

簡易シリアルバッファ

概要

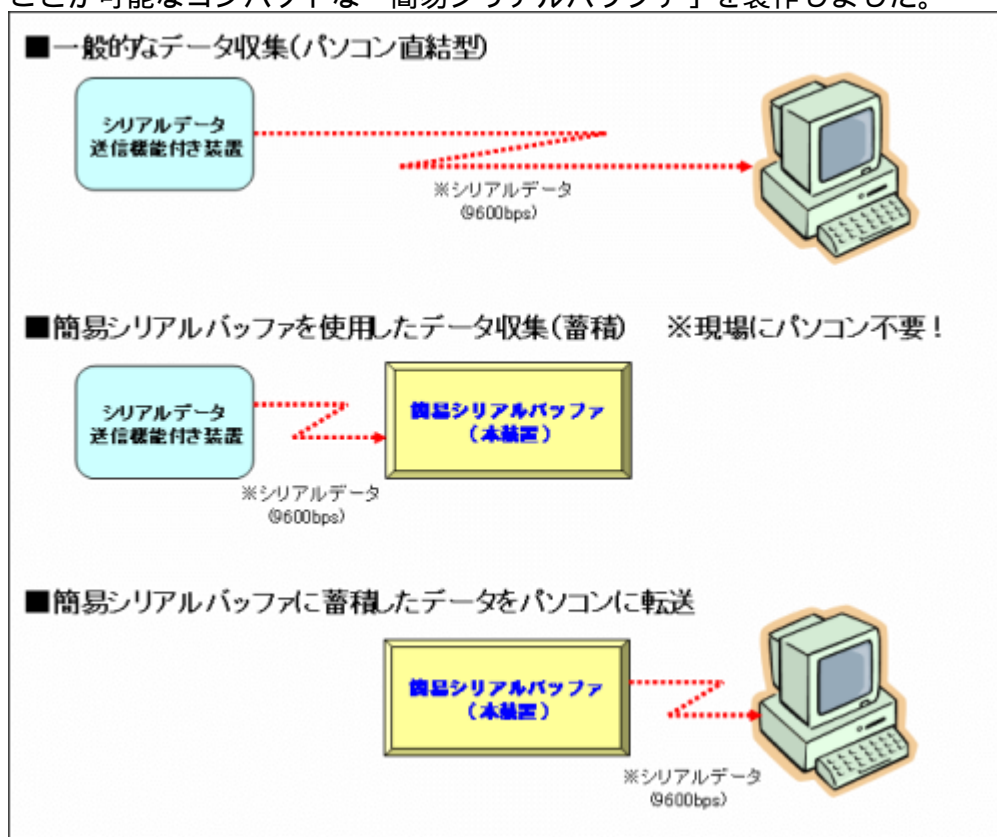
市販測定器や自作測定器の中には、測定したデータをRS232C経由でパソコンに送信する機能を持ったものがあります。

しかし、測定現場によっては、湿度が高い、粉塵が多い、設置場所が狭いなどのために、パソコンを持ち込むことが困難な場合があります。

そこで今回は、

- 測定時には、測定器から送信されてくるデータを一旦バッファリングし、 パソコン不要
- 測定完了後には、バッファリングしたデータを、パソコン側にデータ転送する。 測定器不要

