短講

Tiny マイコンを使ってみよう！

星 貴之

平成 18 年 1 月 31 日

1. はじめに

近年, 組み込み用途のマイコンは周辺機能を内蔵し, システムを小型に実現できるようになってきている. これを利用すると, 分布型センサシステムなども容易に構築できると考えられる. 本稿では, Renesas Technology の Tiny マイコンシリーズの中でもっともピン数の少ない R8C/15 をテーマとする. R8C/Tiny マイコン・リファレンス・ブック [1], 及びトランジスタ技術 2005 年 4 月号 ～ 2006 年 2 月号 [2] を参考にしている.

2. R8C/15

2·1 外観

R8C/15 (R5F21154SP) の寸法は 6.4 mm × 6.5 mm × 1.45 mm, ピンのピッチは 0.65 mm である [3]. 20 本のピンにはそれぞれ機能が割り当てられている (Fig.1).

2·2 仕様

CPU 16 bit. M16C/60 (三菱電機, 1995) と同じもの.

内蔵メモリ ROM: 16 KB, データフラッシュ: 2 KB, RAM: 1 KB.

I/O ポート 入力専用 (P4_6-7): 2 本, 入/出力: 13 本.

シリアル I/O 同期/非同期 (TxD0, RxD0, CLK0): 1 本, 同期 (SSCK, SSI, SSO, SCS): 1 本.

タイマ 16 bit: 1 本, 8 bit: 2 本.

A-D 変換器 4ch, 10 bit (繰り返しモードでは 8 bit).

クロック 内部: 高速 8 MHz or 低速 125 kHz, 外部: 0 ～ 20 MHz (XIN, XOUT).

電源電圧 3.0 ～ 5.5 V (本稿では Vcc = 5 V とする).

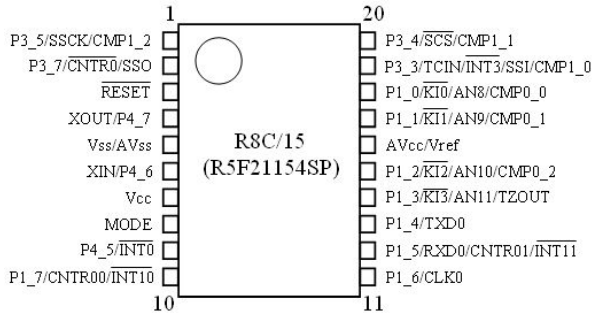

Fig.1 R8C/15 のピンアサイン. ひとつのピンに複数の機能が割り当てられ, その中から選択して使う.

2·3 開発環境

開発に必要なソフトウェアは, 試用期限なしの無償版がトラ技 2005 年 4 月号の付録 CD-ROM に収録されている. これらはルネサステクノロジの HP [4] からダウンロードすることもできる (2. は未確認). 1. には統合開発環境 HEW4 (High-Performance Embedded Workshop 4) が含まれ, 主にこれを使って, C 言語で R8C/15 の動作を記述していく.

1. C コンパイラパッケージ M3T-NC30WA V.5.30 Release 02 無償評価版.
2. Renesas Debugger Package for M16C family V.1.00 Release 00.
3. M16C Flash Starter Ver.2.0.0.04.
4. R8C/14 グループ用レジスタ定義ファイル sfr_r815.inc, sfr_r815.h.
5. フラッシュ開発ツールキット Version 3.3.

