LỜI GIỚI THIỆU

Để đáp ứng yêu cầu giảng dạy chương trình đào tạo nghề "Tự động hóa công nghiệp" cũng như việc cung cấp tài liệu giúp cho sinh viên học tập, khoa Cơ Điện tử chúng tôi đã tiến hành biên soạn giáo trình " Vi điều khiển" .

Giáo trình này giúp các bạn có thêm kỹ năng:

- Lắp ráp và vận hành thiết bị Vi điều khiển.
- Khắc phục các lỗi của các phần tử cơ khí, điện và phần mềm của hệ thống Vi điều khiển.
- Tổ chức nơi làm việc gọn gàng, ngăn nắp và đúng các biện pháp an toàn.

Đây là công trình được viết bởi đội ngũ giáo viên đã và đang công tác tại trường TCN KTCN Hùng Vương cùng với sự góp ý và phản biện của các doanh nghiệp trong lĩnh vực liên quan, tuy vậy, cuốn sách chắc chắn vẫn không tránh khỏi những khiếm khuyết. Chúng tôi mong nhận được ý kiến đóng góp của bạn đọc để cuốn sách được hoàn thiện hơn trong lần tái bản.

Xin trân trọng giới thiệu cùng bạn đọc!

Quận 5, ngày tháng năm 2018

Tham gia biên soạn

Chương 0 :

GIỚI THIỆU SƠ LƯỢC VỀ VI ĐIỀU KHIỂN 89V51

I.Khái quát các tính năng:

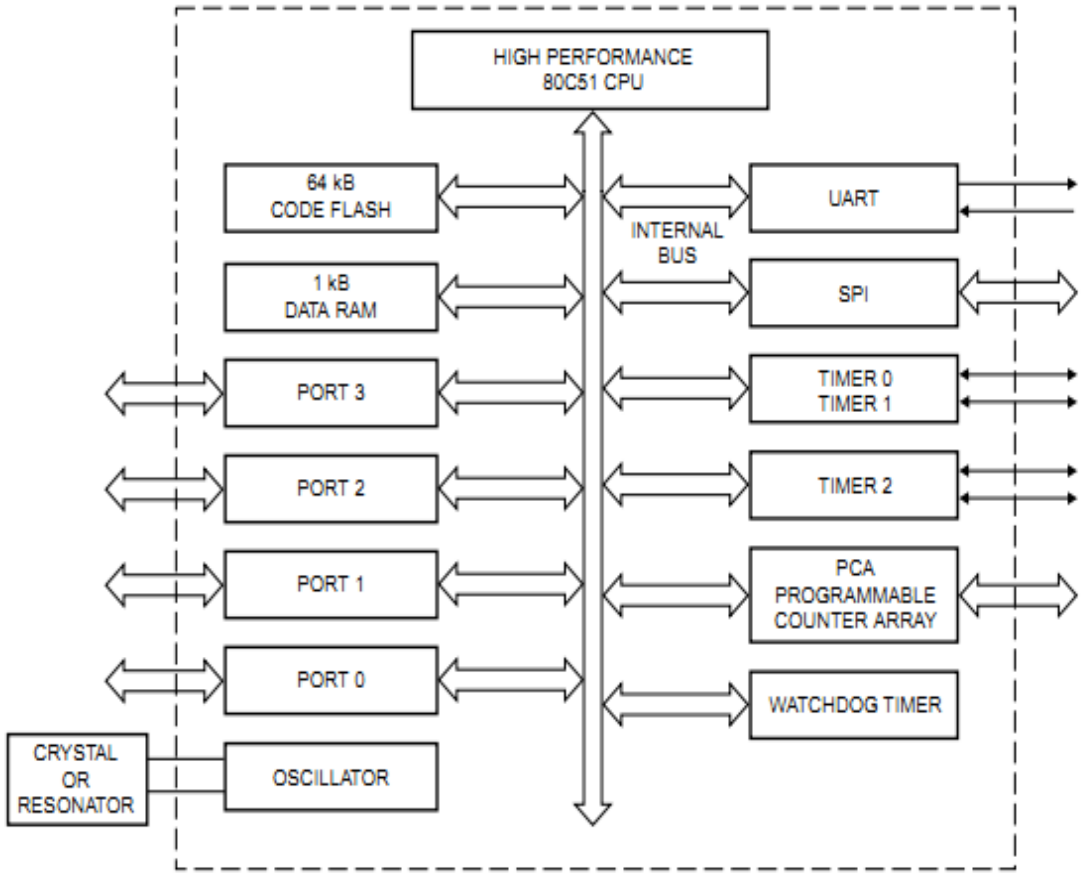
+ Khái quát:

- P89V51RD2 là vi điều khiển 80C51 có 64kB Flash và 1024bytes (1kB) bộ nhớ dữ liệu RAM.
- Tính năng đặc biệt của P89V51RD2 là ở chế độ hoạt động mode x2, ở chế độ này để tăng đôi tốc độ khi hoạt động ở cùng tần số dao động (một chu kì máy=6 chu kì xung nhịp).
- Bộ nhớ chương trình Flash cho phép lập trình ISP hoặc/và song song. Chế độ lập trình song song được đưa ra để thích ứng với tốc độ cao, giảm thời gian và giá thành.
- IAP/ISP.

+ Các tính năng:

- CPU 80C51.
- Hoạt động ở 5VDC trong tầm tần số dao động đến 40MHz.
- 64kB ISP.
- SPI
- 5 PCA với chức năng PWM/capture/compare 16bits.
- 4 cổng xuất nhập.
- 3 Timers/Couters 16bits.
- Watchdog Timer có thể lập trình được.
- 8 nguồn ngắt.
- 2 thanh ghi DPTR.
- Tương thích mức logic TTL và CMOS.
- Phát hiện nguồn yếu <Brownout Detect>
- Chế độ Low-power, Power down, Idle.

+ Sơ đồ khối:



Sơ đồ khối vi điều khiển 89V51

II.Sơ đồ chân và tính năng:

Gồm 4 port với các chân được sắp xếp giống vi điều khiển 89C51

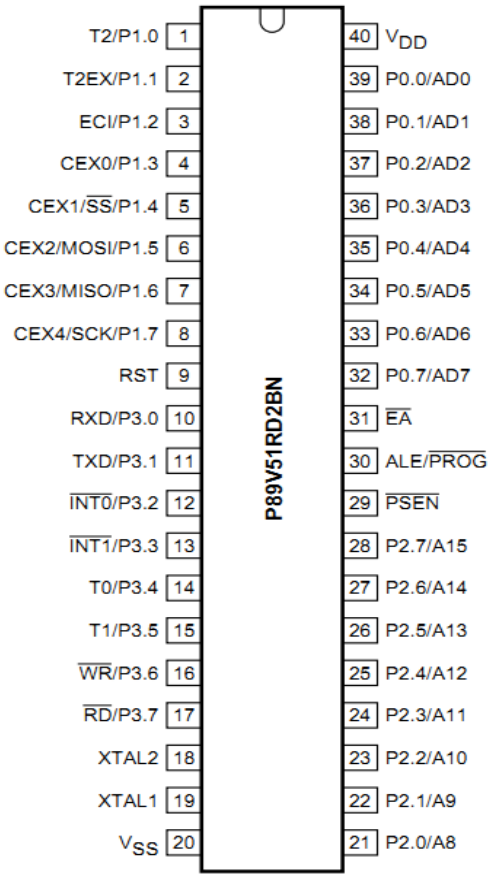
+Port 0 : Loại cực thu để hở

- Từ P0.0 đến P0.7 ứng với chân 39 đến chân 32 của vi điều khiển. Có chức năng là port vào/ra. Khi tất cả các chân đều ở mức logic 1 được dùng như trở kháng cao ở đầu vào.

- Port 0 ở mức thấp dùng ghép địa chỉ trong quá trình truy cập bộ nhớ dữ liệu.

+ Port 1 :

- Từ P1.1 đến P1.7 ứng với chân 1 đến chân 8 của vi điều khiển. Ngoài chức năng là port vào/ra nó còn được dùng với các chức năng đặc biệt khác: Như từ chân P1.4 đến P1.7 có chức năng Capture/Compare.



Port	Chân	Tên	Chức năng
P3.0	39	RXD	Ngõ vào port nối tiếp
P3.1	38	TXD	Ngõ ra port nối tiếp
P3.2	37	$\overline{INT0}$	Ngắt ngoài 0
P3.3	36	$\overline{INT1}$	Ngắt ngoài 1
P3.4	35	T0	Ngõ vào đếm cho Timer/Couter0
P3.5	34	T1	Ngõ vào đếm cho Timer/Couter1
P3.6	33	$\overline{WR}$	Ghi dữ liệu
P3.7	32	$\overline{RD}$	Đọc dữ liệu

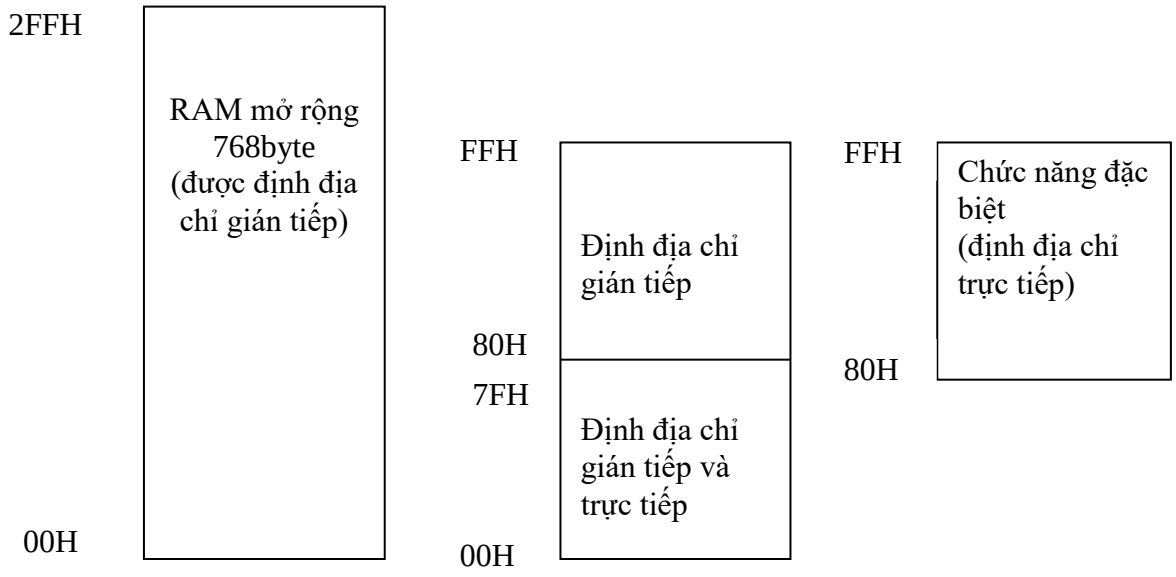
+ Port 2: Gồm các chân từ 21 đến chân 28, dùng như các đường xuất nhập.

+ Port 3: Từ chân 32 đến chân 39, ngoài chức năng dùng như các đường xuất nhập nó còn nhiều chức năng đặc biệt khác như bảng trên.

III.Tổ chức bộ nhớ:

Vi điều khiển 89V51 có không gian địa chỉ riêng biệt cho chương trình và dữ liệu.

+ Bộ nhớ chương trình Flash: Có hai khối
  - Khối 0 gồm 64 kbytes, 512 ô nhớ mỗi ô nhớ chứa 128 bytes để chứa mã của người dùng.
  - Khối 1 chứa ISP/IAP do Philips cung cấp và 8 kbytes đầu tiên chứa bộ nhớ mã người dùng.



Cấu trúc bộ nhớ mở rộng

+ Bộ nhớ mở rộng: 89V51 có 1 kbytes của bộ nhớ mở rộng, bao gồm 4 phần:
  - Từ 00H đến 7FH được định địa chỉ trực tiếp và gián tiếp.
  - 80H đến FFH được định địa chỉ gián tiếp.
  - 80H đến FFH chức năng đặc biệt, chỉ được định địa chỉ trực tiếp.
  - RAM mở rộng từ 00H đến 2FH được định địa chỉ gián tiếp hướng bên ngoài.

Chương 1 :

HỆ TỐI THIỂU CỦA MÁY TÍNH

I- Đại số biến logic:

1- Biến và hàm:

Hàm là một quy luật mà qua đó ta xác định được biến thứ hai y từ biến x thứ nhất.

Nếu giá trị của x không nhiều thì có thể biểu diễn bởi một bảng sau:

Thí dụ: Hàm $y = 5x^2 + 4$, với x = 0, 1, 2, 3, 4 thì

x	y
0	4
1	9
2	24
3	49

Khái niệm này được mở rộng cho các biến không phải là số.

Thí dụ:

Gọi x là 3 màu của một đèn giao thông, và y là đáp ứng của người lái xe, ta có:

x	y=f(x)
xanh	chạy
vàng	chạy chậm
đỏ	dừng

2- Biến logic:

Một biến logic thỏa hai tính chất sau:

- Chỉ có thể nhận được 1 trong 2 giá trị có thế có.
- Hai giá trị nhận được phải mang tính loại trừ nhau.

Thí dụ:

Nếu chỉ xét đèn xanh và đỏ, thì ta có:

$$\overline{\text{xanh}} = \text{đỏ}$$

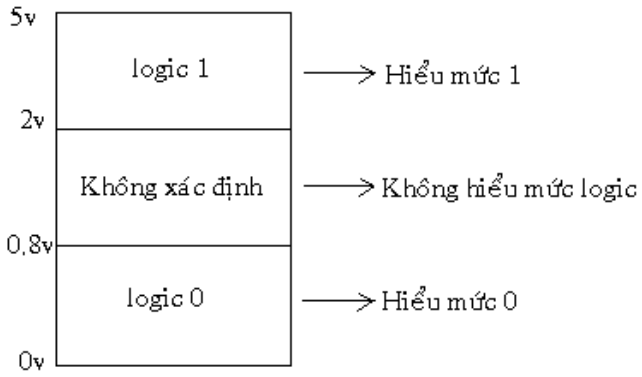
$$\overline{\text{chạy}} = \text{dừng}$$

3- Biểu diễn giá trị của biến logic bằng mức điện áp:

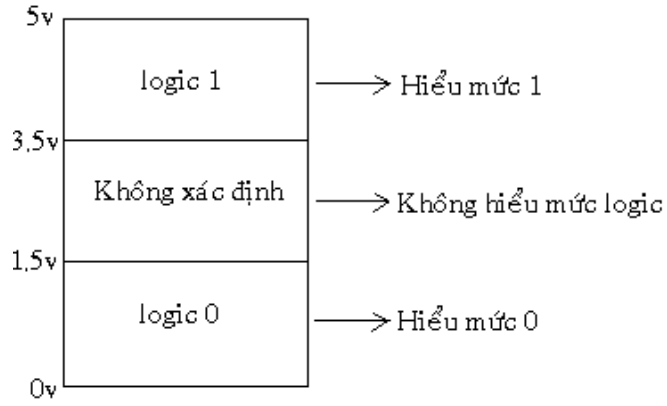
Trong điện tử người ta thường biểu diễn giá trị của biến logic bằng các mức điện áp như sau:

Biến logic (x)	Mức điện áp (v)	y = f(x) (Đèn)
0	0v	Đèn tắt
1	5v	Đèn sáng

+ Mức logic cho họ TTL:



+ Mức logic cho họ CMOS:



II- Hệ thống số và mã: (Number system and codes)

1- Mã thập phân: (Decimal number system)

Trong hệ thập phân dùng 10 chữ số(digit) từ 0 đền 9 để diễn tả số lượng từ 0 đến 9, nếu số lượng lớn hơn 9 ta phải dùng số có nhiều con số nhưng phải theo quy ước về giá trị hang.

Thí dụ:

$$(2546)_{10} = 2,000 + 500 + 40 + 6$$

$$= 2.10^3 + 5.10^2 + 4.10^1 + 6.10^0$$

(MSD) (LSD)

Còn gọi là hệ có cơ số 10, trong đó số mũ sẽ giảm dần về 0.

- MSD: Most significant digit : Là số cao nhất.
- LSD: Least significant digit : Là số thấp nhất.

Nếu có N số thập phân thì có 10^N số khác nhau để diễn tả con số từ 0 -> 10^N - 1

- N=2 thì có $10^2 = 100$ số khác nhau từ 0 đến 99.
- N=3 thì có $10^3 = 1000$ số khác nhau từ 0 đến 999.

2- Mã nhị phân: (Binary number system)

Là mã có cơ số 2, chỉ dùng hai con số 0 và 1 để diễn tả. Nếu diễn tả con số lớn hơn 1 người ta phải dùng nhiều con số nhưng phải theo quy ước về giá trị hang.

Thí dụ:

$$(110101)_2 = 1.2^5 + 1.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0$$

(MSB) (LSB)

$$= 32 + 16 + 4 + 1 = (53)_{10}$$

- MSB: Most significant bit: bit cao nhất.
- LSB: least significant bit: bit thấp nhất.
- Một con số trong số nhị phân được gọi là 1 bit.

- Số nhị phân 4 bit được gọi là 1 nipple (1001).
- Số nhị phân 8 bit được gọi là 1 byte (10011101).
- Số nhị phân 16 bit (2 bytes) được gọi là 1 từ (word).

Một số nhị phân n bit thì diễn tả được 2^n trạng thái từ (000…0) -> (111…1)

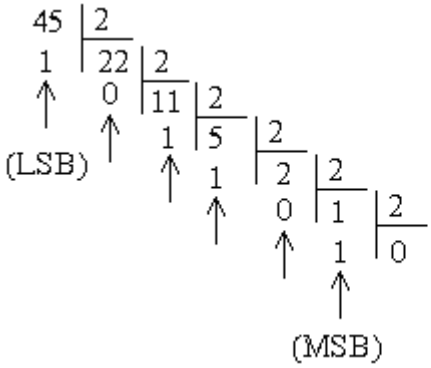
- Số nhị phân 2 bit thì có $2^2 = 4$ số khác nhau từ $(00)_2$ đến $(11)_2$ tương ứng từ 0 đến 3 thập phân.

- Số nhị phân 4 bit thì có $2^4 = 16$ số khác nhau từ $(0000)_2$ đến $(1111)_2$ tương ứng từ 0 đến 15 thập phân.

- Nếu muốn diễn tả số thập phân > 15 thì số nhị phân tương ứng phải có n>4 bit.

+ Đổi từ thập phân sang nhị phân:

Bằng cách chia 2 liên tiếp và lấy phần dư:



Vậy $(45)_{10} = (101101)_2$

3- Mã thập lục phân: (Hexadecimal number system)

Còn gọi là mã Hex hay mã có cơ số 16, dùng 16 chữ số 0,1,2,…, 9, A, B, C, D, E, F để diễn tả.

Thí dụ:

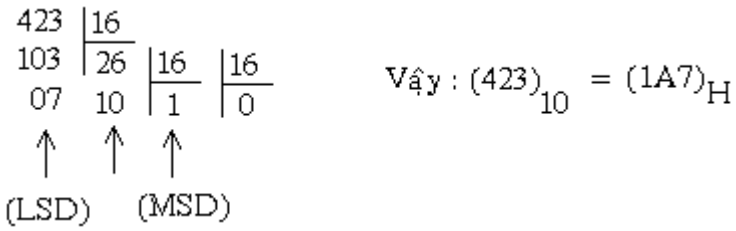
$$(356)\ H = 3.16^2 + 5.16^1 + 6.16^0 = 768 + 80 + 6 = (854)_{10}$$

(MSD) (LSD)

$$(2AF)_H = 2.16^2 + 10.16^1 + 15.16^0 = 512 + 160 + 15 = (687)_{10}$$

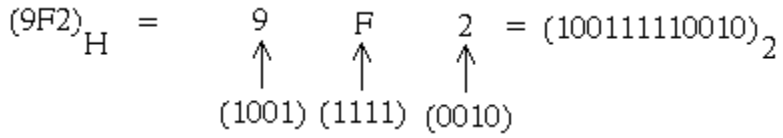
a- Đổi từ thập phân sang Hex:

Chia 16 liên tiếp và lấy phần dư:



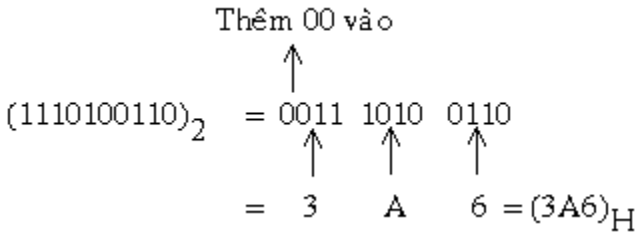
b- Đổi từ Hex sang nhị phân:

Dùng nhóm 4 bit để diễn tả:



c- Đổi từ nhị phân sang Hex:

Dùng nhóm 4 bit để diễn tả, nếu không đủ thì them 0 vào :



d- Đếm trong hệ Hex:

Sử dụng 16 chữ số từ 0 đến F và mỗi lần tăng 1:

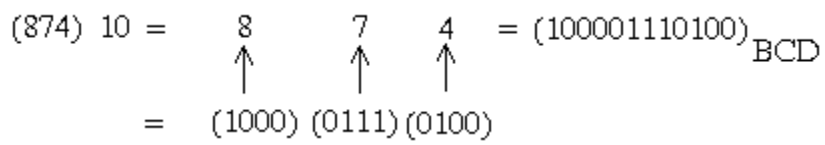
+ 38, 39, 3A, 3B, 3C, 3D, 3E, 3F, 40, 41, 32.

+ 6F8, 6F9, 6FA, 6FB, 6FC, 6FD, 6FE, 6FF, 700.

4- Mã BCD: (Binary Coded Decimal)

Là mã thập phân được mã hóa theo nhị phân, mã BCD dùng nhóm 4 bit để biểu diễn số thập phân từ 0 đến 9.

+ Đổi thập phân sang BCD:



+ Đổi BCD sang thập phân:

$$(0110100000111001)_{BCD} = 0110\ \ 1000\ \ 0011\ \ 1001 = (6839)_{10}$$
$$= 6\ \ 8\ \ 3\ \ 9$$

+ Lưu ý: (01101100) không phải mã BCD do (1100) >9

BẢNG SO SÁNH CÁC MÃ

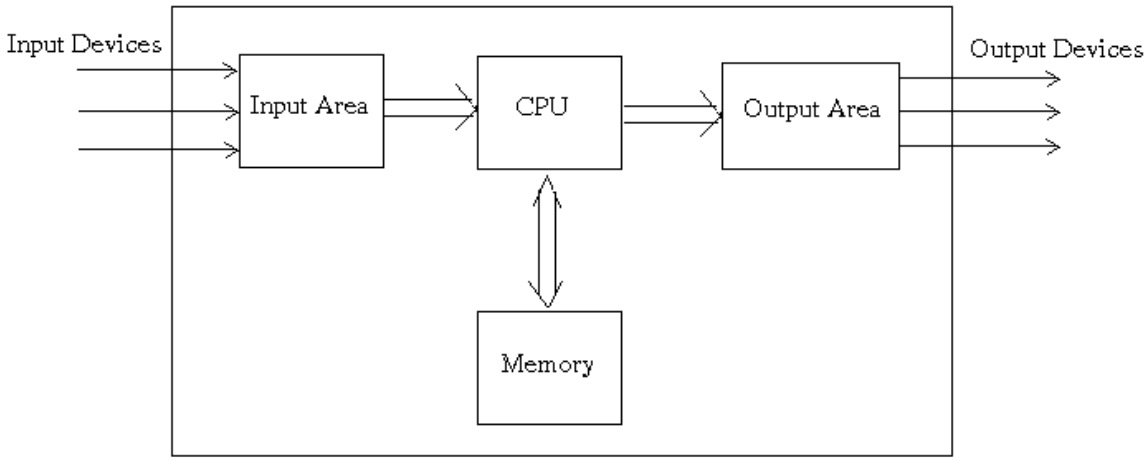
Thập phân	Nhị phân	Hex	BCD
0	0000	0	0000
1	0001	1	0001
2	0010	2	0010
3	0011	3	0011
4	0100	4	0100
5	0101	5	0101
6	0110	6	0110
7	0111	7	0111
8	1000	8	1000
9	1001	9	1001
10	1010	A	0001 0000
11	1011	B	0001 0001
12	1100	C	0001 0010
13	1101	D	0001 0011
14	1110	E	0001 0100
15	1111	F	0001 0101

III- Hệ tối thiểu máy tính:

1- Cấu trúc:

Hệ tối thiểu máy tính gồm 4 phần chính sau :

+ Input Area: Các tín hiệu được nhận từ bên ngoài sẽ được lưu vào vùng này.
+ Output Area: Để lưu tạm các lệnh điều khiển ở đầu ra trước khi đưa đến các thiết bị ngoại vi.
+ CPU: Là đơn vị xử lý trung tâm, nơi sẽ xứ lý và thực thi chương trình.
+ memory: Là nơi lưu giữ chương trình điều khiển và các trạng thái nhớ trung gian trong quá trình thực thi chương trình.
+ Input Device: Là nút nhấn, công tắc, các loại cảm biến, Encoder,…
+ Output Device: Là Motor, Relay, Đèn…



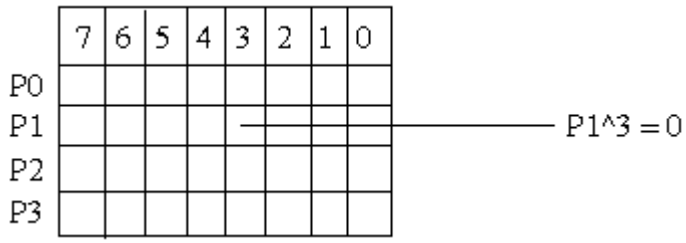
+ Các thiết bị là hệ tối thiểu như : uP, AVR, PIC, PLC, RTU, PC, IPC,DSP,PCI . . .

2- Cách định địa chỉ:

Cấu trúc địa chỉ trong hệ tối thiểu:

<Tiền tố><Địa chỉ Port>^<Địa chỉ bit>

Đối với uP 8051 thì tiền tố luôn là chữ P, có 4 port thứ tự là P0 đến P1, mỗi port có 8 bit.



Thí dụ : P0 = 0 ; Nghĩa là 8 bit của port 0 (từ 0 đến 7) đều bằng 0.

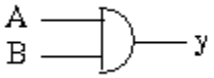
P1^3 = 0 ; Nghĩa là chỉ có bit 3 của port 1 bằng 0.

IV- Các hàm logic cơ bản:

1- Hàm AND:

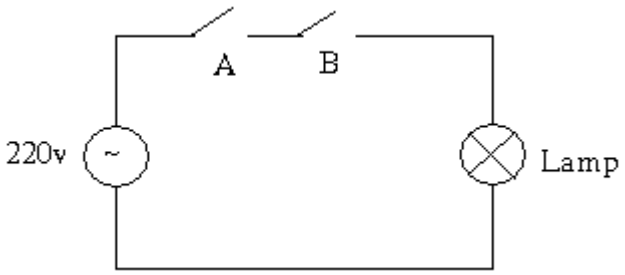
+ Phương trình điều khiển: y = AB

+ Ký hiệu:

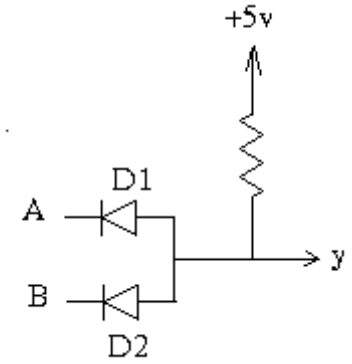


+ Diển tả bằng mạch điện:

Hai công tắc A và B nối tiếp nhau



+ Diển tả bằng mạch điện tử: Hai diod và điện trở kéo lên R



+ Bảng sự thật:

A	B	y
0	0	0
0	1	0
1	0	0
1	1	1

+Phát biểu:

Hàm AND có ngỏ ra bằng 0 khi chỉ cần 1 ngỏ vào bằng 0 bất chấp ngỏ còn lại. Phát biểu này đúng cho trường hợp cổng AND có n ngỏ vào.

+Công dụng :

Hàm AND dùng để duy trì cho ngỏ ra bằng 0 bằng cách làm mất tác dụng ngỏ vào.

+ Chương trình C :

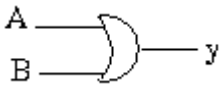
y = (unsigned)A*(unsigned)B;

hay y = A&B;

2- Hàm OR:

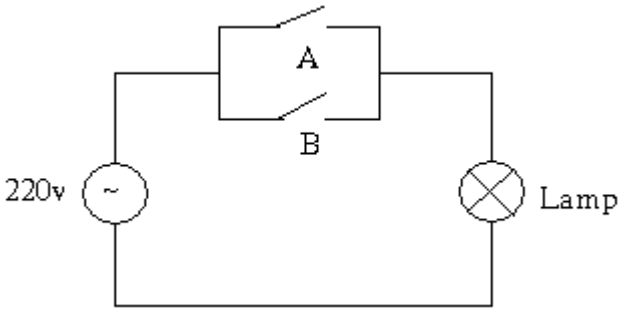
+ Phương trình điều khiển: y = A+B

+ Ký hiệu:

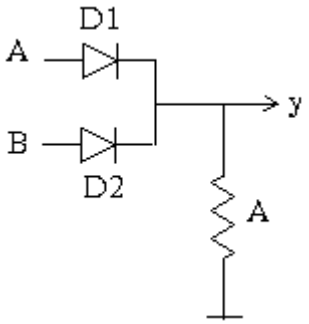


+ Diển tả bằng mạch điện:

Hai công tắc A và B song song nhau



+ Diển tả bằng mạch điện tử: Hai diod và điện trở kéo xuống R



+ Bảng sự thật:

A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

+Phát biểu:

Hàm OR có ngỏ ra bằng 1 khi chỉ cần 1 ngỏ vào bằng 1 bất chấp ngỏ còn lại. Phát biểu này đúng cho trường hợp cổng OR có n ngỏ vào.

+Công dụng :

Hàm OR dùng để duy trì cho ngỏ ra bằng 1 bằng cách làm mất tác dụng ngỏ vào.

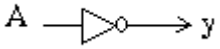
+ Chương trình C : y = (unsigned)A+(unsigned)B;

hay y = A|B;

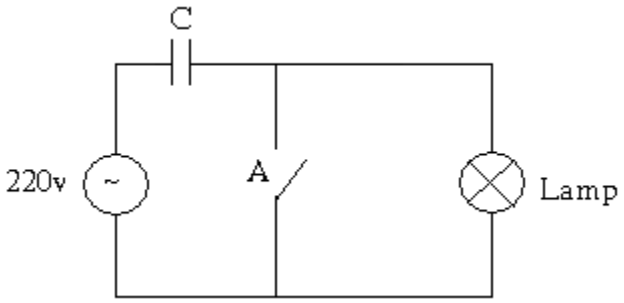
3- Hàm NOT:

+ Phương trình điều khiển: $y = \overline{A}$

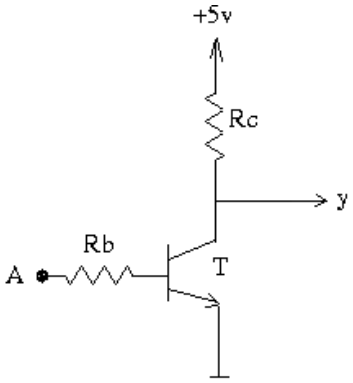
+ Ký hiệu:



+ Diển tả bằng mạch điện:



+ Diển tả bằng mạch điện tử:



+ Bảng sự thật:

A	y
0	1
1	0

+ Chương trình C :

y = !A;

V- Cấu trúc chương trình C:

Cấu trúc chương trình C bao gồm 4 phần cơ bản:

1- Khai báo hàm thư viện.

2- Khai báo I/O, relay trung gian và các chân điều khiển.

3- Chương trình con (nếu cần).

4- Hàm main.

