```
/************************************************************************/
/************************************************************************/
/***                                                                  ***/
/***                    LCDモジュールSC2004Cの制御                    ***/
/***                     (LCD controller ST7066U)                     ***/
/***                                                  by:  S.Suzuki   ***/
/************************************************************************/
/************************************************************************/
/* 注意 : ・PCLK=20MHzであること。                                      */
/*      : ・RX220の64ピンパッケージであること。                         */

#include    <machine.h>
#include    <stdlib.h>
#include    <string.h>
#include    "typedefine.h"
#include    "iodefine.h"
#include    "InitSystemClock.h"         /* PCLK_HZ=20.0MHzであること */

#include    "LCD_Control.h"
#include    "CGRamMap.h"

#pragma section
#pragma bit_order right
/************************************************************************/

#define TMR_CLK_TIME    (1e9 / PCLK_HZ * 8)         /* PCLKの周期 (=400nS) */
#define LCD_EPW_TCNT    (30000 / TMR_CLK_TIME)      /* Enable Pulse Width = 30,000nS (TCNT= 75) */
#define LCD_CYC_TCNT    (50000 / TMR_CLK_TIME)      /* Enable Cycle Time  = 50,000nS (TCNT=125) */

#define LCD_RS      PORTB.PODR.BIT.B0
#define LCD_RW      PORTB.PODR.BIT.B1
#define LCD_E       PORTB.PODR.BIT.B3
#define LCD_DATA_D  PORTC.PDR.BYTE          /* '0'=入力ポート、'1'=出力ポート */
#define LCD_DATA_W  PORTC.PODR.BYTE
#define LCD_DATA_R  PORTC.PIDR.BYTE
#define LCD_BKLIGHT PORT5.PODR.BIT.B5       /* LCDのバックライト (0=ON, 1=OFF)  */
#define ATTR_BFCHK  (0x01)
#define ATTR_RAM    (0x02)

#define LCDBUFFSZ   (COLUMN_SZ * LINE_SZ)

typedef union {
    BYTE BYTE;
    struct {
        BYTE CheckBF:1;                 /* b0:0:BFを待たずに書き込み、1:BF=0を待って書き込み */
        BYTE RamData:1;                 /* b1:0:レジスタにアクセス、  1:DATA-RAMにアクセス  (未使用
)   */
        BYTE B0:1;                      /* b2:(reserved)  */
        BYTE B1:1;                      /* b3:(reserved)  */
        BYTE B2:1;                      /* b4:(reserved)  */
        BYTE B3:1;                      /* b5:(reserved)  */
        BYTE B4:1;                      /* b6:(reserved)  */
        BYTE Execute:1;                 /* b7:1=実行中  (作業用)  */
    } BIT;
} S_ATTWORK;
typedef union {
    BYTE BYTE;                          /* ATTR_BFCHK, ATTR_RAM で指示 */
    struct {
        BYTE CheckBF:1;                 /* b0:0:約40uS後に書き込み、1:BF=0を待って書き込み */
        BYTE RamData:1;                 /* b1:0:レジスタにアクセス、1:DATA-RAMにアクセス    */
        BYTE B2:1;                      /* b2:(reserved)  */
        BYTE B3:1;                      /* b3:(reserved)  */
        BYTE B4:1;                      /* b4:(reserved)  */
        BYTE B5:1;                      /* b5:(reserved)  */
        BYTE B6:1;                      /* b6:(reserved) */
        BYTE B7:1;                      /* b7:(reserved) */
