アナログデータロガー(複数PIC使用)

概要

最近の殆どのPICにはA/Dコンバータが標準で実装されているので、簡単なロガーであれば容易に実現可能です。しかし、チャネル数としては、現時点で入手が容易な、40ピンのPIC18F4550でも、13チャネルが最大です。

測定の分野によっては、もう少しチャネル数が必要となる場合があります。

そこで今回は、複数のPICを組み合わせることにより、最大49チャネルのアナログデータロガーを製作してみました。

<仕様>

- チャネル数は、最小7チャネル、最大49チャネルとします。(7チャネル×7PIC)
- A/D変換の精度は10ビットとします。
- 測定データは、パソコンにRS232C経由で転送します(9600bps)
- PICを最大、8個使用し、全体の制御は、MASTER/SLAVE方式(ポーリング)とします。
- 一つのPICに、MASTERの機能とSLAVEの機能を実装し、起動時に選択可能とします。

動作原理

MASTER配下には、最大7台のSLAVEをシリアル(USART)で接続します。各SLAVEには、アドレス(1~7)が割り当てられます。(各SLAVEのスイッチで設定する)

- 全てのSLAVEのRx(受信信号)は一つにつながれMASTERのTx(送信信号)に接続されます。
  ※つまり全てのSLAVEが、MASTERからの制御コードを同時に受信することができます。

- 全てのSLAVEのTx(送信信号)は一つにつながれMASTERのRx(受信信号)に接続されます。
  ※PICのTxピンは、出力モードなので、直接にTx同士を繋ぐとPICが破壊されてしまいます。
  ※そこで、トランジスタによる非反転回路をかませることにより、破壊を防止します。
  ※回路の簡素化のために、1個のトランジスタ(2SA1015)による非反転回路(1)を採用しました。

<MASTER>

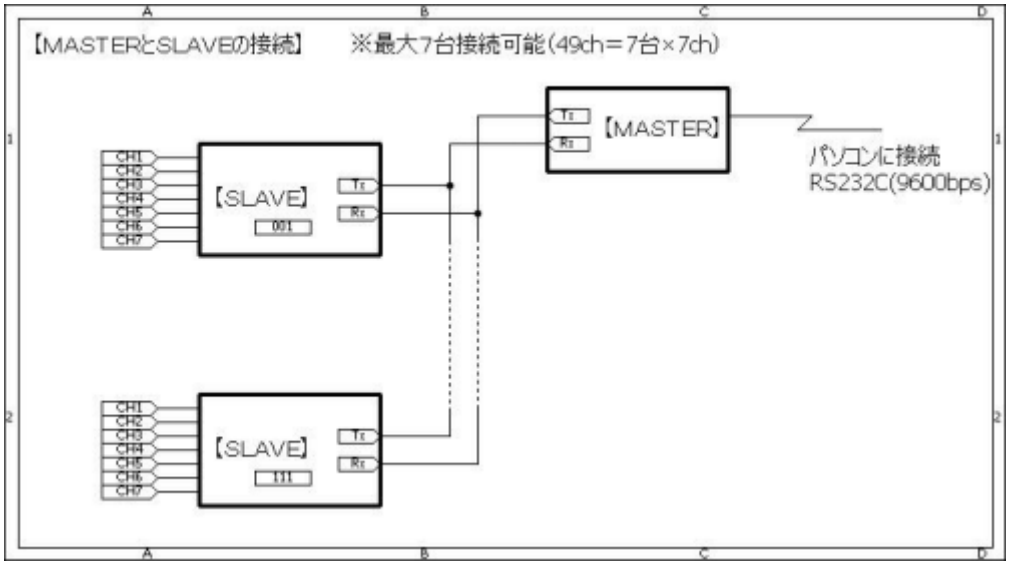
1. スレーブ(アドレス=1)に対して、データ要求の制御コードを送信します(9600bps)
2. スレーブ(アドレス=1)よりETB(End of Transmission Block)コードが来るまで、データを受信します。
3. 受信したデータを、パソコンにRS232C経由で送信します(9600bps)
4. 受信したデータを、LCDに表示します。
5. 1.~4.の処理を、スレーブ数分だけ繰り返します。(スレーブ数は、スイッチで指定します)

