簡易ADロガー(高速シリアル出力)(PIC12F1822)

概要

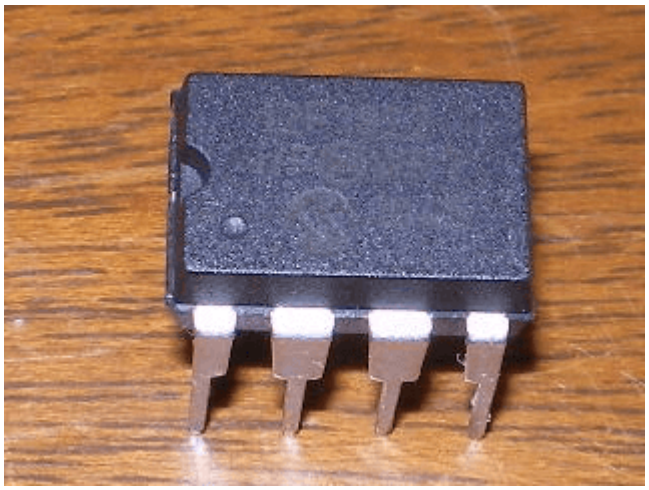
拡張ミッドレンジのコアを持つPICマイコンが格安で手に入るようになりました。 今まで、8ピンPICには、ミッドレンジタイプの「PIC12F683」を使用していましたが、機能強化され、且つ、価格が安くなった、拡張ミッドレンジタイプの「PIC12F1822」を使用した、シリアル出力タイプのADロガーを製作しました。

<仕様>

- サンプリング速度：1msec
- 通信速度：115200bps
- 入力チャネル数：1チャネル
- 入力電圧範囲：0V~5V
- 送信データのフォーマット：6バイト<数字4桁+CRLFコード>

※数字4桁は、A/D変換したデータ(A/D変換値)なので、受信側で電圧値に換算します。 ※電圧値は、A/D変換値×4.8828125mVで求めることができます。

<PIC12F1822の概観>



<PIC12F683とPIC12F1822の機能比較>

機能	PIC12F683(ミッドレンジタイプ)	PIC12F1822(拡張ミッドレンジタイプ)
クロック	最大20MHz	最大32MHz
内蔵クロック	最大8MHz	最大32MHz
動作電圧	2.0V~5.5V	1.8V~5.5V
プログラムメモリ	2Kワード	2Kワード
データメモリ	128バイト	128バイト
EEPROM	256バイト	256バイト
I/O	6	6
ADC	10ビット×4チャネル	10ビット×4チャネル
タッチセンサー	無し	4チャネル
コンパレータ	1チャネル	1チャネル
TIMER	8ビット×2個、16ビット×1個	8ビット×2個、16ビット×1個

USART	無し	1チャネル
MSSP	無し	1チャネル
CCP	1チャネル	1チャネル
価格	150円	80円

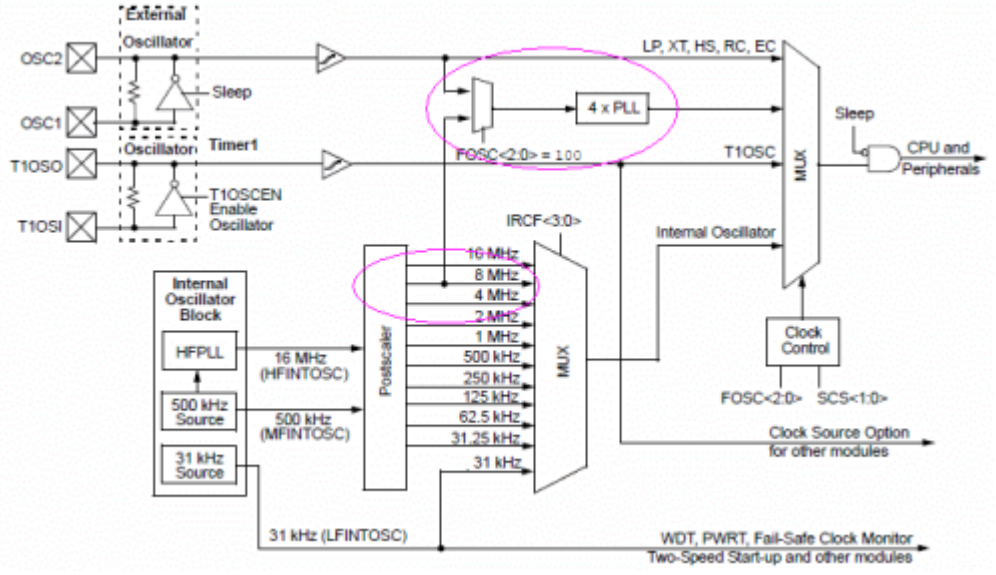
※価格は、2011年4月時点の秋月電子での販売価格です。(参考)