Thí dụ:

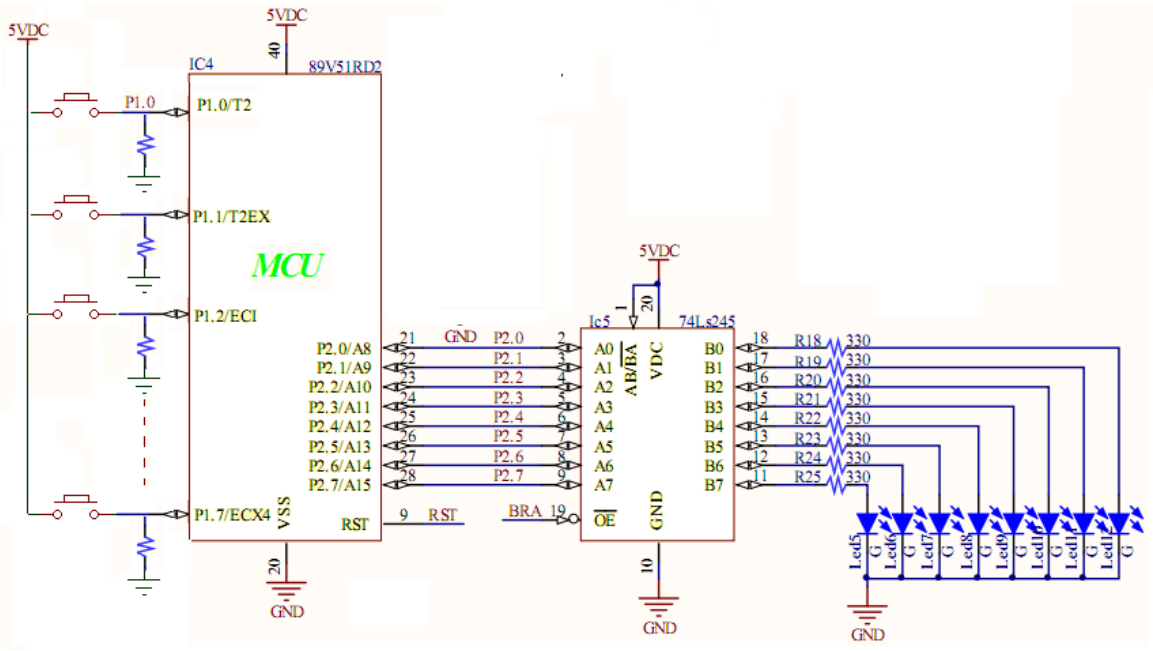
Viết chương trình điều khiển một hệ thống có phương trình sau :

Lamp = (start + Lamp)*(!stop)

```
//Khai báo hàm thư viện.
#include<reg52.h>
//Khai báo ngỏ vào.
sbit  start = P1^0;
sbit  stop = P1^1;
//Khai báo ngỏ ra.
sbit  Lamp = P2^0;
//Khai báo ngỏ vào.
sbit  EN   = P3^7;
//Hàm main
void main(void)
{
  P2 = 0; //Gán trị đầu cho biến
  EN = 0; //Cho phép xuất ngỏ ra.
  while(1)
   {
      //Phương trình điều khiển
      Lamp = (start|Lamp)&(!stop);
   } }
```

VI- Các bài tập:

+ Sơ đồ phần cứng:



- Port 1 có điện trở nối thấp để bình thường là 0 và tác động ở mức cao.
- Port 2 được đệm qua 74LS245 và có chân cho phép OE.

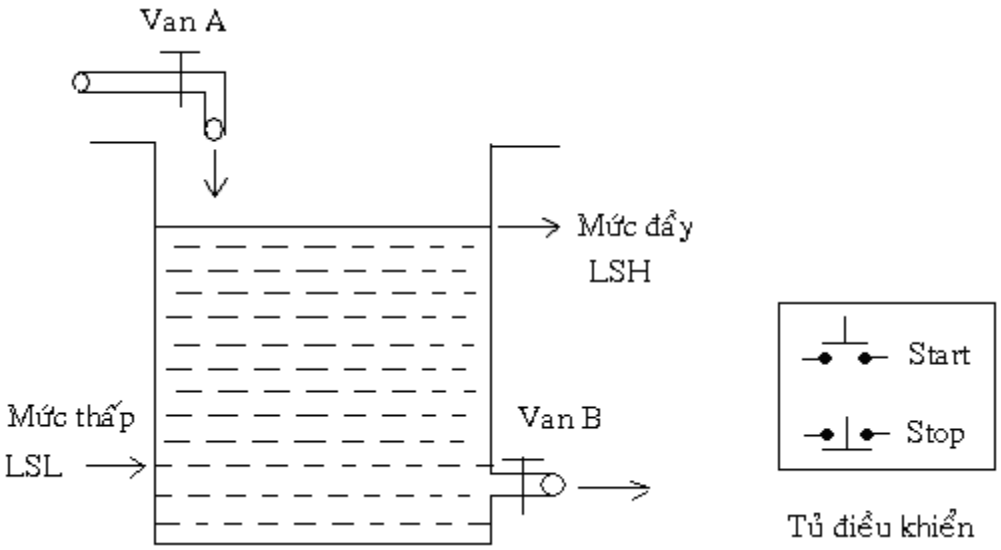
+ Cách xây dựng phương trình điều khiển:

- Xác định số ngỏ vào và ra của hệ thống, có bao nhiêu ngỏ ra thì có bấy nhiêu phương trình điều khiển cần tìm.
- Liệt kê tất cả các điều kiện làm phương trình =1.
- Liệt kê tất cả các điều kiện làm phương trình =0.
- Xét các ngỏ ra cần duy trì nếu có.

Bài tập 1:

Cho hệ thống bơm nước như hình vẽ, với nguyên tắc hoạt động như sau:

- Bình thường van A đóng, van B mở.
- Khi nhấn start van A = ON, van B = OFF.
- Khi mực nước đầy (LSH = ON) thì hệ thống đảo trạng thái nghĩa là van A = OFF, van B = ON.
- Khi mực nước thấp (LSL = ON) thì thì hệ thống trở về trạng thái ban đầu, nghĩa là van A = ON, van B = OFF.
- Quá trình cứ tuần tự tiếp diễn.



a- Xây dựng phương trình điều khiển.

b- Viết chương trình C.

+ Giải:

a-Phương trình điều khiển có dạng:

```
den_do = (start + den_do)*(!stop);
van_A  = den_do*(!LSH * !Relay); // den_do là điều kiện van_A = ON
van_B  = (!van_A)*den_do;
Relay  = (LSH + Relay)*(!LSL);
```

b-Chương trình C:

```
#include<reg52.h>
sbit  start  = P1^0;
sbit  stop   = P1^1;
sbit  LSH    = P1^2;
sbit  LSL    = P1^3;
sbit  den_do = P2^0;
sbit  van_A  = P2^1;
sbit  van_B  = P2^2;
sbit  EN     = P3^7;
bit   Relay  ;
```

```
void main(void)
{
    P2 = 0;P1 = 0;
    EN = 0;
    while(1)
     {
        den_do = (start | den_do)&(!stop);
        van_A  = den_do&(!LSH & !Relay);
        van_B  = (!van_A)&den_do;
         Relay  = (LSH | Relay)&(!LSL);
     }
 }
```

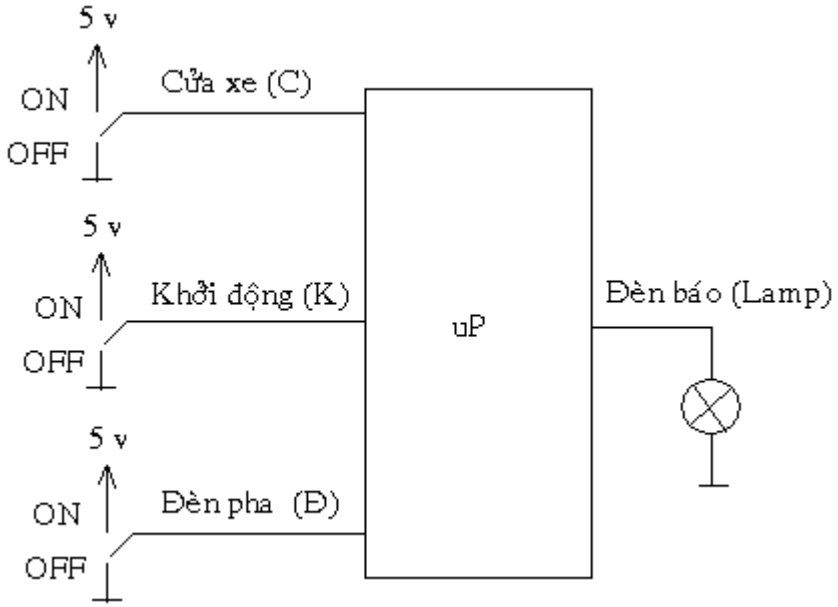
Bài tập 2:

Cho hệ thống cảnh báo trong xe ô tô như hình vẽ. Đèn báo sẽ sang khi:

- Cửa ở vị trí ON trong khi công tắc khởi động ở vị trí ON.
- Đèn pha ở vị trí ON trong khi công tắc khởi động ở vị trí OFF.

a- Xây dựng phương trình điều khiển.

b- Viết chương trình C.



+ Giải:

a- Phương trình điều khiển có dạng:

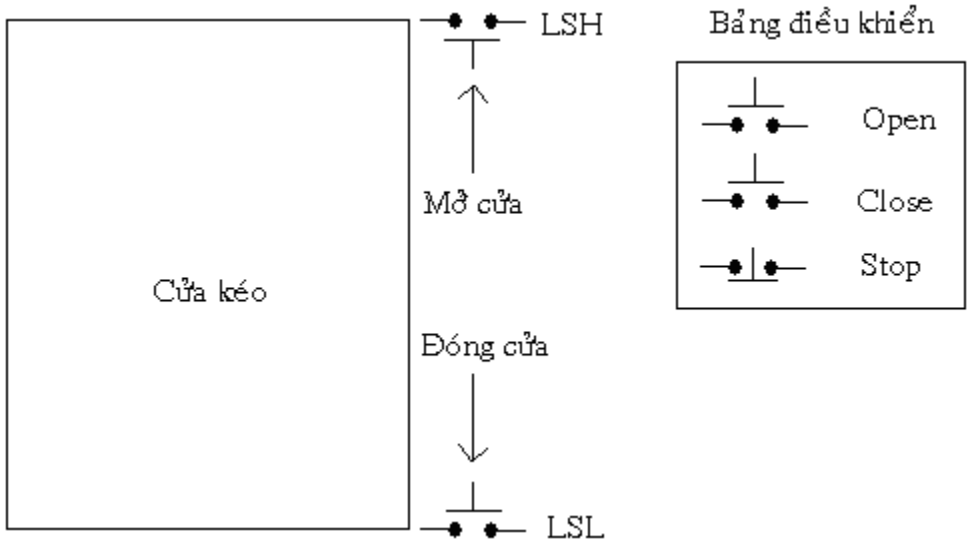
Lamp = CK + Đ(!K)

b-Chương trình C:

```
#include<reg52.h>
sbit  C     = P1^0;
sbit  K     = P1^1;
sbit  D     = P1^2;
sbit  Lamp  = P2^0;
sbit  EN    = P3^7;
void main(void)
{
    P2 = 0;  EN = 0;
    while(1)
     {
       Lamp = (C&K) | D&(!K);
    }
 }
```

Bài tập 3:

Cho hệ thống điều khiển mở cửa từ xa như hình vẽ.



a- Xây dựng phương trình điều khiển.

b- Viết chương trình C.

+Giải:

a- Phương trình điều khiển có dạng:

Gọi MOpen là motor mở cửa, MClose là motor đóng cửa. Ta có:

MOpen = (Open+ MOpen)*(!LSH) *(!Close) *(!stop)

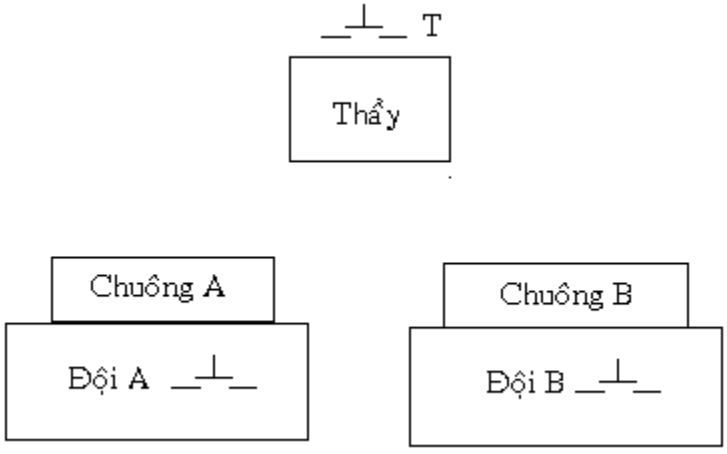
MClose = (Close+ MClose)*(!LSL) *(!Open) *(!stop)

b-Chương trình C:

```
#include<reg52.h>
sbit  Open   = P1^0;
sbit  Close  = P1^1;
sbit  stop   = P1^2;
sbit  LSH    = P1^3;
sbit  LSL    = P1^4;
sbit  MOpen  = P2^0;
sbit  MClose = P2^1;
sbit  EN     = P3^7;
void main(void)
{
  P2 = 0;
  EN = 0;
  while(1)
   {
        MOpen  = (Open|MOpen)  &(!LSH)&(!Close)&(!stop);
        MClose = (Close|MClose)&(!LSL)&(!Open) &(!stop);
  }
 }
```

Bài tập 4:

Cho hệ thống như hình vẽ, với nguyên tắc hoạt động sau:



Sau khi thầy giáo đặt câu hỏi và nhấn nút, thì đội nào có người nhấn nút trước chuông của đội đó sẽ reo trong khi chuông của đội kia không reo mặc dù được nhấn sau đó.

a- Xây dựng phương trình điều khiển.

b- Viết chương trình C.

+Giải:

a- Phương trình điều khiển có dạng:

Chuông_A = T*(Đội_A*! Chuông_B);

Chuông_B = T*(Đội_B*! Chuông_A);

b- Viết chương trình C:

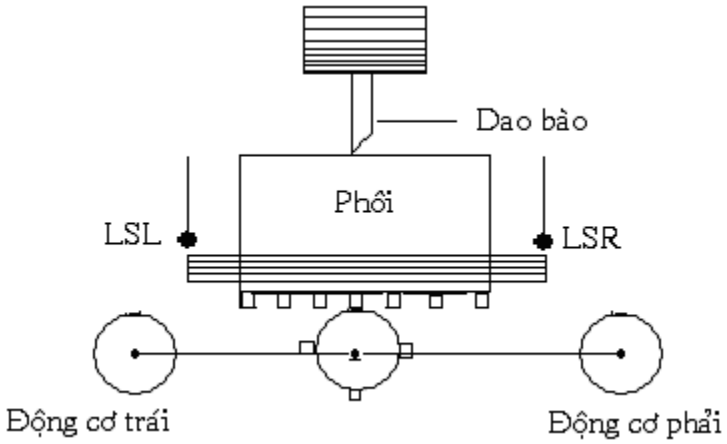
```
#include<reg52.h>
sbit  T        = P1^0;
sbit  Doi_A    = P1^1;
sbit  Doi_B    = P1^2;
sbit  chuong_A  = P2^0;
sbit  chuong_B  = P2^1;
sbit  EN    = P3^7;
void main(void)
{
```

```
P2 = 0; EN = 0;
while(1)
 {
    chuong_A = T&(Doi_A&chuong_B);
    chuong_B = T&(Doi_B&chuong_A);
 } }
```

Bài tập 5: Cho hệ thống máy bào như hình vẽ.

a- Xây dựng phương trình điều khiển.

b- Viết chương trình C.



+Giải:

a- Phương trình điều khiển có dạng:

Gọi MR là motor phải và ML là motor trái, ta có:

```
Relay = (start + Relay)*(!stop);
MR    =  (Relay*LSL + MR)*(!LSR);
ML    =  (Relay*LSR + ML)*(!LSL);
```

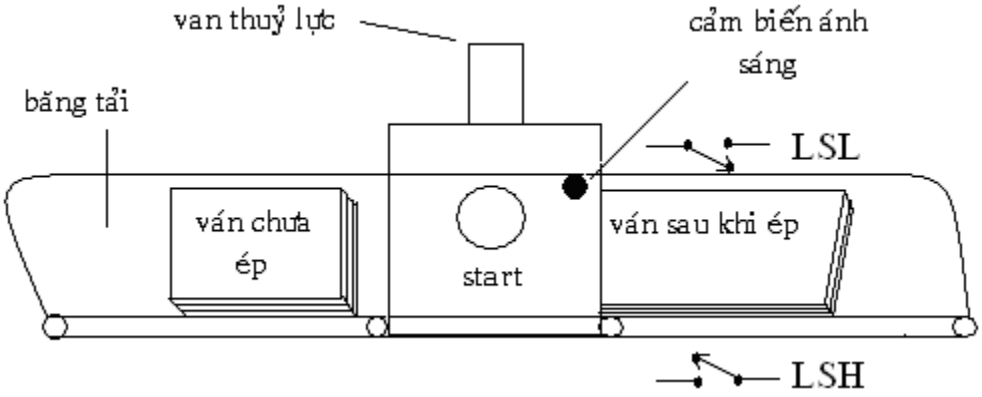
b- Viết chương trình C:

```
#include<reg52.h>
sbit  LSL   = P1^0;
sbit  LSR   = P1^1;
sbit  start = P1^2;
```

```
sbit  stop  = P1^3;
sbit  MR    = P2^0;
sbit  ML    = P2^1;
sbit  EN    = P3^7;
bit   Relay ;
void main(void)
{
    P2 = 0;
    EN = 0;
    while(1)
     {
        Relay = (start|Relay)&(!stop);
        MR    = (Relay&LSL|MR)&(!LSR);
        ML    = (Relay&LSR|ML)&(!LSL);
    } }
```

Bài tập 6:

Cho hệ thống máy ép ván như hình vẽ.



Với nguyên lý hoạt động như sau:

- Nhấn start băng tải hoạt động để lần lượt đưa ván vào bộ phận ép, khi cảm biến quang phát hiện ván đã nằm đúng vị trí thì băng tải dừng và van = ON để thực hiện quá trình ép.
- Khi LSH = ON thì van = OFF và trở về vị ví ban đầu.

- Cho đến khi LSL = ON thì băng tải hoạt. Quá trình cứ tuần tự tiếp diễn.

a- Xây dựng phương trình điều khiển.

b- Viết chương trình C.

+Giải:

a- Phương trình điều khiển có dạng:

```
den_do   = (start + den_do)*(!stop);
bang_tai = LSL*(!van) * den_do;
van      = cam_bien*(!LSH)* (!Relay)* den_do;
Relay    = (LSH + Relay)*(!LSL);
```

b- Viết chương trình C:

```
#include<reg52.h>
sbit  LSL      = P1^0;
sbit  LSH      = P1^1;
sbit  start    = P1^2;
sbit  stop     = P1^3;
sbit  cam_bien  = P1^4;
sbit  bang_tai  = P2^0;
sbit  van      = P2^1;
sbit  den_do    = P2^2;
sbit  EN       = P3^7;
bit   Relay ;
void main(void)
{
  P2 = 0;   EN = 0;
  while(1)
   {
        den_do   = (start | den_do)&(!stop);
        bang_tai = LSL&(!van)&den_do;
        van      = cam_bien&(!LSH)&(!Relay)&den_do;
        Relay    = (LSH | Relay)&(!LSL);} }
```

VII- Các hàm thao tác bit:

1-Hàm xoay tròn:

a- Hàm xoay tròn phải:

Cú pháp: ***_cror_(port,b)***, xoay tròn phải b bit.

Hàm thư viện : ***#include<intrins.h>***

Thí dụ: Xoay phải P2 1 bit.

```
#include<reg52.h>
#include<intrins.h>
sbit  EN        = P3^7;
//Hàm delay ms
void delay_ms(int counter)
{
    int i , j ;
    for(i=0; i <= counter ; i++)
      {
         for(j=0; j <= counter ; j++);
      }
 }
void main(void)
  {
      P2 = 0x80;
      EN = 0;
      while(1)
        {
           P2 = _cror_(P2,1);
           delay_ms(100);
         }
    }
```

b- Hàm xoay tròn trái:

Cú pháp: ***_crol_(port,b)***, xoay tròn trái b bit.

Hàm thư viện : ***#include<intrins.h>***

Thí dụ: Xoay trái P2 1 bit.

```
#include<reg52.h>
#include<intrins.h>
sbit  EN       = P3^7;
void main(void)
{
     P2 = 0x01;
     EN = 0;
     while(1)
      {
         P2 = _crol_(P2,1);
         delay_ms(100);
       }
}
```

2-Hàm ghi chuyển:

a- Shift phải:

Cú pháp: ***_iror_(port,b)***, shift phải b bit và thêm 0 vào.

Hàm thư viện : ***#include<intrins.h>***

Thí dụ: Shift phải P2 1 bit và thêm 0 vào.

```
#include<reg52.h>
#include<intrins.h>
sbit  EN       = P3^7;
void main(void)
{
    int i;
    P2 = 0x80;  EN = 0;
```

```
while(1)
 {
   if(++i==7)
    {
        i=0;  P2 = 0x80;//Gán lại P2
        delay_ms(500);
     }
    P2 = _iror_(P2,1);
    delay_ms(100);
   }  }
```

b- Shift trái:

Cú pháp: ***_irol_(port,b)***, shift trái b bit và thêm 0 vào.

Hàm thư viện : ***#include<intrins.h>***

Chương 2:

CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ C

Các vấn đề cơ bản của ngôn ngữ C cho vi điều khiển :

+ Cấu trúc cơ bản của một chương trình.
+ Các phép toán của C .
+ Các kiểu dữ liệu (int , float , double , char , unsigned char , …)
+ Các hàm trong C
+ Cấu trúc điều khiển hay các tập lệnh.

I. Cấu trúc chương trình C:

Cấu trúc một chương trình C gồm 4 bước :

1- Khai báo hàm thư viện.

2- Khai báo I/O và các chân điều khiển.

3- Chương trình con (nếu cần).

4- Hàm main

```
void main(void)
{
  //Gán trị đầu cho các biến
  while(1)
  {
     //Các phương trình điều khiển.
   }
}
```

II. Các phép toán của C:

1. Phép toán số học hai ngôi:

+ *Toán tử gán* (=)

+ *Các toán tử số học* (+ , - ,* , / , %)

+ cộng

- trừ

* nhân

/ chia

% lấy phần dư (trong phép chia)

+*Các toán tử gán phức hợp* : (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)

a -= 5; tương đương với a = a - 5;

a /= b; tương đương với a = a / b;

a*=2 ; tương đương với a = a*2

+*Tăng và giảm* (++ , --)

a++; <=> a+=1; <=> a=a+1;

a--; <=> a+=1 <=> a=a-1

+*Tiền tố hay hậu tố* (++a ; a++)

B=3;

A=++B; // A là 4, B là 4

Hay

B=3;

A=B++; // A là 3, B là 4

2. Phép toán quan hệ :

+*Các toán tử quan hệ* (== , != ,< , > , <= , >=)

== Bằng

!= Khác

> Lớn hơn

< Nhỏ hơn

>= Lớn hơn hoặc bằng

<= Nhỏ hơn hoặc bằng

3. Phép toán quan logic:

+*Các toán tử logic* (!, &&, ||)

! NOT

&& AND

|| OR

4. Phép toán thao tác bit:

Cho phép xử lý từng bit, không dùng cho kiểu float và double.

+*Các toán tử thao tác bit* (&, |, ^, ~, <<, >>)

& AND Logical AND

| OR Logical OR

^ XOR Logical exclusive OR

~ NOT Đảo ngược bit

<< SHL Dịch bit sang trái

>> SHR Dịch bit sang phải

Lưu ý:

a << n có nghĩa a*(2) $a.2^n$

a >> n có nghĩa a*(2) $\frac{a}{2^n}$

***Thứ tự ưu tiên**

1-	() [ ] -> .	Trái qua phải
2-	! ~ & * - ++ --	Phải qua trái
3-	* / % .	Trái qua phả
4-	+ -	Trái qua phải
5-	<< >>	Trái qua phải
6-	< <= > >=	Trái qua phải
7-	== !=	Trái qua phải
8-	&	Trái qua phải
9-	^	Trái qua phải
10-	|	Trái qua phải
11-	&&	Trái qua phải
12-	||	Trái qua phải
13-	? :	Phải qua trái
14-	= += -= *= /= %= <<= >>= &= ^= |=	Phải qua trái

III. Các kiểu dữ liệu:

1. Hàm xuất nhập có khuôn: (Dùng để truyền thông)

#include <stdio.h>

printf (khuôn dạng, biến);

Khuôn dạng :

%d cho số nguyên int

%u cho số nguyên int không dấu

%x, %X cho số Hex

%f, %e, %E, %g cho số thực

Chú ý :

- Các ký tự b, B, l, L có thể đứng trước d,u,x như : %bd, %ld

- Các ký tự đặc biệt :

‘ \’ ’ dấu ‘

‘ \” ’ dấu “

‘ \\ ’ dấu \

‘ \n ’ Xuống hàng (LF)

‘ \0 ’ null

‘ \t ’ tab

‘ \b ’ backspace

‘ \r ’ Trở về đầu dòng (CR)

2. Kiểu bit: kiểu bit chỉ có giá trị 0 hoặc 1

Thí dụ:

```
bit relay = 1;
printf("%d\n",(unsigned)relay);
```

3. Kiểu char:

a. signed char: 1byte (-128 -> 127), chứa 256 ký tự

Thí dụ 1:

```
char  a = -25; // Chỉ từ -128 đến +127
printf("%bd\n",a);//Định dạng số nguyên
```

+ Hàm **char** 256 ký tự biểu diễn mã ASCII chia làm 3 nhóm :

- Nhóm 1: Từ 0 -> 31 là các ký tự điều khiển dùng để truyền thong, không in được ra màn hình.

- Nhóm 2: Từ 32 -> 126 là các ký tự in được ra màn hình và máy in.

- Nhóm 3: Từ 127 -> 255 là các ký tự in được ra màn hình nhưng không in được cho máy in.

Thí dụ 2:

```
char  a = 65; // Chỉ từ 32 -> 126
printf("%c\n",a); // Chữ A
```

Thí dụ 3:

```
char  a = 'Q'; // Chỉ 1 ký tự
printf("%c\n",a);
```

b. unsigned char: 1byte (0 -> 256), chứa 256 ký tự

Thí dụ 1:

```
unsigned char a = 254;
printf("%bu\n",a);
```

Thí dụ 2:

```
unsigned char a = 'A';
printf("%bu\n",a);
```

4.Kiểu nguyên:

int có giá trị từ -32768 đến 32768 chứa 2 byte

unsigned int có giá trị từ 0 đến 65535 chứa 2 byte

long (int) có giá trị từ -2147483648 đến +214748364732768 chứa 4 byte

unsigned long (int) có giá trị từ 0 đến 4294697295 chứa 4 byte

Chú ý : Khi gán biến kiểu nguyên có thể dùng số thập phân hay Hex

Thí dụ 1:

```
int b = 12365; // Hay b = 0x0E
printf("%d\n",b);
```

Thí dụ 2:

```
long b = 0x0F;
printf("%ld\n",b);
```

Thí dụ 3:

```
unsigned int  b = 123;
printf("%d\n",b); // printf("%u\n",b);
```

Thí dụ 4:

```
unsigned long  b = 123; // hay b = 2e2;
printf("%lu\n",b);
```

Hay dạng Hex : printf("%lx\n",b);

5. Kiểu số thực: Bao gồm cả âm lẫn dương

float có giá trị từ 3.4e-38 đến 3.4e+38 chứa 4 byte

Thí dụ:

```
float  b = -10.0/3.0;
printf("%10.2f\n",b); //Lấy 2 số lẻ và chùa 10 khoảng trắng
```

double có giá trị từ 1.7e-308 đến 1.7e+308 chứa 8 byte

long double có giá trị từ 3.4e-4932 đến 1.1e+4932 chứa 10 byte

6. Kiểu chuỗi:

Thí dụ:

```
char b[] = "Test String";
printf ("%s\n",b);
```

7. Kiểu hằng:

Hằng số là đại lượng mà giá trị không thay đổi trong suốt quá trình thực thi chương trình :

Thí dụ:

```
#define  MAX  1000
#define  PI   3.141593
printf ("MAX = %d va PI = %f \n",MAX,PI);
```

8. Biến kiểu cấu trúc:

Biến kiểu cấu trúc cho phép lưu trữ và xử lý thông tin dưới dạng phức tạp hơn.

Cú pháp :

```
struct <Tên cấu trúc>
  {
     <Khai báo các thành phần của cấu trúc>
  } ;
```

Thí dụ biến cấu trúc ở các chương sau

9. Hàm chuyển đổi dữ liệu:

#include <stdio.h>

sprintf (buffer,khuôn dạng, biến);

Thí dụ:

```
char buf [10];
float pi = 3.14159 ;
sprintf (buf,"%f\n",pi);
printf (buf);
```

10. Mảng:

Mảng là một biến nhưng phải có từ 2 phần tử trở lên :

Có bao nhiêu kiểu biến thì có bấy nhiêu kiểu mảng

Thí dụ 1:

```
int a[10] ; // Khai báo mảng 1 chiều
float a[3][2] ; // Khai báo mảng 2 chiều
```

Thí dụ 2:

```
int   a[10] = {1,2,3,4,5,6,7,8,9};
float b[10] = {1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9};
printf ("a[0] = %d va b[2] = %f\n",a[0],b[2]);
```

Thí dụ 3:

```
int   a[3][2] = {{1,2},{3,4},{5,6}};
printf ("a[0][1] = %d\n",a[0][1]);
```

Chú ý : Khai báo mảng có thể không cần kích thước

Thí dụ :

```
int   a[] = {1,2,3,4,5,6,7,8,9};
int   b[][2] = {{1,2},{3,4},{5,6}};
```

11. Ép kiểu trong C:

+ Chia lấy phần nguyên :

```
int  a = 11/3;
printf ("%d\n",a);
```

+ Chia lấy phần dư :

```
int  a = 11%3;
printf ("%d\n",a);
```

+ Ép kiểu

```
int  a = 11, b=3;
```

```
float c ;
c = (float)a/(float)b;
printf ("%f\n",c);
```

12.Phạm vi sử dụng của biến:

a. Biến cục bộ:

- Được khai báo bên trong của một hàm, chỉ có giá trị bên trong hàm mà nó được khai báo mà không có giá trị cho các hàm khác.
- Phải luôn gán trị đầu cho biến.

Thí dụ:

```
void main(void)
{
   float a = 19.3; // Biến cục bộ
   printf ("%f\n",a);
}
```

b. Biến toàn cục:

- Được khai báo bên ngoài của một hàm, có giá trị cho toàn chương trình.
- Mặc định biến có giá trị = 0

Thí dụ:

```
float a = 19.3; // Biến toàn cục
void main(void)
 {
    printf ("%f\n",a); }
```

c. Biến tĩnh: Có thể khai báo với static đứng đầu

static int a=12; Chỉ gán giá trị một lần duy nhất

Thí dụ biến tĩnh ở các chương sau

IV. Các hàm trong C:

Có 4 loai hàm trong C:

1.Hàm không có kiểu trả về:

```
void  <Tên hàm>()
  {
     // Các câu lệnh xử lý
  }
```

Thí dụ:

```
      void test()  {
           float a = 12.9;
           printf ("%f\n",a);
       }
      void main(void) {
            test(); //Gọi hàm
       }
```

2.Hàm có kiểu trả về:

```
<Kiểu hàm >  <Tên hàm>
{
     // Các câu lệnh xử lý
     return kq;
}
```

Thí dụ:

```
     float test() //Khai báo hàm kiểu float
       {
          float a = 1.0/3.0;
            return a;
       }
```

```
void main(void)  {
    float kq ;
    kq= test(); //Cách gọi hàm
    printf ("%f\n",kq);
 }
```

3. Hàm có tham số vào:

```
void  <Tên hàm>(biến, biến , ….) {
   // Các câu lệnh xử lý
}
```

Thí dụ:

```
void test(float a , float b) {
    printf ("%f\n",a/b);
 }
void main(void) {
    test(12.0,3.0);//Gọi hàm
 }
```

4.Hàm có tham số vào và ra:

```
<Kiểu hàm >  <Tên hàm>(biến, biến , ….) {
   // Các câu lệnh xử lý
  return kq;
}
```

Thí dụ 1:

```
 #include<math.h>
 #define PI 3.141593
float test(float rad)    {
     return sin(rad);
   }
```

```
void main(void)
  {
     float kq ;
     kq= test(PI/2);
    printf ("%f\n",kq);  //Gọi hàm
  }
```

Thí dụ 2 : Giải phương trình bậc hai y = 3x^2 - 5x + 2 = 0

```
#include<reg52.h>
#include<stdio.h>
#include<math.h>
float x[2];
void roots(float a,float b,float c)
{
  float delta;
  delta = pow(b,2.0) - 4*a*c;
  x[0] = (-b+sqrt(delta))/(2*a);
  x[1] = (-b-sqrt(delta))/(2*a); }
void main(void)
{
   TI=1;
   while(1)
   {
      roots(3.0,-5.0,2.0);
      printf ("Nghiem x1 = %0.3f va Nghiem x2 = %0.3f\r",x[0],x[1]);
   }
}
```

Chương 3:

CÁC TOÁN TỬ ĐIỀU KHIỂN

Một chương trình bao gồm nhiều câu lệnh. Thông thường các câu lệnh được thực hiện một cách lần lượt theo thứ tự mà chúng được viết ra. Các toán tử điều khiển cho phép thay đổi trật tự nói trên, do đó Kit có thể đang từ một câu lệnh này nhảy đến thực hiện một câu lệnh khác ở trước hay sau nó. Đường đi của Kit trở nên linh hoạt hơn và nhờ vậy ta có thể viết chương trình một cách hiệu quả hơn. Xét về mặt công dụng có thể chia toán tử điều khiển thành ba nhóm chính:

+ Nhảy không điều kiện (goto).