<SLAVE>

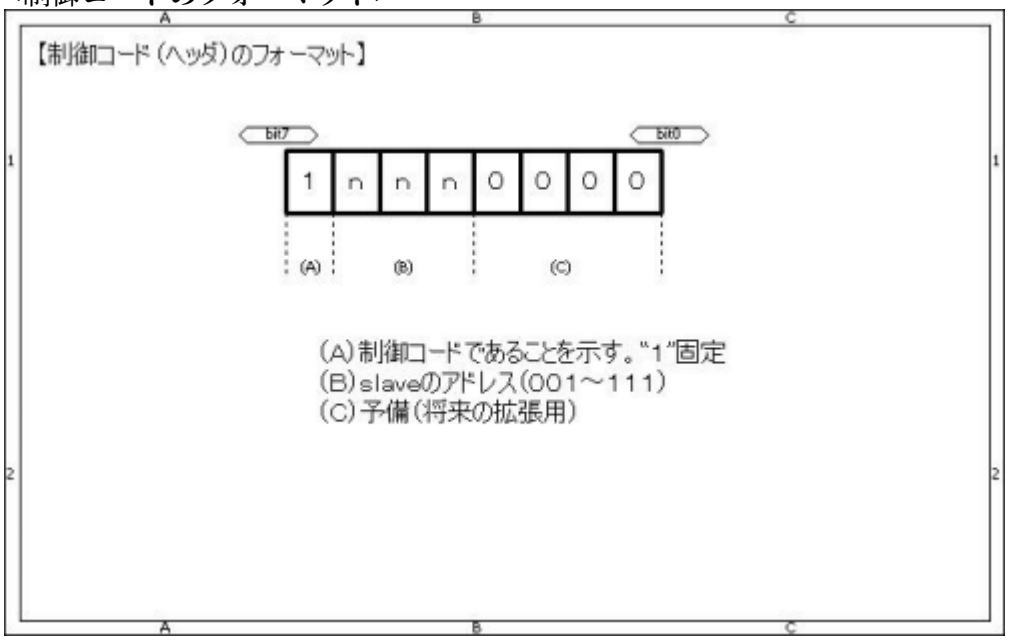
1. MASTERからの、制御コードを受信します。
2. 制御コードが、自分宛で無ければ無視します。(スレーブのアドレスは、スイッチで指定します)
3. 制御コードが、自分宛であればCH1~CH7までをA/D変換します。(このときLEDを点灯させます)
4. 変換したデータを、MASTERに送信します。

