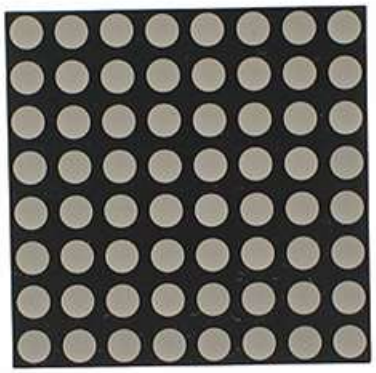
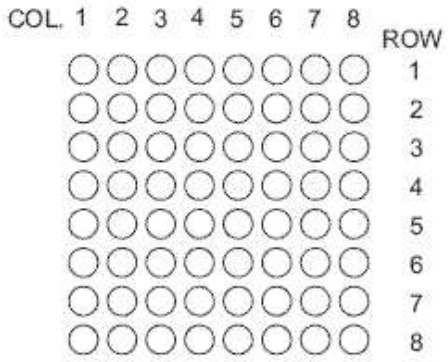
# Bài 5 : Kỹ thuật quét LED

***Nội dung:***
Nd1. Khảo sát cấu tạo, hoạt động của LED ma trận và LED 7 đoạn.
Nd2. Tìm hiểu kỹ thuật quét LED ma trận và LED 7 đoạn.
***Yêu cầu:***
Yc1. Viết chương trình cho phép hiển thị hình ảnh ra LED ma trận.
Yc2. Viết chương trình cho phép hiển thị giá trị số ra LED 7 đoạn.

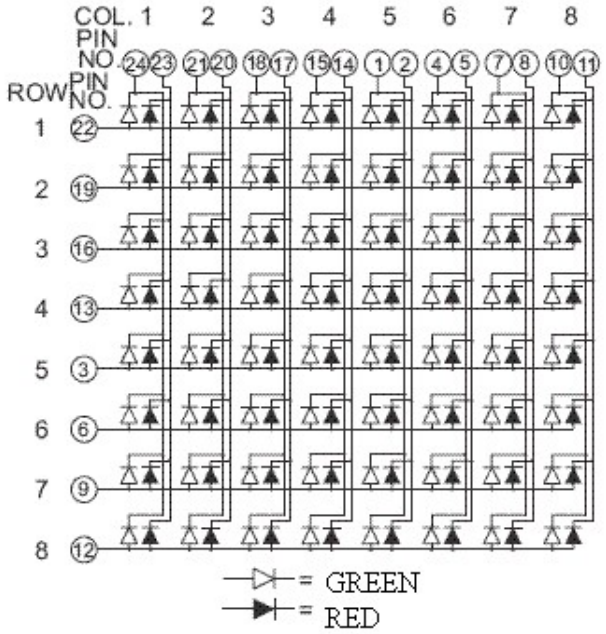
## *5.1 Cấu tạo LED ma trận và LED 7 đoạn:*

LED ma trận 8x8 hai màu được bố trí thành 8 hàng và 8 cột. Mỗi điểm có hai LED.

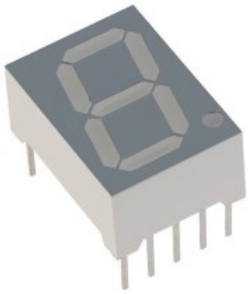


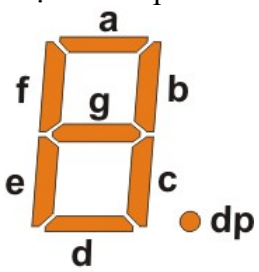



Các LED trên cùng một hàng nối chung Anode, các LED cùng loại trên cùng một cột nối chung Cathode.

