```
/*****************************************************************************/
/*                        IICインタフェースの制御                            */
/*****************************************************************************/
/* ※割り込みは不使用                                                        */

#include <machine.h>
#include <stdlib.h>
#include "typedefine.h"
#include "iodefine.h"


/*****************************************************************************/
/* ICCR1のCKS2～CKS0にセットするコード ***************************************/
                                /* φ=20MHz          */
#define RATE_714     (0)        /* φ/ 28 = 714kHz  */
#define RATE_500     (1)        /* φ/ 40 = 500kHz  */
#define RATE_417     (2)        /* φ/ 48 = 417kHz  */
#define RATE_313     (3)        /* φ/ 64 = 313kHz  */
#define RATE_250     (4)        /* φ/ 80 = 250kHz  */
#define RATE_200     (5)        /* φ/100 = 200kHz  */
#define RATE_179     (6)        /* φ/112 = 179kHz  */
#define RATE_156     (7)        /* φ/128 = 156kHz  */
#define RATE_357     (8)        /* φ/ 56 = 357kHz  */
//#define RATE_250   (9)        /* φ/ 80 = 250kHz  */
#define RATE_208     (10)       /* φ/ 96 = 208kHz  */
//#define RATE_156   (11)       /* φ/128 = 156kHz  */
#define RATE_125     (12)       /* φ/160 = 125kHz  */
#define RATE_100     (13)       /* φ/200 = 100kHz  */
#define RATE_89      (14)       /* φ/224 = 89.3kHz */
#define RATE_78      (15)       /* φ/256 = 78.1kHz */
#define RATE_USE     RATE_250   /* 使用する転送速度 */

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

#define READ_MODE    (0)
#define WRITE_MODE   (1)

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

#pragma abs8(bySlaveID)
static BYTE bySlaveID;                  /* 相手（スレーブ）のアドレス */

/* ソフトウェアタイマの時間 **************************************************/

#define WAIT_100mS (80000)              /* 約100mS */
#define WAIT_50mS  (40000)              /* 約 50mS */
#define WAIT_10mS  ( 8000)              /* 約 10mS */
#define WAIT_1mS   (  800)              /* 約  1mS */
#define WAIT_500uS (  400)              /* 約500uS */
#define WAIT_200uS (  160)              /* 約200uS */
#define WAIT_100uS (   80)              /* 約100uS */

#pragma section
/*****************************************************************************/
/*                           IICインタフェース初期化                         */
/*****************************************************************************/
/* 備考 : H8/3694では、P57,P56はICCR1のICE=1でSCL,SDA端子になる。            */

static void iic2_init(BYTE bySlaveAddr)
{
    /* （相手）スレーブアドレス保存 */
    bySlaveID = bySlaveAddr;
    /* IICリセット */
    IIC2.ICCR2.BIT.IICRST = 1;
    IIC2.ICCR2.BIT.IICRST = 0;
    /* 送受信モード, 転送速度設定 */
    IIC2.ICCR1.BYTE = 0x80 | RATE_USE;  /* ICE=1:バスイネーブル、RCVD=0:受信継続、MST=0,TRS=0:スレーブ受信モード、CKS=4:転送クロック */
    /* 転送モード設定 */
    IIC2.ICMR.BYTE = 0x30;              /* MLS=0:MSBファースト、WAIT=0:DATAとACK連続転送、BCWP=BC2～BC0の書き込み可、BC2～BC0:９ビット転送 */
    IIC2.ICMR.BIT.BCWP = 1;             /* BCWP=BC2～BC0の書き込み禁止 */
    /* 割り込みモード設定 */
    IIC2.ICIER.BYTE = 0x00;             /* TIE=0:送信割り込み禁止、TEIE=0:送信終了割り込み禁止、RIE=0:受信割り込み禁止、NAKIE=0:NAK受信割り込み禁止 */
                                        /* STIE=0:停止条件検出割り込み禁止、ACKE=0:アクノリッジを無視する */
