```
/*****************************************************************************/
/***                    USART：シリアルポートドライバ                        ***/
/***                         調歩同期通信用                                  ***/
/***                                                  by : S.Suzuki    ***/
/*****************************************************************************/

#include <avr/io.h>
#include <avr/interrupt.h>

#include "hdwareInfo.h"     // ←BYTE,WORD等独自の型宣言あり
#include "USART.h"

/*  送受信用バッファ＆ポインタ ***********************************************/

#define TXBUFSZ (64)                    /* 送信バッファバイト数（255以下） */
#define RXBUFSZ (128)                   /* 受信バッファバイト数（255以下） */

static BYTE st_TxSize;                  /* 送信データバイト数 */
static BYTE st_TxRptr;                  /* 送信リードポインタ */
static volatile BYTE st_RxWptr;         /* 受信ライトポインタ */
static BYTE st_RxRptr;                  /* 受信リードポインタ */
static char st_TxBuff[TXBUFSZ];         /* 送信バッファ       */
static char st_RxBuff[RXBUFSZ];         /* 受信バッファ（リングバッファ） */

/* ステータスフラグの構成  ***************************************************/

typedef union {
    BYTE    ALL;
    struct{
        BYTE    B0:1;               /* b0: */
        BYTE    B1:1;               /* b1: */
        BYTE    B2:1;               /* b2: */
        BYTE    B3:1;               /* b3: */
        BYTE    OFERR:1;            /* b4:受信オーバーフローあり */
        BYTE    RXERR:1;            /* b5:受信異常あり */
        BYTE    TXERR:1;            /* b6:送信異常あり */
        BYTE    SENDING:1;          /* b7:送信中       */
    } BIT;
} UARTSTAT;

static volatile UARTSTAT st_stTxRxStat;     /* ステータスフラグ   */

/* ビットレート設定コードのテーブル ******************************************/
/* 注意 : (1)PCLK=8MHzであること。（かっこ）内は誤差。                        */
/*      : (2)下表の値はUBRR0Lに設定する値。この場合UBRR0Hは常に0x00。         */
/*      : (3)UCSR0A.U2X0='0'であること。                                      */

static BYTE st_byBitRateCode[4]={
    103,    /*  4800bps(0.2%) */
    51,     /*  9600bps(0.2%) */
    25,     /* 19200bps(0.2%) */
    12,     /* 38400bps(0.2%) */
};

/*****************************************************************************/
/*                    USART：シリアルポートの初期化                          */
/*****************************************************************************/
/* 注意 : (1)システムクロックは、8MHzであること。                             */
/*      : (2)割り込みマスク状態で、起動後に一度呼ぶこと。                     */
/* 備考 : (1)USCR0B.TXEN0=1でPD1端子はTXDになる。                             */
/*      : (2)UCSR0B.RXEN0=1でPD0端子はRXDになる。                             */
/* 引数 : byMode = [76543210]                                                */
/*      :               ||||||||                                             */
/*      :               |||||||└_0_ X                                        */
/*      :               ||||||└__1_ X                                        */
/*      :               |||||└___2_ X                                        */
/*      :               ||||└____3_ ストップビット (0:1bit, 1:2bit)          */
/*      :               |||└_____4_ パリティ (0:EVEN, 1:ODD)                 */
/*      :               ||└______5_ パリティ (0:なし, 1:あり)               */
/*      :               |└_______6_ データ長 (0:8bit, 1:7bit)               */
/*      :               └________7_ X                                        */
/*      : byRateNum = 通信速度 (0=4800, 1=9600, 2=19200, 3=38400)            */
/* 戻値 : Init_USART() ==  TRUE : 正常                                       */
/*      :               == FALSE : 異常                                      */
```

```
 75
 76 BOOL Init_USART(BYTE byMode, BYTE byRateNum)
 77 {
 78     BYTE    byWork;
 79
 80     /* 全エラーフラグクリア */
 81     st_stTxRxStat.ALL = 0;
 82     /* ポインタ初期化 */
