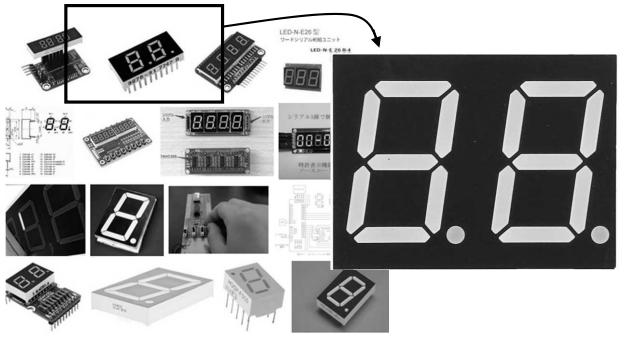
組み込み基礎 4

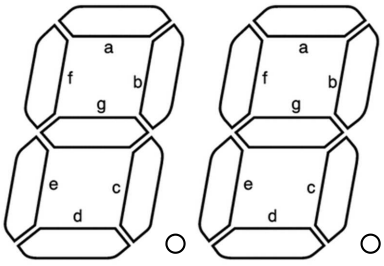
～７セグ表示 2桁で！

Ei2 ハードウエア技術

７セグメントLED （７セグ）のいろいろ

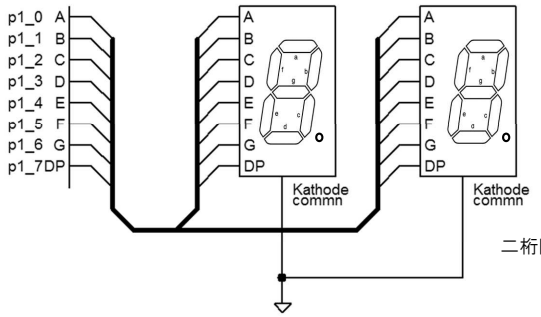


７セグメントLED （７セグ）



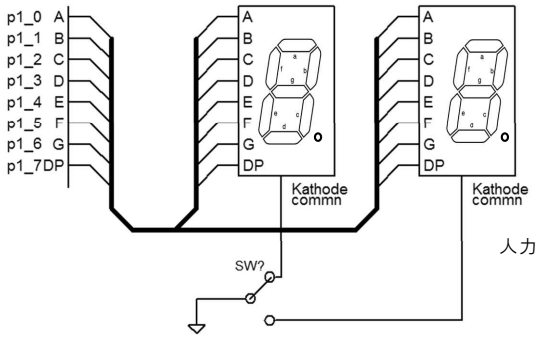
２桁全セグメントをすべて制御するには16本の信号線が必要になる。
３桁４桁・・・ｎ桁では
ｎｘ8本の信号線が必要で、必要な本数だけ準備する方式を“スタティック制御（表示）”と言う。
シフトレジスタ等で制御線の拡張が必要になる。

並列に繋いでみた



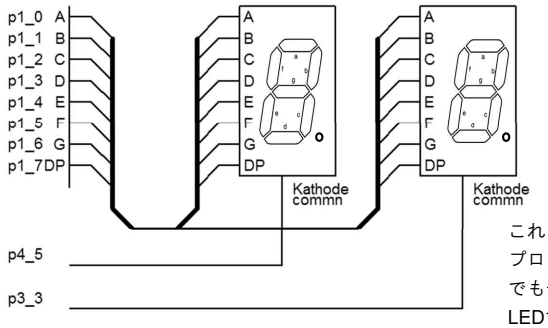
二桁同じ文字がでる

表示を切り替えられる



人力でやるの？

ダイナミック制御を行う。



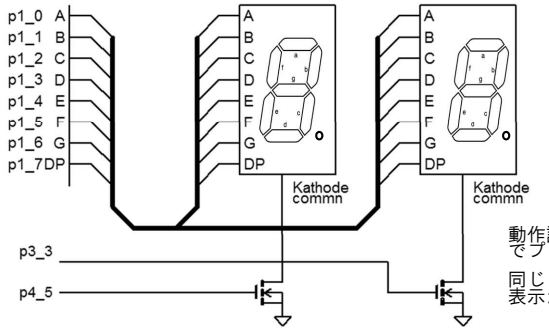
これなら
プログラムできる。
でも普通のポートでは
LED1個がやっと

LEDを表示してみよう　無理を承知で

2桁で！

```
void Led_dsp( dat ){
    if( p4_5 == 0 ){            //  2桁目が点灯中か
        p4_5= 1 ;               //  2桁目を消灯
        p1=seg[ dat % 10];      //  セグメントデータをセット
        p3_3=0;                 //  1桁目を点灯
    }else{
        p3_3= 1 ;               //  1桁目を消灯
        p1=seg[ dat / 10];      //  セグメントデータをセット
        p4_5= 0 ;               //  2桁目を点灯
    }
}
```