ことが可能なコンパクトな「簡易シリアルバッファ」を製作しました。



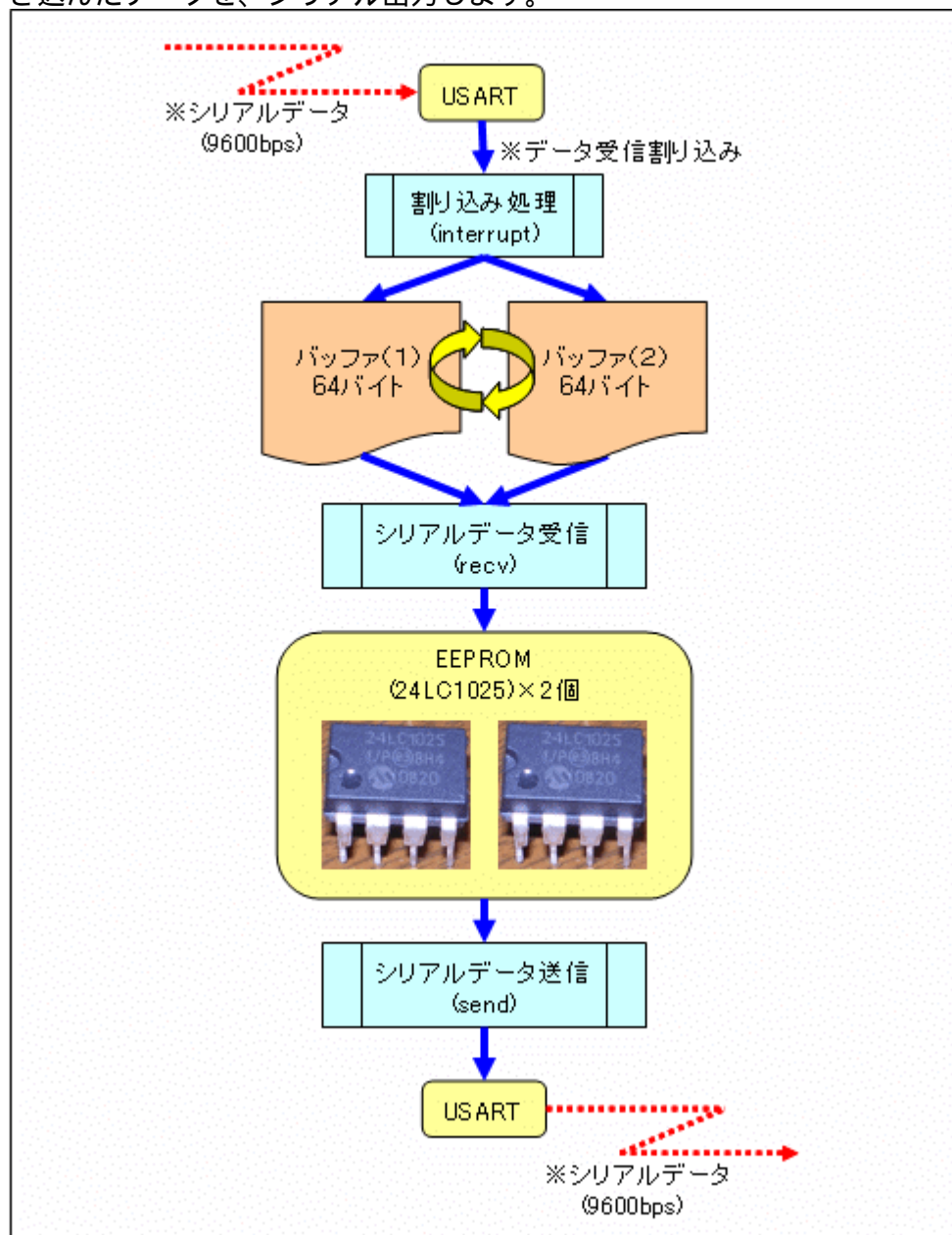
<仕様>

- バッファリング容量□256kバイト(正確には、262,144バイト)
- 通信速度□9600bps
- シリアル信号□TTLレベル(対PIC)□RS232Cレベル(対パソコン)

動作原理

受信データのバッファリング 受信したシリアルデータを、ダブルバッファに交互に書き込み、満杯になったバッファの内容を、随時EEPROMに書き込みます。 バッファリングデータの送信 EEPROMに書

き込んだデータを、シリアル出力します。



動作原理(ハードウェア)

◎USARTモジュール(受信)

- シリアルデータを受信し、割り込みを発生させます。

ダブルバッファ(メモリ)

- 受信したデータは、64バイト×2個のダブルバッファに書き込まれます。

外付EEPROM

- 満杯になったダブルバッファの内容は、2個の外付EEPROM(24LC1025)に書き込まれます[I2C通信使用]

内蔵EEPROM

- 外付EEPROMに書き込んだデータ長は、内蔵EEPROMに書き込まれます。

◎USARTモジュール(送信)