5. データの最後を示す、ETBコードを、MASTERに送信します。
6. 1.に戻ります。

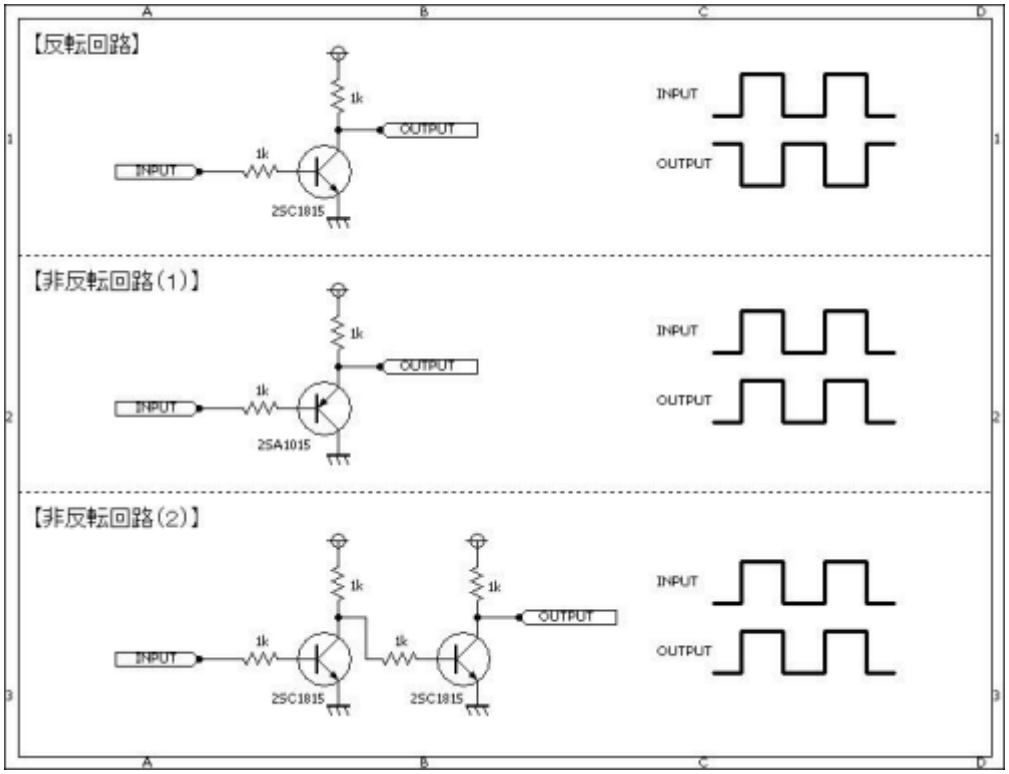
<MASTERとSLAVEの接続>



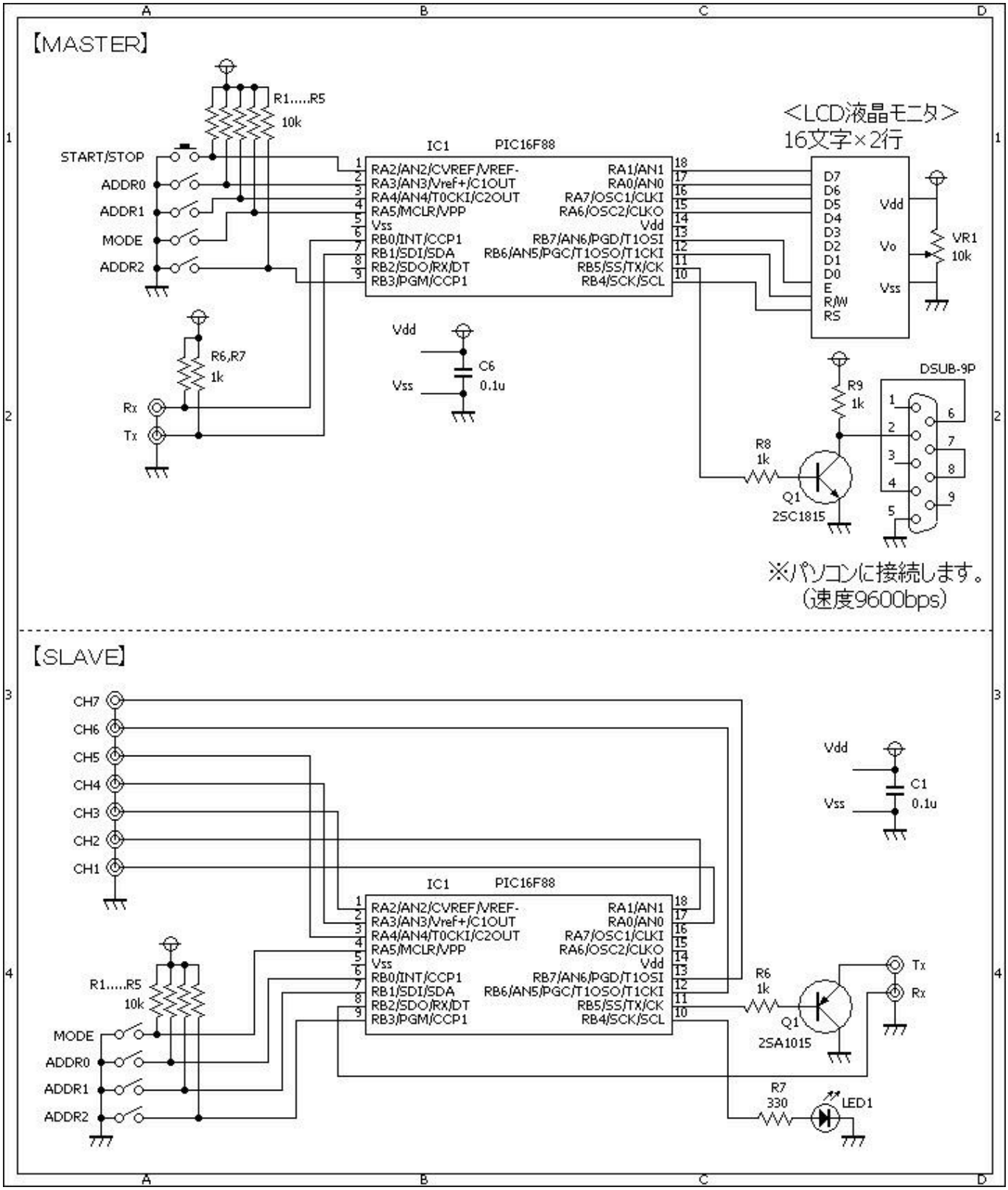
<制御コードのフォーマット>



<トランジスタによる非反転回路>※PICのTxピン同士の接続のために必要となります。


【反転回路】
1k
OUTPUT
INPUT
1k
2SC1815
INPUT
OUTPUT
【非反転回路(1)】
1k
OUTPUT
INPUT
1k
2SA1015
INPUT
OUTPUT
【非反転回路(2)】
1k
1k
OUTPUT
INPUT
1k
1k
2SC1815
2SC1815
INPUT
OUTPUT