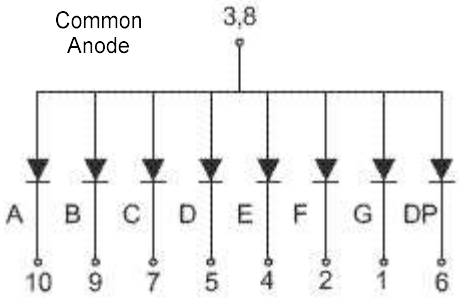


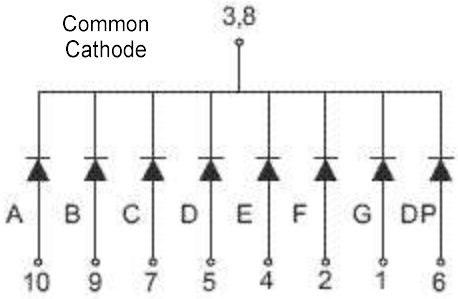
LED 7 đoạn gồm có 7 đoạn được đánh dấu: a, b, c, d, e, f, g và một điểm dp.






LED 7 đoạn có hai loại là Common Anode và Common Cathode, tương ứng các LED nối chung Anode hay nối chung Cathode.

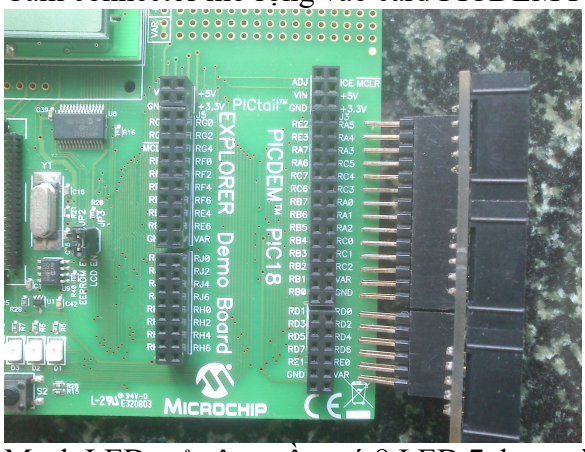


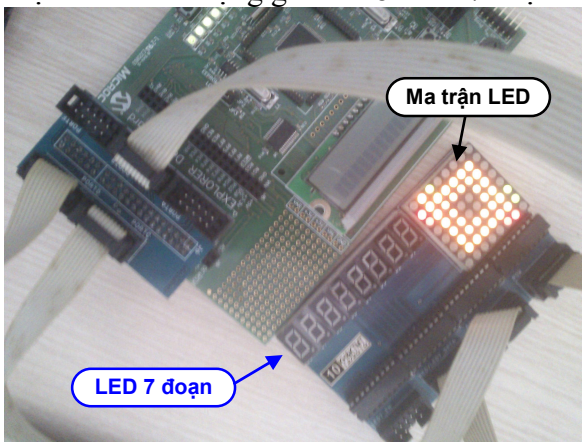

## 5.2 Thực hiện yêu cầu 1: xuất dữ liệu ra 1 hàng LED ma trận

Bước 1. Kết nối mạch.

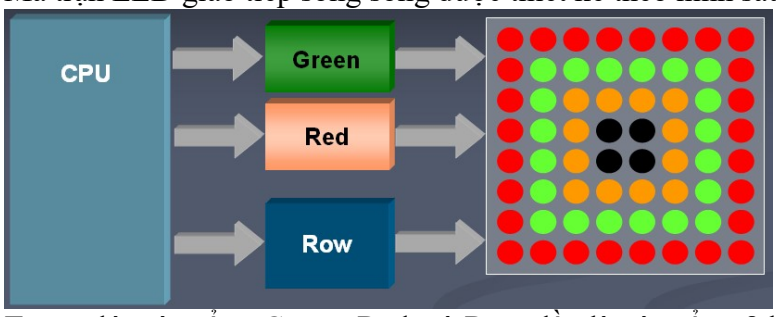
Cắm connector mở rộng vào card PICDEM PIC18 như trong hình.




Mạch LED mở rộng gồm có 8 LED 7 đoạn và 1 LED ma trận hai màu xanh/đỏ (xem hình).



