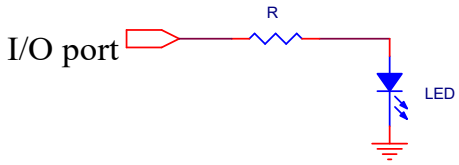
Chương 5. THIẾT KẾ MỘT SỐ GHÉP PHỐI CƠ BẢN VỚI HỆ VI ĐIỀU KHIỂN

Trong chương 5 trình bài một số ví dụ xây dựng chương trình cho một số ứng dụng cơ bản như điều khiển cổng vào ra như Led đơn, Led 7 thanh, LCD, phím bấm, đọc giá trị điện áp tương tự, truyền thông UART.

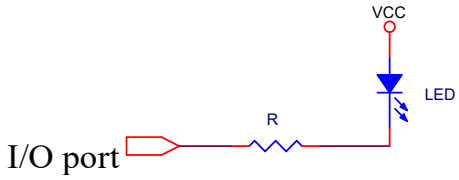
I. Ghép nối LED đơn

Đây là ghép nối đơn giản nhất. Các LED đơn có thể sáng rõ với một vài mA dòng điện cung cấp. Có 02 cách cung cấp dòng phân cực thuận cho LED đơn:

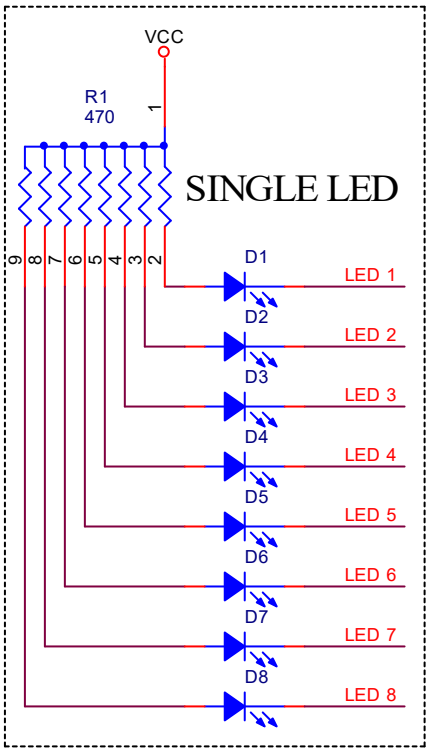
Cách thứ nhất: bơm dòng



Cách thứ hai: Nuốt dòng



Trên mạch các LED đơn được nối với cổng vi xử lý thành vòng tròn qua 1 điện trở băng

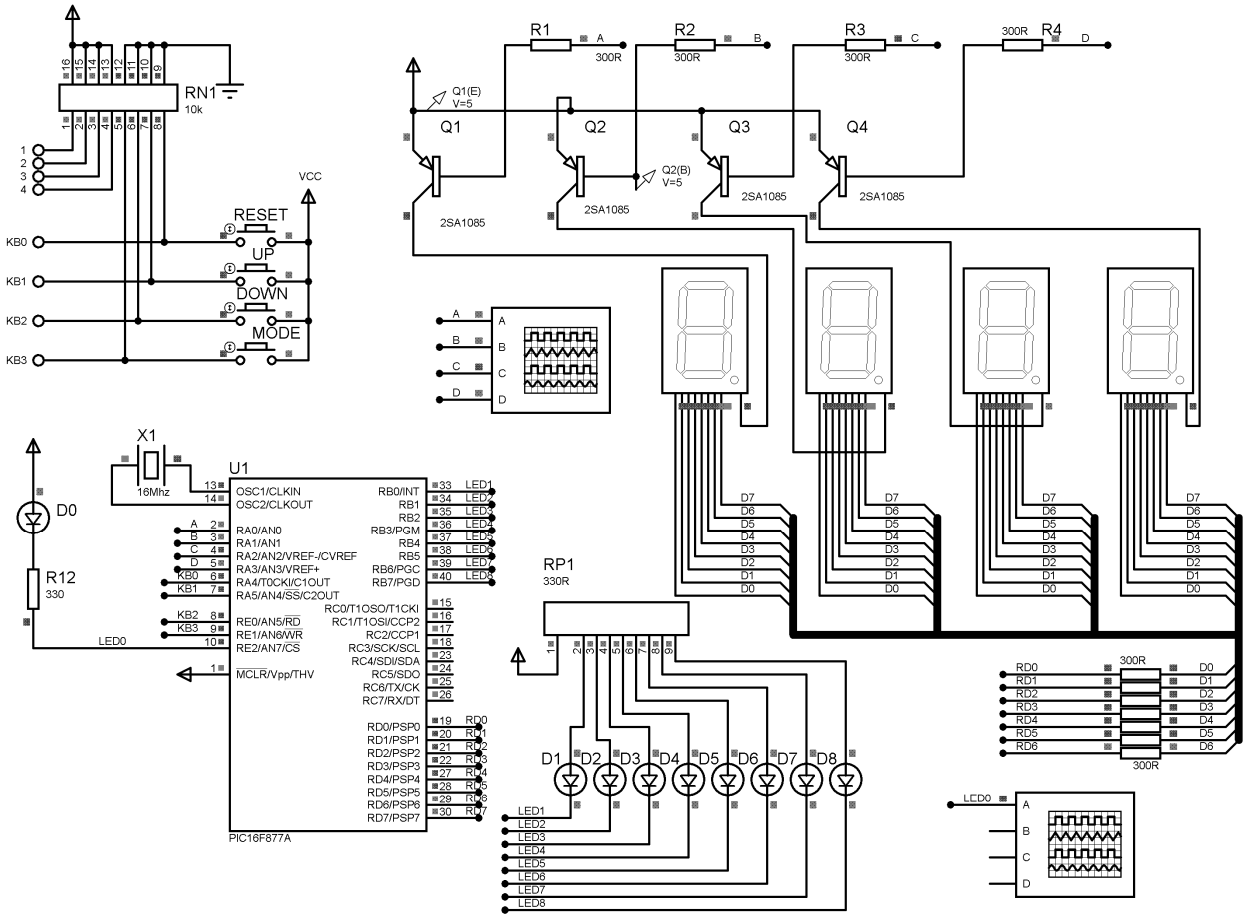


Hình 5.1 Ghép nối Led đơn

Trong cách ghép nối thứ nhất, để LED phát sáng cần đưa ra giá trị logic 1 (ứng với điện áp +5V) để phân cực thuận cho LED. Dòng điện I sẽ chảy từ cổng vi xử lý qua điện trở hạn dòng, qua LED về đất.

Trong cách ghép nối thứ hai, để LED phát sáng cần đưa ra giá trị logic 0 (ứng với điện áp đưa ra là 0V). Dòng điện lúc này sẽ chảy từ dương nguồn qua điện trở hạn dòng, qua LED vào chân vi xử lý.

a) **Bài tập 1:** Viết chương trình điều khiển LED (D0) đơn nháy tần số 10Hz (chu kỳ 100ms).



Hình 5.2 Sơ đồ vi xử lý điều khiển Led đơn, Led 7 thanh và phím bấm

Để tạo ra chu kỳ nháy LED có thể sử dụng 2 cách:

- Sử dụng một hàm trễ thời gian: *__delay_ms(100);*

Hàm này tạo thời gian bằng cách đếm số lượng mã lệnh thực hiện, do chu kỳ thực hiện lệnh đã biết trước. Ví dụ để tạo ra khoảng thời gian 100ms với vi xử lý sử dụng thạch anh 16Mhz (chu kỳ thực hiện 1 mã lệnh là 0.25µs), hàm trên cần thực hiện 400,000 mã lệnh. Với một số trình biên dịch (như XC8) hàm này có sẵn trong thư viện, trong đó giá trị truyền biến chính là số ms tạo khoảng trễ.

```
#define _XTAL_FREQ    16000000
#include <xc.h>

#define Led PORTEbits.RE2
#define Tris_Led TRISEbits.TRISE2
```

```
int main ()
{
 ADCON1 = 0x06; // Các cổng AIN là các cổng số
 Tris_Led=0;  // Cổng Led E2 là cổng ra
while(1)
 {
  Led=0;
 __delay_ms(100);   // Thời gian thực hiện mất 100ms
  Led=1;
  __delay_ms(100);
}
}
```

- Sử dụng bộ định thời (Timer).

```
#include <xc.h>
#define Led PORTEbits.RE2
#define Tris_Led TRISEbits.TRISE2
unsigned int count_t0;

 void timer0_initialize ()
{
   T0CS=0; // Chọn CLK0=FOSC/4 =4Mhz T=0.25us
   PSA=0; //  Có sử dụng hệ số chia
   PS0=1;
   PS1=1;
   PS2=0;  //Hệ số chia  011=1:16 0.25us*16=4us
   TMR0=5;      //250 xung tràn
   TMR0IE=1;
   GIE=1;
}

 void __interrupt()  Count_Up(void)
{
  if (TMR0IE && TMR0IF)
  {
     TMR0=5;
     TMR0IF=0;
     count_t0++;

     if(count_t0==100)  //100ms
        {
          count_t0=0;
        Led=~Led;
      }
  }
```

```
}

int main ()
{
 ADCON1 = 0x06;   // Các cổng AIN là các cổng số
 Tris_Led=0;          // Cổng Led E2 là cổng ra
timer0_initialize ()  ;
while(1)
  {
  }
}
```

b) Bài tập 2: Điều khiển 4 Led lần lượt sáng với nhịp 1 giây, sau đó tắt hết trong 2 giây và lặp lại.