- シリアルデータを送信します。

トランジスタ(2SC1815)

- シリアル信号のTTLレベル(対PIC)をRS232Cレベル(対パソコン)に変換(位相反転含む)します。

動作原理(ソフトウェア)

シリアルデータの受信とバッファリング(メモリ)処理□interrupt関数

- シリアルデータを受信すると、割り込みが発生します。
- 受信したデータをメモリ(ダブルバッファ<64バイト×2個>)に書き込みます。

バッファリング(外付EEPROM)処理□recv関数

- 満杯になったダブルバッファの内容を外付EEPROMに書き込みます。
- 外付EEPROMに書き込んだデータ長を、内蔵EEPROMに書き込みます。

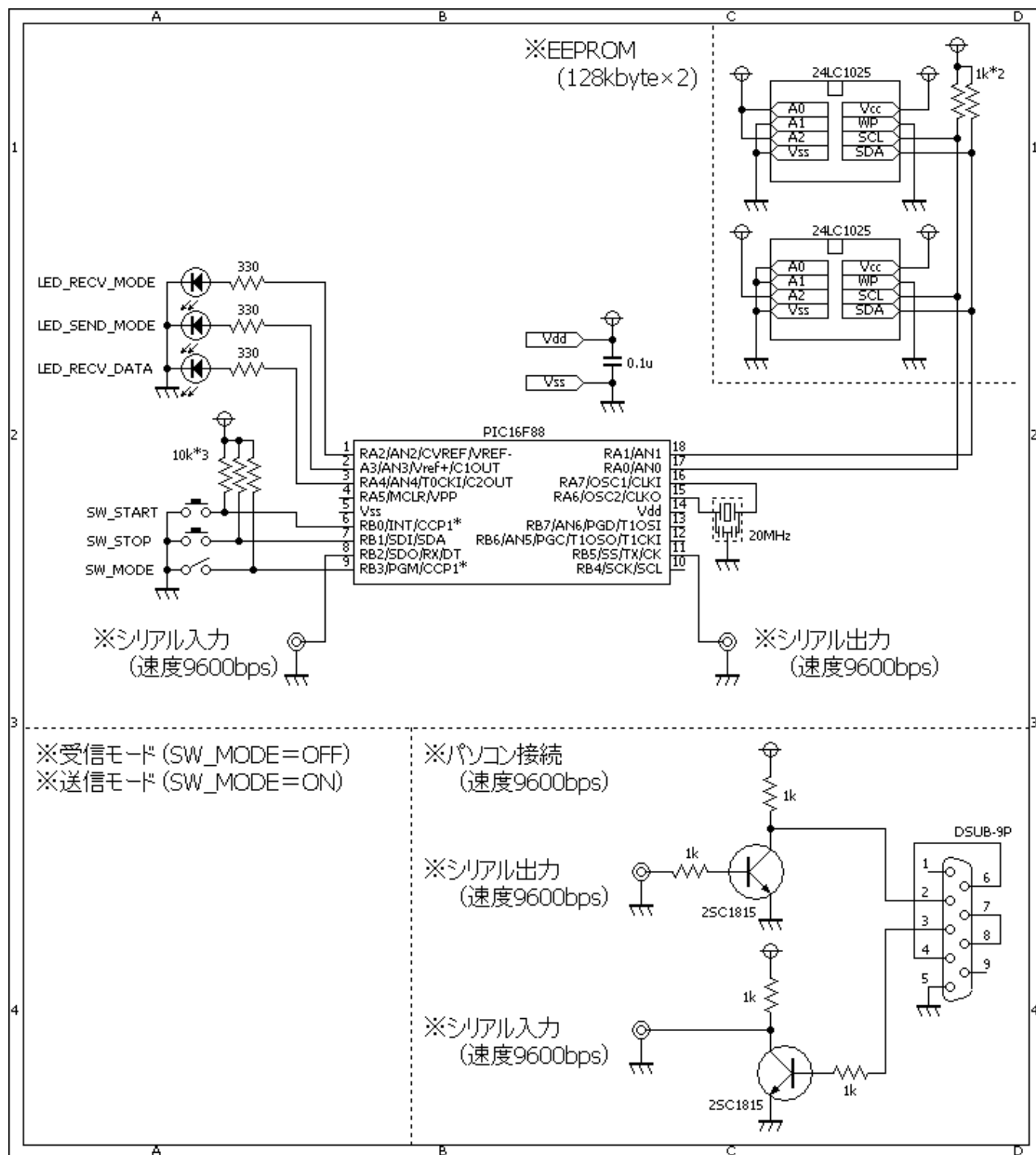
受信データ送信処理□send関数

- 外付EEPROMに書き込まれた受信データを、内蔵EEPROMに書き込まれたデータ長分だけ送信します。

受信モードと送信モードの切り替え処理

- 受信モードの開始
モードスイッチ(SW_MODE)をオフの状態にして、開始スイッチ(SW_START)を押下します。
LED_RECV_MODEが点灯します。
データを受信する毎に、LED_RECV_DATAが反転します。
- 受信モードの停止
停止スイッチ(SW_STOP)を押下します。
LED_RECV_MODEが消灯します。
- 送信モードの開始
モードスイッチ(SW_MODE)をオンの状態にして、開始スイッチ(SW_START)を押下します。
LED_SEND_MODEが点灯します。
- 送信モードの停止
停止スイッチ(SW_STOP)を押下します。
LED_SEND_MODEが消灯します。

回路図



ソースコード

[serial_buffer_v1_00.c](#)

```

//*****
*
/*
「簡易シリアルバッファ」
通信速度→9600bps

```

```

受信データ最大長→256kバイト(131072 * 2) 24LC1025 2個使用
*/
//*****
*
//■■■関数宣言■■■
extern void main();
extern void recv();
extern void send();
extern void init_port();
extern void init_eeprom_ex();
