

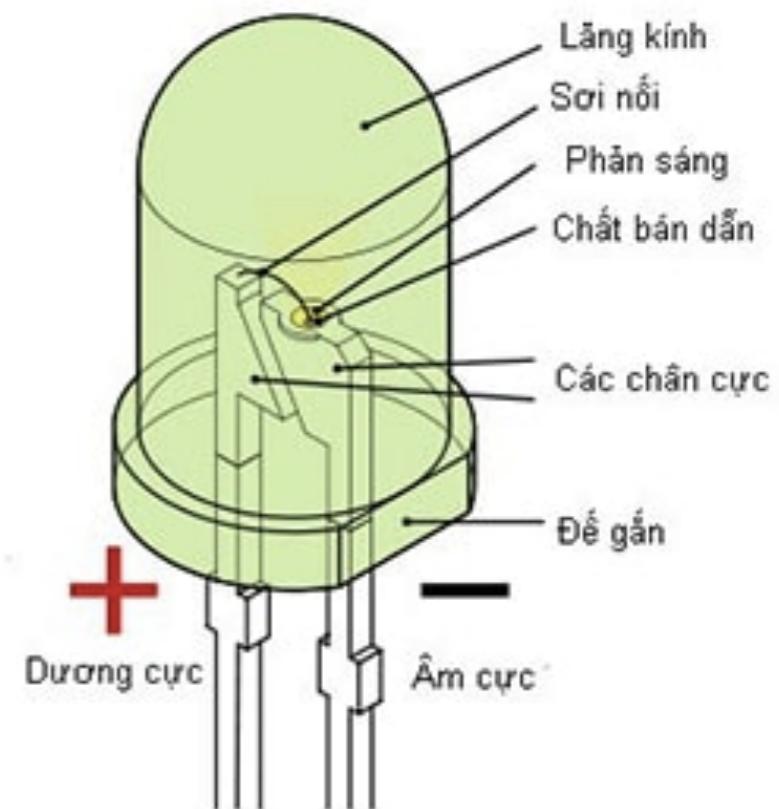
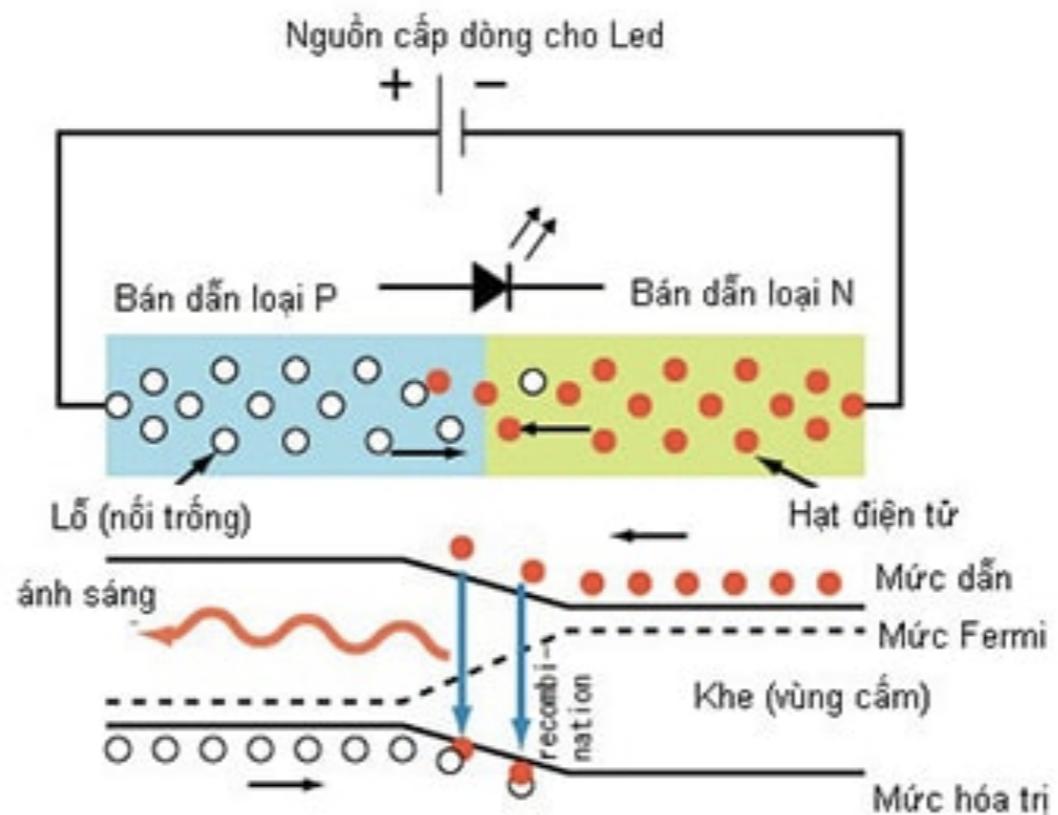
# Phản Lý thuyết

1. Giới thiệu chung - Mô hình hệ VXL - Nguyên tắc hoạt động
2. Cấu trúc và hoạt động của vi xử lý 8085
3. Quá trình thực hiện 1 lệnh trong VXL 8085
4. Giới thiệu về vi điều khiển PIC
5. Bộ công cụ nạp chương trình, công cụ mô phỏng vi điều khiển
6. Bộ định thời Timer
7. **Ghép nối với bộ hiển thị**
8. ADC
9. Giao tiếp truyền dữ liệu
10. Ngắt
11. PWM

# Bộ hiển thị

- Led đơn
- Led 7 thanh
- Ma trận LCD
- Ma trận LED

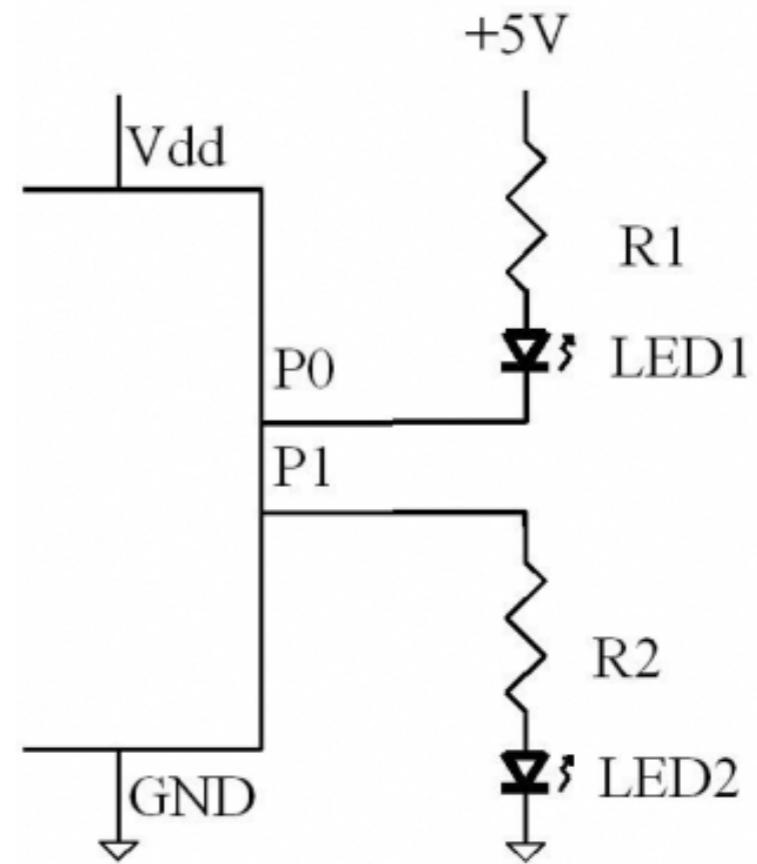
# Kết nối với Led đơn



- Công suất tiêu thụ thấp
- Màu sắc phụ thuộc loại bán dẫn chế tạo nên LED

# Kết nối với Led đơn

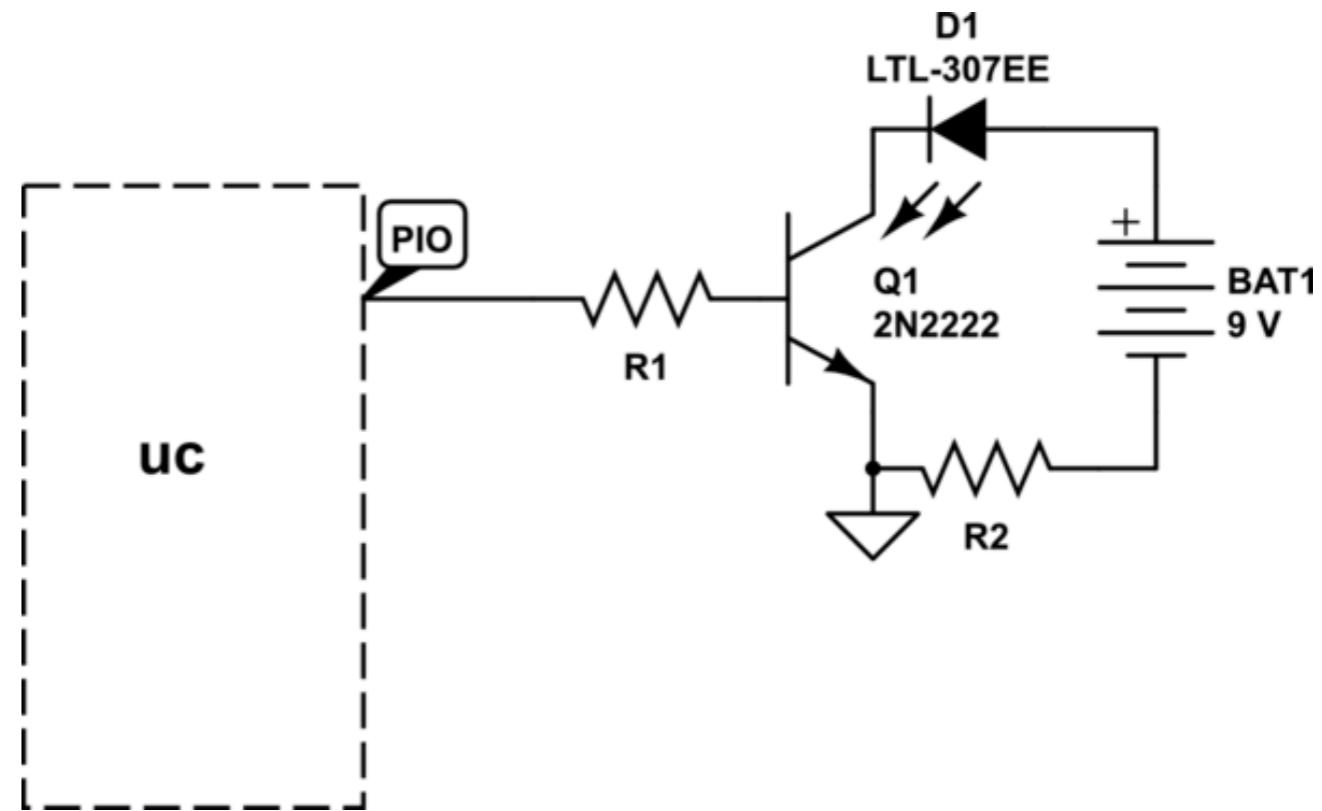
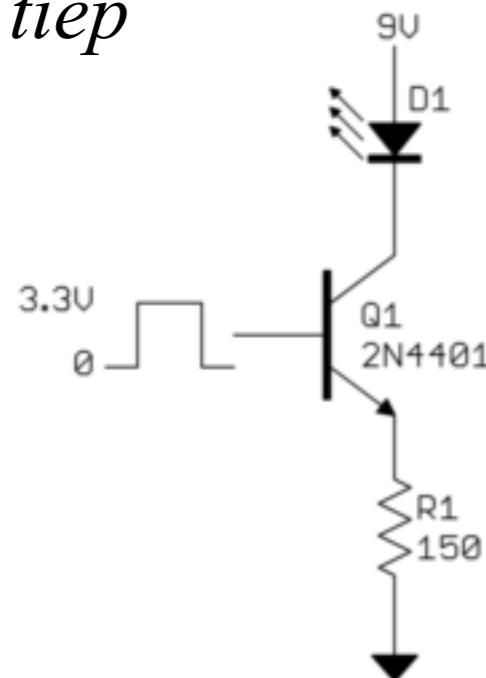
Mắc trực tiếp



