アナログ信号を手軽に扱うための

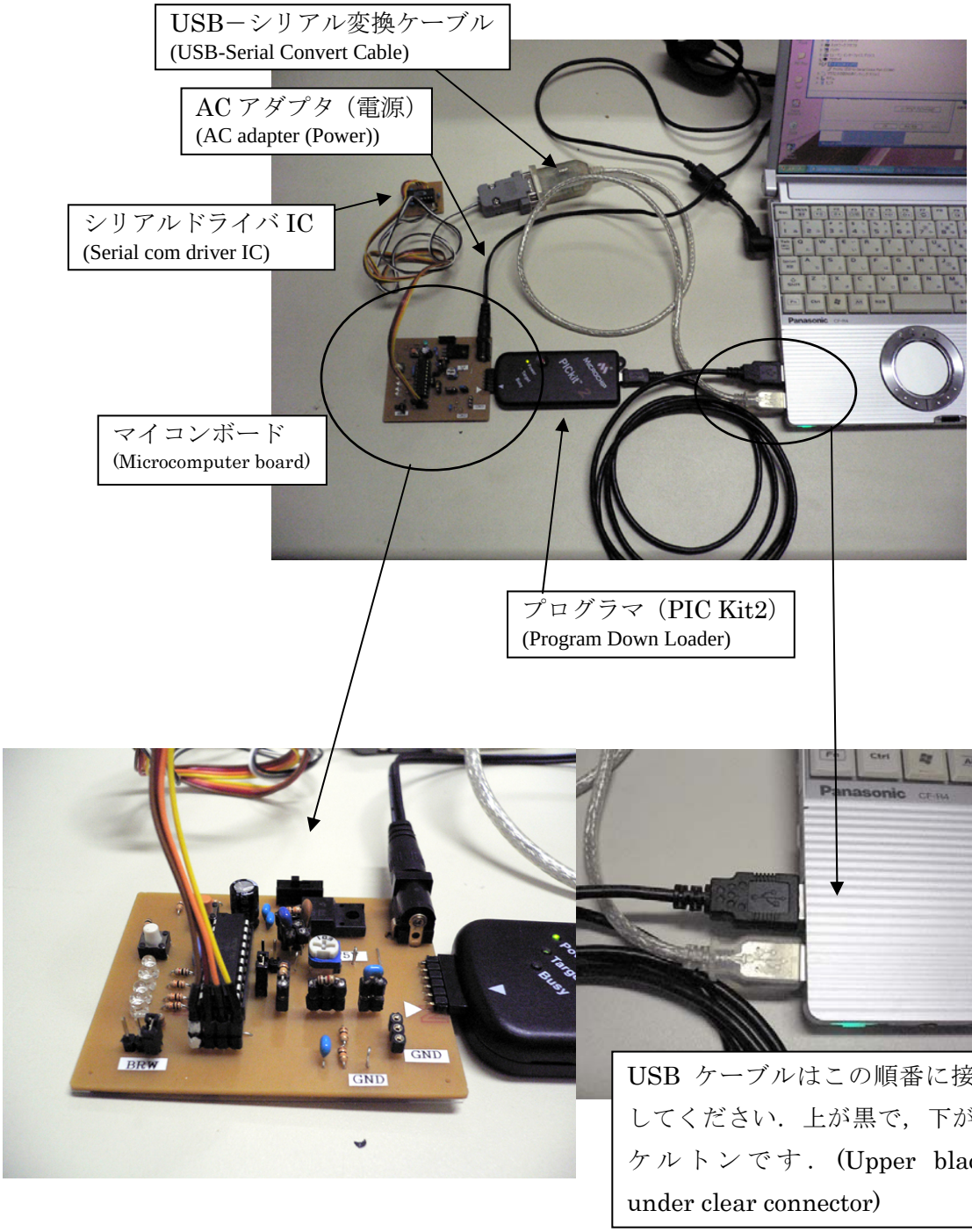
PIC　16F785
学習テキスト

PIC16F785　Learning Guide
for using analog information

森下 武志

マイコン開発環境の基本セットと接続方法
(Microcomputer development Basic Set & connecting)

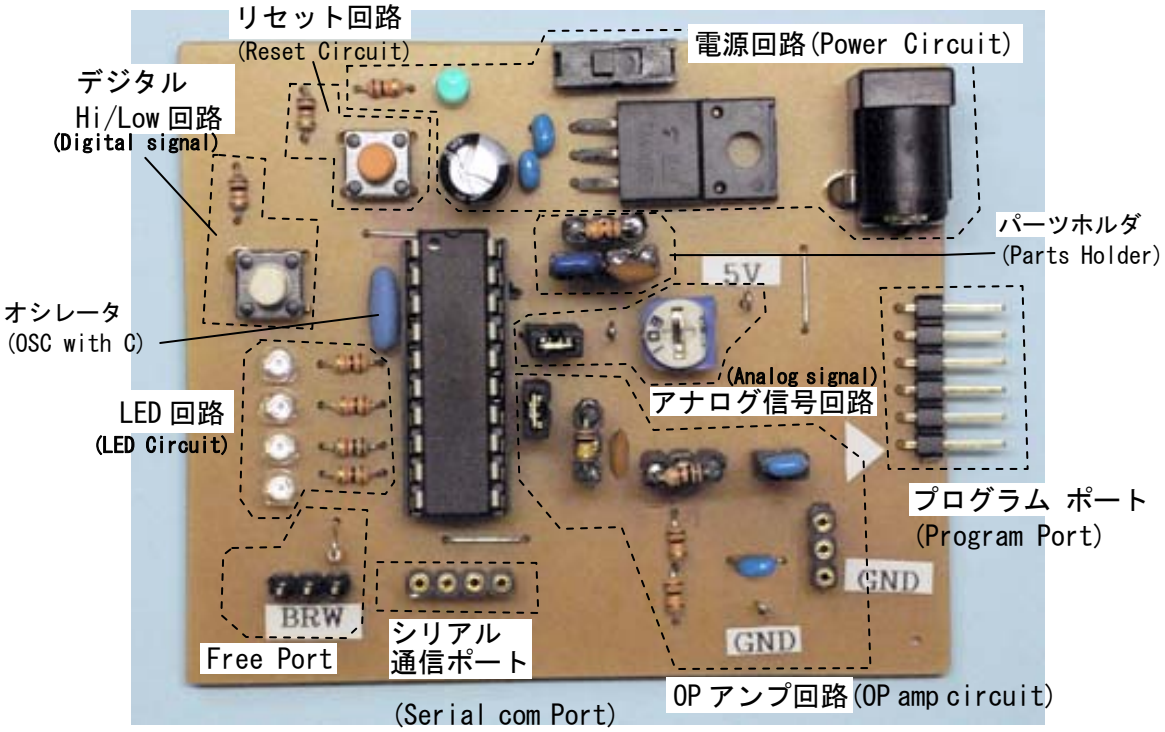
まずは、これらがあれば始められます。図のように接続してください。

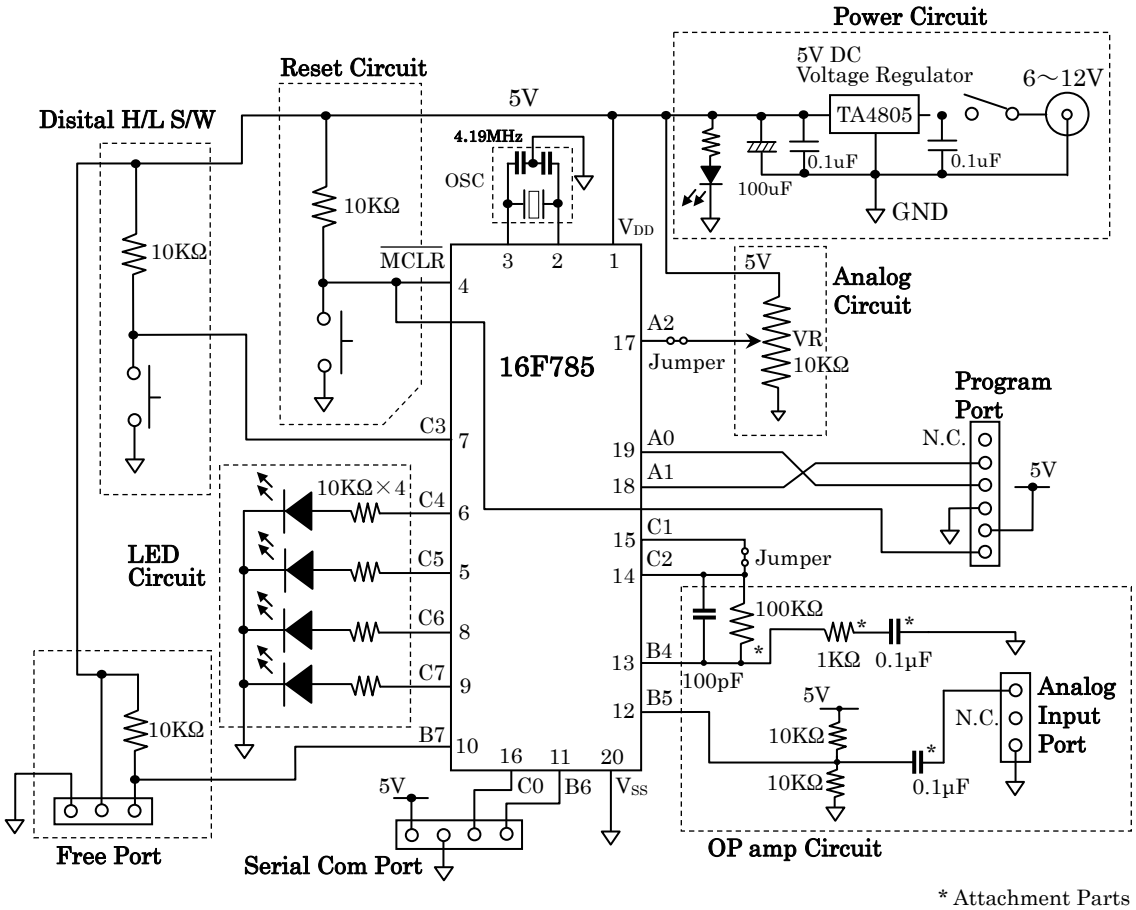


USB ケーブルはこの順番に接続してください．上が黒で，下がスケルトンです．(Upper black, under clear connector)

PIC16F785 学習ボード(PIC16F785 Learning Board)

この学習ボードは、下図に示す各回路が組み合わされた回路となっています。学習を進める中で、注目する回路はどこで、どのような接続になっているかなど、基本回路のハードウエアを意識して学習すればプログラミング技術の理解が深まります。



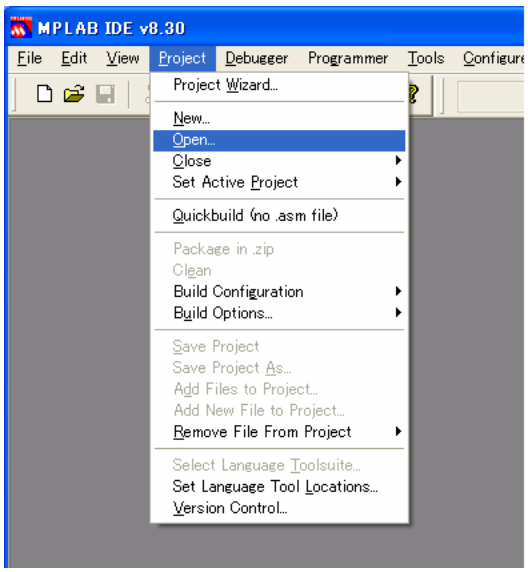


* Attachment Parts

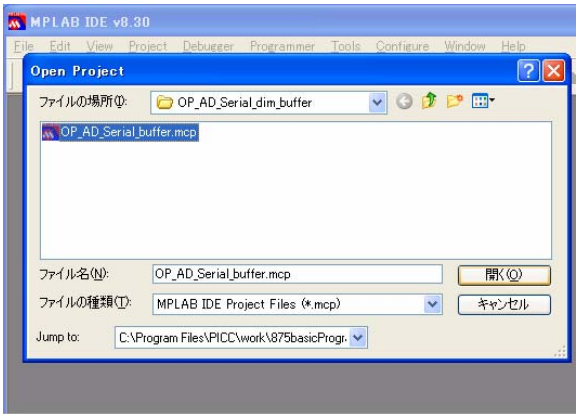
STEP１　プログラム開発 TOOL の基本操作と動作確認（とにかく使ってみる）

(STEP1 Basic Operations of Development TOOL & Operation Check)

No1　このアイコンから開く



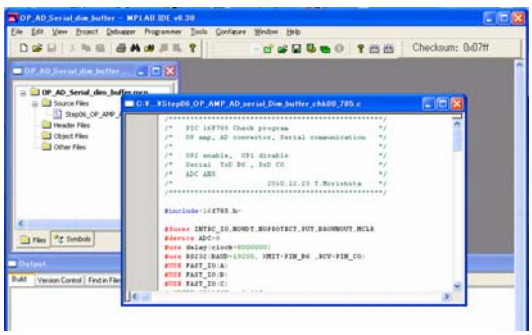
No２　Project → **Open**　します。



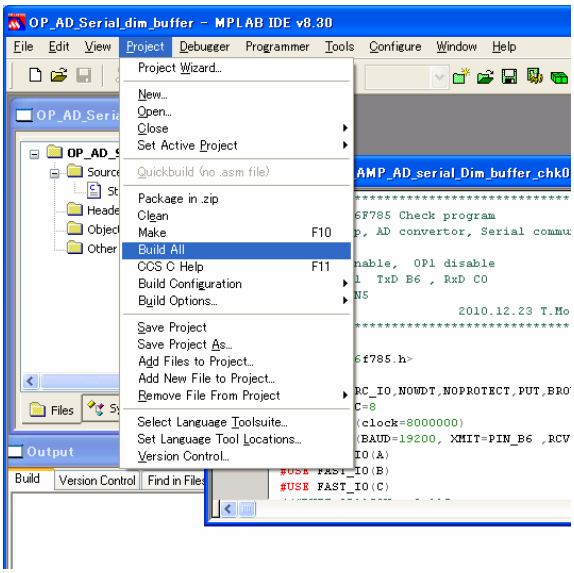
No３　はじめは,

C:¥ProgramFiles¥PICC¥work¥16F785¥875basicProg ram¥STEP01 output¥out00

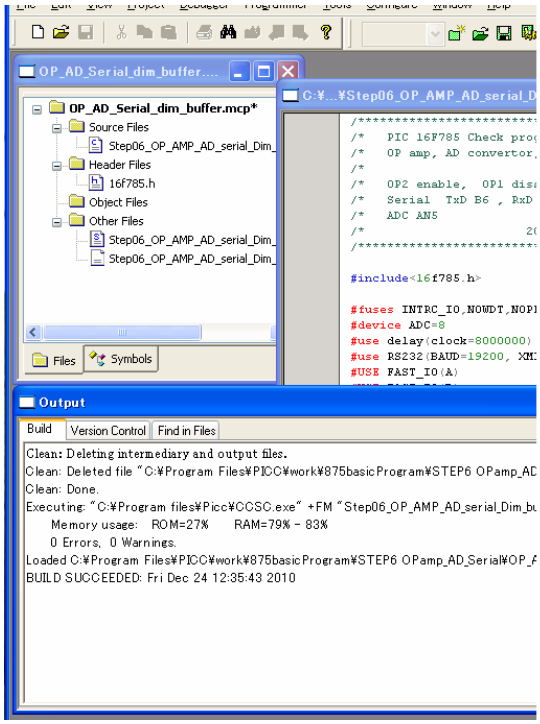
からはじめてください。



No４　このエディタでプログラムの編集などします。今回はサンプルプログラムをそのまま**変更しない**で、試してください。(Without change)

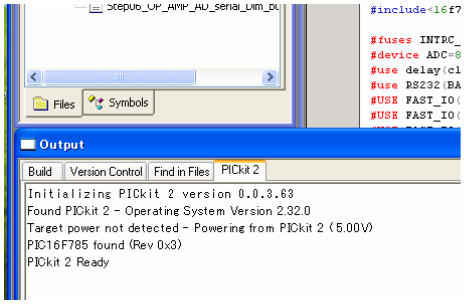


No５　Project → **build All** でプログラムをコンパイルする。



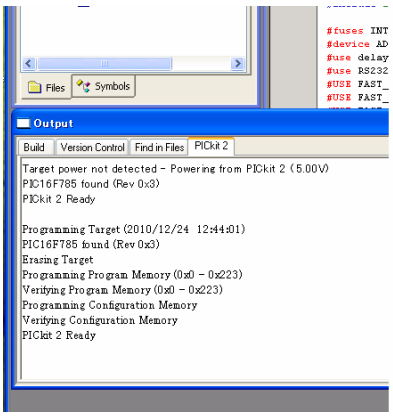
No 6　Build Succeeded が表示されれば、エラーはありません。

（今は、Warning を気にしない）

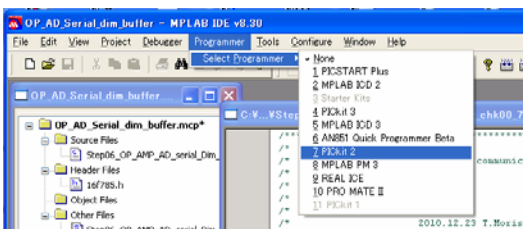


No 8　PICKit2 Ready が表示される。

（接続にエラーはない）



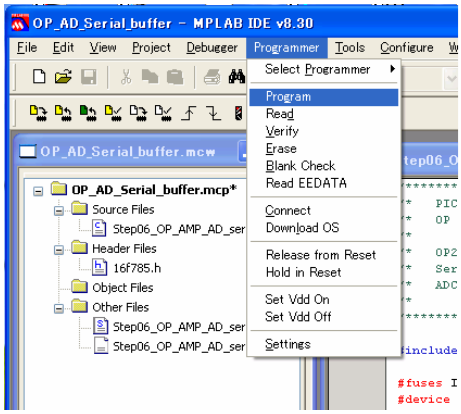
No10　エラーがなければ PICKit2 Ready が表示される.



