

25. 通信(プロジェクト:uart0)

25.1 概要

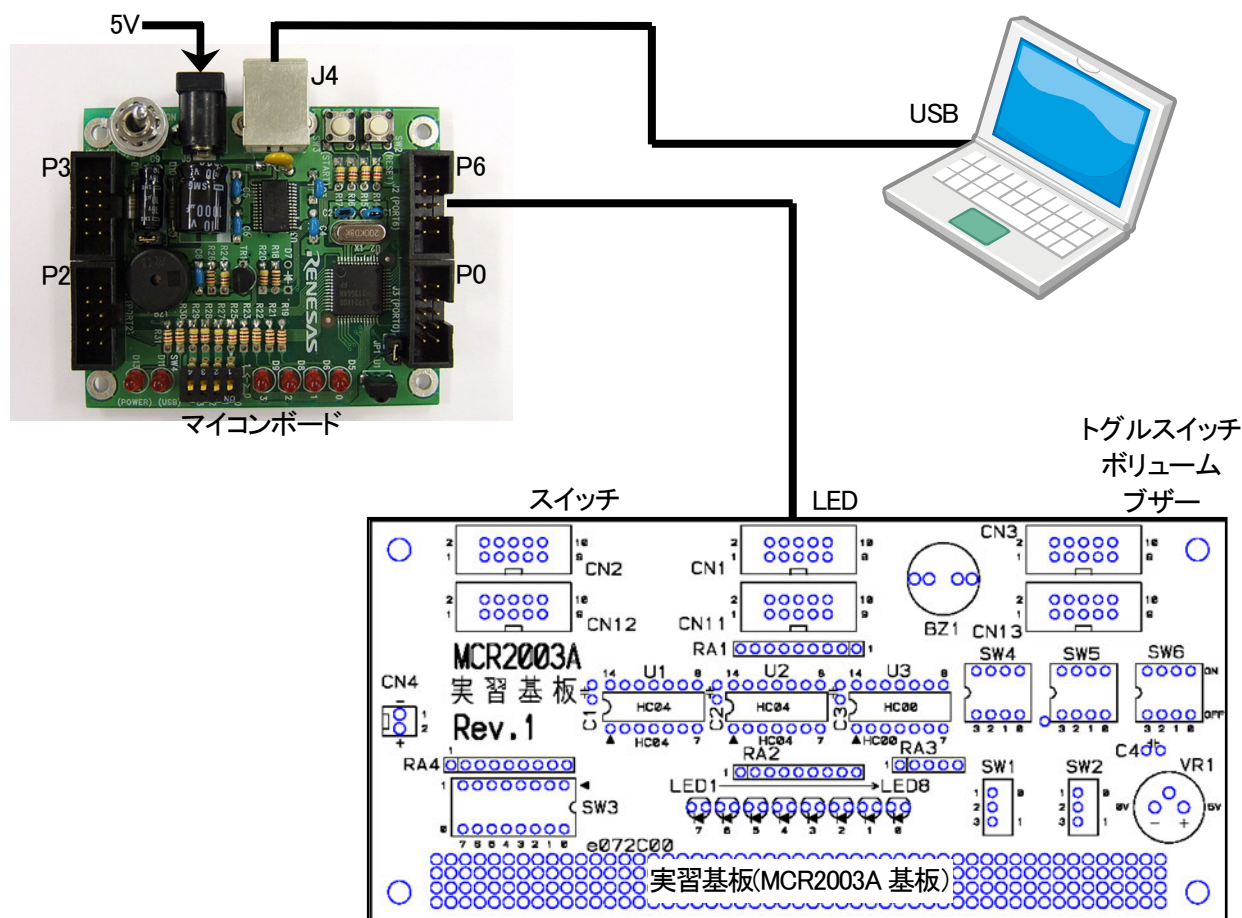
本章では、マイコンとパソコンで RS-232C 通信を行う方法を紹介します。通信は、マイコン内蔵機能の UART0 を使います。シリアル通信についても解説しています。

25.2 接続

■使用ポート

| マイコンのポート | 接続内容 |
|------------------|---|
| J4 (USB コネクタ) | USB コネクタを通して、パソコン(通信ソフト)のキーボードから打ち込んだ文字コードを LED へ出力します。 |
| P6 (J2) | 実習基板の LED 部など、出力機器を接続します。 |

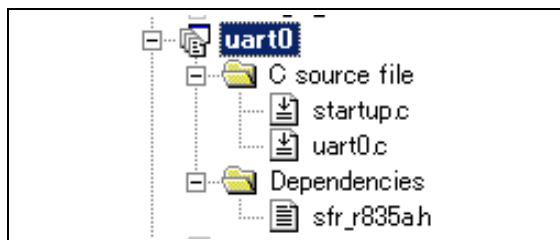
■接続例



■操作方法

パソコン側はTeraTermProやハイパーターミナルなどの通信ソフトを使います。キーボードから打ち込んだ文字コードが、ポート 6 に接続されているLEDに出力されます。また、マイコンで受信した文字コードをそのままパソコンに送信し、通信ソフトの画面上に表示されます。パソコン側の設定など実習方法は、「25.7 実習手順」を参照してください。

25.3 プロジェクトの構成



| | ファイル名 | 内容 |
|---|-------------|---|
| 1 | startup.c | 固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化 (初期値のないグローバル変数、初期値のあるグローバル変数の設定) などを行います。このファイルは共通で、どのプロジェクトもこのファイルから実行されます。 |
| 2 | uart0.c | 実際に制御するプログラムが書かれています。R8C/35A の内蔵周辺機能(SFR)の初期化も行います。 |
| 3 | sfr_r835a.h | R8C/35A マイコンの内蔵周辺機能を制御するためのレジスタ (Special Function Registers)を定義したファイルです。 |

25.4 プログラム「uart0.c」

```

1 :  /******
2 :  /* 対象マイコン R8C/35A */
3 :  /* ファイル内容 UART0の使用例 */
4 :  /* バージョン Ver. 1. 20 */
5 :  /* Date 2010. 04. 19 */
6 :  /* Copyright ルネサスマイコンカーラー事務局 */
7 :  /* 日立インターメディックス株式会社 */
8 :  /******
9 :  /*
10 :  入力 : UART0(パソコンのキーボードで入力したデータ)
11 :  ※パソコンはTeraTermProなどの通信ソフトを使用します
12 :  出力 : UART0(通信ソフトの画面)
13 :
14 :  RS232Cから出力されたデータをUART0で受信、そのままUART0から出力します。
15 :  TeraTermProなどの通信ソフトを通して、キーボードから入力した文字が
16 :  通信ソフトの画面上にそのまま表示されます。
17 :  */
18 :
19 :  /*=====*/
20 :  /* インクルード */
21 :  /*=====*/
22 :  #include "sfr_r835a.h" /* R8C/35A SFRの定義ファイル */
23 :  #include <stdio.h>
24 :
25 :  /*=====*/
26 :  /* シンボル定義 */
27 :  /*=====*/
28 :
29 :  /*=====*/
30 :  /* プロトタイプ宣言 */
31 :  /*=====*/
32 :  void init( void );
33 :  int get_uart0( unsigned char *s );
34 :  int put_uart0( unsigned char r );
35 :

```

```

36 : /*****
37 : /* メインプログラム */
38 : *****/
39 : void main( void )
40 : {
41 :     unsigned char    d;
42 :     int              ret;
43 :
44 :     init();           /* SFRの初期化 */
45 :
46 :     while( 1 ) {
47 :         ret = get_uart0( &d );
48 :         p6 = d;
49 :         if( ret == 1 ) put_uart0( d );
50 :     }
51 : }
52 :
53 : /*****
54 : /* R8C/35A スペシャルファンクションレジスタ (SFR) の初期化 */
55 : *****/
56 : void init( void )
57 : {
58 :     int i;
59 :
60 :     /* クロックをXINクロック (20MHz)に変更 */
61 :     prc0 = 1;          /* プロテクト解除 */
62 :     cm13 = 1;          /* P4_6, P4_7をXIN-XOUT端子にする */
63 :     cm05 = 0;          /* XINクロック発振 */
64 :     for(i=0; i<50; i++ ); /* 安定するまで少し待つ(約10ms) */
65 :     ocd2 = 0;          /* システムクロックをXINにする */
66 :     prc0 = 0;          /* プロテクトON */
67 :
68 :     /* ポートの入出力設定 */
69 :     prc2 = 1;          /* PD0のプロテクト解除 */
70 :     pd0 = 0xe0;        /* 7-5:LED 4:MicroSW 3-0:Sensor */
71 :     p1 = 0x0f;         /* 3-0:LEDは消灯 */
72 :     pd1 = 0xdf;        /* 5:RXD0 4:TXD0 3-0:LED */
73 :     pd2 = 0xfe;        /* 0:PushSW */
74 :     pd3 = 0xfb;        /* 4:Buzzer 2:IR */
75 :     pd4 = 0x83;        /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF */
76 :     pd5 = 0x40;        /* 7:DIP SW */
77 :     pd6 = 0xff;        /* LEDなど出力 */
78 :
79 :     /* UART0の設定 */
80 :     /* U0BRG = カウントソースの周波数/(設定したいbps*16)-1
81 :        = 20MHz / ( 9600 *16)-1
82 :        = 129.208 = 129
83 :     */
84 :     u0sr = 0x05;        /* P14=TXD0, P15=RXD0に設定 */
85 :     u0c0 = 0x00;        /* カウントソースなどの設定 */
86 :     u0c1 = 0x05;        /* 送信,受信許可 */
87 :     u0brg = 129;        /* 通信速度=9600pbs */
88 :     u0mr = 0x05;        /* UART0 データ長8bit 1ストップビット */
89 : }
90 :
91 : /*****
92 : /* 1文字受信 */
93 : /* 引数 受信文字格納アドレス */
94 : /* 戻り値 -1:受信エラー 0:受信なし 1:受信あり 文字は*sに格納 */
95 : *****/
96 : int get_uart0( unsigned char *s )
97 : {
98 :     int ret = 0, i;
99 :     unsigned int data;
100 :
101 :     if (ri_u0c1 == 1){ /* 受信データあり? */
102 :         data = u0rb;
103 :         *s = (unsigned char)data;
104 :         ret = 1;
105 :         if( data & 0xf000 ) { /* エラーあり? */
106 :             /* エラー時は再設定 */
107 :             re_u0c1 = 0;
108 :             for( i=0; i<50; i++ );
109 :             re_u0c1 = 1;
110 :
111 :             ret = -1;
112 :         }
113 :     }
114 :     return ret;
115 : }
116 :

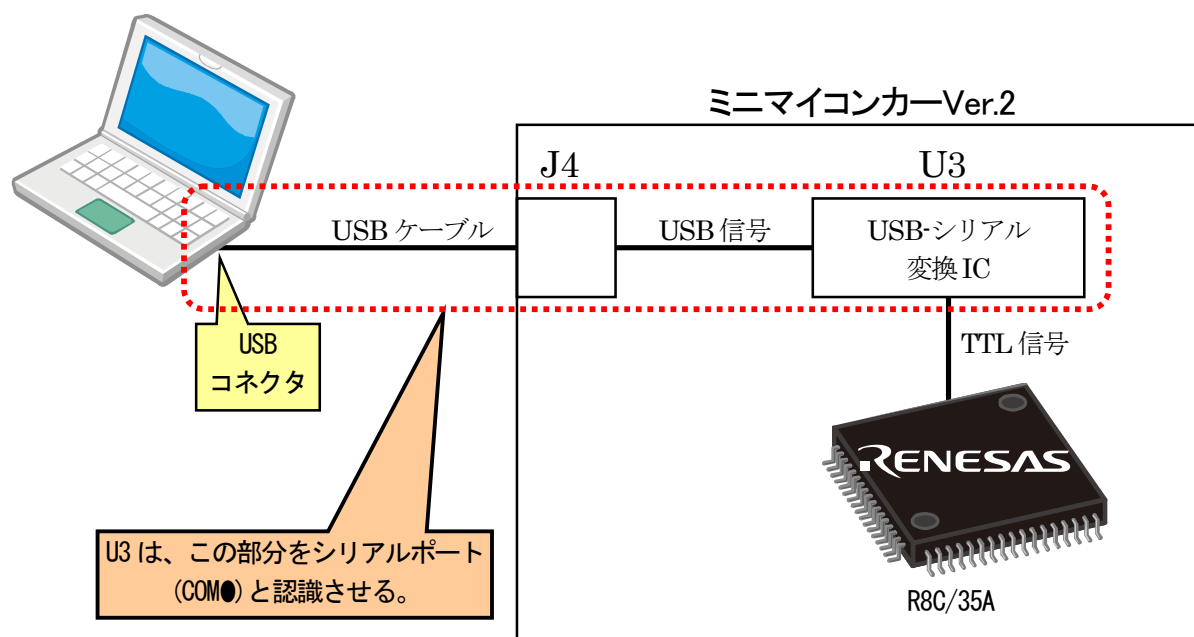
```

```
117 : /******  
118 : /* 1文字出力 */  
119 : /* 引数 送信データ */  
120 : /* 戻り値 0:送信中のため、送信できず 1:送信セット完了 */  
121 : /******  
122 : int put_uart0( unsigned char r )  
123 : {  
124 :     if(ti_u0c1 == 1) { /* 送信データなし? */  
125 :         u0tbl = r;  
126 :         return 1;  
127 :     } else {  
128 :         /* 先に送信中(今回のデータは送信せずに終了) */  
129 :         return 0;  
130 :     }  
131 : }  
132 :  
133 : /******  
134 : /* end of file */  
135 : /******
```