- LED1 mắc kiểu bơm dòng - mắc kiểu anode chung (common anode) → thường được sử dụng
- LED2 mắc kiểu nuốt dòng (tiêu thụ dùng - công suất của VĐK) - mắc kiểu cathode chung (common cathode)
- Các điện trở R1 và R2 được sử dụng để hạn chế dòng điện đi vào/ra các chân vi điều khiển

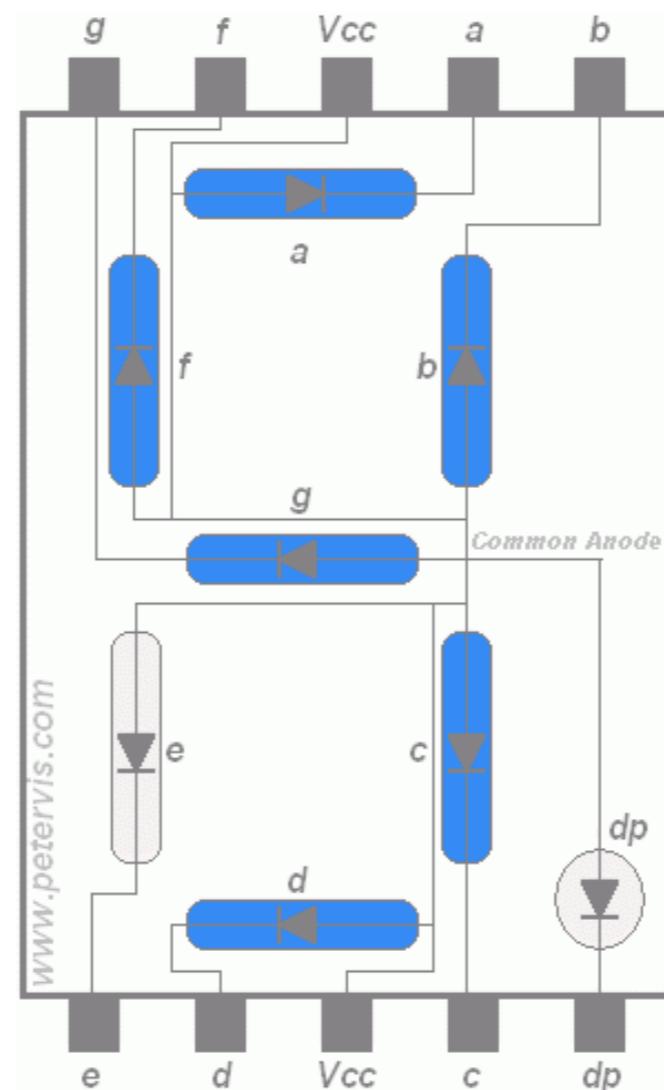
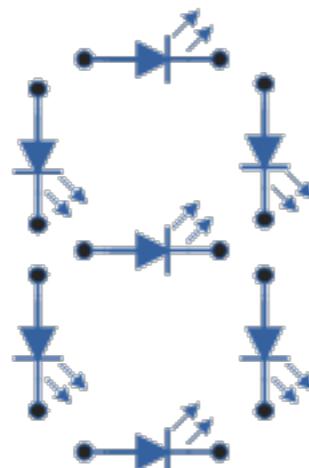
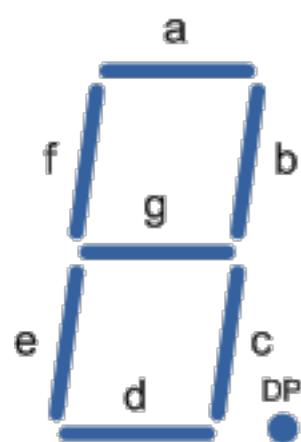
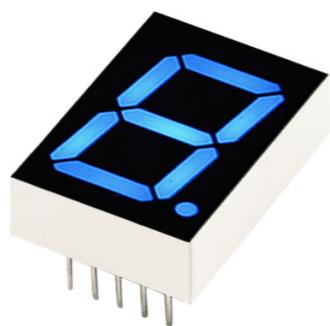
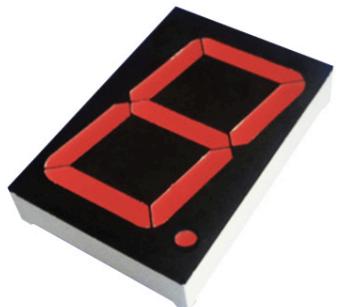
# Kết nối với Led đơn

Mắc gián tiếp

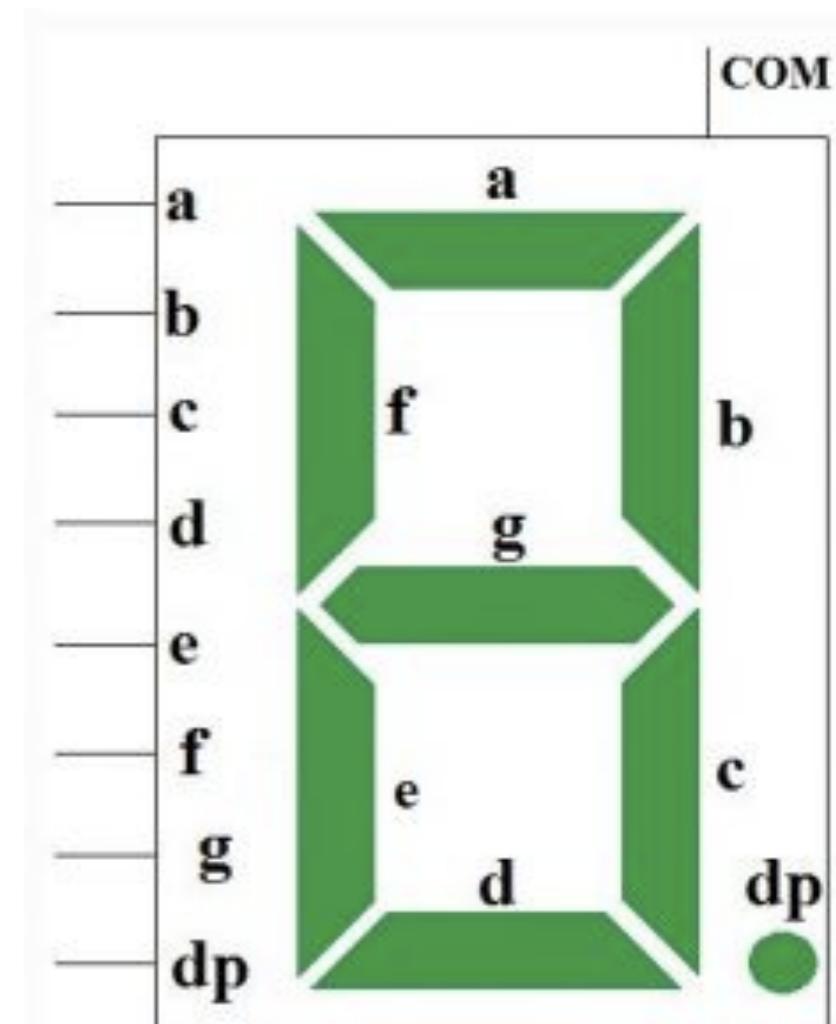


- Mắc gián tiếp sử dụng 1 transistor để cách ly phần công suất giữa LED và chân của vi điều khiển
- Mục đích là để bảo vệ