動作原理

1msec周期でA/D変換を行い、その結果を115200bpsの通信速度でシリアル出力します。

動作原理(ハードウェア)

◎クロック 外部クロックを使用せずに、内蔵クロックのみで32MHzの高速クロックを得ることが出来ます。



す。

◎CCP+TIMER1 CCPモジュールをコンペアモードで使用し、1msecの周期割り込みを発生させます。

◎ADC 入力されたアナログ信号(0V~5V)を、10ビットのデジタルデータに変換します。

動作原理(ソフトウェア)

◎メイン関数※main()

- クロックを32MHzに設定します。
- USART(ソフト)初期化関数を呼び出します。※通信速度=115200bps
- ADC初期化関数を呼び出します。
- LEDを10回点滅させます。
- CCP初期化関数を呼び出します。
- 割り込みを許可します。
- スイッチ(SW)が押下されていれば□LEDを点灯させます。

◎割り込み関数※interrupt()

- A/D変換を行い、A/D変換値を得ます。
- A/D変換値を4バイトの文字列に変換します。(0-1023)
- スイッチ(SW)が押下(オン)されていれば、文字列に変換したデータをシリアル出力します。

◎CCP初期化関数※init_ccp_compare()

- CCPモジュールをコンペアモードで初期化し、1msecの周期割り込みを発生させます。

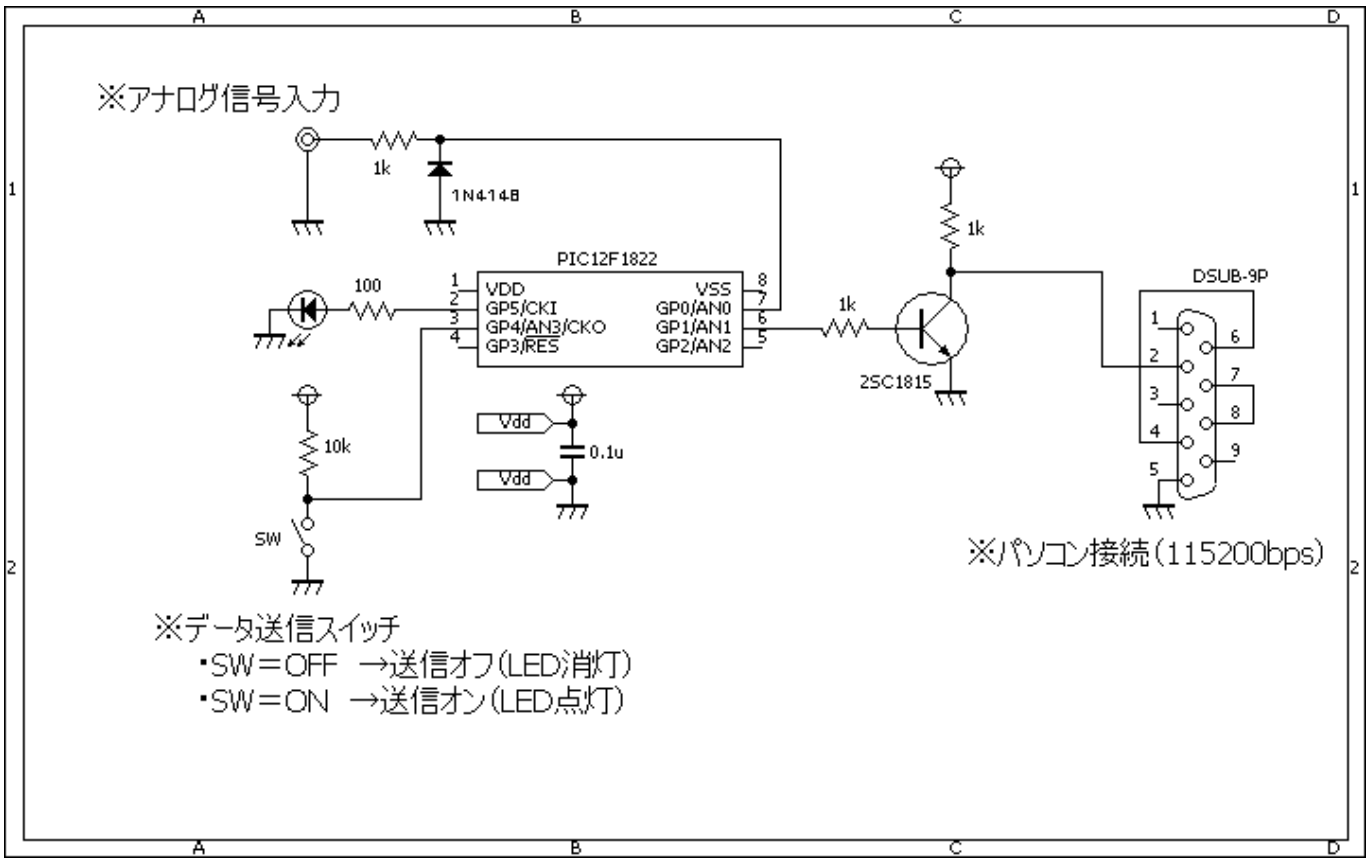
◎ADC初期化関数※adc_init_ex()