25.5 パソコンとミニマイコンカーVer.2 の通信

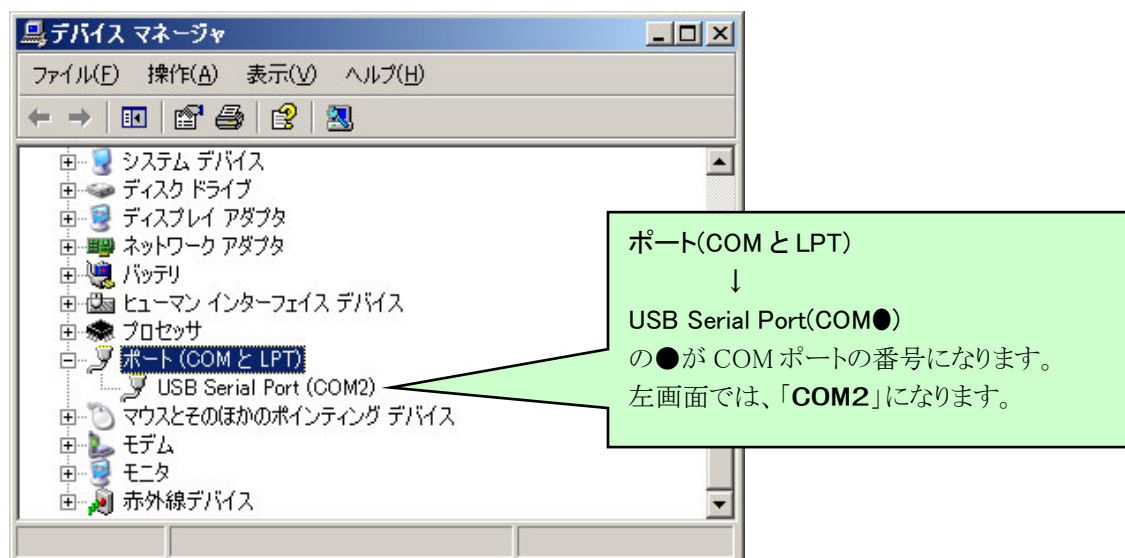
25.5.1 ミニマイコンカーVer.2 の通信方法

ミニマイコンカーVer.2 のマイコンボードは、パソコンとの通信を USB コネクタを通して行います。ミニマイコンカーVer.2 には、USB-シリアル変換 IC(U3)が搭載されており、パソコン(TeraTermPro やハイパーターミナルなどの通信ソフト)はミニマイコンカーVer.2 を COM ポート(シリアルポート)として認識します。



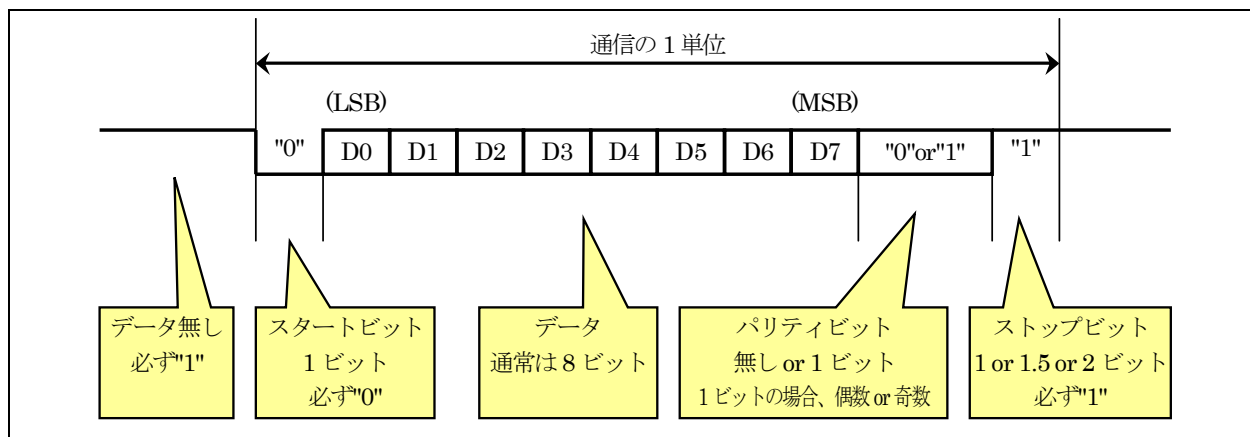
25.5.2 COMポートの確認

ミニマイコンカーVer.2 をパソコンの USB に接続した状態で「コントロールパネル→システム」を選択、システムのプロパティを開きます。「ハードウェア」タブを選択し、**デバイスマネージャ**をクリックします。



25.5.3 シリアル通信の設定

1 文字のデータを送る場合、文字データだけ送るわけではありません。受信側で正しく受け取るために送信側は、「スタートビット、データ、パリティビット、ストップビット」という形で送ります(下図)。



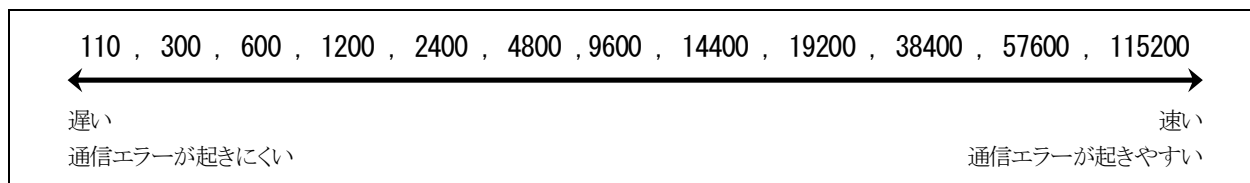
※LSB…Least Significant Bit(最下位ビット)のことです。

※MSB…Most Significant Bit(最上位ビット)のことです。

(1) 通信速度(ボーレート)

1 秒間に何ビット分のデータを送るか設定します。単位は、「Bits Per Second」で略して「bps(ビーピーエス)」です。

例えば 9600bps は、1 秒間に 9600 個のビットを送ることができます。TeraTermPro で設定できる通信速度と特徴を下記に示します。



(2) ストップビット

ストップビットは必ず"0"です。ちなみに、データが無いときは通信線は"1"になっています。"1"→"0"になると通信スタートと判断して、設定されている通信速度でデータを送信、または受信します。

(3) データ長

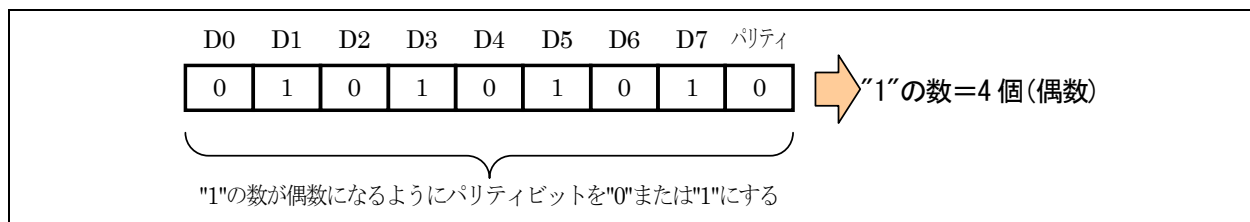
データのビット数を設定します。7ビット、8ビット、9ビットなどがあります。1文字=8ビットなので、8ビットにすることがほとんどです。

(4) パリティビット

パリティビットは、データが正しいかどうか誤り検出をするためのビットです。パリティ(parity)とは、「等しいこと」です。パリティビットは、無し/偶数パリティ/奇数パリティがあり、無しのときは、このビット自体がありません。

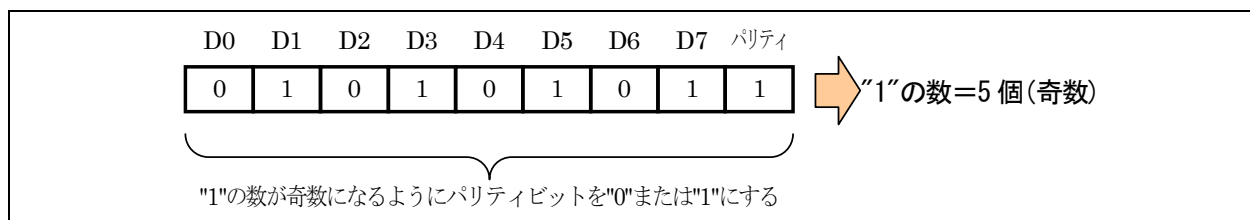
● 偶数パリティに設定した場合

データ8ビット分+パリティビットの各ビットの“1”の数が偶数になるようにパリティビットを設定して、送信します。受信側にも、“1”の数が偶数と設定しておけば、偶数ならエラー無し、奇数ならエラーと判断できます。エラーなら、パリティエラーとなります。



● 奇数パリティに設定した場合

データ8ビット分+パリティビットの各ビットの“1”の数が奇数になるようにパリティビットを設定して、送信します。受信側にも、“1”の数が奇数と設定しておけば、奇数ならエラー無し、偶数ならエラーと判断できます。エラーなら、パリティエラーとなります。



(5) ストップビット

ストップビットは必ず“1”です。1文字の通信が終わったことを示します。ビット幅は1ビット、1.5ビット、2ビットなどがあります。通常は1ビットです。受信側でストップビットを検出できない場合は、フレーミングエラーとなります。

(6) フロー制御

フロー制御とは、受信側で受信処理が間に合わない、または受信準備が整っていないときに、送信側に送信を待つように知らせ、受信準備ができれば送信を行うように知らせる機能です。

フロー制御を行うには、送信側、受信側にフロー制御を行うプログラムが入っていないといけません。今回の演習ではフロー制御は行いません。

(7) 今回の演習の設定

今回の演習の通信設定を下記に示します。送信側、受信側の両方で設定します。

| 項目 | 設定内容 |
|-------------|---------|
| 通信速度(ボーレート) | 9600bps |
| データ長 | 8ビット |
| パリティビット | 無し |
| ストップビット | 1ビット |
| フロー制御 | 無し |

25.6 プログラムの解説

25.6.1 init関数(UART0 の設定)