回路図



ソースコード

ad_logger_multi_pic.c

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

```
/*
□□□□□ロガー（マルチ□□□□□
*/
//*****************************************************************************
*

//master & slave
#define       SW_MODE        PORTA.F5

//master
#define       LCD_D6         PORTA.F0
#define       LCD_D7         PORTA.F1
#define       SW             PORTA.F2
#define       SW_ADDR0M     PORTA.F3
#define       SW_ADDR1M     PORTA.F4
#define       LCD_D5         PORTA.F6
#define       LCD_D4         PORTA.F7
#define       Rx_1          PORTB.F0
#define       Tx_1          PORTB.F1
#define       Rx_2          PORTB.F2
#define       SW_ADDR2M     PORTB.F3
#define       LCD_RS         PORTB.F4
#define       Tx_2          PORTB.F5
#define       LCD_RW         PORTB.F6
#define       LCD_E         PORTB.F7

//slave
#define       AN0             PORTA.F0
#define       AN1             PORTA.F1
#define       AN2             PORTA.F2
#define       AN3             PORTA.F3
#define       AN4             PORTA.F4
#define       NON_1          PORTA.F6
#define       NON_2          PORTA.F7

#define       SW_ADDR0S     PORTB.F0
#define       SW_ADDR1S     PORTB.F1
#define       Rx              PORTB.F2
#define       SW_ADDR2S     PORTB.F3
#define       LED             PORTB.F4
#define       Tx              PORTB.F5
#define       AN5             PORTB.F6
#define       AN6             PORTB.F7

#define       CR              0x0D
#define       LF              0x0A
#define       ETB             0x23

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

```
static char    msg[48], buf[16];

//*****************************************************************************
*

void   Usart_Write_Str(unsigned short *pData)
{
    while (*pData != 0x00) {
        Usart_Write(*pData);
        pData++;
        Delay_ms(10);
    }
}

//*****************************************************************************
*

void   master()
{
    static    unsigned    short    rd, *rec, len, addr, addr_max, hd;
    static    unsigned    int        cnt;
    //
    addr_max = 0;
    addr_max.F0 = SW_ADDR0M;
    addr_max.F1 = SW_ADDR1M;
    addr_max.F2 = SW_ADDR2M;
    //
    cnt = 0;
    //
    while (1) {
        if (SW == 0) {
            while (SW == 0) {
                Delay_ms(100);
            }
            //
            while (1) {
                for (addr = 1; addr <= addr_max; addr++) {
                    hd = addr << 4;
                    hd.F7 = 1;
                    Soft_Uart_Write(hd);
                    //
                    len = 0;
                    while (1) {
                        rd = Soft_Uart_Read(rec);
                        if (rd == ETB) {
                            msg[len] = 0x00;
                            break;
                        }
                        msg[len] = rd;
                        len++;
```

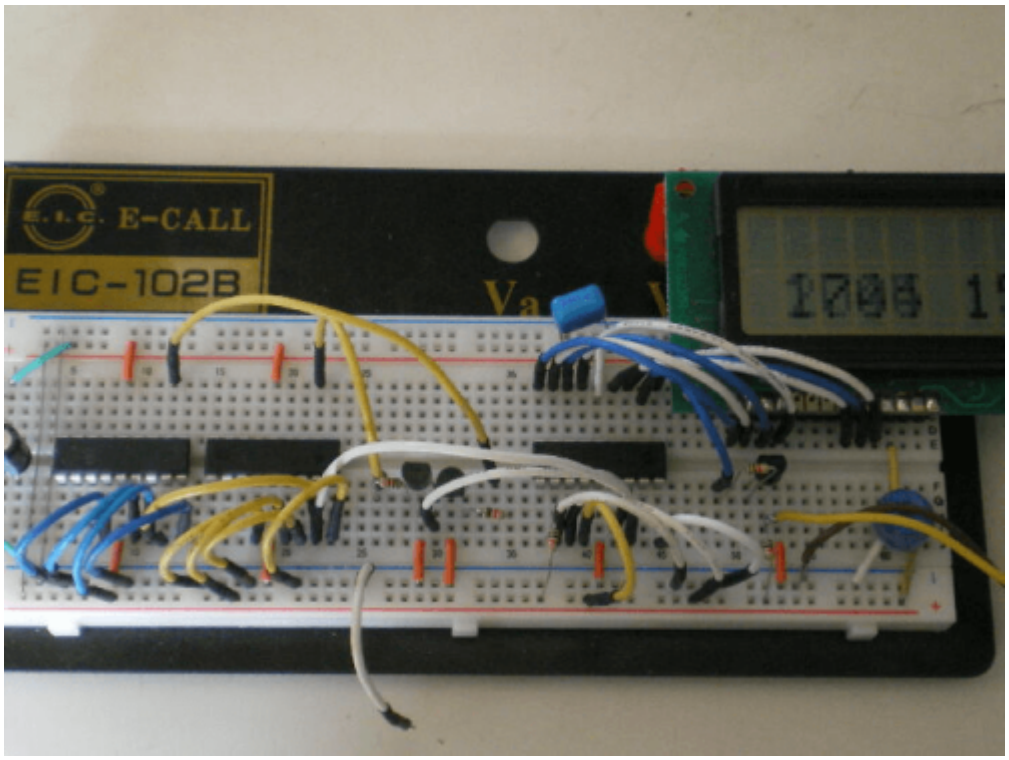
```
                    }
                    Usart_Write_Str(msg);
                    //
                    Lcd_Custom_Chr(1, 9 + addr, 0xFF);
                    //
                    msg[16] = 0x00;
                    Lcd_Custom_Out(2, 1, msg);
                }
                Usart_Write(CR);
                Usart_Write(LF);
                //
                cnt++;
                WordToStr(cnt, buf);
                Lcd_Custom_Out(1, 1, buf);
                Lcd_Custom_Out(1, 10, "       ");
                //
                if (SW == 0) {
                    while (SW == 0) {
                        Delay_ms(100);
                    }
                    cnt = 0;
                    break;
                }
                //
                Delay_ms(100);
            }
        }
    }
}