No 7　Programmer →

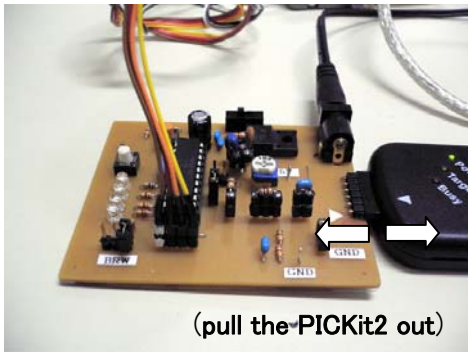
Select Programmer →**PICKit2** を選ぶ.

2 回目以降、この操作は必要ない.



No 9　Programmer → **Program**

プログラムをデバイスにプログラムする。



No11　マイコンボードから **PICKit2 をはずします。**すると、マイコンが動作します。LED が点灯すれば OK です.

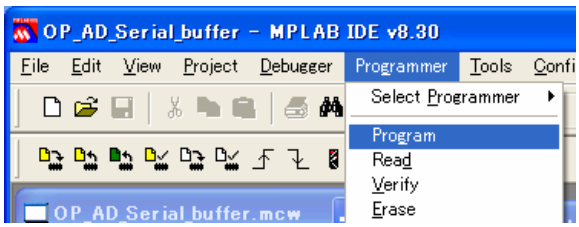
【STEP1-0】マイコンの動作確認（信号を出力する）（ファイル名：Step01 out01_785.c）

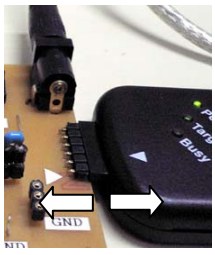
まず、**MPLAB** から **STEP01→out00→Step01 out00_785** のプロジェクトを開く。

```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)          //internal oscillator clock setting
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

main()
{
      set_tris_b(0x00);          // B port bit in/out setting, All output: 0=output, 1=input
      set_tris_c(0x00);          // C port bit in/out setting, All output

      do{                         // endless loop

            output_low(PIN_C7);      // C port-C6 bit output Low level
            delay_ms(100);           // delay time (msec)
            output_high(PIN_C7);     // C port-C6 bit output Hi level
            delay_ms(100);

       }while(1);


}
```

まずは、上記のプログラムを実行してみましょう。

Programmer → **Program** で上記のプログラムをマイコンに書き込み、PICKIT2 をマイコンボードから外すと LED が点滅していれば動作の確認は終了です。





(pull the PICKit2 out)

プログラムの 1 行目で、使用するデバイスを指定して、2 行目でデバイスのコンフィグレーションを設定しています。今はこの設定をあまり気にしないで先に進みましょう。

設定内容は、

INTRC_IO：内部オシレータの利用

NOWDT：ウオッチドックタイマ無し（正常動作かどうかを監視する設定。通常使用しない）

NOPROTECT：プログラムのプロテクト無し（書き込み、読み出しが自由）

PUT：パワーアップタイマ利用（回路全体の通電目的で、電源 ON 後少し時間をおいて実行）

BROWNOUT：電源電圧の低下で自動リセットする機能（電池の場合 NOBROWNOUT が良い）

MCLR：リセット端子をリセットとして利用（IO として使う場合は NOMCLR）

【STEP1－1】 LED 点滅プログラムのバリエーション（ファイル名：Step01 out01_785.c）

ここでは、output_BIT(PIN_C6,led);　を使って信号を出力します。

```
#include<16f785.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)          //internal oscillator clock setting
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

main()
{

      int led=0;

      set_tris_b(0x00);              // B port bit in/out setting, 0=output, 1=input
      set_tris_c(0x00);              // C port bit in/out setting

      do{                                  // endless loop

            delay_ms(100);
            output_BIT(PIN_C6,led);
            led=led^64;                    // Hi-Low bit inversion

       }while(1);

}
```

output_BIT(PIN_C6,led);では出力先のピンを C６ピンと指定して、変数 led の値を出力に利用するプログラムに変更しています。

また、led=led^64;　で、式中の ^　が XOR の論理演算を行わせるもので、出力ビットの値を１と０　を反転させています。ちょっとした工夫をしています。

ここまででこの課題のポイントは終了ですが、
#use delay(clock=8000000)　は内部オシレータのクロック周波数を設定しています。
内部オシレータは、８M、４M、２M、１M、500K、250K、125K、31K　各 Hz の８種類が(clock=****000)と設定できます。

また、#USE FAST_IO(port)と#USE STANDARD_IO(port)が代表的です。
CCS コンパイラが入出力ピンを切り替えるたびに「入力」「出力」を設定するように作られていますが、
#USE FAST_IO(port)：　はじめの一度設定するだけにして、この設定時間を短縮させる。
#USE STANDARD_IO(port)：io の入出力の設定をその都度設定し、処理速度より信頼性を優先するプログラミング。

【STEP1－2】 LED 点滅プログラムのバリエーション２（ファイル名：Step02 out02_785.c）

出力関数　output_c(led);　を使って出力する（byte 単位で入出力を実行する）

```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)          //internal oscillator clock setting
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

main()
{

      int led=0;

      set_tris_b(0x00);           // B port bit in/out setting, 0=output, 1=input
      set_tris_c(0x00);           // C port bit in/out setting

      while(1){                   // endless loop

            delay_ms(100);
            output_c(led);       //C port-all bit output
                                  //but C0,C1,C2,C3-> LED No connection
            led=led^240;         //bit inversion: 240(decimal number)
                                 // -> 11110000(binary number)
                                 // ->C7=1,C6=1,c5=1,C4=1,C3=0,c2=0,c1=0,c0=0
       }
}
```

Byte 単位で入出力をできていることを意識してプログラムと動作を確認する。

【STEP1－3】 LED 点滅プログラムのバリエーション３（ファイル名：Step03 out02_785.c）

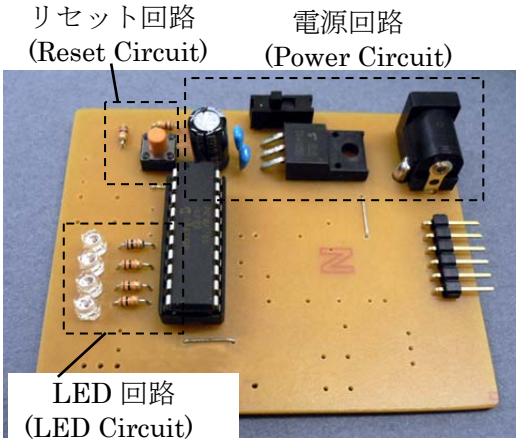
```
  ～ プリプロセッサ等はこれまでと同じため省略 ～
main()
{
      int i,led=0;

      set_tris_b(0x00);              // B port bit in/out setting, 0=output, 1=input
      set_tris_c(0x0F);              // C port bit in/out setting

      while(1){                      // endless loop
                  led=16;
                  for(i=0;i<4;i++){
                  delay_ms(100);
                  output_c(led);        // C port-all bit output(but C0,C1,C2,C3-> No LED)
                  led=led<<1;
            }
       }
}
```

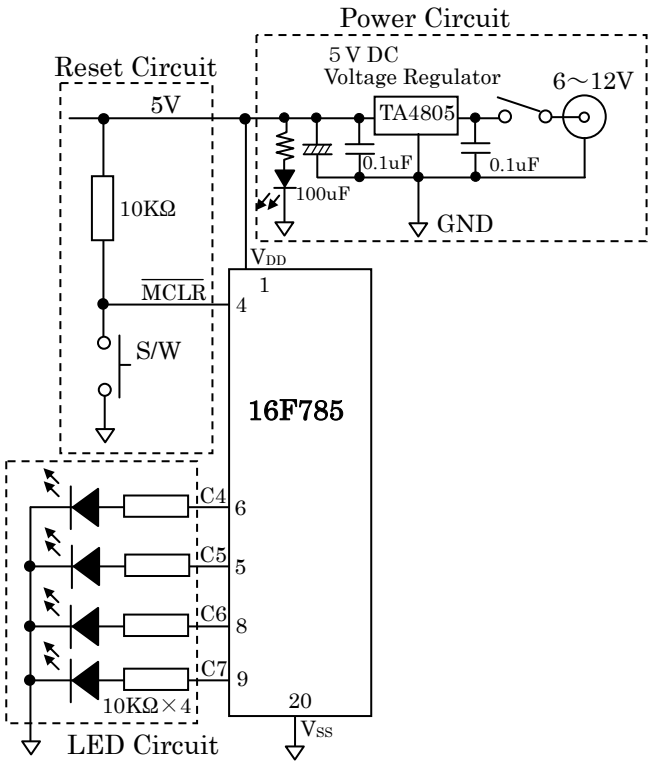
For 文の利用で、**LED の光が流れる**ように工夫したプログラミング。

LED の点灯場所一つずつしシフトさせるため、led<<1;　で点灯のシフトを実現している。

STEP１で使った回路でハードウエアを理解する

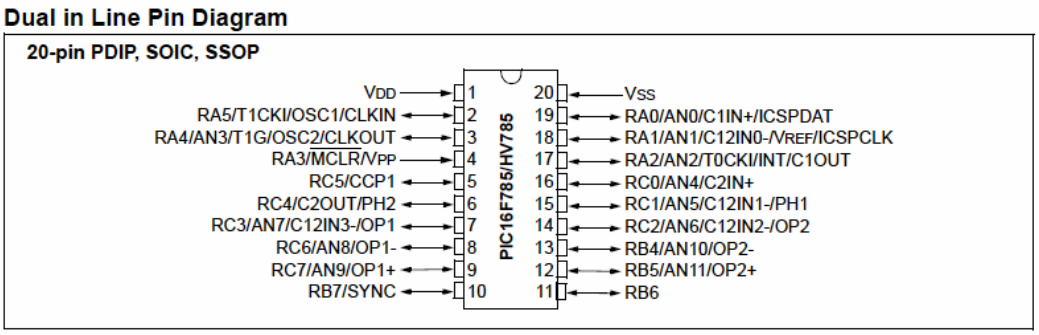
組み込みマイコン開発では、ハードウエアとソフトウエアの関連を理解する必要があります。ここで使ったハードウエアの回路を理解し、よりマイコン技術の理解を深めましょう。





マイコンや半導体を利用して開発する時は、それぞれのデバイスのデータシートを Web などから手に入れてください。

開発には PIC16F785 の Data Sheet が必ず必要になります。下図は抜粋資料です。この図をみることでデバイスのそれぞれの端子の役割がわかります。（詳細は Page6, 7）



【課題 1-１】

リセット回路(Reset)について、ハードウエアマニュアルの 110 頁(page)と比べて理解してください。実際の回路ではコンデンサは省略してあります。

【課題 1-２】

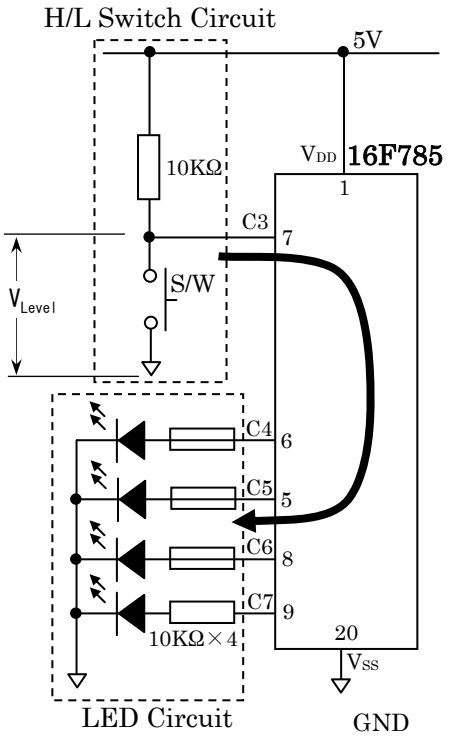
また、回路図と実際の基板のパターンを見比べて、回路図との対応を確認してください。

【課題 1-３】

電源回路の回路図と実際の回路との対応を確認して、回路を理解してください。

STEP２　信号の INPUT と OUTPUT を実行する(INPUT/OUTPUT Operating)

STEP2 では、High(５V)と Low(０V)の信号をマイコンに入力して、その信号を使って出力信号（LED ドライブ）を制御する練習を行います。
この回路では、スイッチ回路の信号線が入力情報として C port の 3 番(7 番ピン)に接続されています。LED はSTEP1と同じで、C portの４番から７番で、デバイスのピン番号では６番から９番に接続されています。



入力情報となる電圧レベル(H/L)は、
S/W を離している時は、５V ラインそのものなので
→　High Level
S/W を押している時は、GND とショートするので
→　Low Level
となります。なお、この回路はリセット回路で用いた回路と全く同じです。

それでは演習を行います。ここではデジタル情報の入出力と、その情報の扱い方や基本的なテクニックを学習します。
ここで実行するプログラムは、**Step02io00_785.c** というファイルで、保存先は
ProgramFiles¥PICC¥work¥PIC16F785¥785basicProgram¥STEP02input_output¥io00¥Step02 io00_785.c です。

このプログラムで学習する STEP02－１でのポイントは、Byte 入出力関数の扱いで、

C ポートの入出力設定　：　set_tris_c(0x09);　C ポート 1,4 ビットは入力に設定し、それ以外を出力ビットに設定する関数
Byte 入出力関数の確認　：　led=input_c();　C ポートのデータを変数 led に代入
入力ビットの反転　：　led=led ^ 8;　4 ビット目を XOR で反転させる
入力ビットのマスキング　：　4 ビット目だけ有効ビットとして抜き出して使用する

などの関数に慣れることです。

STEP2-2 での学習のポイントは、bit 単位の情報を入出力するプログラムに慣れることです。

【課題 2-1】

Main プログラム内の各１行の意味を理解してください。

【課題 2-2】

Bit と Byte 入力と出力の関数の違いを理解してください。

【STEP2-1】デジタル情報の入出力を実行する（**Step02io00_785.c**）

```
#include<16f785.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)          //internal oscillator clock setting
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

main()
{

      int led;

      //set_tris_A(0xFF);           // A port bit in/out setting, 0=output, 1=input
      //set_tris_b(0x00);           // B port bit in/out setting, 0=output, 1=input
      set_tris_c(0x09);             // C port bit in/out setting 1,4bit input, other output

      output_c(0);

      do{                                 // endless loop

            led=input_c();              // C port Byte Input
            led=led ^ 8;                // bit inversion
            led=led & 8;                // bit maskking

            if(led==8){
                  output_c(240);    // C port Byte Output
            }else{
                  output_c(0);      // C port Byte Output
            }
       }while(1);
}
```

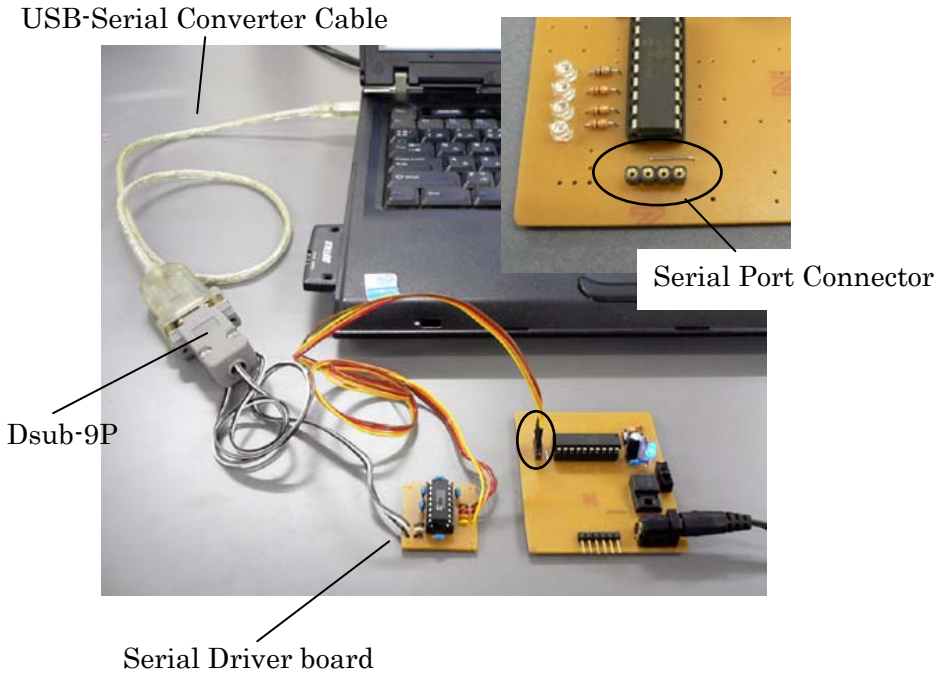
【STEP2-1】デジタル情報の入出力をバリエーション（**Step02io01_785.c**）

ポートとビットを指定した bit 単位で扱う関数を扱ってみる。

```
   ～ プリプロセッサ等は上記と同じため省略 ～
main()
{
      int led=0;
      //set_tris_A(0xFF);           // A port bit in/out setting, 0=output, 1=input
      //set_tris_b(0x00);           // B port bit in/out setting, 0=output, 1=input
      set_tris_c(0x09);             // C port bit in/out setting 1,4bit input, other output

      output_c(0);

      do{                                 // endless loop
            output_BIT(PIN_C5,!input(PIN_C3));     //output_BIT
       }while(1);
}
```