- Giải pháp 1 là dùng hàm trễ tạo thời gian, điều khiển các Led trong chu kỳ quét chính.

```
#define _XTAL_FREQ    16000000
#include <xc.h>
    #define Led1  PORTBbits.RB0
    #define Led2  PORTBbits.RB1
   #define Led3  PORTBbits.RB2
   #define Led4  PORTBbits.RB3
int main ()
{
TRISB=0x00;   // Cổng PortB là cổng ra
PORTB=0xff;  // ban đầu các đèn Led đều tắt
while(1)
 {
  Led1=0;
 __delay_ms(1000);   // Thời gian thực hiện mất 1s
  Led2=0;
  __delay_ms(1000);
  Led3=0;
  __delay_ms(1000);
  Led4=0;
  __delay_ms(1000);
PORTB|=0x0f;  // Tắt 4 Led
  __delay_ms(2000);
}
}
```

- Giải pháp dùng bộ định thời

```
#include <xc.h>
    #define Led1  PORTBbits.RB0
    #define Led2  PORTBbits.RB1
   #define Led3  PORTBbits.RB2
   #define Led4  PORTBbits.RB3
   unsigned int count_t0,buoc;
 void timer0_initialize ()
```

```
{
   T0CS=0; // Chọn CLK0=FOSC/4 =4Mhz T=0.25us
   PSA=0; //  Có sử dụng hệ số chia
   PS0=1;
   PS1=1;
   PS2=0;       //Hệ số chia  011=1:16  0.25us*16=4us
   TMR0=5;      //250 xung sẽ tràn
   TMR0IE=1; // cho pháp ngắt Timer 0
   GIE=1;       // cho phép ngắt toàn cục
}
 void __interrupt()  Count_Up(void)
{
  if (TMR0IE && TMR0IF)
  {
     TMR0=5;
     TMR0IF=0;
     count_t0++;
     if(count_t0==1000)  //1s
          {
           count_t0=0;
            buoc++;
            switch (buoc)
            {
             case 1:   Led1=0;
                        break;
            case 2:    Led2=0;
                       break;
            case 3:   Led3=0;
                       break;
            case 4:  Led4=0;
                     break;
            case 5:
           case 6:   PORTB|=0x0f;  // Tắt 4 Led
                     break;
            case 7:  buoc=0;
            }

        }
  }
}
int main ()
{
  TRISB=0x00;   // Cổng PortB là cổng ra
 PORTB=0xff;  // ban đầu các đèn Led đều tắt

timer0_initialize ()  ;
```

```
    while(1)
      {
      }
}
```

c) ***Bài tập 3:*** Sử dụng phím ấn KB1 nối vào cổng A5 làm sáng đèn Led ở cổng E2, khi ấn phím KB0 (cổng A4) đèn Led tắt.

```
#include <xc.h>

#define Led          PORTEbits.RE2
#define Tris_Led     TRISEbits.TRISE2
#define KB0          PORTAbits.RA4
#define Tris_KB0     TRISAits.TRISA4
#define KB1          PORTAbits.RA5
#define Tris_KB1     TRISAits.TRISA5

int main ()
{
   Tris_Led=0;    // Đặt E2 là cổng ra
  Tris_KB0=1;   //Đặt A4 là cổng vào
 Tris_KB1=1;   //Đặt A5 là cổng vào
 while(1)
  {
    if(KB0)                //Phím KB0 ấn
     Led=0;               // Đèn Led sáng
   if(KB1)               // Phím KB1 ấn
    Led=1;              //Đèn Led tắt
  }
}
```

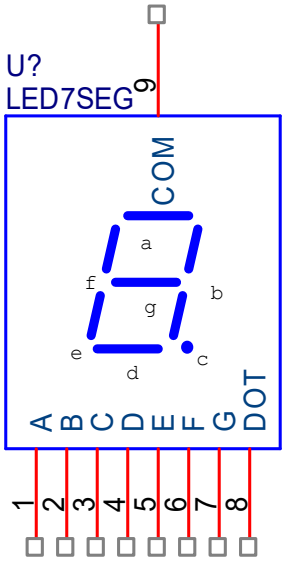
Khi lập trình đọc phím ấn cần lưu ý tới việc chống rung phím. Hiện tượng rung phím được hiểu là sự nối mạch không liên tục, tiếp điểm đóng mở chập chờn khi phím được bấm. Nguyên nhân chủ yếu là do chế tạo cơ khí của phím. Đặc điểm của hiện tượng chống rung phím là xảy ra trong khoảng thời gian rất ngắn (thường dưới 1ms) trong giai đoạn đầu khi phím được bấm. Hậu quả gây ra của hiện tượng rung phím là làm cho việc nhận biết phím bấm thành nhiều lần so với thực tế một lần được bấm. Để chống rung phím, người ta có thể dùng phần cứng hoặc phần mềm. Với chống rung bằng phần cứng thì chỉ cần nối song song với phím bấm một tụ điện, hoặc có thể dùng mạch chốt. Chống rung bằng phần mềm có ưu điểm là giảm thiểu thiết kế phần cứng. Với ma trận phím thường người ta không chống rung bằng phần cứng vì sẽ làm phức tạp mạch. Cách chống rung bằng phần mềm đơn giản và cũng rất hiệu quả, đó là dùng trễ. Sau khi phát hiện được phím bấm lần đầu tiên, ta trễ một khoảng thời gian từ vài ms đến vài chục ms để loại bỏ khoảng thời gian diễn ra rung phím ban đầu.

II. GHÉP NỐI LED 7 THANH

LED 7 thanh được cấu tạo từ các LED đơn sắp xếp theo các thanh nét để có thể biểu diễn các chữ số hoặc vài ký tự đơn giản. Tuỳ vào kích thước của chữ số mà mỗi thanh/nét được cấu tạo

bởi một hay nhiều LED đơn. Các LED 7 thanh cỡ nhỏ thường có các thanh được cấu tạo bằng một LED đơn, người ta đặt tên cho các thanh bằng các chữ cái a…g , ngoài các nét còn có thêm một dấu chấm dot (cũng có thể sáng hoặc tắt) được cấu tạo bằng một LED đơn. Như vậy cần 08 tín hiệu để điều khiển từng nét (từng diode phát quang) của LED.

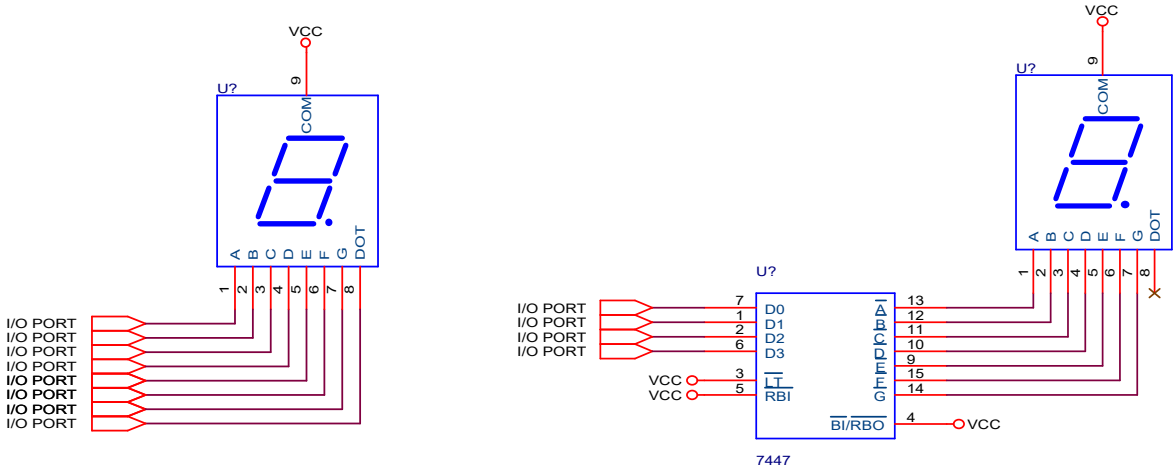
LED 7 thanh được sản xuất theo 2 loại: Anode chung và Cathode chung, điều khiển làm việc tương tự như cách bơm dòng và nuốt dòng của LED đơn. Tuy nhiên cách điều khiển LED 7 thanh hay nhất là sử dụng loại Anode chung cấp dòng, đảo trạng thái thông qua đệm với chú ý phải và quét LED. Lúc này chân Anode chung có mức 1 (điện áp dương), để cho từng thanh LED (a,b,c…) sáng thì mức logic ở các chân A, B, C … sẽ là 0.