ダイナミック制御を行う。　　電流容量を考えた回路



動作論理が逆になるのでプログラム注意。

同じプログラムだと桁表示が逆になります

LEDを表示してみよう

2桁で！

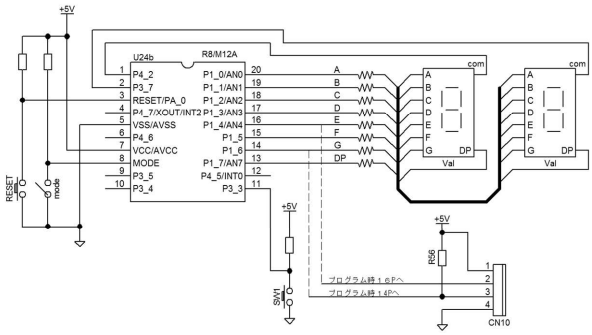
```
void Led_dsp( dat ){
    if( p4_5 ==  1 ){           //  2桁目が点灯中か
        p4_5=0;                 //  2桁目を消灯
        p1=seg[ dat % 10];      //  セグメントデータをセット
        p3_3= 1 ;               //  1桁目を点灯
    }else{
        p3_3=0;                 //  1桁目を消灯
        p1=seg[ dat  / 10];     //  セグメントデータをセット
        p4_5=1;                 //  2桁目を点灯
    }
}
```

※制御ビットを反転する

ダイナミック点灯の注意点

- 時分割で表示するので表示の明るさがN桁分暗くなる。
  - ➢LEDの限界まで明るくする
    （パルス駆動は5～10倍ぐらいにできる）
- 桁の表示中に数値を変えると文字が重なったり、ちらつく。
  - ➢セグメントのデータ変更は表示を消して行う。
- 各桁の表示時間に比例して表示の明るさが変わってしまう。
  - ➢割り込み処理で行うのが一般的。

以下の回路で2桁のカウンタを作ろう（回路が変わったので注意）



回路から入出力の信号線と信号の方向（I/O方向）を書き出す

- 7セグメントLED
  - バイトアクセス _____　segA:_____　DP : _____
  - com1:_____　com2:_____
  - ﾎﾟｰﾄ初期値 _____　設定ﾚｼﾞｽﾀ_____　ｺﾝﾄﾛｰﾙﾜｰﾄﾞ_____
  - com1・ com2 ：(p4)　設定ﾚｼﾞｽﾀ_____　ｺﾝﾄﾛｰﾙﾜｰﾄﾞ_____
- SW1
  - ビットアクセス _____　動作論理 _____
  - 設定ﾚｼﾞｽﾀとｺﾝﾄﾛｰﾙﾜｰﾄﾞ_____　_____

回路から入出力の信号線と信号の方向（I/O方向）を書き出す

- 7セグメントLED．
- ．
  - バイトアクセス　p１　segA:　p1_0　　DP：p１_7
  - com1:　p4_2 com2:　p3_7
  -
  - P1ﾎﾟｰﾄ初期値 seg[0]　設定ﾚｼﾞｽﾀ　pd1　ｺﾝﾄﾛｰﾙﾜｰﾄﾞ 0x00　.
  - com１ビットの設定　pd4_2　=　1;
  - com２ビットの設定　pd3_7　=　1;
- SW1
  - ビットアクセス　＿＿＿＿　動作論理　＿＿＿＿
  - 設定ﾚｼﾞｽﾀとｺﾝﾄﾛｰﾙﾜｰﾄﾞ＿＿＿＿　＿＿＿＿

SW1 ビットの設定　pd3_3 = 0;　でもOK

define　定義　（便利に使おう）

```
#define  SW1      p3_3    //スイッチ定義
#define  SW_ON    0       //スイッチのON論理
#define  SW_OFF   1       //スイッチのOFF論理
#define  COM1     p3_7    //1の位　コモン出力制御信号
#define  COM2     p4_2    //１０の位　コモン出力制御信号
#define  COM_ON   1       //com１制御線のON論理
#define  COM_OFF  0       //com2 制御線のOFF論理
```