# Led 7 thanh

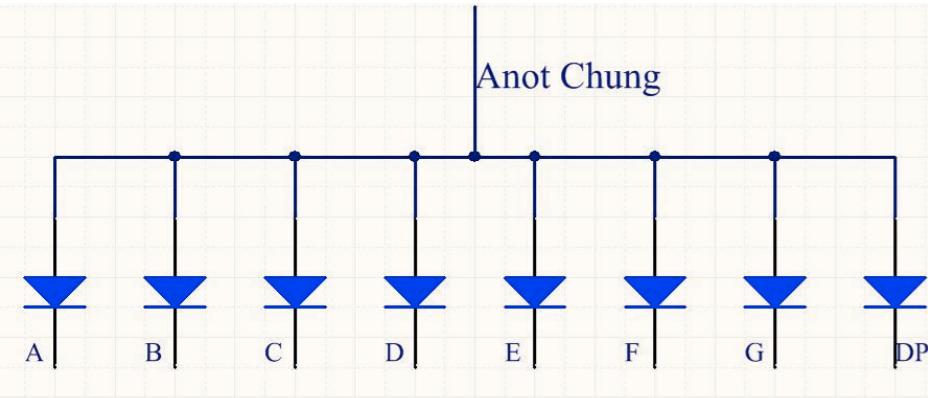


© www.petervis.com

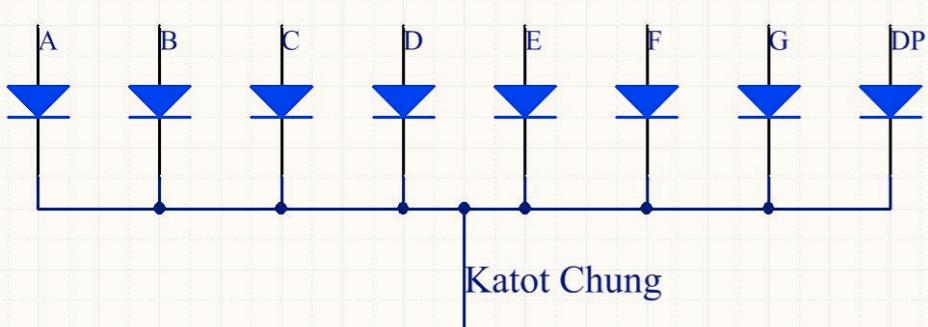


Các led đơn sắp đặt theo hình số 8 để hiển thị số

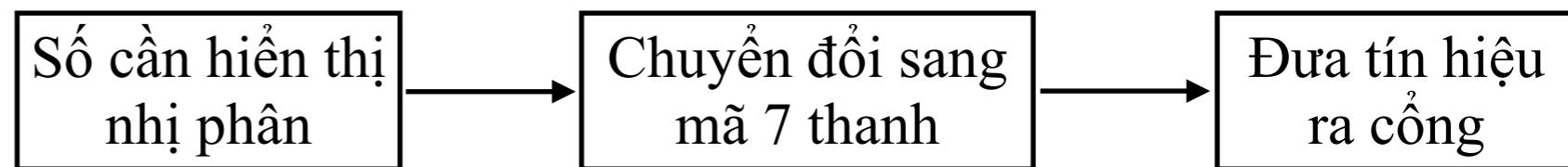
# Led 7 thanh



0 sáng. 1 tối



1 sáng. 0 tối

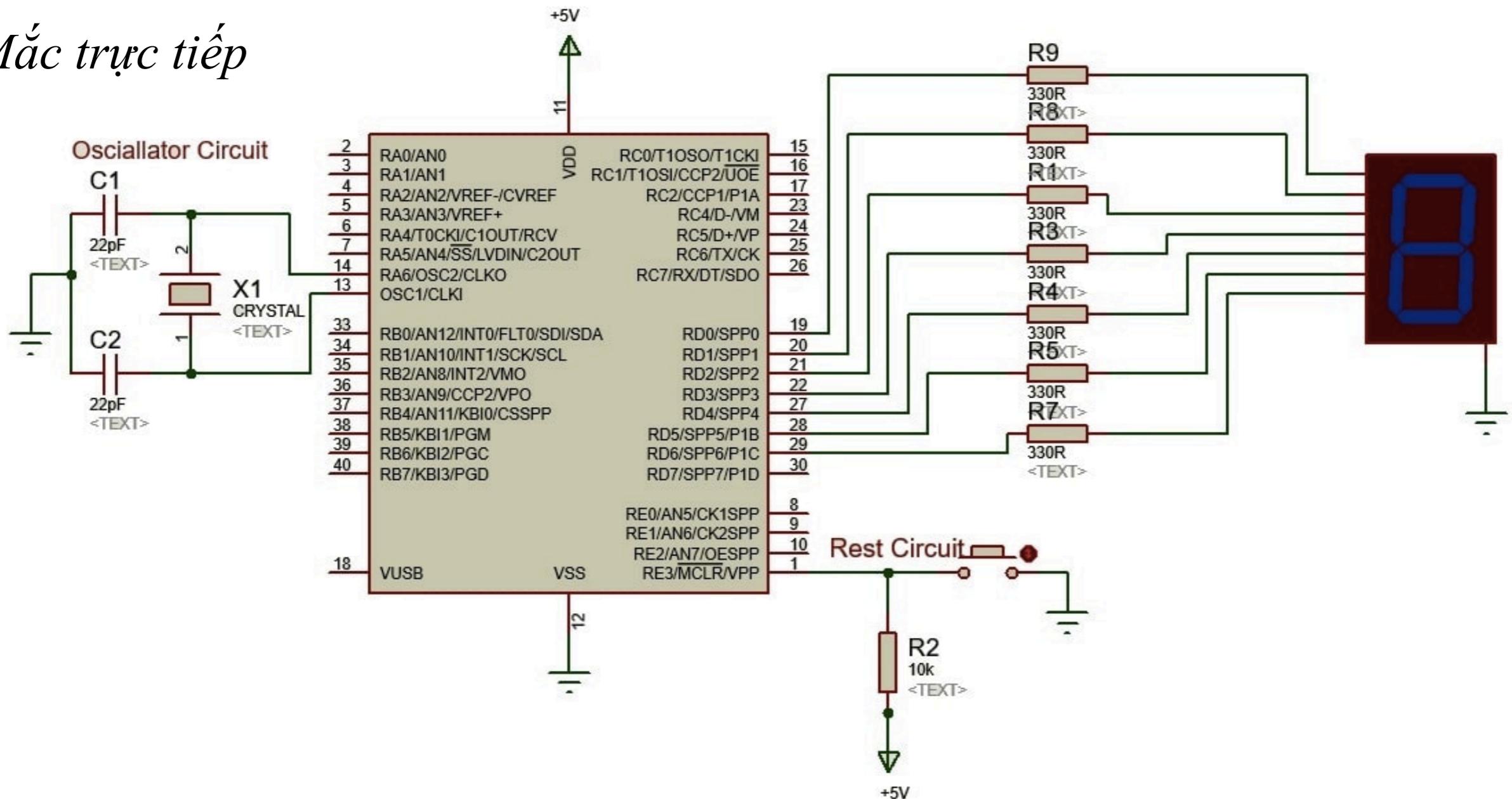


Numbers	Common Cathode		Common Anode	
	(DP)GFEDCBA	HEX Code	(DP)GFEDCBA	HEX Code
0	00111111	0x3F	11000000	0xC0
1	00000110	0x06	11111001	0xF9
2	01011011	0x5B	10100100	0xA4
3	01001111	0x4F	10110000	0xB0
4	01100110	0x66	10011001	0x99
5	01101101	0x6D	10010010	0x92
6	011111101	0x7D	10000010	0x82
7	00000111	0x07	11111000	0xF8
8	01111111	0x7F	10000000	0x80
9	01101111	0x6F	10010000	0x90

Hiển thị số nào cần đưa mã 7 thanh tương ứng vào led

# Led 7 thanh

Mắc trực tiếp



- Mỗi Led 7 thanh nối trực tiếp với 1 cổng truyền dữ liệu
- Thích hợp với việc hiển thị số lượng led 7 thanh ít

# Led 7 thanh

Măc trực tiếp

- Cổng portD nối với led 7 thanh măc theo kiểu cathode chung

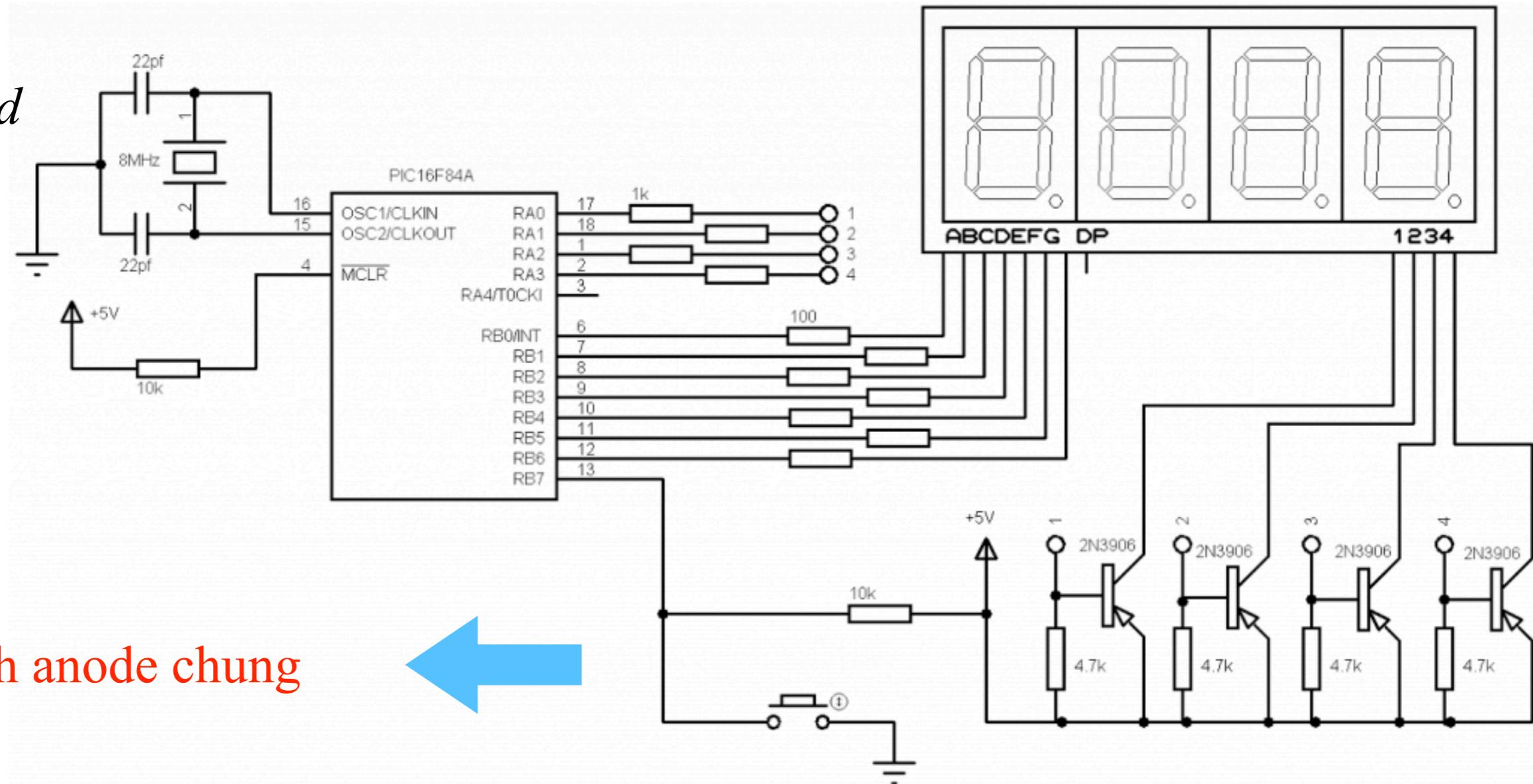
```
#include<16F877A.h>
#use delay(clock=20000000)
Unsigned char ma7thanh[]={0x3F,0x06,
0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F} → Khai báo mảng để chuyển
mã từ nhị phân sang 7 thanh
Void main()
{ while(1)
{
    for (int i=0;i<10;i++)
    {
        output_D(ma7thanh[i]); → Đưa ra cổng D giá trị thứ 7 trong mảng ma7thanh
        delay_ms(1000);
    }
}
}

```

Phân tử	0	1	2	3	4	5	6	7	8	9
Giá trị	3Fh	06h	5Bh	4Fh	66h	6Dh	7Dh	07h	7Fh	6Fh

# Led 7 thanh

Quét Led



Mạch anode chung

- Dữ liệu của tất cả các led đều được cung cấp từ 1 cổng
- Mỗi led được kích hoạt khi tín hiệu điều khiển từ BJT tích cực
- Mỗi led sáng trong khoảng thời gian ngắn đủ lưu trữ hình ảnh trong võng mạc của người xem (tối thiểu 24 hình/s)

