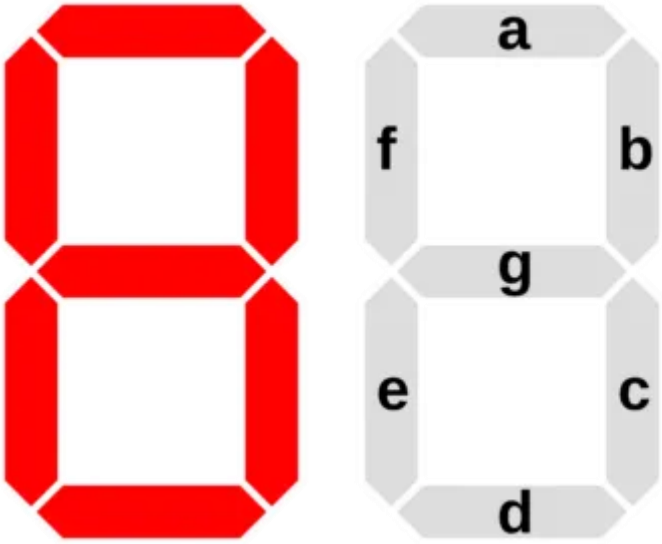
Display & HMI

7-Segment Display

Led 7 đoạn là một thiết bị điện tử đơn giản bao gồm các thanh 7 đèn LED được sắp xếp theo cách đại diện cho số thập phân [0 - 9] nó cũng có thể được sử dụng để hiển thị các chữ số thập lục phân [0 - F]. Tuy nhiên, các thiết bị này thực sự có đèn LED thứ tám đại diện cho dấu thập phân.

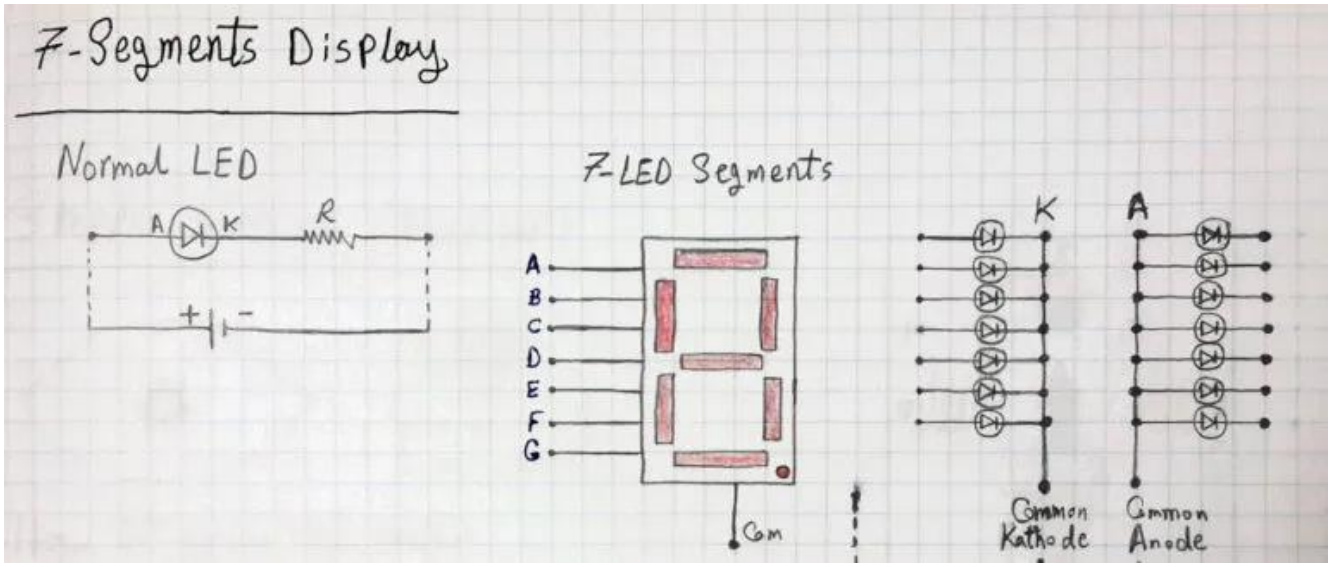


Màn hình 7 đoạn đang được sử dụng cho các tác vụ hiển thị đơn giản trong nhiều ứng dụng (ví dụ: máy sưởi, nồi hơi, đầu DVD, đầu phát âm thanh, thiết bị đeo được, khóa kỹ thuật số, v.v.).

Học cách giao diện với màn hình 7 phân đoạn sẽ giúp bạn tạo các ứng dụng thân thiện với người dùng. Hãy tưởng tượng rằng bạn đang tạo ra Máy **X có** thể là một nồi hơi chẳng hạn hoặc bất cứ thứ gì. Thiết bị của bạn phải hiển thị cho người dùng giá trị nhiệt độ hiện tại của nó. Mặc dù bạn có thể ghi giá trị của nhiệt độ (nhị phân) vào cổng đầu ra và móc một số đèn LED để hiển thị con số này. Chà, bạn có thể tưởng tượng điều này có thể khủng khiếp như thế nào không?

7 phân đoạn thực sự hoạt động như thế nào?

Có hai loại 7 phân đoạn cho mỗi mô hình là "Cực âm chung - Common Cathode" và "Cực dương chung - Common Anode". Điều này đề cập đến quy ước phân cực kết nối bên trong của các thanh LED. Sự khác biệt được thể hiện trong hình dưới đây.

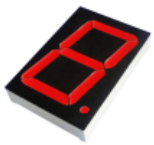


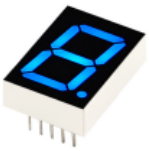
Như có thể nhận thấy, các thiết bị này về cơ bản là một số đèn LED được nối với nhau trong các gói DIP. Do đó, sự dễ dàng trong việc vận hành / giao diện 7 đoạn. Nó không nhiều hơn một số đèn LED có thể được nối với cổng i / o thông qua điện trở 330Ω và dễ dàng điều khiển. Tuy nhiên, bạn nên chú ý đến một vài ys sau đây

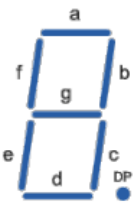
- Cathode 7 phân đoạn chung: Dây dẫn catốt của tất cả các đèn LED được kết nối với nhau bên trong. Vì vậy, bạn sẽ phải kết nối dây âm này với Gd (0v) và cực dương dẫn đến cổng i / o. Do đó, việc kích hoạt (Chuyển mạch) của đèn LED sẽ đạt được bằng cách **đặt** (ghi 1) các chân của cổng tương ứng (**Active High**).

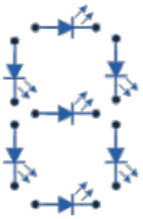
- Anode 7 phân đoạn chung: Dây dẫn cực dương của tất cả các đèn LED được kết nối với nhau bên trong. Vì vậy, bạn sẽ phải kết nối cực dương này với Vdd (+ 5v) và cực âm dẫn đến cổng i / o. Do đó, việc kích hoạt (Chuyển mạch) của đèn LED sẽ đạt được bằng cách **xóa** (ghi 0) các chân của cổng tương ứng (**Hoạt động thấp**).