    /* 全ステータスクリア */
    if(IIC2.ICSR.BYTE);                 /* ICSR読み込み */
    IIC2.ICSR.BYTE = 0x00;              /* ACKB=0,全ステータスビットクリア */
    /* 転送フォーマット選択 */
    IIC2.SAR.BYTE  = 0x00;              /* 自局アドレス=0x00、FS=0:IICフォーマット選択 */
}

```

```
/*****************************************************************************/
/*                IICインタフェースによる送受信（割り込み不使用）                */
/*****************************************************************************/

/* スタートコンディション＋スレーブアドレス  ********************************/

static BOOL iic2_start(BYTE byFirst, BYTE byWrite)
{
    WORD    wWaitCnt;
    BOOL    bResult;
    BYTE    byAddr,byTry;

    /* 最初のみバスの開放を待つ（BUSYなら待つ） */
    if(byFirst != 0){
        byTry = 20;
        wWaitCnt = WAIT_10mS;
        while(IIC2.ICCR2.BIT.BBSY != 0){
            if(--wWaitCnt == 0){
                if(--byTry == 0){
                    return FALSE;
                }
                wWaitCnt = WAIT_10mS;
            }
        }
    }

    /* R/Wを加えたスレーブアドレス作成 */
    byAddr = bySlaveID;
    if(byWrite == READ_MODE)
        byAddr |= 0x01;
    /* スレーブアドレス送信 */
    bResult = FALSE;
    byTry = 0;
    do{
        /* マスター送信モード指定 */
        IIC2.ICCR1.BYTE = 0xB0 | RATE_USE;  /* ICE=1:バスイネーブル、RCVD=0:受信継続、MST=1,TRS=1:マスター送信モード、CKS=4:転送クロック */
        /* 開始条件発行 */
        IIC2.ICCR2.BYTE = 0xBD;             /* BBSY=1,SCP=0:開始条件発行、SDAO=1,SDAOP=1:SDA出力制御なし、IICRST=0:IICリセット無効 */
        /* 送信エンプティを待つ */
        wWaitCnt = WAIT_500uS;
        while(IIC2.ICSR.BIT.TDRE == 0){
            if(--wWaitCnt == 0)
                return FALSE;
        }
        /* スレーブアドレス送信 */
        IIC2.ICDRT = byAddr;
        /* 送信終了を待つ */
        wWaitCnt = WAIT_500uS;
        while(IIC2.ICSR.BIT.TEND == 0){
            if(--wWaitCnt == 0)
                goto _RETRY;
        }
        /* ACK'0'受信をチェック */
        if(IIC2.ICIER.BIT.ACKBR == 0){
            bResult = TRUE;
            break;
        }

_RETRY:
        wWaitCnt = WAIT_500uS;
        while(--wWaitCnt != 0);
    }while(++byTry < 3);

    return bResult;
}

/* ストップコンディション  ***************************************************/
/* 備考: RTS命令で10ステートかかるので、RTC-8564NBのtBUF(1.3uS)は確保される。 */

static BOOL iic2_stop(void)
{
    WORD    wWaitCnt;
    BOOL    bResult;

    /* 停止条件検出フラグクリア */
    IIC2.ICSR.BIT.STOP = 0;
    /* 停止条件発行 */
    IIC2.ICCR2.BYTE = 0x3D;                 /* BBSY=0,SCP=0:停止条件発行、SDAO=1,SDAOP=1:SDA出力制御なし、IICRST=0:IICリセット無効 */
    /* 停止条件の検出を待つ */
    bResult = TRUE;
    wWaitCnt = WAIT_500uS;
```

```
162     while(IIC2.ICSR.BIT.STOP == 0){
163         if(--wWaitCnt == 0){
164             bResult = FALSE;
165             break;
166         }
167     }
168     /* 送信終了フラグをクリア */
169     IIC2.ICSR.BIT.TEND = 0;
170     /* スレーブ受信モードに設定 */
171     IIC2.ICCR1.BYTE = 0x80 | RATE_USE;      /* ICE=1:バスイネーブル、RCVD=0:受信継続、MST=0,TRS=0:スレーブ受信モード、CKS=4:転送クロック */
172 
173     return bResult;
174 }
175 
176 
177 /* データブロック送信  ******************************************************/
178 
179 static BOOL iic2_TxBlock(WORD wSize, BYTE *pbyData)
180 {
181     WORD    wWaitCnt;
182 
183     while(wSize > 1){
184         /* データ送信 */
185         IIC2.ICDRT = *(pbyData++);
186         --wSize;
187         /* 送信エンプティを待つ */