Ma trận LED giao tiếp song song được thiết kế theo hình sau:




Trong đó, các cổng Green, Red, và Row đều là các cổng 8 bit xuất. Ta dùng 3 sợi cáp để kết nối với connector theo như sau:

- Connector PortB ↔ Card LED Row (xuất chọn hàng 0-7).
- Connector PortC ↔ Card LED Green (xuất dữ liệu cột xanh).
- Connector PortD ↔ Card LED Red (xuất dữ liệu cột đỏ).

Bước 2. Tạo dự án mới Tn05, tập tin nguồn Tn05_mtled.c.

Bước 3. Định nghĩa, khởi động các port đã chọn và nút nhấn RA5.

```
#define XUAT          0
#define NHAP          1
#define NUTNHAN       PORTAbits.RA5
#define NUTNHAN_IO    TRISAbits.RA5
volatile unsigned char row   @ 0xF8A;     //LATB
volatile unsigned char row_io  @ 0xF93; //TRISB
volatile unsigned char gcol @ 0xF8B;     //LATC
volatile unsigned char gcol_io @ 0xF94; //TRISC
volatile unsigned char rcol @ 0xF8C;     //LATD
volatile unsigned char rcol_io @ 0xF95; //TRISD
void init()
{    ADCON1=0x0F;
     NUTNHAN_IO=NHAP;      //Nut nhan RA5 nhap
     row_io=XUAT;          //Cac port LED xuat
     gcol_io=XUAT;
     rcol_io=XUAT;
}
```

Bước 4. Để xuất dữ liệu cho ma trận LED sáng hàng 0 như hình sau:

Đỏ Xanh Cam Đen

Điều khiển trộn màu tuân theo quy luật sau:

Màu đỏ xuất hiện là do LED xanh tắt (bit dữ liệu xanh=0), LED đỏ sáng (bit dữ liệu đỏ=1).

Màu xanh là do LED xanh sáng, LED đỏ tắt.

Màu cam là do LED xanh, LED đỏ cùng sáng.

Màu đen là do LED xanh, LED đỏ cùng tắt.

Sinh viên cần xác định dữ liệu xanh G_DATA (8 bit) và dữ liệu đỏ R_DATA (8 bit) sao cho khi trộn các bit tương ứng lại sẽ cho ra màu theo mẫu yêu cầu.

Trong hàm **`main()`** đánh đoạn code thực hiện quy trình sau:

- Xoá tất cả các hàng (nghĩa là tắt hết các hàng, tuy nhiên do hiện tượng lưu ảnh trên LED nên ta vẫn thấy LED sáng).
- Xuất G_DATA ra cổng dữ liệu xanh (gcol).

- Xuất R_DATA ra cổng dữ liệu đỏ (rcol).
- Chọn hàng 0 bằng cách xuất dữ liệu **0b00000001** ra cổng chọn hàng (row).

Bước 5. Chạy chương trình và quan sát.

Bước 6. Thêm code kiểm tra nút RA5 thực hiện xoay vòng chọn hàng từ hàng 0 đến hàng 7 như sau:

```
#define G_DATA      00
#define R_DATA      00
void main(void)
{   init();
    row=0;gcol=G_DATA;rcol=R_DATA;row=0b00000001;
    while(1)
    {   if  (NUTNHAN==0)
        {   row<<=1;
            if (row==0) row=1;
            while(NUTNHAN==0);
        }
    }
}
```

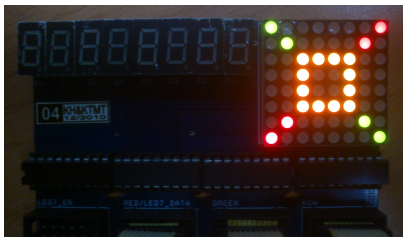
Bước 7. Chạy chương trình và quan sát.

## 5.3 Thực hiện yêu cầu 1: xuất hình ra LED ma trận

Bước 8. Module Tn05_mtled2.c xuất khung hình. Để xuất 1 khung hình có 8 hàng ra ma trận LED, ta cần luân phiên xuất dữ liệu ra từng hàng từ hàng 0 đến hàng 7 theo chỉ số **rowidx**.