 83     st_TxSize = 0;                  /* 送信データバイト数 */
 84     st_TxRptr = 0;                  /* 送信リードポインタ */
 85     st_RxWptr = 0;                  /* 受信ライトポインタ */
 86     st_RxRptr = 0;                  /* 受信リードポインタ */
 87
 88     /* I/Oポートの初期化 (USCR0B.TXEN0=0,USCR0B.RXEN0=0としたときPD1,PD0が入力ポートになるよう
    明示的に設定) */
 89     DDRD  &= ~(_BV(DDD1) | _BV(DDD0));      /* PD1,PD0を入力ポートにする */
 90     PORTD &= ~(_BV(PORTD1) | _BV(PORTD0));  /* PD1,PD0のプルアップOFF    */
 91
 92     /* USARTの通信データ構成設定 */
 93     byWork = 0x00;                  /* UCSR0B.UCSZ02='0'とする */
 94     /* パリティあり指定の場合 */
 95     if((byMode & 0x20) != 0){
 96         byWork |= _BV(UPM01);
 97         /* 奇数パリティの場合 */
 98         if((byMode & 0x10) != 0){
 99             byWork |= _BV(UPM00);
100         }
101     }
102     /* ２ストップビットの場合 */
103     if((byMode & 0x08) != 0){
104         byWork |= _BV(USBS0);
105     }
106     /* ７ビットデータの場合 */
107     if((byMode & 0x40) != 0){
108         byWork |= _BV(UCSZ01);
109     }
110     /* ８ビットデータの場合 */
111     else{
112         byWork |= (_BV(UCSZ01) | _BV(UCSZ00));
113     }
114     UCSR0C = byWork;
115     /* 通信速度（ビットレート）設定 */
116     UBRR0H = 0;
117     UBRR0L = st_byBitRateCode[byRateNum];
118     /* USARTレジスタ初期化 */
119     UCSR0A = 0x00;                  /* U2X0='0', MPCM0='0' */
120     UCSR0B = _BV(RXCIE0) | _BV(RXEN0) | _BV(TXEN0); /* RX割り込み許可、受信許可、送信許可、UCSZ
    02='0' */
121
122     return TRUE;
123 }
124
125 /*****************************************************************************/
126 /*                              送信完了割り込み処理                         */
127 /*****************************************************************************/
128 /* 未使用                                                                    */
129
130 ISR(USART_TX_vect)
131 {
132     /* 送信完了割り込み禁止 */
133     UCSR0B &= ~_BV(TXCIE0);
134 }
135
136 /*****************************************************************************/
137 /*                              送信割り込み処理                             */
138 /*****************************************************************************/
139
140 ISR(USART_UDRE_vect)
141 {
142     /* 送信データなし */
143     if(st_TxSize == 0){
144         /* 送信割り込み禁止 */
145         UCSR0B &= ~_BV(UDRIE0);
146         /* 送信完了を示す */
```

```
147         st_stTxRxStat.BIT.SENDING = 0;
148     }
149     /* 送信データあり  */
150     else{
151         /* データ送信 */
152         UDR0 = st_TxBuff[st_TxRptr++];
153         /* 送信データポインタ更新 */
154         --st_TxSize;
155         if(st_TxSize == 0){
156             /* 送信割り込み禁止 */
157             UCSR0B &= ~_BV(UDRIE0);
158             /* 送信完了を示す */
159             st_stTxRxStat.BIT.SENDING = 0;
160             /* 送信異常フラグをクリア */
161             st_stTxRxStat.BIT.TXERR   = 0;
162         }
163     }
164 }
165 
166 /*****************************************************************************/
167 /*                                受信割り込み処理                           */
168 /*****************************************************************************/
169 
170 ISR(USART_RX_vect)
171 {
172     BYTE    byFlag;
173     char    cRxDat;
174 
175     /* エラーフラグチェック */
176     byFlag = (BYTE)(UCSR0A & (_BV(FE0) | _BV(DOR0) | _BV(UPE0)));
177     if(byFlag != 0){
178         cRxDat = UDR0;              /* Dummy Read */
179         st_stTxRxStat.BIT.RXERR = 1;
180         return;
181     }
182     /* データ受信 */
183     cRxDat = (char)UDR0;
184     /* 終端コード以外の制御コードは無視する */
185     if((cRxDat != UART_TERM) && ((BYTE)cRxDat < 0x20)){
186         return;
187     }
188     /* 受信データを受信バッファに格納する */
189     st_RxBuff[st_RxWptr++] = cRxDat;
190     /* 書き込みポインタ更新 */
191     if(st_RxWptr == RXBUFSZ){
192         st_RxWptr = 0;
193     }
194     /* バッファフルの場合は古い受信データ１行分を破棄する */