//*********************************************************************
*

void   slave()
{
    static    unsigned    short    addr, hd, rd;
    static    double                ad;
    //
    addr = 0;
    addr.F0 = SW_ADDR0S;
    addr.F1 = SW_ADDR1S;
    addr.F2 = SW_ADDR2S;
    hd = addr << 4;
    hd.F7 = 1;
    //
    LED = 0;
    while (1) {
        //ヘッダーの受信
        if (Usart_Data_Ready() == 0)
            continue;
        rd = Usart_Read();
```

```
        if (rd.F7 != 1)
            continue;
        if ((rd & 0xF0) != (hd))
            continue;
        //
        LED = 1;
        //
        ad = Adc_Read(0);
        ad = ad * 4.8828125;
        WordToStr(ad, msg);
        Usart_Write_Str(msg);
        //
        ad = Adc_Read(1);
        ad = ad * 4.8828125;
        WordToStr(ad, msg);
        Usart_Write_Str(msg);
        //
        ad = Adc_Read(2);
        ad = ad * 4.8828125;
        WordToStr(ad, msg);
        Usart_Write_Str(msg);
        //
        ad = Adc_Read(3);
        ad = ad * 4.8828125;
        WordToStr(ad, msg);
        Usart_Write_Str(msg);
        //
        ad = Adc_Read(4);
        ad = ad * 4.8828125;
        WordToStr(ad, msg);
        Usart_Write_Str(msg);
        //
        ad = Adc_Read(5);
        ad = ad * 4.8828125;
        WordToStr(ad, msg);
        Usart_Write_Str(msg);
        //
        ad = Adc_Read(6);
        ad = ad * 4.8828125;
        WordToStr(ad, msg);
        Usart_Write_Str(msg);
        //
        Usart_Write(ETB);
        //
        LED = 0;
    }
}