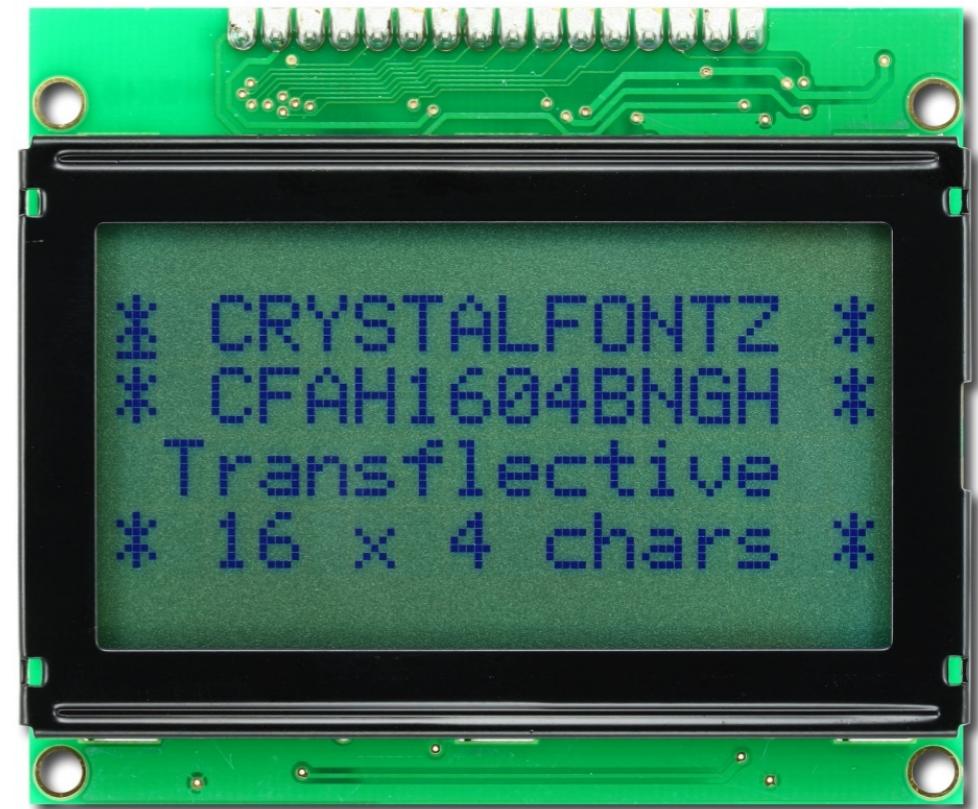
# Led 7 thanh

```
#include <16F877A.h>
#use delay(crystal=8000000)
short s; // Used to know button position
unsigned int digit, digit1, digit10, digit100,digit1000;
unsigned long i;
unsigned int seg(unsigned int num) {
    switch (num) {
        case 0 : return 0xC0;
        case 1 : return 0xF9;
        case 2 : return 0xA4;
        case 3 : return 0xB0;
        case 4 : return 0x99;
        case 5 : return 0x92;
        case 6 : return 0x82;
        case 7 : return 0xF8;
        case 8 : return 0x80;
        case 9 : return 0x90;
    }
}
void main() {
    while(TRUE) {
        if(input(PIN_B7) == 1)
            s = 1;
        if(s == 1) {
            if(input(PIN_B7) == 0) {
                s = 0;
                i++;
                if(i > 9999)
                    i = 0;
            }
        }
        output_a(0x0F); // Turn off all displays
        output_b(digit1000); // Send thousands digit
        output_a(0x0E); // Turn on display for thousands
        delay_ms(5);
    }
    digit = i % 10;
    digit1 = seg(digit);
    output_a(0x0F); // Turn off all displays
    output_b(digit1); // Send ones digit
    output_a(0x07); // Turn on display for ones
    delay_ms(5);
    digit = (i / 10) % 10;
    digit10 = seg(digit);
    output_a(0x0F); // Turn off all displays
    output_b(digit10); // Send tens digit
    output_a(0x0B); // Turn on display for tens
    delay_ms(5);
    digit = (i / 100) % 10;
    digit100 = seg(digit);
    output_a(0x0F); // Turn off all displays
    output_b(digit100); // Send hundreds digit
    output_a(0x0D); // Turn on display for hundreds
    delay_ms(5);
    digit = (i / 1000) % 10;
    digit1000 = seg(digit);
    output_a(0x0F); // Turn off all displays
    output_b(digit1000); // Send thousands digit
    output_a(0x0E); // Turn on display for thousands
    delay_ms(5);
}
```

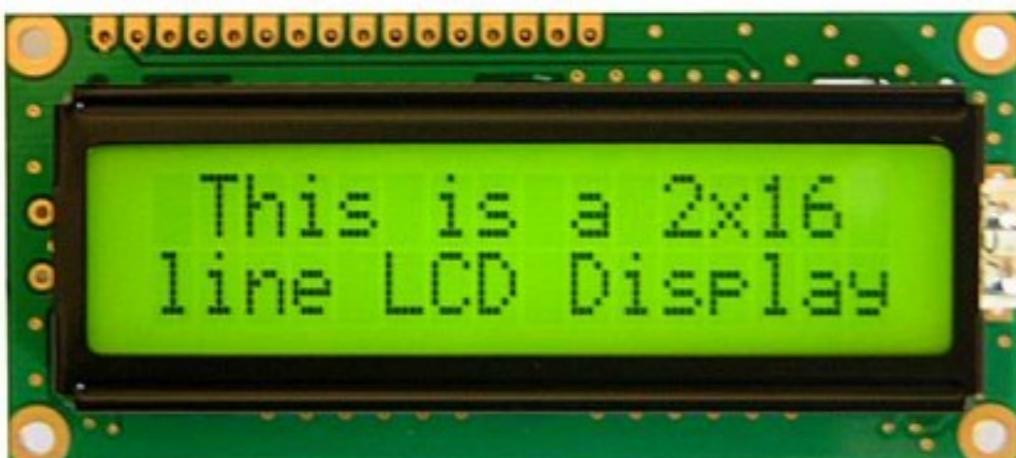
# LCD



Loại 1 hàng ký tự



Loại 4 hàng ký tự

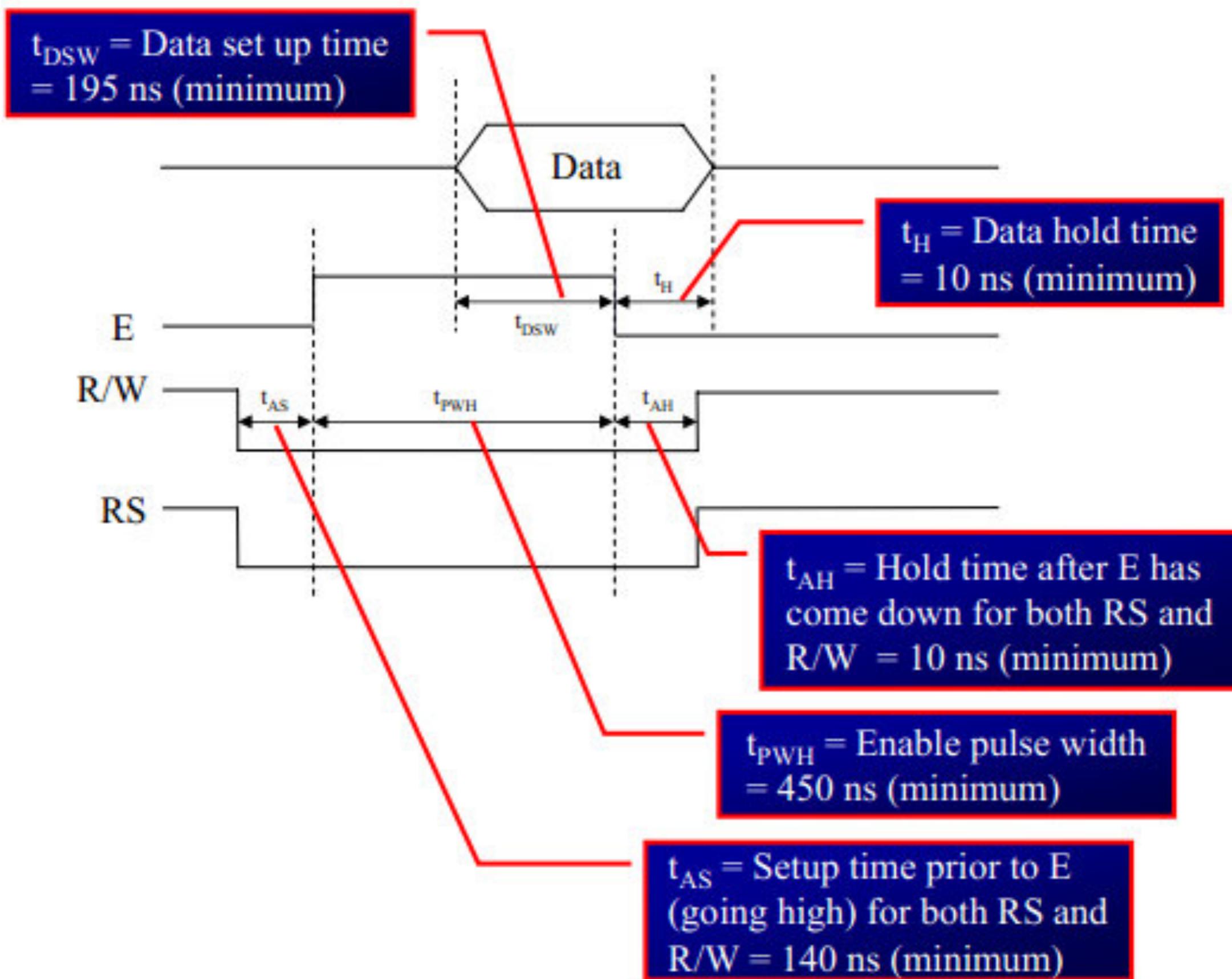


Loại 2 hàng ký tự

# LCD

Chân	Ký hiệu	Mô tả
1	V <sub>ss</sub>	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với <b>GND</b> của mạch điều khiển
2	V <sub>DD</sub>	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với <b>V<sub>CC</sub>=5V</b> của mạch điều khiển
3	V <sub>EE</sub>	Điều chỉnh độ <b>tương phản</b> của LCD.
4	RS	Chân chọn thanh ghi (Register select). + Logic “0”: LCD nhận lệnh từ vi điều khiển + Logic “1”: Bus DB0-DB7 nhận dữ liệu từ VĐK.
5	R/W	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic “0” để LCD hoạt động ở chế độ ghi, hoặc nối với logic “1” để LCD ở chế độ đọc.
6	E	Read/Write enable
7 - 14	DB0 - DB7	Tám đường của bus dữ liệu dùng để trao đổi thông tin với MPU. Có 2 chế độ sử dụng 8 đường bus này : + <b>Chế độ 8 bit</b> : Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. + <b>Chế độ 4 bit</b> : Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7
15	-	Nguồn dương cho đèn nền
16	-	GND cho đèn nền

# LCD



# LCD với giao tiếp 8 bit

## Gửi lệnh ra LCD

Bước 1: R/W = 0 để  
chọn chế độ ghi

Bước 2: RS = 0 để xác định  
là ghi lệnh

Bước 3: Gửi lệnh ra D<sub>7</sub>...D<sub>0</sub>

Bước 4: Tạo xung trên chân E để  
cho phép ghi vào LCD

Bước 5: Tạo trễ để LCD  
thực hiện xong lệnh

Mã lệnh	Mô tả
0x01	Xóa màn hình
0x02	Đưa con trỏ về vị trí home (đầu màn hình)
0x06	Đưa con trỏ tới vị trí tiếp theo sau khi hiển thị
0x0C	Bật hiển thị và tắt con trỏ
0x0E	Bật hiển thị và bật con trỏ
0x80	Di chuyển con trỏ về đầu dòng 1
0xC0	Di chuyển con trỏ về đầu dòng 2
0x38	Giao tiếp 8 bit, 2 dòng, font 5 x 7
0x28	Giao tiếp 4 bit, 2 dòng, font 5 x 7