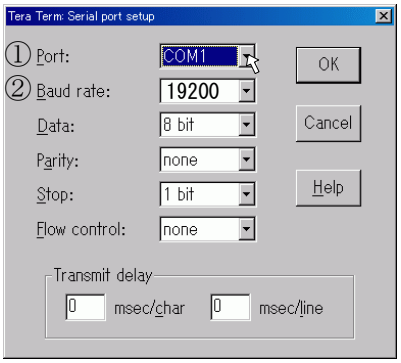
STEP３　パソコンとマイコンでシリアル通信させる(Serial Communications)

この STEP では、マイコンとパソコンをワイヤーケーブルで接続して、情報のやり取りを実現させます。下図のように、**USB－Serial Convert Cable と Serial Driver Board をマイコンボードの Serial Port にコネクト**してください。接続ができたら、サンプルプログラム(**Step03_serial00_785.c**)を実行して、シリアル通信に必要なハードウエアの構成と基本的な情報処理方法を学んでください。



パソコン側(PC side)のターミナルソフト(TeraTerm)を起動する

TeraTerm をこの  アイコンで起動します。
次にシリアル通信の各設定とターミナルソフトの設定との対応を確認します。
メニューから「**Setup**」→「**Serial port...**」（右図）。



①、Port は、USB-Serial Cable をパソコンに接続し、パソコンが割り振った COM 番号。
スタート→コントロールパネル→システム→ハードウエア→デバイスマネージャ→COM ポートで確認できます。
この場合は、**①を COM3** に合わせます

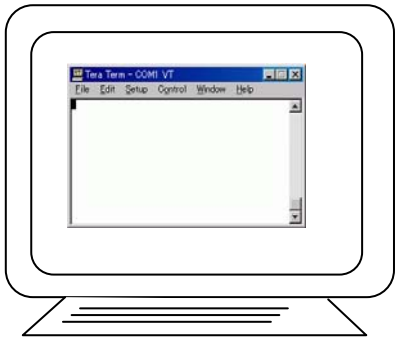
②、Baud rate は、次ページのプログラムの通信レートにあわせます。プログラムより **19200 bps** となります。

プログラミングと実行(Down Load of serial com program & Run)

MPLAB を使って、以下のプログラムを**マイコンにプログラミング(program)**します。成功したら、**PICKit2 をマイコンボードから抜いて(pull the PICKit2 out)**，TeraTarm のウインドウをクリック（アクティブ）**(TeraTerm Window Click)**して、キーボードのキーを何かタイプ**(Hit any Key of PC Keyboard)**してください。

Step03_serial00_785.c

```
#include<16f785.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)
#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)
// RS232(baud rate, transmission Pin No. , receiving Pin no.)
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

main()
{
        int cmnd,i;
        while(1){
                printf("Hit any Key ");
                cmnd=getc();
                i=10;
                printf("Input= %d¥r¥n",i);
                //putc(cmnd);
        }
}
```



このプログラム中、マイコンでシリアル通信をさせるための関数が、2 行目の

#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)

です。通信レートは　19200bls、送信ポートが PIN_B6 と指定し、受信ポートが RCV=PIN_C0 と設定していることになります。また、この C コンパイラは、シリアル通信用のモジュールが無くとも、ソフト的にシリアル通信を実行できるようにコンパイルしてくれます。

【課題 3-1】

シリアル通信を行うための接続を、回路図とハードウエアの対応を確認してください。

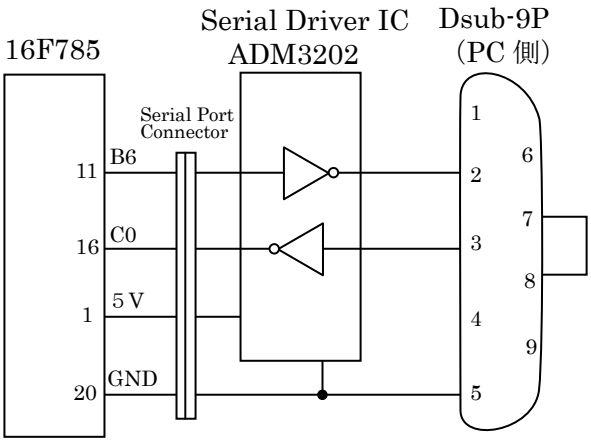
【課題 3-2】

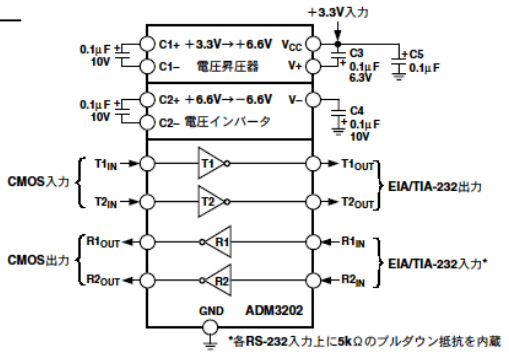
PIC 用 C コンパイラの Prontf コマンドの引数や使い方を確認してください。

【課題 3-3】

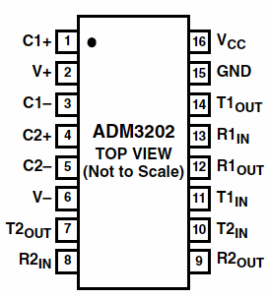
プログラム中の、#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)
BAUD を 9600 とし、TeraTarm の通信設定を変えてシリアル通信を確認してください。

シリアル通信をするための接続ハードウエア
（CONNECTIONS for Serial Communication）





ADM3202 FUNCTIONAL BLOCK DIAGRAMS



PIN CONNECTIONS DIP

【STEP3-1】ASCII コードの読み取りと、送信を実現する

このプログラムは、パソコン側でタイプした任意のキーのアスキーコードを読み取り、その値を再びパソコン側に送信します。つまり、シリアル通信の動作チェックです。

Step03_serial01_ascii_785.c

```
#include<16f785.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)
#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)
// RS232(baud rate, transmission Pin No. , receiving Pin no.)
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

main(void)
{
        char cmd;
        for(;;){
                printf("Hit any Key ¥r¥n");
                cmd=getc();
                printf("Key = %c¥r¥n",cmd);
                printf("Ascii = %x¥r¥n",cmd);
        }
}
```

【STEP3-２】ASCII コードの上位 bit を LED にバイナリで表示させる。

Step03_serial01_binary_785.c

```
#include<16f785.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)
#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)
// RS232(baud rate, transmission Pin No. , receiving Pin no.)8#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

main(void)
{
        int cmd;

    set_tris_c(0x01);       // C port bit in/out setting, All output

        for(;;){
                printf("Hit any Number Key ¥r¥n");
                cmd=getc();
                printf("Key = %c¥r¥n",cmd);
                printf("Ascii = %x¥r¥n",cmd);
                //cmd=35;
                cmd=cmd<<4;     // For 4LED, to shift upper bit
                output_c(cmd);
        }
}
```

Printf をより機能的に使うためのエスケープシーケンスコード

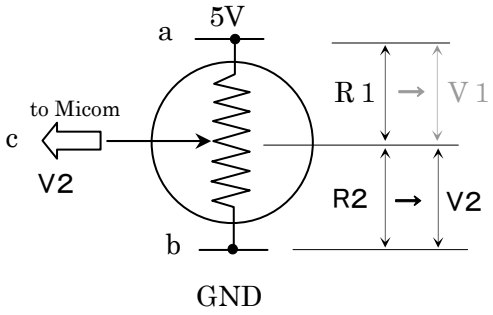
¥r：復帰(return code)
¥n：改行(linefeed code)
¥b：バックスペース(backspace)
¥t：水平タブ(tab)
¥v：垂直タブ(vertical tab)

%C：文字
%S：文字列か文字
%u：符号なし整数
%x：小文字の 16 進数　0x00-0xff
%X：大文字の 16 進数　0x00-0xFF
%d：符号付 10 進数
%e：実数の指数
%f：浮動小数点の実数
%Lx：Long 型の小文字の 16 進数　0x0000-0xffff
%LX：Long 型の大文字の 16 進数　0x0000-0xFFFF
%lu：Long 型の符号なし 10 進数
%ld：Long 型の符号あり 10 進数
%%：文字列そのもの

STEP４　アナログーデジタル変換を体験する(Analog-Digital Conversion)

STE４ではマイコンに内蔵された AD コンバータを使って、AD 変換を体験します。

まずはじめに、マイコンに入力する０V～５V まで連続的に変化するアナログ信号を、テスターを使って確認します。
可変抵抗（VR）を使ったアナログ信号を作る回路とその仕組みを下図に示します。



R1: R2 = IR1: IR2 = V1: V2

R1: **R2** = V1: **V2**

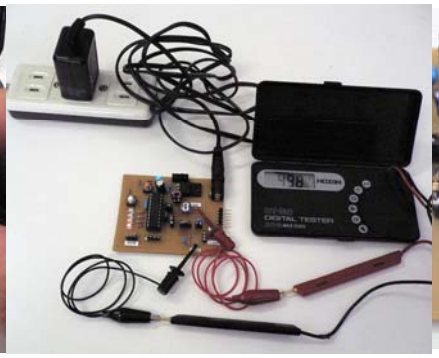
つまり、抵抗値 R2 はアナログ信号 V2 としてマイコンへ入力されます。

R2 → V2

【実験 1】

① ジャンパ(Jumper Pin)をはずしてください。
（ジャンパは無くさないように片側だけ隣の１本に付けておくとよい）

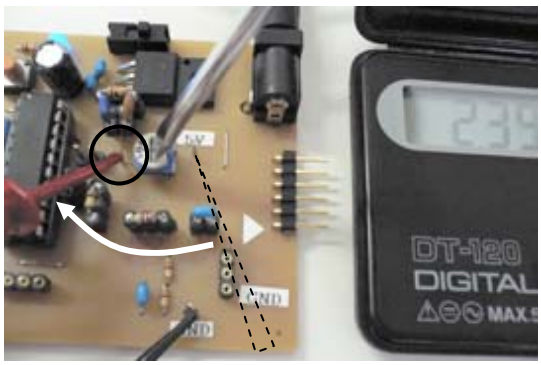
② 上図の **a - b 間**の電圧は，**5V** になります.
Digital Multi-Meter で確かめてください。
（DMM は電圧 **Voltage range** にする）

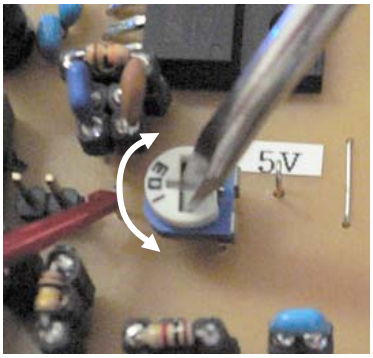


【実験２】

① ジャンパをはずしたままで。

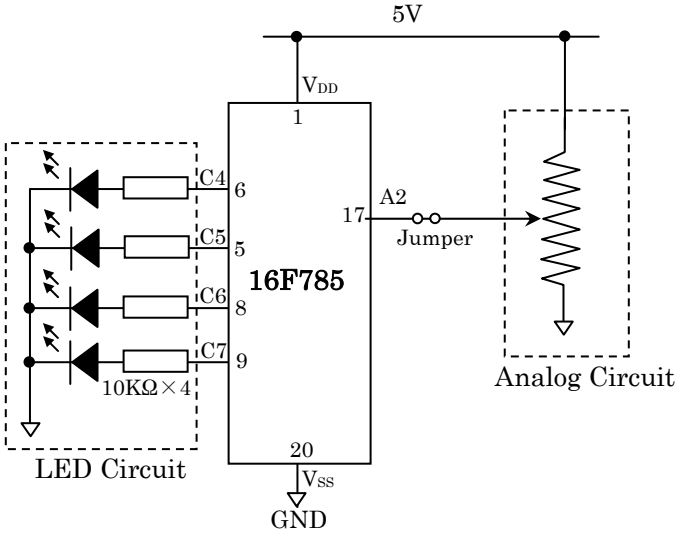
② 上図の **b - c** 間に DMM をコネクトし VR を回転させたときの電圧の変化を DMM で確かめてください。
（電圧レンジ: **Voltage range**）



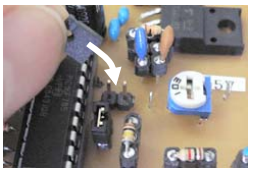


アナログ－デジタル変換の実行（Execution of Analog-Digital Conversion）

マイコンとアナログ回路の構成を下図に示します。**アナログ電圧をマイコンに入力し ADC を実行します。**



ジャンパを接続する
(Connect Jumper)



それでは、Step04_AD_00_manual_input_785.c　を実行してみましょう。

【実験 3】
可変抵抗（VR）をまわしたとき、LED の光り方を確認してください。

```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#device ADC=8                          //8bitAD setting,    8bit/10bit AD converter
#use delay(clock=8000000)
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

void main(void)
{
        int VR_data;

        set_tris_A(0xff);               // all input
        set_tris_B(0x30);               // output:B7,6,     input:B5,4 (N.C. B3,2,1.0)
        set_tris_C(0x03);               // output:C7,6,5,4,3,2 input:C1,0

        setup_adc(ADC_CLOCK_DIV_16);      //ADC initialization
        setup_adc_ports(sAN2);          // ADC setting port

        for(;;){
            set_adc_channel(2);       // A2(No17pin) = Channel2 select
            delay_us(50);             // Sample Hold Capacitor Charge time
            VR_data = read_adc();    // A2(17pin)=channel2 VR analog voltage input

            output_c(VR_data);
        }
}
```

【課題 4-1】
　このプログラムは 8bit の AD 入力で設定されています。つまり、５V が２５６分解能でデジタル変換されていますので、０V から５V が、０から２５５に変換されています。
そこで問題です。LED は４つしかありません。この LED は入力された信号をどのように点灯させているのでしょうか。

AD 変換とシリアル通信で入力結果を PC に表示する(ACD&serial&PC display)

　Step04_AD_01_manual_input_785.c を実行してください。このプログラムでは、10bit の AD 変換入力を実行し、その結果をシリアル通信プログラムによってパソコン側へ送信します。

【課題 4-2】3 行目、10bit 設定の場合、AD 変換値の範囲はいくつからいくつまでですか。

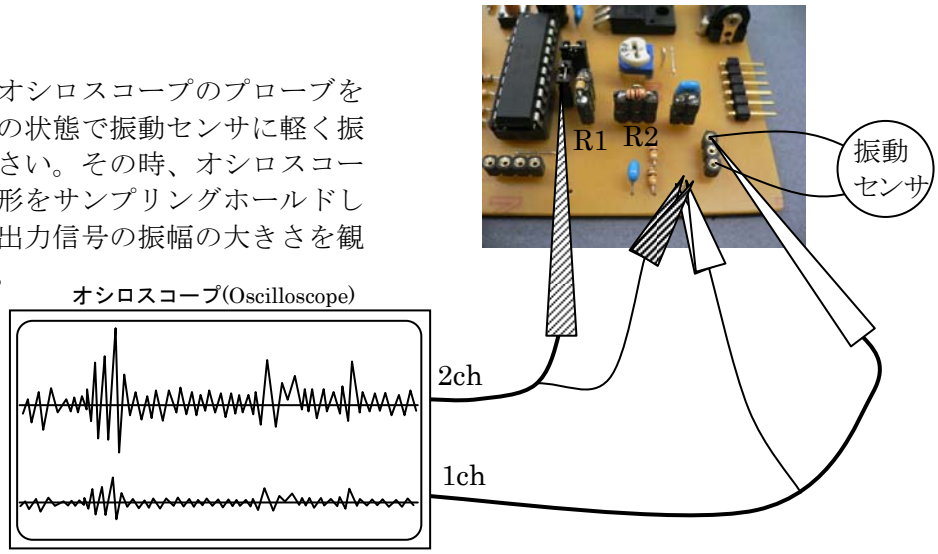
```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#device ADC=10                          // 10bit AD convert
#use delay(clock=8000000)
#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

void main(void)
{
        int cmd,led=0;
        long ct=0, VR_data;

        set_tris_A(0xff);               //All input
        set_tris_B(0x30);               //output:B7,6,     input:B5,4 (N.C. B3,2,1.0)
        set_tris_C(0x03);               //output:C7,6,5,4,3,2 input:C1,0

        setup_adc(ADC_CLOCK_DIV_16);
        setup_adc_ports(sAN2);

        printf("Hit any Key !    Start¥r¥n");   // waiting
        cmd=getc();

        while(1){
                set_adc_channel(2);     // A2(No17pin)=channel2 analog voltage input
                delay_us(50);
                VR_data = read_adc();

                printf("No.%4lu, VR Volts = %4lu¥r¥n",ct,VR_data);
                                        //voltage of resistance voltage divider
                output_BIT(PIN_C7,led);     // execution LED monitor
                led=led^1;

                delay_ms(50);                   // sampling time
                ct=ct+1;                         // data number counter
        }
}
```

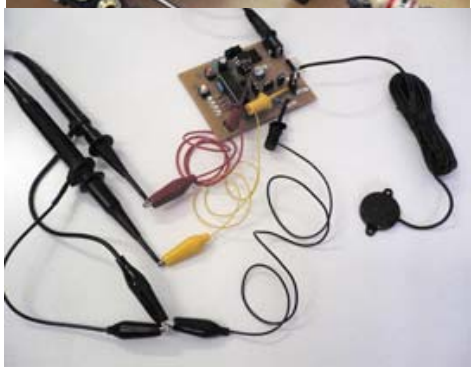
STEP5　オペアンプをアクティブにする（Operational Amplifier）