Dữ liệu của 8 hàng sẽ được tổ chức theo bảng trong ROM thành 16 byte liên tiếp (8 byte đầu là dữ liệu xanh, 8 byte sau là dữ liệu đỏ).

```
#define COLMAX      8
#define ROWMAX      8
#define FRAME       2
const unsigned char maurom[FRAME][ROWMAX]={
    {0x01,0x02,0x3C,0x24,0x24,0x3C,0x40,0x80},
    {0x80,0x40,0x3C,0x24,0x24,0x3C,0x02,0x01}
};
```



Định nghĩa trong RAM vùng đệm dữ liệu led 16 byte tương tự ROM.

```
unsigned char rambuf[FRAME][ROWMAX],rowidx;
```

Thực hiện chép 16 byte dữ liệu từ **maurom** trong ROM sang **rambuf** của RAM.

```
#define RED         0
#define GREEN       1
void rom2ram()
{   unsigned char i;
    for(i=0;i<COLMAX;i++)
    {   rambuf[RED][i]=maurom[RED][i];
        rambuf[GREEN][i]=maurom[GREEN][i];
    }
}
```

Bước 9. Thêm module Timer sử dụng Timer0 10 ms. Trong **timer_process()** thực hiện quy trình quét LED theo chỉ số **rowidx**.

```
void timer_process()
{   row=0;                          //bo chon cac hang
    gcol=rambuf[GREEN][rowidx]; //xuat du lieu xanh
    rcol=rambuf[RED][rowidx];   //xuat du lieu do
    row=1<<rowidx++;            //chon hang theo rowidx
    rowidx%=ROWMAX;             //xoay vong rowidx
}
```

Bước 10. Chạy chương trình và quan sát.

Bước 11. Chỉnh lại thời gian quét để ma trận LED không bị chớp.

## 5.4 Thực hiện yêu cầu 2: hiển thị số ra LED 7 đoạn.

Bước 12. Kết nối lại dây cáp như sau:

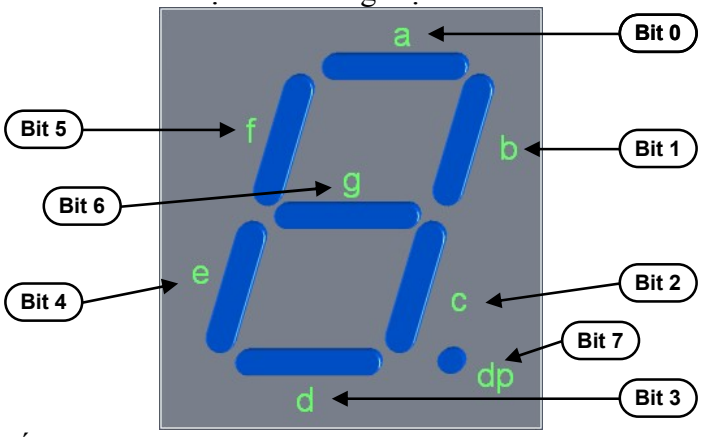
- PortB : LED_EN (SELECT)
- PortD : RED (SMENT)

Bước 13. Tạo file mới Tn05_led7doan.c.

Bước 14. Dữ liệu xuất ra 8 LED 7 đoạn gồm 8 byte (chứa số từ 0 đến 9).

Bước 15. Tuy nhiên, do đặc thù của LED 7 đoạn là các nét dữ liệu (a-g) và chấm dp nên để hiện được số từ 0 đến 9, dữ liệu phải qua bước giải mã 7 đoạn như sau:

Kết nối nét và bit dữ liệu theo trong mạch như hình sau:




- Sáng số 0: các nét a-b-c-d-e-f = 1, các g-dp = 0, dữ liệu sẽ là 0b00111111=0x3F.
- Sáng số 1: dữ liệu là 0x06.
- . . .

```
const unsigned char dl7doan[MAX7S]={
        0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,
        0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71};
unsigned char dbuf[SELMAX],selidx;
unsigned char gm_7doan(unsigned char so)
{
    if (so>0x1F) return 0;
    if (so<0x10) return dl7doan[so];
    return dl7doan[so&0x0F]|0x80;
}
```

