

Ta có số lần đếm:

$$\text{Count} = (T_{osc} \times 10^{-6}) / (T_c \times 10^{-6}) \text{ (s)} = (100 \times 10^{-6}) / (1.085 \times 10^{-6}) \approx 92$$

=> Giá trị cần nạp cho thanh ghi TH/TL: $\text{Value} = (2^n - \text{Count}) + 1$

+ Nếu $n = 16$ (16 bit) $\text{Value} = (65535 - \text{Count}) + 1 = 65444 = (\text{FFA4})\text{Hex}$

=> $\text{TH0} = 0xFF$ & $\text{TL0} = 0xA4$

+ Nếu $n = 13$ (13 bit) $\text{Value} = (8191 - \text{Count}) + 1 = 7271 = (\text{1FA4})\text{Hex}$

=> $\text{TH0} = 0x1F$ & $\text{TL0} = 0xA4$

+ Nếu $n = 8$ (8 bit) $\text{Value} = (255 - \text{Count}) + 1 = 7271 = (\text{A4})\text{Hex}$

=> $\text{TH0} = 0xA4$, $\text{TL0} = 0xA4$

Về các biến ngắn và giá trị ngắn, xem lại phần trên.

Bài 2- Sử dụng ngắt TIMER0 (ngắt tràn TIMER0) chớp tắt LED ở chân P1.0 thời gian chớp tắt 50ms.

Tính giá trị nạp cho thanh ghi TH/TL

Tần số thạch anh 11.0592MHz => Tosc=1/11.0592 => T=12*Tosc=1.085us (T chu kỳ máy).

Muốn tạo thời gian Tc=50ms=50.000us:

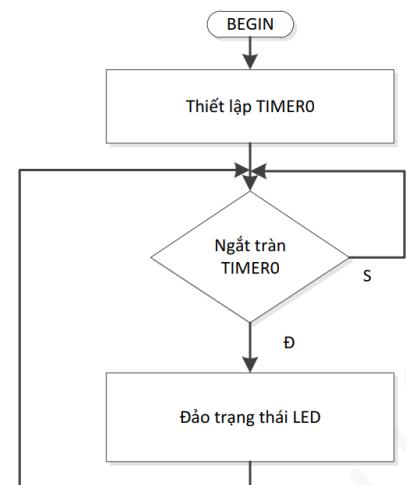
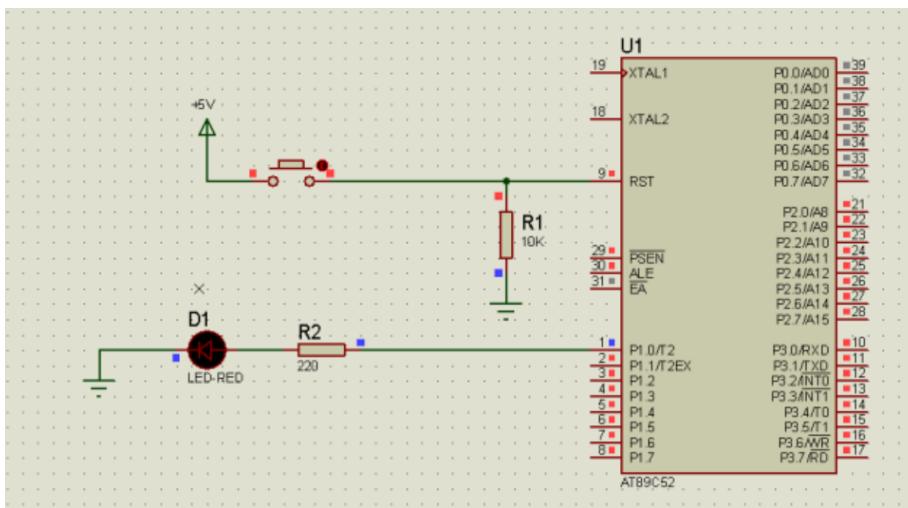
Ta có số lần đếm:

$$\text{Count} = (\text{Tc} \times 10^{-6}) / (\text{T} \times 10^{-6}) \text{ (s)} = (50000 \times 10^{-6}) / (1.085 \times 10^{-6}) \approx 46083$$