+ Rẽ nhánh (if, switch).

+ Tổ chức chu trình (for, while, do - while).

Ngoài ra còn có một số toán tử khác có chức năng bổ trợ như break, continue. Trong chương này sẽ giới thiệu cách viết và nguyên tắc hoạt động của các toán tử nêu trên. Chúng ta sẽ thấy các toán tử điều khiển của C có khả năng làm việc rất linh hoạt, phong phú và mạnh mẽ.

I. Toán tử if:

1. if – else:

Cú pháp :

```
if (biểu thức điều kiện)
  {
     <Lệnh thực hiện >
  }
else
  {
     <Lệnh thực hiện >
  }
```

+ biểu thức điều kiện có thể nguyên hay thực.

Thí dụ 1: Tìm số lớn nhất

```
#include<reg52.h>
#include<stdio.h>
float sln,a,b;
void max(float a,float b)
{
   if(a>b)
      sln= a;
   else
      sln = b;
}
void main(void)
{
  SCON  = 0x50;
  TMOD |= 0x20;
  TH1 = 0xFD;//9600
  TR1= 1;
   EA   = 1;// Cho phep interrupt toan cuc
   TI=1;
   while(1)
    {
        printf("Vao so a : ");
        scanf("%f",&a);
        printf("Vao so b : ");
        scanf("%f",&b);
        max(a,b);
        printf ("So lon nhat = %f\n\n",sln);
    }
}
```

Thí dụ 2: Tìm số lớn nhất và nhỏ nhất

```
#include<reg52.h>
#include<stdio.h>
float max,min,a,b;
void max_min(float a,float b)
{
   if(a>b)
    {
        max = a;
         min = b;
    }
  else
  {
      max = b;
      min = a;
  }
}
void main(void)
{
  SCON  = 0x50;
  TMOD |= 0x20;
  TH1 = 0xFD;//9600
  TR1= 1;
   EA   = 1;// Cho phep interrupt toan cuc
  TI=1;
  while(1)
  {
       printf("Vao so a : ");
       scanf("%f",&a);
       printf("Vao so b : ");
```

```
        scanf("%f",&b);
        max_min(a,b);
        printf ("max = %f và min = %f\n\n",max,min);
   }
}
```

2. if :

Cú pháp :

```
        if (biểu thức điều kiện)
          {
            <Lệnh thực hiện >
          }
```

Thí dụ : Tìm số lớn nhất

```
#include<reg52.h>
#include<stdio.h>
float sln,a,b;
void max(float a,float b)
{
    sln= a;
    if(b>sln)
       sln = b;
}
void main(void)
{
  SCON  = 0x50;
  TMOD |= 0x20;
  TH1 = 0xFD;//9600
  TR1= 1;
  EA   = 1;// Cho phep interrupt toan cuc
  TI=1;
```

```
  while(1)
  {
     printf("Vao so a : ");
     scanf("%f",&a);
     printf("Vao so b : ");
     scanf("%f",&b);
       max(a,b);
       printf ("So lon nhat = %f\n\n",sln);
  }
}
```

3. Toán tử if có thể lồng nhau:

Thí dụ:

```
        if (n>0)
          {
              if(a>b)
                z = a;
              else
                z = b;
          }
```

Bài tập :

Chương trình hàm AND và OR của hai nút nhấn

```
#include<reg52.h>
sbit nut_0       = P1^0;
sbit nut_1       = P1^1;
sbit nut_2       = P1^2;
sbit nut_3       = P1^3;
sbit Lamp_max = P2^0;
sbit Lamp_min  = P2^1;
sbit EN           = P3^7;
```

```
bit max()
{
     bit kq;
     if((unsigned)nut_0>(unsigned)nut_1)
      {
         kq = nut_0;
      }
    else
     {
       kq = nut_1;
     }
   return kq;
 }
bit min()
{
     bit kq;
     if((unsigned)nut_2>(unsigned)nut_3)
      {
        kq = nut_3;
      }
    else
     {
        kq = nut_2;
     }
    return kq;
}
void main(void)
{
  P2 = 0;
  EN = 0;
```

```
  while(1)
  {
        Lamp_max = max();
        Lamp_min = min();
  }
}
```

II. Toán tử switch:

Toán tử switch cho phép căn cứ vào biểu thức nguyên để chọn một trong nhiều cách nhảy. Có cú pháp sau:

```
        switch (Biểu thức nguyên)
        {
             case 1 :
                  <Lệnh thực hiện>
                   break; //Thoát khỏi vòng switch
            case 2 :
                  <Lệnh thực hiện>
                   break; //Thoát khỏi vòng switch
            ..........................................
           case n :
                  <Lệnh thực hiện>
                   break; //Thoát khỏi vòng switch
            default :
                 <Lệnh thực hiện>
        }
```

Thí dụ :

```
#include<reg52.h>
sbit EN = P3^7;
void main(void)
{
   P2 = 0;
```

```
    EN = 0;
    while(1)
     {
        switch (P1)
         {
            case 1 :
                P2 = 1;
                break;
            case 2 :
                P2 = 2;
                break;
             case 3 :
                P2 = 0x04;
                break;
              case 4 :
                P2 = 0x08;
                break;
             default :
                P2 = 0;
         }
   }
}
```

III. Toán tử goto và nhãn:

Cú pháp:

```
        nhan :  <Câu lệnh 1>
                <Câu lệnh 2>
                . . . . . .
                <Câu lệnh n>
       if (Biểu thức)
         {
```

```
            goto nhan;
          }
```

Thí dụ:

```
#include<reg52.h>
sbit EN = P3^7;
sbit start = P1^0;
sbit Lamp = P2^0;
void delay_ms(long cnt)
{
   long i,j;
   for(i=0;i<=cnt;i++)
    {
      for(j=0;j<=cnt;j++);
    }
 }
void main(void)
{
    P2 = 0;
    EN = 0;
    while(1)
    {
        lap_lai_1 :
            Lamp = ~Lamp;
           delay_ms(500);

         lap_lai_2 :
            Lamp = 1;
         if(start)
         {
            goto lap_lai_1;    }
```

```
        else
        {
           goto lap_lai_2;
        }
  }
}
```

IV. Toán tử for:

1. Toán tử for:

Cú pháp:

```
 for (Biểu thức 1 ; Biểu thức 2 ; Biểu thức 3)
  {
     <Câu lệnh>
  }
```

Vòng for sẽ tự động kết thúc khi không thỏa <Biểu thức 2>

Thí dụ: #include<reg52.h>

```
sbit EN = P3^7;
sbit start = P1^0;
int i ;
void main(void)
{
     P1 = 0;  EN = 0;
     while(1)
      {
         if(!start)
         {
           P2 = 1;
           for (i=0;i<8; i++)
            {
              delay_ms(500);
               P2 = P2<<1; //Nhân 2  } }
```

```
        else
        {
           P2 = 0x80;
            for (i=0;i<8; i++)
             {
                delay_ms(500);
                 P2 = P2>>1; //Chia 2
             } } } }
```

2. Toán tử for - break:

Lệnh break sẽ thoát khỏi vòng for mà không cần xét đến <Biểu thức 2>

```
#include<reg52.h>
sbit EN = P3^7;
sbit start = P1^0;
int lamp_buf[] = {0,1,2,4,8,16,32,64,128};
int i,n ;
void main(void)
 {
     EN = 0;
     n = sizeof(lamp_buf)/sizeof(int);
     while(1)
      {
        for (i=0;i<n; i++)
         {
            if(start)
               break;
            P2 = lamp_buf[i];
            delay_ms(500);  }
         P2=255;
      }
  }
```

3. Toán tử for - continue:

Lệnh continue không thoát khỏi vòng for,chỉ quay lại thực hiện <Biểu thức 3> rồi đến <Biểu thức 2>

```
#include<reg52.h>
sbit EN = P3^7;
int lamp_buf[] = {0,1,2,4,8,16,32,64,128};
int i,n ;
void main(void)
{
  EN = 0;
  n = sizeof(lamp_buf)/sizeof(int);
  while(1)
  {
     for (i=0;i<n; i++)
      {
        if(i==3) //Không thực hiện i=3;
            continue;
        P2 = lamp_buf[i];
        delay_ms(500);
      } } }
```

V. Toán tử while:

Cũng giống như for dùng để xây dựng vòng lặp.

Cú pháp :

```
while (Biểu thức)
  {
        <Câu lệnh>
   }
```

Thí dụ 1: Chương trình tính tích vô hướng của hai mảng nguyên

```
#include<reg52.h>
#include<stdio.h>
```

```
int a[] = {1,2,3,4,5}, b[] = {1,2,3,4,5};
void main(void)
{
   int kq, i, n;
   TI=1;
   n = sizeof(a)/sizeof(int);
   while(1)
   {
      i=0;  kq = 0;
      while(i<n)
        {
          kq += a[i]*b[i];
          i++;
        }
        printf("Tich vo huong = %8.2d\r",kq)    ;    }}
```

Thí dụ 2: Chương trình xác định phím được nhấn

```
#include<reg52.h>
sbit start_stop = P1^0;
sbit Lamp = P2^0;
sbit EN = P3^7;
void main(void)
{
   P2=0; EN=0;
   while(1)
    {
        while(!start_stop); //Chờ phím được nhấn
        while(start_stop); //Chờ phím thả ra
        Lamp = ~Lamp;
  }
}
```

VI. Toán tử do - while:

Cú pháp :

```
do
  {
      <Câu lệnh>
   }
while (Biểu thức);
```

Thí dụ: Chương trình tính $\sqrt{a}$ dùng phương pháp lặp:

x(0) = a;

x(n+1) = (x(n)* x(n)+a)/(2* x(n)), với n>=0

Quá trình lặp kết thúc khi :

abs((x(n+1)-x(n))/x(n)) < 0.00001

```
#include<reg52.h>
#include<stdio.h>
#include<math.h>
double a, c, xn;
void main(void)
{
  TI=1; a=2.0;
  while(1)
  {
       xn = a;
       do
        {
          c = xn;
          xn = (xn*xn+a)/(2*xn);
         }
         while(fabs((xn-c)/c)>=1e-5);
        printf("a = 2, sqrt(a) = %8.2f\r",xn); }}
```

Lưu ý:

+ Lệnh break sử dụng được cho : for, while, do – while và switch.

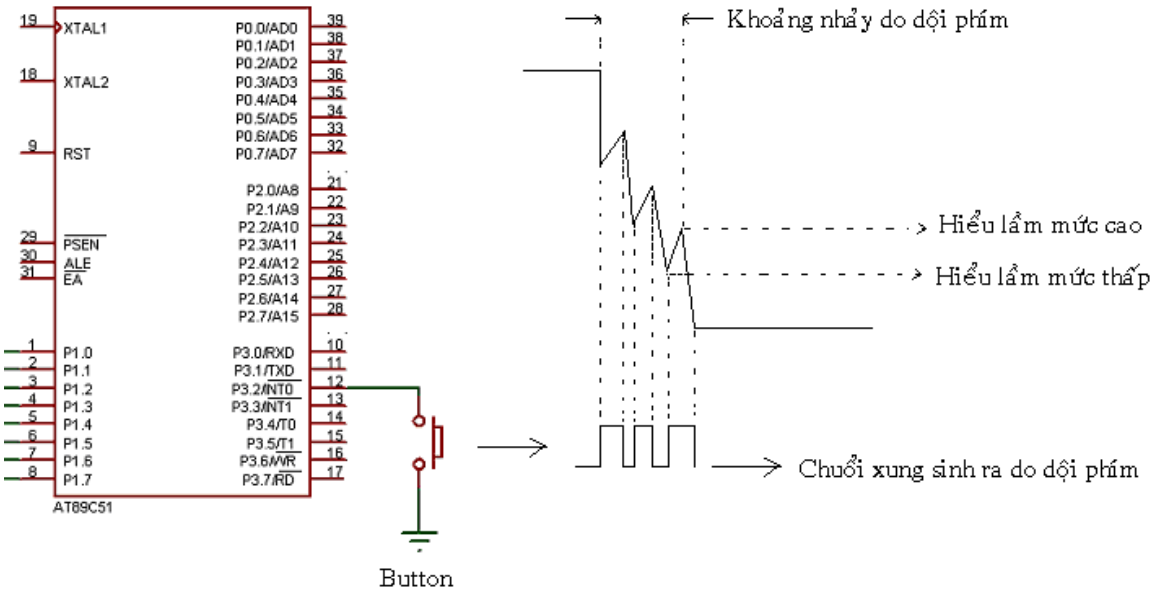
+ Lệnh continue không sử dụng được cho : switch.

Bài tập :

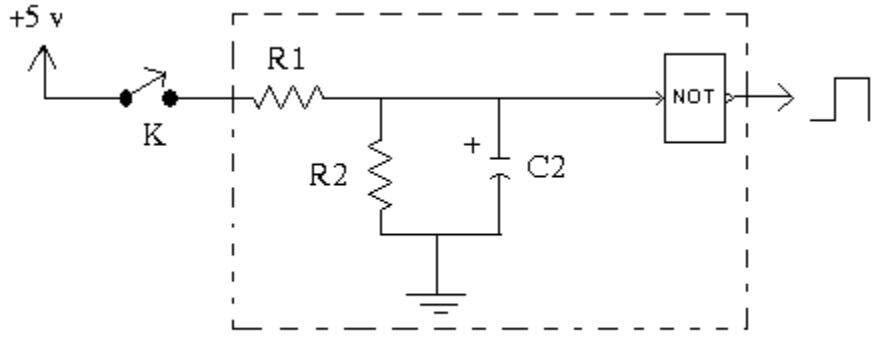
1- Phân chia phần nguyên và thực của số a

```
#include<reg52.h>
#include<stdio.h>
#include<math.h>
float x,int_part, float_part;
void main(void)
{
  TI=1; x = 123.456;
   while(1)
   {
      float_part = modf (x, &int_part);
      printf ("%0.3f = %0.3f + %0.3f\n", x, int_part,float_part);  }}
```

2- Chống dội phím :



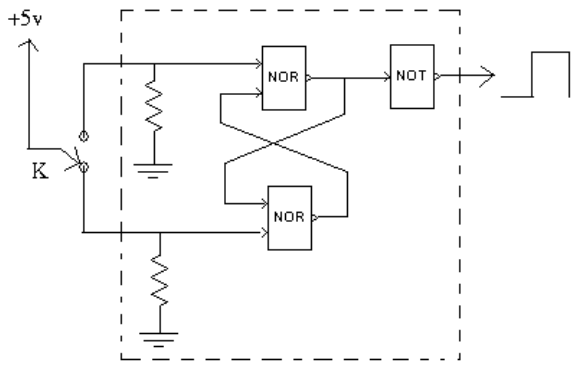
+ Chống dội bằng mạch thụ động



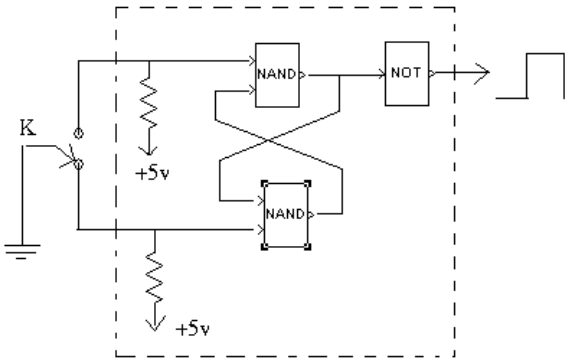
Mạch thụ động

Tại thời điểm t = 0 đóng khóa k, tụ C2 nạp điện bắt đầu từ 0 ngang qua cầu phân thế R1,R2, trong khoảng thời gian rất ngắn trong khoảng dội điện thế nạp ở hai đầu tụ C2 không đủ mức logic cho cổng NOT nên ngỏ ra không thay đổi.

+ Chống dội bằng mạch tích cực:



Chống dội bằng cổng NOR



Chống dội bằng cổng NAND

+ Chống dội bằng phần mềm:

```
#include<reg52.h>
sbit EN  = P3^7 ;
sbit start  = P1^0 ;
int solannha,solannhan;
bit FlagRecv;
unsigned int chong_doi()
{
    unsigned int kq;
```

```
    if(start) // Nếu phím được nhấn
     {
        solannha = 0;
        if(!FlagRecv)
           {
              if(++solannhan==50) //Nếu phím hết dội
                  FlagRecv = 1;//Đã nhận phím nhấn
           }
     }
  else  // Nếu phím được thả ra
   {
      solannhan=0;
      if(FlagRecv)
       {
         if(++solannha == 50) //Nếu phím hết dội
          {
              FlagRecv = 0;
              kq = kq+1;
          } } }
    return kq;
}
 void main(void)
    {
       P2=0;P1=0;
       EN = 0;
      while(1)
      {
          P2 = chong_doi();
      }
   }
```

VII. Biểu thức điều kiện:

Cú pháp:

kq = (Biểu thức) ? a : b ;

kq = a, nếu (Biểu thức) đúng.

kq = b, nếu (Biểu thức) sai.

Thí dụ : Tìm max và min của hai số a và b

```
#include<reg52.h>
#include<math.h>
sbit EN = P3^7;
sbit start = P1^0;
sbit stop = P1^1;
sbit lamp = P2^0;
bit max(unsigned a,unsigned b)
{
     return (a >b)? a :b;
}
bit min(unsigned a,unsigned b)
{
    return (a < b)? a :b;
}
void main(void)
{
  P2=0;  EN = 0;
  while(1)
   {
      lamp = min(start,stop);
   }
}
```

VIII. Con trỏ và địa chỉ:

+ Con trỏ là một biến dùng để chứa đại chỉ. Vì có nhiều loại địa chỉ nên cũng có nhiều kiểu con trỏ tương ứng. Con trỏ kiểu int dùng để chứa địa chỉ các biến kiểu int, Tương tự con trỏ kiểu float dùng để chứa địa chỉ các biến kiểu float.

+ Khai báo con trỏ :

Type *<Tên con trỏ>

Thí dụ:

```
int x, y, *px, *py;
py = &y;  //Gán địa chỉ biến y vào con trỏ py
px = &x; //Gán địa chỉ biến x vào con trỏ px
```

+ Lúc này ta nói px, py lần lượt trỏ đến biến x và y.

+ Khi px trỏ đến x, py trỏ đến y thì các cách viết sau là tương đương:

x hay *px

y hay *py

+ Theo cách khai báo trên thì 3 câu lệnh sau là tương đương:

```
y     = 3*x + 2;
*py = 3*x + 2;
*py = 3*(*px) + 2;
```

+ Từ đây ta rút ra kết luận quan trọng là: Khi biết được địa chỉ của một biến thì chẳng những có thể sử dụng giá trị của biến đó mà còn có thể gán cho biến một giá trị mới. Điều này sẽ được áp dụng để nhận kết quả của một hàm thong qua đối số.

+ Hai chương trình sau là tương đương:

Thí dụ 1 :

```
#include<reg52.h>
#include<stdio.h>
int a[] = {1,2,3},*p;
void main(void)
{
   int i;
```

```
  TI=1;
  while(1)
  {
        p = a; // Hay p = &a[0];
        for(i=0;i<3;i++)
        {
           printf (" p = %d\n",*(p+i));
        }
  }
}
```

Thí dụ 2:

```
#include<reg52.h>
#include<stdio.h>
#include<math.h>
int a[] = {1,2,3},*p;
void main(void)
{
  int i;
  TI=1;
  while(1)
  {
        p = &a; // p = &a[0];
        for(i=0;i<3;i++)
        {
          printf (" p = %d\n",p[i]);
        }
   }
}
```

IX. Các bài tập mẫu:

1- Xét dấu của số a

```
#include<reg52.h>
#include<stdio.h>
#include<math.h>
int a;
int sign(int a)//Ham dau
 {
   return (abs(a)/a);
 }
void main(void)
 {
    SCON  = 0x50;
    TMOD |= 0x20;
    TH1 = 0xFD;//9600
    TR1= 1;
    EA   = 1;// Cho phep interrupt toan cuc
   TI=1;
  while(1)
  {
       printf("Vao so nguyen a : ");
       scanf("%d",&a);
        printf ("Dau cua a = %d\n\n",sign(a));
  }
}
```

2- Tính đạo hàm gần đúng của hàm rời rạc.

```
#include<reg52.h>
#include<stdio.h>
int n,i;
float a[] = {1.0,2.0,4.0,7.0,11.0},kq[5],p;
```

```
float diff(float a[],int n)
{
    for(i=0;i<(n-1);i++)
     {
          kq[i] = a[i+1] - a[i];
      }
     return *kq;
 }
void main(void)
{
    TI=1;
     n = sizeof(a)/sizeof(float);
     while(1)
      {
          p = diff(a,n);
          for(i=0;i<(n-1);i++)
            {
               printf (" diff = %f\n",(p+i));
            }
      }
 }
```

3- Tính tích phân gần đúng của hàm rời rạc.

```
#include<reg52.h>
#include<stdio.h>
float a[] = {1.0,2.0,3.0,4.0,5.0};
//Tính tích phân gần đúng của hàm rời rạc.
float sum(float a[],int n)
{
  float kq;
  int i;
```

```
  kq = 0.0;
  for(i=0;i<n;i++)
  {
    kq += a[i];
  }
  return kq;
 }
//Tính trị trung bình của 1 dãy số.
float mean(float a[], int n)
{
  float kq;
  kq = sum(a,n);
  kq /= (float)n;
  return kq;
 }
void main(void)
{
   int n;
   n = sizeof(a)/sizeof(float);
   TI=1;
   while(1)
   {
          printf (" Sum = %f\n",sum(a,n));
          printf (" mean = %f\n",mean(a,n));
   }
}
```

4- Chương trình tính tích phân hàm lien tục f(x) trên đoạn [a,b] theo phương pháp hình thang bằng cách chia [a,b] thành 1000 đoạn nhỏ có chiều dài như nhau.

a- Tính tích phân hàm f(x) = sin(x), trong [0,pi/2]

```
#include<reg52.h>
#include<stdio.h>
#include<math.h>
#define PI 3.141593
double tp(double a, double b)
{
    int i, n=1000;
    double s,h = (b-a)/n;
    s=(sin(a)+sin(b))/2;
    for(i=1;i<n;i++)
     {
        s+=sin(a+i*h);
     }
    return h*s;
}
void main(void)
{
    TI=1;
     while(1)
       {
           printf("%f\n",tp(0,PI/2));
        }
 }
```

b- Tính tích phân hàm f(x) = cos(x), trong [0,pi]

```
#include<reg52.h>
#include<stdio.h>
#include<math.h>
#define PI 3.141593
double tp(double a, double b)
{
    int i, n=1000;
    double s,h = (b-a)/n;
    s=(cos(a)+cos(b))/2;
    for(i=1;i<n;i++)
     {
        s+=cos(a+i*h);
     }
     return h*s;
}
 void main(void)
  {
     TI=1;
     while(1)
      {
          printf("%f\n",tp(0,PI));
       }
 }
```

c- Tính tích phân hàm f(x) = e^x, trong [0,1], e = 2.718282

```
#include<reg52.h>
#include<stdio.h>
#include<math.h>
#define PI 3.141593
```

```
double tp(double a, double b)
 {
    int i, n=1000;
    double s,h = (b-a)/n;
    s=(exp(a)+exp(b))/2;
    for(i=1;i<n;i++)
     {
        s+=exp(a+i*h);
     }
     return h*s;
   }
 void main(void)
   {
      TI=1;
      while(1)
       {
            printf("%f\n",tp(0,1));
       }
    }
```

d- Tính tích phân hàm f(x) = $(e^x - 2*\sin(x^2))/(1+x^4)$, trong [-1.2,3.5]

```
 #include<reg52.h>
#include<stdio.h>
#include<math.h>
#define PI 3.141593
double tp(double a, double b)
{
    int i, n=1000;
    double s,h = (b-a)/n;
    s = ((exp(a)-2*sin(a*a))/(1+pow(a,4)) + (exp(b) -
              2*sin(b*b))/(1+pow(b,4)))/2;
```

```
    for(i=1;i<n;i++)
    {
       s+=(exp(a+i*h)-2*sin((a+i*h)*(a+i*h)))/(1+pow((a+i*h),4));
    }
   return h*s;
}
void main(void)
 {
    TI=1;
    while(1)
      {
         printf("%f\n",tp(-1.2,3.5));
       }
   }
```

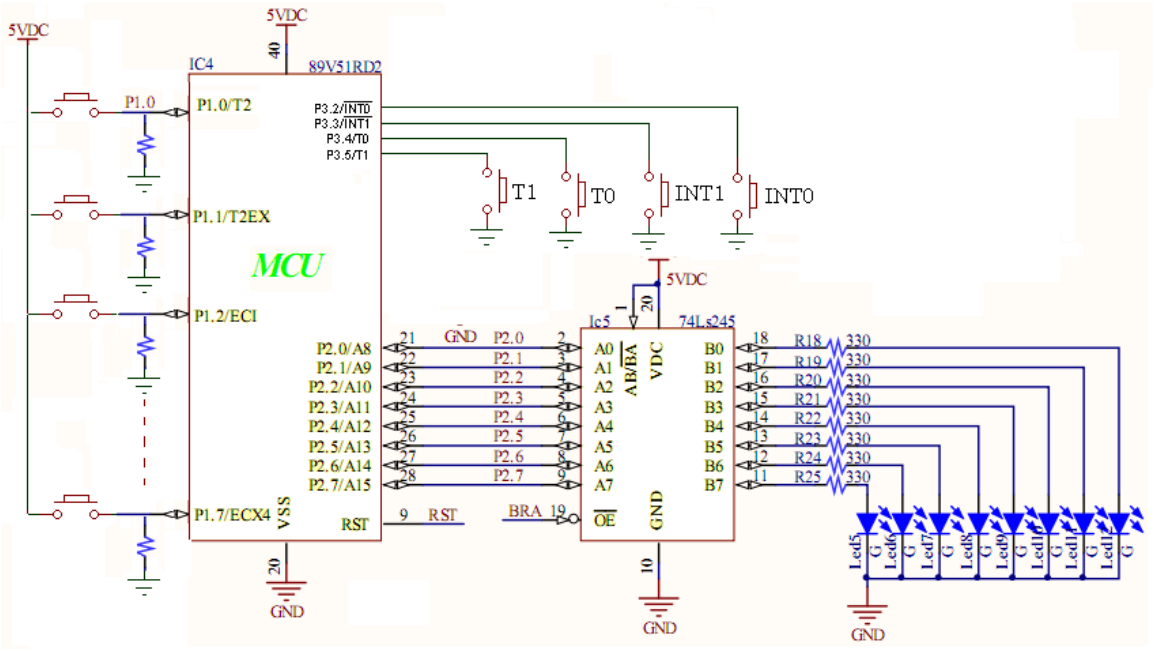
5- Viết các chương trình với các vòng lặp sử dụng các hàm thư viện sau

log(x) = ln(x), log10(x) = log(x)

Chương 4 :

TIMER VÀ INTERRUPT

+ Sơ đồ phần cứng:



I- Timer:

1- Timer 0:

a- Timer 0 Mode 2: (8 bit)

+ Timer 0 Mode 2 có 8 bit : thì mặc định sẽ timer từ 0 đến 255 us (2^8 = 256 trạng thái) thì cờ TF0 tràn

+ Giá trị timer được đặt trong hai thanh ghi TH0 và TL0, và chỉ 1 lần duy nhất khi khởi tạo timer:

+ Gọi x>0 là gía trị cần khởi tạo thanh ghi, và y là thời gian cần timer. Ta có:

$$y = 2^8 - x$$

Thí dụ: Cần khởi tạo timer có thời gian từ 0 us đến 250 us thì cờ TF0 = 1. Thay vào công thức trên ta có:

250 = 256 – x => x = 256 – 250 = 6;

Do đó cần khởi tạo thanh ghi là:

```
TH0 = 0x06 //Hay TH0 = 0x06
TL0 = TH0
```

Chương trình Timer 0 Mode 2:

```
#include<reg52.h>
sbit Lamp = P2^0;
sbit EN   = P3^7;
int msec, sec ,cycle ;
bit timer_sec(int giay)
{
  static bit TIM0 = 0;
  if(TF0) //Xét cờ tràn Tmer 0
   {
     TF0 = 0;

     if(++cycle==4) //1000 us
       {
         cycle = 0;
         if(++msec==1000) //1000000 us = 1s
          {
             msec = 0;
             if(++sec==giay)
               {
                  TIM0 = ~TIM0;
                  sec  = 0;
               }
          }    }
     }
   return TIM0;
}
```

```
void main(void)
{
  //Khởi tạo Timer 0 Mode 2 (8 bit)
  TMOD = TMOD|0x02;
  TH0 = 0x06;
  TL0 = TH0;
  TR0=1; //Cho phép Timer 0 hoạt động
   P2 = 0;   EN = 0;
   while(1)
    {
       Lamp = timer_sec(2); //Timer 2 giây
    }
}
```

b- Timer 0 Mode 1: (16 bit)

+ Timer 0 Mode 1 có 16 bit : thì mặc định sẽ timer từ 0us đến 65535 us (2^{16} = 65536 trạng thái) thì cờ TF0 tràn

+ Giá trị timer được đặt trong hai thanh ghi TH0 và TL0, và luôn được gán lại mỗi khi cờ TF0 tràn:

TH0 = Chứa byte cao.

TL0 = Chứa byte thấp.

Thí dụ: Cần khởi tạo timer có thời gian 250 us, ta có:

250 = 65536 – x => x = 65536 – 250 = 65286 = 0xFF06;

Do đó cần khởi tạo thanh ghi là:

TH0 = 0xFF

TL0 = 0x06

Chương trình Timer 0 Mode 1:

```
#include<reg52.h>
sbit Lamp = P2^0;
sbit EN   = P3^7;
```

```
int msec, sec ,cycle ;
bit timer_sec(int giay)
 {
    static bit TIM0 = 0;
    if(TF0) //Xét cờ tràn Tmer 0
     {
        TF0 = 0;
         TH0 = 0xFF; //Gán lại thanh ghi sau mỗi lần TF0=1
         TL0 = 0x06;
    if(++cycle==4) //1000 us
     {
           cycle = 0;
           if(++msec==1000) //1000000 us = 1s
            {
              msec = 0;
             if(++sec==giay) // giay  s
              {
                 TIM0 = ~TIM0;
                 sec  = 0;
              } } }
     }
    return TIM0;
 }
void main(void)
{
  TMOD = TMOD|0x01;    //Khởi tạo Timer 0 Mode 1 (16 bit)
  TH0 = 0xFF; //Khởi tạo ban đầu
  TL0 = 0x06;
  TR0=1; //Cho phép Timer 0 hoạt động
  P2 = 0; EN = 0;
```

```
  while(1)
  {
        Lamp = timer_sec(2);
  }
}
```

2- Timer 1:

a- Timer 1 Mode 2: (8 bit)

+ Timer 1 Mode 2 có 8 bit : thì mặc định sẽ timer từ 0 đến 255 us (2^8 = 256 trạng thái) thì cờ TF1 tràn

+ Giá trị timer được đặt trong hai thanh ghi TH1 và TL1, và chỉ 1 lần duy nhất khi khởi tạo timer

Chương trình Timer 1 Mode 2 có thời gian timer 250 us:

```
#include<reg52.h>
sbit Lamp = P2^0;
sbit EN   = P3^7;
int msec, sec ,cycle ;
bit timer_sec(int giay)
 {
  static bit TIM1 = 0;
  if(TF1) //Xét cờ tràn Tmer 1
   {
    TF1 = 0;
    if(++cycle==4) //1000 us
     {
          cycle = 0;
          if(++msec==1000) //1000000 us = 1s
           {
              msec = 0;
             if(++sec==giay) // giay  s
               {
```

```
                TIM1 = ~TIM1;
                 sec  = 0;
               } } }
        }
   return TIM1;
 }
void main(void)
{
  TMOD = TMOD|0x20;     //Khởi tạo Timer 1 Mode 2 (8 bit)
  TH1 = 0x06;
  TL1 = TH1;
  TR1=1; //Cho phép Timer 1 hoạt động
  P2 = 0;  EN = 0;
  while(1)
  {
    Lamp = timer_sec(2);
   }
}
```

b- Timer 1 Mode 1: (16 bit)

+ Timer 1 Mode 1 có 16 bit : thì mặc định sẽ timer từ 0us đến 65535 us (2^{16} = 65536 trạng thái) thì cờ TF1 tràn

+ Giá trị timer được đặt trong hai thanh ghi TH1 và TL1, và luôn được gán lại mỗi khi cờ TF1 tràn:

TH1 = Chứa byte cao.