- ADCモジュールを初期化します。

◎ADC読み込み関数※adc_read_ex()

- ADCを開始し、変換結果を読み込みます。

回路図



ソースコード

ad_serial_logger_v1_00.c

```
//*****************************************************************
*
```

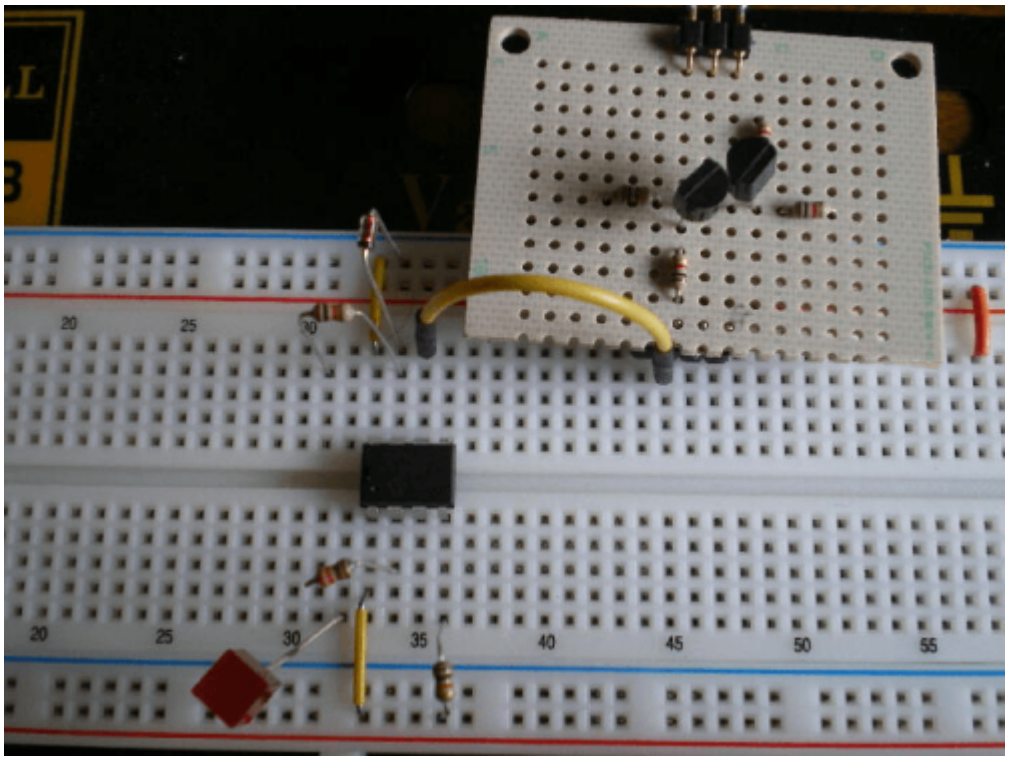
```
/*
    <簡易□□ロガー（シリアル出力）>
*/
//*********************************************************************
*
//■マクロ定義
sbit    LED     at      RA5_bit;
#define LED_ON  1
#define LED_OFF 0
//
sbit    SW      at      RA3_bit;
#define SW_ON   0
#define SW_OFF  1
//
#define BYTE    unsigned short
#define WORD    unsigned int
#define DWORD   unsigned long
//*********************************************************************
*
//■関数宣言
extern  void    main();
extern  void    interrupt();
extern  void    init_ccp_compare();
extern  WORD    adc_read_ex(BYTE channel);
extern  void    adc_init_ex();
//*********************************************************************
*
//■メイン関数です。
void    main()
{
        static  short   cnt;
        //
        OSCCON = 0b01110000;  //clock=32MHz
        TRISA  = 0b00011001;
        APFCON = 0b00100000;
        //
        Soft_UART_Init(&PORTA, 4, 1, 115200, 0);
        adc_init_ex();
        //
        for (cnt = 0; cnt < 10; cnt++) {
                LED = LED_ON;
                Delay_ms(100);
                LED = LED_OFF;
                Delay_ms(100);
        }
        //
        init_ccp_compare();
        //割り込みを許可します。
        INTCON.PEIE = 1;