シリアルインターフェース 0(UART0)の設定プログラムは、次のようになります。

```

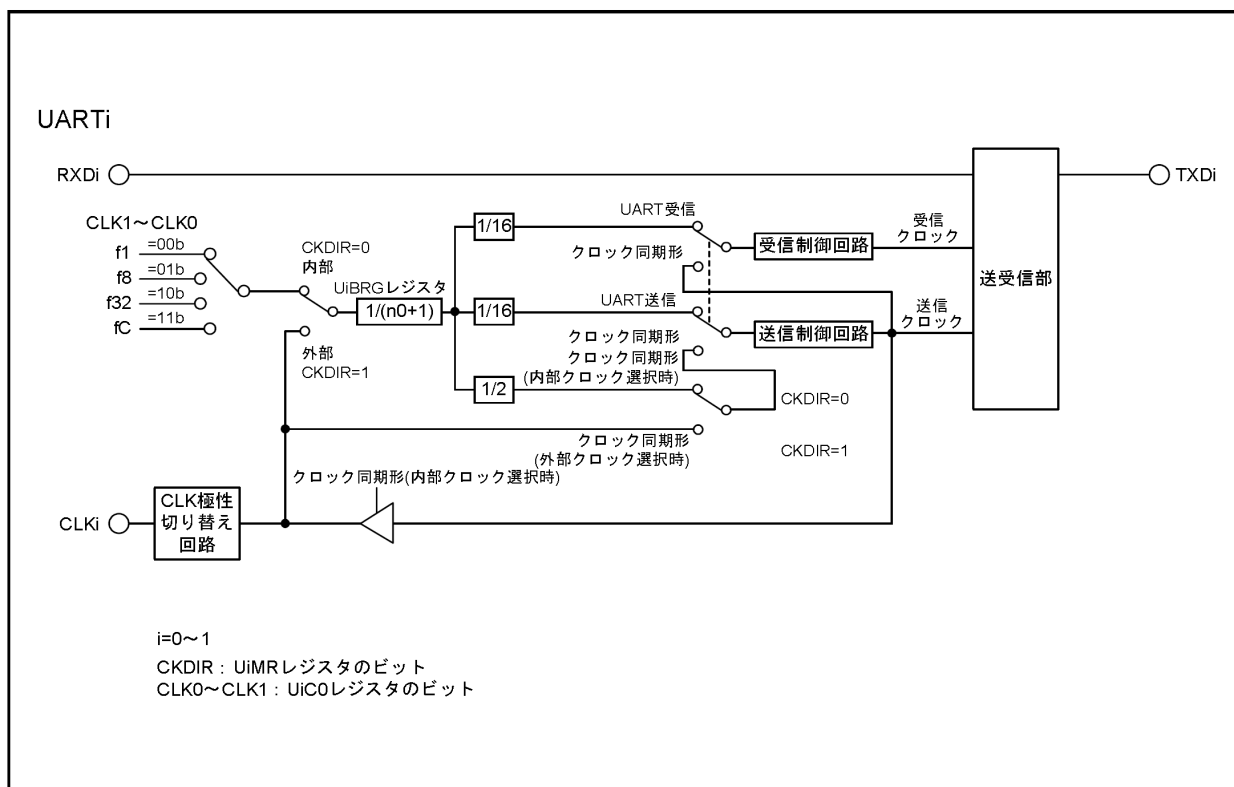
79 :      /* UART0の設定 */
80 :      /* U0BRG = カウントソースの周波数/(設定したいbps*16)-1
81 :          = 20MHz          /(      9600      *16)-1
82 :          = 129.208 = 129
83 :      */
84 :      u0sr = 0x05;          /* P14=TXD0, P15=RXD0に設定      */
85 :      u0c0 = 0x00;          /* カウントソースなどの設定      */
86 :      u0c1 = 0x05;          /* 送信、受信許可                  */
87 :      u0brg = 129;          /* 通信速度=9600pbs                */
88 :      u0mr = 0x05;          /* UART0 データ長8bit 1ストップビット */

```

(1) シリアルインターフェース

R8C/35Aには、シリアルインタフェース(UART)がUART0～UART2の3チャンネルあります。本プロジェクトでは、シリアルインタフェース 0(UART0)を使って実習します。ちなみに UART とは、「Universal Asynchronous Receiver Transmitter」の略称で、訳は特に決まっていますが「調歩同期方式によるシリアル信号を行うための集積回路」というような訳し方をすることが多いです。

シリアルインタフェースとは、1本の送信線、1本の受信線でデータのやり取りを行う通信方式です。RS-232Cは、シリアルインタフェースのひとつです。下記に、UARTi (i=0～1)のブロック図を示します。



端子構成を下表に示します。

| 端子名 | 割り当てる端子 | 入出力 | 機能 |
|------|--------------------|-----|-----------|
| TXD0 | P1_4 | 出力 | シリアルデータ出力 |
| RXD0 | P1_5 | 入力 | シリアルデータ入力 |
| CLK0 | P1_6 | 入出力 | 転送クロック入出力 |
| TXD1 | P0_1 または P6_3 | 出力 | シリアルデータ出力 |
| RXD1 | P0_2 または P6_4 | 入力 | シリアルデータ入力 |
| CLK1 | P0_3、P6_2 または P6_5 | 入出力 | 転送クロック入出力 |

(2) シリアルインタフェース 0(UART0)の設定

シリアルインターフェース 0(UART0)を使用して、パソコンと通信を行います。レジスタの設定手順を下記に示します。



①UART0 端子選択レジスタ(U0SR:UART0 function select register)の設定

UART0 の入出力をどの端子に割り当てるかを選択するレジスタです。UART0 は、端子が決まっています。ここでは、使うか使わないかを選択します。

| 設定 bit | 上:ビット名 下:シンボル | 内容 | 今回の 内容 |
|--------|--------------------------|---|-----------|
| bit7～5 | | "000"を設定 | 000 |
| bit4 | CLK0 端子選択ビット clk0sel0 | 0:CLK0 端子は使用しない 1:P1_6 に割り当てる CLK0 端子は、同期通信で使います。シリアル通信は非同期通信なので使用しません。 | 0 |
| bit3 | | "0"を設定 | 0 |
| bit2 | RXD0 端子選択ビット rx0sel0 | 0:RXD0 端子は使用しない 1:P1_5 に割り当てる RXD0 端子は、外部(パソコン)から通信信号を受信する端子です。使います。 | 1 |
| bit1 | | "0"を設定 | 0 |
| bit0 | TXD0 端子選択ビット tx0sel0 | 0:TXD0 端子は使用しない 1:P1_4 に割り当てる TXD0 端子は、外部(パソコン)へ通信信号を送信する端子です。使います。 | 1 |

UART0 端子選択レジスタ(U0SR)の設定値を下記に示します。

| bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|----------|---|---|----------|----------|----------|---|----------|
| 設定値 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 16 進数 | 0 | | | | 5 | | | |

②UART0 送受信制御レジスタ 0 (U0C0:UARTi transmit/receive control register0)の設定

カウントソースなどの設定を行います。

| 設定 bit | 上:ビット名 下:シンボル | 内容 | 今回の 内容 |
|--------|---|--|-----------|
| bit7 | 転送フォーマット選択ビット uform_u0c0 | 0:LSB ファースト 1:MSB ファースト LSB ファーストを選択します。通常のシリアル通信は、LSB ファーストです。 LSB とは、Least Significant Bit (最下位ビット) のことで、LSB ファーストとはデータをいちばん下のビットから送信することです。MSB とは、Most Significant Bit (最上位ビット) のことで、MSB ファーストとはデータをいちばん上のビットから送信することです。 | 0 |
| bit6 | CLK 極性選択ビット ckpol_u0c0 | 0:転送クロックの立ち下がりで送信データ出力、立ち上がりで受信データ入力 1:転送クロックの立ち上がりで送信データ出力、立ち下がりで受信データ入力 CLK 端子は使いませんのでどちらでも構いません。今回は"0"にしておきます。 | 0 |
| bit5 | データ出力選択ビット nch_u0c0 | 0:TXD0 端子は CMOS 出力 1:TXD0 端子は N チャネルオープンドレイン出力 通常は CMOS 出力です。 | 0 |
| bit4 | | "0"を設定 | 0 |
| bit3 | 送信レジスタ空フラグ txept_u0c0 | 0:送信レジスタにデータあり(送信中) 1:送信レジスタにデータなし(送信完了) このビットは、読み込みしかできません。設定するときは 0 にしておきます。 | 0 |
| bit2 | | "0"を設定 | 0 |
| bit1,0 | BRG カウントソース選択ビット (注 1) bit1:clk1_u0c0 bit0:clk0_u0c0 | 00:f1 を選択 (f1 =20MHz÷1=20MHz) 01:f8 選択 (f8 =20MHz÷8=2.5MHz) 10:f32 を選択 (f32 =20MHz÷32=0.625MHz) 11:fC を選択 (fC =本ボードでは無し) 今回は"00"を設定します。 | 00 |

注 1. BRG カウントソースを変更した場合は、U0BRG レジスタを再設定してください。

UART0 送受信制御レジスタ 0 (U0C0)の設定値を下記に示します。BRG カウントソース選択ビット(bit1,0)の設定方法は、UART0 ビットレートレジスタ(U0BRG)で説明します。

| bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|---|---|---|---|---|---|---|---|
| 設定値 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 進数 | 0 | | | | 0 | | | |

③UART0 送受信制御レジスタ 1 (U0C1:UART0 transmit/receive control register1)の設定

送信、受信を許可／禁止の設定をします。

| 設定 bit | 上:ビット名 下:シンボル | 内容 | 今回の 内容 |
|--------|---|---|-----------|
| bit7,6 | | "00"を設定 | 00 |
| bit5 | UART0 連続受信モード許可 ビット(注 2) u0rrm_u0c1 | 0:連続受信モード禁止 1:連続受信モード許可 | 0 |
| bit4 | UART0 送信割り込み要因選 択ビット u0irs_u0c1 | 0:送信バッファ空(TI=1) 1:送信完了(TXEPT=1) | 0 |
| bit3 | 受信完了フラグ(注 1) ri_u0c1 | 0:U0RB にデータなし 1:U0RB にデータあり このビットは、読み込みしかできません。設定は"0"に しておきます。 | 0 |
| bit2 | 受信許可ビット re_u0c1 | 0:受信禁止 1:受信許可 データの受信をするので、受信許可にします。 | 1 |
| bit1 | 送信バッファ空フラグ ti_u0c1 | 0:U0TB にデータあり 1:U0TB にデータなし このビットは、読み込みしかできません。設定は"0"に しておきます。 | 0 |
| bit0 | 送信許可ビット te_u0c1 | 0:送信禁止 1:送信許可 データの送信をするので、送信許可にします。 | 1 |

注 1. RI ビット(bit3)は U0RB レジスタの上位バイトを読み出したとき、“0”になります。

注 2. UART モード時、U0RRM ビットは“0”(連続受信モード禁止)にしてください。

UART0 送受信制御レジスタ 1 (U0C1)の設定値を下記に示します。

| bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|---|---|---|---|---|---|---|---|
| 設定値 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 16 進数 | 0 | | | | 5 | | | |

④UART0 ビットレートレジスタ (U0BRG:UART0 bit rate register)の設定

UART0 ビットレートレジスタ(U0BRG)は、UART0 送受信制御レジスタ 0(U0C0)の BRG カウントソース選択ビット (bit1,0)と組み合わせて、通信速度を設定します。

U0C0 の bit1,0 の値と U0BRG の計算方法の関係を、下表に示します。

| | U0C0 の bit1,0 | U0BRG の計算 |
|---|---------------|---------------------------------------|
| ① | 00 | $U0BRG = 1,250,000$ / 設定したいビットレート - 1 |
| ② | 01 | $U0BRG = 156,250$ / 設定したいビットレート - 1 |
| ③ | 10 | $U0BRG = 39,062.5$ / 設定したいビットレート - 1 |
| ④ | 11 | 本ボードでは設定できません |

設定したいビットレートは、下記のように計算します。結果は、四捨五入して整数にします。

- ・①を計算し、結果が 1～255 の値なら確定(U0C0 の bit1,0 は"00"にする)、それ以外なら②で再計算
- ・②を計算し、結果が 1～255 の値なら確定(U0C0 の bit1,0 は"01"にする)、それ以外なら③で再計算
- ・③を計算し、結果が 1～255 の値なら確定(U0C0 の bit1,0 は"10"にする)、
それ以外なら設定できないので、ビットレートを再検討

今回は、通信速度を 9600bps にします。

- ・①を計算、 $U0BRG = 1,250,000 / 9600 - 1 = 129.208 \div 129$

結果は、1～255 の値なのでこれで OK です。よって、U0BRG には 129 を設定します。

```
u0brg = 129;    // 9600bpsを設定
```

⑤UART0 送受信モードレジスタ(U0MR:UART0 transmit/receive mode register)の設定

ビット長、パリティビット、ストップビットなどを設定します。

| 設定 bit | 上:ビット名 下:シンボル | 内容 | 今回の 内容 |
|--------|---|---|-----------|
| bit7 | | "0"を設定 | 0 |
| bit6 | パリティ許可ビット prye_u0mr | 0:パリティ禁止 1:パリティ許可 今回はパリティを設定しません。 | 0 |
| bit5 | パリティ奇/偶選択ビット pry_u0mr | 0:奇数パリティ 1:偶数パリティ 今回は、パリティを使用しないのでどちらでも構いません。 "0"にしておきます。 | 0 |
| bit4 | ストップビット長選択ビット stps_u0mr | 0:1 ストップビット 1:2 ストップビット 1 ストップビットを設定します。 | 0 |
| bit3 | 内/外部クロック選択ビット ckdir_u0mr | 0:内部クロック 1:外部クロック(CLK0 端子) 内蔵クロックを選択します。 | 0 |
| bit2～0 | シリアル I/O モード選択ビット bit2:smd2_u0mr bit1:smd1_u0mr bit0:smd0_u0mr | 000:シリアルインタフェースは無効 001:クロック同期形シリアル I/O モード 100:UART モード転送データ長 7 ビット 101:UART モード転送データ長 8 ビット 110:UART モード転送データ長 9 ビット 上記以外:設定しないでください モードは、UART モードの転送データ長 8 ビットに設定 します。 | 101 |