TL1 = Chứa byte thấp.

Thí dụ: Cần khởi tạo timer có thời gian 25000 us, ta có:

25000 = 65536 – x => x = 65536 – 25000 = 40536 = 0x9E58;

Do đó cần khởi tạo thanh ghi là:

TH0 = 0x9E

TL0 = 0x58

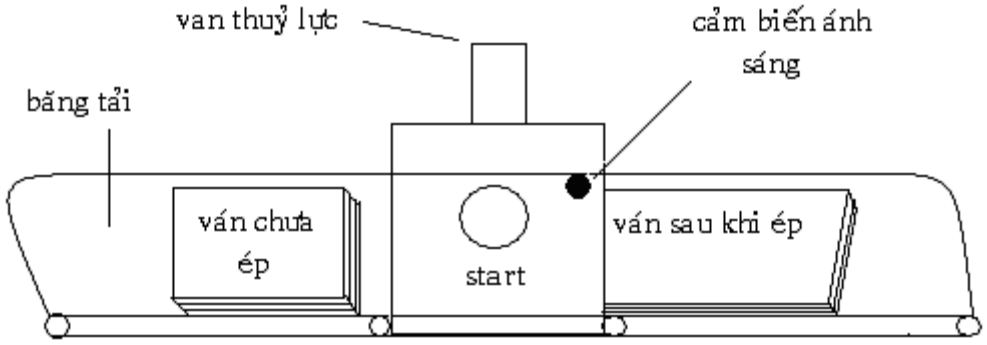
```
#include<reg52.h>
sbit Lamp = P2^0;
sbit EN   = P3^7;
int msec, sec ,cycle ;
bit timer_sec(int giay)
 {
   static bit TIM1 = 0;
   if(TF1)
    {
        TF1 = 0;
        TH1 = 0x9E; //Gán lại sau mỗi lần tràn
        TL1 = 0x58;
        if(++cycle==4) //4*25000us=100000 us
         {
             cycle = 0;

             if(++msec==10) //1000000 us = 1s
               {
                  msec = 0;
                   if(++sec==giay) // giay  s
                      {
                         TIM1 = ~TIM1;
                          sec  = 0;
                      } } } }
    return TIM1;
 }
void main(void)
{
   TMOD = TMOD|0x10;     //Khởi tạo Timer 1 Mode 1 (16 bit)
   TH1 = 0x9E; //Khởi tạo ban đầu
```

```
  TL1 = 0x58;
  TR1=1;
  P2 = 0;  EN = 0;
  while(1)
  {
    Lamp = timer_sec(2);
   }
}
```

Ứng dụng:

Xét hệ thống ép ván sử dụng Timer, với nguyên lý hoạt động như sau:

- Nhấn start băng tải hoạt động để lần lượt đưa ván vào bộ phận ép, khi cảm biến quang phát hiện ván đã nằm đúng vị trí thì băng tải dừng và van = ON để thực hiện quá trình ép trong 10 giây.
- Sau 10 giây thì van = OFF và băng tải = ON.
- Quá trình cứ tuần tự tiếp diễn.




```
#include<reg52.h>
sbit start    = P1^0;
sbit stop     = P1^1;
sbit cam_bien = P1^2;
sbit den_do   = P2^0;
sbit bang_tai = P2^1;
sbit van      = P2^2;
```

```
sbit EN   = P3^7;
#define PERIOD 0x06
int msec, sec ,cycle ;
bit TIM0 = 1;
bit timer_sec(int giay)
 {
      if(TF0)
       {
          TF0 = 0;
          if(++cycle==4) //1000 us
            {
               cycle = 0;
              if(++msec==1000) //1000000 us = 1s
                {
                    msec = 0;

                     if(++sec==giay)
                        {
                          TIM0 = 0;
                          sec  = 0;
                        }  }  }
          }
       return TIM0;
  }
void main(void)
{
  //Khai baoTimer 0 Mode 2
  TMOD = TMOD|0x02;
  TH0 =  PERIOD; //Timer = 250 us
  TL0 =  TH0;
```

```
    TR0 = 1;
    P2   = 0 ;EN=0;
    while(1)
     {
         den_do = (start|den_do)&(!stop);
         bang_tai = (!van)&den_do;
         if(cam_bien)
          {
             TR0 = TIM0;//Cho phép Timer 0 hoạt động
             van = timer_sec(10);
          }
          else//Reset Timer
              TIM0    =1;
      }
}
```

II- Counter:

1- Counter 0:

a- Timer 0 Mode 2 làm Counter: (8 bit)

+ Counter 0 Mode 2 có 8 bit : thì mặc định sẽ đếm từ 0 đến 255 (2^8 = 256 trạng thái) thì cờ TF0 tràn

+ Khai báo Counter 0 Mode 2 : TMOD = TMOD|0x06;

+ Mỗi lần chân P3^4 từ ON xuống OFF thì bộ đếm trong thanh ghi tăng lên 1 cho đến khi đạt đến 256 thì tràn.

+ Chế độ mặc định TH0 = TL0 = 0:

Trong chế độ này, sử dụng bộ đếm chứa trong TL0, đếm từ 0 đến 255 thì tự động trở lại từ đầu, do không sử dụng cờ tràn nên không cần reset lại counter.

```
#include<reg52.h>
sbit EN = P3^7;
void main(void)
{
  TMOD = TMOD|0x06; //Khai báo Timer0 mode 2 làm Counter 0
  TR0 = 1;
  P2   = 0 ;EN=0;
  while(1)
   {
        P2 = TL0;
   }
}
```

+ Chế độ đặt lại:

Trong chế độ này sử dụng cờ tràn TF0 nên cần reset lại counter.

+ Chương trình chớp tắt đèn sau mỗi 6 sản phẩm:

```
#include<reg52.h>
sbit EN   = P3^7;
sbit Lamp = P2^0;
bit counter_0()
 {
      static bit CNT0 = 0;
      if(TF0) //Xét cờ tràn
        {
          TF0 = 0;
           CNT0=~CNT0;
        }
      return CNT0;
 }
```

```
void main(void)
{
  TMOD = TMOD|0x06;
  TH0 = 250;  //Đặt giá trị đếm đến 6 thì cờ tràn.
  TL0 = TH0;
  TR0 = 1;
  P2   = 0 ;EN=0;
  while(1)
   {
         Lamp = counter_0();
    }
  }
```

+ Chế độ mở rộng bộ đếm:

+ Trong chế độ này cần:

- Khai báo biến ngoài: long san_pham
- Đặt TH0 = 255; sao cho mỗi sản phẩm thì cờ TF0 tràn.

```
#include<reg52.h>
#include<stdio.h>
#define PERIOD 0xFF
//Khai báo biến ngoài
long sanpham ;
long counter_0()
 {
    if(TF0)
      {
          TF0 = 0;
          sanpham++;
       }
    return sanpham;
 }
```

```
void main(void)
  {
      TMOD = TMOD|0x06;
      TH0 =  PERIOD;
      TL0 =  TH0;
      TR0 = 1;
       TI=1;
      while(1)
       {
          printf("san pham = %ld\r",counter_0());
       }
   }
```

b- Timer 0 Mode 1 làm Counter: (16 bit)

+ Counter 0 Mode 1 có 16 bit : thì mặc định sẽ đếm từ 0 đến 65536 thì cờ TF0 tràn

+ Khai báo Counter 0 Mode 1 : TMOD = TMOD|0x05;

+ Mỗi lần chân P3^4 từ ON xuống OFF thì bộ đếm trong thanh ghi tăng lên 1 cho đến khi đạt đến 65535 thì tràn.

+ Chương trình đếm 10 sản phẩm thì cờ tràn

```
#include<reg52.h>
#include<stdio.h>
long sanpham ;
long counter_0()
 {
   if(TF0)
    {
        TF0 = 0;
        TH0 =  0xFF; //Gán lại thanh ghi
        TL0 =  246;
        sanpham++;
```

```
    }
    return sanpham;
  }
void main(void)
 {
    TMOD = TMOD|0x05; //Khai báo counter 0 mode 1
    TH0 =  0xFF;
    TL0 =  246; // 10 sản phẩm thì cờ tràn
    TR0 = 1;
    TI=1;
     while(1)
       {
          printf("san pham = %ld\r",counter_0());
       }    }
```

2- Counter 1:

a- Timer 1 Mode 2 làm Counter: (8 bit)

+ Counter 1 Mode 2 có 8 bit : thì mặc định sẽ đếm từ 0 đến 255 (2^8 = 256 trạng thái) thì cờ TF1 tràn

+ Khai báo Counter 1 Mode 2 : TMOD = TMOD|0x60;

+ Mỗi lần chân P3^5 từ ON xuống OFF thì bộ đếm trong thanh ghi tăng lên 1 cho đến khi đạt đến 256 thì tràn.

+ Sử dụng tương tự như counter 0 mode 2

```
#include<reg52.h>
sbit EN = P3^7;
void main(void)
{
  TMOD = TMOD|0x60; //Khai báo Timer1 mode 2 làm Counter 1
  TR1 = 1;
  P2   = 0 ;EN=0;
```

```
  while(1)
   {
        P2 = TL1;
   } }
```

b- Timer 1 Mode 1 làm Counter: (16 bit)

+ Khai báo Counter 1 Mode 1 : TMOD = TMOD|0x50;

+ Sử dụng tương tự như counter 0 mode 1

+ Chương trình đếm 10 sản phẩm thì cờ tràn

```
#include<reg52.h>
#include<stdio.h>
long sanpham ;
long counter_1()
  {
      if(TF1)
       {
          TF1 = 0;
           TH1 =  0xFF; //Gán lại thanh ghi
           TL1 =  246;
           sanpham++;
        }
       return sanpham;
   }
 void main(void)
    {
       TMOD = TMOD|0x50; //Khai báo counter 1 mode 1
       TH1 =  0xFF;
       TL1 =  246; // 10 sản phẩm thì tràn
       TR1 = 1;
        TI=1;
        while(1)
```

```
        {
            printf("san pham = %ld\r",counter_1());
        }
    }
```

III- Interrupt:

1- Interrupt trong:

a- Interrupt Timer 0 Mode 2:

+ Loại này không cần xét cờ tràn, vì mỗi lần tràn sẽ tự động gọi hàm Interrupt 1

```
#include<reg52.h>
sbit Lamp = P2^0;
sbit EN   = P3^7;
#define PERIOD 0x06
bit TIM0;
int msec, sec ,cycle ;
void timer_sec() interrupt 1  //Hàm này được gọi tự động khi cờ TF0 tràn
{
   if(++cycle==4) //1000 us
     {
        cycle = 0;
        if(++msec==1000) //1000000 us = 1s
         {
            msec = 0;
           if(++sec==2) // 2 giây
            {
              TIM0 = ~TIM0;
               sec  = 0;
             }
        }
    }
}
```

```
void main(void)
 {
    //Khai baoTimer 0 Mode 2
    TMOD = TMOD|0x02;
    TH0 =  PERIOD; //Timer = 250 us
    TL0 =  TH0;
    TR0 = 1;
    ET0 = 1;  //Cho phép interrupt Timer 0
    EA=1; //Cho phép interrupt toàn cục
    P2   = 0 ;EN=0;
   while(1)
     {
        Lamp = TIM0;
     }
 }
```

b- Interrupt Timer 0 Mode 1:

```
#include<reg52.h>
sbit Lamp = P2^0;
sbit EN   = P3^7;
bit TIM0;
int msec, sec ,cycle ;
void timer_sec() interrupt 1
 {
   TH0 = 0xFF; //Gán lại thanh ghi
   TL0 = 0x06;
   if(++cycle==4) //1000 us
    {
          cycle = 0;
          if(++msec==1000) //1000000 us = 1s
           {
```

```
            msec = 0;
             if(++sec==2) // 2 giây
             {
               TIM0 = ~TIM0;
               sec  = 0;
          }
      }
   }
 }
void main(void)
{
   TMOD = TMOD|0x01;//Khai báo timer 0 mode 1
    TH0 =  0xFF;
    TL0 =  0x06; //Timer = 250 us
    TR0 = 1;
     ET0 = 1;  //Cho phép interrupt Timer0
     EA=1;
     P2   = 0 ;EN=0;
    while(1)
      {
          Lamp = TIM0;
     }
 }
```

c- Interrupt Timer 1 Mode 2:

```
#include<reg52.h>
sbit Lamp = P2^0;
sbit EN   = P3^7;
bit TIM1;
int msec, sec ,cycle ;
```

```
void timer_sec() interrupt 3
{
    if(++cycle==4) //1000 us
     {
          cycle = 0;
          if(++msec==1000) //1000000 us = 1s
          {
              msec = 0;



               if(++sec==2) // 2 giây
               {
                  TIM1 = ~TIM1;
                   sec  = 0;
               } } }
    }
 void main(void)
{
    TMOD = TMOD|0x20;
    TH1 =  0x06;
    TL1 =  TH1; //Timer = 250 us
    TR1 = 1;
    ET1 = 1;  //Cho phép interrupt Timer1
    EA=1;
    P2   = 0 ;EN=0;
    while(1)
     {
         Lamp = TIM1;
     }
 }
```

d- Interrupt Timer 1 Mode 1:

```
#include<reg52.h>
sbit Lamp = P2^0;
sbit EN   = P3^7;
bit TIM1;
int msec, sec ,cycle ;
void timer_sec() interrupt 3
 {
    TH1 = 0xFF; //Gán lại thanh ghi
    TL1 = 0x06;
    if(++cycle==4) //1000 us
     {
        cycle = 0;
        if(++msec==1000) //1000000 us = 1s
          {
             msec = 0;
             if(++sec==2) // 2 giây
               {
                  TIM1 = ~TIM1;
                  sec  = 0;
               }
          }
       }
     }
 void main(void)
 {
    //Khai baoTimer 1 Mode 1
    TMOD = TMOD|0x10;
    TH1 =  0xFF;
    TL1 =  0x06; //Timer = 250 us
```

```
    TR1 = 1;
    ET1 = 1;  //Cho phép interrupt Timer0
    EA=1;
    P2   = 0 ;EN=0;
    while(1)
     {
       Lamp = TIM1;
     }
  }
```

2- Interrupt ngoài:

a- Interrupt 0:

+ Sử dụng interrupt 0 ngoài (INT0) mỗi khi kích vào chân P3^2.

+ Hàm interrupt 0 được gọi sau mỗi lần kích vào chân P3^2.

- IT0 = 1; Chân INT0 tác động cạng xuống.
- IT0 = 0; Chân INT0 tác động mức thấp.

Thí dụ chương trình đếm sản phẩm:

```
#include<reg52.h>
sbit EN   = P3^7;
long san_pham;
void bo_dem() interrupt 0
 {
   san_pham++;
 }
void main(void)
{
   IT0=1;  //Tac dong canh xuong
   EX0 = 1;  //Cho phép interrupt 0 ngoai
   EA=1;
   P2   = 0 ;EN=0;
```

```
  while(1)
    {
          P2 = san_pham;
    }
}
```

b- Interrupt 1:

+ Sử dụng interrupt 1 ngoài (INT1) mỗi khi kích vào chân P3^3.

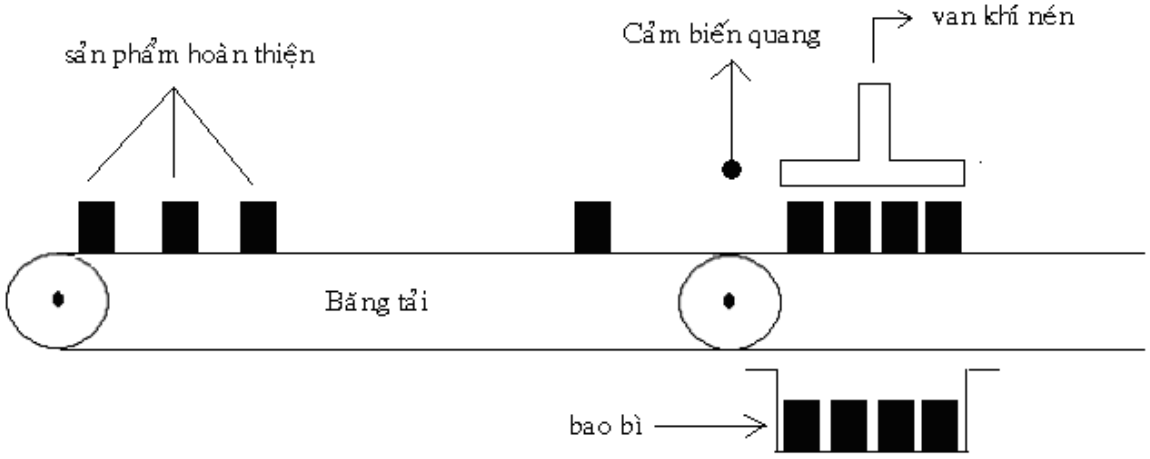
+ Hàm interrupt 2 được gọi sau mỗi lần kích vào chân P3^3.

- IT1 = 1; Chân INT1 tác động cạng xuống.
- IT1 = 0; Chân INT1 tác động mức thấp.

```
#include<reg52.h>
#include<stdio.h>
sbit EN   = P3^7;
long san_pham;
void bo_dem() interrupt 2
{
   san_pham++;
}
void main(void)
{
   IT1=1;  //Tac dong canh xuong
   EX1 = 1;  //Cho phép interrupt 1 ngoai
   EA=1;
   P2   = 0 ;EN=0;
   while(1)
     {
          P2 = san_pham;
     }
}
```

Ứng dụng:

Cho hệ thống đóng gói sản phẩm như hình vẽ:



- Các sản phẩm hoàn thiện được băng tải chuyển đến thiết bị đóng bao, cứ 4 sản phẩm thì được đóng vào một bao.
- Đúng 4 sản phẩm thì van đẩy làm việc trong 2 giây.

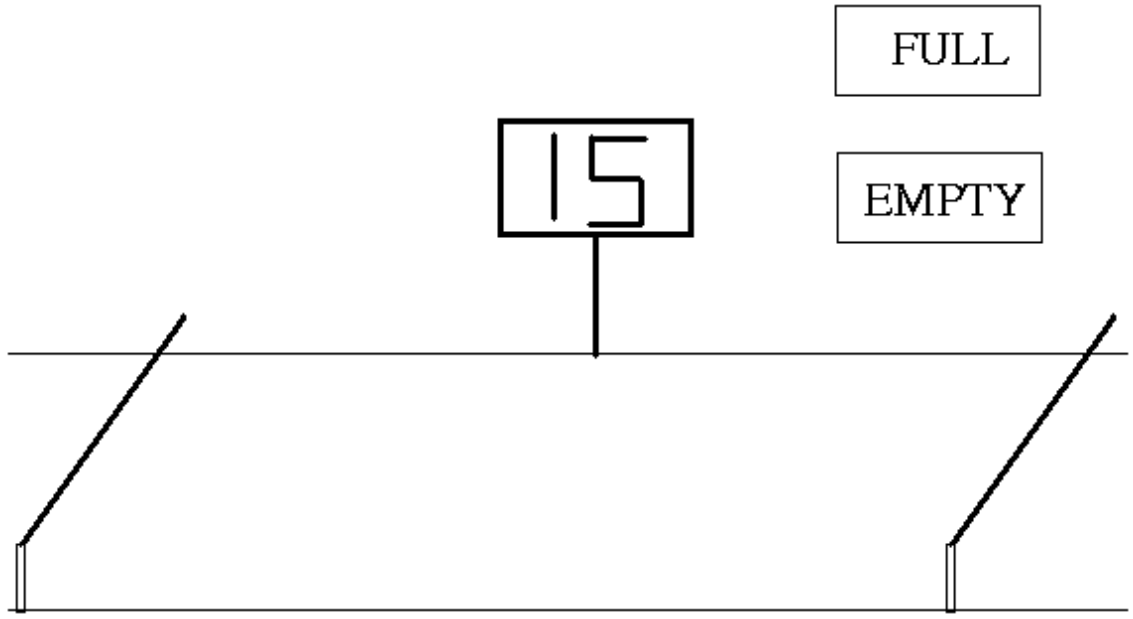
+ Cảm biến được đưa vào chân P3^2 (INT0)

```
#include<reg52.h>
#include<stdio.h>
sbit start       = P1^0;
sbit stop        = P1^1;
sbit den_do      = P2^0;
sbit bang_tai    = P2^1;
sbit van         = P2^2;
sbit EN          = P3^7;
int san_pham;
bit Relay,TIM0=1;
int msec, sec ,cycle ;
bit timer_sec(int giay)
 {
   if(TF0)
    {
```

```
        TF0 = 0;
        if(++cycle==4) //1000 us
         {
            cycle = 0;
            if(++msec==1000) //1000000 us = 1s
             {
               msec = 0;
              if(++sec==giay)
               {
                 TIM0 = 0;
                 sec  = 0;  } } } }
    return TIM0;
 }
void bo_dem(void) interrupt 0
{
  if(++san_pham==4)
    {
        Relay = 1;
        san_pham = 0; //Reset bộ đếm
    }
 }
void main(void)
{
    //Khai báo Timer 0 Mode 2 (8 bit)
  TMOD = TMOD|0x02;
  TH0 = 0x06;  TL0 = TH0;
  TR0=1; //Cho phép Timer 0 hoạt động
  //Khai báo INT0 ngoài
  IT0=1;  //Tác động cạng xuống
  EX0 = 1;  //Cho phép interrupt 0 ngoài
```

```
  EA=1;
  P2   = 0 ;EN=0;
  while(1)
   {
         den_do = (start|den_do)&(!stop);
         bang_tai = (!van)&den_do;
         van = Relay&timer_sec(2)&den_do;
        if(!TIM0)//Reset Timer0
         {
            TIM0 = 1;
             Relay = 0;
         }     }
}
```

Bài tập: Hãy viết chương trình cho bãi đậu xe như sau



Bãi chứa được 15 xe:

- Khi còn trống thì sáng đèn EMPTY.
- Khi đầy thì sáng đèn FULL.
- Mỗi khi có xe vào bãi thì mạch đếm tự động tăng lên 1.
- Mỗi khi có xe ra bãi thì mạch đếm tự động giảm xuống 1.

Chương 5 :

LCD VÀ LED

I- LCD 4 line 20 ký tự:

1 Địa chỉ LCD:

Hàng 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	98	92	93
Hàng 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
Hàng 3	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
Hàng 4	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

2 Các Frame truyền thông của LCD:

+ Frame 1:

Xóa LCD và đưa con trỏ về hàng 1 cột 1

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	1

+ Frame 2:

Đưa con trỏ về hàng 1 cột 1

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	1	X

+ Frame 3:

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	1	ID	S

ID = 1 Tự động tăng địa chỉ mỗi khi Đọc / Viết đến LCD.

S = 1 Tự động shift con trỏ về phải.

ID = 0 Giảm địa chỉ mỗi khi Đọc / Viết đến LCD.

S = 0 Tự động shift con trỏ về trái.

+ Frame 4:

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	D	C	B

D = 1 cho phép hiển thị Data lên LCD.

D = 0 không cho phép hiển thị Data lên LCD.

C = 1 cho phép hiển thị Cursor.

C = 0 không cho phép hiển thị Cursor.

B = 1 cho phép Blink Cursor

B = 0 không cho phép Blink Cursor

+ Frame 5:

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	DL	N	F	X	X

DL = 1 chiều dài Data mỗi lần Đọc / Viết là 8 bit.

DL = 0 chiều dài Data mỗi lần Đọc / Viết là 4 bit.

N = 0 số line = 1.

N = 1 số line = 2.

F = 1 đặt font chữ 5x10 điểm.

F = 0 đặt font chữ 5x7 điểm.

+ Frame 6: Đọc cờ bận BF

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
0	1	BF	A	A	A	A	A	A	A

BF = 1 LCD đang bận.

BF = 0 chấp nhận lệnh tiếp theo.

AAAAAAA = là địa chỉ counter.

+ Frame 7: Viết Data đến LCD

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
1	0	D	D	D	D	D	D	D	D

DDDDDDDD = là 8 bit Data cần viết vào LCD

+ Frame 8: Đọc Data từ LCD

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
1	1	D	D	D	D	D	D	D	D

DDDDDDDD = là 8 bit Data Đọc được LCD

3 <u>Khởi LCD</u>: Qua 4 bước tuần tự

+Bước 1:

RS = 0 RW = 0 E = 0;

Data = 0x38; // Frame 5

E = 1 ;

delay(30);

E = 0;

+Bước 2:

RS = 0 RW = 0 E = 0;

Data = 0x0F; // Frame 4

E = 1 ;

delay(30);

E = 0;

+Bước 3:

RS = 0 RW = 0 E = 0;

Data = 0x06; // Frame 3

E = 1 ;

delay(30);

E = 0;

+Bước 4 Xóa LCD

RS = 0 RW = 0 E = 0;

Data = 0x01; // Frame 1

E = 1 ;

delay(30);

E = 0;

Thí dụ: Chương trình đồng hồ báo thức

buff[0] = Chứa hàng đơn vị của giây.

buff[1] = Chứa hàng chục của giây.

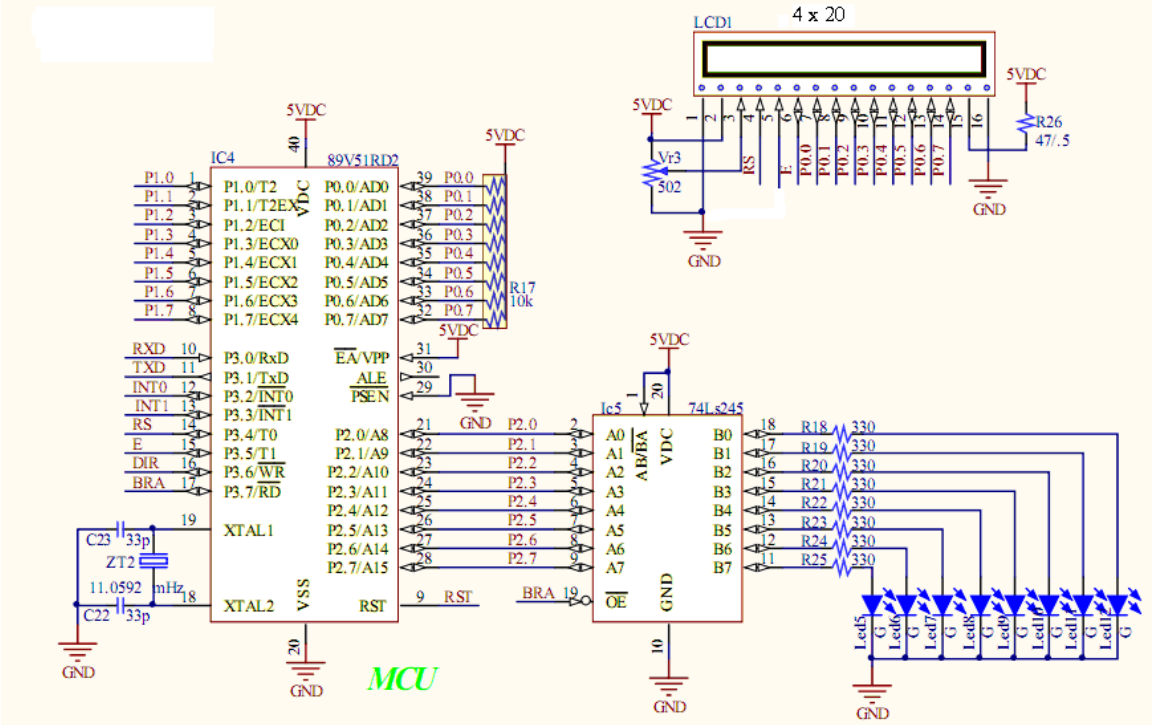
buff[2] = Chứa hàng đơn vị của phút.

buff[3] = Chứa hàng chục của phút.

buff[4] = Chứa hàng đơn vị của giờ.

buff[5] = Chứa hàng chục của giờ.