=> Giá trị cần nạp cho thanh ghi TH/TL:

$$\text{Value} = (65535 - \text{Count}) + 1 = 65535 - 46083 + 1 = 19453 = (4BFD)\text{Hex}$$

=> TH0 = 0x4B & TL0 = 0xFD



```

//=====dinh nghia cac ham=====
void Timer_init();
//=====dinh nghia cac chan vao ra=====
sbit LED1 = P1^0; //khai bao LED1 o chan P1.0
//=====khai bao bien va hang=====
unsigned char dem=0;
//=====HAM MAIN=====
int main()
{
    EA = 1;           //cho phet ngat toan cuc
    ET0 = 1;          //cho phep ngat tran Timer0
    Timer_init();
    while(1)
    {
        if(dem==6)//moi lan ngat la 50ms=>6 lan ngat lì 300ms
        {
            //sau 4 ln ngat thi dao trang thao LED
            LED1 = ~LED1;      //dao trang thai LED
            dem=0;
        }
    }
}

```

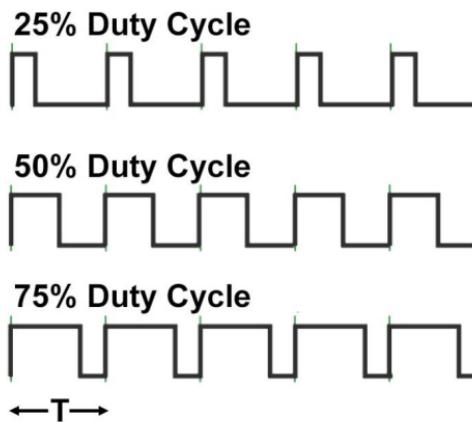
```
void Timer_init()
{
    TMOD = 0x01;      //chon Timer0, che do dinh thoi 16 bit
    TH0 = 0x4B;       //tao thoi gian tre 50ms nhay vao ngat
    TL0 = 0xFD;
    TR0 = 1;         //cho Timer 0 hoat dong
}
/*=====
 *chuc nang: phuc vu ngat. Timer0 dem tran nhay vao phuc vu ngat
 *tham bien: khong
 *tra ve: khong
=====
 */
void Timer0_ISR() interrupt 1 /* Timer0 interrupt service routine (ISR) */
{
    dem=dem+1;          //tao thoi gian chup tat led
    TH0 = 0x4B;         //nap gia tri cho TH0 va TL0 de tao ra dem 50ms tren Timer0
    TL0 = 0xFD;
}
```

VI. PWM (Pulse with Modulation)

1. PWM là gì?

- Phương pháp điều xung **PWM (Pulse Width Modulation)** là phương pháp điều chỉnh điện áp ra tải, hay nói cách khác, là phương pháp điều chế dựa trên sự thay đổi độ rộng của chuỗi xung vuông, dẫn đến sự thay đổi điện áp ra.

- Đây là phương pháp điều chỉnh điện áp ra tải dựa vào trung bình tín hiệu điều chế. Khi độ rộng xung tăng, trung bình điện áp ra tăng. Các module PWM thường sử dụng tần số điều chế không đổi, và điều chỉnh dựa trên sự thay đổi của chu kỳ nhiệm vụ (duty cycle).



2. PWM trong AT89C52

2.1. Cách tạo PWM trong AT89C52

- Sử dụng Timer0/1/2 để tạo ra tần số và thay đổi độ rộng xung cho PWM.
- Xuất PWM ở ngõ ra bất kỳ ở các PORT của AT89C52.