195     if(st_RxWptr == st_RxRptr){
196         do{
197             st_RxRptr++;
198             if(st_RxRptr == RXBUFSZ){
199                 st_RxRptr = 0;
200             }
201             /* 受信データ超過エラー */
202             if(st_RxRptr == st_RxWptr){
203                 /* 受信バッファオーバーフローエラーフラグセット */
204                 st_stTxRxStat.BIT.OFERR = 1;
205                 return;
206             }
207         }while(st_RxBuff[st_RxRptr] != UART_HEADER);
208     }
209 }
210 
211 /*****************************************************************************/
212 /***                                                                       ***/
213 /***                        USART送信／受信支援関数                        ***/
214 /***                                                                       ***/
215 /*****************************************************************************/
216 
217 /*****************************************************************************/
218 /*                            USARTタイマーカウント                          */
219 /*****************************************************************************/
220 /* 1mS周期割り込みから呼ばれる                                               */
```

```
221 static volatile WORD wTmrDwnCnt1_1mS;       /* 1mS周期、ダウンカウント */
222
223 void Timer01mS_USART(void)
224 {
225     if(wTmrDwnCnt1_1mS != 0){
226         --wTmrDwnCnt1_1mS;
227     }
228 }
229
230 /*****************************************************************************/
231 /*                              受信データの破棄                             */
232 /*****************************************************************************/
233 /* 引数 : なし                                                               */
234 /* 戻値 : なし                                                               */
235
236 void PurgeRxd_USART(void)
237 {
238     BYTE    bySREG;
239
240     /* 割り込みマスク */
241     bySREG = SREG;
242     cli();
243     /* エラーフラグをすべてクリア */
244     st_stTxRxStat.ALL &= 0x80;
245     /* 受信ポインタ類をクリア */
246     st_RxWptr = 0;                          /* 受信ライトポインタ */
247     st_RxRptr = 0;                          /* 受信リードポインタ */
248     /* 割り込みマスク復帰 */
249     SREG = bySREG;
250 }
251
252 /*****************************************************************************/
253 /*                          受信データの有無をチェック                       */
254 /*****************************************************************************/
255 /* 引数 : なし                                                               */
256 /* 戻値 : CheckRcv_USART() ==  TRUE : 受信あり                               */
257 /*      :                  == FALSE : 受信なし                               */
258
259 BOOL CheckRcv_USART(void)
260 {
261     /* 受信データの有無をチェック */
262     if(st_RxWptr != st_RxRptr){         /* 受信データあり */
263         return TRUE;
264     }
265     return FALSE;
266 }
267
268 /*****************************************************************************/
269 /*                              ASCII文字列送信                              */
270 /*****************************************************************************/
271 /* 注意 : (1)送信終了を待たないで戻る。                                      */
272 /*      : (2)ウォッチドッグタイマWDTは関知していない。WDTを使用していて待ち   */
273 /*      :    時間wTimeOutが長いと、WDTが働いてリセットしてしまう。           */
274 /* 引数 : wTimeOut = 送信待ち時間 (0～65535, 1mS単位)                        */
275 /*      :  strData = 送信文字列のアドレス (UART_TERMで終端)                  */
276 /* 戻値 : TxString_USART() ==  TRUE : 正常                                   */
277 /*      :                  == FALSE : 異常、タイムアウト                     */
278
279 BOOL TxString_USART(WORD wTimeOut, const char* strData)
280 {
281     int     nLpcnt;
282     char    cTxDat;
283
284     /* 送信待ち時間セット */