+ Sơ đồ phần cứng:



```
//Bắt đầu chương trình
#include<reg52.h>
#include<string.h>
#include<intrins.h>
#include<stdio.h>
#define PERIOD 0x06
sbit RS   = P3^4;
sbit E    = P3^5;
sbit EN   = P3^7;
sbit lamp = P2^0;
//Khai báo biến
char buff_time[6],*str[]={"OFF","ON"};
bit display;
unsigned char intcycle,buf_lcd[20];
//Khai báo biến kiểu cấu trúc
struct time
{
   unsigned char    hour; // hour
   unsigned char     min; // minute
   unsigned char     sec; // second
   unsigned int     msec; // milli second
 }  time;
//Hàm xuất 1 ký tự ra LCD
void write_char(char Hcode,bit mode)
{
   RS=mode;
```

```
    P0=Hcode; // Data chứa trong P0
    E=1;
    _nop_();//Delay phần cứng cho 1 chu kỳ CPU
   _nop_();
    E=0;
}
//Hàm xuất 1 chuỗi ra LCD
void write_str(char str[],char row)
{
    int i;
    write_char(row,0);

    for(i=0;i<strlen(str);i++)
    {
       write_char(str[i],1);
    }
 }
//Hàm xóa và khởi tạo lại LCD
void start_LCD()
{
     int i;
     char str[3]={0x38,0x0C,0x01};
     for(i=0;i<3;i++)
      {
          write_char(str[i],0);
      }}
```

```
//Hàm tạo giao diện
void giao_dien(char str1[],char str2[],char str3[],char str4[])
{
    write_str(str1,0X80); //Hàng 1
    write_str(str2,0XC0); //Hàng 2
    write_str(str3,0X94); //Hàng 3
    write_str(str4,0XD4); //Hàng 4
}
//Hàm mã giây
void magiay(void)
{
    if (++buff_time[0]>9)
     {
       buff_time[0] = 0; //Hàng đơn vị của giây
       buff_time[1]++;  //Hàng chục của giây
     }
}
//Hàm mã phút
void maphut(void)
{
      buff_time[1] = 0;//Xóa hàng chục của giây.
      if (++buff_time[2]>9)
      {
        buff_time[2] = 0; //Hàng đơn vị của phút
        buff_time[3]++; //Hàng chục của phút.
      }}
```

```
//Hàm mã giờ
void magio(void)
{
    buff_time[3] = 0; //Xóa hàng chục của phút.
    if (++buff_time[4]>9)
    {
       buff_time[4] = 0; //Hàng đơn vị của giờ.
       buff_time[5]++;  //Hàng chục của giờ.
    }
}
//Hàm interrupt trong
void timer0(void) interrupt 1
{
    if (++intcycle == 4)//1 msec = 4 * 250 usec cycle
     {
        intcycle = 0;
        if (++time.msec == 1000) //Mỗi 1 giây
         {
            time.msec = 0;
            magiay();
            display = 1; //Xuất ra LCD sau mỗi giây.
           if (++time.sec == 60) // Sau 1 phút
            {
                  time.sec = 0;
                  maphut();
                  lamp =  !lamp ; //Báo thức sau mỗi phút.
```

```
                if (++time.min == 60)// Sau 1 giờ
                 {
                    time.min = 0;
                    magio();
                   if (++time.hour == 24) //update hour counter
                     {
                        time.hour = 0;
                        buff_time[4] = 0;
                        buff_time[5] = 0;
                    }}  }  } } }
void main(void)
{
   TMOD = TMOD | 0x02;   //Timer 0, mode 2
   TH0 = PERIOD ;        // set timer period
   TR0 = 1;              // start timer 0
   ET0 = 1;             // enable timer 0 interrupt
   EA  = 1;            // global interrupt enable
   start_LCD();
   giao_dien("TRUONG DHSPKT TPHCM","DONG HO BAO GIO","00 : 00 : 00",
                 "BAO THUC : OFF");
   P2=0;  EN=0;
   while(1)
   {
     if(display)
      {
         sprintf(buf_lcd,"%bd%bd : %bd%bd : %bd%bd",
```

```
                buff_time[5],buff_time[4],buff_time[3],
                buff_time[2],buff_time[1],buff_time[0]);
          write_str(buf_lcd,0X94);
           sprintf(buf_lcd,"%3s",str[lamp]);//Chuổi có 3 ký tự
           write_str(buf_lcd,0xD4+11);
          display = 0;
        } } }
```

+ Trường hợp mô phỏng trên máy tính:

```
sprintf(buf_lcd,"%bd%bd : %bd%bd : %bd%bd  %3s\r",
            buff_time[5],buff_time[4],buff_time[3],
            buff_time[2],buff_time[1],buff_time[0],str[lamp]);
printf(buf_lcd);
```

Ứng dụng: Hệ thống đo chiều dài sản phẩm

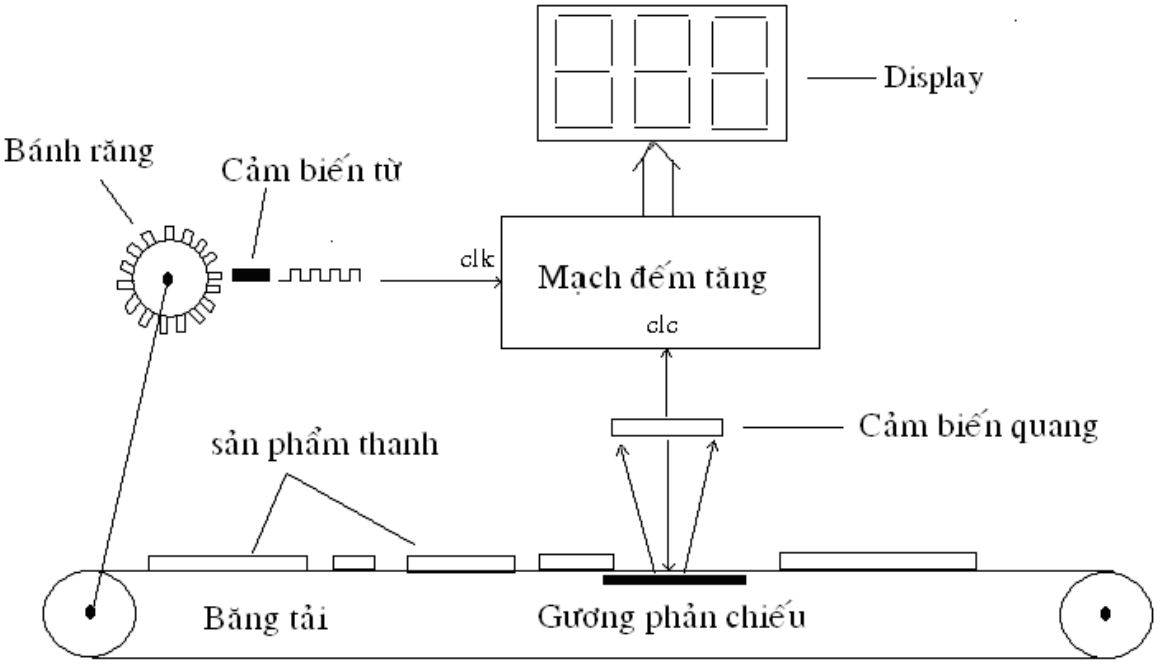
Cho hệ thống như hình vẽ, với nguyên lý hoạt động như sau:

- Khi có thanh sản phẩm che cảm biến quang thì bộ đếm xóa về 0, đồng thời báo hiệu bắt đầu 1 độ dài cần đo và bộ đếm sẽ đếm tăng.
- Cho đến khi thanh sản phẩm đi qua hết cảm biến thì báo hiệu kết thúc một độ dài cần đo.
- Bộ đếm sẽ đếm số xung được phát ra tại cảm biến từ mỗi khi có một răng tiếp cận ngang cảm biến.
- Nếu gọi khỏang cách của mỗi răng đều nhau là d = 1 cm, n là số xung được phát ra. Ta có chiều dài L của thanh sản phẩm cần đo là:

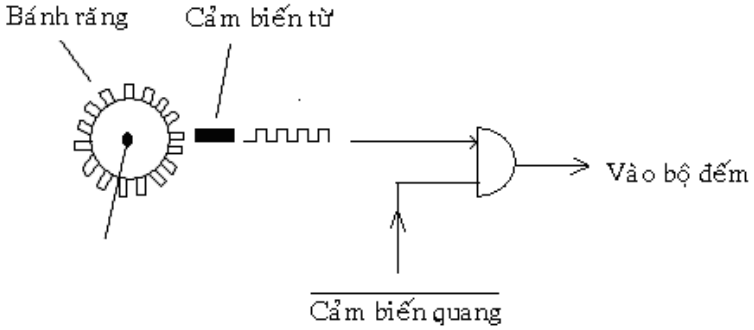
L = n x d (cm)

+ Phương trình điều khiển:

- Cảm biến quang tác động mức thấp
- Cảm biến quang =0, xóa bộ đếm và bắt đầu đếm số xung.
- Cảm biến quang = 1, kết thúc độ dài cần đo.



Vậy dùng cảm biến quang làm xung mở và đóng cổng như sau:



- Cảm biến quang = 0, mở cổng
- Cảm biến quang = 1, đóng cổng

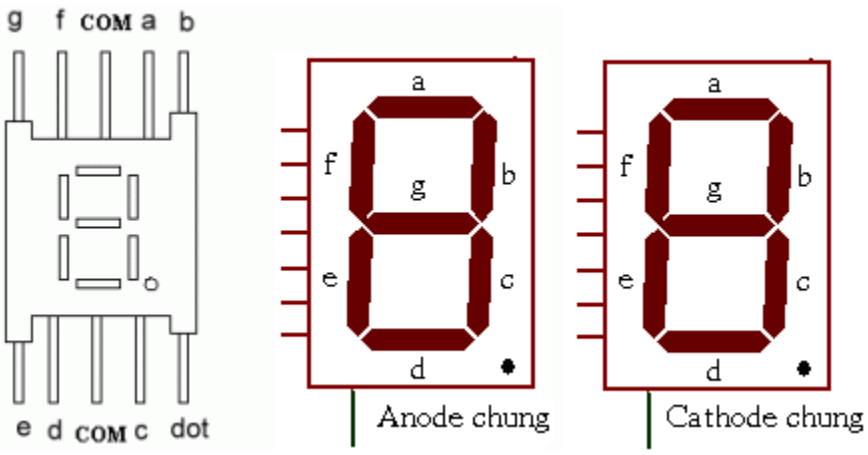
```
//Chương trình đo độ dài, cảm biến quang = INT0
#include<reg52.h>
#include<string.h>
#include<intrins.h>
#include<stdio.h>
sbit RS   = P3^4;
sbit E    = P3^5;
```

```
sbit EN   = P3^7;
sbit Lamp = P2^0;
sbit cbt     = P1^0; //Cảm biến từ
long L; //chiều dài
unsigned char buf_lcd[20];
//ham xuat 1ky tu LCD
void write_char(char Hcode,bit mode)
{
    RS=mode;
    P0=Hcode;
    E=1;
    _nop_();_nop_();  E=0;
}
//ham xuat chuoi ra LCD
void write_str(char str[],char row)
{
    int i;
    write_char(row,0);
    for(i=0;i<strlen(str);i++)
    {
       write_char(str[i],1);
    } }
//ham khoi tao LCD
void start_LCD()
{
     int i;
```

```
    char str[3]={0x38,0x0C,0x01};//khai bao cac thong so cho LCD
    for(i=0;i<3;i++)
     {
     write_char(str[i],0);
     }}
//Ham tao giao dien
void giao_dien(char str1[],char str2[],char str3[],char str4[])
{
    write_str(str1,0X80);
    write_str(str2,0XC0);
    write_str(str3,0X94);
    write_str(str4,0XD4);
}
//Cảm biến quang để reset bộ đếm
void reset(void) interrupt 0
{
    L=0;  Lamp=0;
}
void main(void)
{
    IT0=1;EX0=1;EA=1;
    start_LCD();
    giao_dien("TRUONG DHSPKT TPHCM","DO CHIEU DAI",
                "POWER = ON","L = 0");
    P2=0;
    EN=0;
```

```
        while(1)
        {
            while (!INT0&cbt) ; //INT0 = Cảm biến quang
             while (!(!INT0&cbt));
             L++;
             sprintf(buf_lcd,"%5ld\r",L);
             write_str(buf_lcd,0XD4+5);
             Lamp=1;
        }
 }
```

II- LED 7 đoạn:



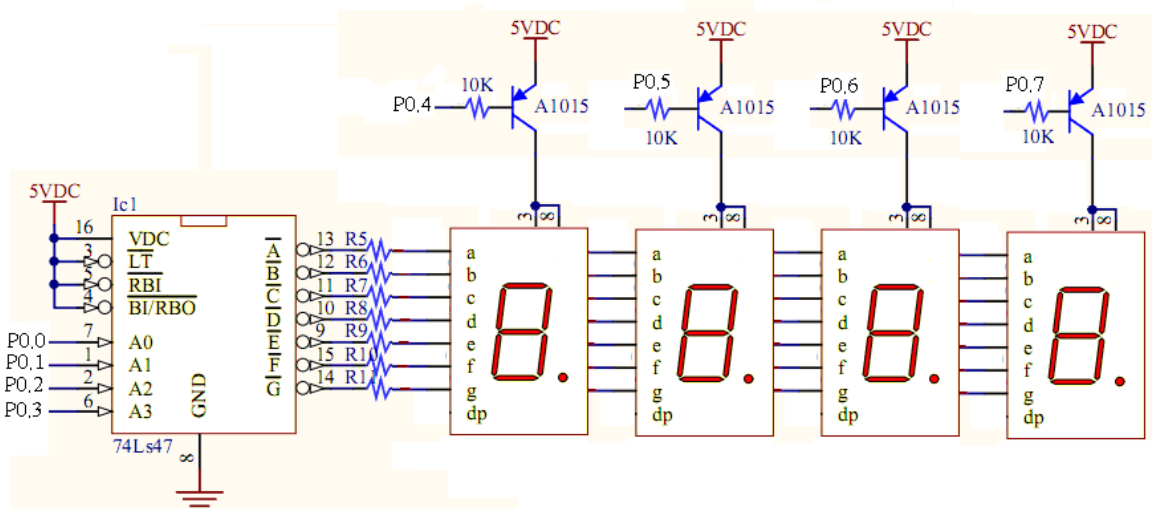
+ Sơ đồ phần cứng:

- IC 7447 giải mã BCD ra đèn 7 đoạn.

- LED 7 đoạn dùng loại có anod chung.

- P0^0 -> P0^3 làm data cho đèn.

- P0^4 -> P0^7 làm anod cho đèn, tác động mức thấp.



+ Bảng tra anod cho đèn:

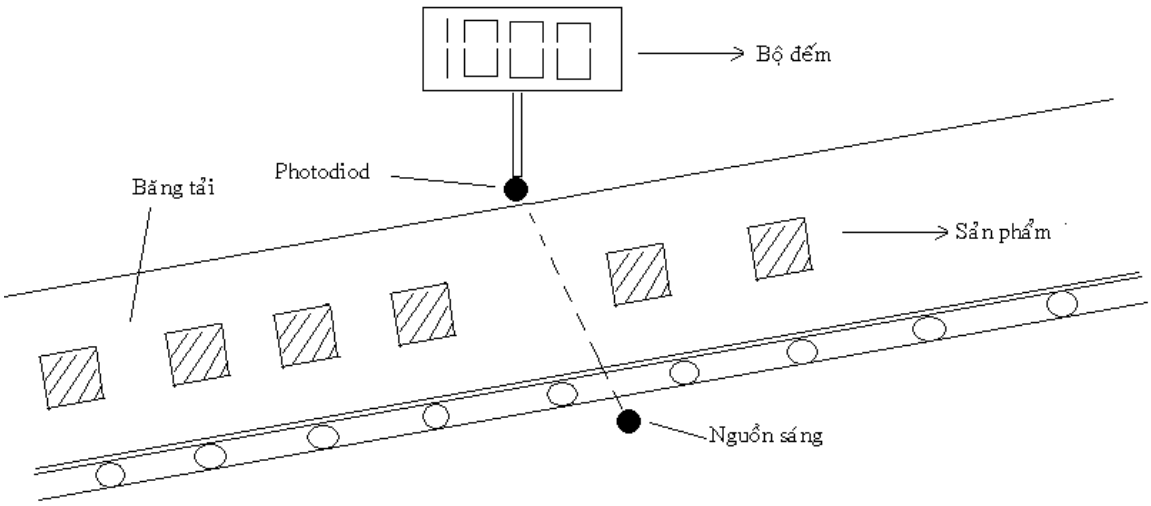
Hàng ngàn P0^4	Hàng trăm P0^5	Hàng chục P0^6	Đơn vị P0^7	Giá trị
1	1	1	0	0x70
1	1	0	1	0xB0
1	0	1	1	0xD0
0	1	1	1	0xE0

+ Giải thuật quét tuần tự 4 đèn:

- Mắt người có độ lưu ảnh 1/16 giây, như vậy mỗi lần chỉ cần cho sáng duy nhất 1 đèn sao cho thời gian quét lần lượt 4 đèn không quá 1/16 giây thì mắt không phân biệt được mà có cảm giác sang cả 4 đèn.
- Có 4 đèn nên khả năng đếm từ 0 đến 9999.

+Chương trình quét đèn: Hệ thống đếm sản phẩm

- Dùng P1^0 = 1 để reset bộ đếm.
- Cảm biến quang phát hiện sản phẩm di chuyển trên băng tải.
- Sử dụng INT0

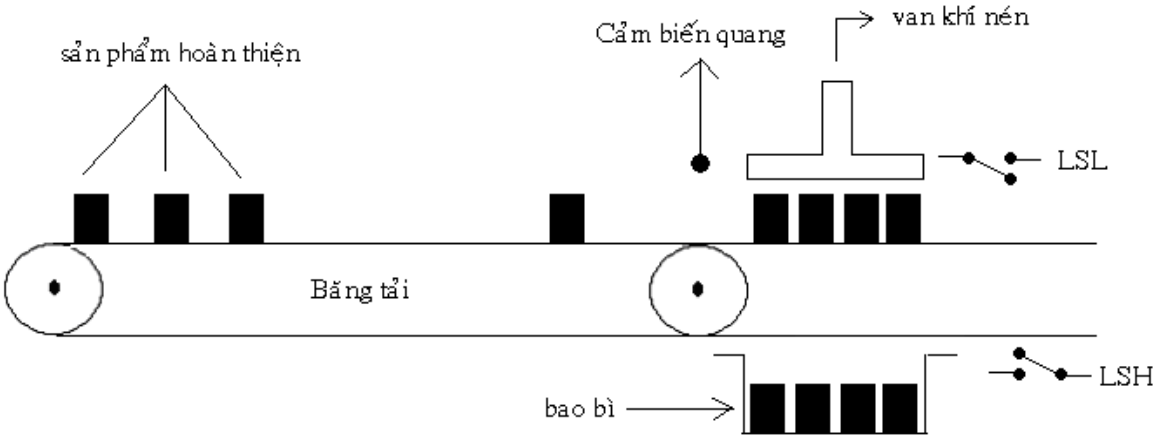



```
#include<reg52.h>
unsigned int sanpham ;
sbit reset = P1^0;
char den[4]     = {0x70,0xB0,0xD0,0xE0};//Khai báo bảng tra cho anod đèn.
//Delay phần mềm 1ms, cho thạch anh  11.0592 MHz
void delay_ms(unsigned int count)
{
    unsigned int i;
    while(count)
     {
        i = 115;
        while(i>0)
            i--;
        count--;
      }
}
```

```
void quetden(int anodden,unsigned int dataden)
{
    P0 = dataden;
    P0 = P0|den[anodden];
}
void LED()
{
    int i ; long k;
    for(i=0,  k = 10 ;i<=3;i++, k = k*10 ) //4 digit
     {
        quetden(i,(sanpham%k)/(k/10)); //Tách ra từng Digit để hiển thị
        delay_ms(2);
     }
}
void bo_dem(void) interrupt 0
{
     sanpham++;
}
void main(void)
{
    IT0=1;EX0=1;EA=1;
    while(1)
    {
        LED();
        sanpham = sanpham*(unsigned)(!reset); //Reset bộ đếm.
    }}
```

Ứng dụng: Cho hệ thống đóng gói sản phẩm như hình vẽ:

- Các sản phẩm hoàn thiện được băng tải chuyển đến thiết bị đóng bao, cứ 4 sản phẩm thì được đóng vào một bao.
- Băng tải hoạt động khi van nằm tại vị trí LSL = 1;
- Van hoạt động khi đủ 4 sản phẩm, và van = 0 khi LSH=1;
- Cảm biến quang được đưa vào chân P3^2 (INT0)
- LSH được đưa vào chân P3^3 (INT1)
- Xóa bộ đếm thùng P1^3.
- Xem số thùng P1^4.




```
#include<reg52.h>
sbit EN   = P3^7;
sbit start    = P1^0;
sbit stop     = P1^1;
sbit LSL      = P1^2;
sbit reset    = P1^3; //Xóa bộ đếm thùng
sbit display  = P1^4; //Xem bộ đếm thùng
sbit den_do   = P2^0;
sbit bang_tai = P2^2;
sbit van      = P2^3;
//sbit LSH      = INT1;  sbit cam_bien_quang  = INT0;
bit  Relay;
```

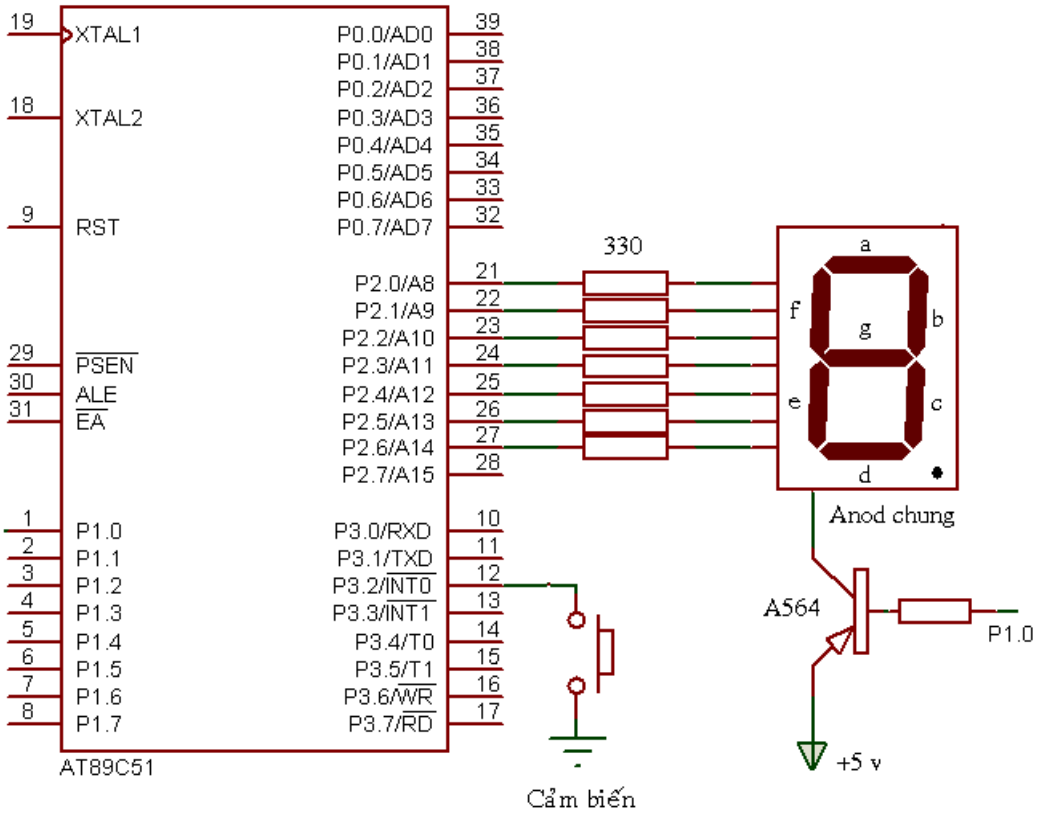
```
unsigned int sanpham, sothung;
//Khai báo bảng tra cho anode đèn
char den[4]     = {0x70,0xB0,0xD0,0xE0};
//Delay phần mềm 1ms, cho thạch anh  11.0592 MHz
void delay_ms(unsigned int count)
{
      unsigned int i;
      while(count)
       {
         i = 115;
         while(i>0)
            i--;
         count--;
       }
}
void quetden(int anodden,unsigned int dataden)
{
      P0 = dataden;
      P0 = P0|den[anodden];
}
void LED(unsigned int ndata)
{
      int i ; long k;
      for(i=0,  k = 10 ;i<=3;i++, k = k*10 ) //4 digit
       {
          quetden(i,(ndata%k)/(k/10)); //Tách từng Digit
          delay_ms(2);
       }
}
```

```
void bo_dem(void) interrupt 0
{
       if(++sanpham==4)
       {
          Relay = 1;
          sanpham = 0;
        }
}
// bộ đếm thùng
void bo_dem_tank(void) interrupt 2
{
    sothung++;
}
void chong_doi() //Chống dội cho INT1
{
    static int solannha,solannhan;
    static bit FlagRecv;
    if(!INT1)
     {
          solannha = 0;
          if(!FlagRecv)
            {
               if(++solannhan==5)
                     FlagRecv = 1;
            }
     }
    else
    {
        solannhan=0;
```

```
        if(FlagRecv)
         {
            if(++solannha == 5)
             {
                 FlagRecv = 0;
                 EX1=1;INT1 = 0;EX1=0;INT1 = 1;
             } }
        }
 }
void main(void)
{
    IT0=1;EX0=1;
    IT1=1;EX1=0;
    EA=1;
    EN = 0; P2=0;
    while(1)
    {
        den_do = (start|den_do)&(!stop);
         bang_tai = (!van)&LSL&den_do;
         van = Relay&den_do;
        chong_doi();
         if(INT1)
             Relay = 0; //Reset bộ đếm
         if(display)//Xem bộ đếm thùng
            LED(sothung);
         else//Xem bộ đếm sản phẩm
            LED(sanpham);
         if(reset)//Xóa bộ đếm thùng
            sothung = 0;
    }}
```

III- Bài tập:

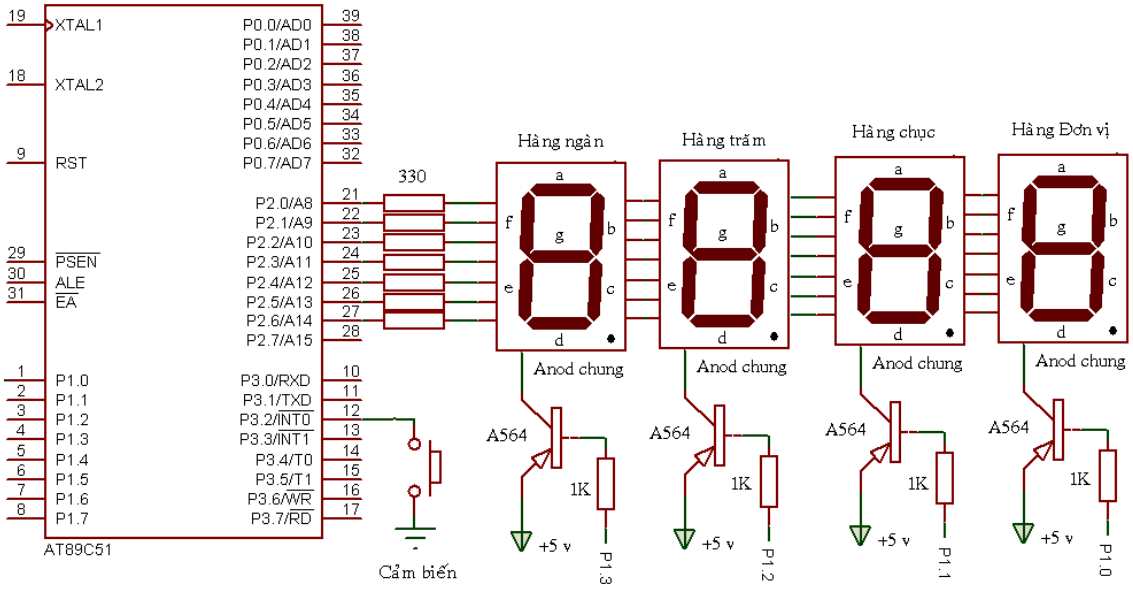
Bài 1: Viết chương trình quét LED 7 đoạn không dùng ic giải mã 7447 như sau:



+ Bảng tra giá trị đèn:

	g	f	e	d	c	b	a		
P2^7	P2^6	P2^5	P2^4	P2^3	P2^2	P2^1	P2^0	Hiển thị	Giá trị
1	1	0	0	0	0	0	0	Số 0	0xC0
1	1	1	1	1	0	0	1	Số 1	0xF9
1	0	1	0	0	1	0	0	Số 2	0xA4
1	0	1	1	0	0	0	0	Số 3	0xB0
1	0	0	1	1	0	0	1	Số 4	0x99
1	0	0	1	0	0	1	0	Số 5	0x92
1	0	0	0	0	0	1	0	Số 6	0x82
1	1	1	1	1	0	0	0	Số 7	0xF8
1	0	0	0	0	0	0	0	Số 8	0x80
1	0	0	1	0	0	0	0	Số 9	0x90

Bài 2: Viết chương trình quét 4 LED 7 đoạn không dùng ic giải mã 7447:



+ Bảng anode đèn:

Hàng ngàn P1.3	Hàng trăm P1.2	Hàng chục P1.1	Đơn vị P1.0	Giá trị
1	1	1	0	0x0E
1	1	0	1	0x0D
1	0	1	1	0x0B
0	1	1	1	0x07

Chương 6 :

KEYBOARD VÀ ADC

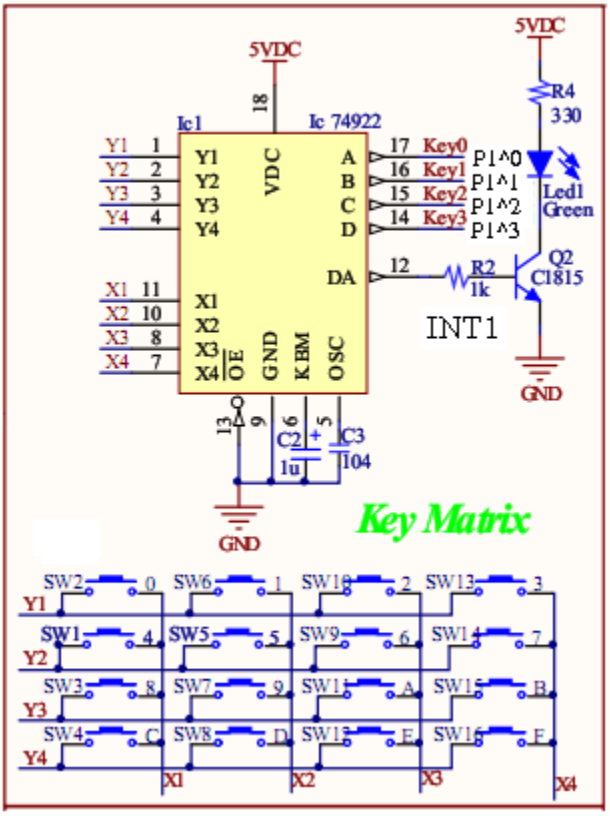
I- Keyboard:

1. Giải mã bàn phím:

- IC 74922 dùng giải mã bàn phím

- Chân 12 lên 1 được đưa vào INT1 để báo hiệu đã giải mã xong 1 phím được nhấn.

- Ngỏ ra A,B,C,D đưa vào P1^0 đến P1^3.



2. Chương trình ứng dụng:

+ Chương trình nhận phím và hiển thị lên LCD

```
#include<reg52.h>
#include<intrins.h>
#include<string.h>
```

```
#include<stdio.h>
#define low_byte(x)   (x&0x0F) // Chỉ lấy 4 bit thấp P1
bit nFlags_phim; //Cờ phát hiện phím nhấn
unsigned char buf_lcd[20];
unsigned int ndata;
sbit RS   = P3^4;
sbit E    = P3^5;
//Chương trình LCD đã viết ở phần trước
void write_char(char Hcode,bit mode) //Hàm xuất ký tự
{
}
void write_str(char str[],char row) //Hàm xuất chuỗi
{
 }
void start_LCD()//Hàm khởi tạo LCD
{
 }
void giao_dien(char str1[],char str2[],char str3[],char str4[])//Hàm tạo giao diện
{
}
void keyboard(void) interrupt 2
{
    nFlags_phim = 1;
    ndata = low_byte(P1);
}
```

```
void main()
{
  //Khai báo INT1
    IT1 = 1;EX1 = 1; EA  = 1;
   start_LCD();
    giao_dien("TRUONG DHSPKT TPHCM","BO MON DKC",
                "KEYBOAR","PHIM = 0");
     while(1)
        {
          if(nFlags_phim)
           {
                nFlags_phim = 0;
                sprintf(buf_lcd,"%2d",ndata);
                 write_str(buf_lcd,0XD4+7);
            }
        }
 }
```

+ Thiết kế tủ điện điều khiển 8 tải từ P2^0 đến P2^7

- Phím 15 xóa tất cả đèn.
- Các phím từ 0 đến 7 đều có hai chức năng ON và OFF tải tương ứng.

```
#include<reg52.h>
#include<intrins.h>
#include<string.h>
#include<stdio.h>
#include<math.h>
#define low_byte(x)   (x&0x0F)
```

```
bit nFlags_phim;
unsigned char buf_lcd[20];
unsigned int  ndata,n;
unsigned char nLamp[] = {0,0,0,0,0,0,0,0};//Trạng thái 8 đèn
sbit RS   = P3^4;
sbit E    = P3^5;
sbit EN   = P3^7;
void keyboard(void) interrupt 2
{
    nFlags_phim = 1;
    ndata = low_byte(P1);
}
//Đổi số nhị phân sang thập phân
unsigned int bin2dec(unsigned char bin[])
{
    unsigned int n,nbuf = 0;
    for (n=0;n<8;n++)
     {
        nbuf = nbuf + (pow(2,n)*bin[n]);
     }
  return nbuf ;
}
void main()
{
    IT1 = 1;EX1 = 1; EA  = 1;
    start_LCD();
```

```
giao_dien("TRUONG DHSPKT TPHCM","BO MON DKC",
          "KEYBOAR = ","TAI = 0");
P2=0;EN=0;
while(1)
 {
    if(nFlags_phim)
     {
          nFlags_phim = 0;
          if(ndata<8)
           {
             nLamp[ndata] = !nLamp[ndata];
             P2 = bin2dec(nLamp);
           }
           if(ndata==15)//Xóa tất cả đèn
           {
              for (n=0;n<8;n++)
               {
                  nLamp[n] = 0;
               }
              P2 = bin2dec(nLamp);
           }
           sprintf(buf_lcd,"%2d",ndata);
           write_str(buf_lcd,0XD4+10);
           sprintf(buf_lcd,"%3d",(unsigned)P2);
           write_str(buf_lcd,0XD4+5);
     }  } }
```

+ Chương trình nhận nhiều phím và thực hiện cộng dồn.