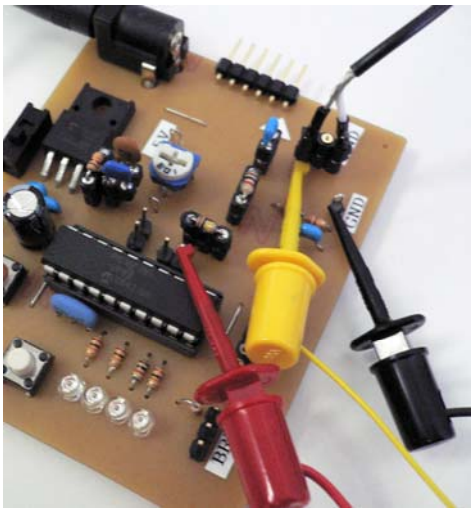
　このマイコンには、OP アンプ（Operational Amplifier）が内蔵されています。このモジュールを稼動させると、少しの外付け部品で微小信号を増幅させることができます。

まず、今の状態で（オペアンプを実行しない）入力信号と出力信号の変化をオシロスコープで見てみましょう。

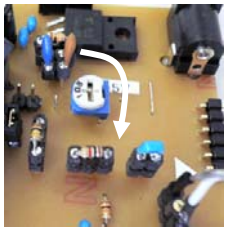
【実験４】
　右図のようにオシロスコープのプローブを接続します。その状態で振動センサに軽く振動を与えてください。その時、オシロスコープが表示する波形をサンプリングホールドして、入力信号と出力信号の振幅の大きさを観察してください。

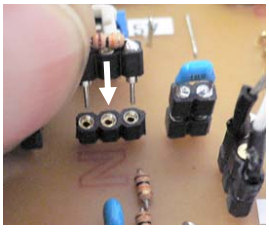






抵抗・コンデンサ差し替え(Changing R/C)





【実験５】
　次のプログラムを実行して、実験４と同様に、入出力の波形を観察してください。

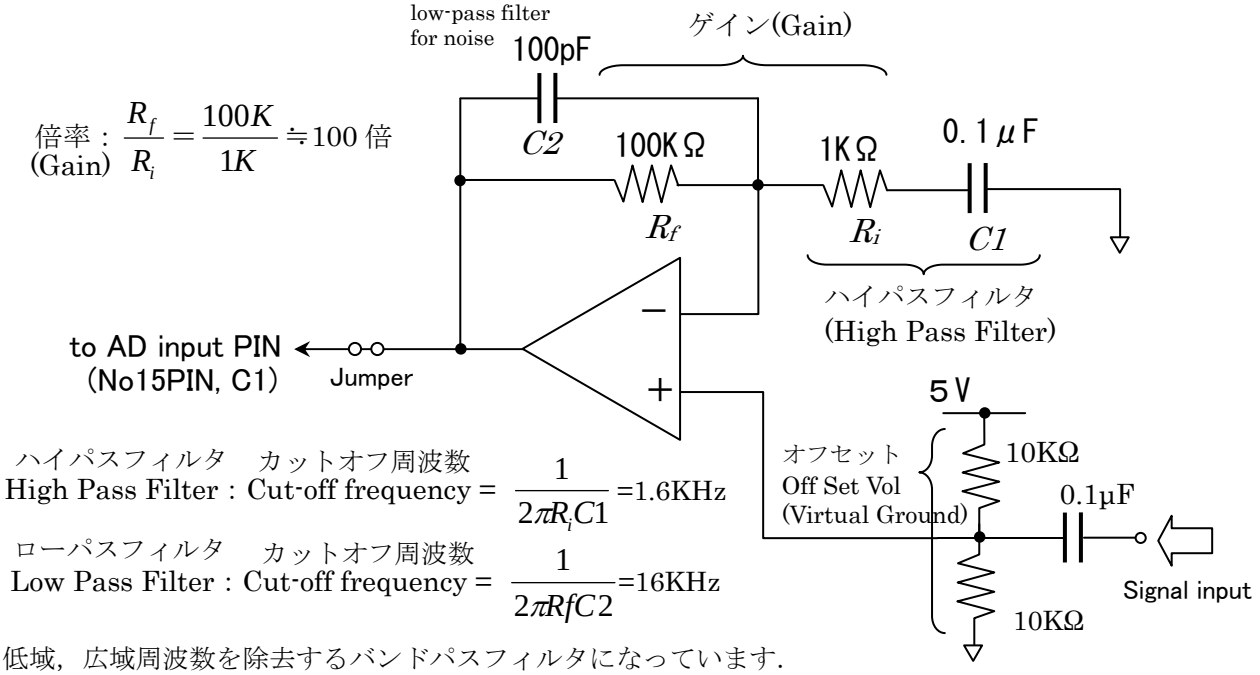
【実験６】
　ホルダーにある抵抗を R1 や R2 と差し替えて､その時の増幅の様子を観察してください。

Step05_OP_AMP_CHK00_785.c

```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

//#BYTE OPA1CON = 0x11C          // Register for OP AMP1
#BYTE OPA2CON = 0x11D            // Register for OP AMP2

main()
{
        set_tris_B(0x30);
        set_tris_C(0x03);

        //OPA1CON = 0x80;      // disable OP AMP1
        OPA2CON = 0x80;        // enable OP AMP2

        while(1){
                ;
        }
}
```

単電源の場合の OP アンプ増幅回路(Single-Supply Op-Amp Circuit)

微小信号の増幅には OP アンプがしばしば用いられます。今回の回路ではマイコンやその周辺回路を動作させるための５V 電源を利用しますので、増幅回路としては単電源タイプの OP アンプ回路となります。また、単電源では、交流波形のマイナス成分の信号は増幅できないため、入力信号の基準電圧を通常の０V から２.５V へと引き上げる、Offset 回路が必要となります。以下に単電源タイプの OP アンプを利用した増幅回路を示します。

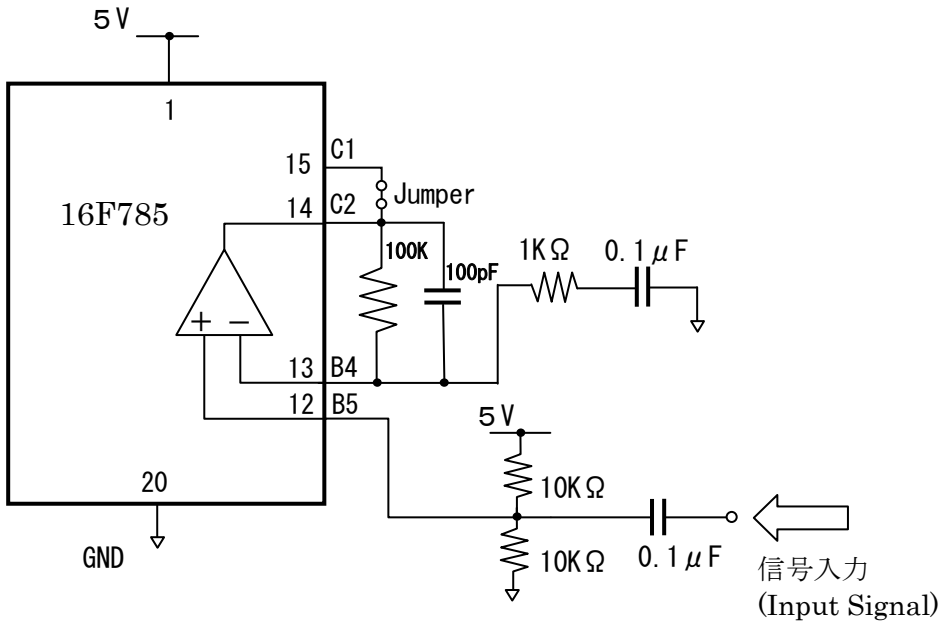


ハイパスフィルタ　カットオフ周波数
High Pass Filter：Cut-off frequency = $\frac{1}{2\pi R_i C1}$ =1.6KHz

ローパスフィルタ　カットオフ周波数
Low Pass Filter：Cut-off frequency = $\frac{1}{2\pi RfC2}$ =16KHz

低域，広域周波数を除去するバンドパスフィルタになっています.
つまり，この回路の周波数帯は 1.6KHz～16KHz です.

マイコン内部の OP アンプを利用した回路(Circuit using PIC16F785 Internal OP AMP)



STEP６　アナログ電圧を PC モニタに表示させる(OPA & ADC & SerialCom)

ここでは、STEP5 の発展プログラムとして、アナログ電圧値を AD 変換した値をシリアル通信によって、パソコンの画面に表示させます。
これまで学んだ、STEP３，４，５が組み合わされたプログラムへと統合します。
なお、set_adc_channel(5);を２に変更すれば可変抵抗(VR)の値がモニタできます。

Step06_OP_AMP_AD_serial_chk00_785.c

```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#device ADC=10
#use delay(clock=8000000)
#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)
//#BYTE OPA1CON = 0x11C
#BYTE OPA2CON = 0x11D

main()
{
        int cmd,led=0;
        long analog_data;

        set_tris_B(0x30);
        set_tris_C(0x03);

        //OPA1CON = 0x00;
        OPA2CON = 0x80;

        setup_adc(ADC_CLOCK_DIV_16);
        setup_adc_ports(sAN5);

        printf("Hit any Key !    Start¥r¥n");
        cmd=getc();

        while(1){

        set_adc_channel(5);          //5=analog input terminal data
        //set_adc_channel(2);        //2= Variable Resistance(VR) analog data
        delay_us(50);
        analog_data = read_adc();

        printf("Data = %lu¥r¥n",analog_data);      // Analog data Display
        delay_ms(40);                              // sampling time

        output_BIT(PIN_C7,led);                     // execution LED monitor
        //output_BIT(PIN_C7,1);
        led=led^1;

        }
}
```

Step06_OP_AMP_AD_serial_Dim_chk00_785.c

```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#device ADC=8
#use delay(clock=8000000)
#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)
//#BYTE OPA1CON = 0x11C
#BYTE OPA2CON = 0x11D
int anlg_data_dim[90];

void main(void)
{
        int ct,cmd,led=0,analog_data;

        set_tris_B(0x30);
        set_tris_C(0x03);

        //OPA1CON = 0x00;
        OPA2CON = 0x80;

        setup_adc(ADC_CLOCK_DIV_16);
        setup_adc_ports(sAN5);

        for(;;){
                printf("Hit any Key !    Start¥r¥n");
                cmd=getc();

                for(ct=0;ct<90;ct++){
                        set_adc_channel(5);                     //manual VR analog -> ch-2
                        delay_us(50);
                        analog_data = read_adc();
                        anlg_data_dim[ct]=analog_data;
                        analog_data=anlg_data_dim[ct];

                        printf("%3u    Data= %u¥r¥n",ct, analog_data); // Analog data Disp
                        delay_ms(40);                                   // sampling time

                       output_BIT(PIN_C7,led);                          // exe LED monitor
                       led=led^1;
                }

                printf("Hit any Key !    Dim Display Start¥r¥n");
                cmd=getc();

                for(ct=0;ct<90;ct++){
                        analog_data=anlg_data_dim[ct];
                        printf("%3u    Data= %u¥r¥n",ct, analog_data);   // Analog data Disp
                }
        }
}
```

STEP7　タイマ割り込みを使う（Timer Interrupt）

STEP7 では、タイマ割り込み(Timer Interrupt)という、テックニックを使います。このタイマ割り込みを使うとメインプログラム(main program)を実行中、一定の時間の間隔でサブプログラム(Sub program)を実行するプログラムが実現しますから、このサブプログラム側に定期的に作動するプログラムを書いておくと、一定間隔で仕事をしてくれます。

下記のサンプルプログラムは、メインプログラム(main proguram)には、プログラムは何もありません。しかし、プログラムを実行すると、一定の間隔でサブプログラム（sub program）が実行され、LED を点滅する（Flashing）動作を行います。

timer00_785 interrupt.c

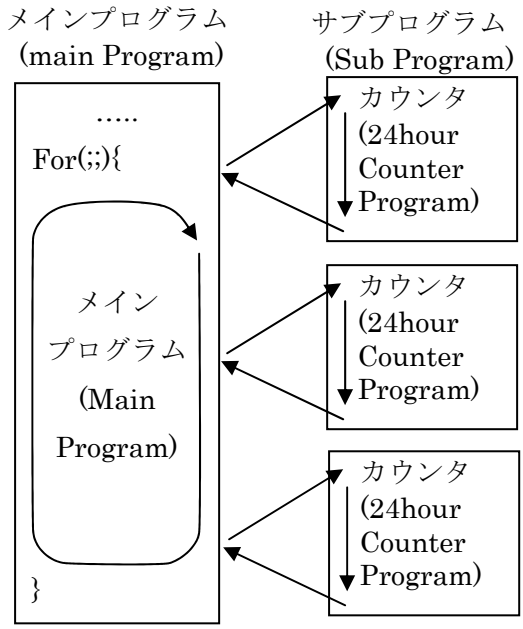
```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)          //internal oscillator clock setting
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

#INT_RTCC
rtcc_isr(){
          output_high(PIN_C7);                          // C port-C6 bit output Low level
          delay_ms(100);                                // delay time (msec)
          output_low(PIN_C7);                           // C port-C6 bit output Hi level
          delay_ms(100);
}

void main(void)
{
     setup_timer_0(RTCC_INTERNAL | RTCC_DIV_256);   // timer0 unit counter setting
     enable_interrupts(INT_TIMER0);                  // Timer0 enable
     enable_interrupts(GLOBAL);                      // interrupt enable

     set_tris_c(0x00);                               // C port bit in/out setting, All output

     for(;;){                                        // endless loop
               ;                                     // No proguram
     }
}
```

タイマ割り込みを使って時計をつくる(24 hour Timer Programming using Timer Interrupt)

STEP7 では、マイコンで 24 時間(h)、60 分(min)をカウントする時計を実現します。このプログラムではタイマ割り込み(Timer Interrupt)という、テックニックを使います。

タイマ割り込みを使うとメインプログラム(main program)を実行中、一定の時間の間隔でサブプログラム(Sub program)を実行するプログラムが実現しますから、このサブプログラム側に 24 時間カウンタをプログラムすれば時計ができます。

そのイメージは下図のようになります。



１秒(sec)間隔の時間の求め方(1sec Interval Timer Interrupt)

4194304Hz のオシレータ(oscillator)が良い周波数となります。このマイコンはハードウエアの仕様によって、ベースクロック(base clock)を 1/4 に分周し、更にこのクロックを 1/256 に分周します。このクロックが、更にプログラムにより分周されます。この最後の分周比をプログラムにコーディングします。この設定は、
setup_timer_0(RTCC_INTERNAL | RTCC_DIV_256); の関数が行っています。
この場合は、更に 1/256 に分周するので、

$$4{,}194{,}304 \div (4 \times 256 \times 256) = 16$$

となります。つまりこの設定の場合、16 回割り込みで、1 秒の時間が経過することになります。

なお、timer01_785 60number.c のプログラムは、PIC16F785 の内部オシレータを８MＨｚに設定して使っているので、およそ 2 倍の、32 回、割り込みすると 1 秒になる計算になります。

また、この割り込みプログラムで作り出す 1 秒より、更に正確な 1 秒を求める場合は、トリマコンデンサを付けるなど、ハードウエアの調整機能が必要です。

timer01_785 60number.c

```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)          //internal oscillator clock setting
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

signed int msec,sec,min,hour;

// 60 number counter sub program
#INT_RTCC
rtcc_isr(){
        msec ++;
        if(msec == 32){             //32 -> 1 sec product parameter by 8M Hz
                msec = 0;
                sec ++;
        }
        if(sec==60){
                sec = 0;
                min ++;
        }
        if(min==60){
                min = 0;
                hour ++;
        }
        if(hour==24){
                hour = 0;
        }
}
void main(void)
{
    signed int disp_time;

    msec = 0;
    sec = 0;
    min = 0;
    hour = 0;

    setup_timer_0(RTCC_INTERNAL | RTCC_DIV_256);
                                    // timer0 unit counter time interval setting
    enable_interrupts(INT_TIMER0);      // Timer0 enable
    enable_interrupts(GLOBAL);          // interrupt enable

    set_tris_c(0x00);                   // C port bit in/out setting, All output

    for(;;){                            // endless loop
        disp_time=sec<<4;               // Displaying second time
        //disp_time=min<<4;             // for example minutes and hours
        //disp_time=hour<<4;
        output_c(disp_time);            // C port Byte Output
    };
}
```

4.19MHz 外部クロックを使ったタイマプログラム（LED ＆ PC モニタ）
(4.19MHz external Clock Timer programming, LED & PC monitor)

このプログラムでは#fuse の行で HS と書き換えることで外部オシレータを使用するように設定をしています。また、LED は 4bit しかないので秒を表示させ、シリアル通信によって PC 側のモニタに時間、分、秒を表示させています。

timer02_785 clock ajust.c

```
#include<16f785.h>

#fuses HS,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR   //INTRC_IO->HS
#use delay(clock=4194304)                     //external oscillator clock setting
#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

signed int msec,sec,min,hour;

#INT_RTCC
rtcc_isr(){
        msec ++;
        if(msec == 16){     //16 -> 1 sec product parameter by 8M Hz
                msec = 0;
                sec ++;
        }
        if(sec==60){
                sec = 0;
                min ++;
        }
        if(min==60){
                min = 0;
                hour ++;
        }
        if(hour==24){
                hour = 0;
        }
}
void main(void)
{
    signed int disp_time;

    msec = 0;
    sec = 0;
    min = 0;
    hour = 0;

    setup_timer_0(RTCC_INTERNAL | RTCC_DIV_256);      // counter time interval setting
    enable_interrupts(INT_TIMER0);                    // Timer0 enable
    enable_interrupts(GLOBAL);                        // interrupt enable

    set_tris_c(0x00);                                 // C port bit setting, All output

    for(;;){                                          // endless loop
        disp_time=sec<<4;        // Displaying second time(4bit shift to LED connection bit)
        //disp_time=min<<4;                           // for example min(minute) and hour
        output_c(disp_time);                          // C port Byte Output
        printf("%2u:%2u:%2u¥r¥n",hour,min,sec);       // to PC monitor
        delay_ms(1000);                               // Display timming
     }
}
```