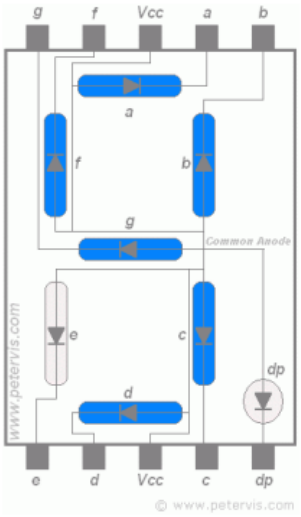
Led 7 thanh

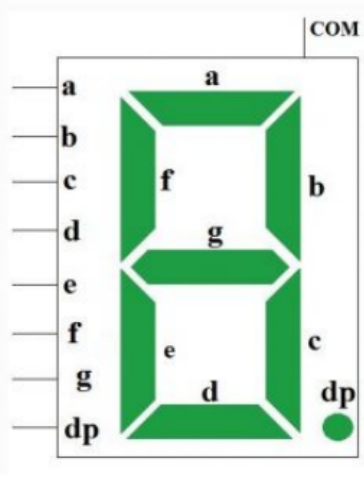






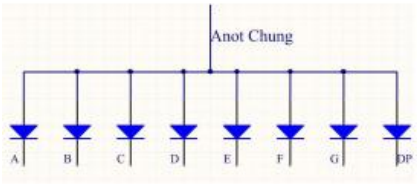




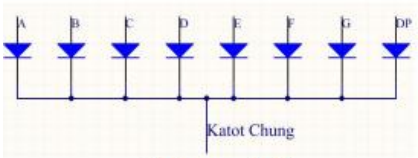


Các led đơn sắp đặt theo hình số 8 để hiển thị số

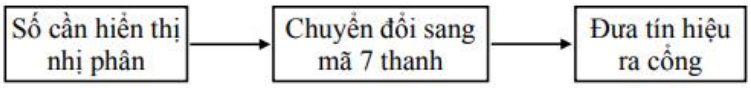
Led 7 thanh



0 sáng. 1 tối



1 sáng. 0 tối

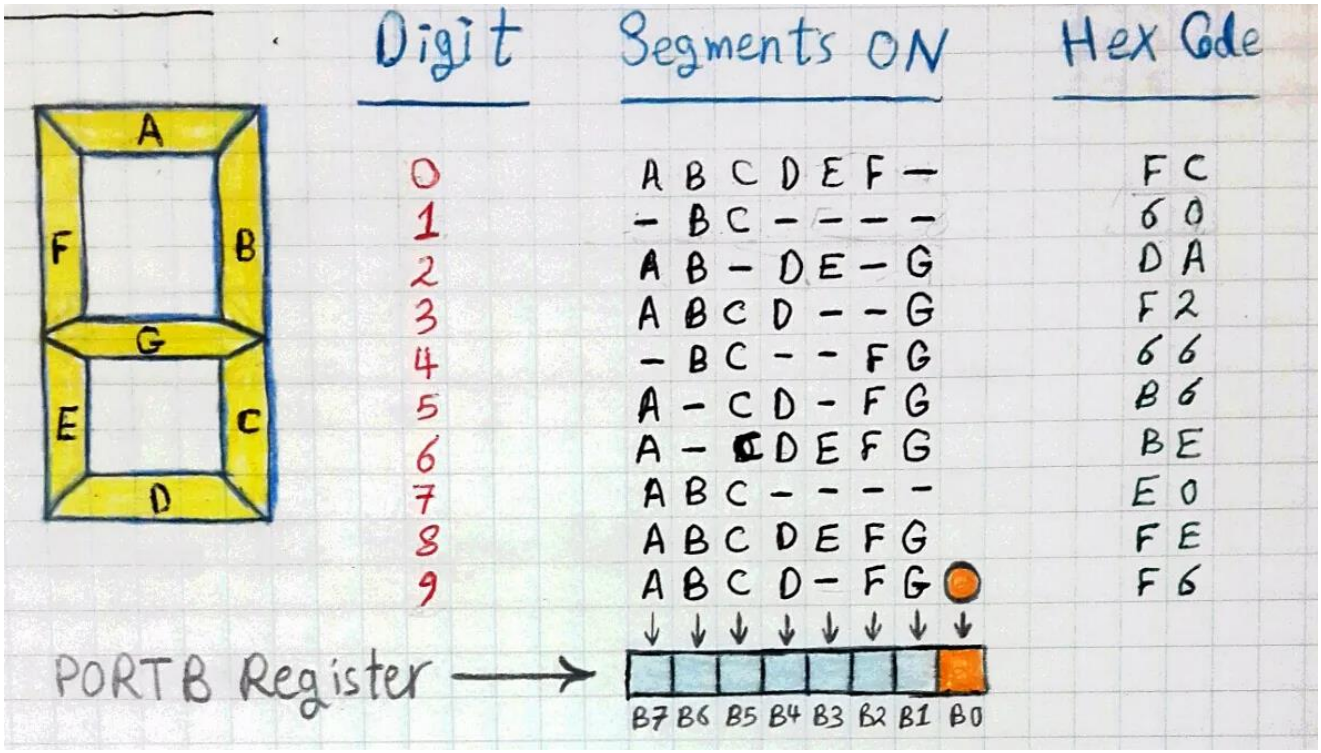


Numbers	Common Cathode		Common Anode	
	(DP)GFEDCBA	HEX Code	(DP)GFEDCBA	HEX Code
0	00111111	0x3F	11000000	0xC0
1	00000110	0x06	11111001	0xF9
2	01011011	0x5B	10100100	0xA4
3	01001111	0x4F	10110000	0xB0
4	01100110	0x66	10011001	0x99
5	01101101	0x6D	10010010	0x92
6	011111101	0x7D	10000010	0x82
7	00000111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

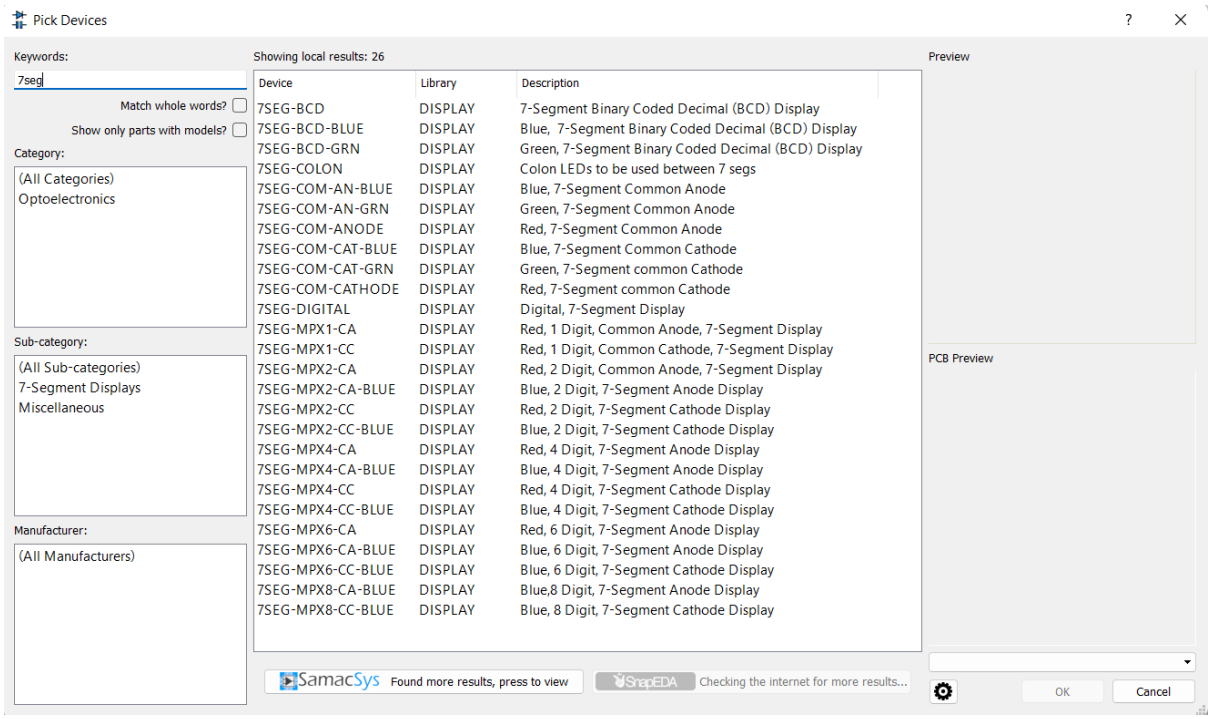
Hiển thị số nào cần đưa mã 7 thanh tương ứng vào led