- Phím 14 xóa bộ đếm.
- Phím 15 xuất ra P2.
- Các phím từ 0 đến 9 thực hiện cộng.

```
#include<reg52.h>
#include<intrins.h>
#include<string.h>
#include<stdio.h>
#define low_byte(x)   (x&0x0F)
bit nFlags_phim;      unsigned char buf_lcd[20];
unsigned long  ndata,san_pham;
sbit RS   = P3^4; sbit E    = P3^5; sbit EN   = P3^7;
void keyboard(void) interrupt 2
{
   nFlags_phim = 1;
   ndata = low_byte(P1);
}
//xét phím chức năng và cộng dồn
void KeyFcn(unsigned long ma_phim)
{
   if(ma_phim<10)//Chỉ nhận từ 0 đến 9
   {
     san_pham =  10*san_pham + ma_phim;
     if(san_pham>=256)
          san_pham = 0;
   } }
```

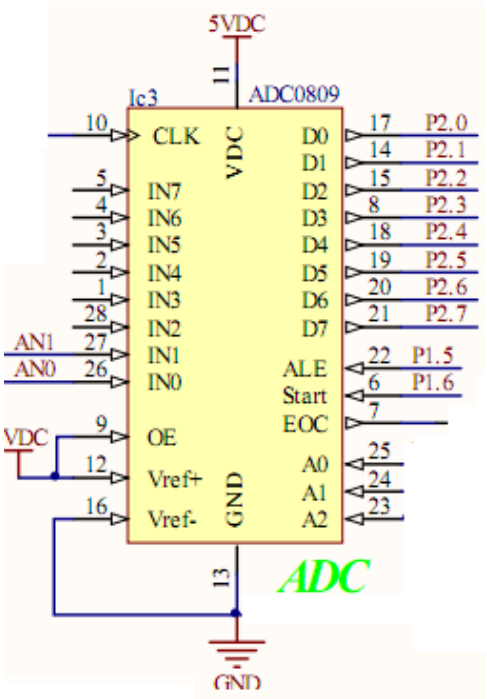
```
void main()
{
    IT1 = 1;EX1 = 1; EA  = 1;
    start_LCD();
    giao_dien("TRUONG DHSPKT TPHCM","BO MON DKC",
                "KEYBOARD","SAN PHAM = 0");
    P2=0;EN=0;
    while(1)
    {
       if(nFlags_phim)
        {
           nFlags_phim = 0;
           if(ndata<10)
                KeyFcn(ndata);
           sprintf(buf_lcd,"%3ld",san_pham);
           write_str(buf_lcd,0XD4+10);
           if(ndata==14)//Xóa
            {
                san_pham = 0;
                P2 = san_pham;
            }
           if(ndata==15)//enter
            {
              P2 = san_pham;
              san_pham=0;
            }  } } }
```

II. ADC:

1. ADC 0809:

- Có 8 ngỏ vào từ IN0 đến IN7.

- Data ra 8 bit nhị phân từ D0 đến D7.



- Chọn ngỏ vào được thực hiện bởi Ao, A1 và A2

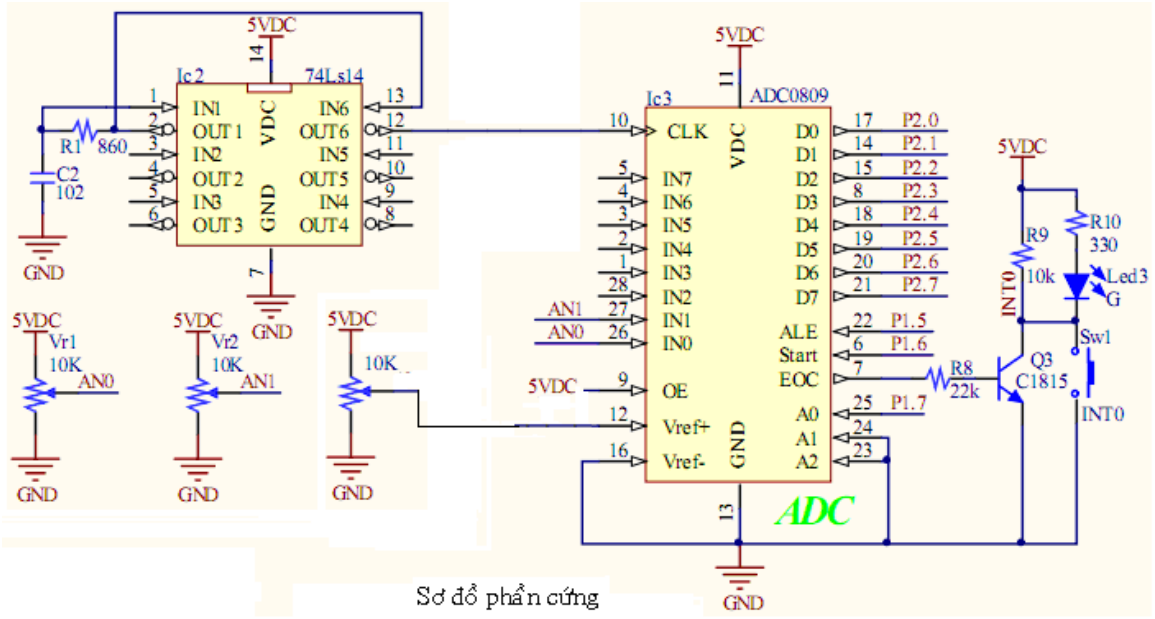
Ao	A1	A2	Ngỏ vào được chọn
0	0	0	IN0
0	0	1	IN1
0	1	0	IN2
0	1	1	IN3
1	0	0	IN4
1	0	1	IN5
1	1	0	IN6
1	1	1	IN7

- Chân ALE đi từ 1 xuống 0 để chốt địa chỉ ngỏ vào được chọn.
- Chân start đi từ 1 xuống 0 để bắt đầu quá trình chuyển đổi.
- Chân EOC lên cao báo hiệu kết thúc quá trình chuyển đổi.

2. Chương trình ứng dụng:

a. Giải thuật chuyển đổi ADC qua 4 bước:

- Chọn ngỏ vào cần chuyển đổi.
- Chân ALE đi từ 1 xuống 0 để chốt địa chỉ ngỏ vào được chọn.
- Chân start đi từ 1 xuống 0 để bắt đầu quá trình chuyển đổi.
- Chân EOC lên cao báo hiệu kết thúc quá trình chuyển đổi.



Sơ đồ phần cứng

- Nếu gọi Vin là điện thế Analog cần chuyển đổi ở ngỏ vào, ta có điều kiện sau:

$(V_{in})_{\max} \le V_{ref} \le 5v$, với $(V_{in})_{\max}$ là điện thế vào toàn giai.

- Nếu thỏa điều kiện trên thì 1 ADC 8 bit có khả năng chuyển đổi được $2^8 = 256$ trạng thái tương ứng ngỏ ra D0 đến D7 là Vo đi từ:

$$(00000000)_2 = (0)_{10} \quad -> (11111111)_2 = (255)_{10}$$

- Điện thế Analog vào đi được cho bởi công thức sau:

$$V_{in} = \frac{Vo}{2^n - 1} V_{ref}$$, n là số bit của ADC.

Thí dụ:

Một ADC có n = 8 bit, Vref = 5v. Tìm ngỏ ra Vo của ADC khi Vin = 2.5v.

+Giải:

$$V_{in} = \frac{Vo}{2^n - 1} V_{ref} \Rightarrow V_o = \frac{Vin}{V_{ref}} \left(2^n - 1\right) = (127.5)_{10} = (127)_{10} = (01111111)_2$$

Lưu ý : Vo chỉ lấy phần nguyên

b. Độ phân giải ADC: (Sai số chuyển đổi)

Độ phân giải còn được gọi là bước thay đổi ở ngỏ vào, cho bởi công thức sau:

$$\Delta V_{in} = \frac{V_{ref}}{2^n - 1}$$

Nghĩa là ngỏ vào Vin phải thay đổi mỗi lần 1 lượng là bội số của ΔV_{in} thì ngỏ ra Vo mới thay đổi 1 lượng là 1.

Thí dụ 1:

Một ADC có n = 8 bit, Vref = 5v. thì có độ phân giải:

$$\Delta V_{in} = \frac{V_{ref}}{2^n - 1} = \frac{5}{255} = 0.0196 \text{ (volt/bit)} = 19.6 \text{ (mV/bit)}$$

Vin = 0.0196

Vo = (Vin/Vref)*(2^n-1) = 1

Vin = 2*0.0196

Vo = (Vin/Vref)*(2^n-1) =2

Nhưng trường hợp Vin không bằng bội số của 0.0196 thì ngỏ ra không đổi:

Vin = 0.0588 %<3*dpg

Vo = (Vin/Vref)*(2^8-1) = 2

c. Chương trình ứng dụng ADC:

+ Chương trình đo điện thế tại 2 kênh IN0 và IN1, từ 0v đến 5v, Vref = 5v.

```
#include<reg52.h>
#include<string.h>
#include<stdio.h>
unsigned char buf_lcd[20];
```

```
sbit RS   = P3^4; sbit E    = P3^5;  sbit EN   = P3^7;
sbit ale    = P1^5; sbit start  = P1^6; sbit Ao     = P1^7;
bit nFlags_ADC,nADC ;
float  data_ADC0,data_ADC1,Vref=5.0;
int n;
void delay(long cnt) //Hàm delay
{
     long i;
     for(i=0;i<cnt;i++);
}
void read_ADC(void) interrupt 0
{
   if (nADC)
     data_ADC1 = ((float)P2/255.0)*Vref;
   else
     data_ADC0 = ((float)P2/255.0)*Vref;
   nFlags_ADC = 1;
}
void convert_ADC()
{
   for(n=0;n<=1;n++)
    {
        nADC = n;
       Ao=n;//Chọn ngỏ vào IN0 hoặc IN1
       ale = 1; ale = 0; //Chốt địa chỉ  Ao, A1, A2
       start = 1; //Bắt đầu chuyển đổi ADC
       start = 0;
       while(!nFlags_ADC);//Chờ chuyển đổi xong
       delay(10);   } }
void main()
```

```
{
   IT0 = 1;EX0 = 1; EA  = 1;   EN=0; //Khai báo INT1
   start_LCD();
  giao_dien("TRUONG DHSPKT TPHCM","BO MON DKC",
              "KENH 0=0","KENH 1=0");
  while(1)
    {
        convert_ADC();
        if(nFlags_ADC)
        {
             nFlags_ADC = 0;
             sprintf(buf_lcd,"%f",data_ADC0);  write_str(buf_lcd,0X94+7);
             sprintf(buf_lcd,"%f",data_ADC1); write_str(buf_lcd,0XD4+7);
        } } }
```

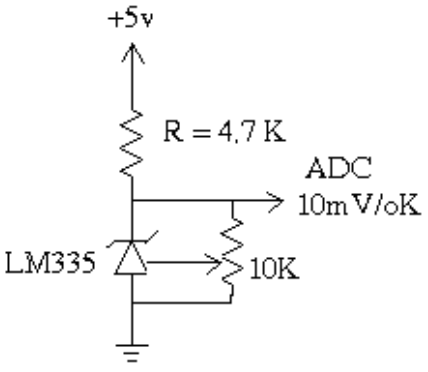
III. Ứng dụng:

1. Cảm biến nhiệt LM335:

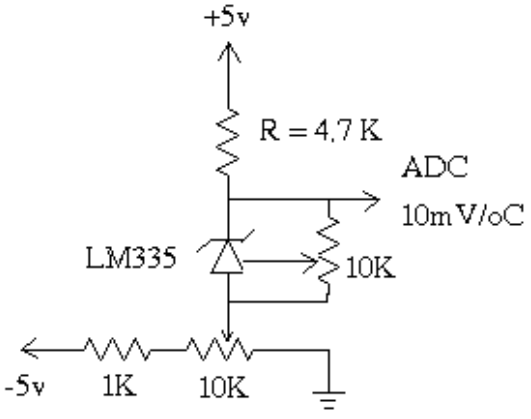
+ Dãy nhiệt độ làm việc từ -55 ^0C đến + 150 ^0C.

+ Ngỏ ra thay đổi 10mV/^0K, với K là độ Kelvin.

Ta có : $^0K = 273.15 + ^0C$



Hình 1 Ngỏ ra thay đổi 10mV/oK



Hình 2 Ngỏ ra thay đổi 10mV/oC

Chương trình cho hình 1:

+ Nếu gọi điện thế ra của cảm biến là Vo được tính bằng volt.

+ Đổi điện thế ra mV bằng cách nhân cho 1000*Vo(mV).

+ Nhiệt độ tính theo độ K = 1000*Vo(mV)/10.

⇨ Vậy nhiệt độ tính theo độ C là:

độ C = 1000*Vo(mV)/10 – 273.15

+ Chọn Vref = 5v.

+Chương trình:

```
#include<reg52.h>
#include<string.h>
#include<stdio.h>
unsigned char buf_lcd[20];
sbit RS   = P3^4;
sbit E    = P3^5;
sbit EN   = P3^7;
sbit ale    = P1^5;
sbit start  = P1^6;
sbit Ao     = P1^7;
bit nFlags_ADC;
float  dienthe,nhietdo,Vref=5.0;
//Hàm delay
void delay(long cnt)
{
     long i;
     for(i=0;i<cnt;i++);
}
void write_char(char Hcode,bit mode)
```

```
{
    RS=mode;
    P0=Hcode;
    E=1;
    delay(10);
    E=0;
}
void write_str(char str[],char row)
{
    int i;
    write_char(row,0);
    for(i=0;i<strlen(str);i++)
    {
        write_char(str[i],1);
    }
 }
void start_LCD()
{
     int i;
     char str[3]={0x38,0x0C,0x01};
     for(i=0;i<3;i++)
      {
          write_char(str[i],0);
      }
  }
void giao_dien(char str1[],char str2[],char str3[],char str4[])
```

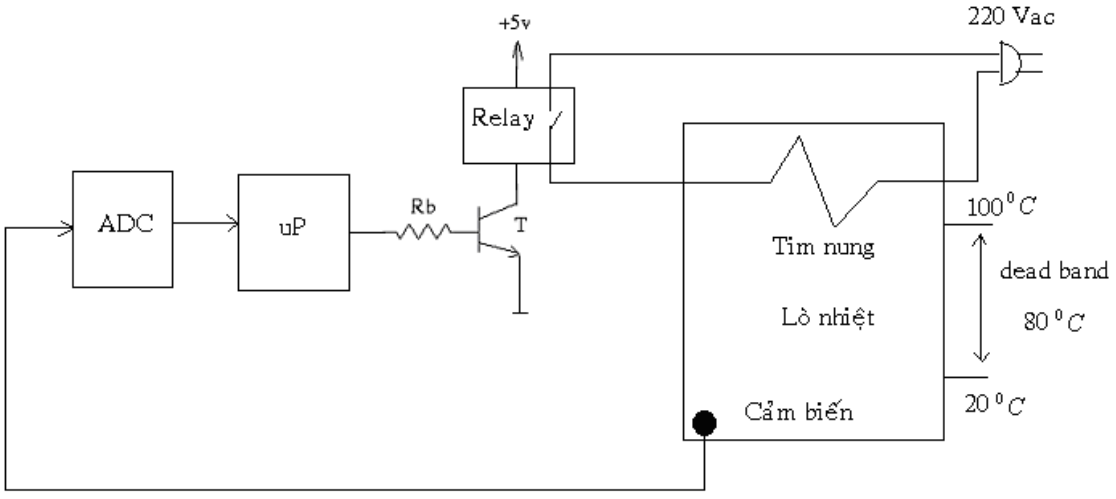
```
{
    write_str(str1,0X80); //Hàng 1
    write_str(str2,0XC0); //Hàng 2
    write_str(str3,0X94); //Hàng 3
    write_str(str4,0XD4); //Hàng 4
}
void read_ADC(void) interrupt 0
{
    dienthe = ((float)P2/255.0)*Vref;
    nhietdo= (dienthe*1000)/10 - 273.15;
    nFlags_ADC = 1;
}
void convert_ADC()
{
     Ao=1;
     ale = 1; ale = 0;
     start = 1;   start = 0;
     while(!nFlags_ADC);
     delay(10);
}
void main()
{
     //Khai báo INT1
    IT0 = 1;EX0 = 1; EA  = 1;   EN=0;
    start_LCD();
    giao_dien("TRUONG DHSPKT TPHCM","DO NHIET DO",
```

```
            "DIEN THE=0","NHIET DO(oC)=0");
  while(1)
    {
        convert_ADC();
        if(nFlags_ADC)
        {
            nFlags_ADC = 0;
            sprintf(buf_lcd,"%0.2f",dienthe);
            write_str(buf_lcd,0X94+9);
            sprintf(buf_lcd,"%0.2f",nhietdo);
            write_str(buf_lcd,0XD4+13);
        } } }
```

a. Thiết kế:

Thiết kế hệ thống điều khiển ON/OFF nhiệt độ lò từ 0^0C đến 100^0C, với sai số $\pm 0.4^0C$. Theo điều kiện sau:

- Tại 20^0C ngỏ ra ON.
- Tại 100^0C ngỏ ra OFF.



+Giải:

- Chọn cảm biến có ngỏ ra thay đổi $10mV/^0C$, vậy tại nhiệt độ lớn nhất thì ngỏ ra của cảm biến là:

$$V_{\max} = 10\frac{mV}{^0C}.100^0C = 1v = V_{ref}$$

- Độ phân giải của ADC:

$$\Delta V_{in} = \pm 0.4^0C\ .\ 10mV/^0C = 0.004\text{v}$$

$$\Delta V_{in} = \frac{V_{ref}}{2^n - 1} \Rightarrow 2^n = \frac{V_{ref}}{\Delta V_{in}} + 1$$

- Lấy log hai vế:

$$n\log(2) = \log\left(\frac{V_{ref}}{\Delta V_{in}} + 1\right) \Rightarrow n = \frac{\log\left(\frac{V_{ref}}{\Delta V_{in}} + 1\right)}{\log(2)} = 7.9715 \text{ bit}$$

Chọn ADC 0809 có Vref = 1v, n=8 bit, LM335 có mạch chuyển đổi như hình 2:

+ Nếu gọi điện thế ra của cảm biến là Vo được tính bằng volt.

+ Đổi điện thế ra mV bằng cách nhân cho 1000*Vo(mV).

+ Nhiệt độ tính theo độ C = 1000*Vo(mV)/10.

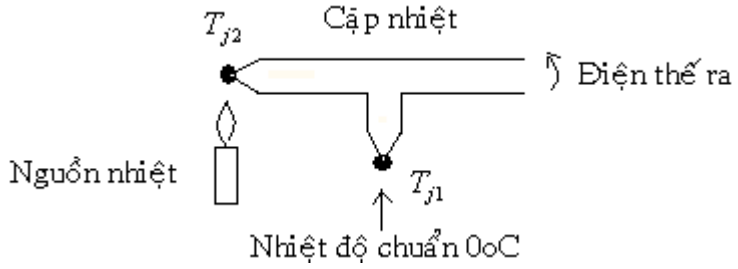
+Chương trình:

```
#include<reg52.h>
#include<string.h>
#include<stdio.h>
unsigned char buf_lcd[20];
sbit RS   = P3^4;
sbit E    = P3^5; sbit EN   = P3^7;
sbit ale    = P1^5; sbit start  = P1^6;
sbit Ao     = P1^7;
bit  nFlags_ADC ,Tim_nung ; //Làm tải giả
```

```
float  dienthe,nhietdo,Vref=5.0;
char *str[]={"OFF","ON"};
void read_ADC(void) interrupt 0
{
    dienthe = ((float)P2/255.0)*Vref;
    nhietdo= (dienthe*1000)/10;
    nFlags_ADC = 1;
}
void convert_ADC()
{
       Ao=1;
       ale = 1; ale = 0;
       start = 1;   start = 0;
       while(!nFlags_ADC);
       delay(10);
}
void main()
{
    IT0 = 1;EX0 = 1; EA  = 1;   EN=0;  //Khai báo INT1
    start_LCD();
    giao_dien("DO NHIET DO","POWER = OFF",
                  "DIEN THE=0","NHIET DO(oC)=0");
     while(1)
      {
           convert_ADC();
           if(nFlags_ADC)
```

```
        {
            nFlags_ADC = 0;
            sprintf(buf_lcd,"%0.2f",dienthe);
           write_str(buf_lcd,0X94+9);
            sprintf(buf_lcd,"%0.2f",nhietdo);
           write_str(buf_lcd,0XD4+13);
            sprintf(buf_lcd,"%3s",str[Tim_nung]);
            write_str(buf_lcd,0XC0+8);
        }
        if(nhietdo<=20)   //ON
              Tim_nung = 1;
        if(nhietdo>=100) //OFF
               Tim_nung = 0;
      }
}
```

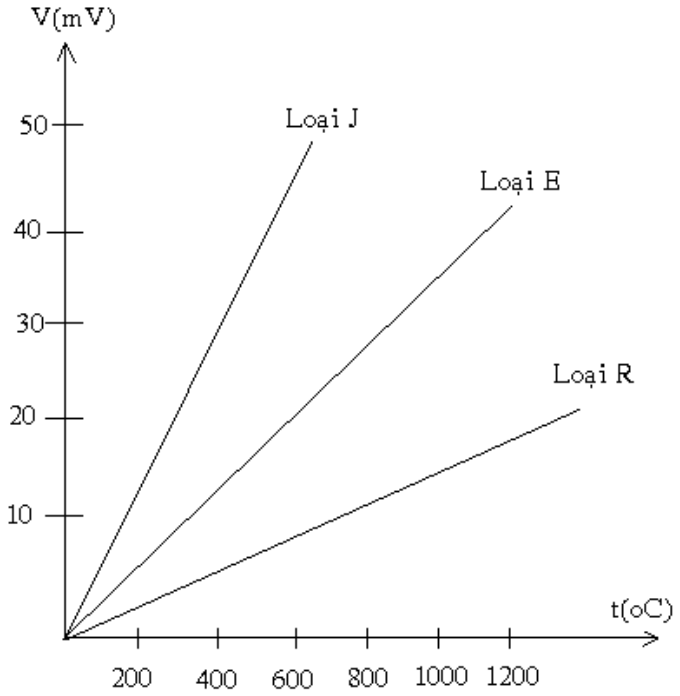
b. Cặp nhiệt:



- Hiệu ứng Seebeck: Điện thế sinh ra là do sự sai biệt về nhiệt độ giữa hai mối nối.

- Hiệu điện thế bằng 0 khi hai dây dẫn là cùng loại hoặc nhiệt độ tại hai mối nối là như nhau.

- Điện thế ra của cặp nhiệt được cho bởi hệ thức sau: $V = \alpha\left(T_{j2} - T_{j1}\right)$



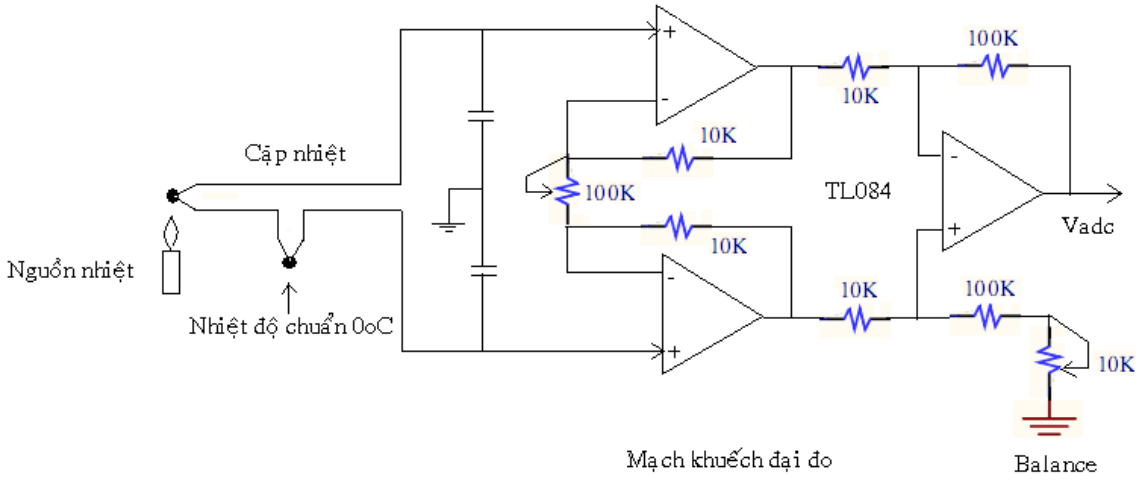
Với α >0 là hệ số nhiệt ($V/^oC$), T_{j2}, T_{j1} là nhiệt độ tại hai mối nối (^oC).

+ Thí dụ: Một cặp nhiệt loại J có $\alpha = 50\mu V/^oC$, nhiệt độ chuẩn $T_{j1} = 20\ ^oC$.

Tìm điện thế Seebeck tại $T_{j2} = 1000\ ^oC$.

+ Giải: $$V = \alpha(T_{j2} - T_{j1}) = 50\frac{\mu V}{^oC}(1000^oC - 20^oC) = 40mV$$

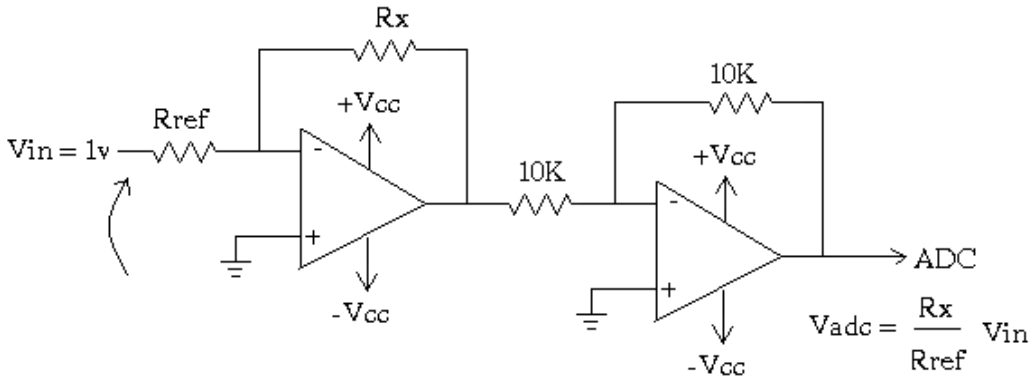
+ Khuếch đại đo: Thiết kế mạch đo có ngỏ ra toàn giai 4v



- Biến trở Balance dùng để chỉnh cân bằng cả hai kênh tại nhiệt độ chuẩn.

+ Bài tập: Viết chương trình cho hệ thống trên.

2. Đo điện trở:



Mạch đo dùng TL082

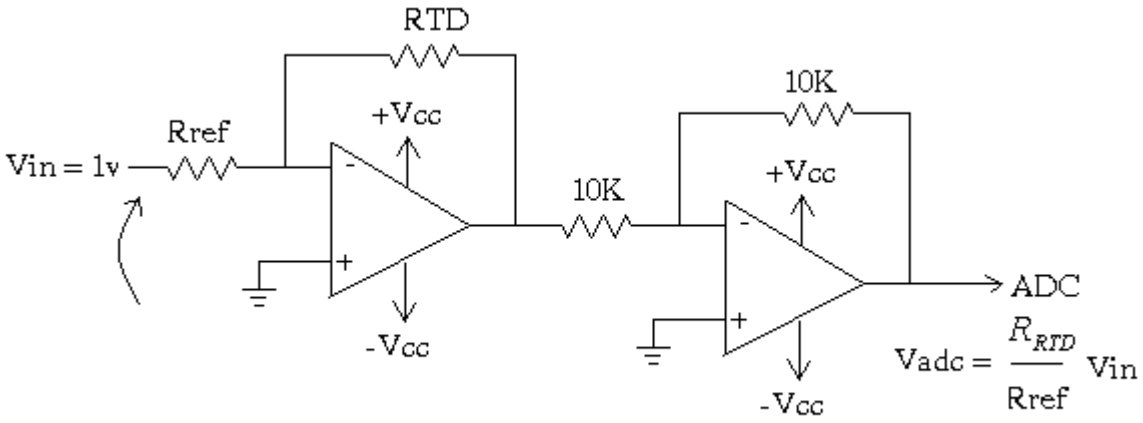
Vậy điện trở cần đo được tính theo công thức:

$$R_x = \frac{V_{adc}}{V_{in}} R_{ref} = V_{adc} R_{ref}$$

+ Ứng dụng:

- Đo độ ẩm gỗ, tre, kho hàng dùng cảm biến điện trở, do điện trở gỗ có giá trị $M\Omega$ do đó chọn $R_{ref} = 1M\Omega, V_{in} = 1v$

- Đo nhiệt độ bằng RTD:



Mạch đo nhiệt độ dùng RTD

$$R_{RTD} = \frac{V_{adc}}{V_{in}} R_{ref} = V_{adc} R_{ref} \quad (1)$$

- RTD còn được cho bởi công thức:

$$R_{RTD} = R_o\left[1+\alpha(T\ -T_o)\right] \ (2)$$

Với : $\alpha\ >0$ là hệ số nhiệt phụ thuộc vào vật liệu cấu tạo nên RTD

R_o là điện trở ban đầu tại nhiệt độ chuẩn T_o

$T \geq T_o$ là nhiệt độ cần đo

Đồng nhất phương trình (1) và (2) ta có nhiệt độ cần đo được cho bởi phương trình:

$$T = \frac{V_{adc}R_{ref}}{\alpha R_o} + T_o - \frac{1}{\alpha} \ (3)$$

Thí dụ: Một RTD tại $T_o = 20^oC$ có $R_o = 135\Omega$, hệ số nhiệt $\alpha = 0.00392/^oC$. Tìm điện trở của nó tại T = 100 ^oC.