	D7	D6	D5	D4	D3	D2	D1	D0	
P0	dot	g	f	e	d	c	b	a	HEX
0	1	1	0	0	0	0	0	0	C0h
1	1	1	1	1	1	0	0	1	F9h
2	1	0	1	0	0	1	0	0	A4h
3	1	1	1	1	0	0	1	0	B0h
4	1	0	0	1	1	0	0	1	99h
5	1	0	0	1	0	0	1	0	92h
6	1	0	0	0	0	0	1	0	82h
7	1	1	1	1	1	0	0	0	F8h
8	1	0	0	0	0	0	0	0	80h
9	1	0	0	1	0	0	0	0	90h

Hình 5.3 Cấu tạo Led 7 thanh, mã Led thanh Anode chung

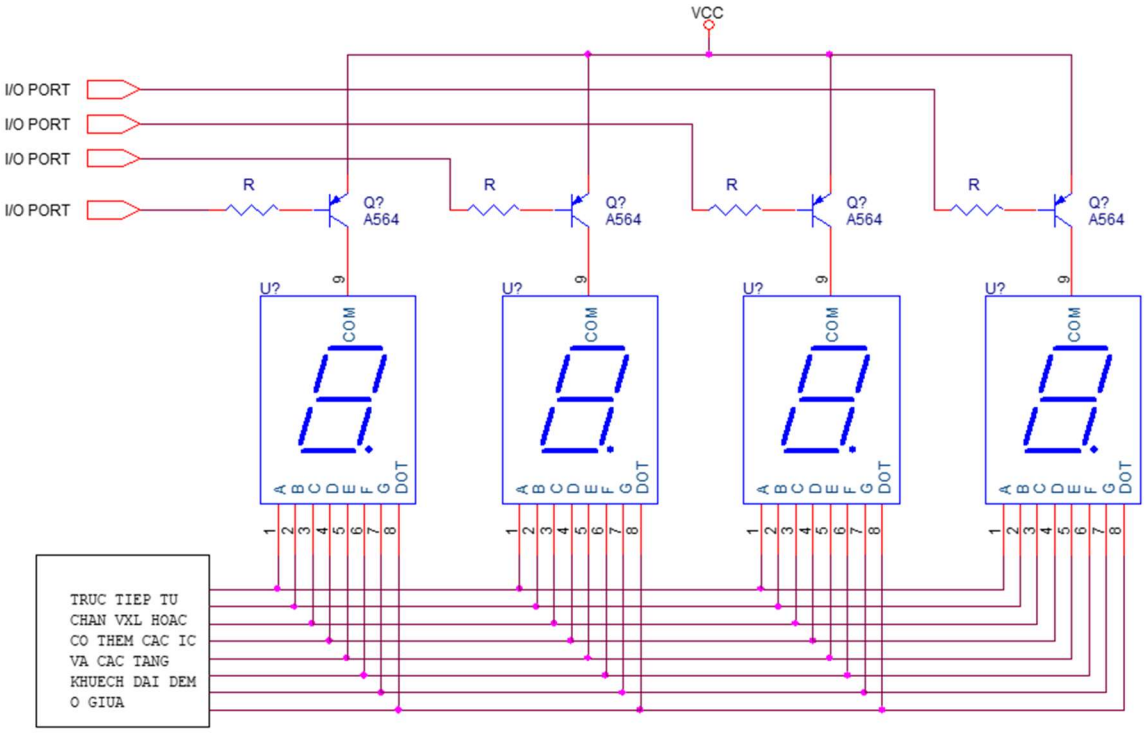
Có thể sử dụng các cách sau để ghép nối Led 7 thanh với vi xử lý:



Dùng trực tiếp các cổng vi xử lý *Dùng IC 7447 (IC giải mã BCD sang LED 7 thanh)*

Hình 5.4 Ghép nối Led 7 thanh với vi xử lý

Thông thường trong các thiết kế, LED 7 thanh được dùng để hiển thị các giá trị số từ 0 ÷ 9 và các giá trị được hiển thị thường bao gồm nhiều chữ số (tức là phải dùng nhiều LED 7 thanh). Để ghép nối với nhiều LED 7 thanh thay vì phải dùng 8 chân riêng rẽ cho mỗi LED (ví 4 Led 7 thanh sẽ mất 4x8 = 32 chân vào ra của vi xử lý), người ta dùng chung các đường dữ liệu cho các LED 7 thanh và thiết kế thêm các tín hiệu điều khiển cấp nguồn riêng rẽ cho từng LED (tức là điều khiển cấp nguồn cho các chân Anode chung hoặc Cathode chung). Sơ đồ ghép nối như sau:



Hình 5.5 Ghép nối nhiều Led 7 thanh với vi xử lý

Mạch tạo dữ liệu là một trong 2 cách ghép nối dùng trực tiếp vi xử lý hoặc qua các tầng công suất riêng. Trong mạch nguyên lý của mạch thí nghiệm KIT1 ta dùng IC ULN2803 để làm tầng đệm nuốt dòng cho các LED. IC ULN2803 là IC bao gồm bên trong 8 mạch transistor mắc Darlington và có khả năng nuốt được dòng tới vài trăm mA cho mỗi mạch Darlington.

Với phương pháp này người ta tiết kiệm được số lượng tín hiệu điều khiển đồng thời tiết kiệm được năng lượng tiêu thụ do phương pháp hiển thị là phương pháp quét sáng từng LED trong thời gian ngắn.

Ví dụ muốn sáng số 1250 với 4 Led 7 thanh thực hiện các bước sau:

- Thời điểm T_1: Điều khiển Anode của Led hàng nghìn ở mức cao, trong khi Anode của Led hàng trăm, chục và đơn vị ở mức thấp. Đưa dữ liệu 0xF9 (mã của số 1) vào đường BUS Cathode, thời điểm này chỉ có Led hàng nghìn sáng số 1.

- Thời điểm T_2: Đưa giá trị 0xFF ra đường BUS Cathode để tắt số 1, điều khiển Anode của Led hàng nghìn mức thấp (tắt Led) và Anode của Led hàng trăm mức cao (sáng Led). Đưa dữ liệu 0xA4 ra đường BUS Cathode, tời điểm này chỉ có Led hàng trăm sáng số 2.

- Thời điểm T_3: Tương tự như trên, chỉ khác dữ liệu ra đường BUS Cathode là 0x92

- Thời điểm T_4: dữ liệu sáng số 0 là 0xC0

- Sau đó lặp lại T_1. Như vậy các Led sẽ lần lượt sáng với khoảng thời gian T giữa các thời điểm, do mắt người có tính chất lưu ảnh nên có cảm nhận các Led cùng sáng.

Thời gian T quyết định cường độ sáng của các Led, chu kỳ lặp lại các Led gọi là thời gian quét Led. Nếu chu kỳ này quá chậm sẽ có hiện tượng nháy Led, do mất hiệu ứng lưu ảnh của mắt người. Nếu chu kỳ này nhỏ sẽ dẫn đến T quá nhỏ làm giảm cường độ sáng của Led.

Một cách khác là sử dụng IC7447 giải mã, tuy nhiên giải pháp này bị hạn chế phải hiển thị dấu phẩy riêng khi sử dụng hiển thị số có giá trị sau dấu phẩy, ví dụ hiển thị nhiệt độ là 28.5 ^{0}C. Thông thường cách này ít được sử dụng hơn so với cách hiển thị trực tiếp.

Chương trình ví dụ dưới đây thực hiện nhiệm vụ sau:

- Hiển thị giá trị 4 số trên Led 7 thanh với T= 5ms
- Khi ấn phím UP: giá trị số sẽ tăng thêm 1 giá trị
- Khi ấn phím DOWN: giá trị giảm đi 1
- Khi ấn RESET: giá trị xóa về 0.

```
#include <xc.h>
#define Input  1
#define Out    0
#define kb0 PORTAbits.RA4
#define kb1 PORTAbits.RA5
#define kb2 PORTEbits.RE0

#define Tris_kb0 TRISAbits.TRISA4
#define Tris_kb1 TRISAbits.TRISA5
#define Tris_kb2 TRISEbits.TRISE0
#define Tris_kb3 TRISEbits.TRISE1

#define Led PORTEbits.RE2
#define Tris_Led TRISEbits.TRISE2

#define Led_7SEG       PORTD
#define Tris_Led_7SEG TRISD

#define d1 PORTAbits.RA0
```

```
#define d2 PORTAbits.RA1
#define d3 PORTAbits.RA2
#define d4 PORTAbits.RA3

#define Tris_d1 TRISAbits.TRISA0
#define Tris_d2 TRISAbits.TRISA1
#define Tris_d3 TRISAbits.TRISA2
#define Tris_d4 TRISAbits.TRISA3
unsigned char bLed;
unsigned int i;
char number[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};

 unsigned int num,j;
 unsigned char donvi,truc,tram,ngin,digit;
 unsigned int count, count_s;
 unsigned char pre_kb0,pre_kb1,pre_kb2;
 unsigned int count_t0;

void hienthi(void)
{
      Led_7SEG=0xff;
   switch (digit)
   {
     case 0:
           d4=1;d1=0;d2=1;d3=1;
           Led_7SEG=number[ngin];
       break;
      case 1:
        d1=1;d2=0;d3=1;d4=1;
       Led_7SEG=number[tram];
        break;
      case 2:
       d1=1;d2=1;d3=0;d4=1;
       Led_7SEG=number[truc];
        break;
        case 3:
       d1=1;d2=1;d3=1;d4=0;
       Led_7SEG=number[donvi];
   }
```

```
}
void timer2_initialize ()
{
   T2CKPS0=1;
   T2CKPS1=0;
   PR2=100;     //0.1ms
  /*  Hệ số chia 10   */
   TOUTPS0=1;
   TOUTPS1=0;
   TOUTPS2=0;
   TOUTPS3=1;
   TMR2ON=1;
   TMR2IE=1;
   PEIE=1;
   GIE=1;
}
 void __interrupt()  Count_Up(void)
{
  if (TMR2IE && TMR2IF)
{
  TMR2IF=0;
    count++;
       if(count==5)  //5ms
       {
     count=0;
     hienthi();
     digit++;
     if(digit==4)
       digit=0;
     count_s++;
     if(count_s==20)
     {
           Led=~Led;
        count_s=0;
     ngin=num/1000;
      j=num%1000;
       tram=j/100;
        j=j%100;
      truc=j/10;
```