        INTCON.GIE = 1;
        //
```

```
        while (1) {
                if (SW == SW_ON) {
                        LED = LED_ON;
                } else {
                        LED = LED_OFF;
                }
        }
}
//*********************************************************************
*
//■割り込み関数です。
double  ad;
char    *buf = "00000\r\n";
//
void    interrupt()
{
        PIR1.CCP1IF = 0;
        //
//        ad = adc_read_ex(0);
        ad = ADC_Get_Sample(0);
        //ad *= 4.8828125;  //性能改善のため電圧値への換算はパソコン側で行いま
す。
        WordToStr(ad, buf);
        if (SW == SW_ON) {
                Soft_UART_Write(buf[1]);
                Soft_UART_Write(buf[2]);
                Soft_UART_Write(buf[3]);
                Soft_UART_Write(buf[4]);
                Soft_UART_Write('\r');
                Soft_UART_Write('\n');
        }
}
//*********************************************************************
*
//■□□□初期化関数です□□1msecのインターバルタイマ発生用）
void    init_ccp_compare()
{
        // CCPの設定
        PIE1.CCP1IE = 1;
        PIR1.CCP1IF = 0;
        CCP1CON.CCP1M3 = 1;
        CCP1CON.CCP1M2 = 0;
        CCP1CON.CCP1M1 = 1;
        CCP1CON.CCP1M0 = 1;
        CCPR1L = 0x40;          // 1msec...(1÷32000000)*4*0x1F40
        CCPR1H = 0x1F;          // 2msec...(1÷32000000)*4*0x3E80
        // TIMER1の設定
        PIE1.TMR1IE = 0;
        PIR1.TMR1IF = 0;
        TMR1L = 0;
        TMR1H = 0;
```