CC

0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F



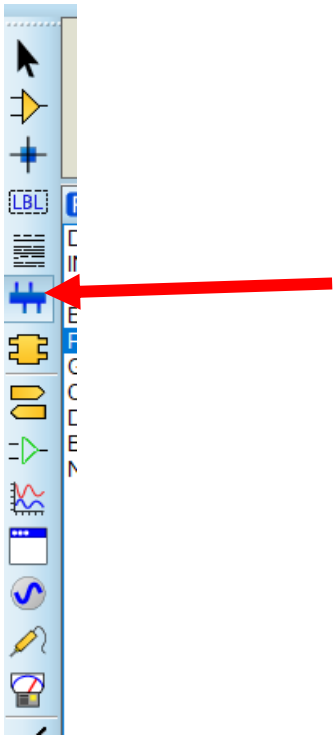
Để lấy được led 7 thanh trong proteus ta gõ “7seg” sẽ ra được hình sau



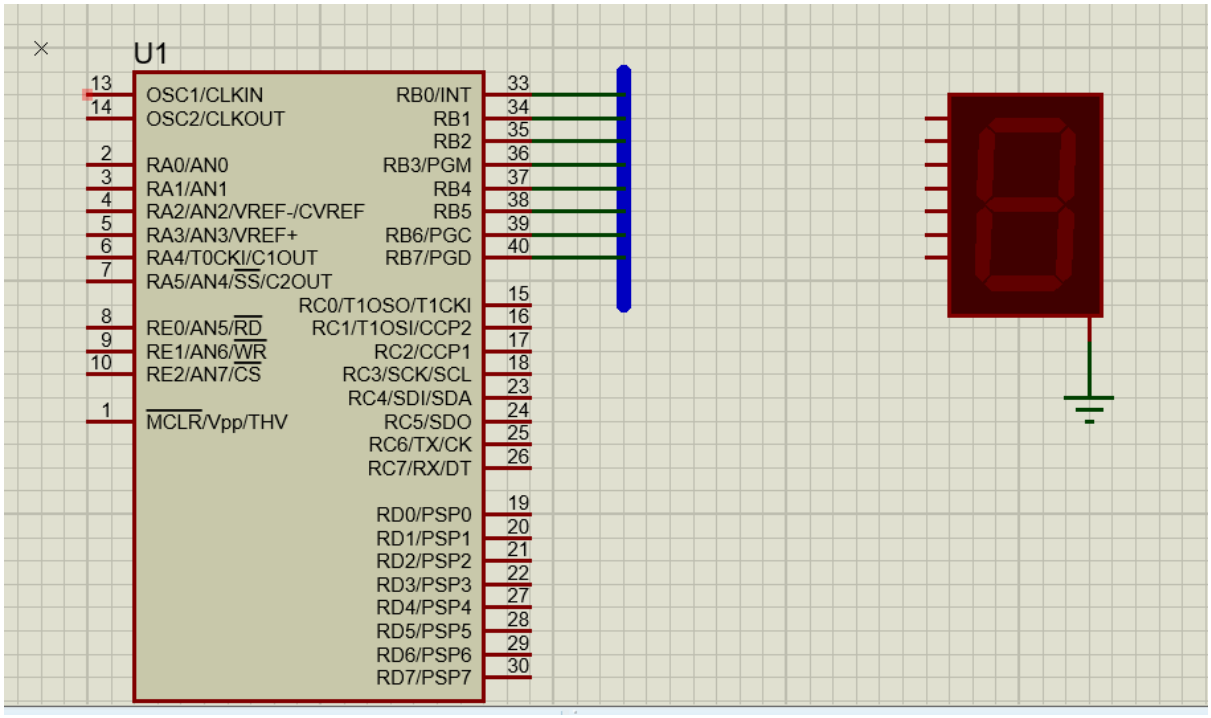
Ở đây kí hiệu CC là “Chân katot chung” và ngược lại CA “Chân anot chung “.

Ví dụ với led CC.

Để cho bài hình mô phỏng đẹp hơn ta dùng thanh bus

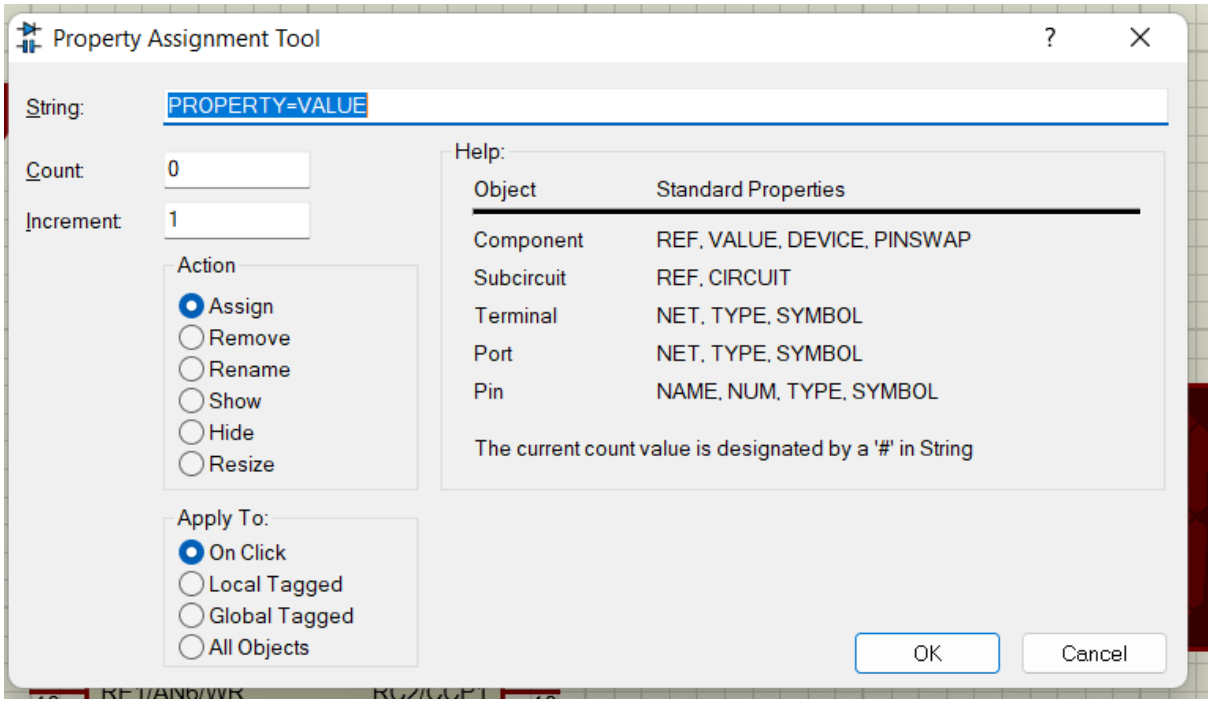


Sau khi click vào đây. Ta ấn chuột trái 1 lần rồi kéo đường bus theo hướng mình muốn. Tiếp đó ta nối các chân của PIC vào đường bus