付録(Appendix)　- サーボモータ制御　Servo Motor Control -

これまでと少し路線を変えて、機械的な対象を制御してみましょう。
ラジオコントロール模型などに使われている模型用サーボモータは比較的安価で制御しやすく、とてもよい学習対象です。

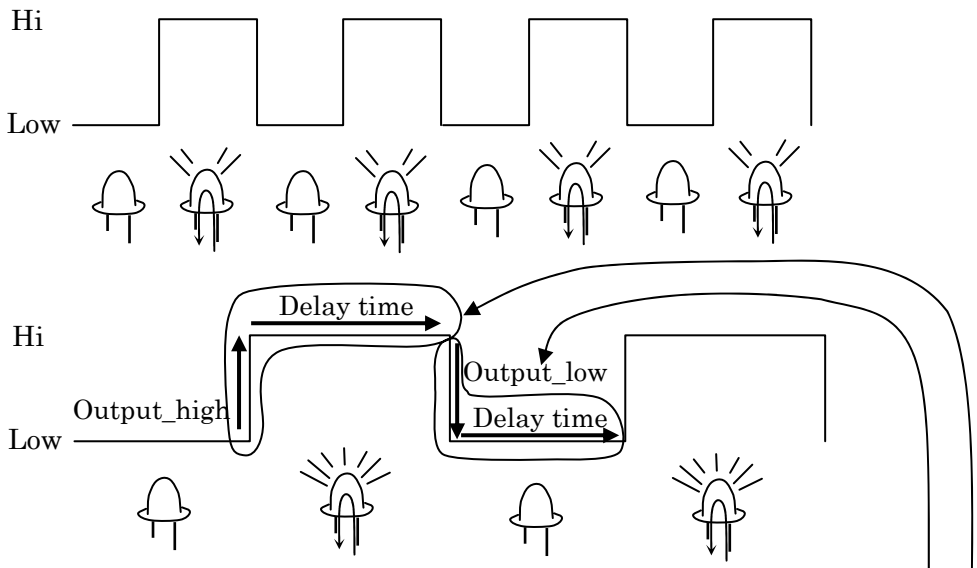
サーボモータの制御　→　パルス幅の制御　と言えます。

ですから、サーボモータの制御は、マイコン側から見ると、単に出力ビットを制御していることと何もかわらないのです。つまり、Hi、Low、Hi、Low と変化する出力の Hi の時間を正確にコントロールすればいいのです。

こんな例から試してみましょう。

「Hi、Low、Hi、Low と変化する出力」から連想できるものに、LED の点滅があります。そうです。実は、この制御もサーボモータの制御もほぼ一緒なのです。不思議ですね。

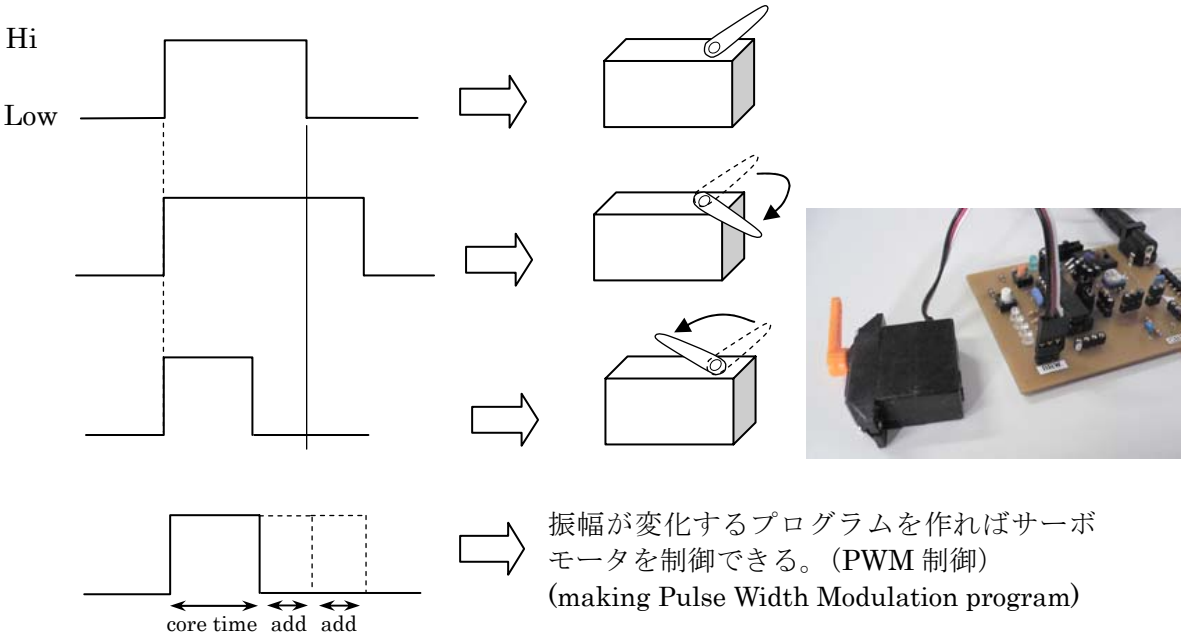
LED 点滅アルゴリズムの仕組み(Mechanism of LED Flashing algorism)



LED 点滅プログラム(LED Flashing program)　　Step01 out00_785.c

```
#include<16f785.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)          //internal oscillator clock setting
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)
main()
{
      set_tris_b(0x00);
      set_tris_c(0x00);

      do{
            output_low(PIN_C7);       // C port-C7 bit output Low level
            delay_ms(100);             // delay time (msec)

            output_high(PIN_C7);      // C port-C7 bit output Hi level
            delay_ms(100);             // delay time (msec)

       }while(1);
}
```

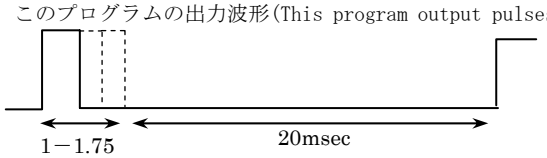
サーボ制御アルゴリズムの仕組み(Mechanism of Servo Control algorism)



Servo01_785 make control pulse.c

```
#include<16f785.h>

#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#use delay(clock=8000000)          //internal oscillator clock setting
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

void main(void)
{
      int pw_v;

      pw_v=0;                 // Let's try change variable number in 0<pw_v<255

      set_tris_b(0x00);     // B port bit in/out setting, All output: 0=output, 1=input
      set_tris_c(0x00);      // C port bit in/out setting, All output

      for(;;){                          // endless loop

            output_high(PIN_C7);     // Flashing ELD bit -> C port-C7 bit
            output_high(PIN_B7);     // External Output Port bit high -> Connecting Servo motor
                                      //
            delay_ms(1);              // Minimum Pulse Width (core time)(msec)->
                                      //                                  pulse width -> Servo control
                                      //    1 msec of minimum Pulse Width is larger yet.
            delay_us(pw_v);           // pulse width Add time (usec):      0<pw_v<255
            delay_us(pw_v);           // pulse width Add time (usec):      0<pw_v<255
            delay_us(pw_v);           // pulse width Add time (usec):      0<pw_v<255

            output_low(PIN_C7);       // C port-C6 bit output Low level
            output_low(PIN_B7);       // External Output Port bit Low

            delay_ms(20);             // blank time

      }
}
```

このプログラムの出力波形(This program output pulses)



タイマ割り込みを利用した PWM　プログラミング(Timer interrupt type PWM)

このプログラムは可変抵抗の回転に合わせて、パルス幅を 0.8-2msec　程度の範囲で変化させてサーボモータを制御します。割り込みは Timer0 と Timer1 を利用しています。

Servo02_785 pulse control by variable number.c

```
#include<16f785.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,PUT,BROWNOUT,MCLR
#device ADC=8                          // 8bit AD convert
#use delay(clock=8000000)              //internal oscillator clock setting
#use RS232(BAUD=19200, XMIT=PIN_B6 ,RCV=PIN_C0)
#USE FAST_IO(A)
#USE FAST_IO(B)
#USE FAST_IO(C)

long VR_data,loop;

// ***** Servo Control Pulse Generater *****
#INT_TIMER1
void timer1_isr(){   // blank time generater by Timer1
   set_timer1(53000);

   set_timer0(0);
   output_high(PIN_B7);                //   pulse level High
   while (get_timer0() < VR_data/7) {};  //   pulse width time by Timer0
   output_low(PIN_B7);                 //   pulse level Low
}
void main(void) {

// *****   initial setting *********
      set_tris_a(0x04);                // A port bit in/out setting, A2 input
      set_tris_b(0x00);                // B port bit in/out setting, All output:
      set_tris_c(0x00);                // C port bit in/out setting, All output

      setup_adc(ADC_CLOCK_DIV_16);
      setup_adc_ports(sAN2);
      set_adc_channel(2);

      setup_timer_0(RTCC_INTERNAL|RTCC_DIV_128);      //timer0 mode setting
      setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);       //timer1 mode setting
      set_timer1(20000);
      set_timer0(100);                 // comfortable number

      enable_interrupts(INT_TIMER1);   //timer1 enable
      enable_interrupts(GLOBAL);       //groval int enbale
      loop=0;

//****** waiting interrupt *******
      for(;;){
            output_low(PIN_c7);          // monitor LED flashing
            VR_data = read_adc();
            delay_ms(20);
            output_high(PIN_c7);
            delay_ms(30);

            loop++;
            if (loop>10){
                  printf("%3Lu¥r¥n",VR_data);
                  loop=0;
            }
      }
}
```

タイマ割り込み時間について

Timer1 の設定　(Timer1 setting) は 20msec の時間間隔をつくる

20msec(low pulse time)　/ ((1/8MHz) * 8(program setting))　=　20000

setup_timer_1(T1_INTERNAL | T1_DIV_BY_8)

Timer0 の設定（Timer0 Setting）は、1msec 前後の時間をつくる

2.4msec(max pulse width)/((1/8MHz)*4)=4800

4800 / 128 = 37.5
(setup_timer_0(RTCC_INTERNAL | RTCC_DIV_128))

ここで、VR の値は 0< VR disital number < 255 だから
255 / 7= 36.4 より　　VR / 7 は 37.5 より大きくならないので
37.5 > (255 / 7) とできる。（可変する範囲を確認した）

while (get_timer0() < VR_data/7) {};　による待ち時間(wait time)は、
可変抵抗 VR の値によって、サーボモータを制御するパルス幅内で、PWM 制御を実現。

PIC16F785/HV785
Data Sheet

20-Pin Flash-Based, 8-Bit CMOS Microcontroller with Two-Phase Asynchronous Feedback PWM Dual High-Speed Comparators and Dual Operational Amplifiers

PIC16F785/HV785

20-Pin Flash-Based 8-Bit CMOS Microcontroller

High-Performance RISC CPU:

- Only 35 Instructions to Learn:
  - All single-cycle instructions except branches
- Operating Speed:
  - DC – 20 MHz oscillator/clock input
  - DC – 200 ns instruction cycle
- Interrupt Capability
- 8-Level Seep Hardware Stack
- Direct, Indirect and Relative Addressing modes

Special Microcontroller Features:

- Precision Internal Oscillator:
  - Factory calibrated to ±1%
  - Software selectable frequency range of 8 MHz to 32 kHz
  - Software tunable
  - Two-Speed Start-up mode
  - Crystal fail detect for critical applications
  - Clock mode switching during operation for power savings
- Power-Saving Sleep mode
- Wide Operating Voltage Range (2.0V-5.5V)
- Industrial and Extended Temperature Range
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Reset (BOR) with Software Control Option
- Enhanced Low-Current Watchdog Timer (WDT) with on-chip Oscillator (software selectable nominal 268 seconds with full prescaler) with Software Enable
- Multiplexed Master Clear with Pull-up/Input Pin
- Programmable Code Protection
- High-Endurance Flash/EEPROM cell:
  - 100,000 write Flash endurance
  - 1,000,000 write EEPROM endurance
  - Flash/Data EEPROM retention: > 40 years

Low-Power Features:

- Standby Current:
  - 30 nA @ 2.0V, typical
- Operating Current:
  - 8.5 μA @ 32 kHz, 2.0V, typical
  - 100 μA @ 1 MHz, 2.0V, typical
- Watchdog Timer Current:
  - 1 μA @ 2.0V, typical
- Timer1 Oscillator Current:
  - 2 μA @ 32 kHz, 2.0V, typical

Peripheral Features:

- High-Speed Comparator module with:
  - Two independent analog comparators
  - Programmable on-chip voltage reference (CV_{REF}) module (% of V_{DD})
  - 1.2V band gap voltage reference
  - Comparator inputs and outputs externally accessible
  - < 40 ns propagation delay
  - 2 mv offset, typical
- Operational Amplifier module with 2 independent Op Amps:
  - 3 MHz GBWP, typical
  - All I/O pins externally accessible
- Two-Phase Asynchronous Feedback PWM module:
  - Complementary output with programmable dead band delay
  - Infinite resolution analog duty cycle
  - Sync Output/Input for multi-phase PWM
  - $F_{OSC}/2$ maximum PWM frequency
- A/D Converter:
  - 10-bit resolution and 14 channels (2 internal)
- 17 I/O pins and 1 Input-only Pin:
  - High-current source/sink for direct LED drive
  - Interrupt-on-pin change
  - Individually programmable weak pull-ups
- Timer0: 8-Bit Timer/Counter with 8-Bit Programmable Prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Option to use OSC1 and OSC2 in LP mode as Timer1 oscillator, if INTOSC mode selected
- Timer2: 8-Bit Timer/Counter with 8-Bit Period Register, Prescaler and Postscaler
- Capture, Compare, PWM module:
  - 16-bit Capture, max resolution 12.5 ns
  - Compare, max resolution 200 ns
  - 10-bit PWM with 1 output channel, max frequency 20 kHz
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Shunt Voltage Regulator (PIC16HV785 only):
  - 5 volt regulation
  - 4 mA to 50 mA shunt range

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Op Amps	Comparators	CCP	Two-Phase PWM	Timers 8/16-bit	Shunt Reg.
	Flash (words)	SRAM (bytes)	EEPROM (bytes)								
PIC16F785	2048	128	256	17+1	12+2	2	2	1	1	2/1	0
PIC16HV785	2048	128	256	17+1	12+2	2	2	1	1	2/1	1

Dual in Line Pin Diagram

20-pin PDIP, SOIC, SSOP

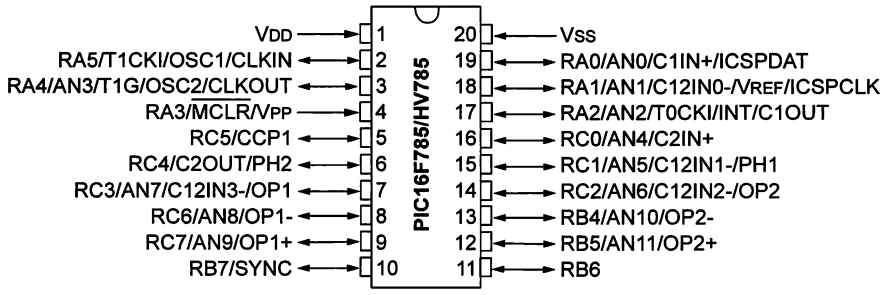


TABLE 1: DUAL IN LINE PIN SUMMARY

I/O	Pin	Analog	Comp.	Op Amps	PWM	Timers	CCP	Interrupt	Pull-ups	Basic
RA0	19	AN0	C1IN+	—	—	—	—	IOC	Y	ICSPDAT
RA1	18	AN1/VREF	C12IN0-	—	—	—	—	IOC	Y	ICSPCLK
RA2	17	AN2	C1OUT	—	—	T0CKI	—	INT/IOC	Y	—
RA3(1)	4	—	—	—	—	—	—	IOC	Y	MCLR/VPP
RA4	3	AN3	—	—	—	T1G	—	IOC	Y	OSC2/CLKOUT
RA5	2	—	—	—	—	T1CKI	—	IOC	Y	OSC1/CLKIN
RB4	13	AN10	—	OP2-	—	—	—	—	—	—
RB5	12	AN11	—	OP2+	—	—	—	—	—	—
RB6(2)	11	—	—	—	—	—	—	—	—	—
RB7	10	—	—	—	SYNC	—	—	—	—	—
RC0	16	AN4	C2IN+	—	—	—	—	—	—	—
RC1	15	AN5	C12IN1-	—	PH1	—	—	—	—	—
RC2	14	AN6	C12IN2-	OP2	—	—	—	—	—	—
RC3	7	AN7	C12IN3-	OP1	—	—	—	—	—	—
RC4	6	—	C2OUT	—	PH2	—	—	—	—	—
RC5	5	—	—	—	—	—	CCP1	—	—	—
RC6	8	AN8	—	OP1-	—	—	—	—	—	—
RC7	9	AN9	—	OP1+	—	—	—	—	—	—
—	1	—	—	—	—	—	—	—	—	VDD
—	20	—	—	—	—	—	—	—	—	VSS

Note 1: Input only.
2: Open drain.