2·4 回路構成

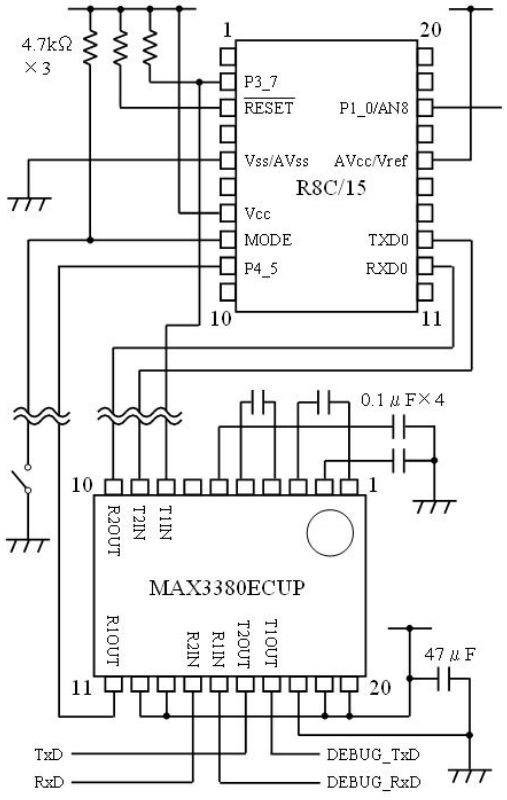

Fig.2 回路構成例. メモリやクロックなどを外部に配置する必要がない.

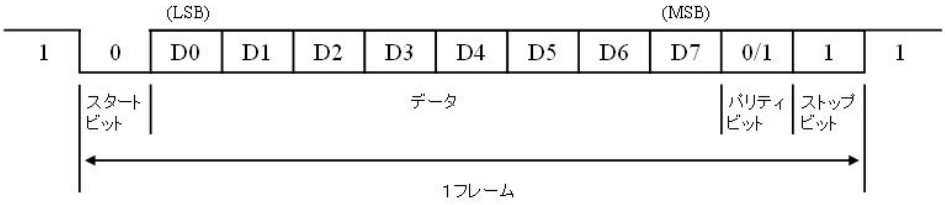

Fig.3 RS-232C の一般的なデータフォーマット.

R8C/15 を使う際の回路の例を Fig.2 に示す. この例では外部クロックは入れず, オンチップオシレータ 8 MHz で動作させる. また 18 番ピンのみを入/出力として使う. またピンを A-D 変換のアナログ入力として使うときには, 過大電圧に対する保護回路を付けると安心 (Fig.5, 最終ページ). MAX3380ECUP は回路側と PC 側の信号電圧を相互に変換する EIA-232 ライン・ドライバ・レシーバである. R8C/15 にプログラムをダウンロードするには MODE 端子のスイッチを ON にし, DEBUG_TxD, DEBUG_RxD を介して RS232C 通信を行なう. 動作中の通信は MODE 端子のスイッチを OFF にし, TxD, RxD を介して行なう.

Fig.2 には描かれていないが, Vcc – Vss 間には適宜 1 μF 程度のバイパスコンデンサを入れる. また未使用ピンは入力モードに設定し, 抵抗を介して Vcc につなぐ (プルアップ). もしくは出力モードに設定して開放.

3. シリアル通信

R8C/15 にはシリアルインターフェース (UART: Universal Asynchronous Receiver Transmitter) 機能があり, それによって PC – R8C/15 間の通信を行なう. そこで用いられるのは, RS232C という調歩同期 (非同期) 通信である.

3·1 データ形式

RS232C で送受信されるデータの 1 フレームを Fig.3 に示す. 無通信状態では HIGH 状態になっており, スタートビット (LOW) を検出すると受信処理を開始する. データは普通は 8 bit (まれに 7 bit). パリティビットはデータ中の HIGH の個数が偶数か奇数かを表すが, 普通は “パリティなし”. ストップビットは 1 (or 2) ビットの HIGH で, フレームの終了を表す.

3·2 接続

R8C/15 は PC のシリアルポートを介して通信を行なう. USB – シリアル変換ケーブルがあれば, シリアルポートのない PC でも OK. 回路側の TxD, RxD は Fig.4 のように D-sub 9 pin メスに接続する. 1, 4, 6 番ピンと 7, 8 番ピンはそれぞれつないでおく.

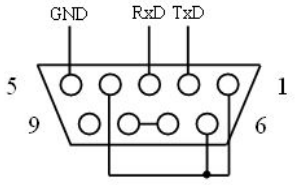

Fig.4 回路側の D-sub 9 pin メスの接続.