    } BIT;
} S_ATTDATA;


static S_ATTDATA    st_astAttrBuff[LCDBUFFSZ];  /* LCDへのコマンド種類格納バッファ */
static BYTE         st_abyCodeBuff[LCDBUFFSZ];  /* LCDへのコマンド格納バッファ     */
static volatile WORD st_wDReadPtr;              /* st_astAttrBuff[],st_abyCodeBuff[]の読み込みポイ
ンタ */
static volatile WORD st_wDWritePtr;             /* st_astAttrBuff[],st_abyCodeBuff[]の書き込みポイ
ンタ */
static volatile S_ATTWORK   st_stAttribute;     /* st_byInstCodeのコマンド種類と実行状況を示す */
static BYTE         st_byBusyFlag;              /* b7=BF(BusyFlag) */
```

```
76 static WORD         st_wWaitCnt;                /* BusyFlagの最大待ち時間を計測（LCD_CYC_TCNT単位でカウント） */
77 static BOOL         st_bScrollOn;               /* !=0で表示をスクロールするモード */
78
79 /*****************************************************************************/
80 /*                          ディスプレイ制御初期化                           */
81 /*****************************************************************************/
82 /* 注意 : MTU0～MTU4,MTU6～MTU10のコンペアマッチ割り込みは、TCNT=TGRとなった  */
83 /*      : 次のカウントで発生するため、TGRで指定したカウント値より１カウント分 */
84 /*      : 多い時間になってしまう。従って、CMCORには希望のカウント値―１の値を */
85 /*      : 設定する。                                                         */
86 /*      : ただし、MTU5,MTU11はTGRnのタイミングで割り込みが発生する。         */
87
88 void Init_DispCtrl(void)
89 {
90     /* I/Oポート（PB,PC）の初期化 */
91     /* PB0,PB1,PB3のモード設定 */
92     PORTB.PMR.BYTE  = (BYTE)(PORTB.PMR.BYTE & 0xF4);     /* PB3,PB1,PB0をI/Oポートとして使う設定  */
93     PORTB.PODR.BYTE = (BYTE)(PORTB.PODR.BYTE & 0xF4);    /* 初期状態としてPB3=0,PB1=0,PB0=0       */
94     PORTB.PDR.BYTE  = (BYTE)(PORTB.PDR.BYTE | 0x0B);     /* PB3,PB1,PB0を出力ポートとして使う設定 */
95     /* 34ピンをPB6→PC0,33ピンをPB7→PC1に設定（64ピンRX220専用） */
96     PORTB.PMR.BYTE = (BYTE)(PORTB.PMR.BYTE & 0x3F);      /* PB7,PB6をI/Oポートとして使う設定  */
97     PORTB.PDR.BYTE = (BYTE)(PORTB.PDR.BYTE & 0x3F);      /* PB7,PB6を入力ポートとして使う設定 */
98     PORTB.PCR.BYTE = (BYTE)(PORTB.PCR.BYTE & 0x3F);      /* PB7,PB6の入力プルアップ抵抗を無効にする設定 */
99     PORT.PSRA.BIT.PSEL6 = 1;             /* ポート切り替えレジスタAで34ピン（PB6）をPC0とする */
100    PORT.PSRA.BIT.PSEL7 = 1;             /* ポート切り替えレジスタAで33ピン（PB7）をPC1とする */
101    /* PC7～PC0のモード設定 */
102    PORTC.PMR.BYTE  = 0x00;              /* PC7～PC0をI/Oポートとして使う         */
103    PORTC.PODR.BYTE = 0x00;              /* PC7～PC0の初期出力状態を全て'0'とする */
104    PORTC.PDR.BYTE  = 0x00;              /* PC7～PC0を入力ポートとして使う設定    */
105    /* P55のモード設定（バックライト制御用） */
106    PORT5.PMR.BIT.B5  = 0;               /* P55をI/Oポートとして使う      */
107    PORT5.PODR.BIT.B5 = 0;               /* P55の初期出力状態を'0'とする  */
108    PORT5.PDR.BIT.B5  = 1;               /* P55を出力ポートとして使う設定 */
109
110    /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書き込みを許可する */
111    SYSTEM.PRCR.WORD = 0xA502;           /* プロテクトレジスタ（PRCR）    */
112    /* TMR2,TMR3のモジュールストップ解除 */
113    MSTP(TMR23) = 0;                     /* SYSTEM.MSTPCRA.BIT.MSTPA4 = 0;（TMR2, TMR3共通） */
114    /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書き込みを禁止する */
115    SYSTEM.PRCR.WORD = 0xA500;           /* プロテクトレジスタ（PRCR）    */