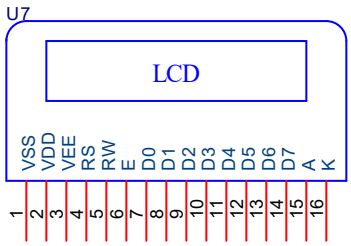
```
      donvi=j%10;
     }
    }
}
}
int  main ()
{
   ADCON1 = 0x06;
     Tris_kb0=Input ;  Tris_kb1=Input; Tris_kb2=Input; // Các phím là kb0-kb2 là cổng vào
     Tris_Led=Out;    //Led đơn là cổng ra
   Tris_d1=Out;Tris_d2=Out;Tris_d3=Out; Tris_d4=Out; // Các chân d1-d4 là cổng ra
   Tris_Led_7SEG=0;    // Cổng ĐK Led 7 thanh là cổng ra
   timer2_initialize ();    // Khởi tạo Timer2
     while(1)
     {
              if(kb0)                    // Phím kb0 bấm
                pre_kb0=1;
               if(pre_kb0&&!kb0)     //Phím kb0 bắt đầu nhả
               {
                pre_kb0=0;
                num=0;
               }
              if(kb1)
               pre_kb1=1;
               if(pre_kb1&&!kb1)
               {
                pre_kb1=0;
               num++;
               if(num==10000)
                num=0;
                }
               if(kb2)
                pre_kb2=1;
                if(pre_kb2&&!kb2)
               {
                pre_kb2=0;
               if(num>0)
                num--;
             }
```

```
        } //while
} //main
```

III. Ghép nối LCD (Liquid Crystal Display)

Đối với một số ứng dụng thực tế, khi cần các hiển thị dạng ký tự sẽ sử dụng LCD thay vì dùng LED 7 thanh. Thực chất sử dụng LCD với mã ASCII để gửi dữ liệu ra LCD, ngoài ra người thiết kế còn có thể sử dụng một vùng nhớ khai báo các mã riêng mà mình muốn.. Một chú ý đặc biệt quan trọng khi sử dụng LCD là mọi LCD khi khởi động được cấp nguồn đều phải đợi một thời gian để chúng ổn định các tinh thể (thường từ 20ms ÷ 30ms). Các lệnh tác động dữ liệu vào LCD lúc này là rất nguy hiểm, làm cho LCD không hoạt động đúng nữa (phải cắt nguồn, khởi động lại).

Một số dạng LCD thông dụng và dạng mã hiển thị ASCII 8 bit:



Bố trí chân LCD

Hình 5.6 Màn hình LCD và bảng mã hiển thị ký tự

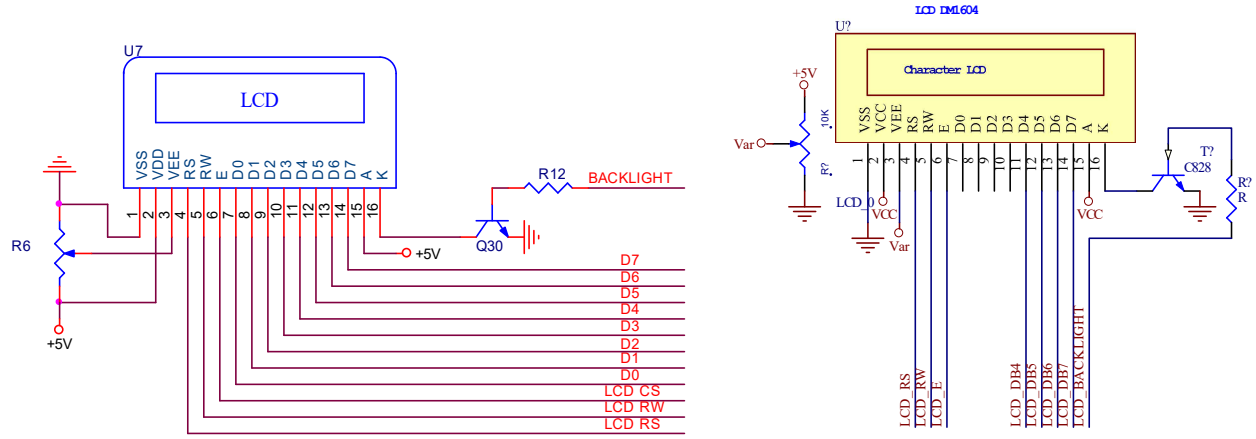
LCD có thể có 1 dòng, 2 dòng, 4 dòng, mỗi dòng có 8 cột, 16 cột, 20 cột hoặc 40 cột. Thực chất mỗi phần tử trên LCD là một ma trận điểm con hiển thị được một ký tự, có kích thước thường là 5x7 hoặc 5x10. Ký tự được hiển thị trên LCD khi ta đưa mã ASCII của ký tự đó vào vùng RAM 40 Byte trên LCD, có địa chỉ từ 00h÷27h và 40h÷67h.

Ý nghĩa các chân của LCD:

Chân	Ký hiệu	Chức năng
1	V_{SS}	Chân cấp nguồn âm
2	V_{DD}	Chân cấp nguồn dương
3	V_{EE}	Chân chỉnh độ tương phản
4	RS	Chân chọn thanh ghi
5	RW	Báo Ghi/Đọc
6	E	Cho phép (tích cực mức cao)
7÷14	D0÷D7	Dữ liệu đưa vào

- Ghép nối với vi xử lý

V_{EE} nằm trong khoảng từ V_{SS} ÷ V_{DD} , người ta thường sử dụng một biến trở để thay đổi điện áp này. Biến trở R6 cỡ 10K, chỉnh biến trở này để có được độ tương phản màn hình mong muốn.



Giao tiếp 8 bits dữ liệu

Giao tiếp 4 bits dữ liệu

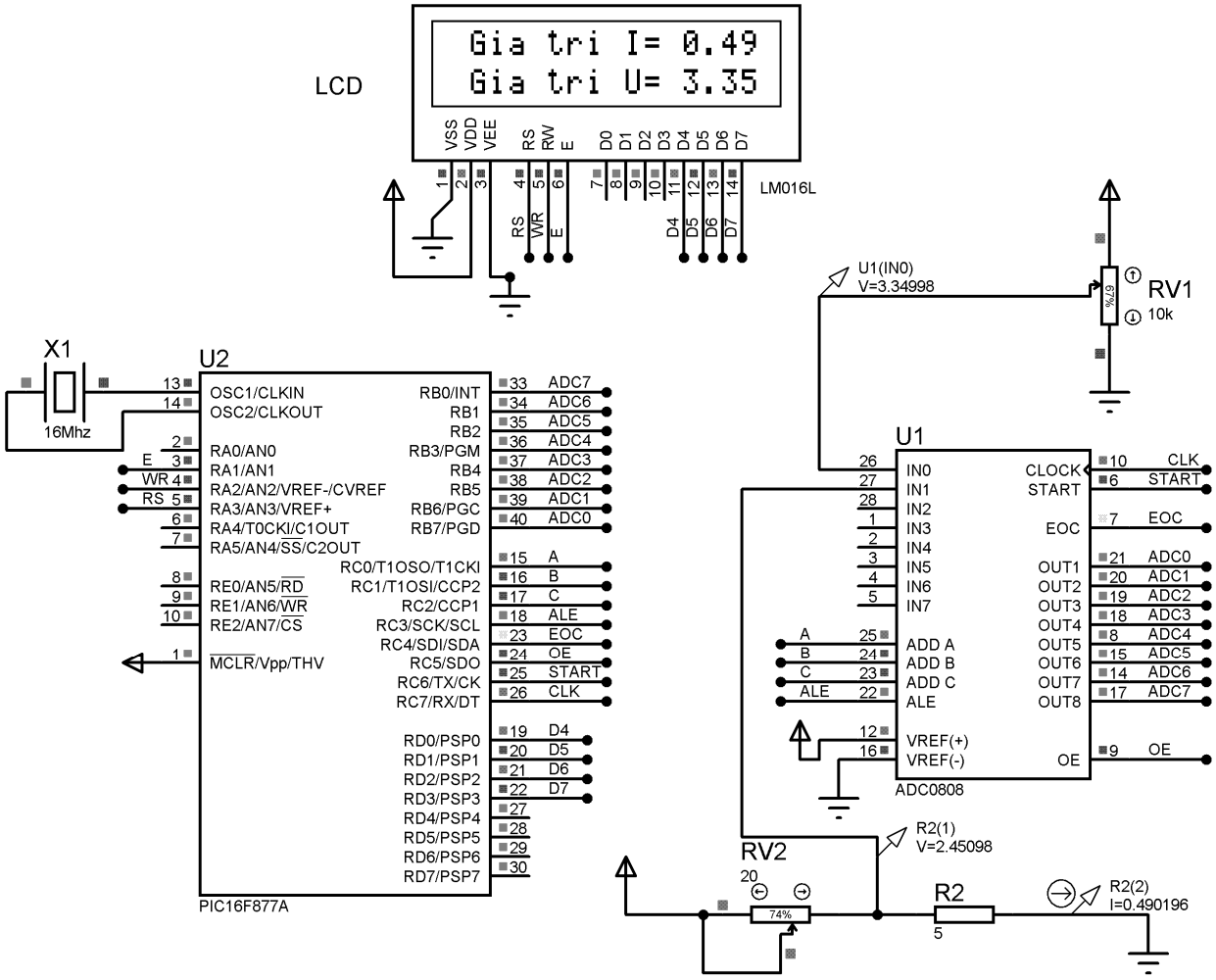
Hình 5.7 Ghép nối LCD với vi xử lý

Giao tiếp LCD có 8 lệnh cơ bản

Trong mỗi LCD có hai thanh ghi 8 bit, 1 chứa lệnh, 1 chứa dữ liệu. Vi xử lý thông qua 8 lệnh này đặt các giá trị tương ứng vào các vị trí cần thiết điều khiển hoạt động của LCD. Trong bảng, ký tự * là mặc định khi cấp nguồn cho LCD, tuy nhiên người lập trình vẫn nên chủ động tác động vào các bit này để buộc LCD làm việc chủ động theo ý mình.