2.2. Các thành phần của PWM

- Duty cycle : tỷ lệ phần trăm xung ở mức cao.
- Period : là chu kì của xung(là tổng thời gian mức cao + mức thấp).
- Pulse width: là giá trị của mức cao so với period.
- Biên độ xung: là giá trị điện áp của xung khi ở mức cao.

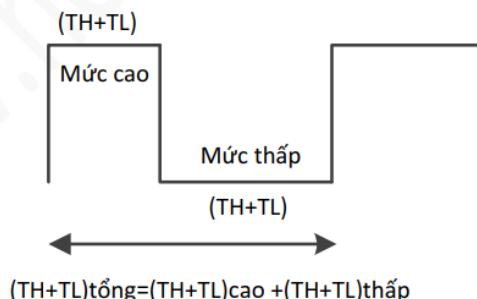
2.3. Các bước tạo PWM:

- Thiết lập Timer và chế độ sử dụng Timer (ở chế độ 16 bit)
- Sử dụng TIMER thiết lập tần số (chu kỳ), tính giá trị nạp cho mức cao và mức thấp.

$$\text{PWM Frequency} = 1(\text{MHz}) / (\text{PWM_Freq_Num} * 255) = 1000000 / (\text{PWM_Freq_Num} * 255) (\text{Hz})$$

Với PWM_Freq_Num có giá trị 8 bit

=> PWM_Freq_Num * 255 nhỏ nhất=255 và lớn nhất=65355 để nạp vào thanh ghi TH và TL.



Do đó ta cần xác định tần số sau đó nạp vào thanh ghi TH và TL, muốn thay đổi độ rộng xung ta chỉ cần thay đổi giá trị TH/TL cao và TH/TL thấp .

- Viết chương trình phục vụ ngắt TIMER thay đổi độ rộng xung.
- Xuất PWM ra chân đã chọn.

Bài 2- Sử dụng TIMER0 16 bit tạo xung PWM tần số 1KHz.

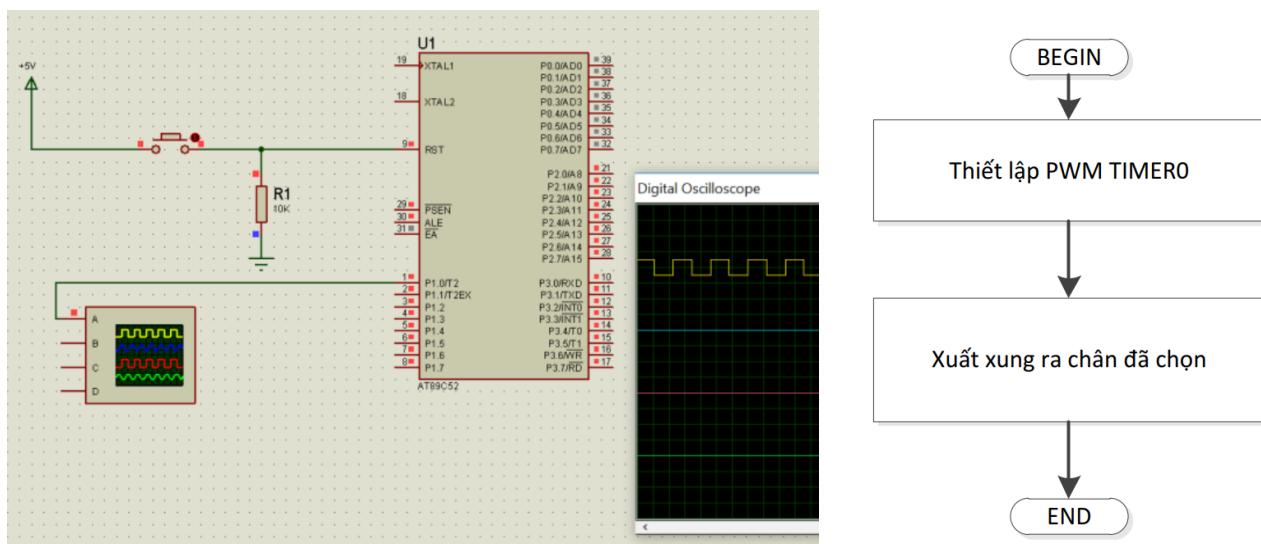
Các bước thiết lập PWM:

- Thiết lập Timer0.
- Tính chu kỳ của PWM.

$$\text{PWM Frequency} = \frac{1(\text{MHz})}{(\text{PWM_Freq_Num} * 255)} \\ = \frac{1000000}{(\text{PWM_Freq_Num} * 255)} (\text{Hz})$$

$$\Rightarrow \text{PWM_Freq_Num} = 1000000 / (1000 * 255) \approx 4;$$

- Nạp giá trị cho thanh ghi TH và TL.



```

//=====khai bao cac thu vien can su dung=====
#include<main.h>
//=====dinh nghia cac ham=====
void PWM_init();
//=====dinh nghia cac chan vao ra=====
sbit PWM_out = P1^0; //khai bao PWM_out o chan P1.0
//=====khai bao bien va hang=====
unsigned char dem=0;
unsigned char PWM = 0; //=0 (0% duty cycle) ,=255 (100% duty cycle)
unsigned int temp=0; // bien nap cho muc cao muc thap
unsigned int PWM_Freq_Num=4;// bien thay doi tan so PWM

/*PWM Frequency = 1(MHz)/(PWM_Freq_Num*255)= 1000000/(PWM_Freq_Num*255)(Hz)
voi PWM Frequency=1khz=1.000hz
=> PWM_Freq_Num=1000000/(1000*255) ~=4;
*/

```

```

//=====HAM MAIN=====
int main()
{
    PWM_out=0;
    EA = 1;          //cho phet ngat toan cuc
    ET0 = 1;          //cho phep ngat tran Timer0
    PWM_init();        //thiet lap PWM
    PWM=127;           //cho duty=50%
    while(1)
    {
    }
}

/*
 *chuc nang: khai tao PWM su dung TIMER0
 *tham bien: khong
 *tra ve: khong
 */
void PWM_init()
{
    PWM=0;
    TMOD = 0x01;      //chon Timer0, che do dinh tho 16 bit
    TH0 = 0x00;
    TL0 = 0x00;
    TR0 = 1;          //cho Timer 0 hoat dong
}

/*
 *chuc nang: phuc vu ngat. Timer0 dem tran nhay vao phuc vu ngat
 *tham bien: khong
 *tra ve: khong
 * PWM la gia tri thay doi do rong xung. PWM min=0, PWM max=255
 *  PWM_Freq_Num la gia tri thay doi tan so :PWM_Freq_Num min=1,PWM_Freq_Num max=257
 * bien temp nap chp TH va TL => temp=temp_xung cao+temp_xung thap
 */

void Timer0_ISR() interrupt 1 /* Timer0 interrupt service routine (ISR) */
{
    TR0 = 0;          //cho Timer0 ngung hoat dong
    if(PWM_out==1) //if PWM_Pin is high
    {
        PWM_out = 0;
        temp = (255-PWM)*PWM_Freq_Num;      //gia tri nap cho do rong xung muc thap
        TH0 = 0xFF - (temp>>8)&0xFF;        //nap vao thanh ghi TH0 (8 bit cao trong bien temp)
        TL0 = 0xFF - temp&0xFF;              //nap vao thanh ghi TL0 (8 bit thap trong bien temp)
    }
    else //if PWM_Pin is low
    {
        PWM_out = 1;
        temp = PWM*PWM_Freq_Num;            //gia tri nap cho do rong xung muc cao
        TH0 = 0xFF - (temp>>8)&0xFF;        //nap vao thanh ghi TH0 (8 bit cao trong bien temp)
        TL0 = 0xFF - temp&0xFF;              //nap vao thanh ghi TL0 (8 bit thap trong bien temp)
    }
    TF0 = 0;          //xa co ngat
    TR0 = 1;          //cho Timer0 dem lai tu dau
}