extern void eeprom_write_ex(long addr, short *buf, short len);
extern void eeprom_read_ex(long addr, short *buf, short len);
extern void init_usart();
extern void interrupt();
//*****
*
//■■■マクロ定義■■■
//USART
sbit TX_Direction at TRISB.B5;
sbit RX_Direction at TRISB.B2;
//EEPROM
sbit Soft_I2C_Scl at RA0_bit;
sbit Soft_I2C_Sda at RA1_bit;
sbit Soft_I2C_Scl_Direction at TRISA0_bit;
sbit Soft_I2C_Sda_Direction at TRISA1_bit;
#define ACK 1
#define NO_ACK 0
//LED
sbit LED_RECV_MODE at PORTA.B2;
sbit LED_RECV_MODE_Direction at TRISA.B2;
sbit LED_SEND_MODE at PORTA.B3;
sbit LED_SEND_MODE_Direction at TRISA.B3;
sbit LED_RECV_DATA at PORTA.B4;
sbit LED_RECV_DATA_Direction at TRISA.B4;
//SW
sbit SW_START at PORTB.B0;
sbit SW_START_Direction at TRISB.B0;
sbit SW_STOP at PORTB.B1;
sbit SW_STOP_Direction at TRISB.B1;
sbit SW_MODE at PORTB.B3;
sbit SW_MODE_Direction at TRISB.B3;
//other
#define INPUT_MODE 1
#define OUTPUT_MODE 0
//
#define CR 0x0D
#define LF 0x0A
//*****
*
char buf1[64], buf2[64];
short len1 = 0, len2 = 0;