Lệnh	D7	D6	D5	D4	D3	D2	D1	D0	Chức năng
1	0	0	0	0	0	0	0	1	Xoá màn hình
2	0	0	0	0	0	0	1	x	Đưa màn hình và con trỏ về đầu
3	0	0	0	0	0	1	*I/D	S	Xác định chế độ nhập dữ liệu
4	0	0	0	0	1	D	U	B	Khởi tạo màn hình, con trỏ
5	0	0	0	1	D/C	R/L	x	x	Dịch con trỏ và màn hình
6	0	0	1	*8/4	2/1*	10/7*	x	x	Khởi tạo chế độ hoạt động

7	0	1	A	A	A	A	A	A	Đặt địa chỉ trong vùng CGRAM
8	1	A	A	A	A	A	A	A	Định địa chỉ trong vùng RAM hiển thị



Hình 5.8 Sơ đồ vi xử lý đọc giá trị điện áp vào dòng điện hiển thị LCD

Chương trình dưới đây sử dụng 1 thư viện điều khiển LCD như sau:

- File lcd_hd44780_pic16.h:

```
#include <xc.h>
#define LCD_DATA        D
#define LCD_DATA_POS    0      //Các chân RD0-RD3 nối với D4-D7
#define LCD_RS_PORT     D
#define LCD_RS_POS      4       // Chân RD4  nối với RS
#define LCD_RW_PORT     D
#define LCD_RW_POS      5       // Chân RD5  nối với RW
#define LCD_E_PORT      D
```

```
#define LCD_E_POS      6            // Chân RD6 nối với E

#define LS_BLINK       0B00000001
#define LS_ULINE       0B00000010
#define LS_NONE        0B00000000

void LCDInit(uint8_t style);         // Khởi tạo LCD
void LCDWriteString(const char *msg);           // Ghi chuỗi ký tự lên LCD
void LCDWriteInt(int val,int8_t field_length);   //Ghi giá trị số lên LCD
void LCDGotoXY(uint8_t x,uint8_t y);             // Chuyển vị trí hiển thị

void LCDByte(uint8_t,uint8_t);
#define LCDCmd(c) (LCDByte(c,0))
#define LCDData(d) (LCDByte(d,1))

void LCDBusyLoop();
#define LCDClear() LCDCmd(0b00000001)
#define LCDHome()  LCDCmd(0b00000010)

#define LCDWriteStringXY(x,y,msg) {\
 LCDGotoXY(x,y);\
 LCDWriteString(msg);\
}

#define LCDWriteIntXY(x,y,val,fl) {\
 LCDGotoXY(x,y);\
 LCDWriteInt(val,fl);\
}
```

- File main.c:

```
#include <xc.h>
#include "lcd_hd44780_pic16.h"
int main ()
{
   LCDInit(LS_NONE);
  LCDWriteString("Do U/I ...");
 __delay_ms(100);
   LCDClear();
while(1)
        {
          LCDGotoXY(1,0);
         LCDWriteString("Gia tri I=  0.49");
        __delay_ms(10);
```

```
        LCDGotoXY(1,1);
       LCDWriteString("Gia tri U=  3.35");
      __delay_ms(10);
      }
}
```

IV. Ghép nối ADC bên ngoài ADC0808

ADC0808 là IC biến đổi tương tự sang số có các đặc tính kỹ thuật sau:

- Độ phân dải 8bit
- Số kênh đầu vào 8 (IN0 đến IN7)
- Dải đo 0 đến *Vref*

Chức năng các chân như sau:

Chân	Tên	Tính năng
1-5 và 26-28	IN0-IN7	Cổng vào tương tự
6	START	Bắt đầu biến đổi (cổng vào)
7	EOC	Báo quá trình biến đổi xong (cổng ra)
8,14,17-21 và 15	2^{-1} - 2^{-8}	Dữ liệu đầu ra 8 bit (cổng ra)
9	OE	Cho phép xuất dữ liệu ra (cổng vào)
10	Clock	Xung nhịp (cổng vào)
11	V_{cc}	Nguồn cấp +
12	$V_{REF}(+)$	Điện áp chuẩn so sánh dương
13	GND	Nguồn cấp (đất)
16	$V_{REF}(-)$	Điện áp chuẩn so sánh âm
23-25	ADD A, ADD B, ADD C	Chọn kênh (đầu vào)
22	ALE	Chốt địa chỉ

Chương trình dưới đây xác định các giá trị điện áp (cổng In0) và giá trị dòng điện qua điện trở R2 (cổng In1), hiển thị trên LCD 2 hàng.

- File adc0808.h:

```
#include <xc.h>
// Define ADC Channels
#define AN0         0
#define AN1         1
#define AN2         2
#define AN3         3
#define AN4         4
#define AN5         5
#define AN6         6
#define AN7         7
```

```
#define Add_A  PORTCbits.RC0     // Address Pin A
#define Add_B  PORTCbits.RC1
#define Add_C  PORTCbits.RC2

#define ALE    PORTCbits.RC3
#define EOC    PORTCbits.RC4
#define OE     PORTCbits.RC5
#define START  PORTCbits.RC6
#define CLK    PORTCbits.RC7

#define Data_Bus      PORTB

#define HalfCycleDelay                10

void InitADC(void);   // Hàm khởi tạo ADC
unsigned int ReadADC(unsigned char);  //Hàm đọc ADC
```

- File adc0808.c:

```
#include "ADC0808.h"
void InitADC(void)
{
    TRISC=0b00001000;  //EOC input RC4
    TRISB=0xff;   //Các cổng đọc dữ liệu ADC là Input
}
unsigned int ReadADC(unsigned char Channel)
{
    unsigned int i = 0;
    unsigned int ADC_value = 0;
    switch(Channel)
    {
    case AN0: Add_C = 0;  Add_B = 0;  Add_A = 0; break;
    case AN1: Add_C = 0;  Add_B = 0;  Add_A = 1; break;
    case AN2: Add_C = 0;  Add_B = 1;  Add_A = 0; break;
    case AN3: Add_C = 0;  Add_B = 1;  Add_A = 1; break;
    case AN4: Add_C = 1;  Add_B = 0;  Add_A = 0; break;
    case AN5: Add_C = 1;  Add_B = 0;  Add_A = 1; break;
    case AN6: Add_C = 1;  Add_B = 1;  Add_A = 0; break;
    case AN7: Add_C = 1;  Add_B = 1;  Add_A = 1; break;
    }
    __delay_us(HalfCycleDelay);
    ALE = 1;                                    // Enable Address Latch
    CLK = 1;                                    // Make CLK High
    __delay_us(HalfCycleDelay);
    CLK = 0;                                    // Make CLK Low
    START = 1;                                  // Start ADC Conversion
    __delay_us(HalfCycleDelay);
    CLK = 1;                                    // Make CLK High
```

```
    ALE = 0;                                    // Disable Address Latch
    __delay_us(HalfCycleDelay);
    CLK = 0;                                    // Make CLK Low
    START = 0;                                  // Complete the start pulse
    for(i=0;i<2000;i++)
    {
        CLK = !CLK;                             // Toggle Clock
        __delay_us(HalfCycleDelay);
        if(!EOC)                            // Wait for EOC to be low
            break;
    }
    for(i=0;i<2000;i++)
    {
        CLK = !CLK;                             // Toggle Clock
        __delay_us(HalfCycleDelay);

        if(EOC)                             // Wait for EOC to be High
            break;
    }
    CLK = 0;                                    // Make CLK Low
    OE = 1;                                         // Enable Output
    __delay_us(HalfCycleDelay);
    CLK = 1;                                    // Make CLK High
    __delay_us(HalfCycleDelay);
    CLK = 0;                                    // Make CLK Low
    __delay_us(HalfCycleDelay);
    CLK = 1;                                    // Make CLK High
    ADC_value = Data_Bus;                   // Read value
    __delay_us(HalfCycleDelay);
    OE = 0;                                         // Disable Output
    CLK = 0;                                    // Make CLK Low
    __delay_us(HalfCycleDelay);
    return ADC_value;                       // Return ADC value
}
```