# LCD với giao tiếp 8 bit

## Gửi dữ liệu ra LCD

Bước 1: R/W = 0 để chọn chế độ ghi

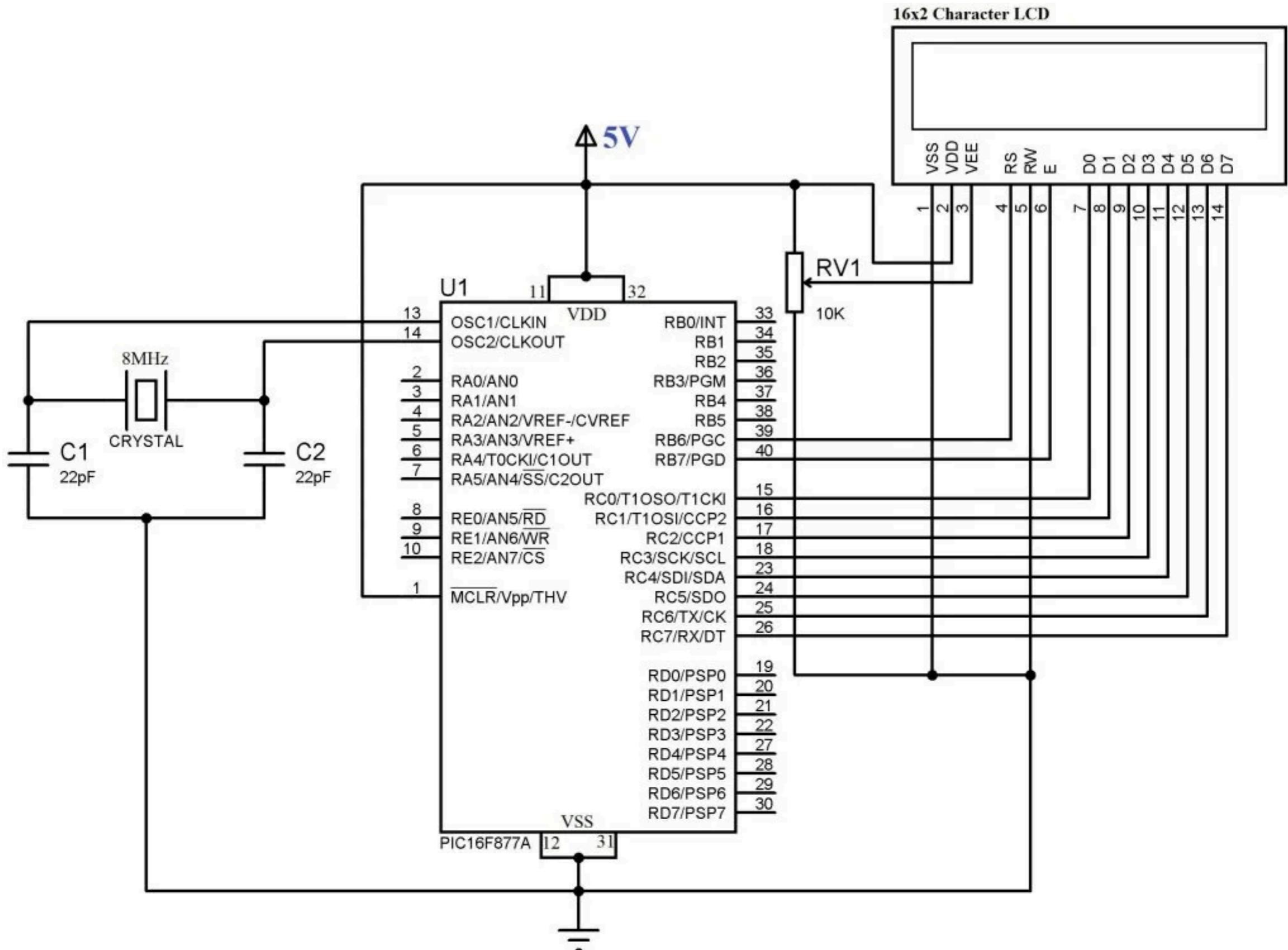
Bước 2: RS = 1 để xác định là ghi dữ liệu chứ không phải là lệnh

Bước 3: Gửi mã ASCII ra chân D<sub>7</sub>...D<sub>0</sub>

Bước 4: Tạo xung trên chân E để cho phép ghi vào LCD

Bước 5: Tạo trễ để LCD hiển thị ký tự (37μs)

# LCD với giao tiếp 8 bit



# LCD với giao tiếp 4 bit

## Gửi lệnh ra LCD

Bước 1: R/W = 0 để chọn chế độ ghi

Bước 2: RS = 0 để xác định là ghi lệnh

Bước 3: Gửi **4 bit cao** của mã lệnh ra D<sub>3</sub>...D<sub>0</sub>

Bước 4: Tạo xung trên chân E để cho phép ghi vào LCD

Bước 5: Tạo trễ để LCD thực hiện xong lệnh cho **4 bit cao**

Bước 6: Gửi **4 bit thấp** của mã lệnh ra D<sub>3</sub>...D<sub>0</sub>

Bước 7: Tạo xung trên chân E để cho phép ghi vào LCD

Bước 8: Tạo trễ để LCD thực hiện xong lệnh cho **4 bit thấp**

# LCD với giao tiếp 4 bit

## Gửi dữ liệu ra LCD

Bước 1: R/W = 0 để chọn chế độ ghi

Bước 2: RS = 0 để xác định là ghi lệnh

Bước 3: Gửi **4 bit cao** của dữ liệu ra D<sub>3</sub>...D<sub>0</sub>

Bước 4: Tạo xung trên chân E để cho phép ghi vào LCD

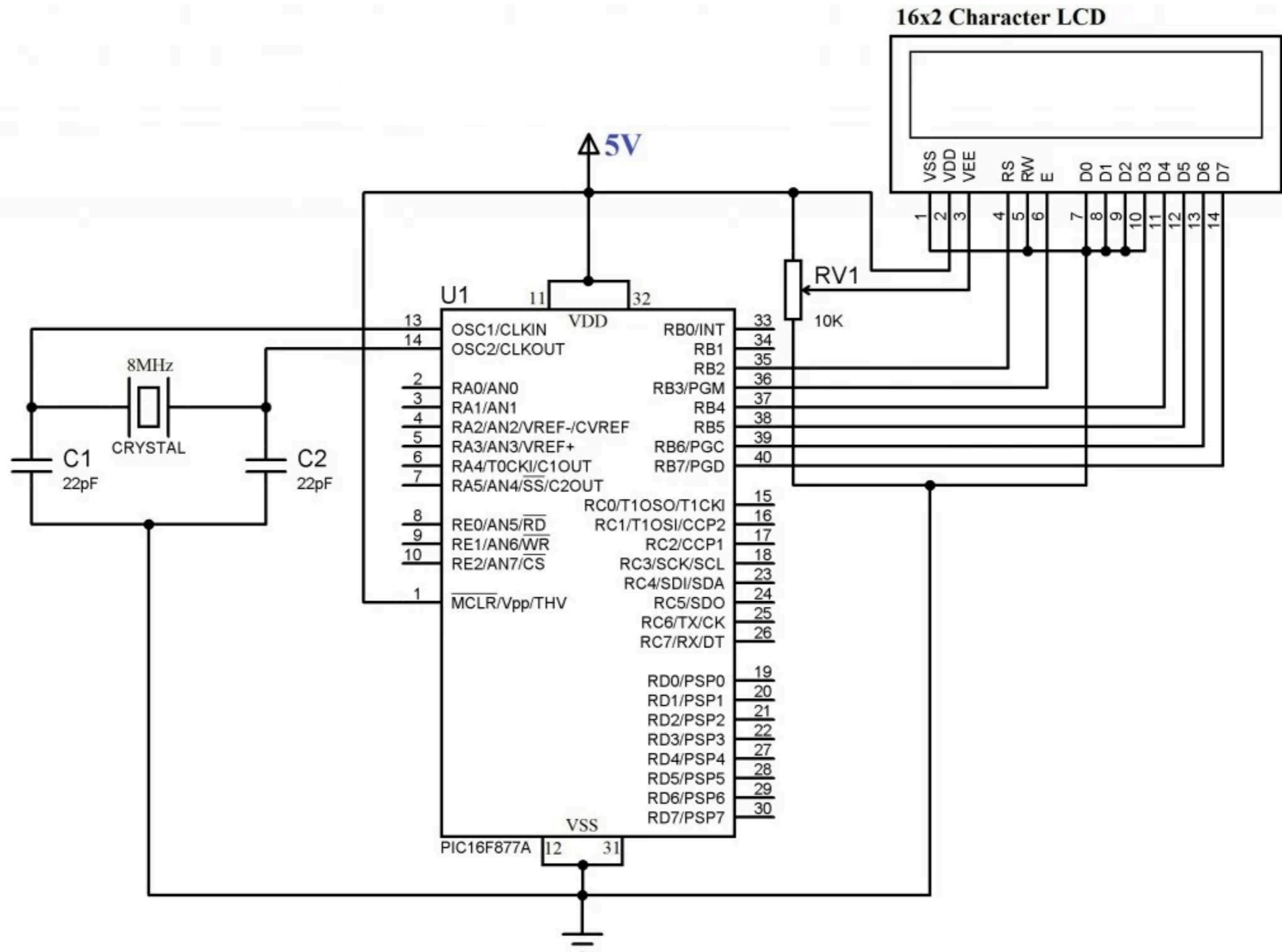
Bước 5: Tạo trễ để LCD thực hiện xong lệnh cho **4 bit cao**

Bước 6: Gửi **4 bit thấp** của dữ liệu ra D<sub>3</sub>...D<sub>0</sub>

Bước 7: Tạo xung trên chân E để cho phép ghi vào LCD

Bước 8: Tạo trễ để LCD thực hiện xong lệnh cho **4 bit thấp**

# LCD với giao tiếp 4 bit



# Thư viện lcd.h

Định nghĩa các chân kết nối với LCD theo từng chế độ

## Chế độ 8 bit

```
#define RS RB6  
#define EN RB7  
#define D0 RC0  
#define D1 RC1  
#define D2 RC2  
#define D3 RC3  
#define D4 RC4  
#define D5 RC5  
#define D6 RC6  
#define D7 RC7
```

## Chế độ 4 bit

```
#define RS RB2  
#define EN RB3  
#define D4 RB4  
#define D5 RB5  
#define D6 RB6  
#define D7 RB7
```

# Thư viện lcd.h

`lcd8_init()` và `lcd4_init()` được sử dụng để khởi tạo hoạt động cho lcd

`lcd8_clear()` và `lcd4_clear()` được sử dụng để xoá tất cả các ký tự trên lcd

`lcd8_set_cursor(x,y)` và `lcd4_set_cursor(x,y)` xác định vị trí con trỏ trên lcd

`lcd8_write_char(char)` và `lcd4_write_char(char)` đưa kí tự `char` ra lcd

`lcd8_write_string(s)` và `lcd4_write_string(s)` đưa chuỗi kí tự `s` ra lcd

`lcd8_shift_left()` và `lcd4_shift_left()` dịch trái nội dung trên lcd

`lcd8_shift_right()` và `lcd4_shift_right()` dịch phải nội dung trên lcd

# Ví dụ giao tiếp LCD chế độ 8 bit

```
#include<pic16F877a.h>

#define RS RB6
#define EN RB7
#define D0 RC0
#define D1 RC1
#define D2 RC2
#define D3 RC3
#define D4 RC4
#define D5 RC5
#define D6 RC6
#define D7 RC7

#include "lcd.h"

void main()
{
    int i;
    TRISB = 0x00;
    TRISC = 0x00;
    Lcd8_Init();

    while(1)
    {
        Lcd8_Set_Cursor(1,1);
        Lcd8_Write_String("Bo mon KTDT");
        for(i=0;i<15;i++)
        {
            delay_ms(1000);
            Lcd8_Shift_Left();
        }
        for(i=0;i<15;i++)
        {
            delay_ms(1000);
            Lcd8_Shift_Right();
        }
        Lcd8_Clear();
        Lcd8_Set_Cursor(2,1);
        Lcd8_Write_Char('K');
        Lcd8_Write_Char('5');
        Lcd8_Write_Char('9');
        delay_ms(2000);
    }
}
```