```

```
short    flag = 0;
//*****
*
//■■■メイン関数■■■
void    main()
{
    short    cnt;
    //
    OSCCON = 0b01110000;           //クロックを8MHzに設定します。
    ANSEL   = 0b00000000;         //A/D変換モジュールは使用しません。
    //
    init_port();
    init_eeprom_ex();
    init_usart();
    //
    for (cnt = 0; cnt < 5; cnt++) {
        LED_RECV_MODE = 1;
        LED_SEND_MODE = 1;
        LED_RECV_DATA = 1;
        Delay_ms(100);
        LED_RECV_MODE = 0;
        LED_SEND_MODE = 0;
        LED_RECV_DATA = 0;
        Delay_ms(100);
    }
    //
    while (1) {
        if ((SW_MODE == 1) && (SW_START == 0)) {
            while (SW_START == 0) {
                Delay_ms(100);
            }
            LED_RECV_MODE = 1;
            recv();
            LED_RECV_MODE = 0;
        }
        if ((SW_MODE == 0) && (SW_START == 0)) {
            while (SW_START == 0) {
                Delay_ms(100);
            }
            LED_SEND_MODE = 1;
            send();
            LED_SEND_MODE = 0;
        }
    }
}
//*****
*
//■■■シリアルデータ受信関数■■■
void    recv()
{
    long    addr = 0;
```

```
//
len1 = 0;
len2 = 0;
flag = 0;
LED_RECV_DATA = 0;
// 割り込みを許可します。
INTCON.PEIE = 1;
INTCON.GIE = 1;
//
while ((SW_STOP != 0) && (addr < (131072 * 2))) {
    if (len1 == 64) {
        len1 = 0;
        eeprom_write_ex(addr, buf1, 64);
        addr += 64;
        //
        EEPROM_Write(0, addr & 0xFF);
        EEPROM_Write(1, (addr >> 8) & 0xFF);
        EEPROM_Write(2, (addr >> 16) & 0xFF);
        EEPROM_Write(3, (addr >> 24) & 0xFF);
    }
    if (len2 == 64) {
        len2 = 0;
        eeprom_write_ex(addr, buf2, 64);
        addr += 64;
        //
        EEPROM_Write(0, addr & 0xFF);
        EEPROM_Write(1, (addr >> 8) & 0xFF);
        EEPROM_Write(2, (addr >> 16) & 0xFF);
        EEPROM_Write(3, (addr >> 24) & 0xFF);
    }
}
// 割り込みを禁止します。
INTCON.PEIE = 0;
INTCON.GIE = 0;
//
if (len1 > 0) {
    eeprom_write_ex(addr, buf1, len1);
    addr += len1;
}
if (len2 > 0) {
    eeprom_write_ex(addr, buf2, len2);
    addr += len2;
}
//
EEPROM_Write(0, addr & 0xFF);
EEPROM_Write(1, (addr >> 8) & 0xFF);
EEPROM_Write(2, (addr >> 16) & 0xFF);
EEPROM_Write(3, (addr >> 24) & 0xFF);
}
//*****
*
```

```
// ■ ■ ■ シリアルデータ送信関数 ■ ■ ■
void    send()
{
    short    cnt;
    long    addr = 0, data_size;
    //
    data_size = EEPROM_Read(3);
    data_size = (data_size << 8) + EEPROM_Read(2);
    data_size = (data_size << 8) + EEPROM_Read(1);
    data_size = (data_size << 8) + EEPROM_Read(0);
    //
    while ((SW_STOP != 0) && (data_size > 0)) {
        if (data_size > 64) {
            eeprom_read_ex(addr, buf1, 64);
            for (cnt = 0; cnt < 64; cnt++) {
                UART1_Write(buf1[cnt]);
            }
            addr += 64;
            data_size -= 64;
        } else {
            eeprom_read_ex(addr, buf1, data_size);
            for (cnt = 0; cnt < data_size; cnt++) {
                UART1_Write(buf1[cnt]);
            }
            data_size = 0;
        }
    }
}

// *****
*

// ■ ■ ■ 入出力ポート初期化関数 ■ ■ ■
void    init_port()
{
    SW_START_Direction = INPUT_MODE;
    SW_STOP_Direction = INPUT_MODE;
    SW_MODE_Direction = INPUT_MODE;
    LED_RECV_MODE_Direction = OUTPUT_MODE;
    LED_SEND_MODE_Direction = OUTPUT_MODE;
    LED_RECV_DATA_Direction = OUTPUT_MODE;
    LED_RECV_MODE = 0;
    LED_SEND_MODE = 0;
    LED_RECV_DATA = 0;
}

// *****
*

// ■ ■ ■ 初期化関数 ■ ■ ■
void    init_eeprom_ex()
{
    Soft_I2C_Init();
}

// *****
```

```

*
//■■■■■■■■書き込み関数■■■■
void    eeprom_write_ex(long addr, short *buf, short len)
{
    unsigned    short    cnt;
    //
    Soft_I2C_Start();
    switch (addr & 0x30000) {
    case 0x00000:
        Soft_I2C_Write(0xA0);
        break;
    case 0x10000:
        Soft_I2C_Write(0xA8);
        break;
    case 0x20000:
        Soft_I2C_Write(0xA2);
        break;
    case 0x30000:
        Soft_I2C_Write(0xAA);
        break;
    }
    /*
    if ((addr & 0x10000) == 0)
        Soft_I2C_Write(0xA0);
    else
        Soft_I2C_Write(0xA8);
    */
    Soft_I2C_Write((addr >> 8) & 0xFF);
    Soft_I2C_Write(addr & 0xFF);
    for (cnt = 0; cnt < len; cnt++) {
        Soft_I2C_Write(buf[cnt]);
    }
    Soft_I2C_Stop();
    Delay_ms(1);
}

//*****
*
//■■■■■■■■読み込み関数■■■■
void    eeprom_read_ex(long addr, short *buf, short len)
{
    unsigned    short    cnt;
    //
    Soft_I2C_Start();
    switch (addr & 0x30000) {
    case 0x00000:
        Soft_I2C_Write(0xA0);
        break;
    case 0x10000:
        Soft_I2C_Write(0xA8);
        break;
    case 0x20000:

```

```

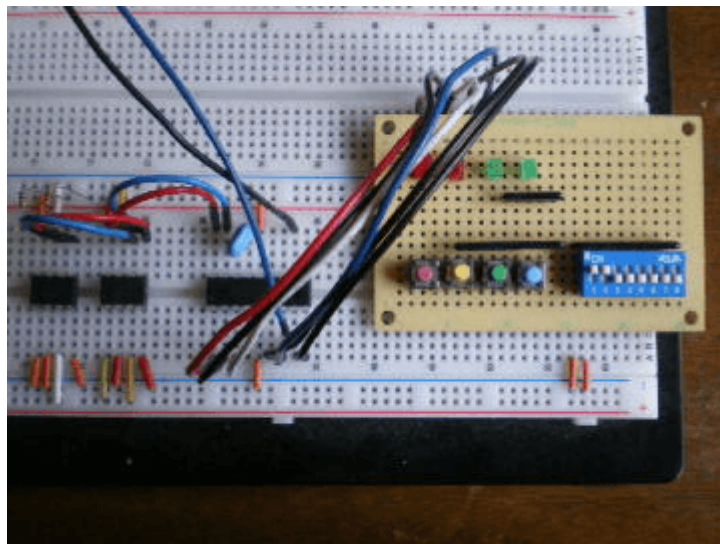
        Soft_I2C_Write(0xA2);
        break;
    case 0x30000:
        Soft_I2C_Write(0xAA);
        break;
    }
    /*
    if ((addr & 0x10000) == 0)
        Soft_I2C_Write(0xA0);
    else
        Soft_I2C_Write(0xA8);
    */
    Soft_I2C_Write((addr >> 8) & 0xFF);
    Soft_I2C_Write(addr & 0xFF);
    Soft_I2C_Start();
    switch (addr & 0x30000) {
    case 0x00000:
        Soft_I2C_Write(0xA1);
        break;
    case 0x10000:
        Soft_I2C_Write(0xA9);
        break;
    case 0x20000:
        Soft_I2C_Write(0xA3);
        break;
    case 0x30000:
        Soft_I2C_Write(0xAB);
        break;
    }
    /*
    if ((addr & 0x10000) == 0)
        Soft_I2C_Write(0xA1);
    else
        Soft_I2C_Write(0xA9);
    */
    for (cnt = 0; cnt < (len - 1); cnt++) {
        buf[cnt] = Soft_I2C_Read(ACK);
    }
    buf[cnt] = Soft_I2C_Read(NO_ACK);
    Soft_I2C_Stop();
    Delay_ms(1);
}
//*****
*
//■■■■初期化関数■■■
void init_usart()
{
    TX_Direction = OUTPUT_MODE;
    RX_Direction = INPUT_MODE;
    UART1_Init(9600);
    PIE1.RCIE = 1;
}