- File main.c:

```
#include <xc.h>
#include "lcd_hd44780_pic16.h"
#include "ADC0808.h"

#define Input  1
#define Out    0

unsigned int i;
float Dong,Dienap;
unsigned int count, count1,count_s;
unsigned char do_dienap;
```

```
float Dong,Dienap;
unsigned char adc_ok;

void timer2_initialize ()
{
   T2CKPS0=1;
   T2CKPS1=0;
   PR2=100;  //0.1ms
   TOUTPS0=1;
   TOUTPS1=0;
   TOUTPS2=0;
   TOUTPS3=1; //0.1/10
   TMR2ON=1;
   TMR2IE=1;
   PEIE=1;
   GIE=1;
}
void __interrupt()  Count_Up(void)    //1ms
{
  if (TMR2IE && TMR2IF)
   {
  TMR2IF=0;
    count++;
      if(count==20)  //20ms
      {
        count=0;
        count1++;
       if(do_dienap)
       {
         Dienap=(float)(ReadADC(AN0));     // Read ADC value from Channel 0
         Dienap=Dienap*1.962;
         adc_ok=1;
       }
        else
        {
        Dong = (float)(ReadADC(AN1));        // Read ADC value from Channel 1
        Dong=Dong/2.55;
        }
        do_dienap=~do_dienap;      // Chuyển đổi giữa đo dòng và áp
      }
  }
}

int main ()
{
   unsigned char Digit[3] = { 0,0,0 };          // To make digits to display on LCD
```

```
 unsigned int ADC_Value = 0;

 ADCON1 = 0x06;
LCDInit(LS_NONE);
InitADC();
timer2_initialize ();
LCDClear();
 LCDWriteString("Do U/I ...");       // Hiển thị khi bắt đồng chương trình
  __delay_ms(100);
 LCDClear();
       while(1)
       {
    if(adc_ok&&count1==25)    //25*20ms=500ms hiển thị lên LCD 1 lần
    {
      count1=0;
      ADC_Value=(unsigned int)Dong;
     Digit[2] = (ADC_Value/100);
    Digit[1] =  (ADC_Value%100)/10;
    Digit[0] = (ADC_Value%100)%10;
   LCDGotoXY(1,0);
  LCDWriteString("Gia tri I= ");
 LCDData(Digit[2]+0x30);
LCDData('.');
LCDData(Digit[1]+0x30);
LCDData(Digit[0]+0x30);
ADC_Value=(unsigned int)Dienap;
Digit[2] = (ADC_Value/100);
Digit[1] =  (ADC_Value%100)/10;
Digit[0] = (ADC_Value%100)%10;
  LCDGotoXY(1,1);
 LCDWriteString("Gia tri U= ");
LCDData(Digit[2]+0x30);
LCDData('.');
LCDData(Digit[1]+0x30);
LCDData(Digit[0]+0x30);
    }
     }
   }
```

V. Ghép nối ADC bên trong

Xác định giá trị dòng điện và điện áp và hiển thị trên màn hình LCD.

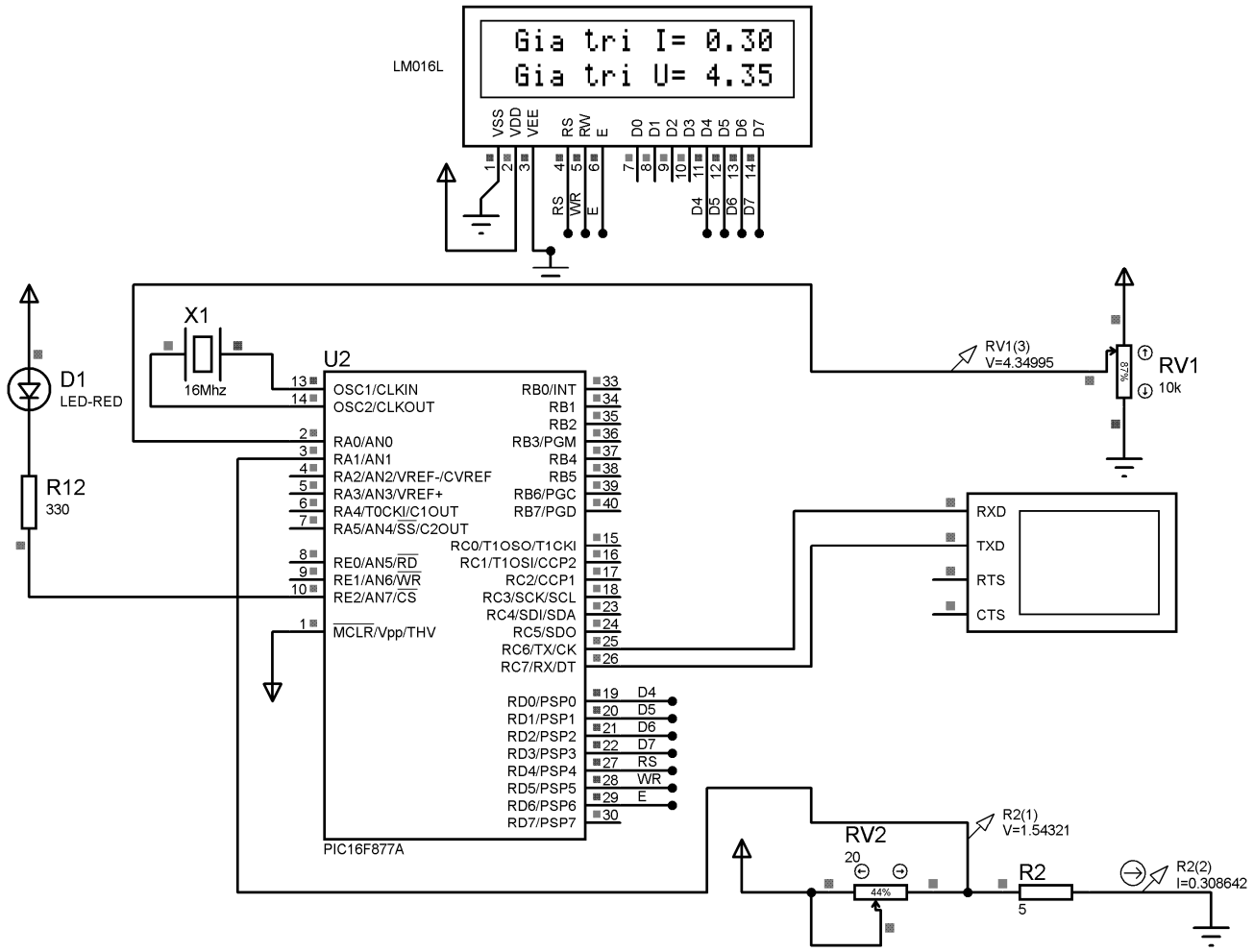
Viết lại hàm cài đặt ADC và đọc ADC sử dụng ADC bên trong vi xử lý, các phần khác sử dụng như ADC bên ngoài.

Thanh ghi ADCON0 (Địa chỉ 1Fh)

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
bit 7							bit 0

Thanh ghi ADCON1 (Địa chỉ 9Fh)

ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0



Hình 5.8 Sơ đồ vi xử lý đọc giá trị điện áp vào dòng điện dùng ADC bên trong

Chương trình:

```
#include <xc.h>
void InitADC(void)
{
 ADCON1 = 0xC4; // Dạng dữ liệu bên phải, Xung nhịp FOSC/64; Cổng Analog AN0-2.
ADCON0 =0x80; // ADMF=1
}

unsigned int ReadAdc(unsigned char Channel)
{
   unsigned int value;
   switch(Channel)
        {
```

```
        case 0: CHS0 = 0;  CHS1 = 0;  CHS2 = 0; break;
        case 1: CHS0=1;  CHS1 = 0;  CHS2 = 0;
      }
    ADON=1;
    __delay_us(5);
    GO_DONE=1;
    while(GO_DONE);
    value=(unsigned int)ADRESH*256+ADRESL;
    ADON=0;
    return value;
}
```

VI. Truyền thông UART

Truyền thông không đồng bộ UART cho phép gửi dữ liệu từ vi xử lý này sang vi xử lý khác thông qua chuỗi xung ở chân TX, ví dụ thực hiện bài tập như sau:

1. Truyền dữ liệu:

- Khi ấn phím kb0 truyền dữ liệu "Send" từ vi xử lý sang vi xử lý khác với tốc độ 9600 b/s.
- Khi ấn phím kb1 truyền dữ liệu "Data" cũng với tốc độ trên.

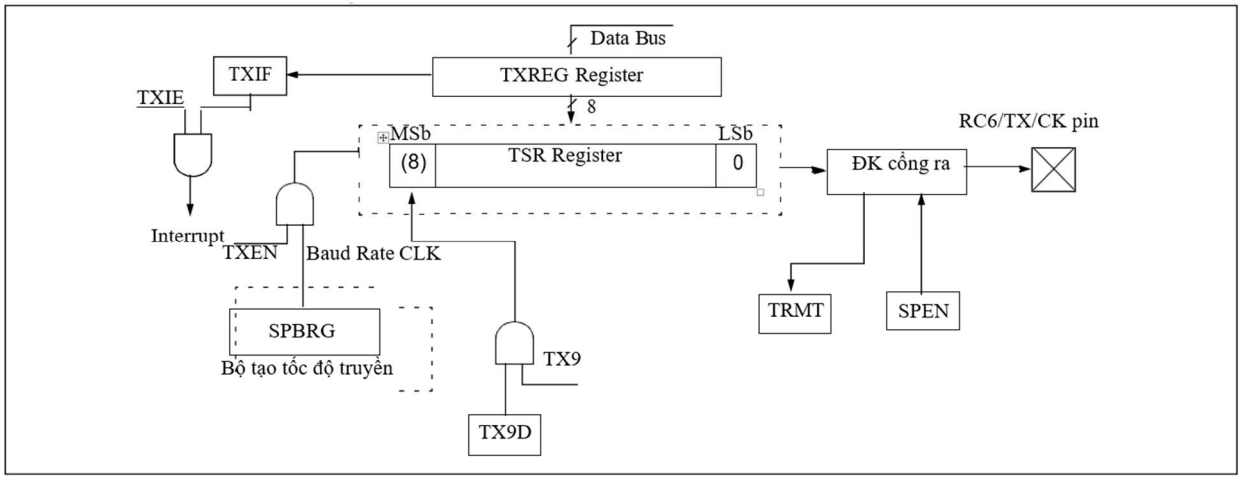
Các bước thực hiện:

- Viết hàm cài đặt UART: Init_uart()

Kích hoạt bộ UART làm việc ở chế độ truyền và nhận, tốc độ truyền thông 9600b/s và cho phép ngắt nhận.