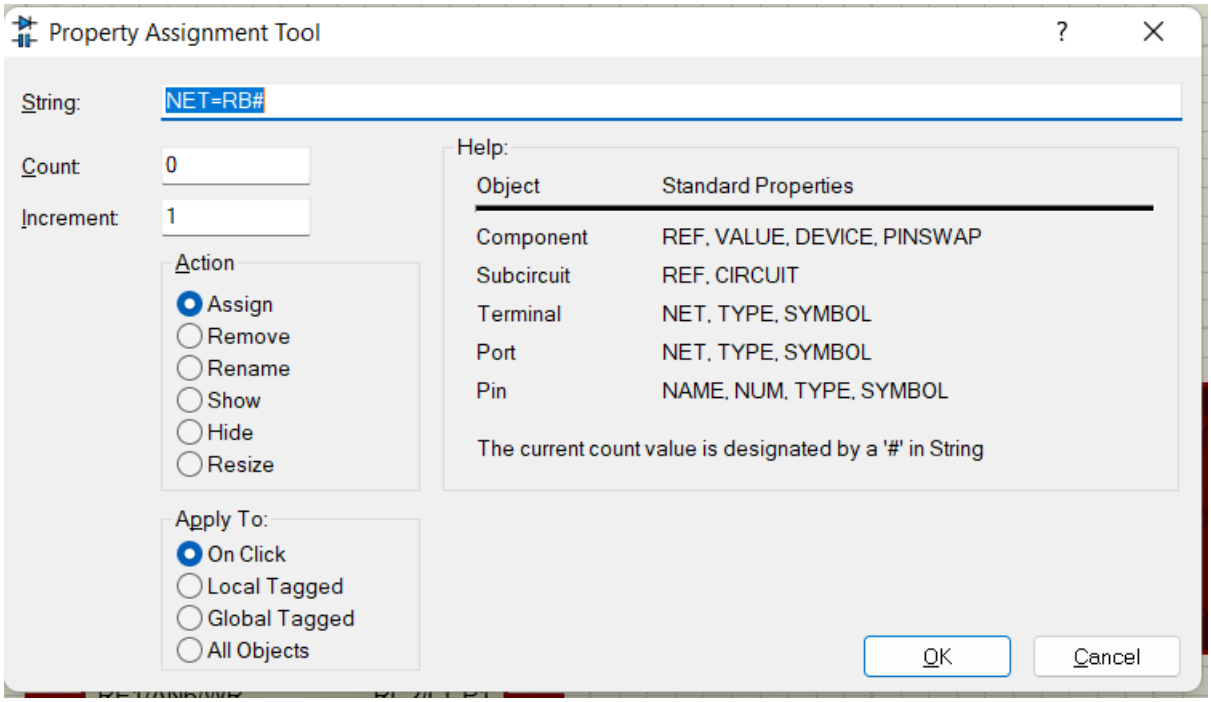


Bước 2: Đặt tên.

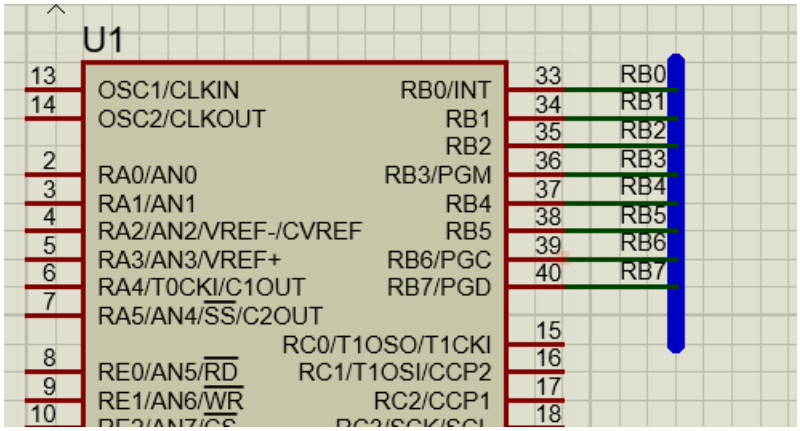
Ở trong chương trình proteus ta ấn phím A.



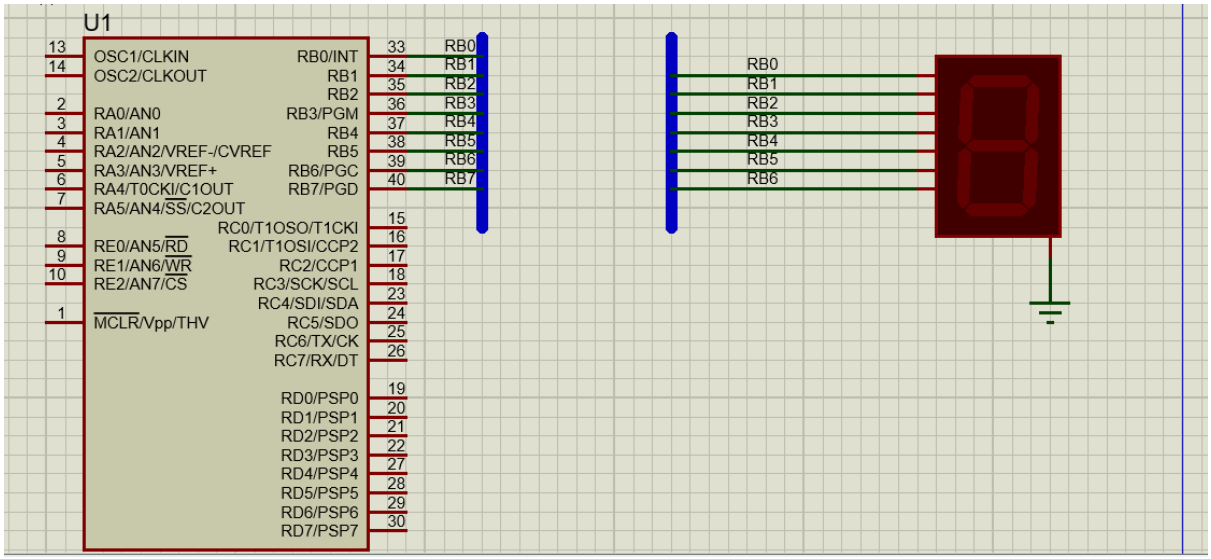
Sau đó đổi PROPERTY thành NET = X# với X là tên mong muốn.