116
117    /* ８ビットタイマ(TMR2)初期化 */
118    TMR2.TCR.BYTE = 0xC8;                /* コンペアマッチAによりカウンタクリア、コンペアマッチA,Bによる割り込み要求を許可 */
119    TMR2.TCSR.BYTE = 0x00;               /* TMO2端子出力は変化しない      */
120    TMR2.TCCR.BYTE = 0x0A;               /* カウンタクロック＝PCLK/8をセット */
121    TMR2.TCSTR.BYTE = 0x00;              /* ELCによるカウント停止         */
122    TMR2.TCORA = LCD_CYC_TCNT -1;        /* Enable Cycle Timeをセット      */
123    TMR2.TCORB = LCD_EPW_TCNT -1;        /* Enable Pulse Widthをセット     */
124    TMR2.TCNT  = 0;                      /* タイマカウンタクリア           */
125
126    /* TMR2の割り込み優先順位セット */
127    IPR(TMR2, ) = 5;                     /* CMIA2,CMIB2,OVI2は共通（レベル５は例） */
128    /* TMR2の割り込み要求ステータスフラグクリア */
129    IR(TMR2, CMIA2) = 0;
130    IR(TMR2, CMIB2) = 0;
131    IR(TMR2,  OVI2) = 0;
132    /* TMR2のオーバフロー割り込み禁止 */
133    IEN(TMR2,  OVI2) = 0;
134    /* コンペアマッチ割り込み許可 */
135    IEN(TMR2, CMIA2) = 1;                /* コンペアマッチＡ */
136    IEN(TMR2, CMIB2) = 1;                /* コンペアマッチＢ */
137
138    /* 表示データポインタ初期化 */
139    st_wDReadPtr  = 0;
140    st_wDWritePtr = 0;
141    st_byBusyFlag = 0xFF;
142    /* LCD制御OFF */
143    LCD_RS = 0;                          /*  RS = '0' */
144    LCD_RW = 1;                          /* R/W = '1'(Read) */
145    LCD_E  = 0;                          /*   E = '0' */
146
147 }
```

```
148
149
150 /*****************************************************************************/
151 /*                    データ読み込み／書き込み処理の補助関数                 */
152 /*****************************************************************************/
153
154 static void Wr_Cycle_Data(void)
155 {
156     /* Function書き込み準備 */
157     /* RegisterSelectを指定する */
158     if(st_astAttrBuff[st_wDReadPtr].BIT.RamData == 0)
159         LCD_RS = 0;             /*  RS = '0' */
160     else
161         LCD_RS = 1;             /*  RS = '1' */
162     /* Read/Writeをライトに */
163     LCD_RW = 0;                 /* R/W = '0' (Write) */
164     /* EnableをHighにする */
165     LCD_E  = 1;                 /*  E  = '1' */
166     /* LCDデータポート (PC7～PC0) に書き込みデータを出力する */
167     LCD_DATA_W = st_abyCodeBuff[st_wDReadPtr];
168     /* 読み込みポインタを進める */
169     if(++st_wDReadPtr >= LCDBUFFSZ){
170         st_wDReadPtr = 0;
171     }
172     /* LCDデータポート (PC7～PC0) を出力ポートにする */
173     LCD_DATA_D = 0xFF;
174 }
175
176 static void Rd_Cycle_Busy(void)
177 {
178     st_byBusyFlag = 0xFF;
179     /* LCDデータポート (PC7～PC0) を入力ポートに */
180     LCD_DATA_D = 0x00;
181     /* RegisterSelectをLow(=Function)にする */
182     LCD_RS = 0;                 /*  RS = '0' */
183     /* Read/Writeをリードに */
184     LCD_RW = 1;                 /* R/W = '1' (Read) */
185     /* EnableをHighにする */
186     LCD_E  = 1;                 /*  E  = '1' */
187 }
188
189
190 /*****************************************************************************/
191 /*           TMR2タイマ割り込みによるデータ読み込み／書き込み処理           */
192 /*****************************************************************************/