- Viết hàm truyền Send_Data(char *str)

Gửi 1 chuỗi ký tự tại địa chỉ con trỏ str chỉ đến, tương ứng với trạng thái của phím kb0 hay kb1 bấm sẽ gọi hàm có chuỗi ký tự phù hợp.



Hình 5.9 Sơ đồ bộ truyền UART

TXSTA:Thanh ghi trạng thái và điều khiển truyền (Địa chỉ 98h)

CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D

bit 7 bit 0

RCSTA: Thanh ghi trạng thái và điều khiển nhận (Địa chỉ 18h)

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

bit 7 bit 0

Chương trình mẫu:

```
 #include <xc.h>

#define kb0 PORTAbits.RA4
#define kb1 PORTAbits.RA5
#define Tris_kb0 TRISAbits.TRISA4
#define Tris_kb1 TRISAbits.TRISA5
unsigned char pre_kb0,pre_kb1;

 void Init_uart(void)
  {
        TXSTA=0x20;  // TXEN=1
        RCSTA=0x90;// SPEN=1 CREN=1
        SPBRG=25;// 9600 16mHZ
        RCIE=1;
   }
      void Sent_data(char* str)
      {
        while(*str)
        {
          TXREG=*str;
         while(TRMT==0);
         str++;
        }
   }

      int  main ()
      {
         ADCON1 = 0x06;// All Analog Input Disable
       Tris_kb0=1;
       Tris_kb1=1;
       Init_uart();
      while(1)
      {
            if(kb0)      // Phím kb0 ấn
              pre_kb0=1;
```

```
        if(pre_kb0&&!kb0)   // Pre_Kb0==1 va phím nhả kb0==0
                {
                 pre_kb0=0;
                 Sent_data("Send")
                }

        if(kb1)      // Phím kb1 ấn
           pre_kb1=1;

        if(pre_kb1&&!kb1)   // Pre_Kb1==1 va phím nhả kb1==0
                {
                 pre_kb1=0;
                 Sent_data("Data")
                }
}//while
}//main
```

2. Nhận dữ liệu

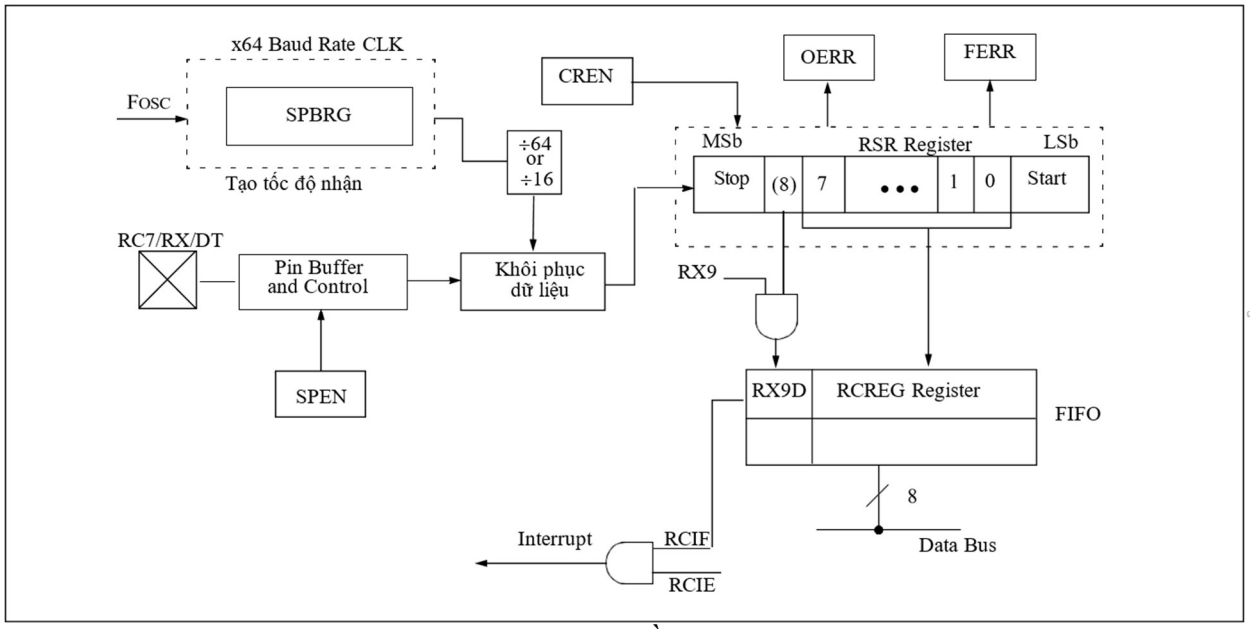
- Khi nhận được dữ liệu "Send" sáng Led.
- Khi nhân được dữ liệu "Data" tắt Led.

Các bước thực hiện:

- Viết hàm cài đặt UART: Init_uart()

Kích hoạt bộ UART làm việc ở chế độ truyền và nhận, tốc độ truyền thông 9600b/s và cho phép ngắt nhận.

- Viết hàm ngắt nhận UART



Hình 5.10 Sơ đồ bộ nhận UART

Chương trình mẫu:

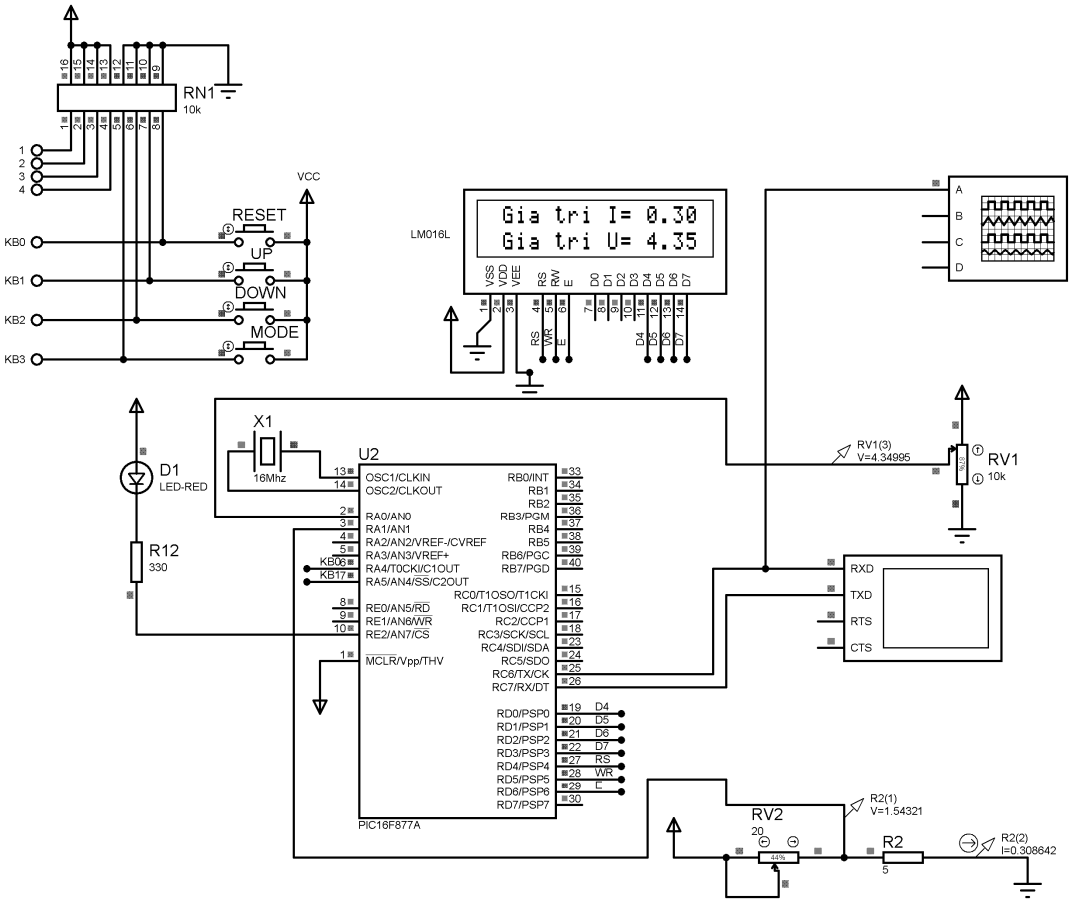
```
#include <xc.h>
#include <string.h>

#define Led PORTEbits.RE2
#define Tris_Led TRISEbits.TRISE2
char in_buffer[6];
unsigned char in_count;
void Init_uart(void)
 {
        TXSTA=0x20;  // TXEN=1
        RCSTA=0x90;// SPEN=1 CREN=1
        SPBRG=25;    // 9600b/s  16MHZ
        RCIE=1;
   }

      void __interrupt()  Receive_UART(void)    //1ms
      {
       char ch;
      if (RCIE && RCIF)
        {
         ch=RCREG;
         in_buffer[in_count++]=ch;
         if(in_count==4)
        {
          in_count=0;
       if(strncmp(in_buffer,"Send",4)==0)
             Led=1;
        if(strncmp(in_buffer,"Data",4)==0)
          Led=0;
         }
         }
      }

      int  main ()
      {
          ADCON1 = 0x06;// All Analog Input Disable
        Tris_Led=0;
       Init_uart();
      while(1)
```

```
    {
    }
}
```



Hình 5.11 Sơ đồ truyền nhận UART sử dụng vi xử lý

Bài tập

1. Tạo xung (điều khiển Led) ở cổng A2 với chu kỳ như sau:
   - Led sáng 2 giây, tắt 0.3 giây, sáng 1 giây, tắt 1 giây và lặp lại.