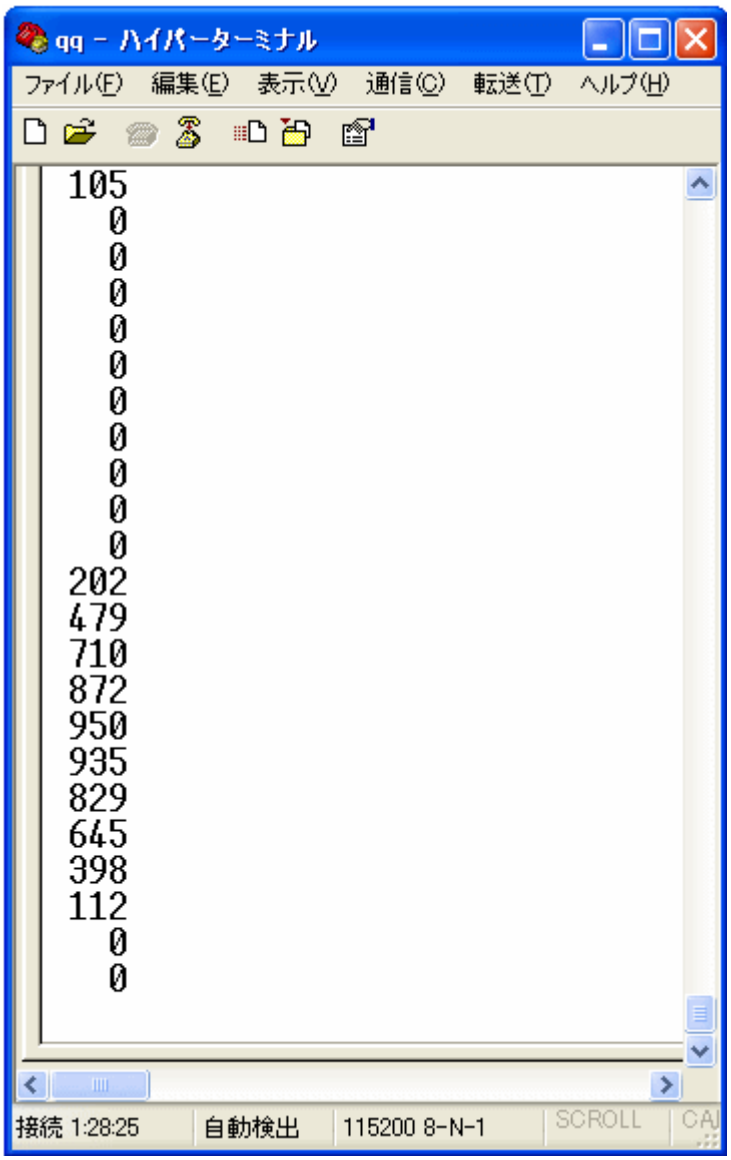
```
        T1CON.T1CKPS0 = 0;
        T1CON.T1CKPS1 = 0;
        T1CON.TMR1ON = 1;
}
//*********************************************************************
*
//■□□□を初期化する関数です。
void    adc_init_ex()
{
        ADC_Init();
        //
        ANSELA.ANSA4 = 0;
        ANSELA.ANSA2 = 0;
        ANSELA.ANSA1 = 0;
        ANSELA.ANSA0 = 1;
        ADCON1.ADFM = 1;
        ADCON1.ADCS2 = 1;
        ADCON1.ADCS1 = 1;
        ADCON1.ADCS0 = 0;
        ADCON1.ADPREF1 = 0;
        ADCON1.ADPREF0 = 0;
}
//*********************************************************************
*
//■□□□から読み込む関数です。
WORD    adc_read_ex(BYTE channel)
{
        static  int     ad;
        //
        ADCON0.GO = 0;
        ADCON0.ADON = 1;
        ADCON0.CHS4 = channel.B4;
        ADCON0.CHS3 = channel.B3;
        ADCON0.CHS2 = channel.B2;
        ADCON0.CHS1 = channel.B1;
        ADCON0.CHS0 = channel.B0;
        ADCON0.GO = 1;
        while (ADCON0.GO == 1) {
        }
        ad = ADRESH & 0b00000011;
        ad <<= 8;
        ad = ad | ADRESL;
        return (ad);
}
//*********************************************************************
*
void    switch_on_check()
{
        while (Button(&PORTA, 3, 1, 0) == 0)
                ;
        while (Button(&PORTA, 3, 1, 1) == 0)
```

```
                    ;
}
//*****************************************************************************
*
```