UART0 送受信モードレジスタ(U0MR)の設定値を下記に示します。

| bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|---|---|---|---|---|---|---|---|
| 設定値 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 16 進数 | 0 | | | | 5 | | | |

25.6.2 get_uart0 関数

get_uart0 関数は、シリアルインターフェース 0(UART0)から 1 文字受信する関数です。

```

91 :  /*****
92 :  /* 1文字受信
93 :  /* 引数   受信文字格納アドレス
94 :  /* 戻り値 -1:受信エラー 0:受信なし 1:受信あり 文字は*sに格納
95 :  *****/
96 :  int get_uart0( unsigned char *s )
97 :  {
98 :      int ret = 0, i;
99 :      unsigned int data;
100 :
101 :      if (ri_u0c1 == 1){
102 :          data = u0rb;
103 :          *s = (unsigned char)data;
104 :          ret = 1;
105 :          if( data & 0xf000 ) {
106 :              /* エラー時は再設定 */
107 :              re_u0c1 = 0;
108 :              for( i=0; i<50; i++ );
109 :              re_u0c1 = 1;
110 :
111 :              ret = -1;
112 :          }
113 :      }
114 :      return ret;
115 :  }
```

(1) get_uart0 関数の使い方

get_uart0 関数の使い方を下記に示します。

```

unsigned char  d;          // 受信データ
int            ret;        // 受信結果

ret = get_uart0( &d );    // 受信データを保存する変数は、アドレス参照にする(&をつける)
```

引数は、受信したデータを保存する変数をアドレス参照で設定します(変数に「&」を付けます)。戻り値と引数(上記の場合は変数 d)の関係は、下表のようになります。

| 戻り値 | 説明 |
|-----|--|
| 1 | 受信データありです。変数 d には、受信した 1 文字が格納されます。 |
| 0 | 受信データなしです。変数 d には、何も代入されません。 |
| -1 | 受信エラーです。何かの問題で、うまく受信できませんでした。変数 d には、受信した 1 文字が格納されていますが、エラーで受信した不完全なデータが格納されます。 |

パソコンから「a」を送信し、下記プログラムを実行したとします。

```
unsigned char  d;           // 受信した値
int           ret;         // 受信結果

ret = get_uart0( &d );     // ret=1 , d=0x61 ( 'a' )
```

変数 ret には 1、変数 d には 0x61 が代入されます。0x61 はアスキーコードの 'a' です。
これから、上記の内容がどのように実行されるのか説明します。

(2) 変数

```
98 :      int ret = 0, i;
99 :      unsigned int data;
```

| | |
|------|---|
| 98 行 | 変数 ret には、戻り値を設定します。まずは 0 を代入して、仮で受信データなしにしておきます。 変数 i は、この関数内での作業用です。 |
| 99 行 | 変数 data には、受信したデータとエラー情報を格納します。 |

(3) データが受信されたときの処理

受信データがあるかどうかチェック、受信データがあったら格納処理などを行います。

```
101 :      if (ri_u0c1 == 1) {                               /* 受信データあり？          */
102 :          data = u0rb;
103 :          *s = (unsigned char)data;
104 :          ret = 1;
```

| 101 行 | <p>「RI_U0C1」とは、UART0 送受信制御レジスタ 1 (U0C1)の受信完了フラグ(bit3)のことです。 受信完了フラグ(bit3)は、 0:UART0 受信バッファレジスタ(U0RB)にデータなし 1:UART0 受信バッファレジスタ(U0RB)にデータあり なので、このビットが“1”なら受信ありと判断できます。</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|---|-----|----|----|----|----|----|----|-----------------|---|---|---|---|---|---|---|---|---|----|--|--|--|--|--|--|--|--|-----------------|--|--|--|--|--|--|--|
| 102 行 | <p>UART0 受信バッファレジスタ(U0RB)は、受信したデータとエラーの有無を保存しているレジスタです。このデータを変数 data に代入します。 U0RB の内容を下記に示します。</p> <table><tr><th>bit</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr><tr><td>内容</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="8">受信データ 0x00～0xff</td></tr></table> <div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div> | bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 内容 | | | | | | | | | 受信データ 0x00～0xff | | | | | | | |
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | | | | | | | | | | | | | |
| 内容 | | | | | | | | | 受信データ 0x00～0xff | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | |
|-------|---|
| 103 行 | UART0 受信バッファレジスタ(U0RB)は、受信したデータとエラーの有無を保存しているレジスタです。このデータをポインタ演算子 s が示すアドレスに代入します。「(unsigned char)」は、U0BR の下位 8 ビット(bit7～bit0)のみ代入するためのキャスト演算子です。 |
| 104 行 | 変数 ret を 1 にして、戻り値を受信ありにします。 |

(3) データにエラーがないかチェック

受信データにエラーが無いかどうかチェックします。

| | | | |
|-------|------------------------|-----------|----|
| 105 : | if(data & 0xf000) { | /* エラーあり？ | */ |
| 106 : | /* エラー時は再設定 */ | | |
| 107 : | re_u0c1 = 0; | | |
| 108 : | for(i=0; i<50; i++); | | |
| 109 : | re_u0c1 = 1; | | |
| 110 : | | | |
| 111 : | ret = -1; | | |
| 112 : | } | | |

| | |
|-----------------|--|
| 105 行 | 変数 data の bit15～bit12 をチェックします。0 以外なら受信エラーがあったと判断して、106 行以降を実行します。 |
| 107 行 ～109 行 | エラーがあった場合、R8C/35A ハードウェアマニュアルには、次のことを行うように書かれています。 RE_U0C1 を"0"にしてから、RE_U0C1 を"1"にしてください そのため、107 行で RE_U0C1 を"0"、108 行でウェイトを入れ、109 行で RE_U0C1 を"1"にします。 |
| 111 行 | 戻り値をエラー(-1)にします。 |

(4) 終了

データありか、受信ありか、エラーかの情報を保存している変数 ret の値を戻り値にして終わります。

| | |
|-------|-------------|
| 114 : | return ret; |
| 115 : | } |

25.6.3 put_uart0 関数

put_uart0 関数は、シリアルインターフェース 0(UART0)へ、1 文字送信する関数です。

```

117 :  /*****
118 :  /* 1 文字出力                                     */
119 :  /* 引数   送信データ                               */
120 :  /* 戻り値 0:送信中のため、送信できず 1:送信セット完了 */
121 :  *****/
122 :  int put_uart0( unsigned char r )
123 :  {
124 :      if(ti_u0c1 == 1) {                               /* 送信データなし? */
125 :          u0tbl = r;
126 :          return 1;
127 :      } else {
128 :          /* 先に送信中(今回のデータは送信せずに終了) */
129 :          return 0;
130 :      }
131 :  }
```

(1) put_uart0 関数の使い方

put_uart0 関数の使い方を下記に示します。

```

unsigned char  d;          // 送信するデータ
int           ret;        // 送信結果

ret = put_uart0( d );     // データ送信
```

引数は、送信するデータを設定している変数を設定します。戻り値は、下表のようになります。

| 戻り値 | 説明 |
|-----|---|
| 1 | 送信セット完了です。シリアルコミュニケーション 0(UART0)は、送信を開始します。 |
| 0 | 前回、送信したデータを送信中のため、今回のデータは送信できません。 |

マイコンから、「a」を送信するために、下記プログラムを実行したとします。

```

unsigned char  d = 'a';    // 送信したいデータ
int           ret;        // 受信結果

ret = put_uart0( d );     // 'a' を送信, ret=1(送信セット完了)
```

送信セットが完了すれば変数 ret には 1 が代入されます。ret=0 の場合は送信セットできませんでしたので、再度送りたいときは戻り値が 1 になるまで put_uart0 関数を実行します。

送信セットされるまで繰り返したい場合は、下記のようにします。

```
while( !put_uart0( d ) );    // 送信セット完了するまで繰り返し続ける
```

while(値)は、値が 0 以外なら繰り返す、という命令です。put_uart0 関数の戻り値は、0 なら送信しなかったということなので、「!」(以外)を付けて、戻り値を「0(送信せず)以外」にしています。

これから、上記の作業がどのようにプログラムで実行されるのか説明します。

(2) 送信処理

```
124 :      if(ti_u0c1 == 1) {                      /* 送信データなし?          */
125 :          u0tbl = r;
126 :          return 1;
```

| | |
|-------|--|
| 124 行 | 「TLU0C1」とは、UART0 送受信制御レジスタ 1(U0C1)の送信バッファ空フラグ(bit1)のことです。 送信バッファ空フラグ(bit1)は、 0:UART0 送信バッファレジスタ(U0TB)にデータあり(現在送信中) 1:UART0 送信バッファレジスタ(U0TB)にデータなし なので、このビットが“1”なら今現在、送信しているデータはなしと判断できます。 |
| 125 行 | UART0 送信バッファレジスタ(U0TB)に送信したいデータを代入して、送信を開始します。 U0TB は 16bit 幅あり、送信データは下位ビットに設定します。「U0TBL」は U0TB の下位 8bit を示す名称です。そのため、U0TBL に値を代入します。 U0TBL に値を代入すると、TLU0C1 が自動で“0”(データあり)になります。送信が完了すると、TLU0C1 が自動で“1”(データなし)になります。 |
| 126 行 | 送信データをセットできたので、戻り値 1 で終わります。 |

(3) 送信中のデータがあった場合

```
127 :      } else {
128 :          /* 先に送信中(今回のデータは送信せずに終了) */
129 :          return 0;
130 :      }
```

| | |
|-------|--|
| 129 行 | 現在、送信中のデータがある場合は、何もせずに戻り値 0 でこの関数を終わります。 |
|-------|--|

25.6.4 main関数

```
39 : void main( void )
40 : {
41 :     unsigned char  d;
42 :     int             ret;
43 :
44 :     init();          /* SFRの初期化          */
45 :
46 :     while( 1 ) {
47 :         ret = get_uart0( &d );
48 :         p6 = d;
49 :         if( ret == 1 ) put_uart0( d );
50 :     }
51 : }
```

| | |
|------|---|
| 44 行 | init 関数を実行します。 init 関数では、ポートの入出力設定、シリアルコミュニケーション 0(UART0)の設定を行っています。 |
| 47 行 | get_uart0 関数で、パソコンから送られてきたデータを受信します。 受信結果は変数 ret に、受信文字は変数 d に代入します。 |
| 48 行 | 受信したデータをポート 6 に出力します。 |
| 49 行 | 変数 ret が 1 なら、すなわち受信データがあるなら、put_uart0 関数で受信したデータと同じデータをパソコンに送信します。 |

25.7 実習手順

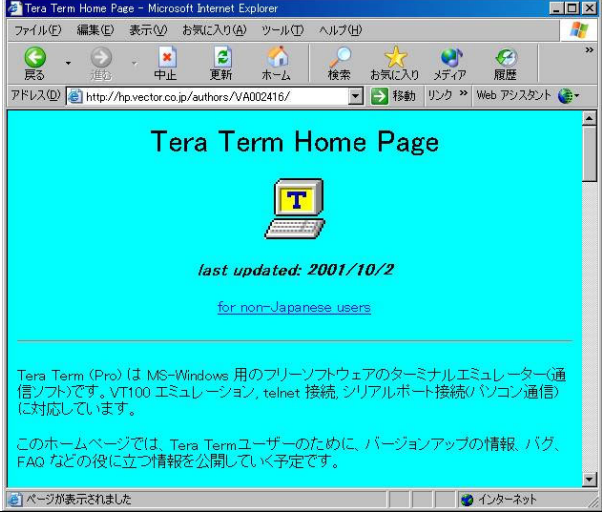
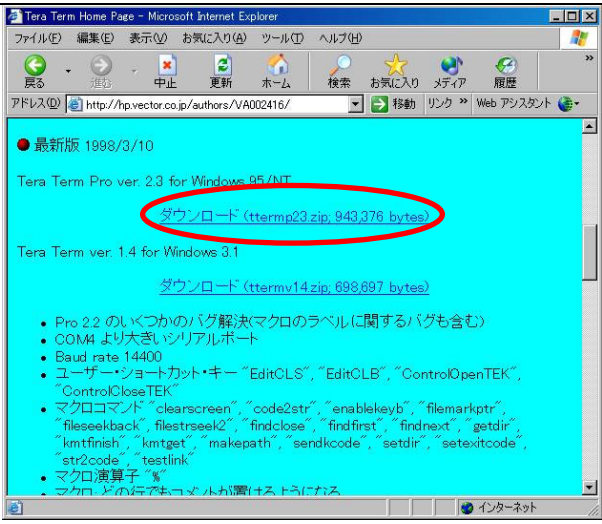
25.7.1 パソコン設定する前準備