```

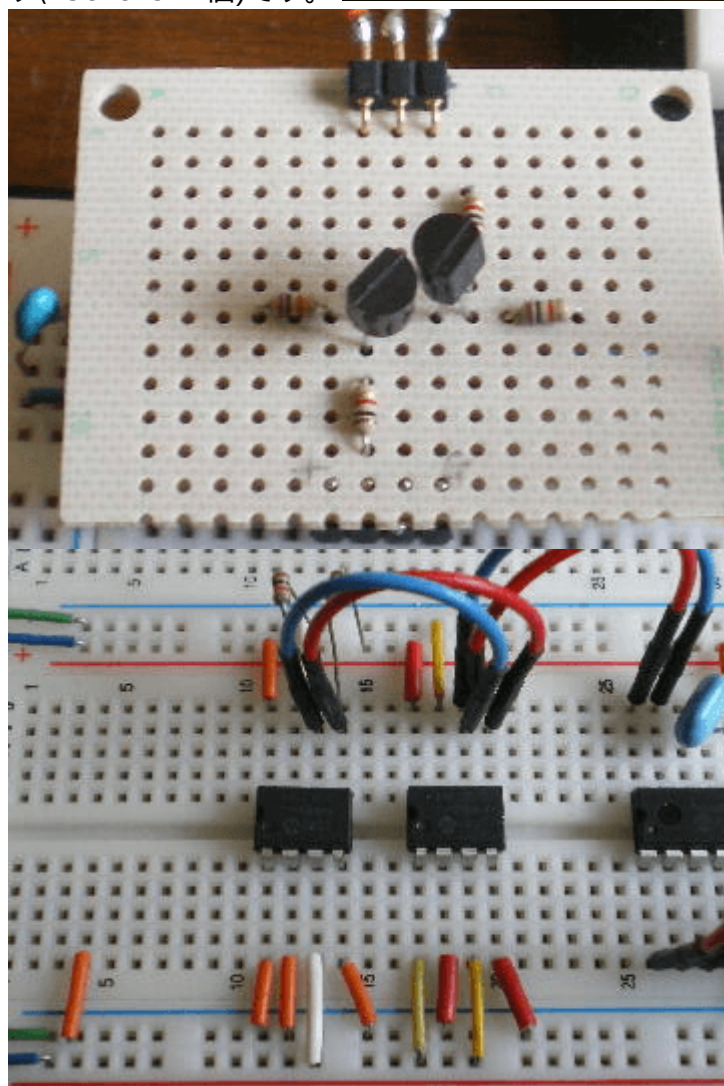
```
    PIR1.RCIF = 0;
}
//*****
*
//■■■割り込み関数■■■
void interrupt()
{
    char    rd;
    //
    if (PIR1.RCIF == 1) {
        PIR1.RCIF = 0;
        //
        rd = UART1_Read();
        LED_RECV_DATA = ~LED_RECV_DATA;
        switch (flag) {
            case 0:
                buf1[len1] = rd;
                len1++;
                if (len1 == 64) {
                    flag = 1;
                }
                break;
            case 1:
                buf2[len2] = rd;
                len2++;
                if (len2 == 64) {
                    flag = 0;
                }
                break;
        }
    }
}
//*****
*
```