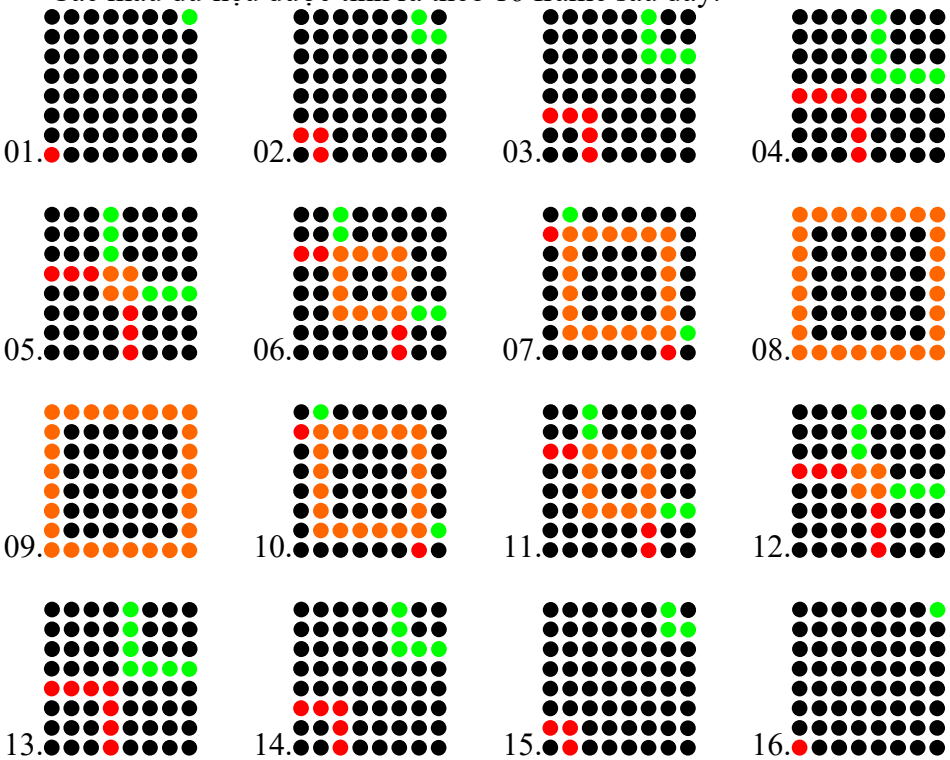
Ví dụ: Để xuất số 7 có chấm dp, ta dùng **`gm_7doan(0x17)`**.

Bước 16. Kết hợp Timer0 10 ms để hiển thị số trên cả 8 LED. Hàm **`timer_process()`** như sau:

```
void timer_process()
{   select=0;
    dcol=gm_7doan(dbuf[selidx]);
    select=1<<selidx++;
    selidx%=SELMAX;
}
```

## 5.5 Bài tập

a) Viết chương trình hiển thị hình thay đổi theo quy luật lên ma trận LED, 100ms thay đổi 1 lần. (Gợi ý: tạo nhiều frame hình, 100ms đổi frame một lần bằng cách chép frame (8 byte xanh, 8 byte đỏ) từ dãy dữ liệu ROM sang dãy dữ liệu RAM).
Các mẫu dữ liệu được tính ra theo 16 frame sau đây:




b) Viết hàm **`unsigned char mirror(unsigned char d)`** thực hiện lật ngược thứ tự bit, đổi giá trị các cặp bit (b7-b0), (b6-b1), (b5-b2), (b4-b3). Áp dụng vào bài tập a) sau mỗi lần nhấn nút RA5.
c) Viết dự án thực hiện số chạy (3 số) từ trái sang phải màn hình LED 7 đoạn.
d) Viết chương trình hiện đồng hồ (hh.mm.ss) lên LED 7 đoạn.