Sau đó ấn vào ok và đặt tên cho chân



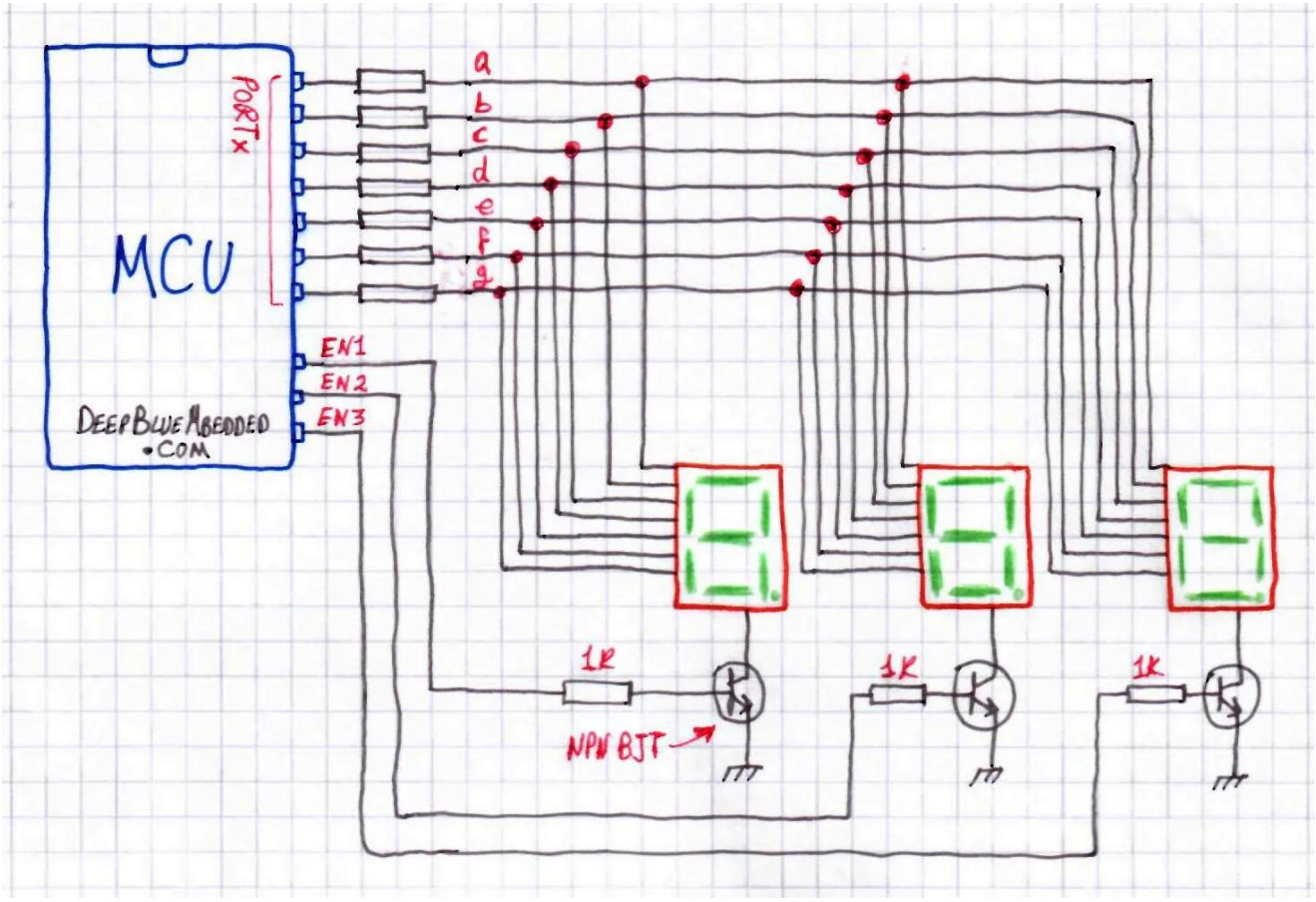
Tương tự như vậy ta có bên 7 đoạn



Code

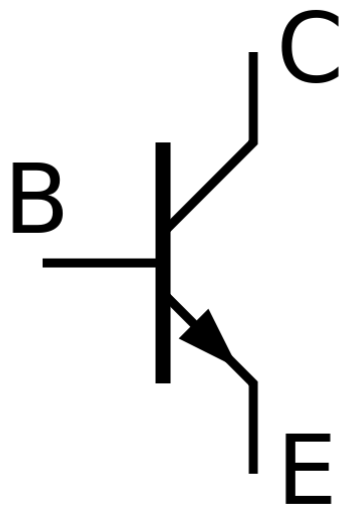
```
void main(void) {
  unsigned char segments_code[10] =
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
  unsigned char counter=0;
  TRISB = 0x00; // Set All Pins To Be Output Pins
  PORTB = 0x00; // Initially Clear All The 8-Pins
  while(1)
  {
    PORTB = segments_code[counter];
    __delay_ms(500);
    counter++;
    if(counter==10)
      counter=0;
  }
  return;
}
```

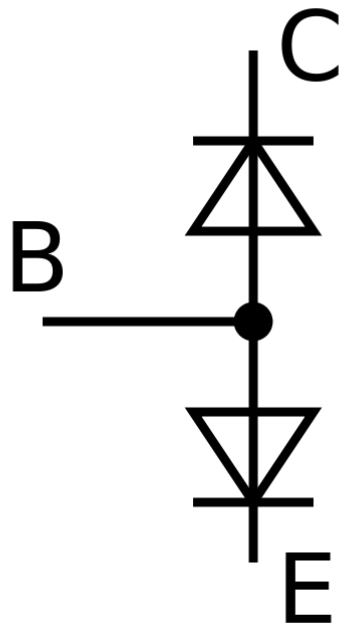
Giao diện hiển thị 7 phân đoạn



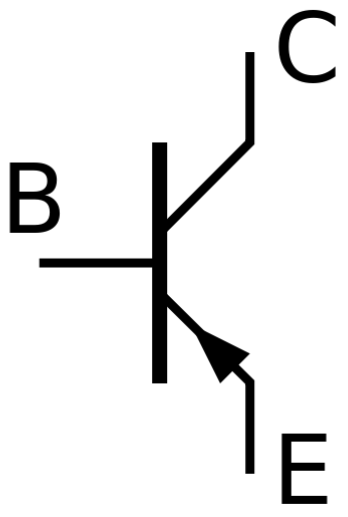
Để sáng được nhiều led 7 thanh ta sẽ cho những led này sáng từng led một và điều khiển điều kiện này ta dùng bộ NPN và PNP (BJT) .