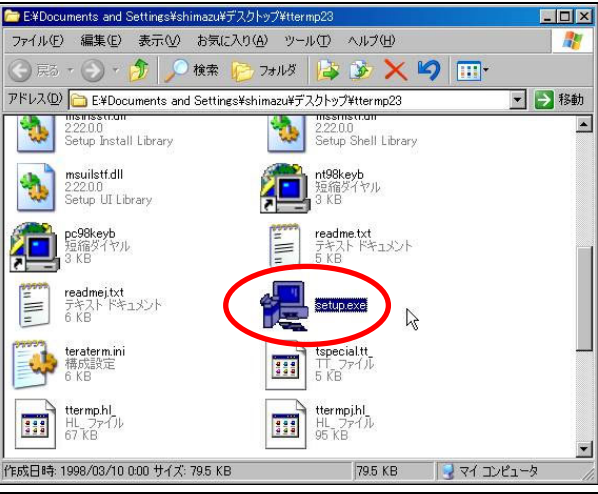
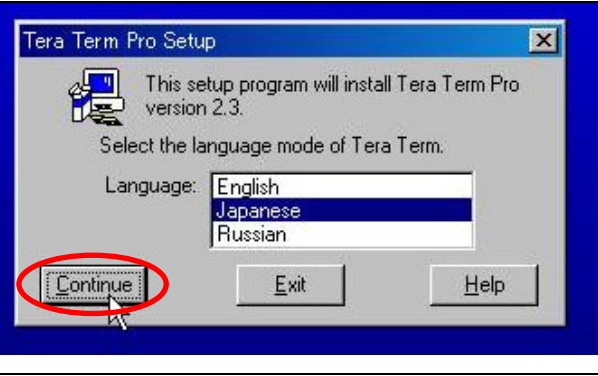
これからパソコンの設定をします。その前に下記作業をあらかじめ済ませておきます。

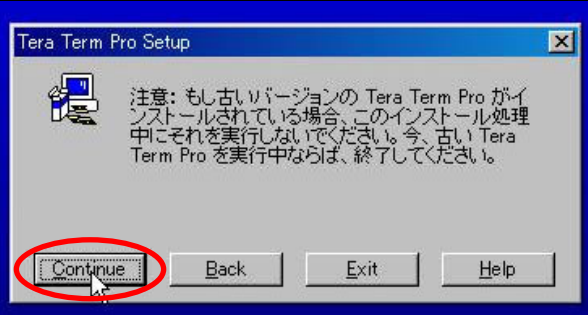


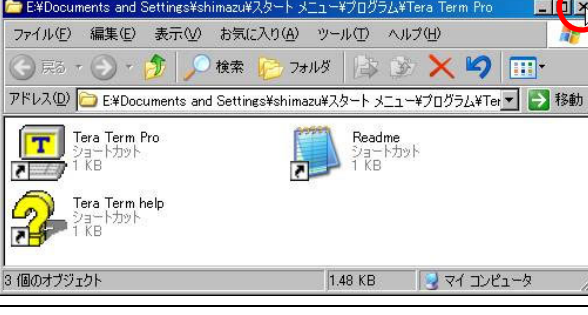
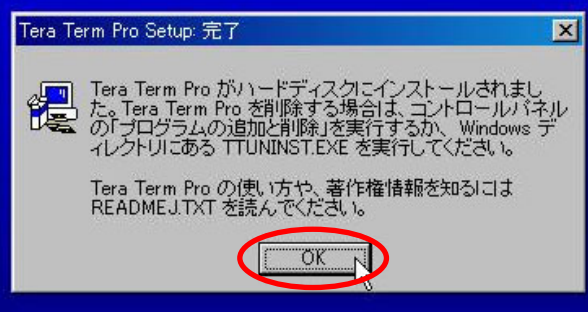
- ・マイコンボードに「uart0.mot」ファイルを書き込みます。
- ・USB ケーブルは接続したままにしておきます。

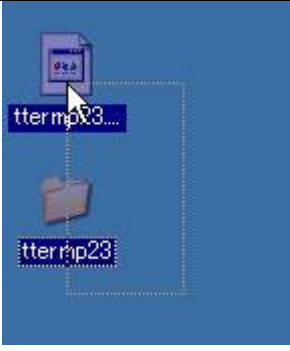
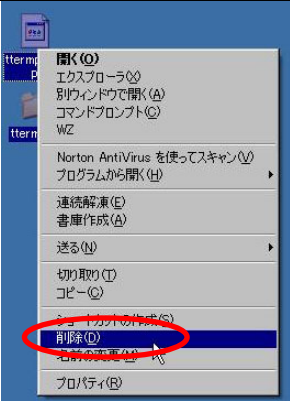
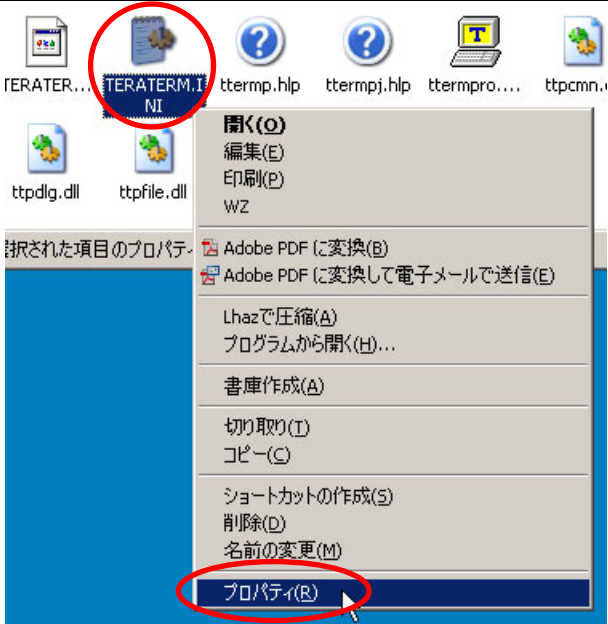
25.7.2 TeraTermProのインストール

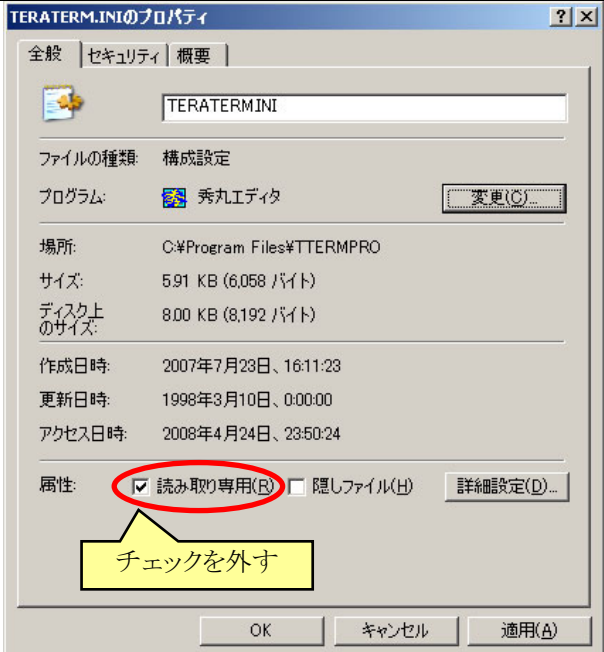
フリーソフトで通信のできる「**Tera Term Pro**」というソフトを使用して、マイコンと通信をおこないます。ここではインストール手順を説明します。既に Tera Term Pro をインストールしている場合は、ここでインストールする必要はありません。

| | | |
|---|---|---|
| 1 |  | <p>まず、ソフトをダウンロードします。インターネットブラウザで http://hp.vector.co.jp/authors/VA002416/ を開きます。</p> |
| 2 |  | <p>下の方に「ダウンロード (tterm23.zip; 943,376 bytes)」とあるので、クリックして保存します。</p> |

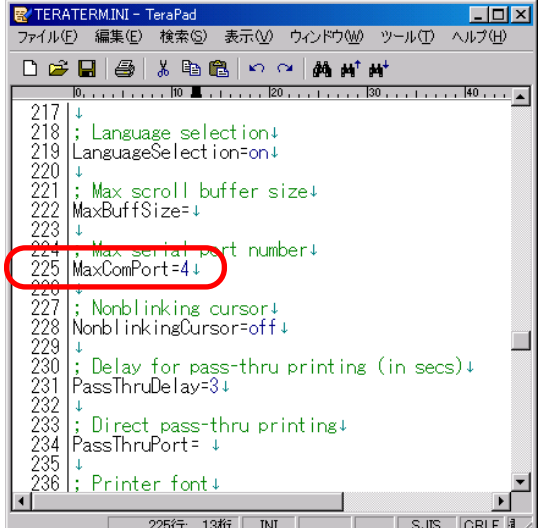
| | | |
|---|---|---|
| 3 |  | <p>保存は何処でも良いですが、ここではデスクトップに保存します。</p> |
| 4 |  | <p>保存されました。</p> |
| 5 |  | <p>tterm23.zip は ZIP 形式で圧縮された形式なので、解凍します。解凍ソフトは、フリーソフトでたくさんありますので、インターネットなどで探してください。左画面は、tterm23 というフォルダに解凍したところです。</p> |
| 6 |  | <p>解凍後のファイルです。setup.exe を実行します。</p> |
| 7 |  | <p>言語の選択です。日本になっていますのでそのまま Continue(続ける)をクリックします。</p> |

| | | |
|----|---|---|
| 8 |  | <p>注意が出ます。Continue(続ける)をクリックします。</p> |
| 9 |  | <p>キーボードの選択です。Continue(続ける)をクリックします。</p> |
| 10 |  | <p>インストール先を確認して、問題なければContinue(続ける)をクリックします。インストールが開始されます。</p> |
| 11 |  | <p>メニューが立ち上がります。Xをクリックして閉じます。</p> |
| 12 |  | <p>OKをクリックして、インストールを完了します。</p> |

| | | |
|----|--|--|
| 13 |  | インターネットからファイルをダウンロードした場合、ダウンロードしたファイルを削除しておきましょう。 tterm23.zip と tterm23 フォルダを選択します。 |
| 14 |  | どちらかのファイルの上で右クリックして「削除」をクリック、ファイルを削除します。 |
| 15 |  | Tera Term Pro をインストールした状態では、COMポートが1～4までしか選択できません。USB-232C 変換 IC は、COMポートの番号がCOM5 以上になることがあります。そのため、COM5 以上も選択できるように変えておきましょう。 「Cドライブ→Program Files→TTERMPRO→TERATERM.INI」を右クリック、「プロパティ」を選択します。 |

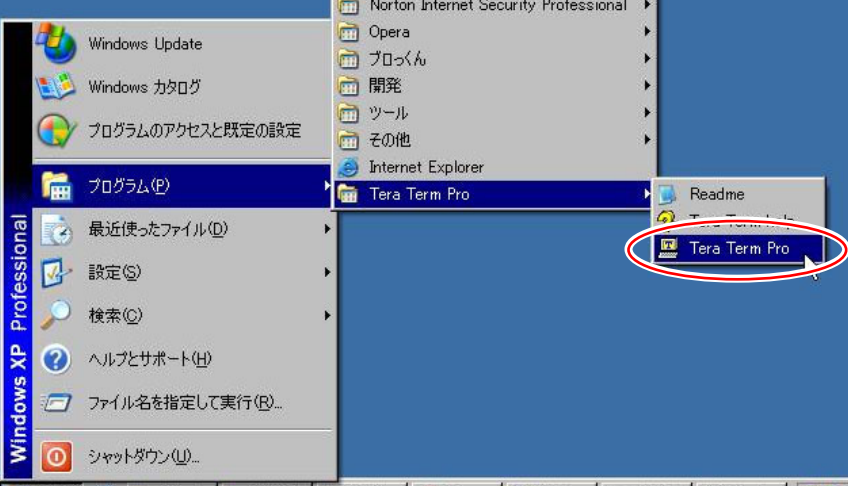
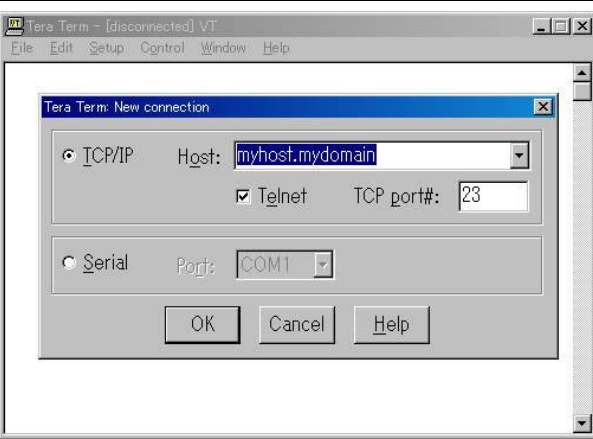
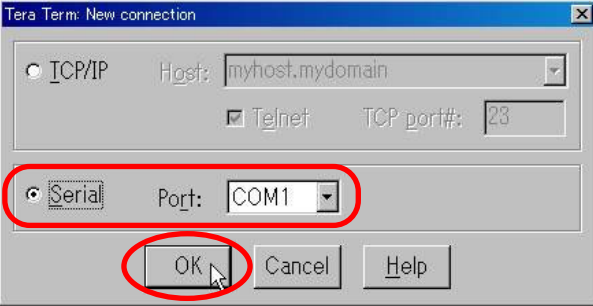
| | | |
|----|---|--|
| 16 |  | <p>「読み取り専用」のチェックが付いている場合、チェックを外します。 チェックが付いていない場合は、何もせずに閉じます。</p> |
|----|---|--|