1.0 DEVICE OVERVIEW

This document contains device specific information for the PIC16F785/HV785. It is available in 20-pin PDIP, SOIC, SSOP and QFN packages. Figure 1-1 shows a block diagram of the PIC16F785/HV785 device. Table 1-1 shows the pinout description.

FIGURE 1-1: PIC16F785/HV785 BLOCK DIAGRAM

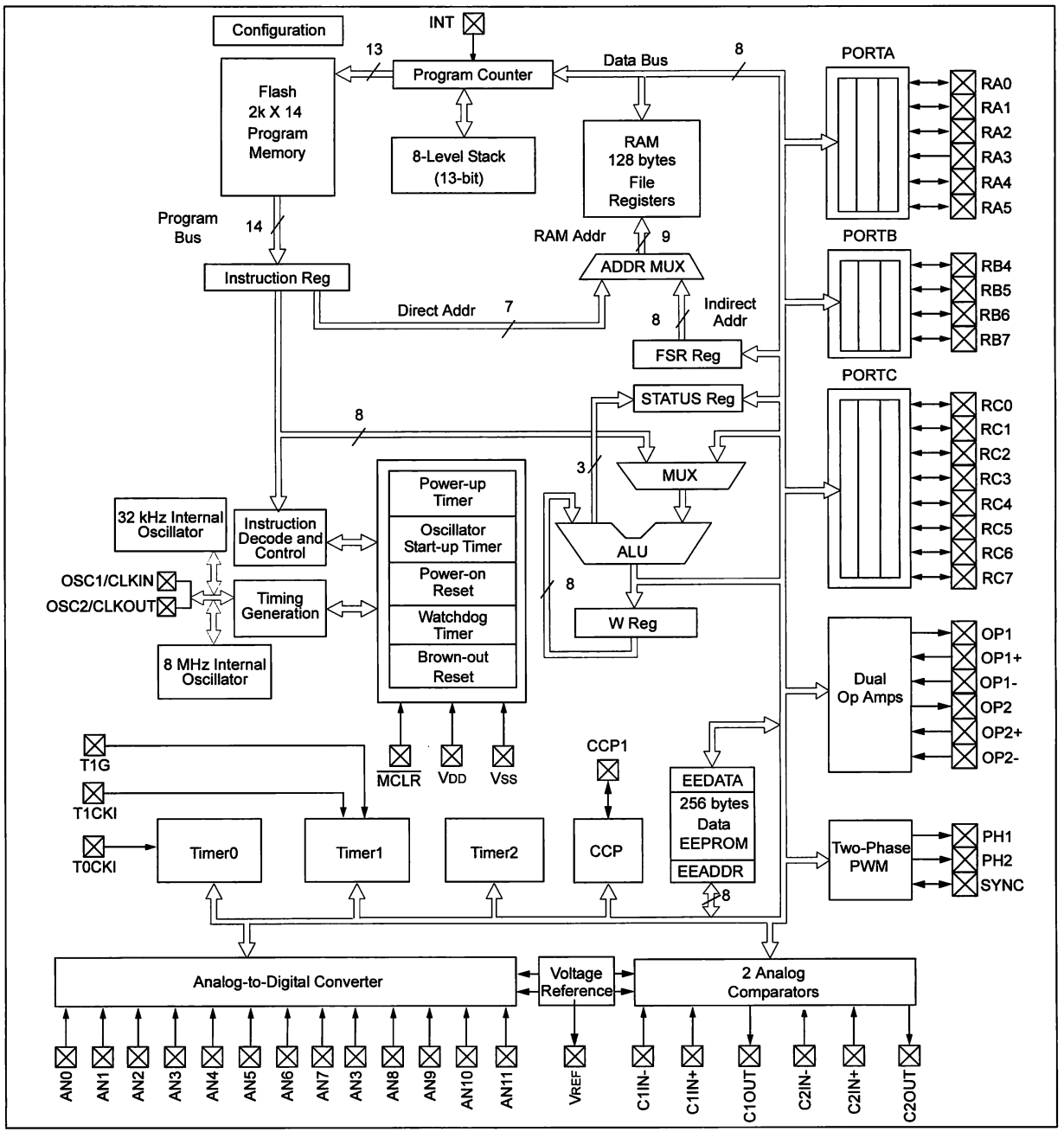


TABLE 1-1: PIC16F785/HV785 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0/C1IN+/ICSPDAT	RA0	TTL	CMOS	PORTA I/O with prog. pull-up and interrupt-on-change
	AN0	AN	—	A/D Channel 0 input
	C1IN+	AN	—	Comparator 1 non-inverting input
	ICSPDAT	ST	CMOS	Serial Programming Data I/O
RA1/AN1/C12IN0-/VREF/ ICSPCLK	RA1	TTL	CMOS	PORTA I/O with prog. pull-up and interrupt-on-change
	AN1	AN	—	A/D Channel 1 input
	C12IN0-	AN	—	Comparator 1 and 2 inverting input
	VREF	AN	AN	External Voltage Reference for A/D, buffered reference output
	ICSPCLK	ST	—	Serial Programming Clock
RA2/AN2/T0CKI/INT/C1OUT	RA2	ST	CMOS	PORTA I/O with prog. pull-up and interrupt-on-change
	AN2	AN	—	A/D Channel 2 input
	T0CKI	ST	—	Timer0 clock input
	INT	ST	—	External Interrupt
	C1OUT	—	CMOS	Comparator 1 output
RA3/$\overline{\text{MCLR}}$/Vpp	RA3	TTL	—	PORTA input with prog. pull-up and interrupt-on-change
	$\overline{\text{MCLR}}$	ST	—	Master Clear with internal pull-up
	VPP	HV	—	Programming voltage
RA4/AN3/T1G/OSC2/ CLKOUT	RA4	TTL	CMOS	PORTA I/O with prog. pull-up and interrupt-on-change
	AN3	AN	—	A/D Channel 3 input
	T1G	ST	—	Timer1 gate
	OSC2	—	XTAL	Crystal/Resonator
	CLKOUT	—	CMOS	FOSC/4 output
RA5/T1CKI/OSC1/CLKIN	RA5	TTL	CMOS	PORTA I/O with prog. pull-up and interrupt-on-change
	T1CKI	ST	—	Timer1 clock
	OSC1	XTAL	—	Crystal/Resonator
	CLKIN	ST	—	External clock input/RC oscillator connection
RB4/AN10/OP2-	RB4	TTL	CMOS	PORTB I/O
	AN10	AN	—	A/D Channel 10 input
	OP2-	—	AN	Op Amp 2 inverting input
RB5/AN11/OP2+	RB5	TTL	CMOS	PORTB I/O
	AN11	AN	—	A/D Channel 11 input
	OP2+	—	AN	Op Amp 2 non-inverting input
RB6	RB6	TTL	OD	PORTB I/O. Open drain output
RB7/SYNC	RB7	TTL	CMOS	PORTB I/O
	SYNC	ST	CMOS	Master PWM Sync output or slave PWM Sync input
RC0/AN4/C2IN+	RC0	TTL	CMOS	PORTC I/O
	AN4	AN	—	A/D Channel 4 input
	C2IN+	AN	—	Comparator 2 non-inverting input

Legend: TTL = TTL input buffer, ST = Schmitt Trigger input buffer, AN = Analog, OD = Open Drain output, HV = High Voltage

TABLE 1-1: PIC16F785/HV785 PINOUT DESCRIPTION (CONTINUED)

Name	Function	Input Type	Output Type	Description
RC1/AN5/C12IN1-/PH1	RC1	TTL	CMOS	PORTC I/O
	AN5	AN	—	A/D Channel 5 input
	C12IN1-	AN	—	Comparator 1 and 2 inverting input
	PH1	—	CMOS	PWM phase 1 output
RC2/AN6/C12IN2-/OP2	RC2	TTL	CMOS	PORTC I/O
	AN6	AN	—	A/D Channel 6 input
	C12IN2-	AN	—	Comparator 1 and 2 inverting input
	OP2	—	AN	Op Amp 2 output
RC3/AN7/C12IN3-/OP1	RC3	TTL	CMOS	PORTC I/O
	AN7	AN	—	A/D Channel 7 input
	C12IN3-	AN	—	Comparator 1 and 2 inverting input
	OP1	—	AN	Op Amp 1 output
RC4/C2OUT/PH2	RC4	TTL	CMOS	PORTC I/O
	C2OUT	—	CMOS	Comparator 2 output
	PH2	—	CMOS	PWM phase 2 output
RC5/CCP1	RC5	TTL	CMOS	PORTC I/O
	CCP1	ST	CMOS	Capture input/Compare output
RC6/AN8/OP1-	RC6	TTL	CMOS	PORTC I/O
	AN8	AN	—	A/D Channel 8 input
	OP1-	AN	—	Op Amp 1 inverting input
RC7/AN9/OP1+	RC7		CMOS	PORTC I/O
	AN9	AN	—	A/D Channel 9 input
	OP1+	AN	—	Op Amp 1 non-inverting input
VSS	VSS	Power	—	Ground reference
VDD	VDD	Power	—	Positive supply

Legend: TTL = TTL input buffer, ST = Schmitt Trigger input buffer, AN = Analog, OD = Open Drain output, HV = High Voltage

3.2 Clock Source Modes

Clock Source modes can be classified as external or internal.

- External Clock modes rely on external circuitry for the clock source. Examples are oscillator modules (EC mode), quartz crystal resonators or ceramic resonators (LP, XT, and HS modes) and resistor-capacitor (RC mode) circuits.
- Internal clock sources are contained internally within the PIC16F785/HV785. The PIC16F785/HV785 has two internal oscillators; the 8 MHz High-frequency Internal Oscillator (HFINTOSC) and 31 kHz Low-frequency Internal Oscillator (LFINTOSC).

The system clock can be selected between external or internal clock sources via the System Clock Selection (SCS) bit (see **Section 3.5 "Clock Switching"**).

3.3 External Clock Modes

3.3.1 OSCILLATOR START-UP TIMER (OST)

When the PIC16F785/HV785 is configured for any of the Crystal Oscillator modes (LP, XT or HS), the Oscillator Start-up Timer (OST) is enabled, which extends the Reset period to allow the oscillator additional time to stabilize. The OST counts 1024 clock periods present on the OSC1 pin following a Power-on Reset (POR), a wake from Sleep, or when the Power-up Timer (PWRT) has expired (if the PWRT is enabled). During this time, the program counter does not increment and program execution is suspended. The OST ensures that the oscillator circuit, using a quartz crystal resonator or ceramic resonator, has started and is providing a stable system clock to the PIC16F785/HV785. Table 3-1 shows examples where the oscillator delay is invoked.

In order to minimize latency between external oscillator start-up and code execution, the Two-Speed Clock Start-up mode can be selected (see **Section 3.6 "Two-Speed Clock Start-up Mode"**).

TABLE 3-1: OSCILLATOR DELAY EXAMPLES

Switch From	Switch To	Frequency	Oscillator Delay	Comments
Sleep/POR	INTRC INTOSC	31 kHz 125 kHz-8 MHz	5 µs-10 µs (approx.) CPU Start-up(1)	Following a wake-up from Sleep mode or POR, CPU start-up is invoked to allow the CPU to become ready for code execution.
Sleep	EC, RC	DC – 20 MHz		
LFINTOSC (31 kHz)	EC, RC	DC – 20 MHz		
Sleep/POR	LP, XT, HS	31 kHz-20 MHz	1024 Clock Cycles (OST)	
LFINTOSC (31 kHz)	INTOSC	125 kHz-8 MHz	1 µs (approx.)	

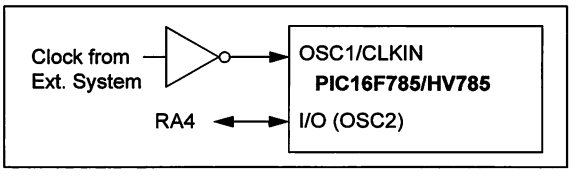
Note 1: The 5 µs-10 µs start-up delay is based on a 1 MHz System Clock.

3.3.2 EC MODE

The External Clock (EC) mode allows an externally generated logic level as the system clock source. When operating in this mode, an external clock source is connected to OSC1 pin and the RA4 pin is available for general purpose I/O. Figure 3-2 shows the pin connections for EC mode.

The Oscillator Start-up Timer (OST) is disabled when EC mode is selected. Therefore, there is no delay in operation after a Power-on Reset (POR) or wake-up from Sleep. Because the PIC16F785/HV785 design is fully static, stopping the external clock input will have the effect of halting the device while leaving all data intact. Upon restarting the external clock, the device will resume operation as if no time had elapsed.

FIGURE 3-2: EXTERNAL CLOCK (EC) MODE OPERATION



3.3.3 LP, XT, HS MODES

The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to the OSC1 and OSC2 pins (Figure 3-1). The mode selects a low, medium or high gain setting of the internal inverter-amplifier to support various resonator types and speed.

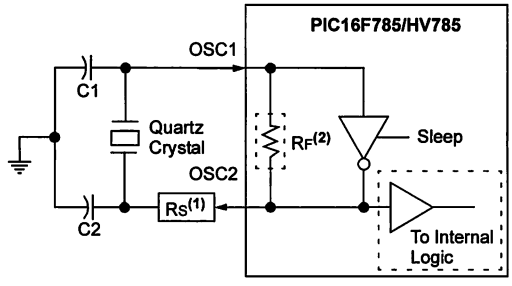
LP Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is best suited to drive resonators with a low drive level specification, for example, tuning fork type crystals.

XT Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. XT mode current consumption is the medium of the three modes. This mode is best suited to drive resonators with a medium drive level specification, for example, AT-cut quartz crystal resonators.

HS Oscillator mode selects the highest gain setting of the internal inverter-amplifier. HS mode current consumption is the highest of the three modes. This mode is best suited for resonators that require a high drive setting, for example, AT-cut quartz crystal resonators or ceramic resonators.

Figure 3-3 and Figure 3-4 show typical circuits for quartz crystal and ceramic resonators, respectively.

FIGURE 3-3: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)



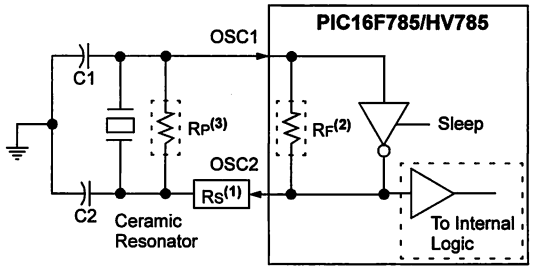
Note 1: A series resistor (RS) may be required for quartz crystals with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

Note 1: Quartz crystal characteristics vary according to type, package and manufacturer. The user should consult the manufacturer data sheets for specifications and recommended application.

2: Always verify oscillator performance over the VDD and temperature range that is expected for the application.

FIGURE 3-4: CERAMIC RESONATOR OPERATION (XT OR HS MODE)



Note 1: A series resistor (RS) may be required for ceramic resonators with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

3: An additional parallel feedback resistor (RP) may be required for proper ceramic resonator operation (typical value 1 MΩ).

TABLE 3-2: CERAMIC RESONATORS

Mode	Freq.	OSC1 (C1)	OSC2 (C2)
XT	455 kHz	68-100 pF	68-100 pF
	2.0 MHz	15-68 pF	15-68 pF
HS	4.0 MHz	10-68 pF	10-68 pF
	8.0 MHz	15-68 pF	15-68 pF
	16.0 MHz	10-22 pF	10-22 pF

Note: These values are for design guidance only. See notes following this table.

TABLE 3-3: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	15-33 pF	15-33 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15-33 pF	15-33 pF
	4 MHz	15-33 pF	15-33 pF
HS	4 MHz	15-33 pF	15-33 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF

Note: These values are for design guidance only. See notes following this table.

Note 1: Higher capacitance increases the stability of the oscillator, but also increases the start-up time.

2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

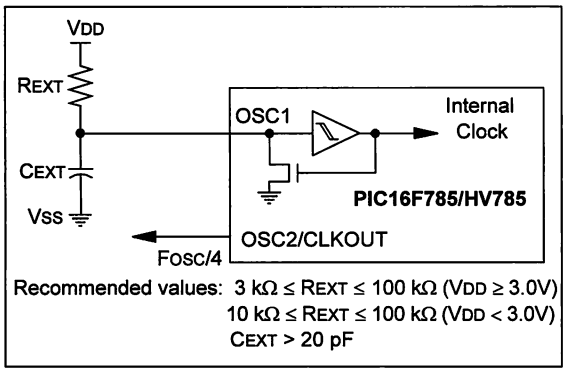
3: RS may be required to avoid overdriving crystals with low drive level specification.

3.3.4 EXTERNAL RC MODES

The External Resistor-Capacitor (RC) modes support the use of an external RC circuit. This allows the designer maximum flexibility in frequency choice while keeping costs to a minimum when clock accuracy is not required. There are two modes, RC and RCIO.

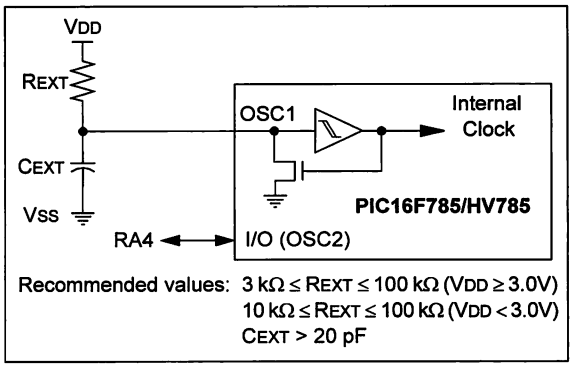
In RC mode, the RC circuit connects to the OSC1 pin. The OSC2/CLKOUT pin outputs the RC oscillator frequency divided by 4. This signal may be used to provide a clock for external circuitry, synchronization, calibration, test or other application requirements. Figure 3-5 shows the RC mode connections.