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

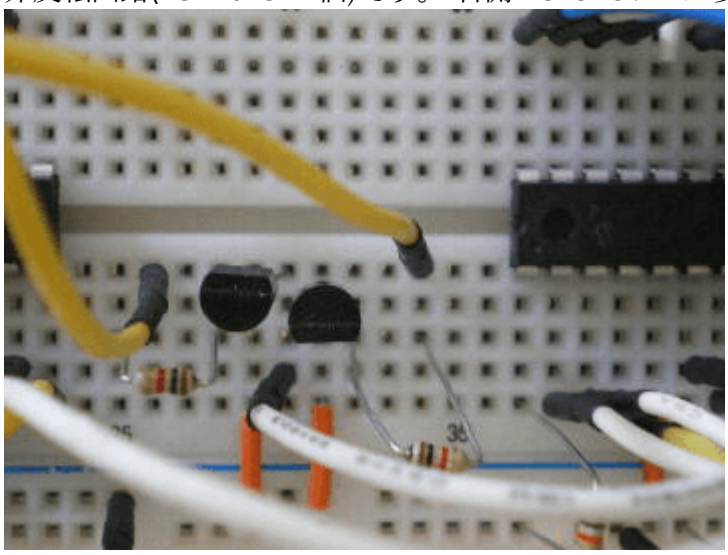
```
void   main()
{
    static    short    mode, cnt;
    //
    OSCCON = 0b01110000;
    CMCON  = 0b00000111;
    mode = SW_MODE;
    if (mode == 1) {
        ANSEL  = 0b00000000;
        TRISA  = 0b00111100;
        TRISB  = 0b00001101;
    } else {
        ANSEL  = 0b01111111;
        TRISA  = 0b11111111;
        TRISB  = 0b11001111;
    }
    //USARTの初期化
    Usart_Init(9600);
    if (mode == 1) {
        Soft_Uart_Init(PORTB, 0, 1, 9600, 0);
    }
    //
    if (mode == 1) {    //masterモード
        Lcd_Custom_Config(&PORTA, 1, 0, 7, 6, &PORTB, 4, 6, 7);