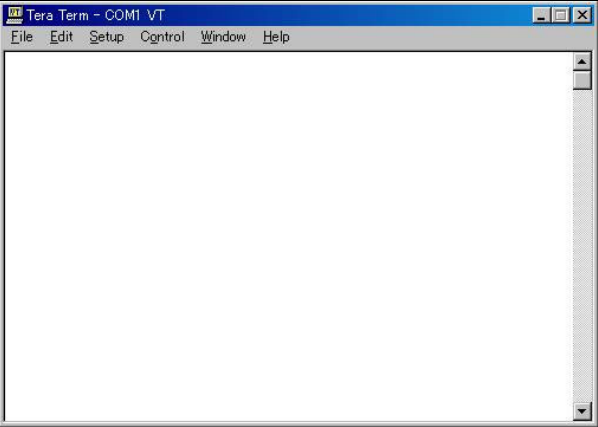
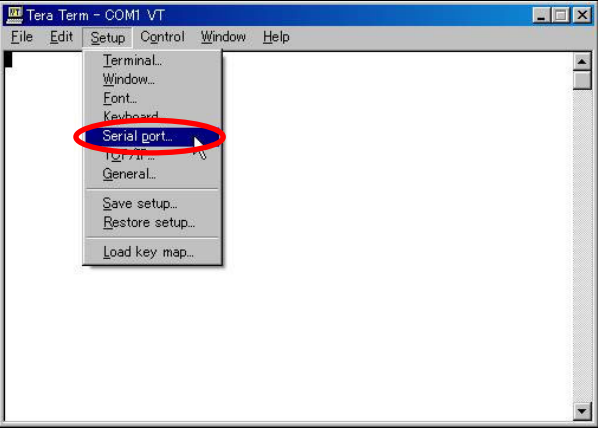
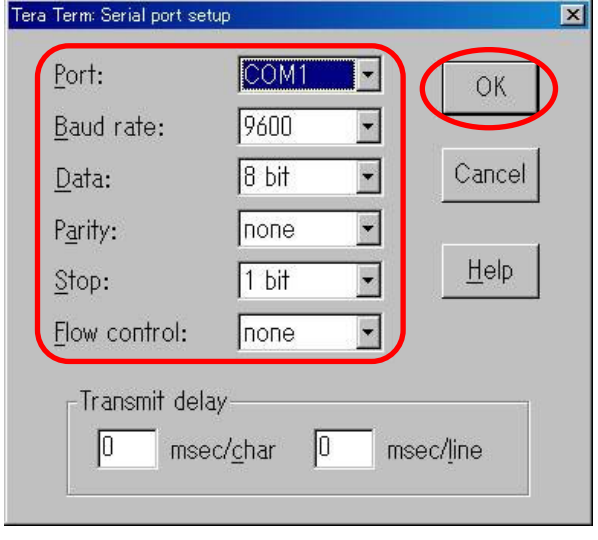
| | | |
|----|--|---|
| 17 |  | <p>再度、「TERATERM.INI」を右クリックし、今度はファイルを開きます。</p> |
|----|--|---|

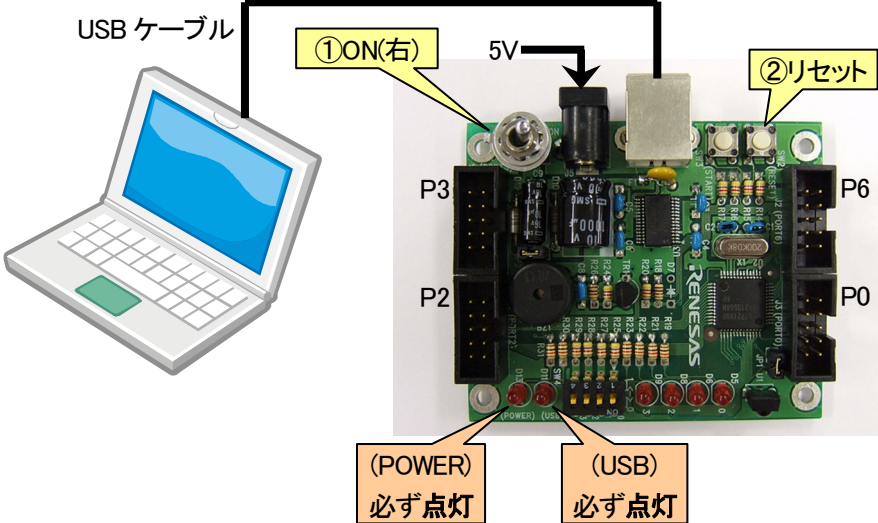
| | | |
|----|---|--|
| 18 |  | <p>225 行に MaxComPort=4 とあります。この「4」という数字が COM 番号の最大値です。 この数値を「16」に書き換えておきましょう。 COM16 まで開くことができます(TeraTermPro は 16 が最大です)。 上書き保存して、完了です。</p> |
|----|---|--|

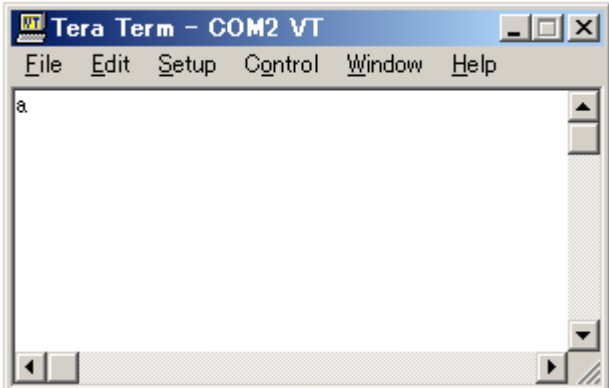
25.7.3 TeraTermProを使って、マイコンと通信しよう

※R8C Writer を使い、「uart0.mot」をマイコンに書き込んでおきます。

| | | |
|---|---|--|
| 1 |  | <p>「スタート」→「すべてのプログラム、またはプログラム」→「Tera Term Pro」→「Tera Term Pro」で Tera Term Pro が立ち上がります。</p> |
| 2 |  | <p>最初に、接続先を確認する画面が出てきます。</p> |
| 3 |  | <p>「Serial」を選んで、ポート番号を選びます。ポート番号は、R8C Writer で選択している番号と同じ番号にします。選択後、OK をクリックして次へ進みます。</p> |

| | | |
|---|---|---|
| 4 |  | 立ち上がりました。 |
| 5 |  | 「Setup→Serial port」を選択します。 |
| 6 |  | <p>通信設定を確認します。画面のように設定して、OKをクリックします。</p> <p>「Port」は R8C Writer で設定していた番号と同じ番号にしてください。</p> <p>「Baud rate」は、UART0 で設定した通信速度と同じ通信速度に合わせてください。</p> |

| | | |
|---|--|---|
| 7 |  | <p>①電源スイッチを ON にします。写真でいうとスイッチを右側に倒します。</p> <p>②リセットボタンを押します。</p> <p>このとき、マイコンボード左下にある2個のLED が次の状態になっているか確認してください。</p> <p>(POWER):点灯 (U S B):点灯</p> |
|---|--|---|

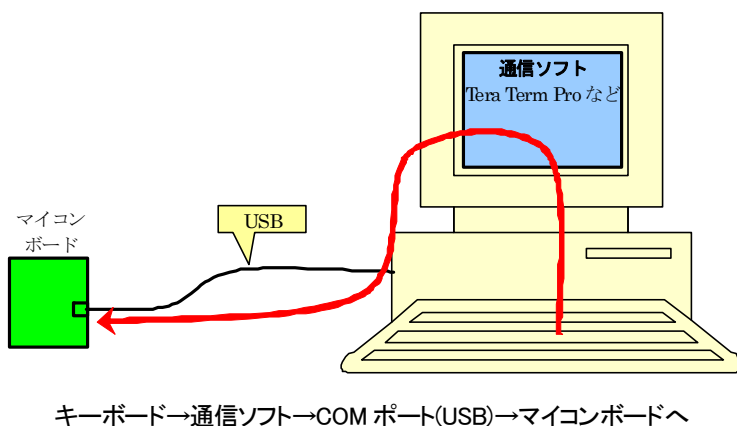
| | | |
|---|--|--|
| 8 |  | <p>これで準備が整いました。</p> <p>aキーを押してみてください。</p> <p>TeraTermPro のウィンドウ内に「a」が表示されます。</p> <p>もし、表示されない場合は、</p> <ul style="list-style-type: none"> ・マイコンボードと USB 接続できていない ・COM ポートの番号が合っていない ・TeraTermPro の設定が合っていない <p>などの理由が考えられます。もう一度、手順を確認してください。</p> |
|---|--|--|

25.7.4 TeraTermProに表示される文字について

ワープロソフト(Microsoft の「Word」など)は、キーボードから入力された文字をソフト上に表示します。TeraTermPro は、一見同じように動作しますが、実はキーボードから打ち込んだ文字が表示されているわけではありません。仕組みを説明します。

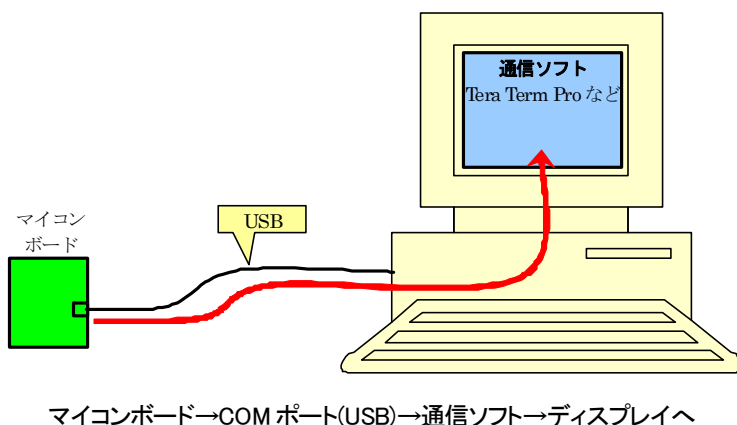
(1) キーボードに入力されたデータをマイコンボードで受信

通信ソフト(TeraTermPro など)は、キーボードに入力された文字を、COM ポートを通じてマイコンボードへ送ります。その様子を下図に示します。



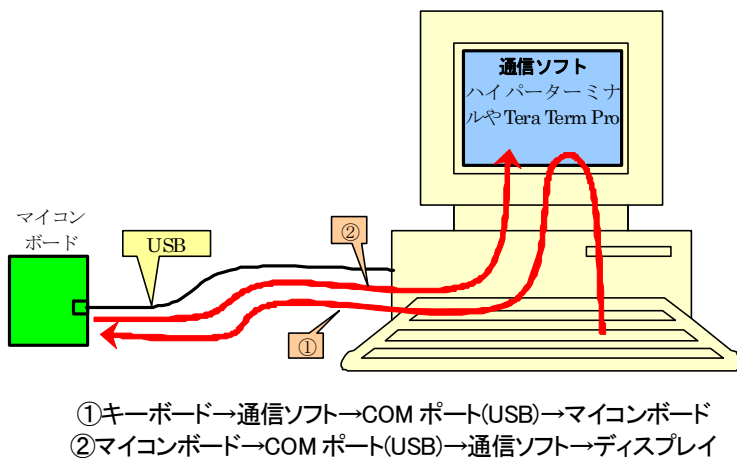
(2) マイコンボードから送られたデータを、パソコンに表示

通信ソフト(TeraTermPro など)は、マイコンボードから送られてきた文字データを、COM ポートを通じて画面に表示します。その様子を下図に示します。



(3) 今回の実習の通信経路

今回の実習の通信経路を下図に示します。



今回、マイコンに書き込んだプログラム(uart0.c)は、受信したデータをそのまま送信するだけなので、キーボードから入力された文字が通信ソフト上に表示されるように見えます。しかし、通信ソフトに表示された文字は、マイコンボードから送られてきたデータを表示させているだけなのです。たまたま、キーボードに打ち込んだ文字と表示された文字が同じだっただけのことです。