FIGURE 3-5: RC MODE



In RCIO mode, the RC circuit is connected to the OSC1 pin. The OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 4 of PORTA (RA4). Figure 3-6 shows the RCIO mode connections.

FIGURE 3-6: RCIO MODE



The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit-to-unit due to normal threshold voltage. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency or low CEXT values. The user also needs to take into account variation due to tolerance of external RC components used.

11.0 OPERATIONAL AMPLIFIER (OPA) MODULE

The OPA module has the following features:

- Two independent Operational Amplifiers
- External connections to all ports
- 3 MHz Gain Bandwidth Product (GBWP)

11.1 Control Registers

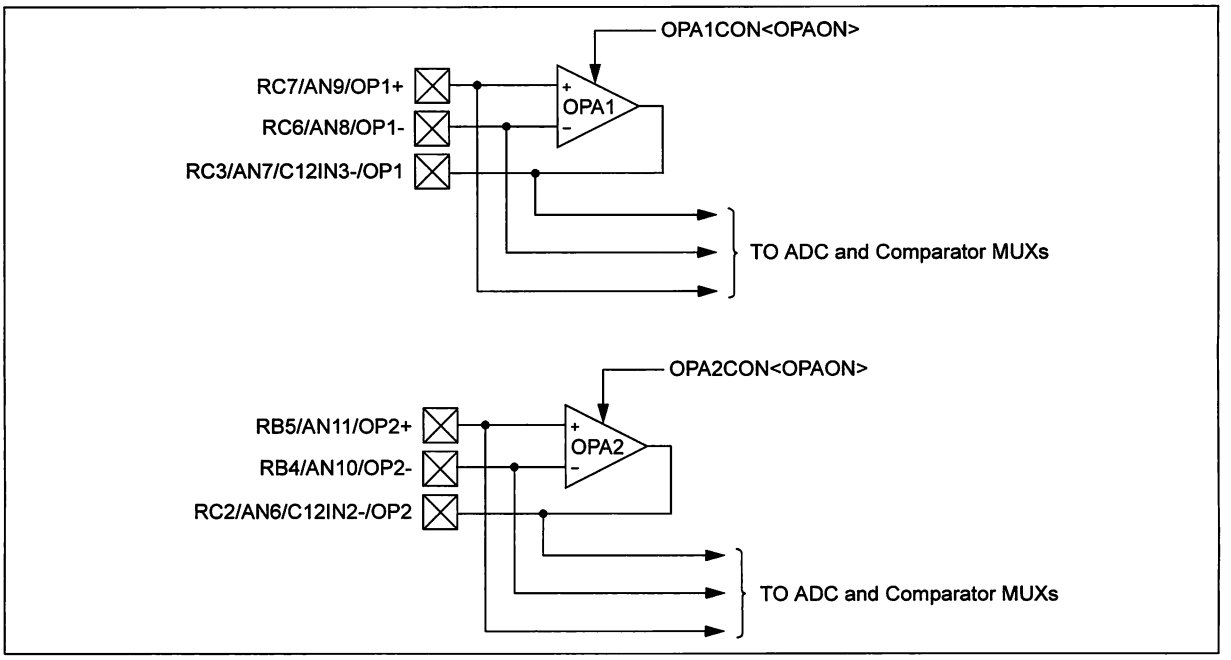
The OPA1CON register, shown in Register 11-1, controls OPA1. OPA2CON, shown in Register 11-2, controls OPA2.

11.2 OPAxCON Register

The OPA module is enabled by setting the OPAON bit of the OPAxCON Register. When enabled, OPAON forces the output driver of RC3/AN7/C12IN3-/OP1 for OPA1, and RC2/AN6/C12IN2-/OP2 for OPA2, into tristate to prevent contention between the driver and the OPA output. The ADC and comparator inputs which share the op amp pins operate normally when the op amp is enabled.

Note: When OPA1 or OPA2 is enabled, the RC3/AN7/C12IN3-/OP1 pin, or RC2/AN6/C12IN2-/OP2 pin, respectively, is driven by the op amp output, not by the PORTC driver. Refer to Table 19-11 for the electrical specifications for the op amp output drive capability.

FIGURE 11-1: OPA MODULE BLOCK DIAGRAM



REGISTER 11-1: OPA1CON: OP AMP 1 CONTROL REGISTER

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
OPAON	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **OPAON:** Op Amp Enable bit
1 = Op Amp 1 is enabled
0 = Op Amp 1 is disabled

bit 6-0 **Unimplemented:** Read as '0'

REGISTER 11-2: OPA2CON: OP AMP 2 CONTROL REGISTER

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
OPAON	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **OPAON:** Op Amp Enable bit
1 = Op Amp 2 is enabled
0 = Op Amp 2 is disabled

bit 6-0 **Unimplemented:** Read as '0'

15.2.1 POWER-ON RESET

The on-chip POR circuit holds the chip in Reset until VDD has reached a high enough level for proper operation. A minimum rise rate for VDD is required. See **Section 19.0 "Electrical Specifications"** for details. If the BOR is enabled, the minimum rise rate specification does not apply. The BOR circuitry will keep the device in Reset until VDD reaches VBOR (see **Section 15.2.4 "Brown-Out Reset (BOR)"**)

The POR circuit, on this device, has a POR re-arm circuit. This circuit is designed to ensure a re-arm of the POR circuit if VDD drops below a preset re-arming voltage (VPARM) for at least the minimum required time. Once VDD is below the re-arming point for the minimum required time, the POR Reset will reactivate and remain in Reset until VDD returns to a value greater than VPOR. At this point, a 1 µs (typical) delay will be initiated to allow VDD to continue to ramp to a voltage safely above VPOR.

When the device starts normal operation (exits the Reset condition), device operating parameters (i.e., voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in Reset until the operating conditions are met.

For additional information, refer to Application Note AN607, *"Power-up Trouble Shooting"* (DS00607).

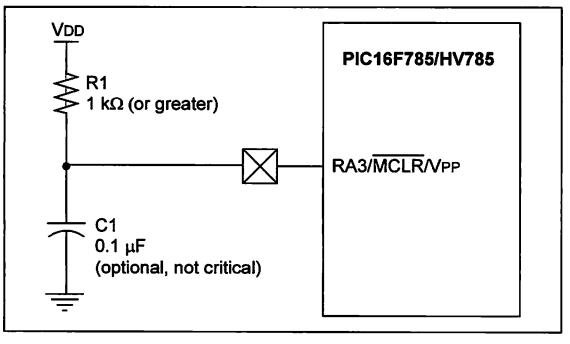
15.2.2 MASTER CLEAR ($\overline{\text{MCLR}}$)

PIC16F785/HV785 has a noise filter in the $\overline{\text{MCLR}}$ Reset path. The filter will detect and ignore small pulses.

It should be noted that a WDT Reset does not drive $\overline{\text{MCLR}}$ pin low.

The behavior of the ESD protection on the $\overline{\text{MCLR}}$ pin has been altered from earlier devices of this family. Voltages applied to the pin that exceed its specification can result in both $\overline{\text{MCLR}}$ Resets and excessive current beyond the device specification during the ESD event. For this reason, Microchip recommends that the $\overline{\text{MCLR}}$ pin no longer be tied directly to VDD. The use of an RC network, as shown in Figure 15-1, is suggested.

FIGURE 15-2: RECOMMENDED $\overline{\text{MCLR}}$ CIRCUIT



An internal $\overline{\text{MCLR}}$ option is enabled by clearing the MCLRE bit in the Configuration Word. When cleared, $\overline{\text{MCLR}}$ is internally tied to VDD and an internal Weak Pull-up is enabled for the $\overline{\text{MCLR}}$ pin. The VPP function of the RA3/$\overline{\text{MCLR}}$/VPP pin is not affected by selecting the internal $\overline{\text{MCLR}}$ option.

15.2.3 POWER-UP TIMER (PWRT)

The Power-up Timer provides a fixed 64 ms (nominal) time out on power-up only, from POR or Brown-out Reset. The Power-up Timer operates from the 31 kHz LFINTOSC oscillator. For more information, see **Section 3.4 "Internal Clock Modes"**. The chip is kept in Reset as long as PWRT is active. The PWRT delay allows the VDD to rise to an acceptable level. A configuration bit, $\overline{\text{PWRTE}}$ can disable (if '1') or enable (if '0') the Power-up Timer. The Power-up Timer should be enabled when Brown-out Reset is enabled, although it is not required.

The Power-up Time Delay will vary from chip-to-chip and vary due to:

- VDD variation
- Temperature variation
- Process variation

See DC parameters for details (**Section 19.0 "Electrical Specifications"**).

15.2.4 BROWN-OUT RESET (BOR)

The BOREN0 and BOREN1 bits in the Configuration Word select one of four BOR modes. Two modes have been added to allow software or hardware control of the BOR enable. When BOREN<1:0> = 01, the SBOREN bit of the PCON Register enables/disables the BOR allowing it to be controlled in software. By selecting BOREN<1:0>, the BOR is automatically disabled in Sleep to conserve power, and enabled on wake-up. In this mode, the SBOREN bit is disabled. See Register 15.2 for the Configuration Word definition.

If VDD falls below VBOR for greater than parameter (TBOR), see **Section 19.0 "Electrical Specifications"**, the Brown-out situation will reset the device. This will occur regardless of the VDD slew rate. A Reset is not assured if VDD falls below VBOR for less than parameter (TBOR).

On any Reset (Power-on, Brown-out Reset, Watchdog, etc.), the chip will remain in Reset until VDD rises above VBOR (see Figure 15-3). The Power-up Timer will now be invoked, if enabled, and will keep the chip in Reset an additional 64 ms.

Note: The Power-up Timer is enabled by the $\overline{\text{PWRTE}}$ bit in the Configuration Word.

If VDD drops below VBOR while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be re-initialized. Once VDD rises above VBOR, the Power-up Timer will execute a 64 ms Reset.

ローパワー、＋3.3 V、RS-232 ライン・ドライバ／レシーバ

ADM3202/ADM3222/ADM1385

特長

460 kbpsのデータ・レート
＋3.3 Vで仕様を規定
EIA-232E規格に適合
0.1μFチャージ・ポンプ・コンデンサ
低消費電力シャットダウン（ADM3222EとADM1385）
DIP、SO、SOIC、SSOP、およびTSSOPパッケージ・オプション
MAX3222/32およびLTC1385の上位互換バージョン

アプリケーション

汎用RS-232データ・リンク
携帯用機器
プリンタ
パームトップ・コンピュータ
PDA

概要

ADM3202/ADM3222/ADM1385トランシーバは、＋3.3 V単電源で動作する高速、2チャネルRS-232/V.28インターフェース・デバイスです。

低消費電力とシャットダウン機能（ADM3222/ADM1385）により、バッテリ駆動型の携帯用機器に最適です。

ADM3202/ADM3222/ADM1385は、EIA-232EおよびCCITT V.28規格に適合し、最高460 kbpsの転送レートで動作します。

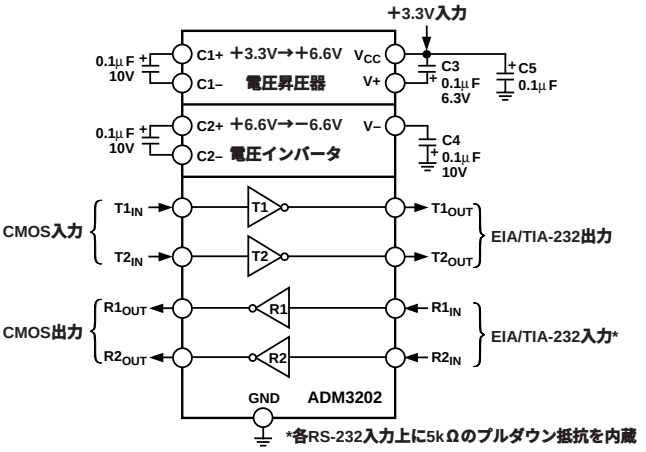
4個の外部0.1μFチャージ・ポンプ・コンデンサを電圧昇圧器/インバータに使用し、＋3.3 V単電源での動作が可能です。

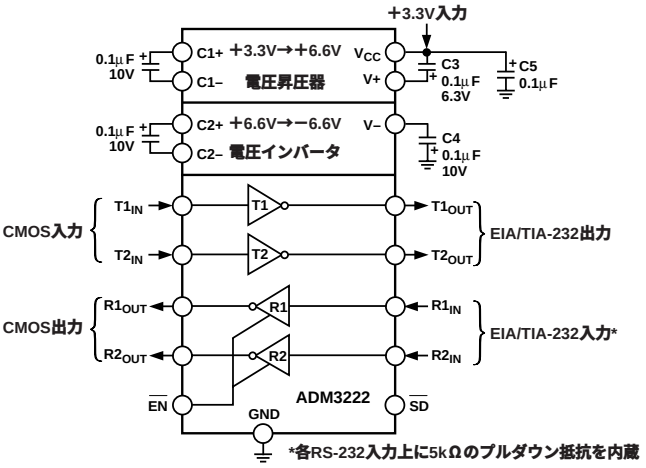
ADM3222は追加イネーブルおよびシャットダウン回路を内蔵しています。$\overline{EN}$入力を使用して、レシーバ出力を3ステート状態にすることができます。また、$\overline{SD}$入力を使用してチャージ・ポンプとトランスミッタ出力をパワーダウンすると、消費電流が0.5μA未満に減少します。レシーバは、$\overline{EN}$を使用してディスエーブルしない限り、シャットダウン中はイネーブルされたままです。

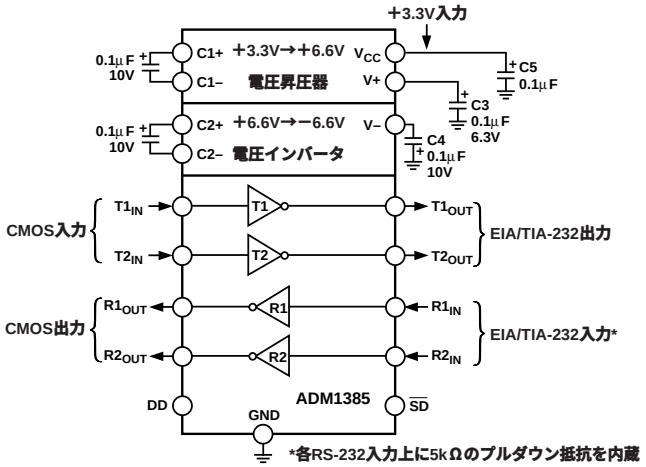
ADM1385はドライバ・ディスエーブル・モードと完全シャットダウン・モードを備えています。

ADM3202は16ピンDIP、ナローおよびワイドSOIC、さらに省スペース型の20ピンTSSOPパッケージで供給されます。ADM3222は18ピンDIP、SO、20ピンSSOPおよびTSSOPで供給されます。また、ADM1385は20ピンSSOPパッケージで供給され、LTC1385 CGとピン・コンパチブルです。

機能ブロック図







REV.0

アナログ・デバイセズ株式会社

本　　社／東京都港区海岸１－１６－１ ニューピア竹芝サウスタワービル 電話03（5402）8200　〒105－6891
大阪営業所／大阪市淀川区宮原３－５－３６ 新大阪第２森ビル 電話06（6350）6868㈹　〒532－0003

ADM3202/ADM3222/ADM1385—仕様

(V_{CC}＝＋3.3 V±0.3 V、C1～C4＝0.1 μF。特に指定のない限り、すべての仕様はT_{MIN}～T_{MAX}）

パラメータ	最小	標準	最大	単位	テスト条件/備考
DC特性					
動作電圧範囲	3.0	3.3	5.5	V	
V_{CC}電源電流		1.3	2.1	mA	無負荷時
		8	10	mA	R_L＝3 kΩをGND
シャットダウン消費電流		0.01	0.5	μA	
ロジック					
入力ロジック・スレッショルドLO、V_{INL}			0.8	V	T_{IN}
入力ロジック・スレッショルドHI、V_{INH}	2.0			V	T_{IN}
CMOS出力電圧LO、V_{OL}			0.4	V	I_{OUT}＝1.6 mA
CMOS出力電圧HI、V_{OH}	V_{CC}－0.6			V	I_{OUT}＝－1 mA
入力リーク電流		0.01	±1	μA	T_{IN}＝GNDからV_{CC}
出力リーク電流			±1	μA	レシーバ・ディスエーブル
RS-232レシーバ					
EIA-232入力電圧範囲	－30		＋30	V	
EIA-232入力スレッショルドLO	0.6	1.2		V	
EIA-232入力スレッショルドHI		1.6	2.4	V	
EIA-232入力ヒステリシス		0.4		V	
EIA-232入力抵抗	3	5	7	kΩ	
RS-232トランスミッタ					
出力電圧振幅（RS-232）	±5.0	±5.2		V	V_{CC}＝3.3 V。すべてのトランスミッタ出力にグラウンドに対して3 kΩの負荷を接続
出力電圧振幅（RS-562）	±3.7			V	V_{CC}＝3.0 V
トランスミッタ出力抵抗	300			Ω	V_{CC}＝0 V、V_{OUT}＝±2 V
RS-232出力短絡回路電流		±15		mA	
出力リーク電流			±25	μA	SD＝Low、V_{OUT}＝12 V
タイミング特性					
最大データ・レート	460			kbps	V_{CC}＝3.3 V、R_L＝3 kΩ ～ 7 kΩ、C_L＝50 pF ～ 1000 pF。1 Txスイッチング
レシーバ伝播遅延					
TPHL		0.4	1	μs	
TPLH		0.4	1	μs	
トランスミッタ伝播遅延		300	750	ns	R_L＝3 kΩ、C_L＝1000 pF
レシーバ出力イネーブル時間		200		ns	
レシーバ出力ディスエーブル時間		200		ns	
トランスミッタ・スキュー		30		ns	
レシーバ・スキュー		300		ns	
遷移領域スルーレート					＋3 V→－3 Vまたは－3 V→＋3 Vで測定、V_{CC}＝＋3.3 V
	6	10	30	V/μs	R_L＝3 kΩ、C_L＝1000 pF、T_A＝＋25℃
	4	10	30	V/μs	R_L＝3 kΩ、C_L＝2500 pF、T_A＝＋25℃