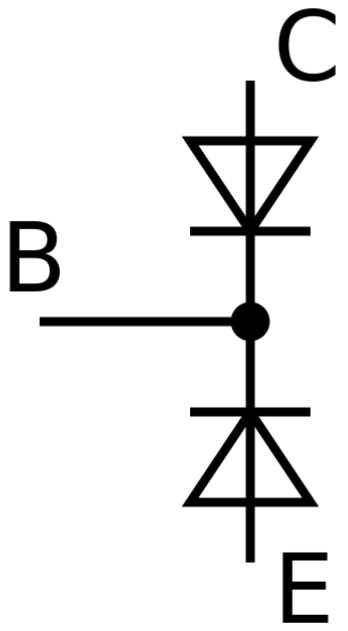
NPN





PNP





16×2 Giao diện LCD

LCD

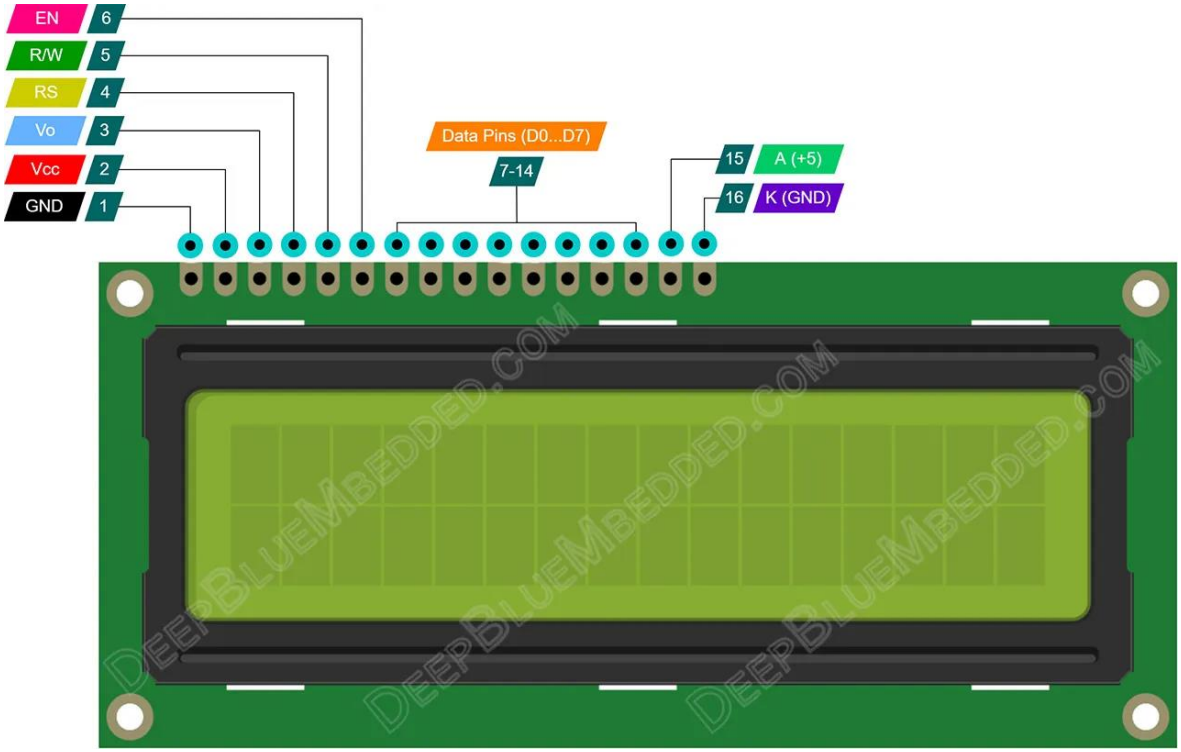
Loại 1 hàng ký tự

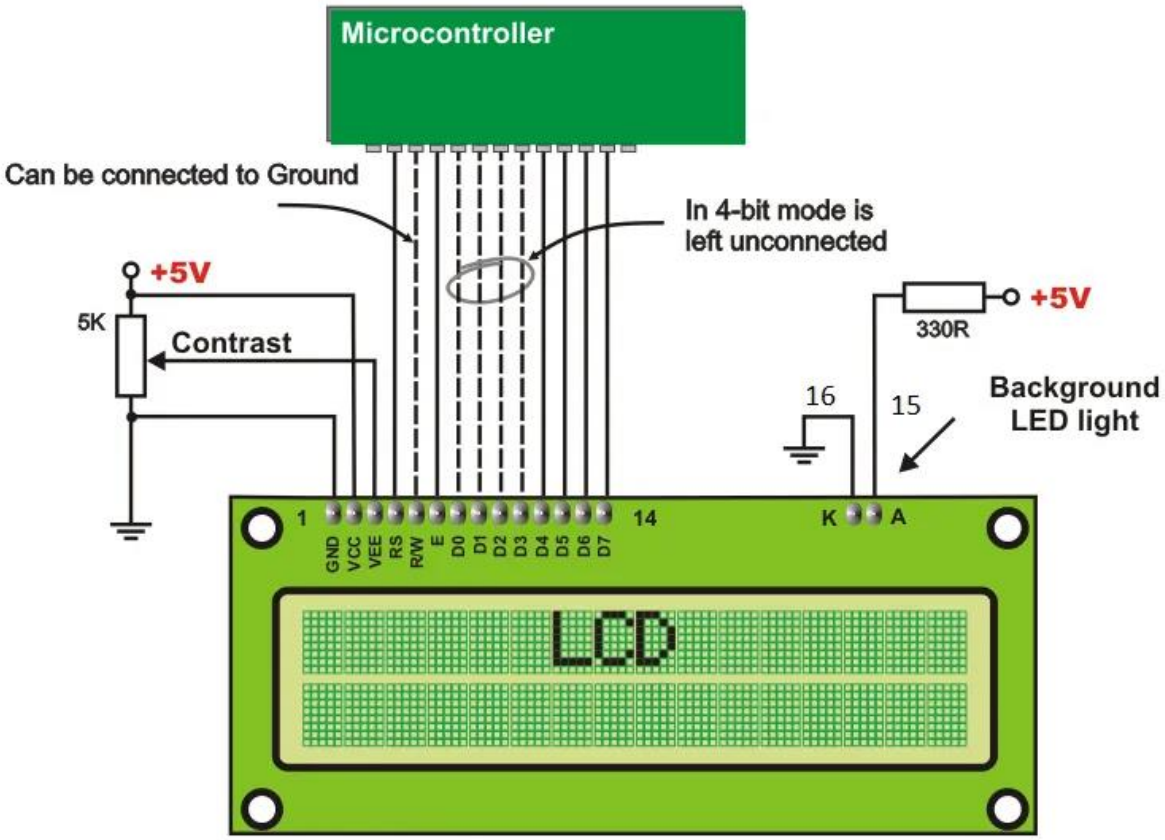
Loại 4 hàng ký tự

Loại 2 hàng ký tự

LCD

Chân	Ký hiệu	Mô tả
1	Vss	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển
2	V_{DD}	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với V_{CC}=5V của mạch điều khiển
3	V_{EE}	Điều chỉnh độ tương phản 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 : + Chế độ 8 bit : Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. + Chế độ 4 bit : 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


EN 6
R/W 5
RS 4
Vo 3
Vcc 2
GND 1
Data Pins (D0...D7)
7-14
15 A (+5)
16 K (GND)
DeepBlueMbedded.com


Microcontroller
Can be connected to Ground
In 4-bit mode is left unconnected
+5V
5K
Contrast
330R
+5V
16
15
Background LED light
1
14
K
A
GND
VCC
VEE
RS
R/W
E
D0
D1
D2
D3
D4
D5
D6
D7
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 $D_7 \ldots D_0$

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, fonr 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 $D_7 \ldots D_0$

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 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 $D_3 \ldots D_0$

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 $D_3 \ldots D_0$

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 $D_3 \ldots D_0$

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

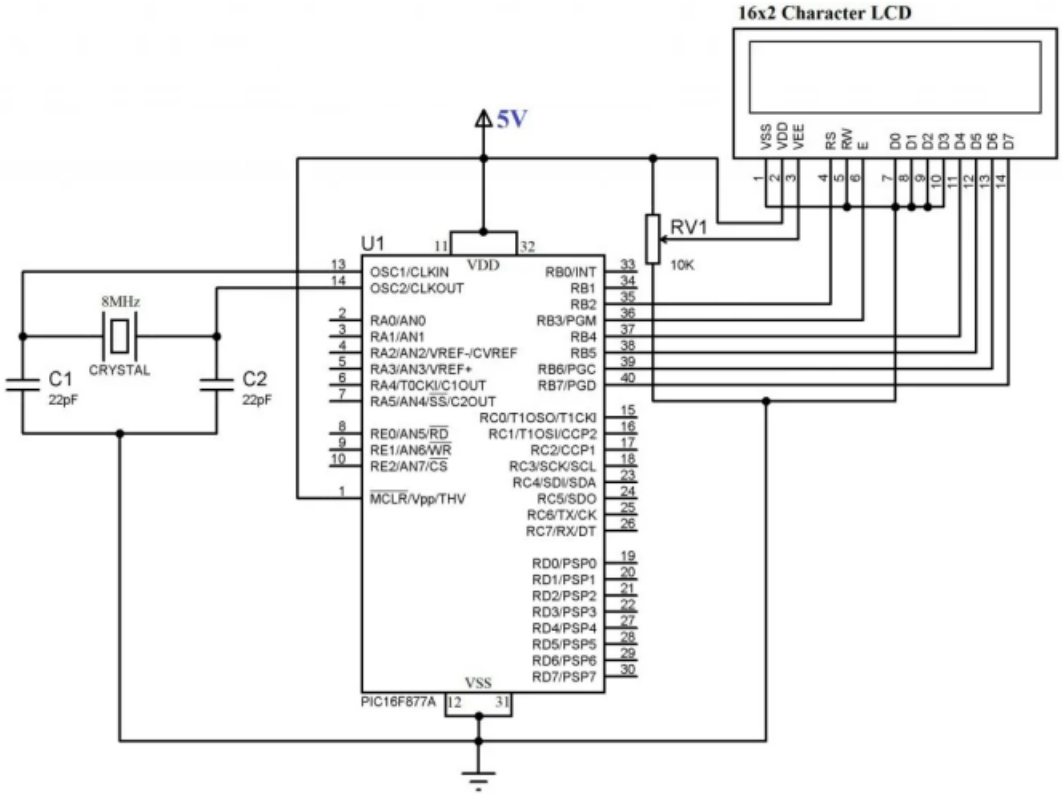
Bước 5: Tạo trễ để LCD thực hiện xong lệnh cho 4 bit cao

Bước 6: Gửi 4 bit thấp của dữ liệu ra $D_3 \ldots D_0$

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

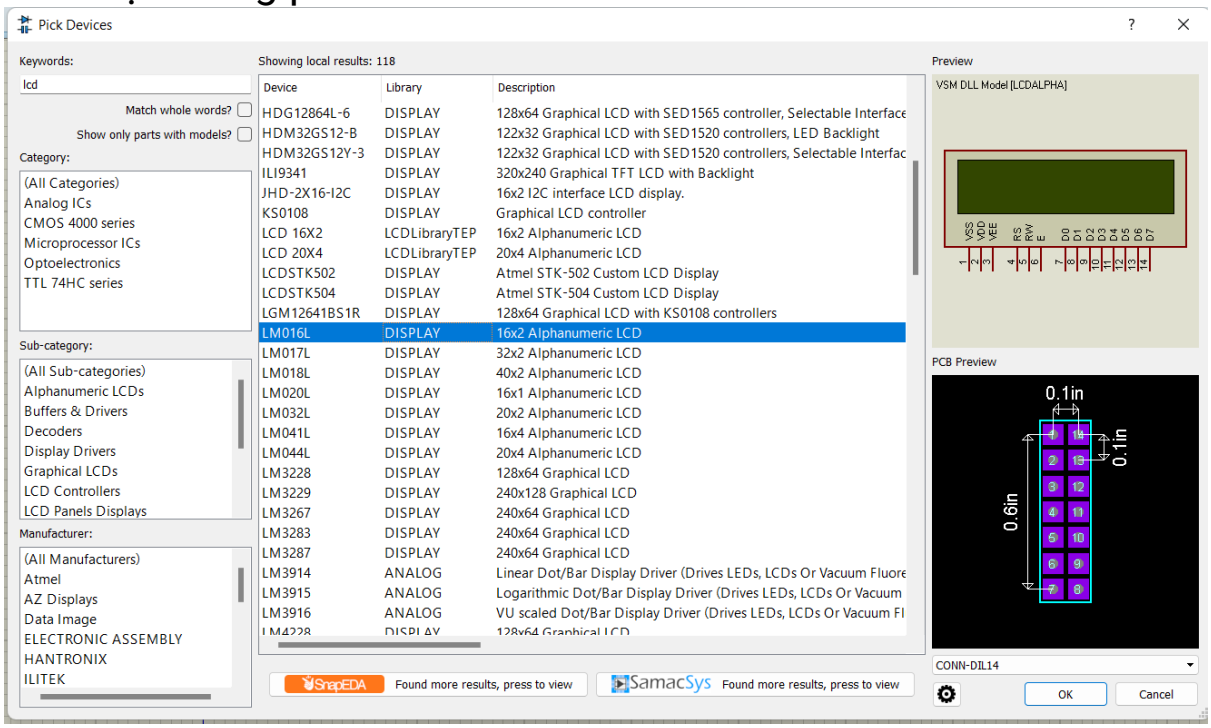
Bước 8: Tạo trễ để LCD thực hiện xong lệnh cho 4 bit thấp