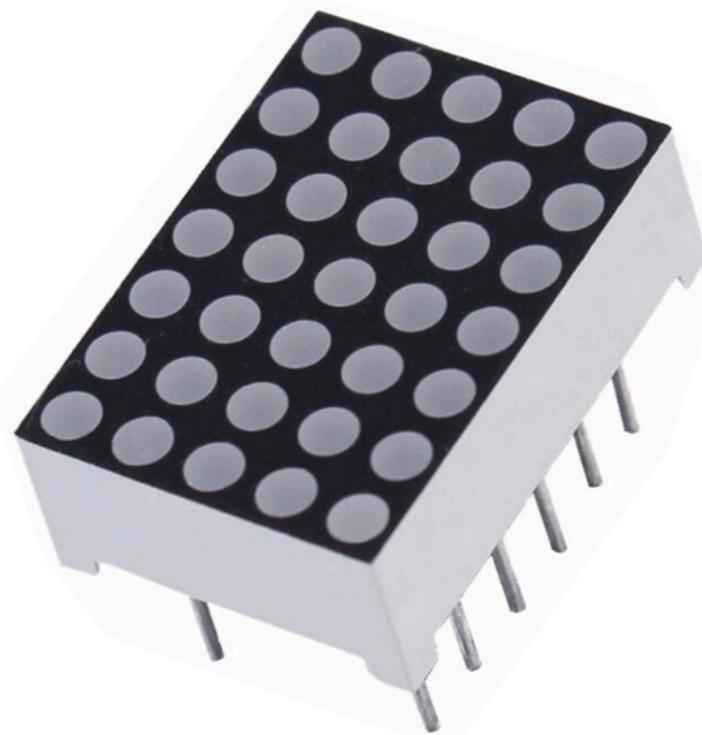
# Ví dụ giao tiếp LCD chế độ 4 bit

```
#include<pic16F877a.h>
#use delay(clock=2000000)
#define RS RB2
#define EN RB3
#define D4 RB4
#define D5 RB5
#define D6 RB6
#define D7 RB7
#include "lcd.h"
void main()
{
    int i;
    TRISB = 0x00;
    Lcd4_Init();
    while(1)
    {
        Lcd4_Set_Cursor(1,1);
        Lcd4_Write_String("Bo mon KTDT");
        for(i=0;i<15;i++)
        {
            delay_ms(1000);
            Lcd4_Shift_Left();
        }
        for(i=0;i<15;i++)
        {
            delay_ms(1000);
            Lcd4_Shift_Right();
        }
        Lcd4_Clear();
        Lcd4_Set_Cursor(2,1);
        Lcd4_Write_Char('K');
        Lcd4_Write_Char('5');
        Lcd4_Write_Char('9');
        delay_ms(2000);
    }
}
```

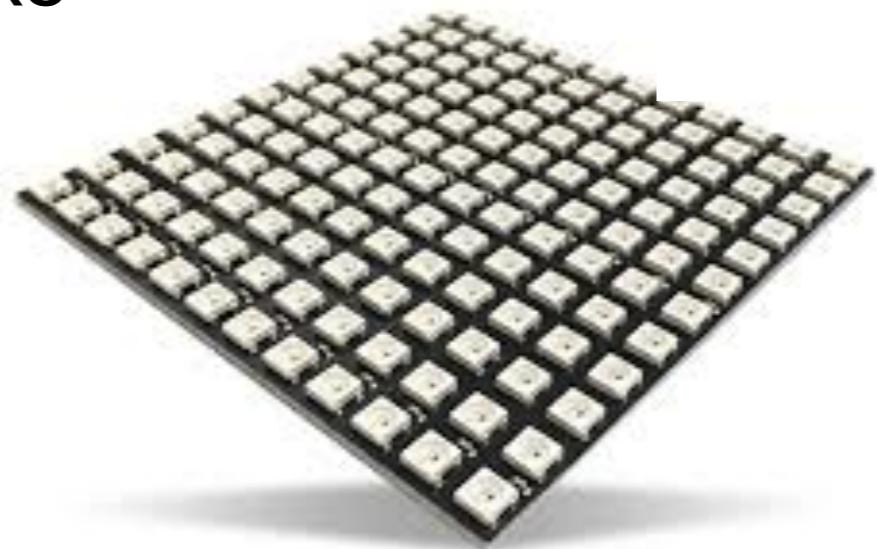
# Ma trận led



Ma trận 8x8

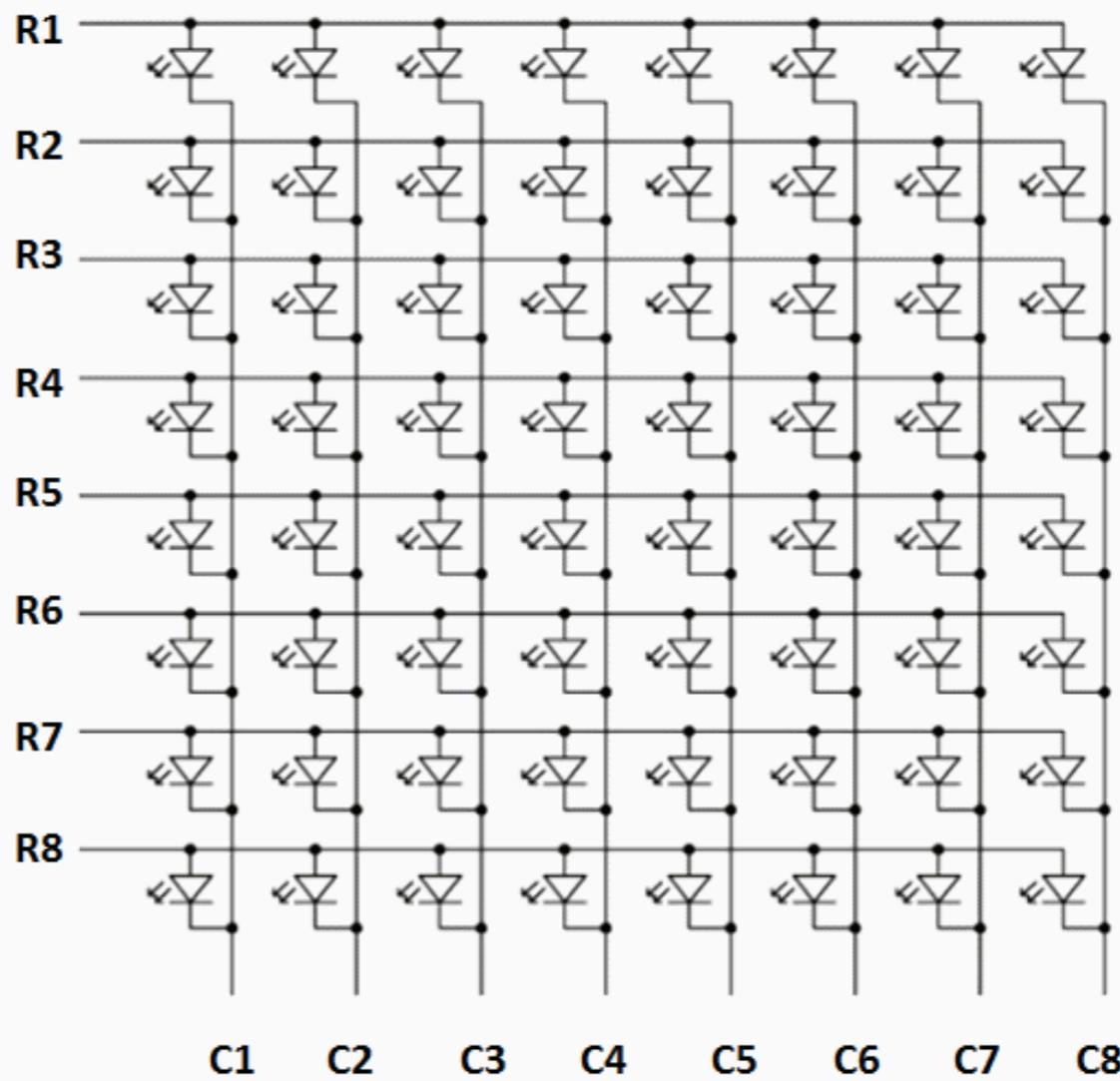
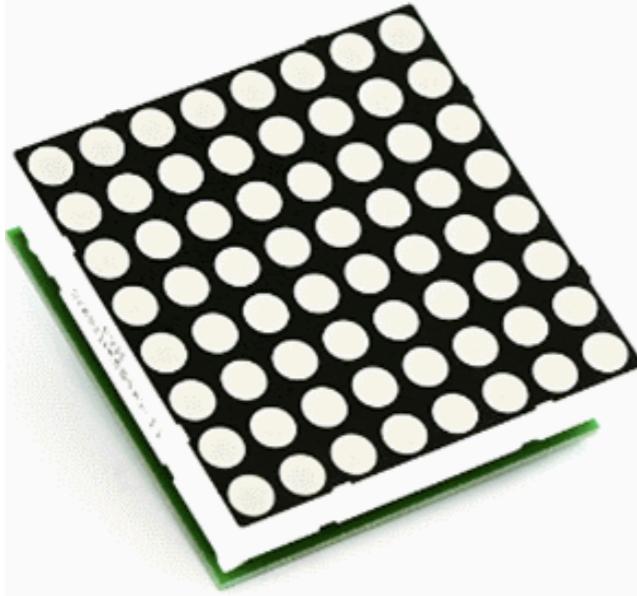