仕様は予告なしに変更することがあります。

絶対最大定格*
（特に指定のない限り、T_A＝＋25°C）

V_{CC} …… −0.3 V ～ ＋6 V
V＋ …… (V_{CC}−0.3 V) ～ ＋14 V
V− …… ＋0.3 V～−14 V
入力電圧
　T_{IN} …… −0.3 V～ (V＋、＋0.3 V)
　R_{IN} …… ±30 V
出力電圧
　T_{OUT} …… ±15 V
　R_{OUT} …… −0.3 V ～ (V_{CC}＋0.3 V)
短絡回路期間
　T_{OUT} …… 連続
消費電力
　消費電力 N-16 …… 450 mW
　（＋50°C以上は6 mW/°Cのディレーテング）
　　θ_{JA}、熱インピーダンス …… 117°C/W
消費電力 R-16 …… 450 mW
　（＋50°C以上は6 mW/°Cのディレーテング）
　　θ_{JA}、熱インピーダンス …… 158°C/W
消費電力 RU-16 …… 500 mW
　（＋50°C以上は6 mW/°Cのディレーテング）
　　θ_{JA}、熱インピーダンス …… 158°C/W
消費電力 R-18 …… 450 mW
　（＋50°C以上は6 mW/°Cのディレーテング）
　　θ_{JA}、熱インピーダンス …… 158°C/W
消費電力 RS-20 …… 450 mW
　（＋50°C以上は6 mW/°Cのディレーテング）
　　θ_{JA}、熱インピーダンス …… 158°C/W
消費電力 RU-20 …… 450 mW
　（＋50°C以上は6 mW/°Cのディレーテング）
　　θ_{JA}、熱インピーダンス …… 158°C/W
動作温度範囲
　産業（Aバージョン）…… −40°C ～ ＋85°C
保管温度範囲 …… −65°C ～ ＋150°C
リード温度（半田付け、10秒） …… ＋300°C
ESDレート …… <1500 V

* これは単にストレス定格を示したもので、これらの条件あるいは動作仕様に示した値を超える条件下でデバイスが機能することを意味するものではありません。絶対最大定格の条件に長時間さらされた場合、デバイスの信頼性は保証されません。

オーダー・ガイド

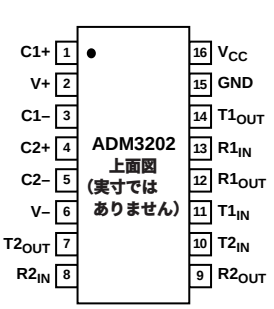
モデル	温度範囲	パッケージ・オプション*
ADM3202AN	−40°C ～ ＋85°C	N-16
ADM3202ARN	−40°C ～ ＋85°C	R-16A
ADM3202ARW	−40°C ～ ＋85°C	R-16
ADM3202ARU	−40°C ～ ＋85°C	RU-16
ADM3222AN	−40°C ～ ＋85°C	N-18
ADM3222ARW	−40°C ～ ＋85°C	R-18
ADM3222ARS	−40°C ～ ＋85°C	RS-20
ADM3222ARU	−40°C ～ ＋85°C	RU-20
ADM1385ARS	−40°C ～ ＋85°C	RS-20

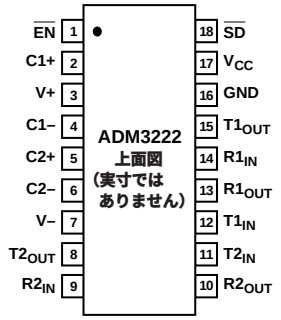
* N ＝ プラスチックDIP、R ＝ スモール・アウトライン、RS ＝ 縮小スモール・アウトライン、RU ＝ 薄型縮小スモール・アウトライン。

ピン機能説明

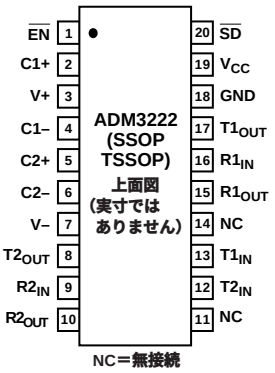
名称	機能
V_{CC}	電源入力：＋3.3 V±0.3 V
V＋	内部で発生する正電源（定格＋6 V）。
V−	内部で発生する負電源（定格−6 V）。
GND	グラウンド・ピン。0 Vに接続します。
C1＋、C1−	これらのピンの間に外部コンデンサ1を接続します。0.1 μFコンデンサが推奨されますが、47 μFまでのコンデンサを使用できます。
C2＋、C2−	これらのピンの間に外部コンデンサ2を接続します。0.1 μFコンデンサが推奨されますが、47 μFまでのコンデンサを使用できます。
Tx_{IN}	トランスミッタ（ドライバ）入力。これらの入力はTTL/CMOSレベルが可能です。
Tx_{OUT}	トランスミッタ（ドライバ）出力。これらはRS-232信号レベルです（標準±9 V）。
Rx_{IN}	レシーバ入力。これらの入力はRS-232信号レベルが可能です。各入力にGNDへの内部5 kΩプルダウン抵抗が接続されています。
Rx_{OUT}	レシーバ出力。これらはCMOS出力ロジック・レベルです。
$\overline{EN}$	（ADM3222）レシーバ・イネーブル、アクティブLO。LOのときに、レシーバ出力はイネーブルとなります。HIのときは3ステート状態になります。
$\overline{SD}$	（ADM3222）シャットダウン・コントロール。アクティブLO。LOのとき、チャージ・ポンプがシャットダウンされ、トランスミッタ出力はディスエーブルとなります。
$\overline{SD}$	（ADM1385）シャットダウン・コントロール。LOのとき、チャージ・ポンプがシャットダウンされ、トランスミッタとレシーバはすべてディスエーブルとなります。
DD	（ADM1385）ドライバ・ディスエーブル。LOのとき、チャージ・ポンプがターンオフされ、トランスミッタはディスエーブルとなります。レシーバはアクティブのままです。

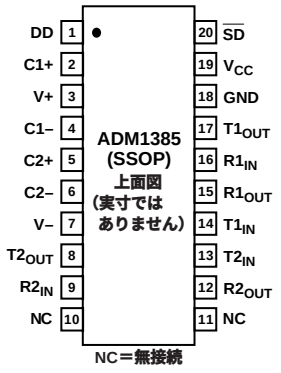
ピン配置DIP（N、Rパッケージ）





ピン配置DIP（RS、RUパッケージ）





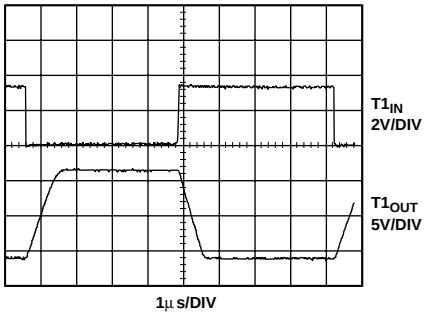


図7. 230 kbpsデータ伝送

機能の説明

ADM3202/ADM3222/ADM1385は、RS-232ライン・ドライバ/レシーバです。昇圧コンバータをレベル・シフト・トランスミッタとレシーバに組み合わせることにより、＋3.3 V単電源動作時にRS-232レベルを生成することができます。

CMOS技術を使用して最小限まで消費電力を抑えており、携帯用アプリケーションでバッテリ寿命を最大限延長可能です。

ADM3202/ADM3222/ADM1385は、AD230-AD241ファミリおよびその派生製品を変更、機能強化、および改良した製品です。基本的にプラグイン・コンパチブルで、ほぼ同じアプリケーションに対応します。

回路説明

内部回路は以下の3つの主要部分から構成されています。

1. チャージ・ポンプ電圧コンバータ
2. 3.3 Vロジック→EIA-232トランスミッタ
3. EIA-232→5 Vロジック・レシーバ

チャージ・ポンプDC-DC電圧コンバータ

チャージ・ポンプ電圧コンバータは、200 kHzオシレータとスイッチング・マトリックスから構成されています。このコンバータは＋3.3 Vレベルの入力から±6.6 Vの電圧を発生します。これは次頁に示すように、2段のスイッチド・キャパシタ技術を使用して達成しています。最初に、電荷を保存するコンデンサC1を使用して、＋3.3 V入力電源を倍の＋6.6 Vにします。次に、この＋6.6 VレベルをC2を使用して－6.6 Vに反転します。C3は図ではV_+とV_{CC}の間に接続されていますが、V_+とGNDの間に接続しても同様に有効です。

コンデンサC3とC4は、出力リップルを低減するために使用しています。この2つのコンデンサの値は重要ではなく、必要なら増やすことができます。コンデンサC3は図ではV_+とV_{CC}の間に接続されていますが、V_+とGNDの間に接続することも可能です。

必要に応じて、コンデンサC1－C4に、より大きな値のコンデンサ（10μFまで）を使用することができます。

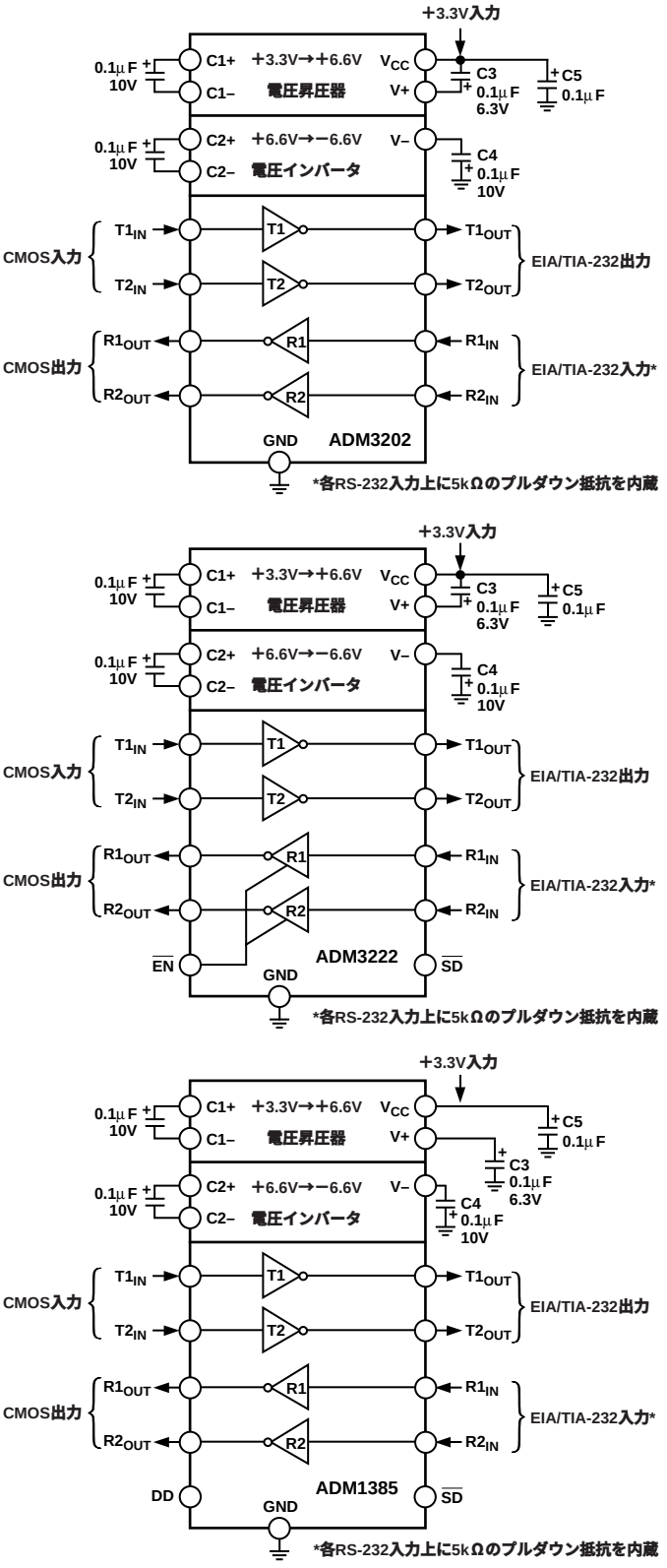


図8. 標準動作回路

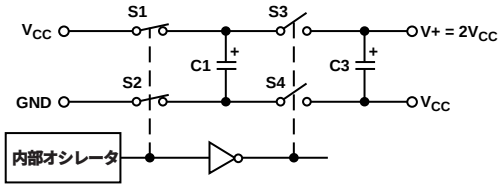


図9. チャージ・ポンプ電圧昇圧器

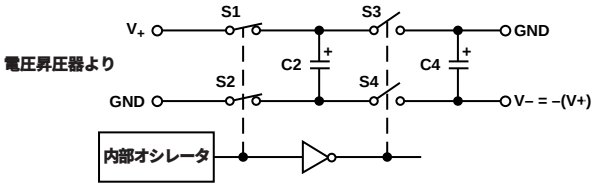


図10. チャージ・ポンプ電圧インバータ

トランスミッタ（ドライバ）部

ドライバは3.3 Vロジック入力レベルをRS-232出力レベルに変換します。V_{CC}＝＋3.3 VでRS-232負荷を駆動する場合、出力電圧振幅は±6 V（typ）です。

レシーバ部

レシーバはRS-232入力レベルを受け入れ、これを3 Vロジック出力レベルに変換する反転レベル・シフタです。入力はグラウンドに対して5 kΩプルダウン抵抗を内蔵しており、±30 Vまでの過電圧に対して保護されています。5 kΩの内部プルダウン抵抗により、無接続の入力は0 Vになります。したがって、無接続の入力あるいは入力がGNDに接続された場合、出力レベルはロジック1になります。

レシーバは0.4 Vのヒステリシス・レベルを持つシュミット・トリガ入力を備えています。これにより、ノイズが入力された場合や遷移時間が長い入力の場合にエラーを生じることがありません。

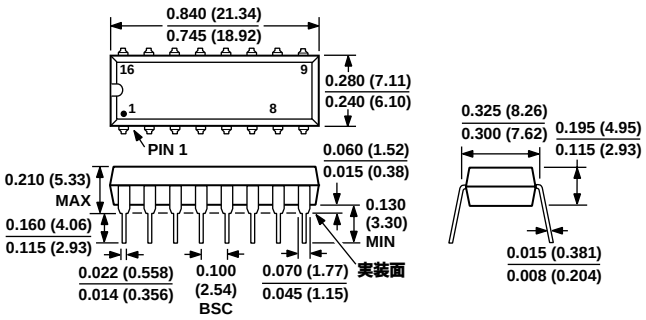
高いボーレート

ADM3202E/ADM3222Eは高いスルーレートを備え、EIA/RS-232E規格を上回る速度でデータ伝送を行うことができます。最悪の負荷条件下でも最高460 kbpsのデータ・レートまでRS-232電圧レベルが維持されます。このため、2台の端末間で高速データ・リンクを実現でき、また230 kbpsのデータ・レートを要求する新世代のISDNモデム規格にも最適です。スルーレートは、EMI干渉を最小限に抑えるために、内部で30 V/μs以下に制限されています。

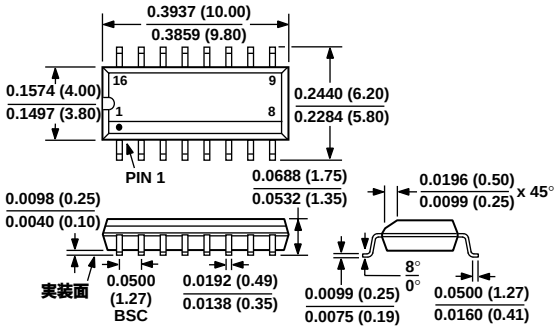
外形寸法

サイズはインチと（mm）で示します。

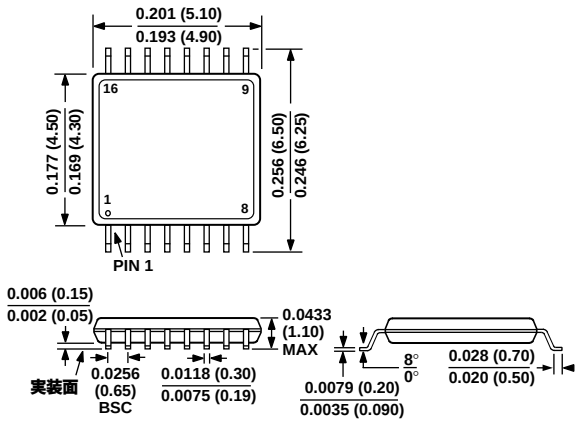
16ピン・プラスチックDIP（N-16）



16ピン・ナロー幅SOIC（R-16A）



16ピン薄型縮小スモール・アウトライン（TSSOP）（RU-16）



16ピン・ワイド幅SOIC（R-16）