LCD với giao tiếp 4 bit

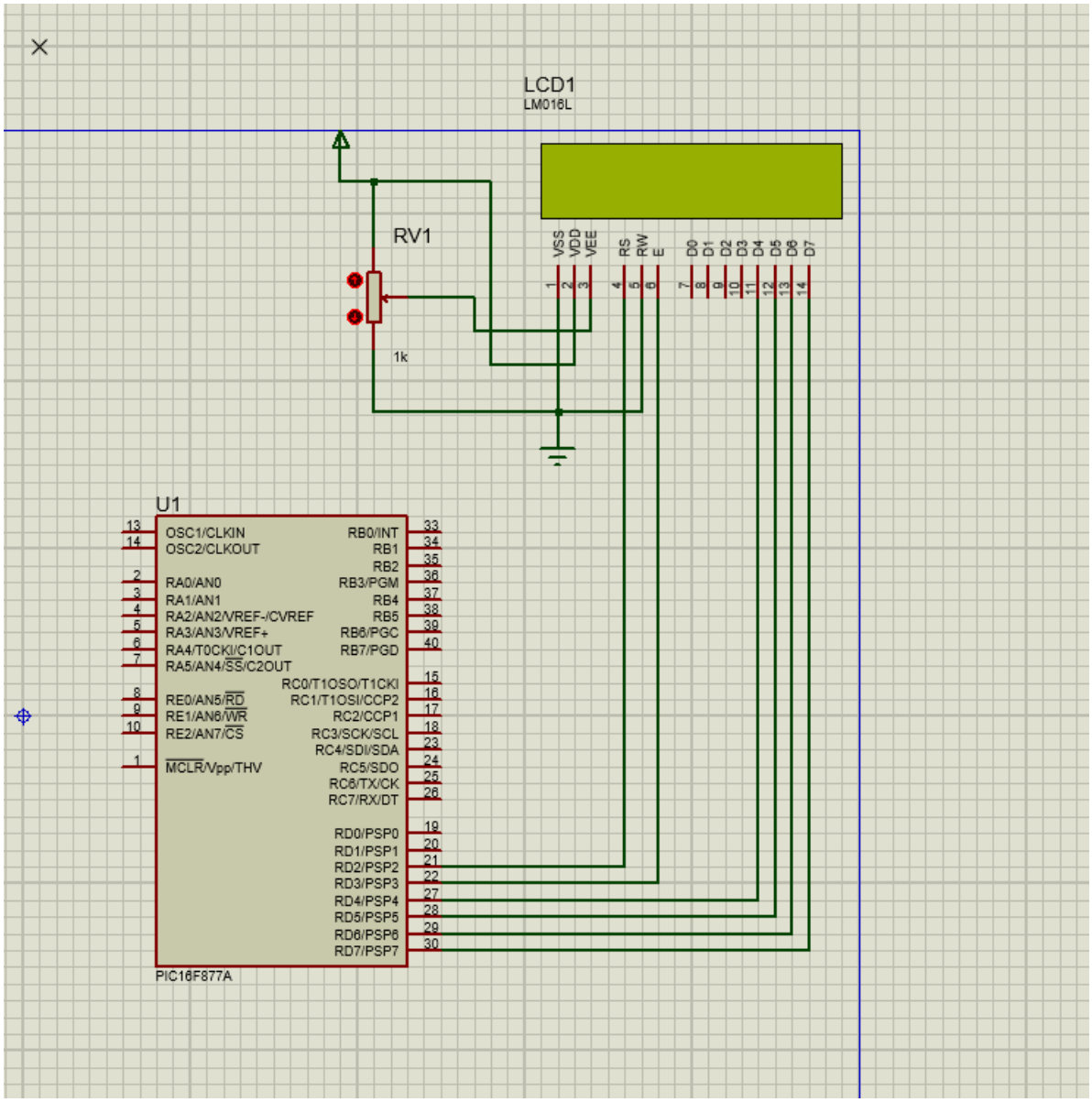


Để cấu hình LCD.

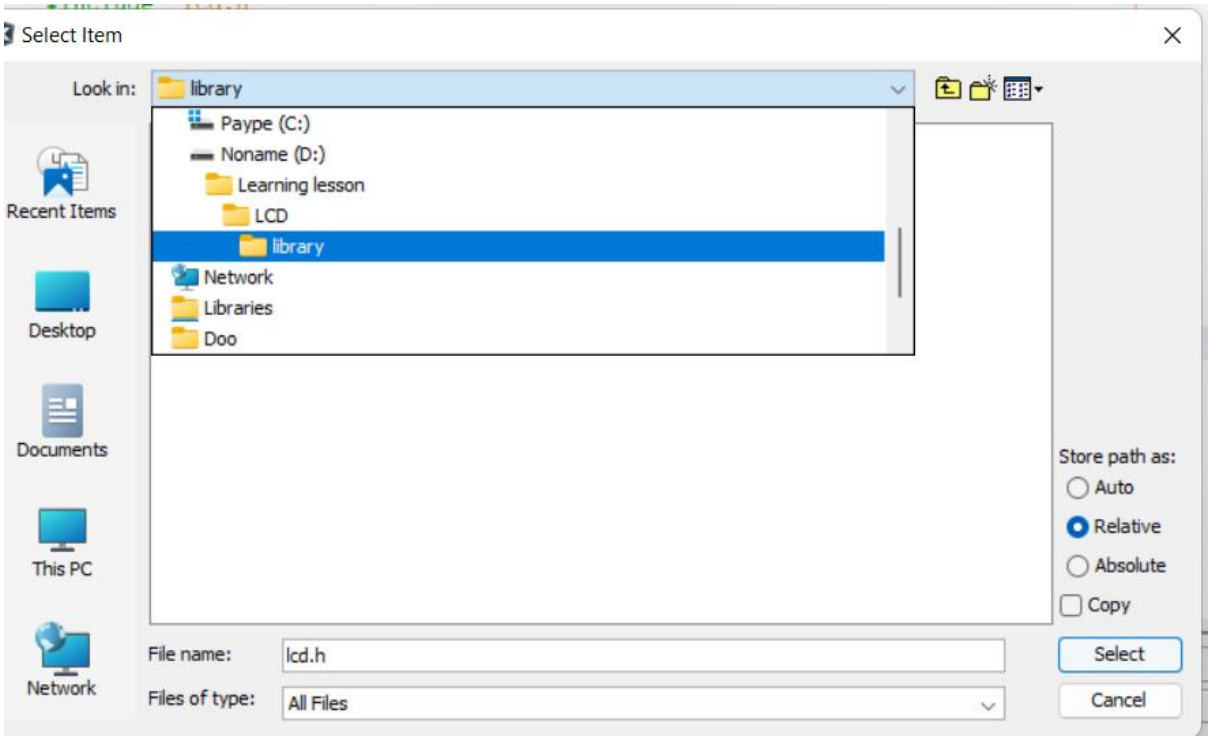
Ta chọn trong proteus



Hoàn thành sơ đồ chân như hình trên.