        Lcd_Custom_Cmd(LCD_CURSOR_OFF);
        Lcd_Custom_Cmd(LCD_CLEAR);
        Lcd_Custom_Out(1, 1, "ad-logger(49ch)");
        Lcd_Custom_Out(2, 1, "      multi pic");
        Delay_ms(1000);
        Lcd_Custom_Cmd(LCD_CLEAR);
        //
        master();
    } else {    //slaveモード
        for (cnt = 0; cnt< 10; cnt++) {
            LED = 1;
            Delay_ms(50);
            LED = 0;
            Delay_ms(50);
        }
        //
        slave();
    }
}

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

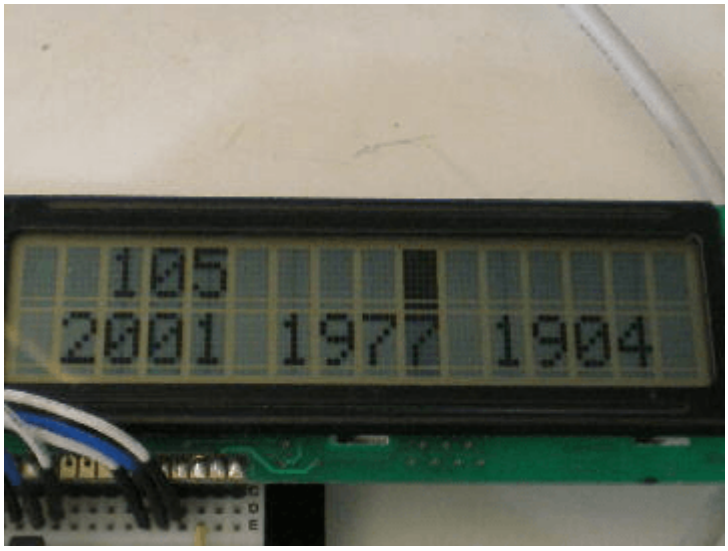
動作確認

左側から□SLAVE(2)□SLAVE(1)□MASTER□LCDです。 この回路では□SLAVEを2台にしました。 従って、14チャネルのアナログデータをロギングすることが出来ます□ MASTER側のスイッチ(ADDR0~ADDR2)で、SLAVEが2台接続されていることを指定します。