試しに、マイコンボードの電源を切って、キーボードに文字を入力してみてください。通信ソフト上に打ち込んだ文字が表示されるでしょうか？

25.8 演習

(1) uart0.c の main 関数を下記のように書き換えてプログラムを書き込み、TeraTermPro を立ち上げ、キーボードから文字を入力して動作を確かめなさい。

```
void main( void )
{
    unsigned char    d;
    int              ret;

    init();           /* SFR の初期化 */

    while( 1 ) {
        ret = get_uart0( &d );
        p6 = d;
        if( ret == 1 ) put_uart0( d + 1 );    // + 1 を追加
    }
}
```

26. printf、scanf文を使った通信(プロジェクト:uart0_printf)

26.1 概要

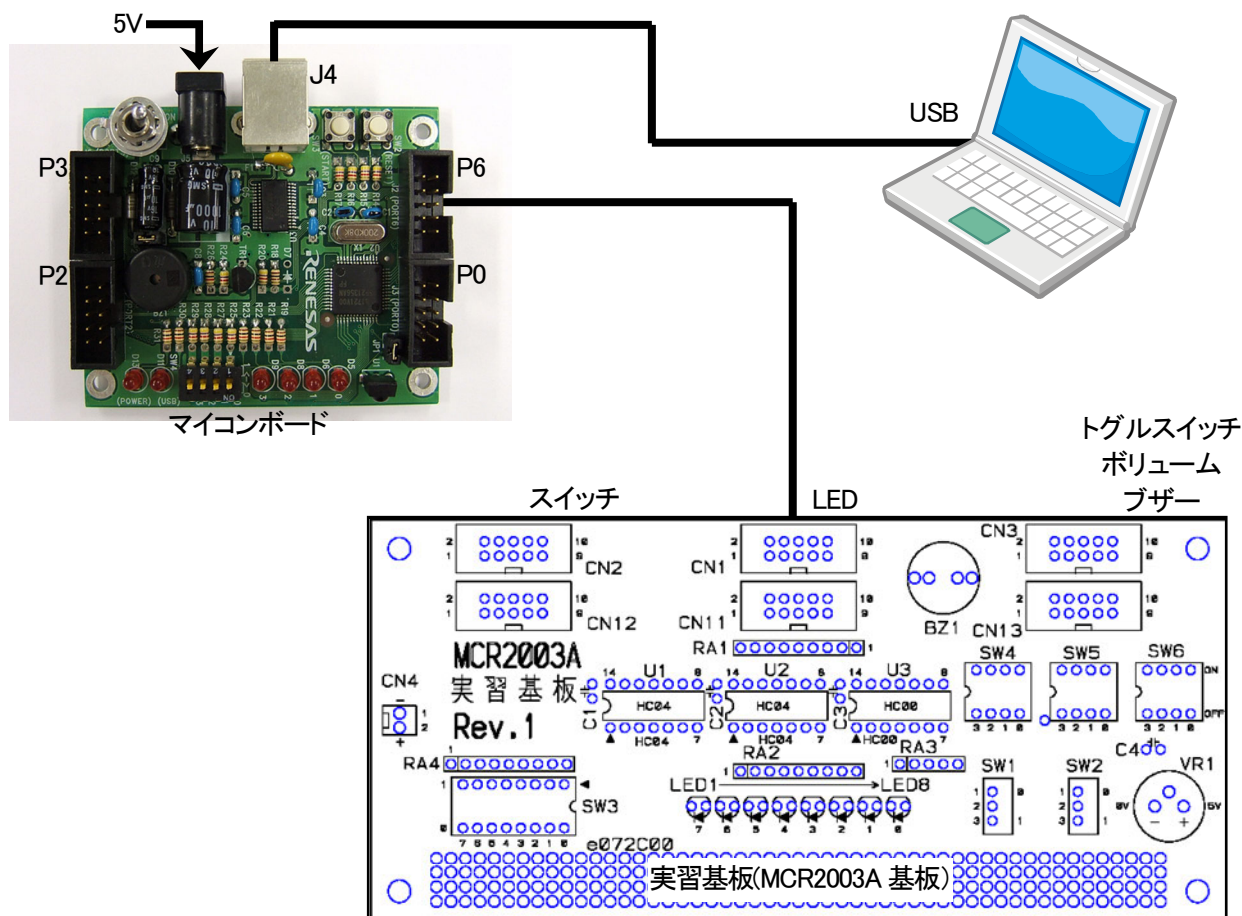
通信は、プロジェクト「uart0」と同じですが、本章ではパソコンの C 言語でよく使われる printf 文、scanf 文をマイコンで使う方法を説明します。

26.2 接続

■使用ポート

| マイコンのポート | 接続内容 |
|------------------|---|
| J4 (USB コネクタ) | USB コネクタを通して、パソコン(通信ソフト)のキーボードから打ち込んだ文字コードを LED へ出力します。 |
| P6 (J2) | 実習基板の LED 部など、出力機器を接続します。 |

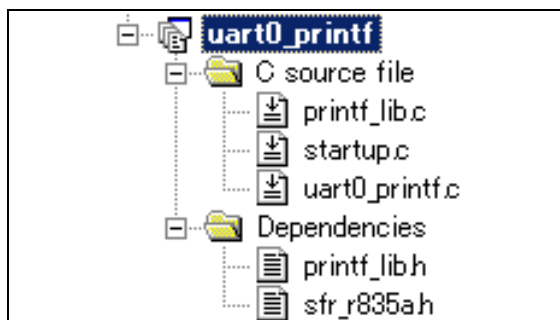
■接続例



■操作方法

パソコン側は TeraTermPro やハイパーターミナルなどの通信ソフトを使います。キーボードから 0～255 までの数値を入力して、エンターキーを押すと、LED へ入力した値が出力されます。256 以上の値や、数値以外の文字も入力して、どうなるか試してみてください。

26.3 プロジェクトの構成



| | ファイル名 | 内容 |
|---|----------------|---|
| 1 | startup.c | 固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化 (初期値のないグローバル変数、初期値のあるグローバル変数の設定) などを行います。このファイルは共通で、どのプロジェクトもこのファイルから実行されます。 |
| 2 | uart0_printf.c | 実際に制御するプログラムが書かれています。R8C/35A の内蔵周辺機能(SFR)の初期化も行います。 |
| 3 | printf_lib.c | printf 文、scanf 文を使用するためのライブラリです。 このファイルは変更しません。 |
| 4 | sfr_r835a.h | R8C/35A マイコンの内蔵周辺機能を制御するためのレジスタ (Special Function Registers)を定義したファイルです。 |

26.4 プログラム「uart0_printf.c」

```

1 :  /******
2 :  /* 対象マイコン R8C/35A
3 :  /* ファイル内容 printf文、scanf文の使用例
4 :  /* バージョン Ver. 1.20
5 :  /* Date 2010.04.19
6 :  /* Copyright ルネサスマイコンカーラー事務局
7 :  /* 日立インターメディックス株式会社
8 :  /******
9 :  /*
10 :  入力 : UART0(パソコンのキーボードで入力したデータ)
11 :  ※パソコンはTeraTermProなどの通信ソフトを使用します
12 :  出力 : P6_7-P6_0(LEDなど)
13 :
14 :  TeraTermProなどの通信ソフトを通して、キーボードから入力した数値を
15 :  ポート6に接続したLEDなどへ出力します。
16 :  C言語でおなじみのprintf文、scanf文を使用した演習です。
17 :  */
18 :
19 :  /*=====
20 :  /* インクルード
21 :  /*=====
22 :  #include <stdio.h>
23 :  #include "sfr_r835a.h" /* R8C/35A SFRの定義ファイル
24 :  #include "printf_lib.h" /* printf使用ライブラリ
25 :
26 :  /*=====
27 :  /* シンボル定義
28 :  /*=====
29 :

```



```

30 : /*=====*/
31 : /* プロトタイプ宣言 */
32 : /*=====*/
33 : void init( void );
34 :
35 : /*=====*/
36 : /* メインプログラム */
37 : /*=====*/
38 : void main( void )
39 : {
40 :     int    i, ret;
41 :     char    c;
42 :
43 :     init();                /* SFRの初期化 */
44 :     init_uart0_printf( SPEED_9600 ); /* UART0とprintf関連の初期化 */
45 :     asm( " fset I " );    /* 全体の割り込み許可 */
46 :
47 :     printf( "Hello World!\n" );
48 :
49 :     while( 1 ) {
50 :         printf( "Input data : " );
51 :         ret = scanf( "%d", &i );
52 :         if( ret == 1 ) {
53 :             printf( "Get data : %d\n", i );
54 :             p6 = i;
55 :         } else {
56 :             printf( "Data Error!!\n" );
57 :             scanf( "%*[^\n]" );
58 :         }
59 :     }
60 : }
61 :
62 : /*=====*/
63 : /* R8C/35A スペシャルファンクションレジスタ (SFR)の初期化 */
64 : /*=====*/
65 : void init( void )
66 : {
67 :     int i;
68 :
69 :     /* クロックをXINクロック (20MHz)に変更 */
70 :     prc0 = 1;                /* プロテクト解除 */
71 :     cm13 = 1;                /* P4_6, P4_7をXIN-XOUT端子にする */
72 :     cm05 = 0;                /* XINクロック発振 */
73 :     for(i=0; i<50; i++ );    /* 安定するまで少し待つ(約10ms) */
74 :     ocd2 = 0;                /* システムクロックをXINにする */
75 :     prc0 = 0;                /* プロテクトON */
76 :
77 :     /* ポートの入出力設定 */
78 :     prc2 = 1;                /* PD0のプロテクト解除 */
79 :     pd0 = 0xe0;              /* 7-5:LED 4:MicroSW 3-0:Sensor */
80 :     p1 = 0x0f;               /* 3-0:LEDは消灯 */
81 :     pd1 = 0xdf;              /* 5:RXD0 4:TXD0 3-0:LED */
82 :     pd2 = 0xfe;              /* 0:PushSW */
83 :     pd3 = 0xfb;              /* 4:Buzzer 2:IR */
84 :     pd4 = 0x83;              /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF */
85 :     pd5 = 0x40;              /* 7:DIP SW */
86 :     pd6 = 0xff;              /* LEDなど出力 */
87 : }
88 :
89 : /*=====*/
90 : /* end of file */
91 : /*=====*/

```

26.5 printf文、scanf文を使おう！

C 言語を習い始めると、パソコン上で動作するプログラムの場合は決まって以下のようなプログラムを作成します。

```
#include <stdio.h>
void main( void )
{
    printf("Hello World!\n") ;
}
```

コンパイルして実行してみるとパソコンの画面に、下記のように表示されます。

```
Hello World!
```

printf 関数が画面に表示する作業を行ってくれているのです。printf 関数は標準入出力ライブラリの「stdio.h」ファイルをインクルードすると使用できます。パソコンの場合は、表示したい内容をディスプレイに出力したり、キーボードから文字を入力したりすることができます。

しかしマイコンが組み込まれている装置は、特定の機器に組み込んで使用することが主な目的のため、ディスプレイやキーボードが付いていることはほとんどありません。ルネサス統合開発環境のCコンパイラは、ANSI C 規格に準じているため printf や scanf などの関数が用意されています。しかし、出力先や入力元が無い(分からない)ため、実行しても何も起こらないのです。**今回は、その printf 関数と scanf 関数を実行できるようにしてみました！！**

といっても、マイコンボードにディスプレイやキーボードを簡単に接続することはできません。接続する回路や制御プログラムが大変になります。

マイコンボードにプログラムを書き込むとき、マイコンボードとパソコンは USB で接続します。パソコンはミニマイコンカーVer.2 を COM ポートとして認識します。この COM ポートを通して、パソコンの画面やキーボードをあたかもマイコンボードの装置であるかのごとく使用することができます。ただ、パソコンの画面やキーボードはマイコンボードとパソコンを USB ケーブルで接続するだけでは使用できず、「ハイパーターミナル」や「Tera Term Pro」などの通信ソフトを経由して使用します。