3·3 送受信 (PC 側)

RS232C を使ってマイコンを制御する多くの例題では, Windows 標準のハイパーターミナルを使ってデータの送受信・表示を行なっている. センサ出力の場合にはそれを用いた演算がしたいので, C 言語で送受信できるとうれしい. そして実は意外と簡単.

3·3.1 オープン

通常のファイル操作と同様にして CreateFile でポートのハンドルを取得する. PortName は ”COM5” のような感じ. そして通信条件とタイムアウトの設定.

```
HANDLE OpenComm(const char *PortName){
  HANDLE hComm;  DCB dcb;  COMMTIMEOUTS ct;
  /* ポートのハンドルを取得 */
  hComm= CreateFile(
    PortName, GENERIC_READ | GENERIC_WRITE,
    0, NULL, OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL, NULL);
  if(hComm != INVALID_HANDLE_VALUE){
    /* 通信条件 */
    GetCommState(hComm, &dcb);
    dcb.BaudRate= CBR_19200; //19200 bps
    dcb.ByteSize= 8;         //data 8 bit
    dcb.Parity  = NOPARITY;  //パリティなし
    dcb.fParity = FALSE;
    dcb.StopBits= ONESTOPBIT;//stop 1 bit
    dcb.fOutxCtsFlow= FALSE; //フロー制御なし
    dcb.fOutxDsrFlow= FALSE;
    SetCommState(hComm, &dcb);
    /* タイムアウト設定 */
    GetCommTimeouts(hComm, &ct);
    ct.ReadIntervalTimeout= 0;
    ct.ReadTotalTimeoutMultiplier= 0;
    ct.ReadTotalTimeoutConstant= 10000;
    ct.WriteTotalTimeoutMultiplier= 0;
    ct.WriteTotalTimeoutConstant= 10000;
    if(!SetCommTimeouts(hComm, &ct))
        return 0;
    /* バッファ初期化 */
    PurgeComm(hComm, PURGE_TXABORT
      |PURGE_RXABORT|PURGE_TXCLEAR
      |PURGE_RXCLEAR);
    return hComm;
  }
  return 0;
}
```

3·3.2 読み出し

ClearCommError でたまってるバイト数を取得, ReadFile で読む.

```
unsigned ReadComm(HANDLE hComm, char *buf){
  DWORD error;  COMSTAT comstat;
  DWORD readsize;  int len;
  /* バイト数取得 */
  comstat.cbInQue= 0;
  ClearCommError(hComm, &error, &comstat);
  if(sizeof(buf) < comstat.cbInQue)
    len = sizeof(buf);
  /* 読み出し */
  ReadFile(hComm, buf, len, &readsize, NULL);
  return (unsigned int)readsize;
}
```

3·3.3 書き込み

WriteFile で書く.

```
unsigned WriteComm(HANDLE hComm, char *buf){
  DWORD writesize;
  /* 書き込み */
  WriteFile(hComm, buf, sizeof(buf),
    &writesize, NULL);
  return (unsigned int)writesize;
}
```

3·3.4 クローズ

すべてが終わったら閉じる.

```
void CloseComm(HANDLE hComm){
  if(hComm != 0) CloseHandle(hComm);
}
```

3·4 送受信 (R8C/15 側)

R8C/15 側ではフラグをチェックして, 入出力用レジスタを適宜操作する. ポートは TxD, RxD である. トラ技 2005 年 5 月号からサンプルの一部を転載する.

3·4.1 通信条件

```
void set_UART0(void){
  smd0_u0mr   =1; //シリアル I/O モード
  smd2_u0mr   =1; //調歩同期式 8 bit
  te_u0c1     =1; //送信許可
  re_u0c1     =1; //受信許可
  u0brg    =0x40; //19200 bps
  ilvl0_s0ric =1; //受信割込優先レベル
  ir_s0ric    =1; //受信割込要求ビットクリア
}
```

3·4.2 受信

1 文字受信.

```
char UART0_rx(void){
  char data;  char err;
  while(ri_u0c1 != 1); //受信待ち
  data = u0rb1;        //受信データ取り出し
  err = u0rbh & 0xf0;  //エラー取り出し
  return data;
}
```