188         wWaitCnt = WAIT_500uS;
189         while(IIC2.ICSR.BIT.TDRE == 0){
190             if(--wWaitCnt == 0)
191                 return FALSE;
192         }
193     }
194     /* 最終データ送信 */
195     IIC2.ICDRT = *(pbyData++);
196 //  --wSize;
197     /* 送信終了を待つ */
198     wWaitCnt = WAIT_500uS;
199     while(IIC2.ICSR.BIT.TEND == 0){
200         if(--wWaitCnt == 0)
201             return FALSE;
202     }
203     /* ACK'0'受信をチェック */
204     if(IIC2.ICIER.BIT.ACKBR == 1){
205         return FALSE;
206     }
207 
208     return TRUE;
209 }
210 
211 /*  データブロック受信  ******************************************************/
212 
213 static BOOL iic2_RxBlock(WORD wSize, BYTE *pbyBuff)
214 {
215     WORD    wWaitCnt;
216     BYTE    byDummy;
217     BYTE    byCCR;
218 
219     /* 0バイト受信は無効 */
220     if(wSize == 0)  return FALSE;
221 
222     /* 割り込みマスク */
223     byCCR = get_ccr();
224     set_imask_ccr(1);
225     /* トランスミットエンドフラグをクリア */
226     IIC2.ICSR.BIT.TEND = 0;
227     /* マスター受信モード指定 */
228     IIC2.ICCR1.BYTE = 0xA0 | RATE_USE;          /* ICE=1:バスイネーブル、RCVD=0:受信継続、MST=1,TRS=0:マスター受信モード、CKS=4:転送クロック */
229     /* トランスミットデータエンプティフラグをクリア */
230     IIC2.ICSR.BIT.TDRE = 0;
231     /* 1バイトのみの受信では最初が最後となる */
232     if(wSize == 1){
233         /* ACKデータに'1'を設定 */
234         IIC2.ICIER.BIT.ACKBT = 1;
235         /* 受信ディセーブル */
236         IIC2.ICCR1.BIT.RCVD = 1;
237     }
238     else{
239         /* ACKデータに'0'を設定 */
240         IIC2.ICIER.BIT.ACKBT = 0;
241         /* 受信イネーブル */
242         IIC2.ICCR1.BIT.RCVD = 0;
243     }
```

```
244     /* 受信データ読み込み（最初のデータはダミー） */
245     byDummy = IIC2.ICDRR;
246     /* 割り込みマスク復帰 */
247     set_ccr(byCCR);
248 
249     /* 最初の受信データを待つ */
250     wWaitCnt = WAIT_1mS;
251     while(IIC2.ICSR.BIT.RDRF == 0){
252         if(--wWaitCnt == 0)
253             return FALSE;
254     }
255 
256     /* 最後以外の受信データを読み込み */
257     while(wSize > 1){
258         /* 最後－１番目の受信 */
259         if(wSize == 2){
260             /* ACKデータに'1'を設定 */
261             IIC2.ICIER.BIT.ACKBT = 1;
262             /* 受信ディセーブル */
263             IIC2.ICCR1.BIT.RCVD = 1;
264         }
265         /* 受信データ読み込み */
266         *(pbyBuff++) = IIC2.ICDRR;
267         --wSize;
268         /* データ受信を待つ */
269         wWaitCnt = WAIT_500uS;
270         while(IIC2.ICSR.BIT.RDRF == 0){
271             if(--wWaitCnt == 0)
272                 return FALSE;
273         }
274 
275     }
276 
277     /* 最後の受信データを読み込み */
278     *(pbyBuff++) = IIC2.ICDRR;
279 //  --wSize;
280 
281     return TRUE;
282 }
283 
284 
285 /*****************************************************************************/
286 /*                              １バイトデータリード                         */
287 /*****************************************************************************/
288 
289 static BOOL Read1_IIC(BYTE byRegAddr, BYTE *pRxBuff)
290 {
291     BOOL    bResult;
292     BYTE    byTry;
293 
294     bResult = FALSE;
295     byTry = 0;
296     do{
297         /* スタートコンディション */
298         if(iic2_start(1, WRITE_MODE)){
299             /* ターゲットアドレス送信 */
300             if(iic2_TxBlock(1, &byRegAddr)){
301                 /* スタートコンディション再開 */
302                 if(iic2_start(0, READ_MODE)){