+Giải:

Từ phương trình (2), ta có:

$$R_{100^oC} = 135\Omega\left[1+0.00392\,(100-20)\right] = 177.3360\ \Omega$$

+ Viết chương trình : $R_{ref}\ =100\Omega, V_{in}\ =1v, V_{ref}\ =5v$

```
#include<reg52.h>
#include<string.h>
#include<stdio.h>
unsigned char buf_lcd[20];
sbit RS   = P3^4; sbit E    = P3^5;
sbit ale    = P1^5; sbit start  = P1^6;sbit Ao     = P1^7;
bit nFlags_ADC ; int n;
float  Vadc,Vin=1.0,Vref=5.0,Rref=100.0,
          Ro=135.0,To=20.0,alpha=0.00392,T=0.0;
```

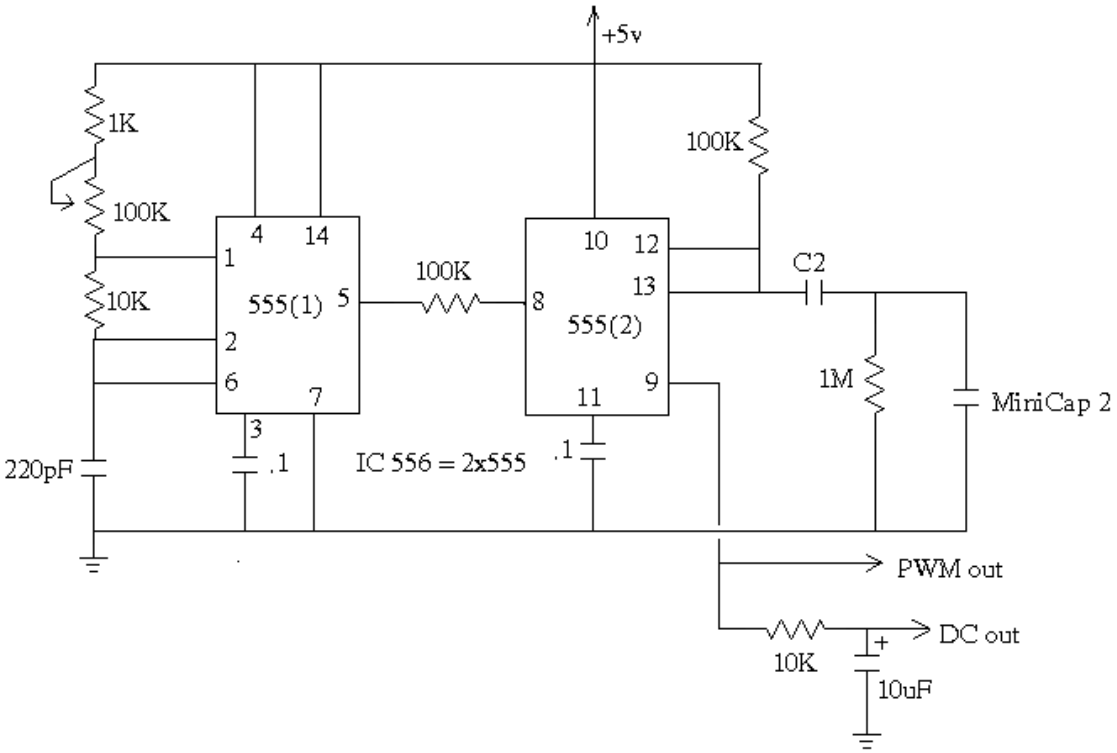
```
void delay(long cnt)
{
     long i;
     for(i=0;i<cnt;i++);
}
void write_char(char Hcode,bit mode)
{
    RS=mode;   P0=Hcode;
    E=1;   delay(10);  E=0;
  }
void write_str(char str[],char row)
{
    int i;
    write_char(row,0);
    for(i=0;i<strlen(str);i++)
    {
       write_char(str[i],1);
    }
  }
void start_LCD()
{
     int i;
char str[3]={0x38,0x0C,0x01};
     for(i=0;i<3;i++)
      {
          write_char(str[i],0);   } }
```

```
void giao_dien(char str1[],char str2[],char str3[],char str4[])
{
    write_str(str1,0X80); //Hàng 1
    write_str(str2,0XC0); //Hàng 2
    write_str(str3,0X94); //Hàng 3
    write_str(str4,0XD4); //Hàng 4
}
void read_ADC(void) interrupt 0
{
    Vadc = ((float)P2/255.0)*Vref;
    T = ((Vadc*Rref)/(Ro*alpha)) + To - (1.0/alpha);
    nFlags_ADC = 1;
}
void convert_ADC()
{
   Ao=0; ale = 1; ale = 0; start = 1;    start = 0;
   while(!nFlags_ADC);  delay(10);
}
void main()
{
    IT0 = 1;EX0 = 1; EA  = 1;    //Khai báo INT0
    start_LCD();
    giao_dien("TRUONG DHSPKT TPHCM","KHAO SAT RTD","Vadc=0","T=0");
    while(1)
     {
          convert_ADC();
```

```
        if(nFlags_ADC)
        {
            nFlags_ADC = 0;
            sprintf(buf_lcd,"%0.2f",Vadc);
            write_str(buf_lcd,0X94+5);
            sprintf(buf_lcd,"%3.3f",T);
            write_str(buf_lcd,0XD4+2);
     }
    }
}
```

3. Cảm biến điện dung:

Xét cảm biến điện dung MiniCap 2 được dùng để đo độ ẩm của một kho hàng. Do cảm biến có tổng trở âm là phi tuyến nên cần kết hợp với mạch tuyến tính hóa:



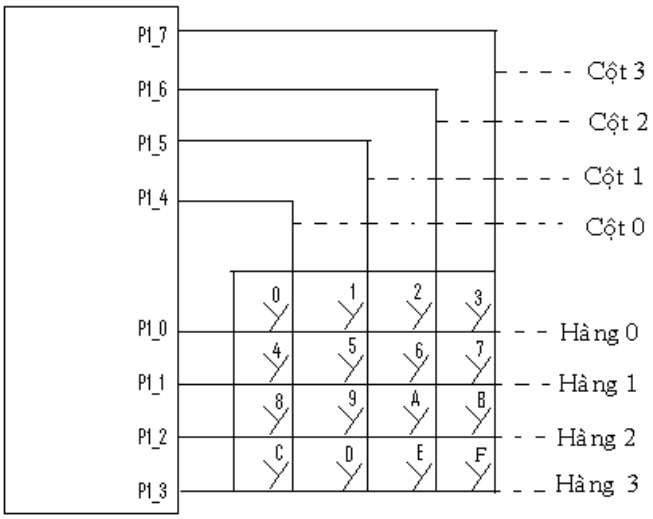
+ Tụ C2 được tính sao cho điện áp AC trên hai đầu cảm biến MiniCap 2 không được vượt quá 1 Vac.

+ Chỉnh biến trở 100K để ở độ ẩm bình thường điện thế ra DC out = 1 Vdc.

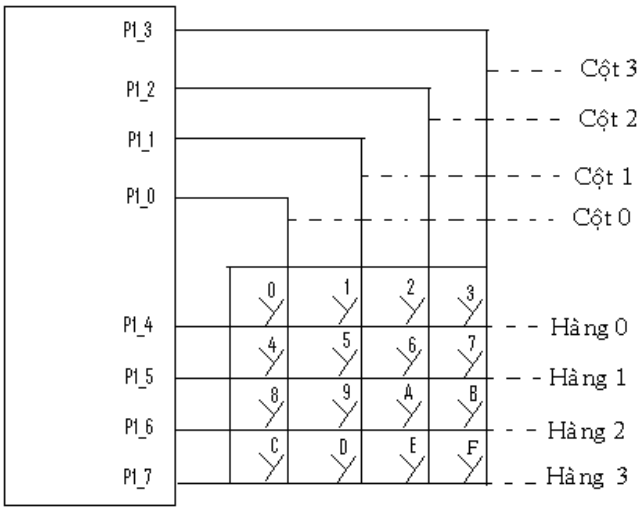
+ Khi độ ẩm tăng làm thay đổi độ rộng xung PWM out dẫn đến làm thay đổi điện thế ra DC out.

IV- Bài tập: Viết chương trình nhận phím nhấn không dùng ic giải mÃ 74922.

- Ma trận bàn phím: Cột và hàng được định nghĩa chỉ mang tính tương đối, nghĩa là trong hình 1 ta có thể qui định như sau:



Hình 1



Hình 2

Trong hình 1:

+ 4 cột từ P1_4 đến P1_7 .

+ 4 hàng từ P1_0 đến P1_3.

Trong hình 2:

+ 4 cột từ P1_0 đến P1_3

+ 4 hàng từ P1_4 đến P1_7

Vì ma trận vuông 4x4 có tính đối xứng do đó ma trận bàn phím giữa hình 1 và 2 là như nhau và được cho bởi phương trình sau:

Giá trị phím nhấn = Trọng số của cột + 4* Trọng số của hàng

Với:

+Trọng số cột = số thứ tự cột.

+Trọng số hàng = số thứ tự hàng.

+ Cách tính trọng số hàng và cột:

Lấy bù 15 của giá trị hàng (hay cột) sau đó chia liên tiếp cho 2.

```
int Weigth(int giatri)
{
    int trongso = -1;
    while (Data>=1)
     {
        Data = Data/2;
        trongso++;
     }
    return (trongso);
}
```

+ Giải thuật: Chọn 4 bit thấp là hàng và 4 bit cao là cột:

- Bước 1: Lần lượt đưa từng cột xuống 0.

- Bước 2: Lần lượt dò xem hàng nào đã được nối đất.

- Bước 3:

.Nếu phát hiện được phím nào được nhấn thì chống dội bằng cách dò 50 lần.

.Nếu sau 50 lần dò mà vẫn còn phát hiện phím nhấn thì nhận.

.Tính số thứ tự của hàng và cột của phím được nhấn.

- Bước 4:

.Chờ cho phím nhấn thả ra.

. Chờ đủ 50 lần thì tính giá trị phím nhấn.

. Giá trị phím nhận được tính bởi phương trình:

Giá trị phím nhấn = Trọng số của cột + 4* Trọng số của hàng

+ Chú ý 1: Nếu có 2 hay nhiều phím được nhấn cùng 1 lúc thì sẽ nhận phím có giá trị cao nhất

- Giả sử có hai phím 0 và 4 cùng được nhấn:

Giá trị hàng = $15 - (P1\&0x0F)$ = 15 - 12 = 3

Số thứ tự của hàng = 1

Giá trị cột = $15 - ((P1\&0xF0)>>4)$ = 15 - 14 = 1

Số thứ tự của cột = 0

- Thay vào công thức, ta có:

Giá trị phím nhấn = $0 + 4* 1 = 4$

+ Chú ý 2:

Do ma trận là vuông nên ta có thể thay đổi được thứ tự giá trị của phím nhấn:

Giả sử giá trị của phím nhấn được cho bởi hình 1, ta có phương trình toán học mô tả bàn phìm như sau:

Giá trị của phím nhấn = 4* Trọng số của cột + Trọng số của hàng

+ Chương trình quét bàn phím:

```
//Hàm xét các phím chức năng
void KeyFcn(int Data)
  {
      if(Data<10)//Chỉ nhận các phím từ 0 đến 9
       {
          if(sanpham>=1000)   sanpham = 0;
          sanpham =  10*sanpham +Data;    nFlag = 1;
      }
      if(Data==10) //Xóa
        {
            sanpham=0;  nFlag = 1;
       }
  }
int Weigth(int Giatri)  //Tính trọng số của cột và hàng
{
    int trongso = -1;
    while (Giatri>=1)
     {
        Giatri = Giatri/2;
        trongso++;
     }
    return (trongso);
}
void KeyBoard()//Chương trình cho 4 bit thấp là hàng và 4 bit cao là cột
{
      if(((P2&0x0F))!=15) // Nếu có phím được nhấn
      {
          solannha = 0; FlagShift = 1; // Không cho xoay
```

```
      if(!FlagRecv) //Xét đủ 50 lần
       {
          if(++solannhan==50)
          {
             DataCot = 15 - ((P2&0xF0)>>4); //Shift phải 4 bit lấy cột
            //DataCot = 15 - (_iror_((P2&0xF0),4));
             trongsoCol =Weigth(DataCot); //Tìm số thứ tự cột
              DataHang = 15 - (P2&0x0F); //Lấy bù 15 của hàng
              trongsoRow =Weigth(DataHang); //Tìm số thứ tự hàng
              //Thay vào công thức tính gía trị của phím nhấn
              Data = trongsoCol*4 + trongsoRow;
              FlagRecv = 1;
            }  }
    }
      else //Đủ 50 lần nhả phím
       {
          solannhan = 0;
          if(FlagRecv)
            {
                if(++solannha == 50)
                 {
                    KeyFcn(Data);
                     FlagShift = 0; FlagRecv = 0 ;//Nhận tiếp lần sau
                 } }
       }
if(!FlagShift)//Nếu chưa phát hiện phím nhấn thì đưa cột kế tiếp xuống 0
  {
     P2 = (shiftCol |0x0F); //Lấy 4 bit cao là cột
     shiftCol = _crol_(shiftCol,1);//Xoay phải  _cror_(shiftCol,1);
 } }
```

Chương 7 :

ENCODER VÀ ỨNG DỤNG

I. Encoder:

Thường là loại có ngỏ ra cực thu để hở, do đó khi sử dụng cần phải có điện trở nối lên cao.

1. Hình dạng:

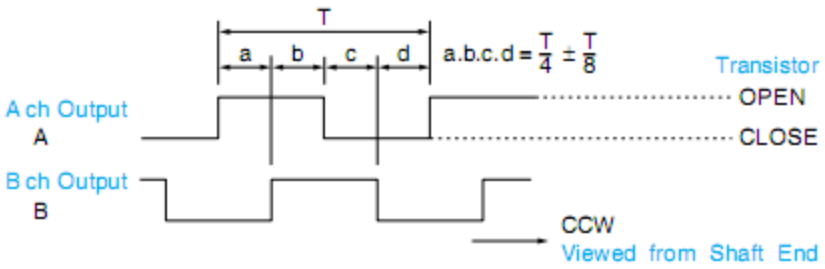
Encoder

2. Bảng màu dây:

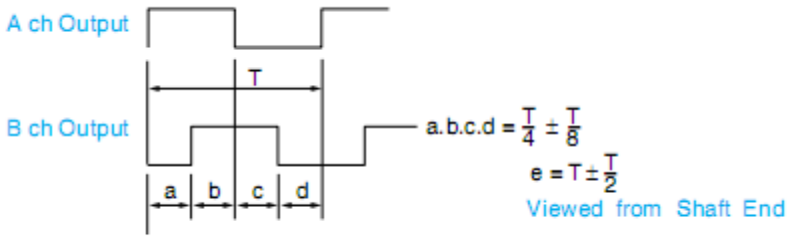
Lead color	Function
BLUE	A ch Output
GREEN	B ch Output
RED	DC+5V
BLACK	GND

Bảng màu dây

3. Ngỏ ra xung:



Xung A và B khi quay thuận



Xung A và B khi quay nghịch

Qua đồ thị ta nhận thấy xung A cố định còn xung B thay đổi theo chiều quay của Encoder.

4. Công dụng:

+ Xác định số vòng quay của đối tượng.

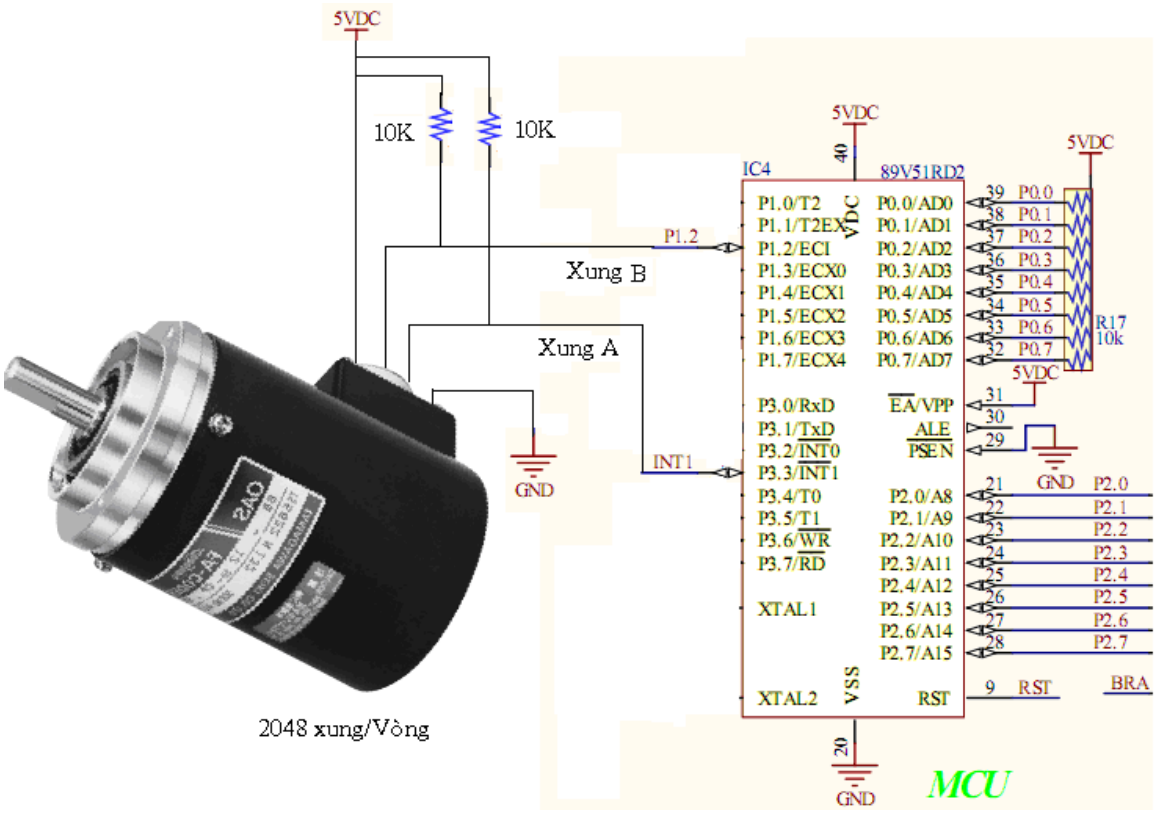
+ Đo góc hay mặt phẳng.

+ Xác định tọa độ x, y của tay máy hay robot.

+ Đo tốc độ động cơ hay máy phát. . .

II. Ứng dụng Encoder:

+ Sơ đồ phần cứng:



1. Xác định số vòng quay và chiều:

```
#include<reg52.h>
#include<string.h>
#include<stdio.h>
unsigned char buf_lcd[20];
sbit RS          = P3^4;  sbit E    = P3^5;  sbit EN   = P3^7;
sbit ale         = P1^5;  sbit start  = P1^6;
sbit Ao          = P1^7;
sbit pulse_B     = P1^2;
bit nFlags, dir;
float  pulse, so_vong;
char *str[] = {"FOR","REV"};
void delay(long cnt) //Hàm delay
{
     long i;
     for(i=0;i<cnt;i++);
}
void write_char(char Hcode,bit mode)
{
  RS=mode;
  P0=Hcode;
  E=1; delay(10);  E=0;
}
void write_str(char str[],char row)
{
  int i;
  write_char(row,0);
  for(i=0;i<strlen(str);i++)
  {
     write_char(str[i],1);   } }
```

```
void start_LCD()
{
   int i;
   char str[3]={0x38,0x0C,0x01};
   for(i=0;i<3;i++)
    {
         write_char(str[i],0);
    }  }
void giao_dien(char str1[],char str2[],char str3[],char str4[])
{
  write_str(str1,0X80); //Hàng 1
  write_str(str2,0XC0); //Hàng 2
  write_str(str3,0X94); //Hàng 3
  write_str(str4,0XD4); //Hàng 4
}
void encoder(void) interrupt 2
{
      pulse++;
      dir = INT1|pulse_B;  nFlags = 1;
 }
void main()
{
   IT1 = 1;EX1 = 1; EA  = 1;   EN=0; //Khai báo INT1 cho encoder
   start_LCD();
   giao_dien("TRUONG DHSPKT TPHCM","ENCODER","SO VONG = ","CHIEU = ");
   while(1)
    {
       if(nFlags)
           {
                 nFlags = 0;
```

```
            so_vong = pulse/2048.0;
            sprintf(buf_lcd,"%0.2f",so_vong);
            write_str(buf_lcd,0X94+10);
            sprintf(buf_lcd,"%s",str[dir]);
          write_str(buf_lcd,0XD4+10);
        } } }
```

2. Xác định góc hay đo mặt phẳng của một bề mặt:

```
#include<reg52.h>
#include<string.h>
#include<stdio.h>
unsigned char buf_lcd[20];
sbit RS   = P3^4; sbit E    = P3^5; sbit EN   = P3^7;
sbit ale    = P1^5; sbit start  = P1^6;
sbit Ao     = P1^7;
sbit pulse_B     = P1^2;
bit nFlags, dir;
float  pulse, so_vong, doo;
char *str[] = {"+","-"};
void delay(long cnt) //Hàm delay
{
    long i;
    for(i=0;i<cnt;i++);
}
//Chương trình LCD đã viết ở chương trình trên
void encoder(void) interrupt 2
{
  pulse++;
  dir = INT1|pulse_B;
  nFlags = 1;
}
```

```
void main()
{
  IT1 = 1;EX1 = 1; EA  = 1;   EN=0; //Khai báo INT1
  start_LCD();
  giao_dien("TRUONG DHSPKT TPHCM","ENCODER","GOC(DO)=","SO VONG=");
 while(1)
   {
       if(nFlags)
         {
               nFlags = 0;
               so_vong = pulse/2048.0;
               doo = so_vong*360.0; //độ
               sprintf(buf_lcd,"%s%0.2f",str[dir],doo);
               write_str(buf_lcd,0X94+10);
               sprintf(buf_lcd,"%0.2f",so_vong);
         }    }
}
```

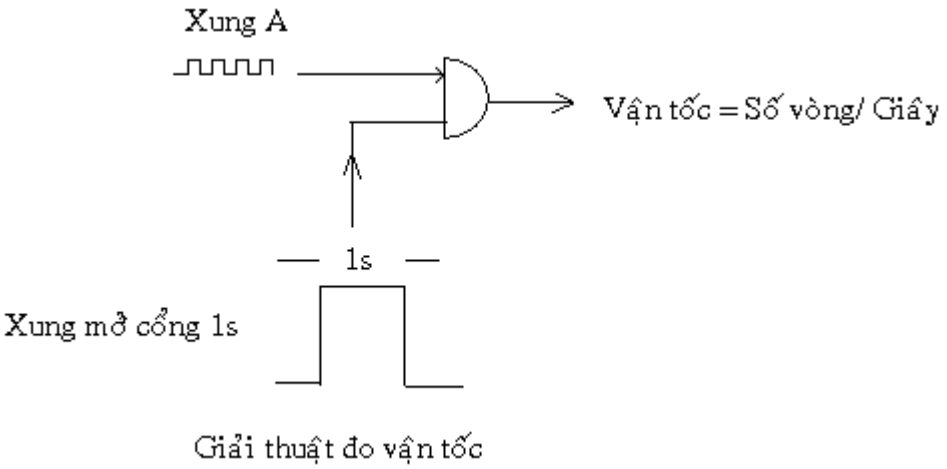
3. Xác định tọa độ x, y:

```
#include<reg52.h>
#include<string.h>
#include<stdio.h>
#include<math.h>
unsigned char buf_lcd[20];
sbit RS   = P3^4;  sbit E    = P3^5;  sbit EN   = P3^7;
sbit ale    = P1^5;  sbit start  = P1^6;
sbit Ao     = P1^7;
bit nFlags;
float  pulse,so_vong,doo,x,y;
float r = 5.0; //Bán kính vòng tròn hay chiều dài tay máy
#define pi 3.141593
```

```
void encoder(void) interrupt 2
{
     pulse++;
 }
void main()
{
  IT1 = 1;EX1 = 1; EA  = 1;   EN=0;  //Khai báo INT1
  start_LCD();
  giao_dien("SO VONG = ","RAD = ", "X = ","Y = ");
   while(1)
    {
      if(nFlags)
        {
           nFlags = 0;
           so_vong = pulse/2048.0;
           doo = so_vong*2.0*pi;
           x = r*cos(doo);//Phương trình đường tròn
           y = r*sin(doo);
          sprintf(buf_lcd,"%0.2f",so_vong);
          write_str(buf_lcd,0X80+10);
          sprintf(buf_lcd,"%0.2f",doo);
          write_str(buf_lcd,0XC0+10);
         sprintf(buf_lcd,"%0.2f",x);
          write_str(buf_lcd,0X94+7);
         sprintf(buf_lcd,"%0.2f",y);
          write_str(buf_lcd,0XD4+7);
       }
    }
 }
```

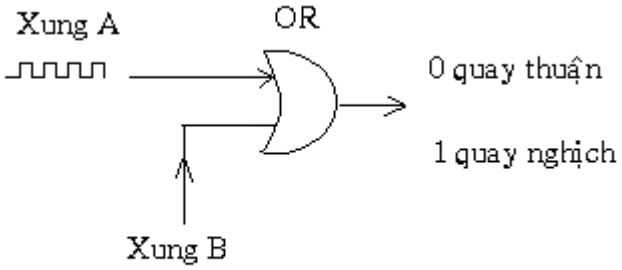
4. Đo vận tốc động cơ:

+ Dùng xung mở cổng 1 giây để đo vận tốc:



Giải thuật đo vận tốc

+ Dùng xung B để xác định chiều quay:



Giải thuật xác định chiều quay

```
#include<reg52.h>
#include<string.h>
#include<stdio.h>
unsigned char buf_lcd[20];
sbit RS   = P3^4;
sbit E    = P3^5;
sbit EN   = P3^7;
sbit ale    = P1^5;
sbit start  = P1^6;
sbit Ao     = P1^7;
sbit pulse_B     = P1^2;
float  pulse,van_toc;
```

```
bit TIM0=1, dir;
int msec, cycle;
char *str[] = {"FOR","REV"};
void encoder(void) interrupt 2
{
  if(TIM0)
  {
     pulse++;
     dir = INT1|pulse_B;
  }
}
// interrupt trong Timer 0
void timer_sec(void)  interrupt 1
{
   TF0=0;
   if(++cycle==4)//1000us
   {
     cycle=0;
     if(++msec==1000)//1000000us = 1sec
        {
            msec = 0;
            van_toc = pulse/2048.0;
            TIM0 = ~TIM0;
            pulse = 0;
         }
    }
}
```

```
void main()
{
  //Khai bao interrupt timer 0 trong
  TMOD |= 0x02;  TH0   = 0x06; TL0 = TH0;TR0 = 1; ET0 = 1;
  IT1 = 1;EX1 = 1; EA  = 1;   EN=0;  //Khai báo INT1
  start_LCD();
  giao_dien("TRUONG DHSPKT TPHCM","BO MON DKC","SPEED = ","CHIEU = ");
 while(1)
   {
          if(!TIM0)
           {
              TIM0 = 1;
              sprintf(buf_lcd,"%0.2f",van_toc);
              write_str(buf_lcd,0X94+10);
              sprintf(buf_lcd,"%s",str[dir]);
             write_str(buf_lcd,0XD4+10);
          }     }    }
```

III. Nhận dạng đối tượng:

Nhận dạng nhằm mục đích tìm ra công thức của một đối tượng bất kỳ nhưng đã bị mất tham số.

Một cảm biến trọng lượng (loadcell) tuyến tính có giá trị toàn giai 10Kg, biết điện thế ra từ 0mV -> 5mV tương ứng khi đo từ 0 Kg -> 10 Kg.

+ Nhận dạng:

- Khi đối tượng tuyến tính ta chỉ cần biết hai cặp giá trị min và max rồi từ đó viết phương trình đường thẳng qua hai điểm A và B là đủ:

min = [0 mV, 0 Kg] ; max = [10 mV, 10 Kg] ;

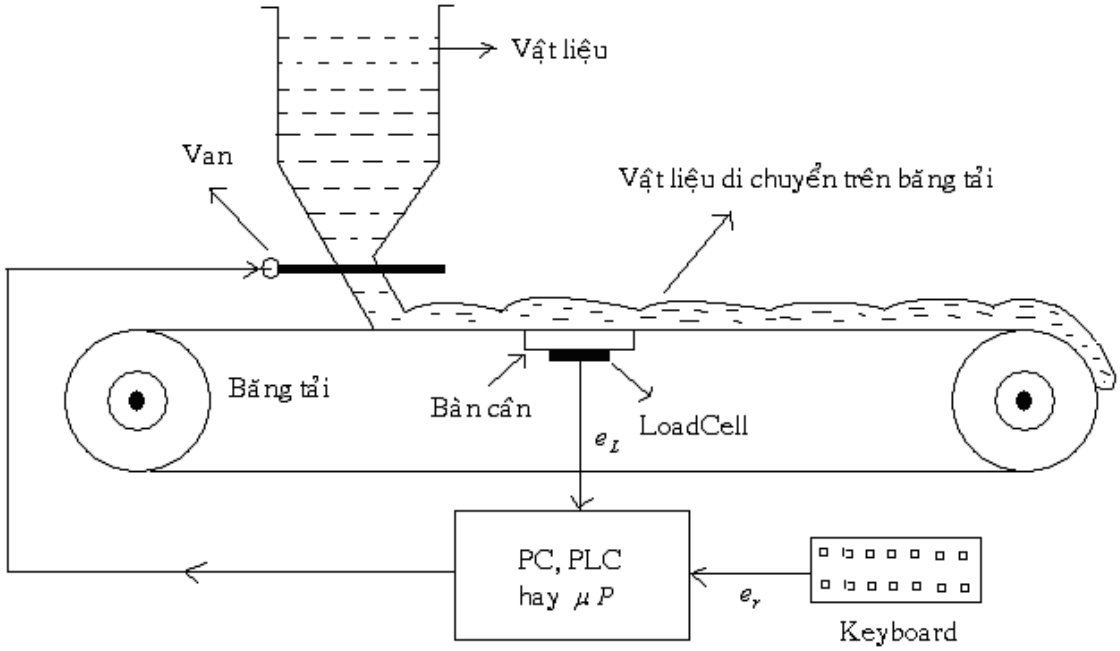
V = [0 mV, 5 mV] ; M = [0 Kg, 10 Kg] ;

- Phương trình đường thẳng có dạng:

Trọng lượng(M) = 2000*điện thế (V)

III. Bài tập:

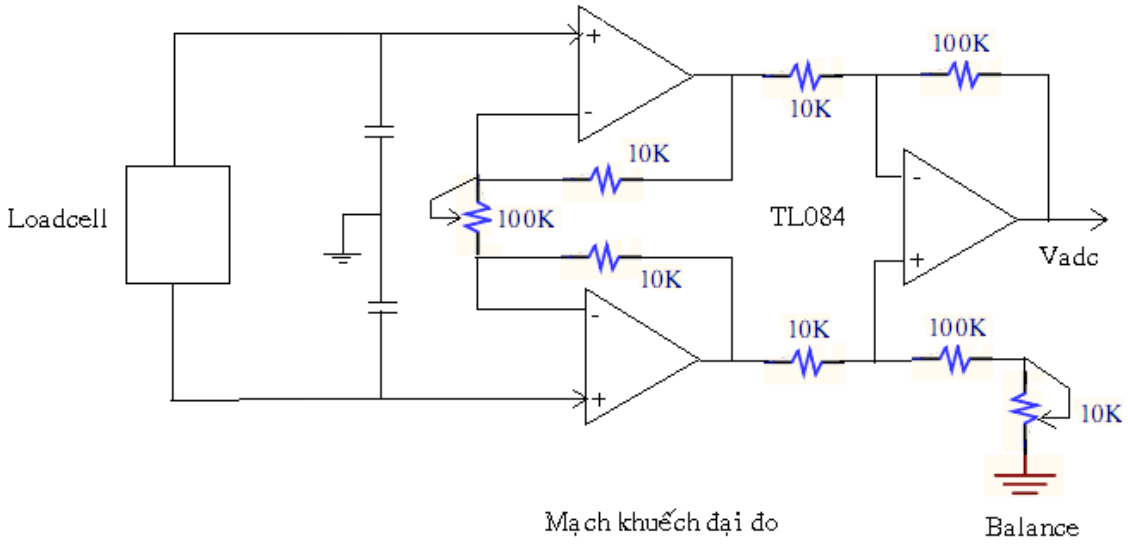
Bài 1:Viết chương trình đo trọng lượng trên băng tải sử dụng Loadcell:



+ Hệ thống làm nhiệm vụ lấy mẫu trung bình trọng lượng của vật liệu thông qua tín hiệu e_L bằng cách lấy mẫu 10 lần trong 0.5 giây.

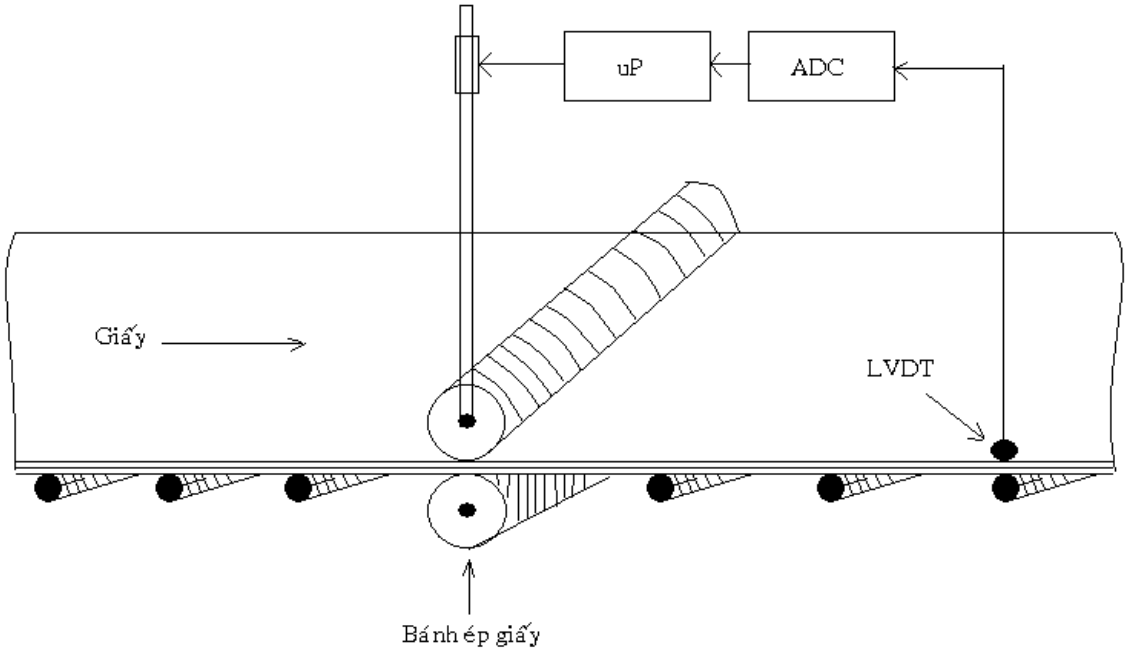
+ Sau đó giá trị trung bình này được so sánh với giá trị đặt e_r, nếu $e_L >= e_r$ thì van sẽ nhận một chỉ thị để đóng lại.