```

VII. UART (Universal Asynchronous Receiver and Transmitter)

1. Giới thiệu

Vi điều khiển AT89S52 được tích hợp một bộ giao tiếp nối tiếp UART và giao tiếp qua hai Chân P3.0 (RX- chân thu dữ liệu), P3.1 (TX- chân phát dữ liệu). Chức năng cơ bản của cổng nối tiếp là chuyển dữ liệu song song sang nối tiếp rồi truyền đi, và chuyển dữ liệu nối tiếp sang song song khi nhận. Cổng nối tiếp dùng để giao tiếp với các thiết bị ngoại vi khác có trang bị chuẩn giao tiếp này như : máy tính để bàn thông qua cổng COM, giao tiếp với một vi điều khiển khác,...

- Truyền dữ liệu bắt đầu bô UART thường dùng để truyền nhận tín hiệu giữa máy tính với AT89C52 hoặc với các thiết bị ngoại vi có giao tiếp UART .

- Các thông số cơ bản trong truyền nhận UART:

+ **Baud rate** (tốc độ baud): Khoảng thời gian dành cho 1 bit được truyền, phải được cài đặt giống nhau ở gửi và nhận.

+ **Frame** (khung truyền): Khung truyền quy định về số bit trong mỗi lần truyền.

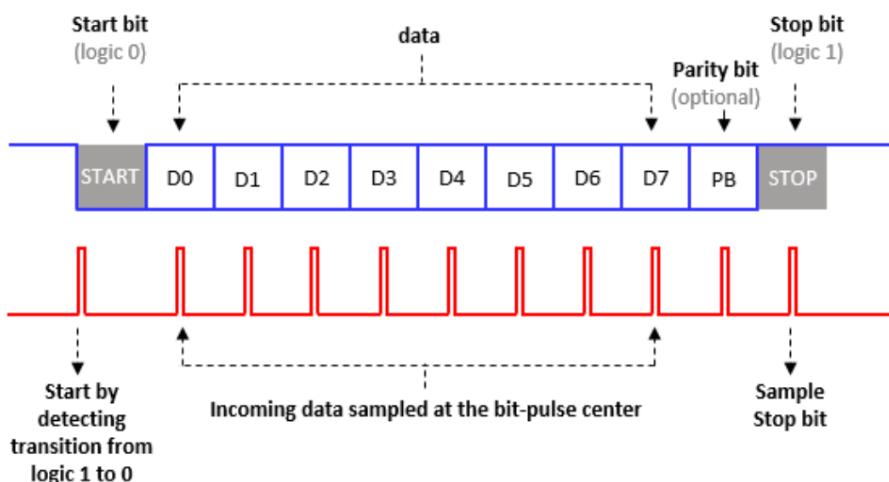
+ **Start bit** : là bit đầu tiên được truyền trong 1 Frame. Báo hiệu cho thiết bị nhận có một gói dữ liệu sắp dc truyền đến. Bit bắt buộc.

+ **Data** : dữ liệu cần truyền. Bit có trọng số nhỏ nhất LSB được truyền trước sau đó đến bit MSB.

+ **Parity bit** : kiểm tra dữ liệu truyền có đúng không.

+ **Stop bit** : là 1 hoặc các bit báo cho thiết bị rằng các bit đã được gửi xong. Thiết bị nhận sẽ tiến hành kiểm tra khung truyền nhằm đảm bảo tính đúng đắn của dữ liệu. Bit bắt buộc.

- Thường thì UART truyền theo chuẩn RS232, có tốc độ baud, 1 bit start, data 8 bit, 1 bit parity và 1 bit stop.



2. Các thanh ghi phục vụ UART:

- **SCON (serial control)** : thanh ghi điều khiển chế độ hoạt động

- **SBUF (serial buffer)** : thanh ghi dữ liệu đệm, có hai thanh ghi SBUF , một dành cho việc thu và còn lại dành cho việc phát.