303                     /* データ受信 */
304                     if(iic2_RxBlock(1, pRxBuff)){
305                         bResult = TRUE;
306                         break;
307                     }
308                 }
309             }
310         }
311     }while(++byTry < 3);
312     /* ストップコンディション */
313     iic2_stop();
314 
315     return bResult;
316 }
317 
318 /*****************************************************************************/
319 /*                                連続データリード                           */
320 /*****************************************************************************/
321 
322 static BOOL ReadP_IIC(BYTE byRegAddr, WORD wSize, BYTE *pRxBuff)
323 {
324     BOOL    bResult;
325     BYTE    byTry;
326 
327     bResult = FALSE;
```

```
328     byTry = 0;
329     do{
330         /* スタートコンディション */
331         if(iic2_start(1, WRITE_MODE)){
332             /* ターゲットアドレス送信 */
333             if(iic2_TxBlock(1, &byRegAddr)){
334                 /* スタートコンディション再開 */
335                 if(iic2_start(0, READ_MODE)){
336                     /* データ受信 */
337                     if(iic2_RxBlock(wSize, pRxBuff)){
338                         bResult = TRUE;
339                         break;
340                     }
341                 }
342             }
343         }
344     }while(++byTry < 3);
345     /* ストップコンディション */
346     iic2_stop();
347 
348     return bResult;
349 }
350 
351 /*****************************************************************************/
352 /*                          １バイトデータライト                             */
353 /*****************************************************************************/
354 
355 static BOOL Write1_IIC(BYTE byRegAddr, BYTE *pRxBuff)
356 {
357     BOOL    bResult;
358     BYTE    byTry;
359 
360     bResult = FALSE;
361     byTry = 0;
362     do{
363         /* スタートコンディション */
364         if(iic2_start(1, WRITE_MODE)){
365             /* ターゲットアドレス送信 */
366             if(iic2_TxBlock(1, &byRegAddr)){
367                 /* データ送信 */
368                 if(iic2_TxBlock(1, pRxBuff)){
369                     bResult = TRUE;
370                     break;
371                 }
372             }
373         }
374     }while(++byTry < 3);
375     /* ストップコンディション */
376     iic2_stop();
377 
378     return bResult;
379 }
380 
381 /*****************************************************************************/
382 /*                            連続データライト                               */
383 /*****************************************************************************/
384 
385 static BOOL WriteP_IIC(BYTE byRegAddr, WORD wSize, BYTE *pRxBuff)
386 {
387     BOOL    bResult;
388     BYTE    byTry;
389 
390     bResult = FALSE;
391     byTry = 0;
392     do{
393         /* スタートコンディション */
394         if(iic2_start(1, WRITE_MODE)){
395             /* ターゲットアドレス送信 */
396             if(iic2_TxBlock(1, &byRegAddr)){
397                 /* データ送信 */
398                 if(iic2_TxBlock(wSize, pRxBuff)){
399                     bResult = TRUE;
400                     break;
401                 }
402             }
403         }
404     }while(++byTry < 3);
405     /* ストップコンディション */
406     iic2_stop();
407 
408     return bResult;
409 }
410 
```