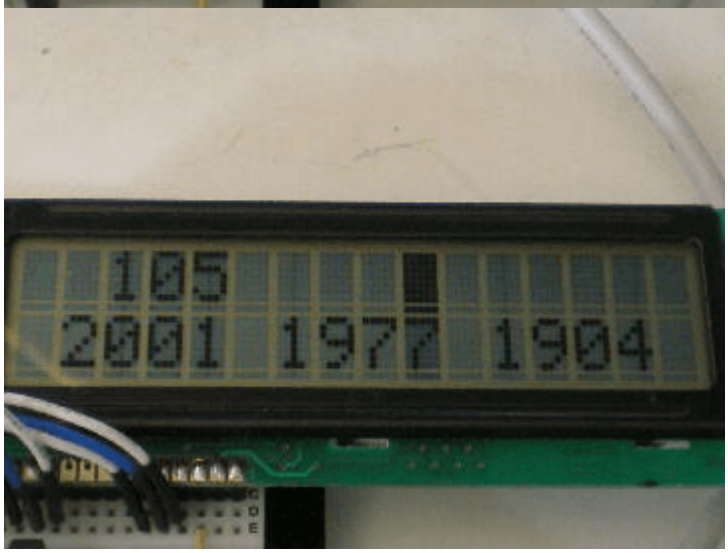


左側:非反転回路(2SA1015×2個)です。 右側:RS232Cレベル変換回路(2SC1815)とDSUB-9Pのコネクタです。

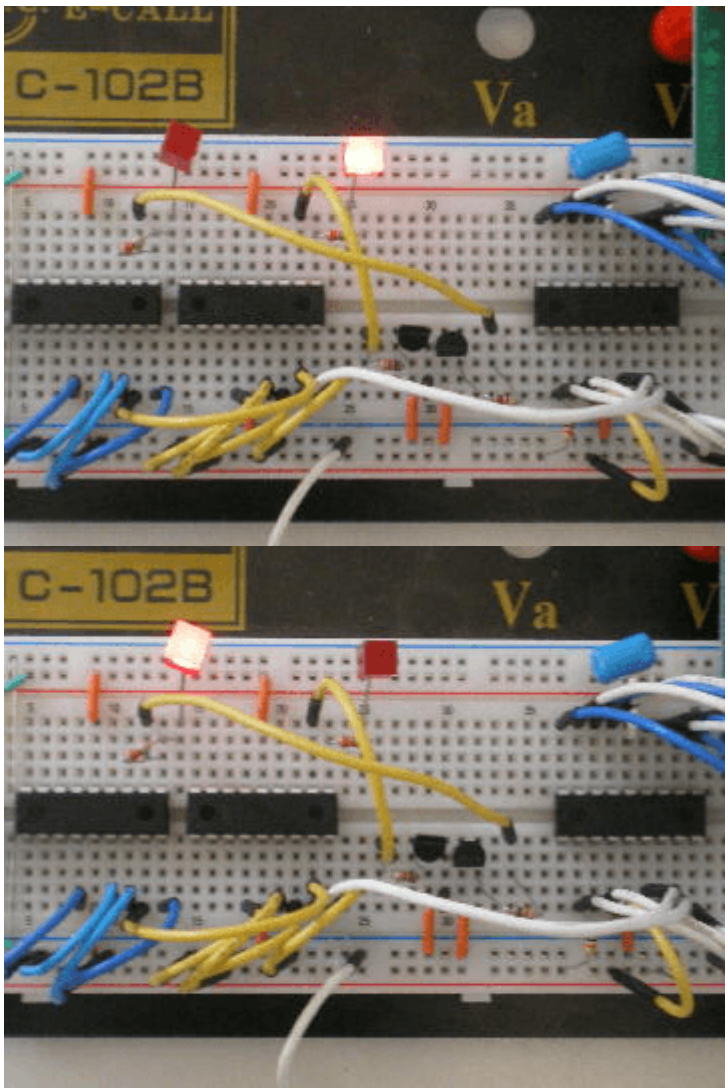




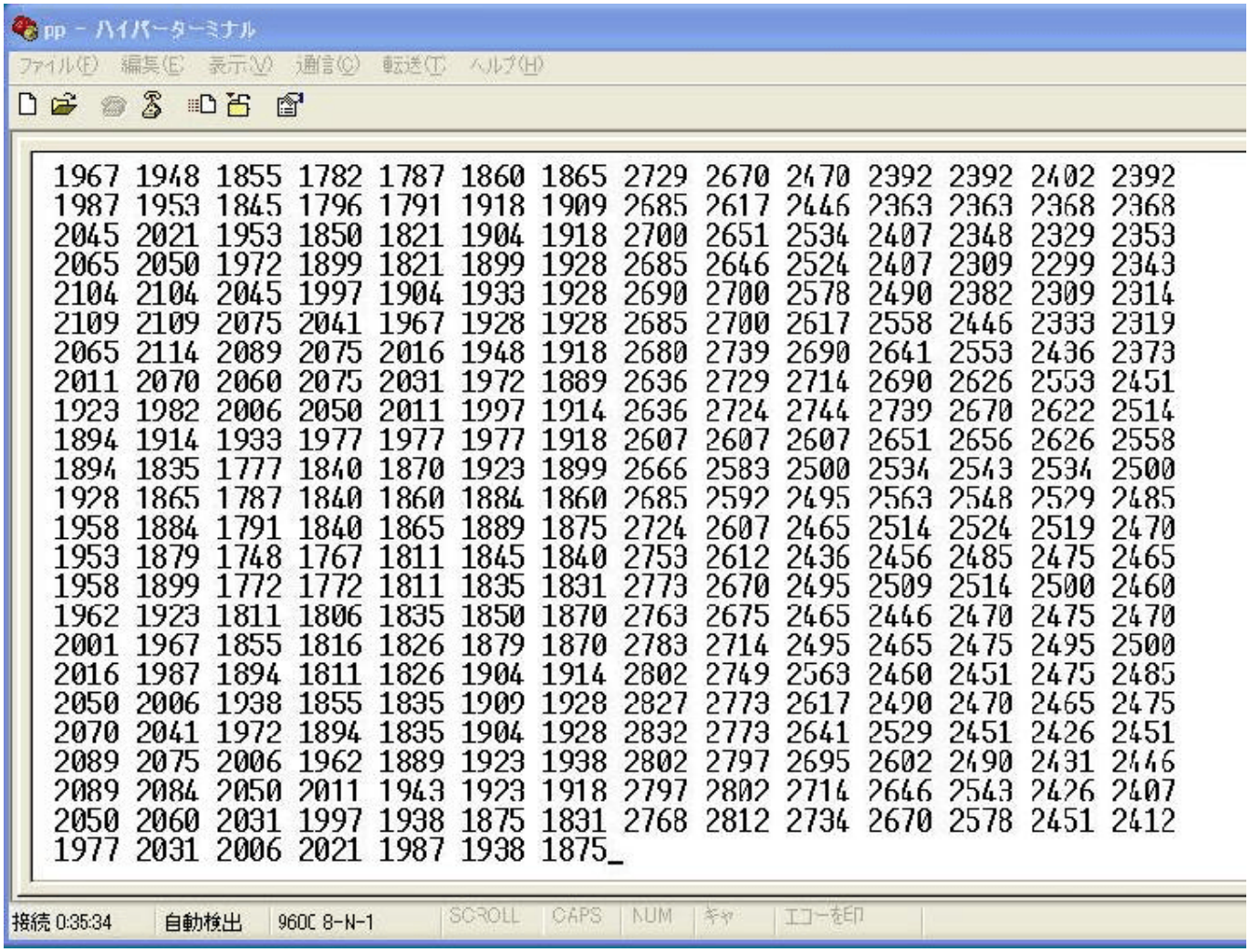
MASTER側のLCDの表示内容です LCDの左上は、SLAVEからのデータの受信カウント数です LCDの右上は、どのSLAVEとやり取りをしているかを示すマークです SLAVE数に応じて増えていきます LCDの2行目は、SLAVEから送られてくるデータ(7CH分)の内の、CH1~CH3の値です。



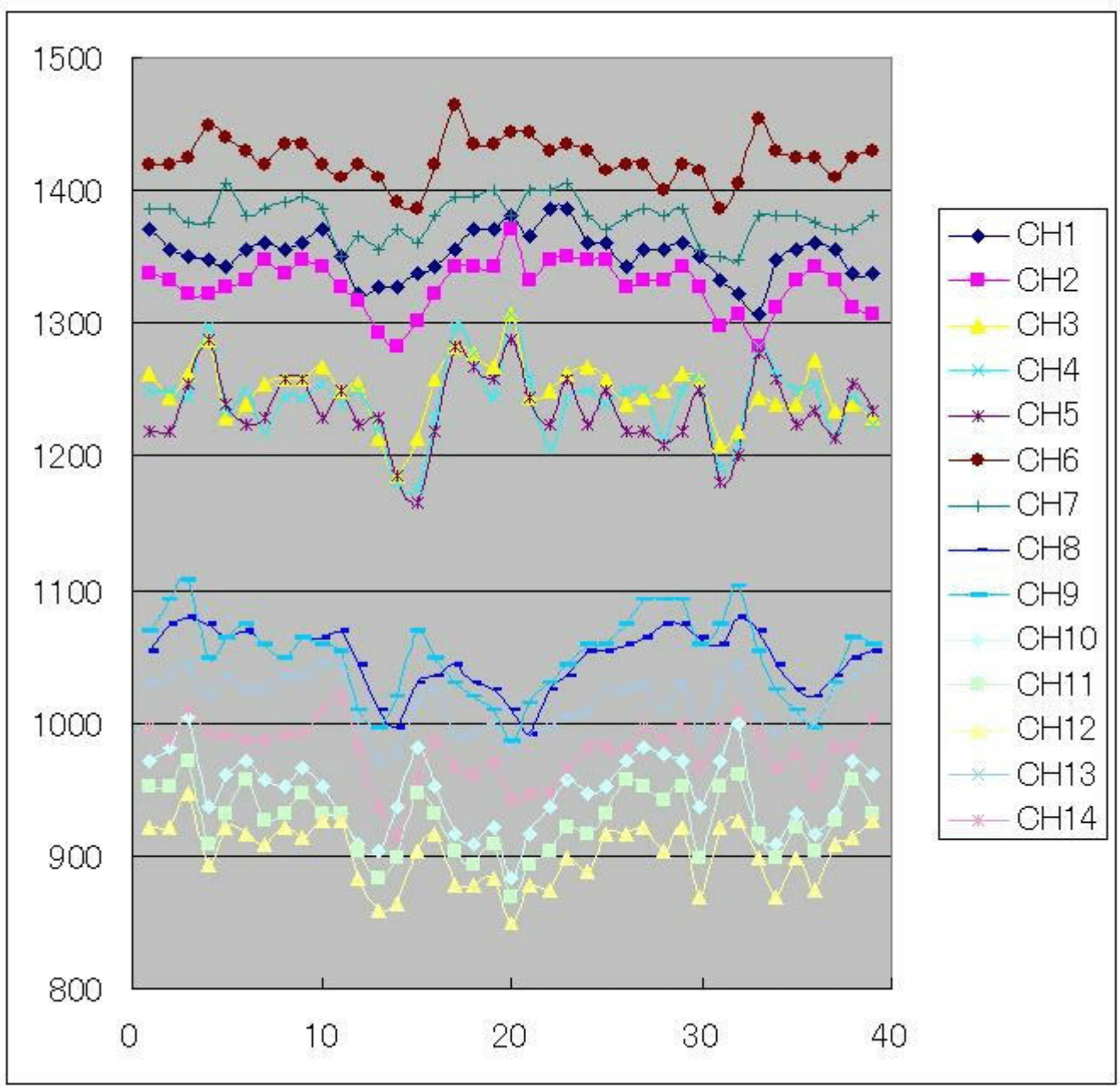
SLAVE側の様子です LEDが点灯しているSLAVEが、MASTERとやり取りをしています。



MASTERが、パソコンにデータを送っている様子です。 ハイパーターミナルで、受信データを表示させて見ました。左側からCH1~CH14です。



受信したデータを、Excelでグラフ表示させて見ました。 入力がオープン状態なので、ノイズ成分が含まれています。



如何ですか? 制御コードには、予備用のビットがまだありますので□SLAVE数を増やしたり、機能コードを追加することにより、いろいろな拡張が考えられます。