3·4.3 送信

1 文字, 及び文字列送信.

```
void UART0_tx(char data){
  while(ti_u0c1 != 1); //送信完了待ち
  u0tb = data;         //送信データをセット
}
void UART0_tx_str(){
  while(*str != '\0'){ //文字が \0 になるまで
    UART0_tx(*str);    //1 文字送信
    str++;             //次の文字に移る
  }
}
```

4. I/O ポート

P1_0 ~ P1_7, P3_3 ~ P3_5, P3_7, P4_5 の 13 本の端子は, 設定によって入力, もしくは出力ポートして使うことができる. また外部クロックを接続しない場合は, P4_6 ~ P4_7 が入力専用ポートとして使える. プログラム中では各ポートを論理型変数として扱う.

入力, 及び出力ポートは基本的にはディジタル値 (HIGH/LOW) を扱うが, P1_0 ~ P1_3 はアナログ入力 AN8 ~ AN11 と兼用になっている. また PWM によって等価的にアナログ出力をすることもできる. 各ポートは通常 5 mA の電流値で駆動されるが, P1_0 ~ P1_3 は設定により, LOW を出力するとき 15 mA まで扱える. これによって LED など大電流が必要なデバイスを直接駆動することができる. また入力ポートとして使うときは, 内部でプルアップする設定もある (未使用ピン処理の代用として使える).

5. A-D 変換

R8C/15 には A-D 変換器が搭載されており, AN8 ~ AN11 の 4 ch のアナログ入力を逐次変換方式で A-D 変換する. 分解能は 10 bit と 8 bit が選択できる. 動作モードは単発モードと繰り返しモードが選択できる (繰り返しモードでは 8bit のみ).

A-D 変換器の制御はレジスタに値を書き込むことによって行なう. トラ技 2005 年 5 月号からサンプルの一部を転載する (一部改変).

```
/* A-D 変換設定 */
void set_AD_10bit(void){
  ch0=0;  ch1=0;  ch2=1; //AN8
  adgsel0 =1; //ポート P1 グループ選択
  cks0    =1; //周波数 f2 選択
  bits    =1; //10 bit モード
  vcut    =1; //Vref 設定
  smp     =1; //サンプル&ホールドあり
}
/* A-D 変換実行 */
unsigned get_AD_10bit(void){
  unsigned ad_data;
  adst =1;                //A-D 変換開始
  while(adst==1);         //変換終了待ち
  ad_data = ad & 0x03ff; //10 bit を保存
  return ad_data;
}
```

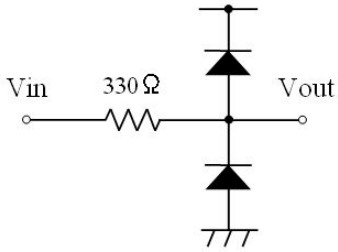

Fig.5 過大電圧に対する保護回路. Vin に Vcc 以上もしくは Vss 以下の電圧が加わったとき, 電流はダイオードを通してやりとりされ, Vout に接続されたピンが保護される.

6. おわりに

本稿では Tiny マイコン R8C/15 の概要を説明した. 他にも割り込み処理やタイマ機能 (ハード的に実現したカウンタ) などが使えて, ワンチップで大抵のことはできるので, 是非活用したい.

またインターネット上には RS232C 通信用のクラスを公開している方もいるので, それを使わせてもらう手もある [5][6].

参考文献

[1] 新海栄治 編著: R8C/Tiny マイコン・リファレンス・ブック, CQ 出版社, 2005.
[2] トランジスタ技術, CQ 出版社, 2005/4–2006/2.
[3] R8C/14, R8C/15 グループデータシート, ルネサステクノロジ, 2005.
[4] ルネサステクノロジ HP, http://japan.renesas.com/homepage.jsp.
[5] 栗原徹: RS-232-C on Visual C++ 6.0, http://www.alab.t.u-tokyo.ac.jp/~bond/doc/rs232c.html.
[6] 源三郎: シリアルポートによるデータ送受信プログラム, http://www.lares.dti.ne.jp/~parasa/200106.html.