285     wTmrDwnCnt1_1mS = wTimeOut;
286     /* 送信中の場合は待つ */
287     while(st_stTxRxStat.BIT.SENDING == 1){
288         if(wTmrDwnCnt1_1mS == 0){
289             /* 強制的に送信完了にする */
290             st_stTxRxStat.BIT.SENDING = 0;
291             /* 送信異常ありを示す */
292             st_stTxRxStat.BIT.TXERR   = 1;
293             return FALSE;
294         }
```

```
295     }
296
297     /* 送信バイトカウンタをリセット */
298     st_TxSize = 0;
299     /* 終端コードまで送信バッファにコピー */
300     nLpcnt = 0;
301     do{
302         /* 送信データを読み込み、送信バッファにコピー */
303         cTxDat = *(strData++);
304         /* 送信データを送信バッファに格納 */
305         st_TxBuff[nLpcnt] = cTxDat;
306         /* 送信バイト数をカウント */
307         st_TxSize++;
308         /* 終端コードの場合 */
309         if(cTxDat == UART_TERM){
310             break;
311         }
312     }while(++nLpcnt < TXBUFSZ);
313     /* 送信リードポインタリセット */
314     st_TxRptr = 0;
315
316     /* 送信中フラグセット */
317     st_stTxRxStat.BIT.SENDING = 1;
318     /* 送信割り込み許可 */
319     UCSR0B |= _BV(UDRIE0);
320
321     return TRUE;
322 }
323
324 /*****************************************************************************/
325 /*                              ASCII文字列受信                              */
326 /*****************************************************************************/
327 /* 注意 : (1)戻り値の受信バイト数には、ヘッダコードUART_HEADERを含むが、       */
328 /*      :    終端コードUART_TERMは含まない。                                 */
329 /*      : (2)戻値の受信バイト数は、受信した文字数（１文字= 1byte）である。    */
330 /*      : (3)ウォッチドッグタイマWDTは関知していない。WDTを使用していて待ち   */
331 /*      :    時間wTimeOutが長いと、WDTが働いてリセットしてしまう。           */
332 /* 引数 : wTimeOut = 受信待ち時間（0～65535, 1mS単位）                       */
333 /*      :  wBuffSz = 受信バッファのサイズ（7～）                             */
334 /*      : pRcvBuff = 受信文字列の格納アドレス                                */
335 /* 戻値 : RxString_USART() == 受信バイト数（終端コードは含まない）           */
336 /*      :                  == 0 : 異常あり                                   */
337
338 int RxString_USART(WORD wTimeOut, WORD wBuffSz, char* pRcvBuff)
339 {
340     int     nLength;
341     char    cRxDat;
342
343     /* 受信待ち時間セット */
344     wTmrDwnCnt1_1mS = wTimeOut;
345     /* ヘッダコードの受信を待つ */
346     do{
347         /* 受信を待つ */
348         while(st_RxRptr == st_RxWptr){
349             /* タイムアウトなら異常終了 */
350             if(wTmrDwnCnt1_1mS == 0){
351                 /* 受信異常ありを示す */
352                 st_stTxRxStat.BIT.RXERR = 1;
353                 return 0;
354             }
355         }
356         /* 受信データを読み込む */
357         cRxDat = st_RxBuff[st_RxRptr++];
358         if(st_RxRptr == RXBUFSZ){
359             st_RxRptr = 0;
360         }
361     }while(cRxDat != UART_HEADER);
362     /* 受信したヘッダコードを格納 */
363     *(pRcvBuff++) = cRxDat;
364     nLength = 1;
365
366     /* 受信待ち時間セット */
367     wTmrDwnCnt1_1mS = wTimeOut;
368     /* ヘッダコード以降を受信して格納 */
```

```
369     while(nLength < wBuffSz){
370         while(st_RxRptr == st_RxWptr){
371             /* タイムアウトなら異常終了 */
372             if(wTmrDwnCnt1_1mS == 0){
373                 /* 受信異常ありを示す */
374                 st_stTxRxStat.BIT.RXERR = 1;
375                 return 0;
376             }
377         }
378         /* 受信データを読み込む */
379         cRxDat = st_RxBuff[st_RxRptr++];
380         if(st_RxRptr == RXBUFSZ){
381             st_RxRptr = 0;
382         }
383         /* 受信データをバッファに格納 */
384         *(pRcvBuff++) = cRxDat;
385         /* 終端コードだった場合は終了 */
386         if(cRxDat == UART_TERM){
387             /* 受信異常フラグをクリア */
388             st_stTxRxStat.BIT.RXERR = 0;
389             st_stTxRxStat.BIT.OFERR = 0;
390             return nLength;
391         }
392         nLength++;
393     }
394     /* 受信異常ありを示す */
395     st_stTxRxStat.BIT.RXERR = 1;
396 
397     return 0;
398 }
399 
```