Ma trận 5x7



Ma trận kích thước lớn hơn

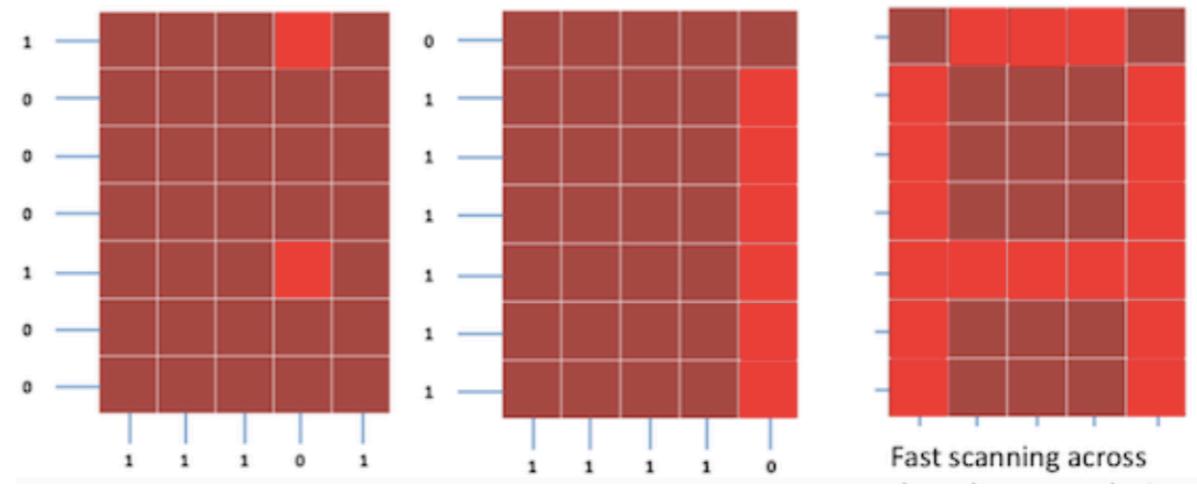
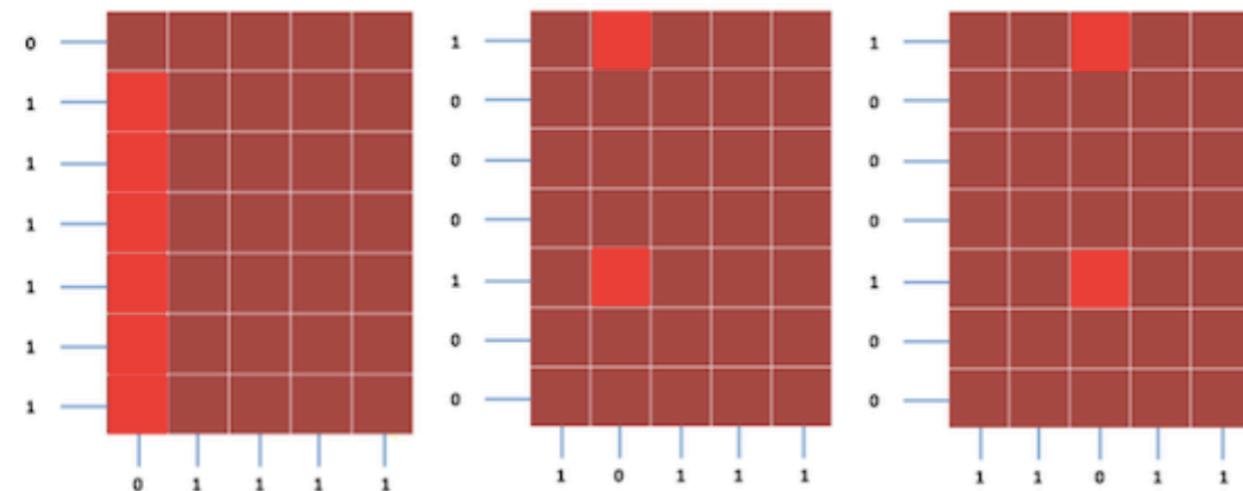
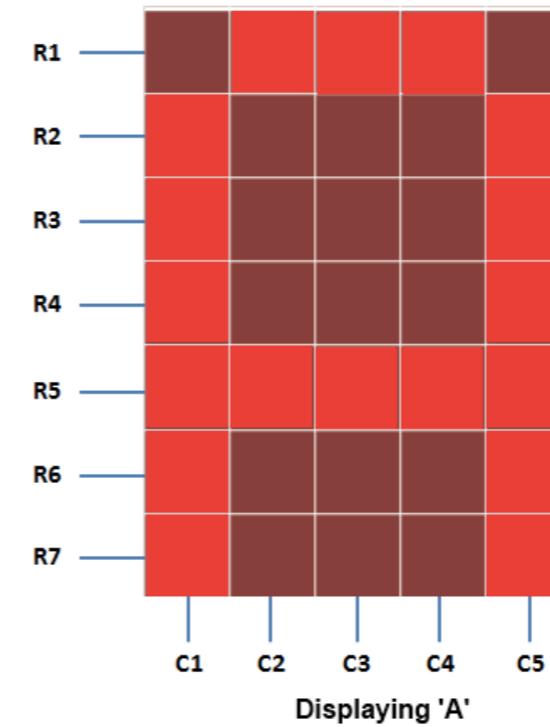
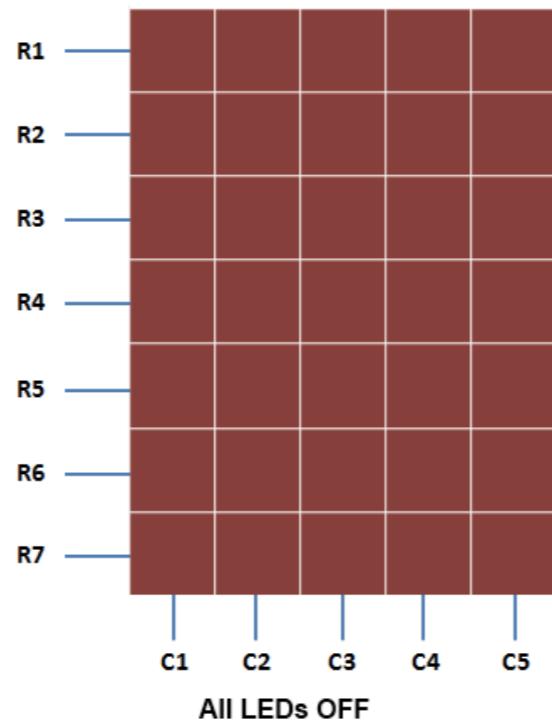
# Ma trận led



- Mỗi một led muốn sáng cần được cung cấp điện áp phù hợp tại đúng vị trí mong muốn
- Dữ liệu để làm led sáng cần được đưa ra cổng theo hàng và cột

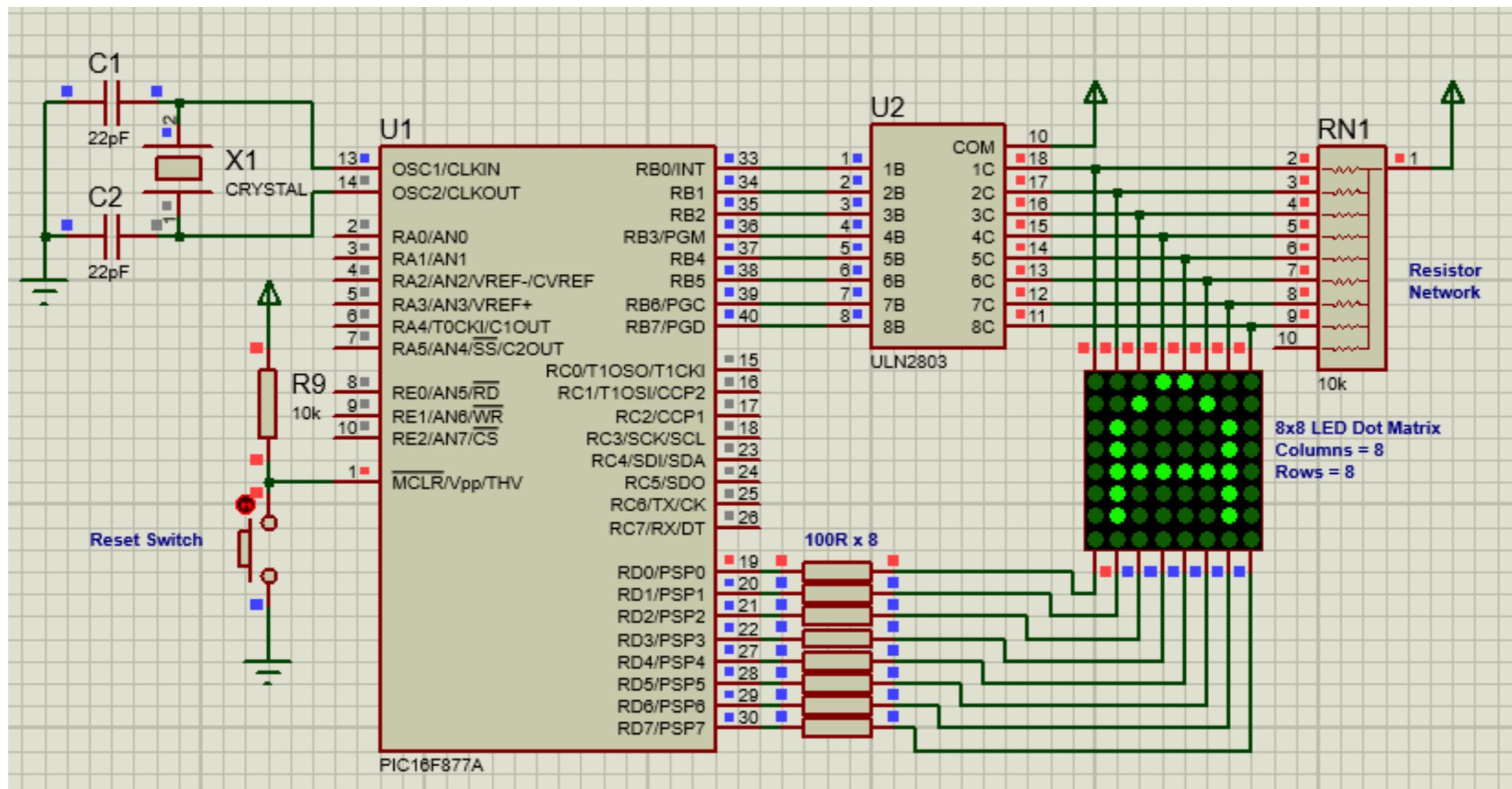
# Nguyên lý để điều khiển ma trận led

5x7 matrix of LEDs



Sáng lần lượt từng cột trong một khoảng thời gian đủ lớn

# Nguyên lý để điều khiển ma trận led



- 8 cột của ma trận led được điều khiển bởi cổng portD
- Dữ liệu (mã hóa ký tự hiển thị) được đẩy ra theo dữ liệu hàng thông qua portB

# Nguyên lý để điều khiển ma trận led

```
#include <16f877a.h>                                // Mảng dữ liệu theo hàng

#use delay (clock=4M)

#fuses brownout, hs, nowdt, nolvp

///////////////////////////////

#define col1 pin_d0
#define col2 pin_d1
#define col3 pin_d2
#define col4 pin_d3
#define col5 pin_d4
#define col6 pin_d5
#define col7 pin_d6
#define col8 pin_d7

///////////////////////////////

// Các biến sử dụng

unsigned int8 i, counter;

const char a_codes[16] = {
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
    0b01111100, Nhìn nghiêng 90° sẽ
    0b00010010, cho ta hình ký tự chữ
    0b00010001, "A"
    0b00010001,
    0b00010010,
    0b01111100,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
};

};
```

# Nguyên lý để điều khiển ma trận led

```
//////////
```

```
// Khai báo các hàm sử dụng
```

```
void matrix_scan (void);  
void mdelay (void);  
//////////
```

```
void main ()  
{
```

```
    output_b(0b00000000);  
    i=0;
```

```
    while(1)  
    {  
        for(counter=0; counter<16; counter++)  
        {  
            i=counter;  
            mdelay();  
        }  
    }
```

```
//////////
```

```
void mdelay (void)  
{  
    unsigned int8 count;  
    for(count=0; count<20; count++)  
        matrix_scan();  
}
```

# Nguyên lý để điều khiển ma trận led

```
void matrix_scan (void)
{
    output_b(a_codes[i+0]);
    output_high(col1);
    delay_ms(1);
    output_low (col1);

    output_b(a_codes[i+1]);
    output_high(col2);
    delay_ms(1);
    output_low (col2);

    output_b(a_codes[i+2]);
    output_high(col3);
    delay_ms(1);
    output_low (col3);

    output_b(a_codes[i+3]);
    output_high(col4);
    delay_ms(1);
    output_low (col4);

    output_b(a_codes[i+4]);
    output_high(col5);
    delay_ms(1);
    output_low (col5);

    output_b(a_codes[i+5]);
    output_high(col6);
    delay_ms(1);
    output_low (col6);

    output_b(a_codes[i+6]);
    output_high(col7);
    delay_ms(1);
    output_low (col7);

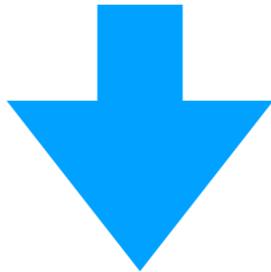
    output_b(a_codes[i+7]);
    output_high(col8);
    delay_ms(1);
    output_low (col8);
}
```