2.1. Thanh ghi SCON:

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
-----	-----	-----	-----	-----	-----	----	----

- Bit SM0 và SM1:

SM0	SM1	Chế độ	Khung dữ liệu	Tốc độ baud
0	0	0	8-bit shift register	Oscillator /12
0	1	1	8-bit UART	Cài đặt bởi timer 1
1	0	2	9-bit UART	Oscillator/64
1	1	3	9-bit UART	Cài đặt bởi timer 1

- Bit SM2: là bit sử dụng cho truyền thông giữa các bộ vi điều khiển.
- REN: Bit cho phép nhận dữ liệu, khi sử dụng chức năng nhận dữ liệu qua cổng UART, phải bật bit này lên 1.
- TB8, RB8: Bit thứ 9 của truyền/nhận trong chế độ truyền 9 bit.
- TI, RI: Cờ báo ngắt truyền, ngắt nhận.

2.2. Thanh ghi SBUF:

- **SBUF** là thanh ghi 8 bit được dùng riêng cho truyền thông nối tiếp trong AT89C52. Đối với một byte dữ liệu muốn truyền qua đường **TxD** thì nó phải được đặt trong thanh ghi **SBUF**. Tương tự, **SBUF** cũng giữ một byte dữ liệu khi nó được nhận từ đường **RxD** của AT89C52:

- Khi một byte được ghi vào thanh ghi **SBUF** nó sẽ được đóng khung với các bit **Start**, **Stop** và được truyền nối tiếp quan chân **TxD**.

- Khi các bit được nhận nối tiếp từ **RxD** thì AT89C52 mở khung đó để loại trừ các bit **Start**, **Stop** để lấy ra một byte từ dữ liệu nhận được và đặt byte đó vào thanh ghi **SBUF**.

3. Tính tốc độ baud.

Có 2 cách thiết lập tốc độ Baud (ứng với Oscillator frequency=11.0592MHz=11059200hz)

3.1. Dùng TIMER1 mode 2 (xem thanh ghi trạng thái TCON)

```
Baud Rate = Oscillator frequency (HZ) / (32*12*(256-TH1)) ; //không nhân đôi
TMOD |= 0x20; //Timer 1 mode 2.
TL1 = 0xFD; // baud rate =9600
TH1 = 0xFD; // baud rate =9600
TR1 = 1; // Bit điều khiển khởi động hoặc dừng bộ Timer1/ Counter1.
```

5. Các bước thiết lập để truyền/nhận dữ liệu qua UART với AT89C52

- Thiết lập chế độ truyền/tốc độ baud:

- + Ghi giá trị 0x20 vào thanh ghi TMOD để cài đặt sử dụng timer 1 ở chế độ 8 bit, tự nạp lại (TMOD = 0x20).
- + Ghi vào thanh ghi TH1 giá trị thích hợp để chọn tốc độ baudrate.

$TH1 = 256 - (F(\text{crystal})/12/32/\text{baudrate})$. (Với $F(\text{crystal})$ tính bằng đơn vị Hz).

- + Ghi vào thanh ghi SCON giá trị 0x50 để thiết lập chế độ truyền 8 bit, 1 bit start, 1 bit stop ($SCON = 0x50$ ($SM0 = 0$, $SM1 = 1$, $REN = 1$)).
- + Bật bit TR1 để khởi động timer 1.

- Truyền 1 ký tự.

- + Ghi ký tự vào thanh ghi SBUF để truyền đi.
- + Chờ cho cờ TI được bật.
- + Xóa cờ TI để sử dụng cho lần truyền ký tự tiếp theo

- Nhận 1 ký tự.

- + Chờ cờ RI được bật báo có dữ liệu đến.
- + Đọc dữ liệu từ thanh ghi SBUF.
- + Xóa cờ RI cho lần nhận dữ liệu tiếp theo.

Tham khảo slides của LAB: [8051 Microcontrollers](#)