セグメント表示　ビットパターン設定

- const char seg[]　={63, 6 ,91, 79 ,102 ,109, 125, 39, 127 ,111}

カウンタ値の各1桁のデータを抽出する

- 1の位：　i%10
- 10の位：　(i/10)%10

SW1が押された回数をカウントしてみよう

2桁で！

```
void main(){
    int i=0;
    p1=seg[i];
    while(1){

        if( sw1== SW_ON ){
              timer(5);
              i++;
        while(sw1== SW_ON );
        }
         Led_dsp( i );
    }
}
```

```
void Led_dsp( char dat ){

      if( COM１ == COM_ON ){
            COM1= COM_OFF;
            p1=seg[ (dat / 10)%10 ];
            COM2= COM_ON;
      }else{
            COM2 = COM_OFF;
            p1=seg[ dat % 10];
            COM1= COM_ ON;
      }
}
```

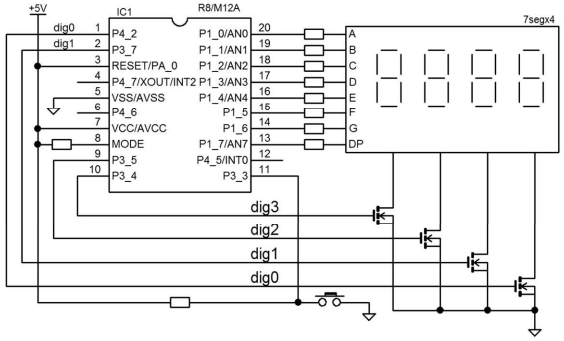

多桁の表示！

組み込み基礎 5

～ 3桁以上のカウンタ

Ei2　ハードウエア技術

以下の回路で4桁のカウンタを作ろう



ほんとに多桁

SW1が押された回数をカウントしてみよう

```
void main(){
    int i=0;
    p1=seg[i];
    while(1){

      if( sw1==sw_ON ){
            times_dat ++;
            timer(5);
        while( sw1==sw_ON);
        }

    }
}
```

```
#pragma interrupt intTRB (vect=24)
void   intTRB( void ){
```

times_dat を表示する
処理が入る

今回は　disp_seg(桁, 表示データ);

説明の都合上コードは後ほど

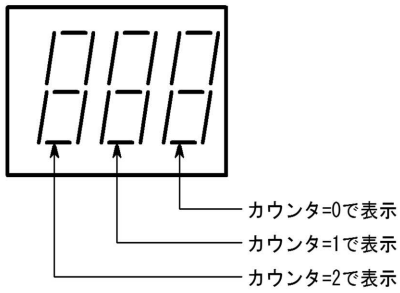
SW1が押された回数をカウントしてみよう

3桁以上で

メインルーチンは同じで良い

考え方
桁カウンタseg_cntを用意する
（変数のタイプは---？）
グローバル？ スタティック？
割り込みのたびにカウントアップ
（ seg_cnt++を実行 ）

対応する桁の表示をする
桁数を超えたら０に戻す



SW1が押された回数をカウントしてみよう

3桁以上で

メインルーチンは同じで良い

考え方
カウンタ用変数を用意する
割り込みのたびにカウントアップ
★ 対応する桁の表示をする
桁数を超えたら０に戻す

サブルーチンを作ってみよう
プロトタイプは
void disp_seg(char keta, int num); で
引数numのketa変数の桁を表示
ただし最下位を０とする。

```
#pragma interrupt intTRB (vect=24)
void   intTRB( void ){
  static char seg_cnt=0;

  disp_seg(seg_cnt, times_dat );
  seg_cnt++;
  If( seg_cnt  > 3 ) seg_cnt =0;

}
```


サブルーチンを作ってみよう

```
void   disp_seg( char   keta , int  num ){
  dig0 = dig1 =  dig2 =  dig3 = OFF;
   switch(keta){
       case 0: //1桁目の点灯処理
           p1 = seg[num%10];
           dig0 = ON;
         break;
       case 1: //2桁目の点灯処理
           p1 = seg[(num/10)%10];
           dig1 = ON;
          break;
       case 2: //3桁目の点灯処理
           p1 = seg[(num/100)%10];
           dig2 = ON;
         break;
```

disp_seg(char keta, int num);
引数numのketa変数で示す桁を表示

```
       case 3: //4桁目の点灯処理
           p1 = seg[(num/1000)%10];
           dig3 = ON;
         break;
       }
}
```


SW1が押された回数をカウントしてみよう

ほんとに多桁

```
void main(){
    int i=0;
    p1=seg[i];
    while(1){
        if( sw1==sw_ON ){
            times_dat ++;
            timer(5);
        while( sw1==sw_ON);
        }

    }
}
```

```
#pragma interrupt intTRB (vect=24)
void   intTRB( void ){

  static char seg_cnt=0;

  disp_seg(seg_cnt, times_dat );
  seg_cnt++;
  If( seg_cnt  > 3 ) seg_cnt =0;

}
```