*Với mỗi cột*

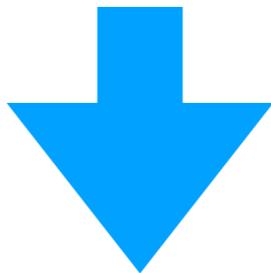
- Hàng thứ i được cấp dữ liệu
- Cột tương ứng được kích hoạt
- Trễ 1ms
- Xoá dữ liệu cột

# Nhược điểm của phương pháp này

Mỗi ma trận led cần được điều khiển  
8 bit hàng và 8 bit cột

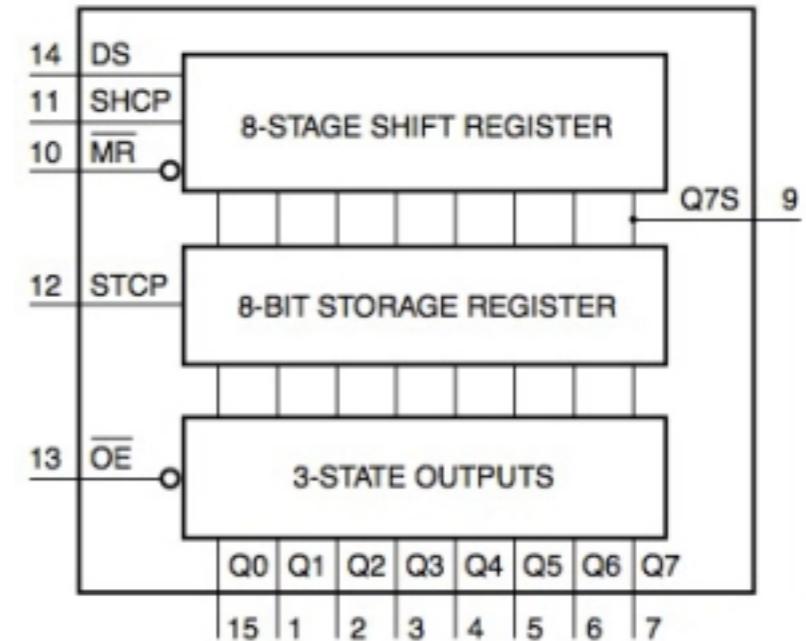
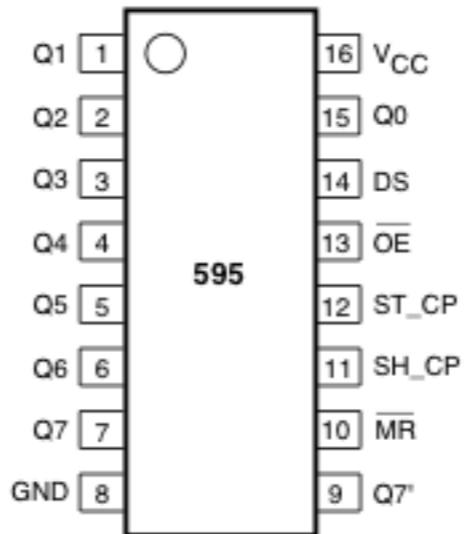
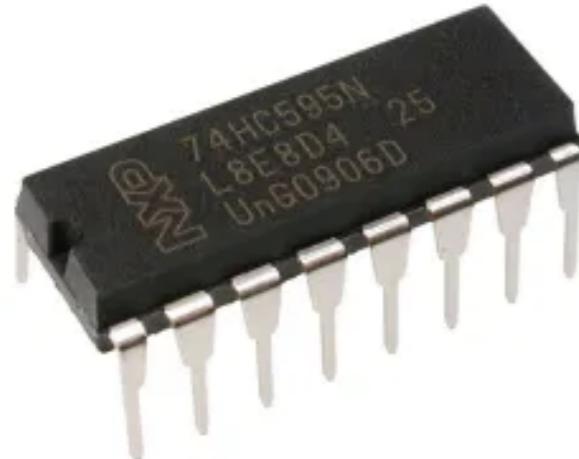


Tốn nhiều cổng điều khiển



Sử dụng IC 595

# IC 74HC595



DS: Data Shift

SH\_CP: SHift Clock Pulse

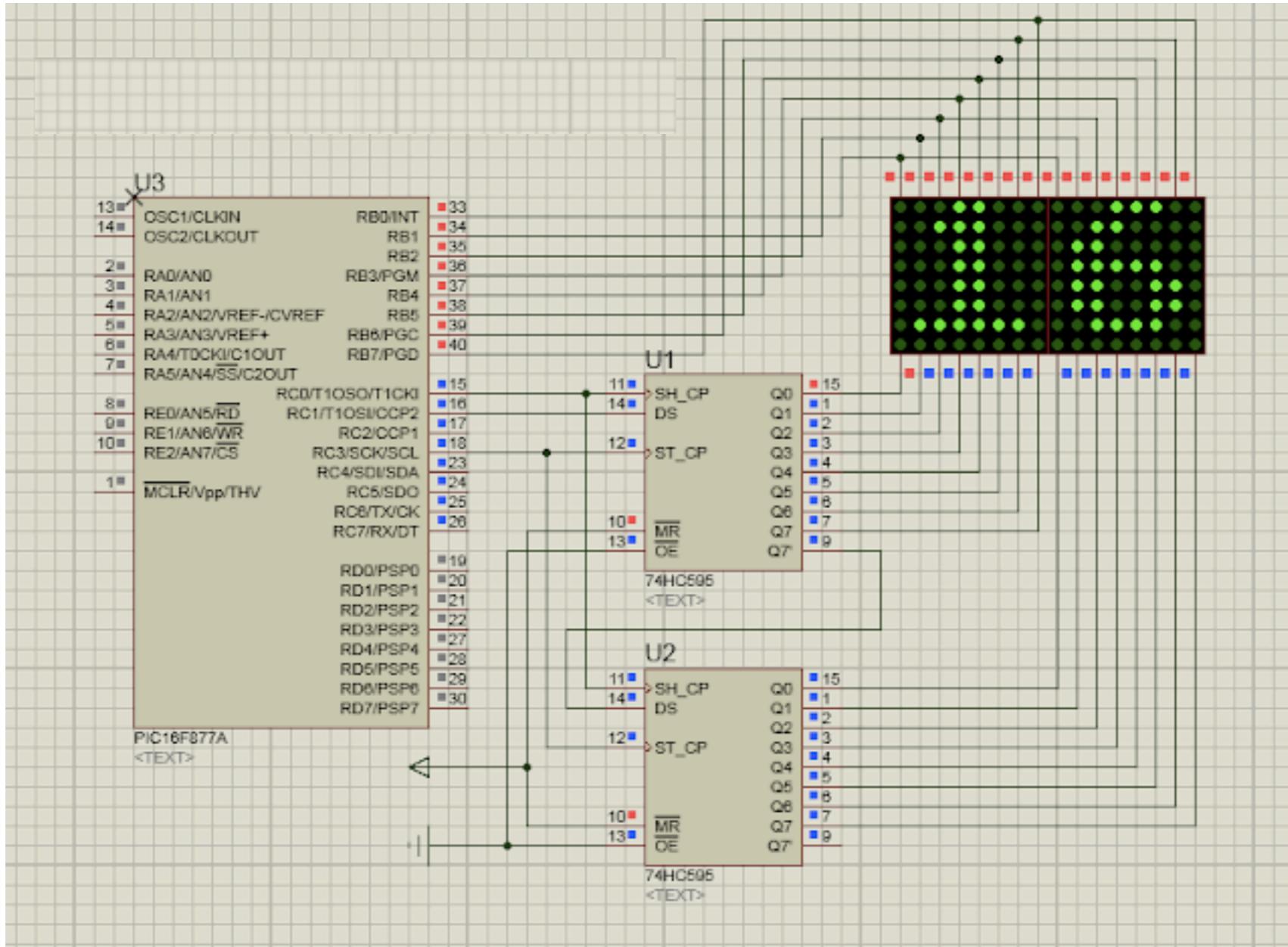
ST\_CP: STorage Clock Pulse

OE: Output Enable

MR: Master Reset

- Dữ liệu nối tiếp được đưa vào DS
- Có xung tác động vào SHCP → Dữ liệu sẽ vào thanh ghi shift
- 8 xung tác động vào SHCP → 8 bit sẽ nằm trong 8-state-shift-register
- có 1 xung tại STCP → dữ liệu sang 8-bit-storage-register
- OE ở mức thấp thì đầu ra sẽ có trạng thái giống 8-bit-storage-register

# Ma trận led dùng IC 74HC595



- PortB đầy dữ liệu theo hàng
- Dữ liệu cột được đẩy ra thông qua đầu ra của các IC 74HC595
- RC0, RC1, RC3 nối với các chân điều khiển SH<sub>CP</sub>, DS và ST<sub>CP</sub>
- Đầu ra Q7' của IC 74HC595 thứ nhất nối với DS' của IC 74HC595 thứ 2

# Ma trận led dùng IC 74HC595

```
#include <16f877a.h>                                RC0=SH_CP  
#use delay(clock=20000000)                            RC1=DS  
#BIT PORTCbits_RC0 = 0X07.0                          RC3=ST_CP  
#BIT PORTCbits_RC1 = 0X07.1  
#BIT PORTCbits_RC3 = 0X07.3  
const unsigned char font[] =  
{  
    127, 127, 14, 28, 14, 127, 127, 0,      // 'M'  
    124, 126, 19, 19, 126, 124, 0, 0,      // 'A'  
    3, 65, 127, 127, 65, 3, 0, 0,          // 'T'  
    65, 127, 127, 9, 25, 127, 102, 0,     // 'R'  
    0, 65, 127, 127, 65, 0, 0, 0,          // 'I'  
    7, 103, 60, 24, 60, 103, 67, 0,      // 'X'  
    0, 0, 0, 0, 0, 0, 0, 0,                // '  
    54, 127, 73, 73, 127, 54, 0, 0,      // '8'  
    68, 108, 56, 16, 56, 108, 68, 0,     // 'x'  
    64, 66, 127, 127, 64, 64, 0, 0,      // '1'  
    60, 126, 75, 73, 121, 48, 0, 0,      // '6'  
    0, 0, 0, 0, 0, 0, 0, 0,                // '  
    0, 0, 0, 0, 0, 0, 0, 0,                // '  
};
```

```
void main() {  
    Ma trận led dùng IC 74HC595  
    unsigned rol, delay, col;  
  
    OUTPUT_B(0x00);  
    OUTPUT_C(0x00);  
    while(1)  
    {  
        for(rol=0;rol<112;rol++) // Số lượng cột trong đoạn chương trình là 8 led x 13 kí tự = 112.  
        {  
            for(delay=0;delay<20;delay++)  
            {  
                PORTCbits_RC0 = 1; PORTCbits_RC0 = 0;  
                PORTCbits_RC1 = 1;  
                PORTCbits_RC3 = 1; PORTCbits_RC3 = 0;  
                for(col=0;col<16;col++) //  
                {  
                    OUTPUT_B (~ font[col + rol]);  
                    delay_us(300);  
                    PORTCbits_RC0 = 1; PORTCbits_RC0 = 0; PORTCbits_RC1 = 0; PORTCbits_RC3 = 1;  
                    PORTCbits_RC3 = 0;  
                }  
            }  
        }  
    }  
}
```

RC0=SH\_CP

RC1=DS

RC3=ST\_CP

Bắt đầu quét từ hàng dữ liệu i

Cấp nguồn cho cột của ma trận led

Từ hàng i đọc 16 hàng dữ liệu liên tiếp và đưa ra công B

↑  
Tắt nguồn cột