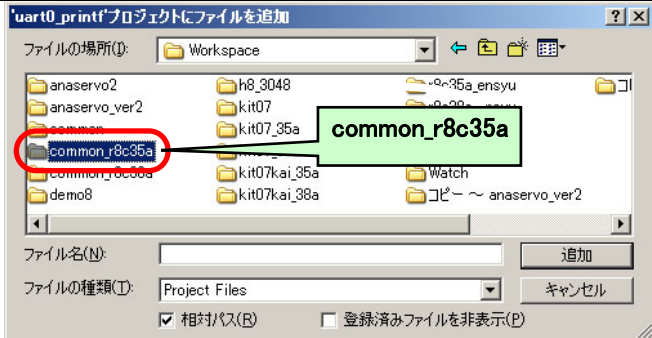
※printf 関数は多くのメモリを消費するため、R8C マイコンでは「%e, %E, %f, %g, %G」の変換指定文字は使用できません。

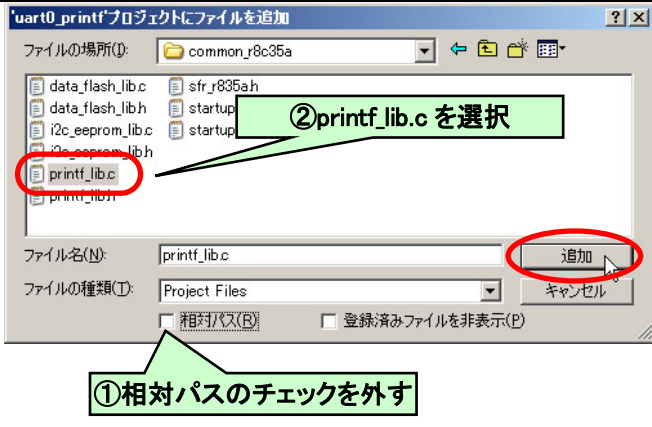
26.6 プログラムの解説「printf_lib.c」

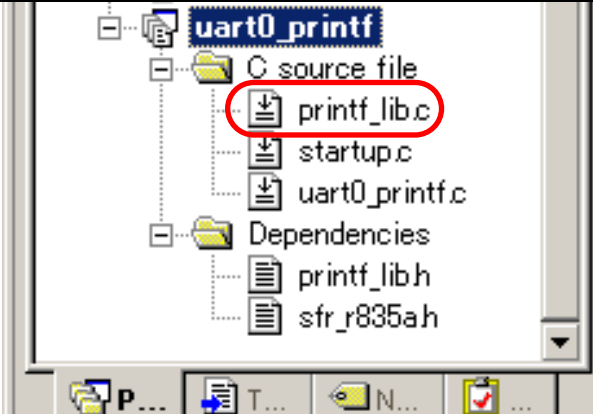
「printf_lib.c」は、R8C/35A で printf 文、scanf 文を COM ポートを通して使えるようにするためのファイルです。printf 文、scanf 文を使うときは、プロジェクトにこのファイルを追加します。このファイルは全プロジェクト共通のファイルです。このファイルを変更すると、このファイルを使っている別のプロジェクトにも影響しますので変更するときは気をつけてください。基本的には変更する必要がありません。

26.6.1 「printf_lib.c」の追加

| | | |
|---|--|---|
| 1 | | <p>「printf_lib.c」がまだ追加されていないとします (サンプルプログラムには最初から追加されています)。</p> |
| 2 | | <p>「プロジェクト→ファイルの追加」を選択します。</p> |
| 3 | | <p>一つ上へボタンを何度かクリックして、Cドライブへ移動します。</p> |

| | | |
|---|---|--|
| 4 |  | <p>Cドライブ ↓ Workspace ↓ common_r8c35a</p> <p>フォルダを開きます。</p> |
|---|---|--|

| | | |
|---|---|---|
| 5 |  | <p>①相対パスのチェックを外します。 ②「printf_lib.c」を選択します。 ③追加をクリックします。</p> |
|---|---|---|

| | | |
|---|---|--------------------------------|
| 6 |  | <p>「printf_lib.c」が追加されました。</p> |
|---|---|--------------------------------|

26.6.2 関数一覧

■init_uart0_printf関数

| | |
|-----|--|
| 書式 | void init_uart0_printf(int sp) |
| 内容 | printf 文の出力先や scanf 文の入力先を、UART0 にするように設定します。 |
| 引数 | <p>int 通信速度</p> <p>通信速度を設定します。設定できる内容は下記のとおりです。</p> <p>SPEED_4800 …通信速度を 4800bps に設定します。</p> <p>SPEED_9600 …通信速度を 9600bps に設定します。</p> <p>SPEED_19200 …通信速度を 19200bps に設定します。</p> <p>SPEED_38400 …通信速度を 38400bps に設定します。</p> <p>その他の設定は、データ長:8bit、パリティビット:無し、ストップビット:1bit、の設定で変更できません。</p> <p>なお、init_uart0_printf 関数実行後、割り込みを許可してください。</p> |
| 戻り値 | 無し |
| 使用例 | <pre>init_uart0_printf(SPEED_9600); // 9600bps で通信 asm(" fset I "); // init_uart0_printf 関数実行後、割り込み許可する</pre> |

■printf関数

| | |
|-----|---|
| 書式 | int printf(const char *format, ...); |
| 内容 | UART0 に format で指定する書式に従い、データを出力します。 |
| 引数 | 付録を参照してください。 |
| 戻り値 | 正常時:出力した文字 異常時:EOF |
| 使用例 | printf("i は、%d です。", i); // i の値を表示 |
| その他 | <p>プログラムの最初に stdio.h をインクルードしてください。</p> <pre>#include <stdio.h></pre> |

■scanf関数

| | |
|-----|---|
| 書式 | int scanf(const char *format, ...); |
| 内容 | UART0 から format で指定する書式に従い、データを読み込みます。 |
| 引数 | 付録を参照してください。 |
| 戻り値 | 正常時:読み込んで変換されたデータの数 入力終了または異常時:EOF |
| 使用例 | scanf("%d, &i); // i に 10 進数を読み込む |
| その他 | <p>プログラムの最初に stdio.h をインクルードしてください。</p> <pre>#include <stdio.h></pre> |

26.7 プログラムの解説「uart0_printf.c」

26.7.1 インクルード部分

printf 文、scanf 文を使うために、専用のヘッダファイルをインクルードします。

```

19 : /*=====*/
20 : /* インクルード */
21 : /*=====*/
22 : #include <stdio.h>
23 : #include "sfr_r835a.h" /* R8C/35A SFRの定義ファイル */
24 : #include "printf_lib.h" /* printf使用ライブラリ */

```

| | |
|--------------|--|
| stdio.h | ファイルがあるフォルダの位置は、 「C:\Program Files\Renesas\Hew\Tools\Renesas\nc30wa\545r00\inc30」です。波線部分は、ルネサス統合開発環境のバージョンによって異なります。 このファイルを取り込む(インクルード)ことにより、printf 文や scanf 文を使えるようにします。 このファイルだけでは、printf 文や scanf 文が使えるようになるだけで、UART0 を使った通信は行えません。 |
| printf_lib.h | ファイルがあるフォルダの位置は、「C:\Workspace\common_r8c35a」です。 このファイルを取り込む(インクルード)ことにより、printf 文の出力先や scanf 文の入力先を、UART0 にするように設定します。 |

26.7.2 main関数－初期化部分

```

35 : /*=====*/
36 : /* メインプログラム */
37 : /*=====*/
38 : void main( void )
39 : {
40 :     int    i, ret;
41 :     char   c;
42 :
43 :     init(); /* SFRの初期化 */
44 :     init_uart0_printf( SPEED_9600 ); /* UART0とprintf関連の初期化 */
45 :     asm( "fset I " ); /* 全体の割り込み許可 */

```

| | |
|------|---|
| 43 行 | init 関数を実行します。 init 関数では、ポートの入出力設定を行います。 シリアルコミュニケーション 0(UART0)の設定は、init 関数では行いません。プロジェクト「uart0」とは違いますので、気をつけてください。 |
| 44 行 | init_uart0_printf 関数で、UART0 の初期化と printf 文、scanf 文を使えるようにしています。UART0 の通信速度は、9600bps に設定します。 |
| 45 行 | printf 文でデータを送信する際、送信割り込みを使うので、UART0 の初期化後に全体の割り込みを許可します。 |

26.7.3 main関数ー通信部分

```

47 :    printf( "Hello World!¥n" );
48 :
49 :    while( 1 ) {
50 :        printf( "Input data : " );
51 :        ret = scanf( "%d", &i );
52 :        if( ret == 1 ) {
53 :            printf( "Get data : %d¥n", i );
54 :            p6 = i;
55 :        } else {
56 :            printf( "Data Error!!¥n" );
57 :            scanf( "%*[^¥n]" );
58 :        }
59 :    }
60 : }

```

| | |
|---------|---|
| 47 行 | 最初に「Hello World!」と表示します。 |
| 50 行 | 「Input data : 」と表示して、データ入力待ちメッセージを表示します。 |
| 51 行 | データの入力を待ちます。エンタキー入力で次の行へ進みます。正しくデータが入力されたら変数 i に入力値が入ります。 |
| 52 行 | scanf 関数の戻り値をチェックします。 |
| 53,54 行 | データが正常に入力されていたら受信データをパソコンに送り返して、ポート 6 へ出力します。 |
| 56,57 行 | データが異常ならエラーメッセージを表示して、受信バッファをクリアします。 |

26.7.4 受信バッファのクリア

プログラムの 57 行目で、バッファのクリアをするために代入抑止文字を使っています。

```
scanf( "%*[^¥n]" );
```

scanf 内の文字を分解すると下記ようになります。

| | | | | | |
|---|---|---|---|----|---|
| % | * | [| ^ | ¥n |] |
| ① | ② | ③ | ④ | ⑤ | |

- ① 変換指定文字の開始です。
- ② 読み捨てる意味です。以後の書式を読み捨てます。
- ③ ⑤と対で、その中の文字のみを読み飛ばします。
- ④ 読み飛ばす文字は¥n、すなわち改行コードです。

総合すると、バッファを読み捨てますが、「¥n」のみ読み捨てるのを飛ばします。すなわち「¥n」のみが残ります。「¥n」により scanf 関数はバッファの終了と判断して次へ進みます。

もう一度、最初の入力部分の scanf 関数を見ると、

```
51 :          ret = scanf( "%d", &i );
```

ここで、「aaa(改行)」を入力したとします(改行=エンタ)。入力バッファには、

```
aaa¥n
```

と文字が保存されます。改行が入力されると変換が開始されます。変換指定文字「%d」は、10 進数入力です。それ以外のアルファベットなどの文字が入力されてしまうと「10 進数入力にもかかわらず数字以外のものが入力されていた場合は、その文字を読み込まずに作業は終了する」に当てはまってしまいます。この場合、エラー終了してしまいます。バッファはクリアされません。ここで再度、数値入力のため、

```
ret = scanf( "%d", &i );
```

という命令を記述するとどうなるでしょう。

バッファはクリアされておらず、親切にも(!!) 改行コードも有りますので、scanf 関数はすでにキー入力されたと勘違いして変換を開始してしまいます。もちろんエラーですのでバッファはクリアしないまま次へ移ります。これを繰り返すと無限ループになり暴走してしまいます。そこで、入力値が正しいかどうか scanf 関数の戻り値を判定してエラーがあればバッファをクリアします。scanf 関数の戻り値は先に説明したとおり正常に終了すると、読み込んで変換されたデータの数が返ってきます。ここでは1です。1以外ならエラーと判断できます。

```
52 :          if( ret == 1 ) {
53 :              printf( "Get data : %d¥n", i );
54 :              p6 = i;
```

戻り値を格納した ret 変数が 1 なら、printf 関数で値を表示して、ポート6へその値を出力します。

```
55 :          } else {
56 :              printf( "Data Error!!¥n" );
57 :              scanf( "%*[^¥n]" );
58 :          }
```

else 文、すなわち戻り値が1でなくエラーであれば、入力エラーと表示して、バッファをクリアします。**書籍によっては scanf 関数は、変換指定文字以外のデータが入力されたとき暴走するので使わない方が良いと書かれています。うまくバッファをクリアすればこれほど便利な関数はありません。**