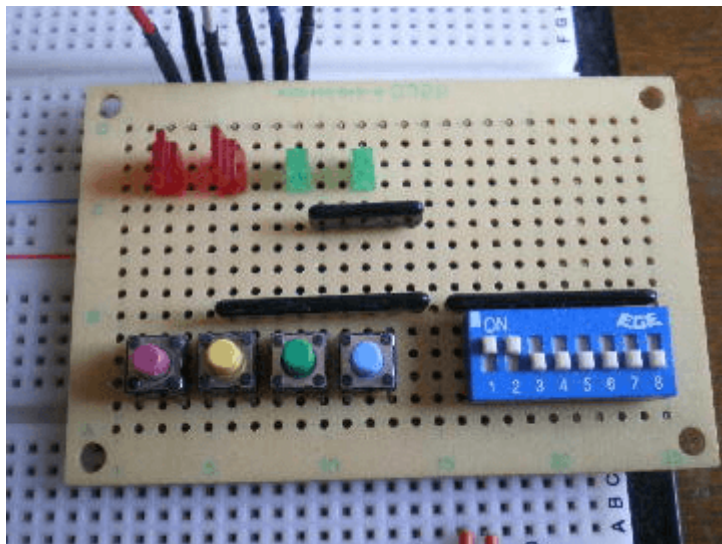
動作確認

左側:24LC1025×2個、PIC16F88□スイッチ関連□LED関連です。 右側:シリアル信号変換用のトランジス

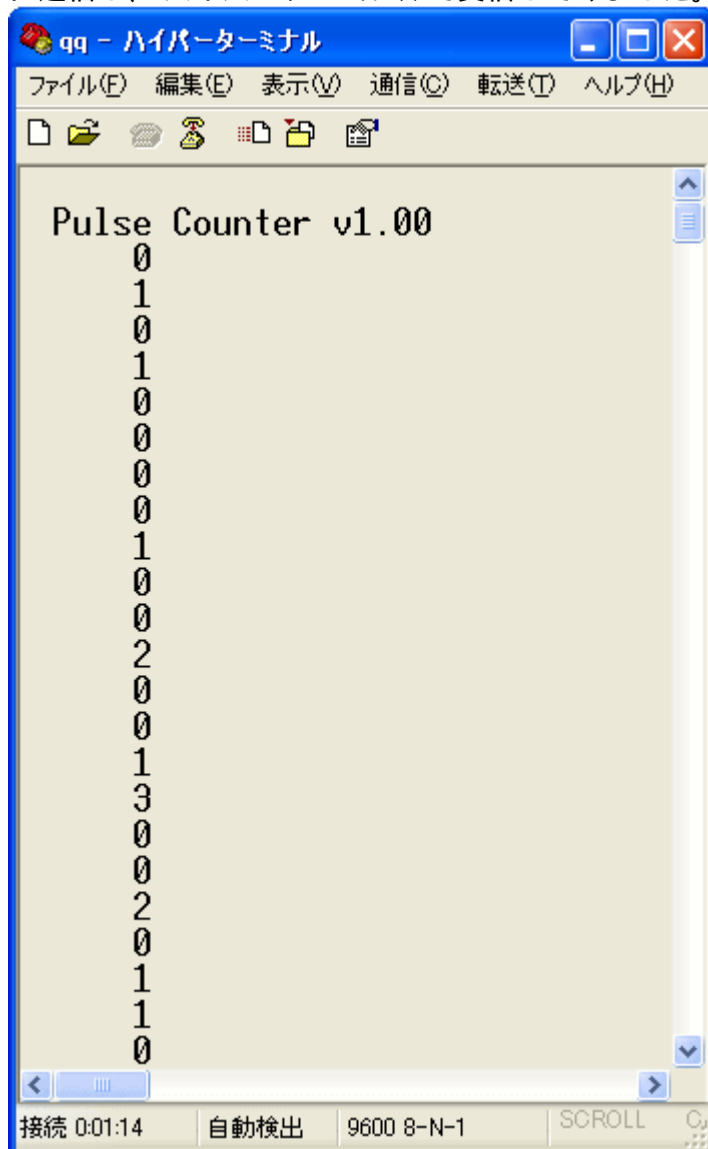


タ(2SC1815×2個)です。





前回製作した、「簡易パルスカウンタ」からのデータを受信し、更に、その受信したデータをパソコンに送信し、ハイパーターミナルで受信してみました。



著作権表示 copyright notice

このページは稲崎様の閉鎖したHPのコピーで、著作権は稲崎様にあります。[詳細](#) This page is a

copy of Mr. Inasaki's closed website, and the copyright is held by him. [Details](#)

From:

<http://www.deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://www.deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:156>

Last update: **2020/05/01 18:37**