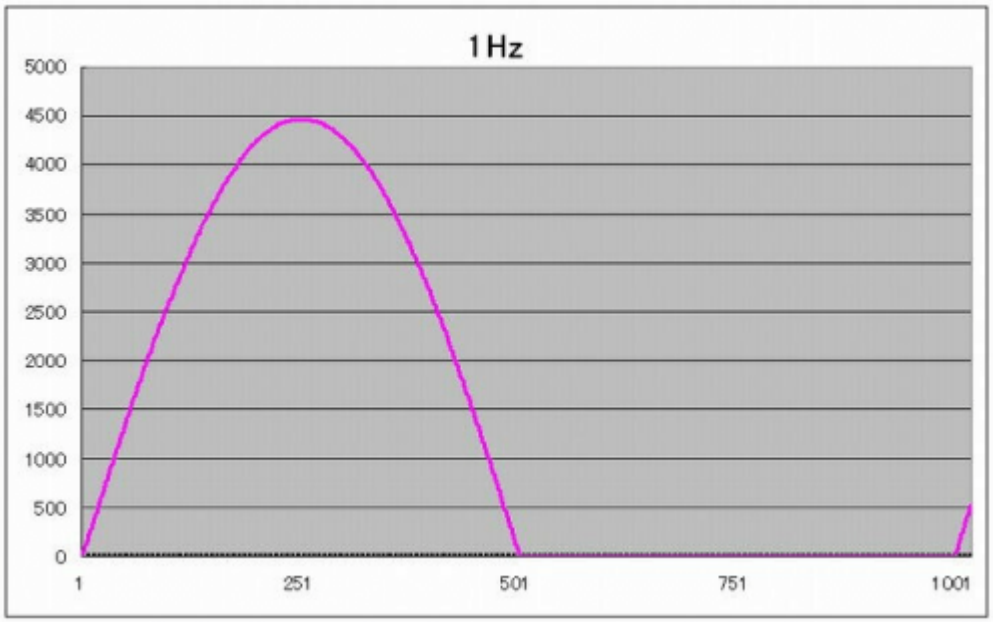
動作確認

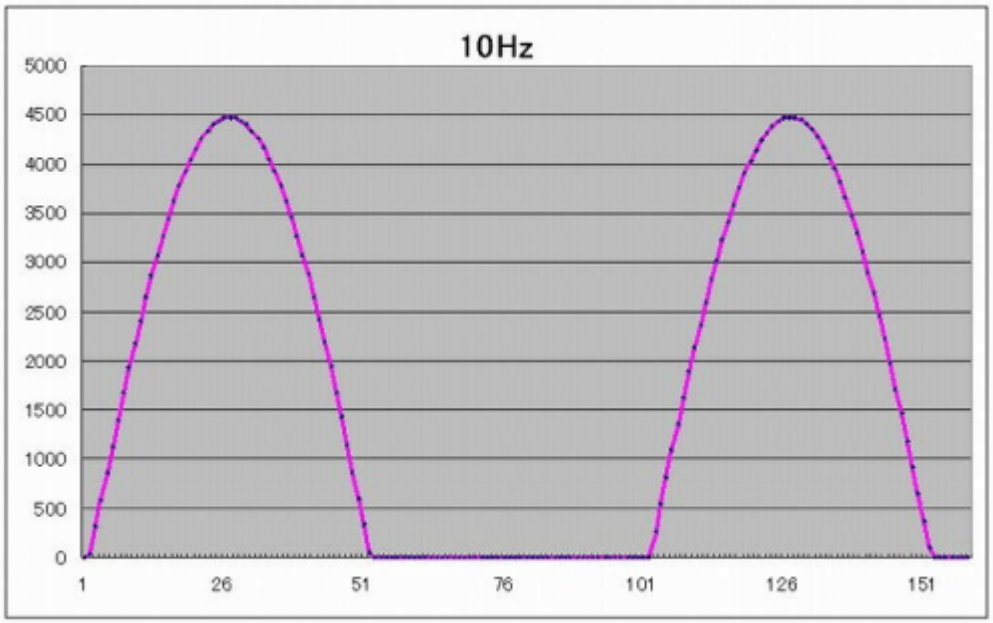


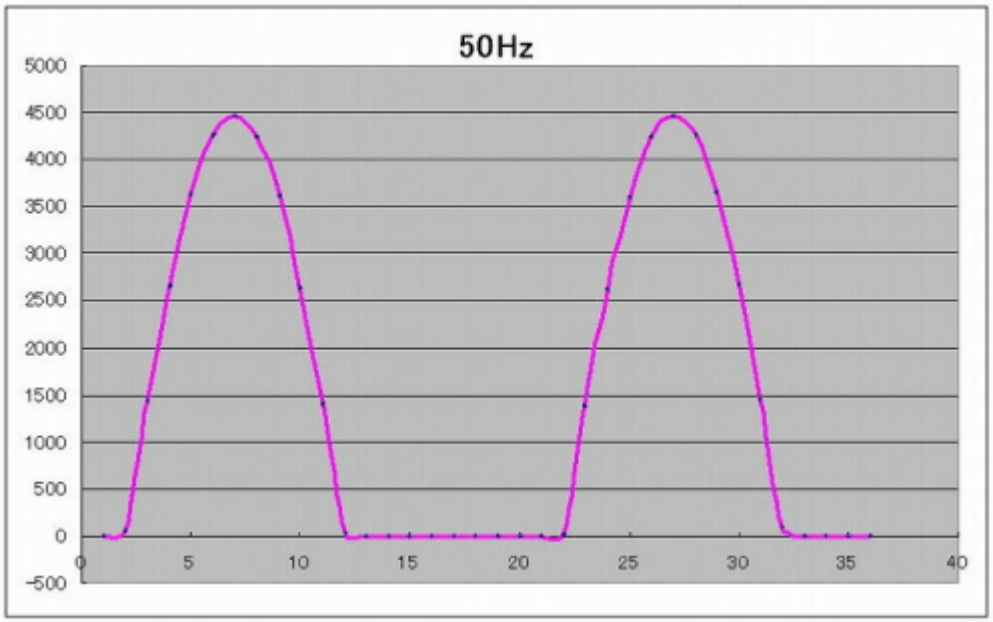
パソコン側の通信ソフト(ハイパーターミナルなど)で、データを受信します。 その結果をキャプチャしてファイルに保存します。(転送メニューのテキストのキャプチャ機能を使用します)

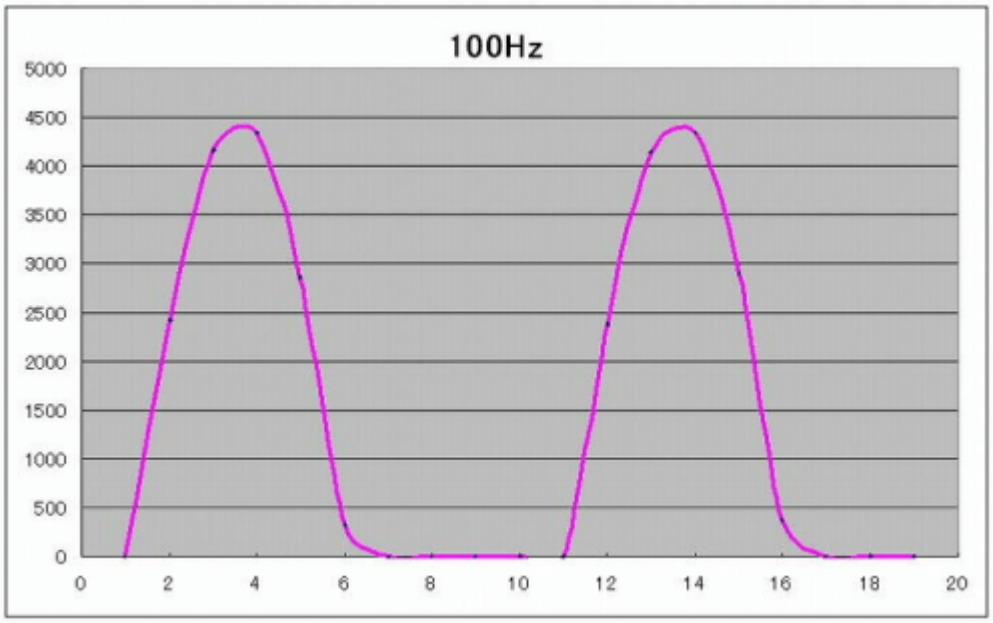


キャプチャしたデータをExcelに取り込み、電圧値に換算した結果をグラフ表示させて見ました。 入力信号は、1Hz□10Hz□50Hz□100Hzの正弦波です。




10Hz
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1
26
51
76
101
126
151


50Hz
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-500
0
5
10
15
20
25
30
35
40


100Hz
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
2
4
6
8
10
12
14
16
18
20