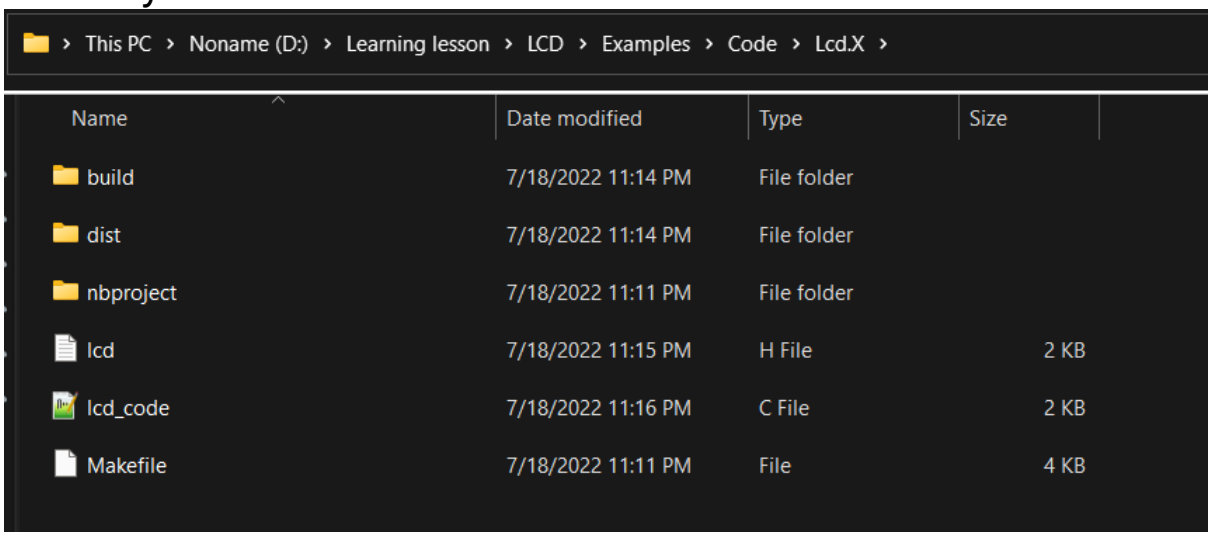

LCD1
LM016L
RV1
1k
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
U1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
PIC16F877A

Chỉ đến mục library.

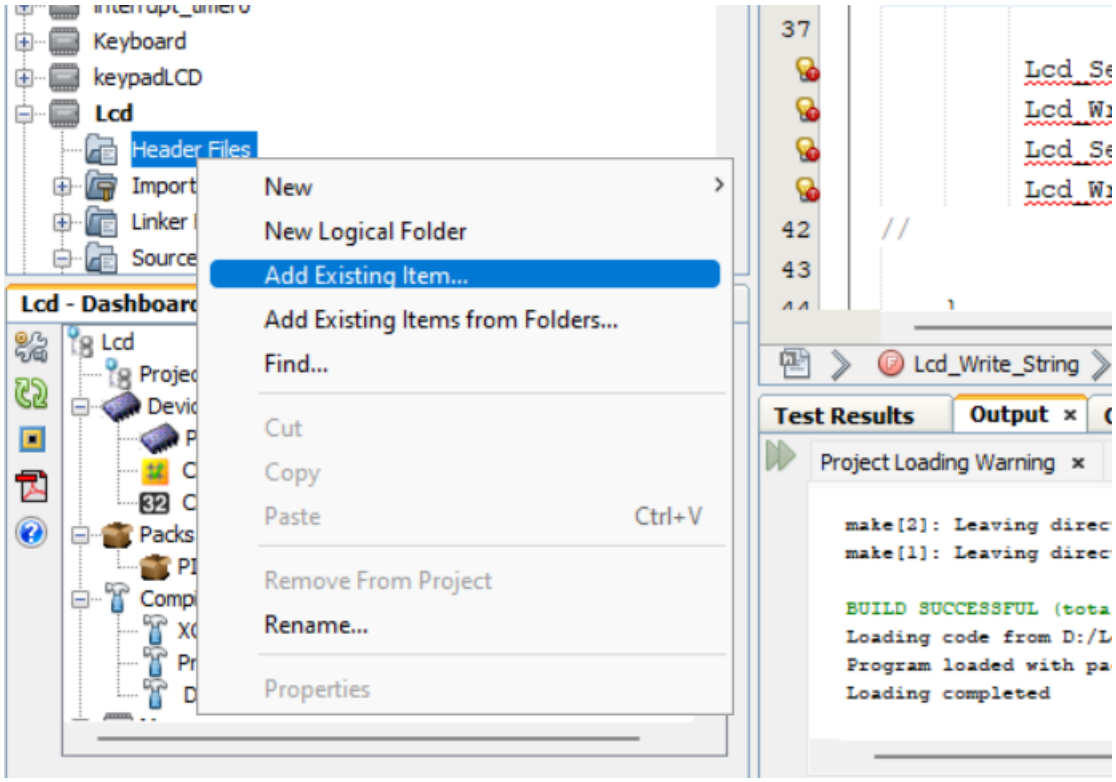


Copy lcd.h vào trong project bạn muốn làm

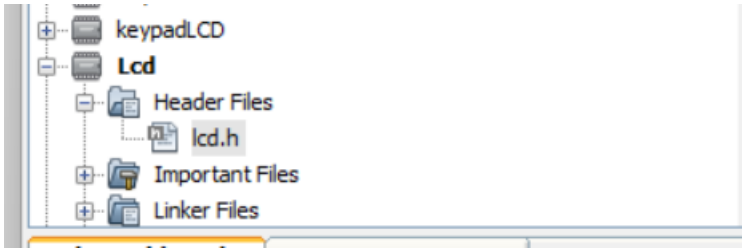
Như này :



Vào Mplab chọn vào header files và Add Existing Iteam...



Chọn vào lcd.h



Sau đó build chương trình

Một số hàm đặc biệt

Lcd_Init(); // cấu hình cho lcd - bắt buộc phải có.

Lcd_Clear(); // xóa bộ nhớ của lcd.

Lcd_Write_Char(char a); // viết một kí tự lên lcd.

Lcd_Write_String(char *a); // viết một chuỗi lên lcd

Lcd_Set_Cursor(char a, char b); // đưa con trỏ với vị trí hàng a cột b.

Vì LCD không thể ghi trực tiếp giá trị như int , float lên màn hình được. Mà ta phải đưa nó về dạng char để đưa lên lcd.

Khai báo hàm sprintf() trong C

Dưới đây là phần khai báo cho hàm sprintf() trong C:

```
int sprintf(char *str, const char *format, ...)
```

Tham số

str − Đây là con trỏ tới một mảng các phần tử char nơi chuỗi kết quả được lưu trữ.

format − Đây là chuỗi chứa text để được ghi tới buffer. Nó có thể tùy ý chứa các thẻ định dạng có thể được nhúng mà được thay thế bởi các giá trị được xác định trong các tham số bổ sung tiếp theo và được định dạng theo yêu cầu. Nguyên mẫu các thẻ định dạng là **%[flags][width][.precision][length]specifier**, được giải thích như dưới đây:

specifier	Kết quả
c	Ký tự
d hoặc i	Số nguyên hệ thập phân có dấu
e	Ký hiệu khoa học (mantissa/exponent) sử dụng ký tự e
E	Ký hiệu khoa học (mantissa/exponent) sử dụng ký tự E
f	Số thực dấu chấm động hệ thập phân

g	Sử dụng rút gọn của %e hoặc %f
G	Sử dụng rút gọn của %E hoặc %f
o	Số bát phân có dấu
s	Chuỗi ký tự
u	Số nguyên hệ thập phân không dấu
x	Số nguyên hệ thập lục phân không dấu
X	Số nguyên hệ thập lục phân không dấu (các chữ cái hoa)
p	Địa chỉ con trỏ
n	Không in cái gì
%	Ký tự

Ví dụ

```
sprintf(s,"%.2d",(value_eeprom));

Lcd_Write_String(s);
```

ở đây ta gán giá trị của biến Value_eeprom cho biến s với kiểu mảng char với định dạng là làm tròn số thập phân thứ 2