+ Mạch khuếch đại đo:



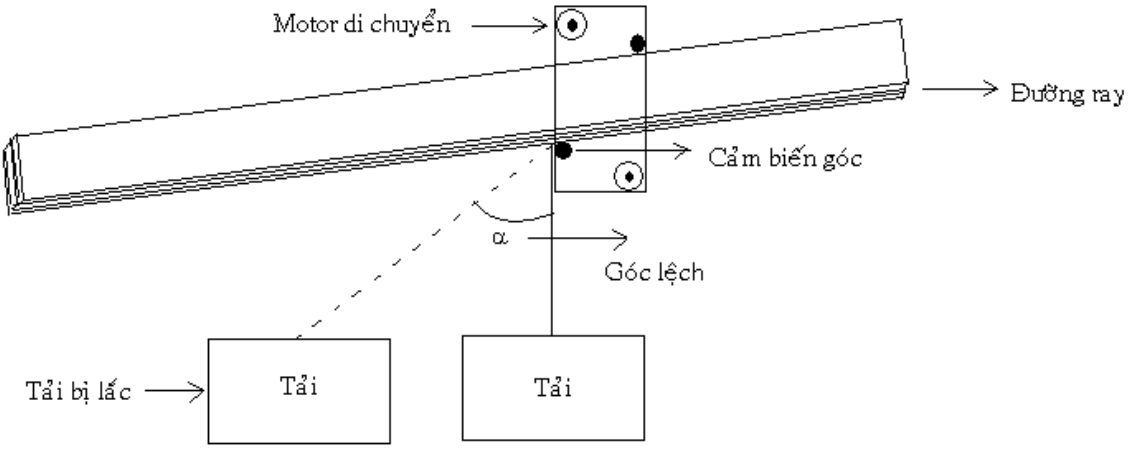
Mạch khuếch đại đo

Bài 2: Viết chương trình cho máy ép giấy sử dụng LVDT:

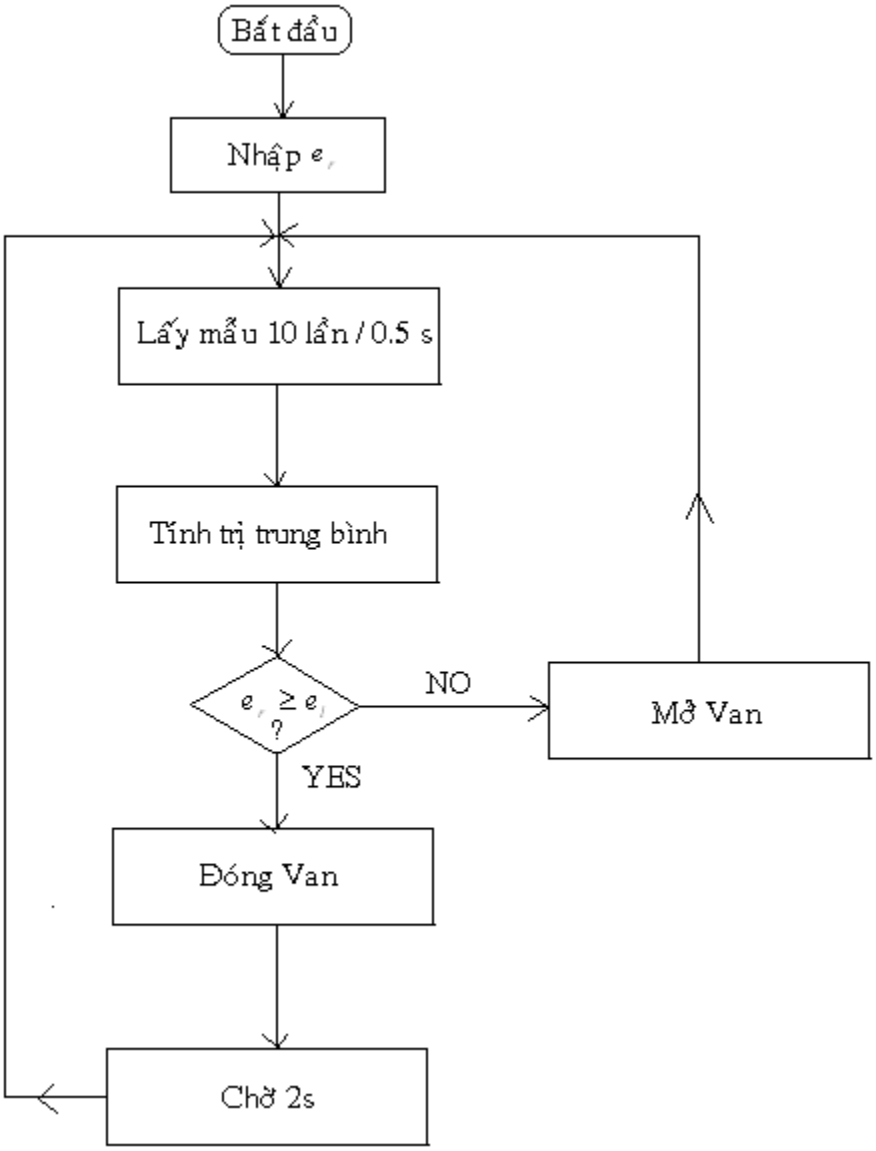


+ Độ dày của giấy được đo bởi LVDT.

Bài 3: Viết chương trình điều khiển cần trục với tốc độ di chuyển của cần trục phụ thuộc vào góc lệch α:



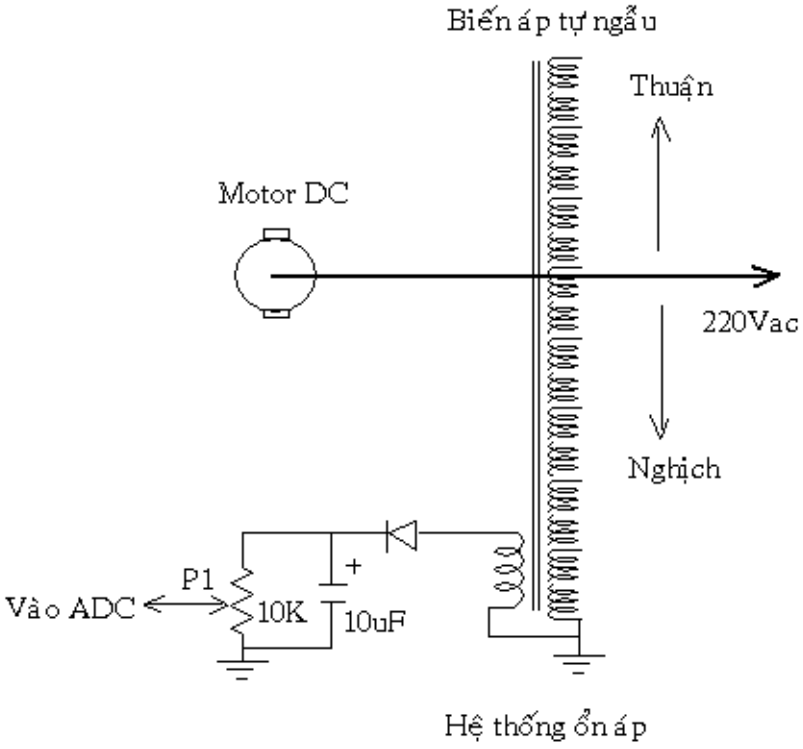
+ Lưu đồ giải thuật cho bài 1:



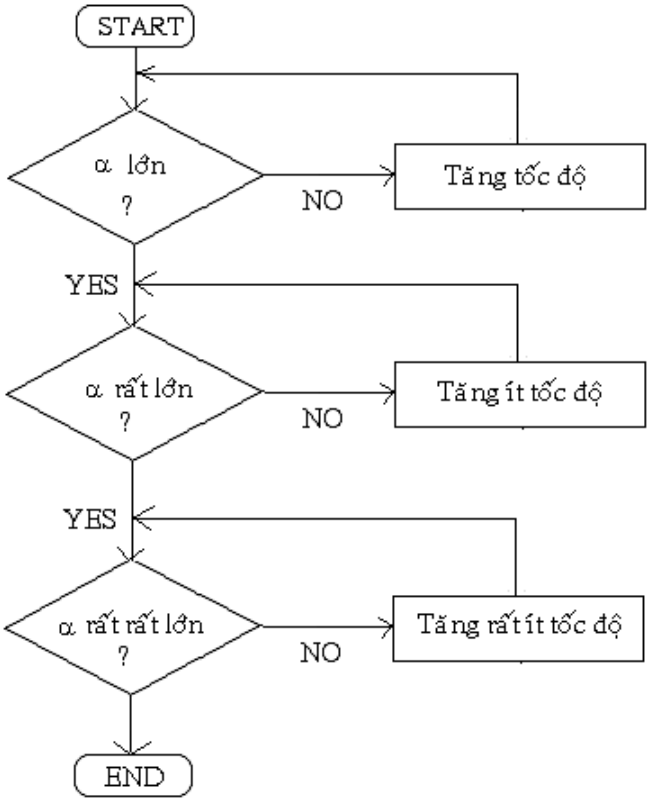
Bài 4: Viết chương trình điều khiển biến áp tự ngẫu sao cho luôn giữ cố định điện áp ra là 220Vac theo điều kiện sau:

+ Nếu ngỏ ra tăng 230Vac thì điện áp ra tại biến áp phụ P1 tăng theo dẫn đến làm motor quay nghịch kéo giảm điện áp về lại 220Vac.

+ Nếu ngỏ ra giảm còn 210Vac thì điện áp ra tại biến áp phụ P1 giảm theo dẫn đến làm motor quay thuận kéo tăng điện áp ra lên lại 220Vac.



+ Lưu đồ giải thuật cho bài 3:

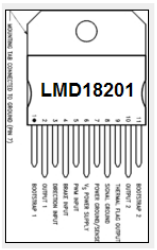


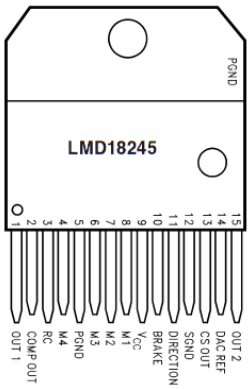
Chương 8 :

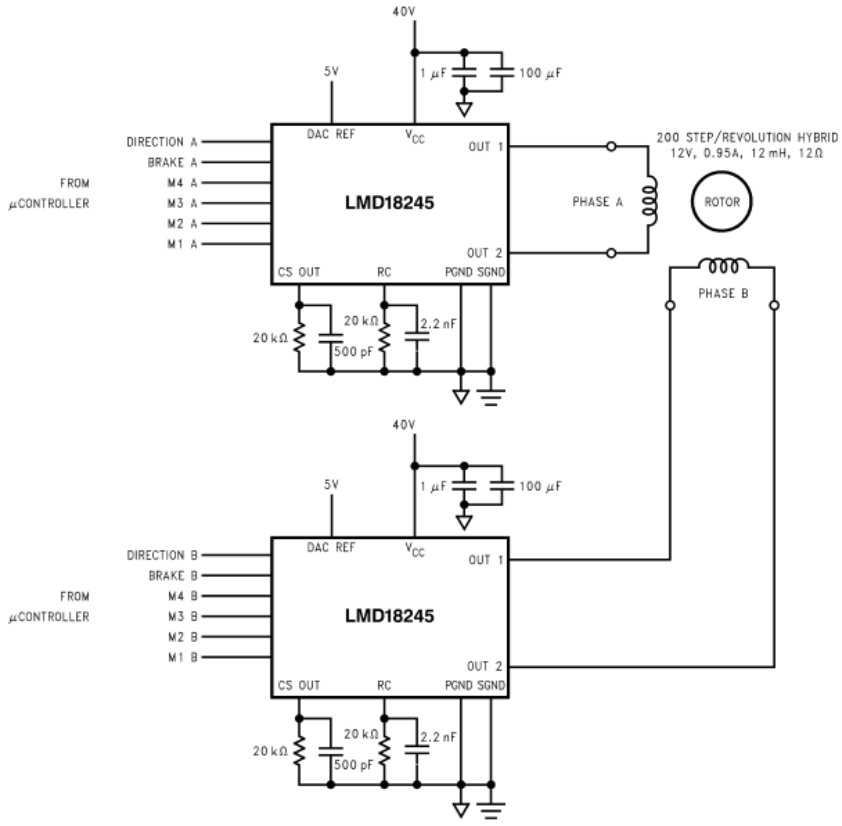
ĐIỀU KHIỂN MOTOR

I. Motor Driver:

1.Các linh kiện cầu H :



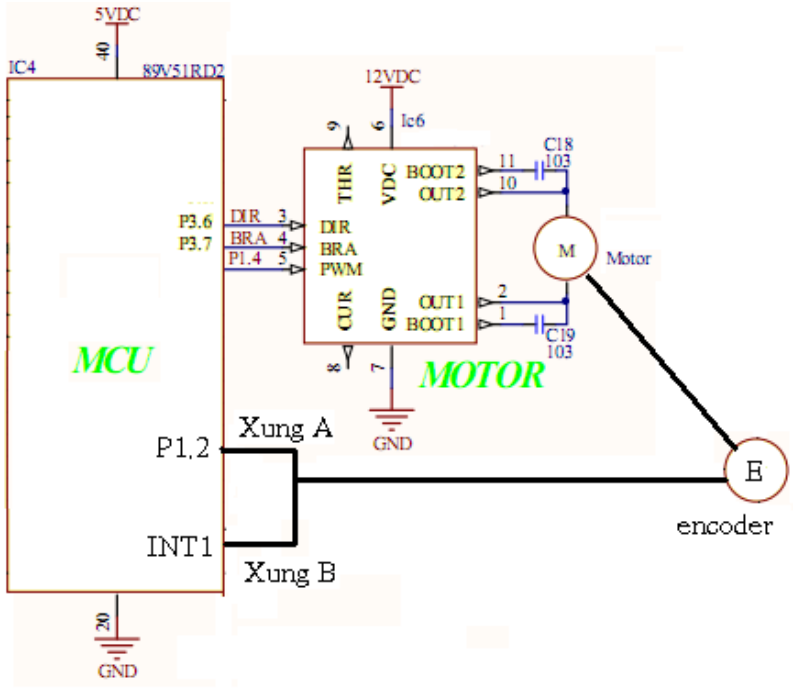




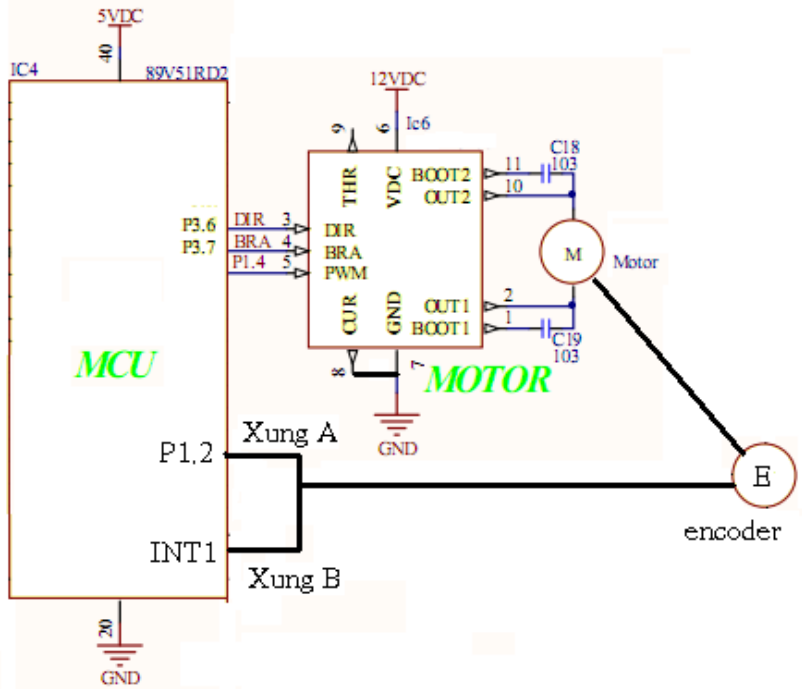
Điều khiển step mortor

+ LDM18201/18200, 3 Amp, điện thế làm việc 55 volt, loại cầu H.

+ LDM18245, 3 Amp, điện thế làm việc 55 volt, loại điều khiển digital có 4 bit DAC.



Sơ đồ cho LDM18200



Sơ đồ cho LDM18201

2.Điều rộng xung PWM :

Trong 89V51RD2 có 5 ngỏ điều khiển PWM thú tự từ P1.3 đến P1.7, với phương pháp khởi tạo như sau:

```
CMOD   = 0x00;
CCAPM**X** = 0x42;
CR = 1;
CCAP**X**H = 0xFF; //Từ 255 -> đến 0
```

Với **X** từ 0 đến 4 tương ứng từ P1.3 đến P1.7

Thí dụ:

+ Khởi tạo chân P1.3 có tốc độ ban đầu = 0

```
CMOD   = 0x00;
CCAPM**0** = 0x42;//Chân P1.3
CR = 1;
CCAP**0**H = 0xFF; //Từ 255 -> đến 0
```

+ Khởi tạo chân P1.6 có tốc độ ban đầu là tối đa

```
CMOD   = 0x00;
CCAPM**3** = 0x42; //Chân P1.6
CR = 1;
CCAP**3**H = 0; //Từ 255 -> đến 0
```

3.Chương trình minh họa :

```
#include<REG51M.h> //Cho PWM
#include<string.h>
#include<stdio.h>
unsigned char buf_lcd[20];
sbit RS   = P3^4;
sbit E    = P3^5;
sbit bra  = P3^7;
sbit dir   = P3^6;
sbit ale    = P1^5;
sbit start  = P1^6;
```

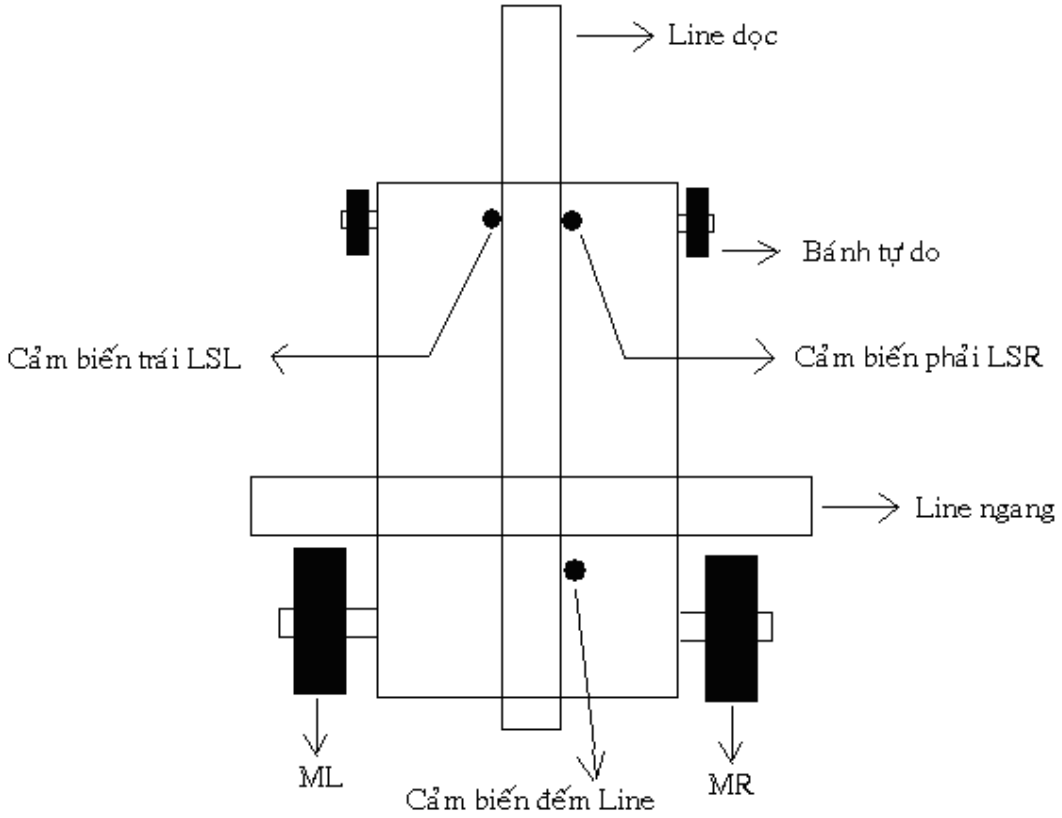
```
sbit Ao     = P1^7;
bit nFlags_ADC;
sbit chieu     = P1^0;
sbit thang     = P1^1;
unsigned int  speed;
//Hàm delay
void delay(long cnt)
{
      long i;
      for(i=0;i<cnt;i++);
}
void write_char(char Hcode,bit mode)
{
    RS=mode;
    P0=Hcode; // Data ch?a trong P0
    E=1;
    delay(10);
    E=0;
}
void write_str(char str[],char row)
{
    int i;
    write_char(row,0);
    for(i=0;i<strlen(str);i++)
    {
       write_char(str[i],1);
    }
 }
```

```
void start_LCD()
{
     int i;
     char str[3]={0x38,0x0C,0x01};
     for(i=0;i<3;i++)
      {
          write_char(str[i],0);
      }
}
void giao_dien(char str1[],char str2[],char str3[],char str4[])
{
    write_str(str1,0X80); //Hàng 1
    write_str(str2,0XC0); //Hàng 2
    write_str(str3,0X94); //Hàng 3
    write_str(str4,0XD4); //Hàng 4
}
void read_ADC(void) interrupt 0
{
    speed = P2;
    CCAP1H = (255-speed);
    nFlags_ADC = 1;
}
void convert_ADC()
{
   Ao=0;    ale = 1; ale = 0;
   start = 1;     start = 0;
   while(!nFlags_ADC);
   delay(10);
}
void main()
```

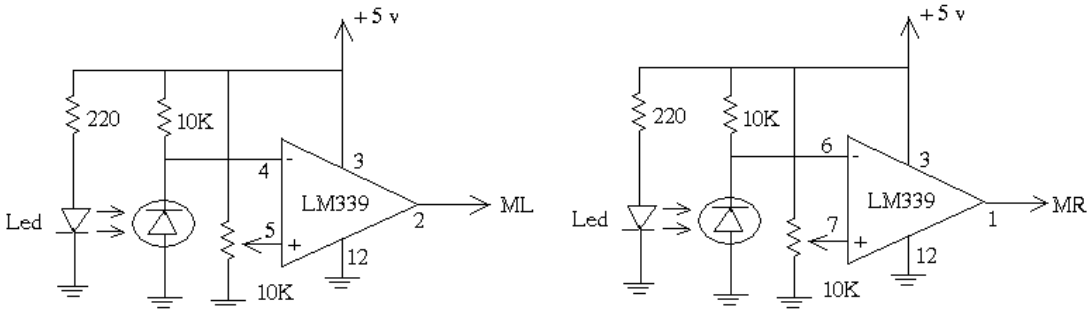
```
{
    IT0 = 1;EX0 = 1; EA  = 1;
   //Khởi tạo PWM
   CMOD   = 0x00;
   CCAPM1 = 0x42;//Chân P1.4
   CR = 1;
   CCAP1H = 0xFF; //Tốc đô = 0
   dir = 0;//quay thuận
   bra=0;//1 là thẳng
   start_LCD();
   giao_dien("TRUONG DHSPKT TPHCM","BRA=0", "ADC=0","DIR=0");
   while(1)
    {
        convert_ADC();
        if(nFlags_ADC)
        {
             nFlags_ADC = 0;
             sprintf(buf_lcd,"%d",(unsigned)bra);
            write_str(buf_lcd,0XC0+4);
             sprintf(buf_lcd,"%3d",speed);
            write_str(buf_lcd,0X94+4);
             sprintf(buf_lcd,"%d",(unsigned)dir);
            write_str(buf_lcd,0XD4+4);
             dir = chieu;
             bra = thang;
          }
      }

}
```

II. <u>Hệ thống dò đường cho Robot tự hành:</u>

1.Sơ đồ vị trí cảm biến:



2.Sơ đồ mạch cảm biến:



+ Hai cảm biến quang LSR, LSL tác động mức thấp khi cán vào line.

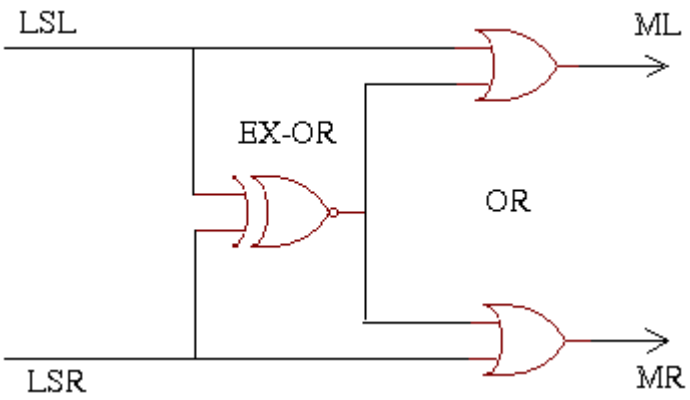
+ Hai motor ML, MR tác động mức cao.

+ Phương trình điều khiển có dạng:

$$ML = LSL + \overline{(LSL \oplus LSR)}$$

$$MR = LSR + \overline{(LSL \oplus LSR)}$$

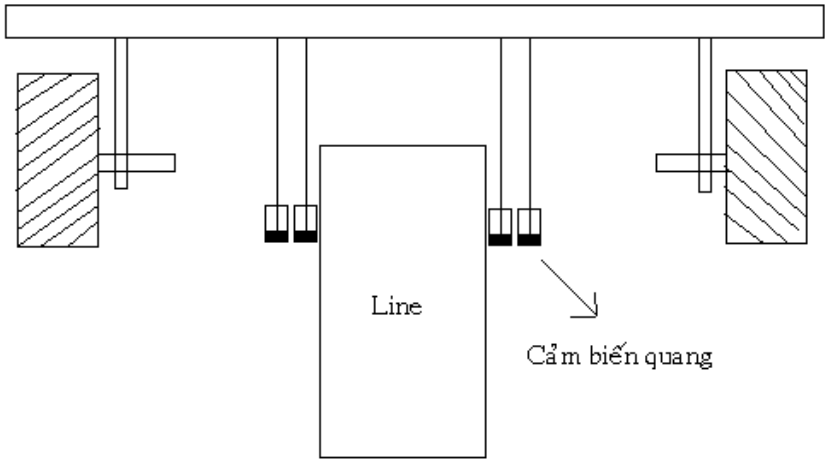
+ Thực hiện bằng phần cứng:



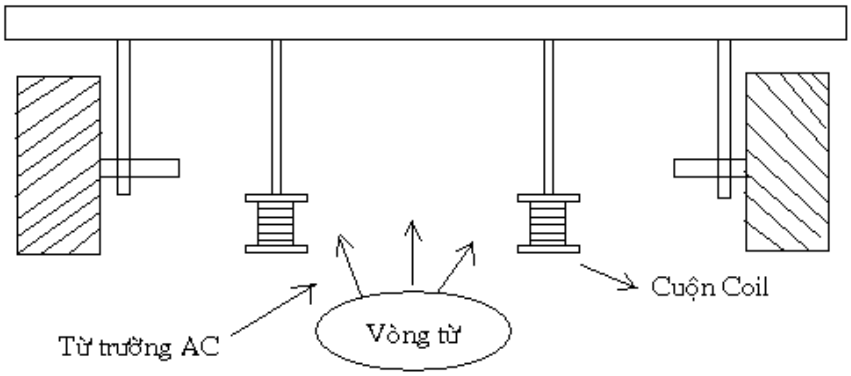
+Thực hiện bằng chương trình:

```
ML = LSL | !(LSL ^ LSR) ;
MR = LSR | !(LSL ^ LSR) ;
```

+ Bố trí cảm biến quang:

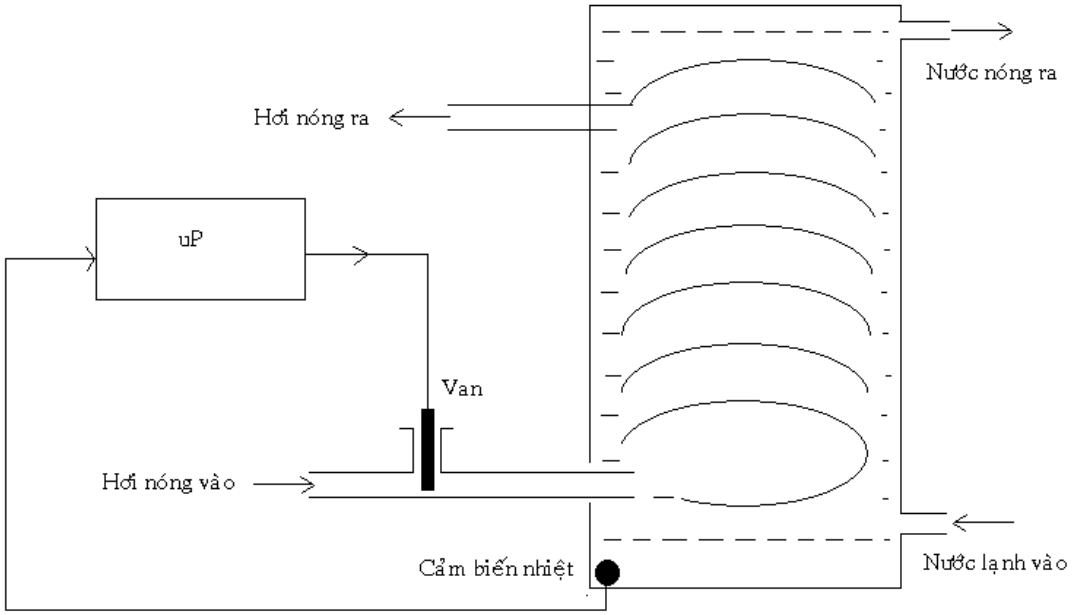


+ loại sử dụng cảm biến vòng từ:



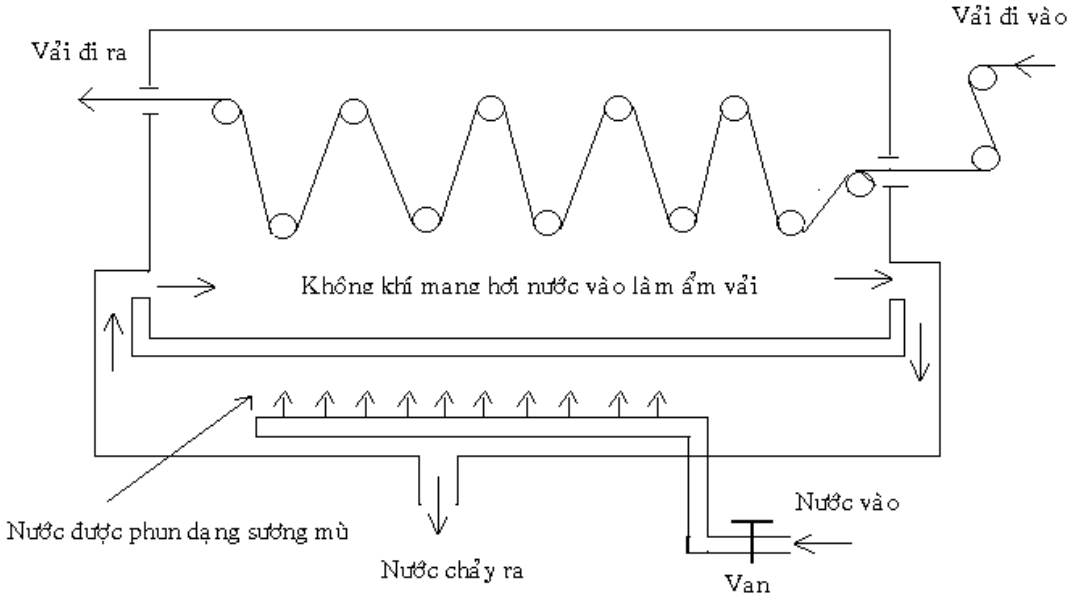
III.Bài tập:

Bài 1 : Viết chương trình điều khiển hệ thống đun nóng bằng hơi nước:



Van được điều khiển sao cho tỉ lệ với độ nóng trong bồn.

Bài 2 : Viết chương trình điều khiển hệ thống làm ẩm vải trong công nghệ nhuộm/dệt:



Van được điều khiển sao cho tỉ lệ với độ ẩm của vải