2. Tạo xung (điều khiển Led) ở cổng C1 (Led1) và C3 (Led 2) như sau:
   - Bước 1 (2s): Led1 sáng, Led2 tắt
   - Bước 2 (3s): Led1 tắt, Led 2 tắt
   - Bước 3 (0.5 s): Led1 sáng, Led2 sáng

   Lặp lại

3. Tạo xung (điều khiển Led) ở cổng C1 (Led1) và C3 (Led 2) như sau:
   - Led 1 sáng nhấp nháy 0.5s sáng, 0.5s tắt
   - Led 2 sáng 0.3s, tắt 0.6s, sáng 0.1s, tắt 0.3s lặp lại

4. Ấn phím kb0 thì sáng "1234" còn ấn kb1 thì sáng "5678"

5. Ấn phím kb0 sáng Led0, nhả phím kb0 Led0 tắt sau thời gian 2 giây. Ấn phím kb1 sáng Led 7 thanh số tăng 1 số.

6. Ấn phím kb0 Led 7 thanh tự động tăng 1 giá trị sau thời gian 1 giây, khi ấn phím kb1 dừng tăng.

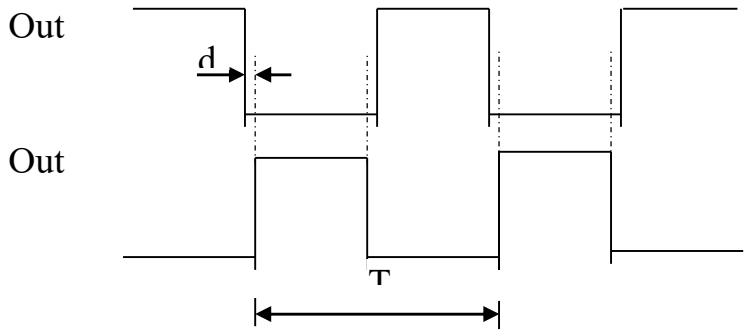
7. Thiết kế mạch, lập trình và mô phỏng điều khiển 3 Led đơn trong đó: led1 sáng nhấp nháy liên tục theo chu kỳ 1ms tắt, 2ms sáng; led2 sáng theo chu kỳ 10ms sáng, 20ms tắt, 30ms sáng, 50ms tắt; led 3 sáng và tắt theo công thức led3=led2 or led1.

8. Thiết kế mạch, lập trình và mô phỏng hiển thị số trên 2 Led 7 thanh theo sự điều khiển của phím bấm: khi phím bấm chữ số tăng dần với thời gian 0.5 giây/1 số, khi nhả phim số giảm dần với thời gian 1 giây/1 số. Giới hạn của giá trị hiển thị từ 0 đến 99.
9. Thiết kế mạch, lập trình và mô phỏng hiển thị số trên 3 Led 7 thanh theo sự điều khiển của phím bấm: khi phím bấm chữ số tăng dần với thời gian 0.5 giây/1 số, khi nhả phim số giảm dần với thời gian 1 giây/1 số. Giới hạn của giá trị hiển thị từ -50 đến 50, số đầu tiên là dấu của chữ số.
10. Hiển thị trên LCD:
    - Hàng trên " Hien thi LCD "nhấp nháy với thời gian 1 giây sáng, 1 giây tắt"
    - Hàng dưới "02 Hang 16 Cot" sáng liên tục.

11. Hiển thị trên LCD:
    - Hàng trên " Bang hien thi"
    - Hàng dưới " So dem n" trong đó giá trị n là số có 3 chữ số, tăng dần với thời gian 0.5 giây sẽ tăng 1 giá trị.

12. Đọc giá trị điện áp từ cổng Analog số 5, điều khiển Led đơn sáng theo giá trị này như sau:
    - Hiển thị trên LCD
    - Điện áp nhỏ 2.5V Led nháy với chu kỳ 0.1s sáng 0.3s tắt
    - Điện áp lớn hơn 2.5V, Led nháy 0.5s sáng 0.5 tắt.

13. Tạo xung ra 2 cổng vi xử lý theo giản đồ thời gian sau:



Trong đó:

Chu kỳ T=50ms

Độ rộng điều chế τ=21ms

Thời gian không trùng xung (DeadTime) d=50µs.

14. Khi ấn phím kb0 truyền dữ liệu " Send" từ vi xử lý sang vi xử lý khác với tốc độ 19200 b/s.

Khi ấn phím kb1 truyền dữ liệu "Data" cũng với tốc độ trên.

Bài tập lớn Kỹ thuật vi xử lý

1. Thiết kế,chế tạo bộ điếm sản phẩm có 2 xung đầu vào:
   - Giá trị đếm tối đa 9999.
   - Có 2 phím bấm mô phỏng sản phẩm đã hoàn thành tương ứng với từng đầu vào của bộ đếm.
   - Hiển thị đồng thời giá trị đếm trên Led 7 thanh.
   - Có 2 phím bấm có tác dụng Reset giá trị đếm tương ứng về 0
   - Điều khiển Led hoạt động như sau:

TT	Nội dung so sánh	Out
1	Giá trị đếm 1=giá trị đếm 2=0	Tắt
2	Giá trị đếm 1=giá trị đếm 2>0	Sáng
3	Giá trị đếm 1<giá trị đếm 2	Sáng nhấp nháy với chu kỳ 1 giây (0.5 giây sáng, 0.5 giây tắt)
4	Giá trị đếm 1>giá trị đếm 2	Sáng nhấp nháy với chu kỳ 2 giây (0.5 giây sáng, 1.5 giây tắt)

2. Thiết kế chế tạo bộ đo điện áp, và dòng điện có tính năng như sau:
   - Dải đo điện áp 0-30VDC, độ chính xác phép đo <=0.1V
   - Dải đo dòng điện 0-1A, độ chính xác phép đo <=0.05A.
   - Hiển thị đồng thời 2 giá trị trên Led 7 thanh.
   - Điều khiển Led báo trạng thái quá áp:
     Khi điện áp <15V, Led tắt. điện áp >=15V Led nháy với chu kỳ 0.5 giây sáng, 0.5 giây tắt.
   - Điều khiển Led báo trạng thái quá dòng:
     Khi dòng <0.75A, Led tắt. dòng >=0.75A Led nháy với chu kỳ 0.5 giây sáng, 0.5 giây tắt.
3. Thiết kế chế tạo bộ điều khiển động cơ điện 1 chiều có đặc tính sau:
   - Nguồn cấp 12VDC
   - Điện áp động cơ 12VDC, không đảo chiều động cơ
   - Sử dụng triết áp để thay đổi tốc độ động cơ.
   - Sử dụng 1 công tắc để cho động cơ chạy.
   - Hiển thị giá trị điện áp của động cơ trên màn hình Led 7 thanh với giá trị sai số <0.1V

4. Thiết kế chế tạo bộ điều khiển động cơ điện 1 chiều sử dụng Thyristor chỉnh lưu:
   - Nguồn cấp 12VAC
   - Điện áp động cơ 12VDC, không đảo chiều động cơ
   - Sử dụng triết áp để thay đổi tốc độ động cơ.

- Sử dụng 1 công tắc để cho động cơ chạy.
- Hiển thị giá trị điện áp của động cơ trên màn hình Led 7 thanh với giá trị sai số <0.1V

5. Thiết kế chế tạo bộ đo và hiển thị giá trị điện trở:
   - Giải đo: 100 - 2000 Ω
   - Sai số phép đo : ±1Ω.
   - Điều khiển Led nhấp nháy theo giá trị điện trở:

TT	Giá trị điện trở	Out
1	100 Ω≤R<500 Ω	Tắt
2	500 Ω≤R<1K Ω	Sáng
3	1K Ω≤R<1.5K Ω	Sáng nhấp nháy với chu kỳ 1 giây (0.3 giây sáng, 0.8 giây tắt)
4	1.5K Ω≤R<2K Ω	Sáng nhấp nháy với chu kỳ 2 giây (0.3 giây sáng, 1.5 giây tắt)

6. Thiết kế chế tạo bộ điều khiển động cơ bước có đặc tính sau:
   - Nguồn cấp 5VDC
   - Điện áp động cơ 5 VDC, không đảo chiều động cơ
   - Sử dụng triết áp để thay đổi tốc độ động cơ.
   - Sử dụng 1 công tắc để cho động cơ chạy,.
7. Thiết kế chế tạo mạch hiển thị trên Led ma trận 8x8 có 1 màu như sau:
   - Số Module ghép nối >=3.
   - Hiển thị chuỗi ký tự không dấu theo ý muốn.
   - Chế độ hiển thị chạy từ phải sáng trái với tốc độ dịch 0.5 giây 1 điểm ảnh.
   - Chế độ sáng nhấp nháy với chu kỳ 0.5 giây sáng 0.5 giây tắt.
   - Sử dụng 1 công tắc để chọn chế độ hiển thị
8. Thiết kế chế tạo đồng hồ thời gian thực với các thông số sau:
   - Hiển thị trên LCD: giờ phút giây, ngày tháng năm.
   - Truyền thông RS232 về máy tính để đồng bộ thời gian từ máy tính.

9. Thiết kế chế tạo mô hình điều khiển đèn giao thông ở ngã tư có hiển thị đếm lùi.

10. Thiết kế chế tạo bộ điều khiển tốc độ quạt sử dụng Triac.