193 /* ※CMIA,CMIB,OVI2はエッジ検知割り込みのため、IRフラグは自動的にクリアされる。 */
194 /* ※タイマカウンタはTCORAのコンペアマッチでクリア                             */
195
196 /* コンペアマッチＡ割り込み  ***********************************************/
197 /* ※LCD_CYC_TCNT (100μS) 周期で発生。                                        */
198
199 #pragma interrupt (Int_TMR2_CMIA2)
200 void Int_TMR2_CMIA2(void)
201 {
202     /* 書き込みデータがない場合 */
203     if(st_wDReadPtr == st_wDWritePtr){
204         if(st_stAttribute.BIT.Execute != 0){
205             /* Read/Writeサイクル停止中を示す */
206             st_stAttribute.BIT.Execute = 0;
207             /* LCDデータポート (PC7～PC0) を入力ポートに */
208             LCD_DATA_D = 0x00;
209             /* RegisterSelectをLow(=Function)にする */
210             LCD_RS = 0;                 /*  RS = '0' */
211             /* Read/Writeをリードに */
212             LCD_RW = 1;                 /* R/W = '1'(Read mode)  */
213             /* EnableをLowにする */
214             LCD_E  = 0;                 /*   E = '0' */
215         }
216         return;
217     }
218     /* 書き込みデータがある場合 */
219     else{
220         st_stAttribute.BIT.Execute = 1;
221         /* BusyFlag読み込みを実行中だった場合 */
222         if(st_stAttribute.BIT.CheckBF != 0){
223             /* BusyFlag=1だった場合 */
224             if((st_byBusyFlag & 0x80) != 0){
225                 /* BusyFlag読み込みをカウント */
```

```
226                 if(++st_wWaitCnt < 100){        /* 10mSまで待つ（100μS×100=10mS） */
227                     /* BusyFlag再読み込み */
228                     Rd_Cycle_Busy();
229                     return;
230                 }
231             }   /* 注：Busyがカウントオーバーの場合は書き込みを強行する */
232             /* BusyFlag=0だった場合 */
233             st_stAttribute.BIT.CheckBF = 0;
234             /* 書き込みデータを出力し、読み込みポインタを進める */
235             Wr_Cycle_Data();
236         }
237         else{
238             /* 次はBusyFlagのチェックかデータ書き込みかメモする */
239             st_stAttribute.BIT.RamData = st_astAttrBuff[st_wDReadPtr].BIT.RamData;
240             st_stAttribute.BIT.CheckBF = st_astAttrBuff[st_wDReadPtr].BIT.CheckBF;
241             /* BusyFlagをチェックする場合（読み込み） */
242             if(st_stAttribute.BIT.CheckBF != 0){
243                 /* BusyFlag読み込みカウンタリセット */
244                 st_wWaitCnt = 0;
245                 /* BusyFlag読み込み */
246                 Rd_Cycle_Busy();
247             }
248             /* BusyFlagをチェックしない場合（書き込みのみ） */
249             else{
250                 /* 書き込みデータを出力し、読み込みポインタを進める */
251                 Wr_Cycle_Data();
252             }
253         }
254     }
255 }
256
257 /* コンペアマッチＢ割り込み  ************************************************/
258 /* ※コンペアマッチＡ割り込みの、LCD_EPW_TCNT（40μS）後に発生。              */
259
260 #pragma interrupt (Int_TMR2_CMIB2)
261 void Int_TMR2_CMIB2(void)
262 {
263     /* BusyFlagをチェックする場合（読み込み） */
264     if(st_stAttribute.BIT.CheckBF != 0){
265         /* LCDデータポート（PC7～PC0）からデータを読み込む */
266         st_byBusyFlag = LCD_DATA_R;
267     }
268     /* EnableをLowにする */
269     LCD_E  = 0;                 /*  E  = '0' */
270 }
271
272
273
274 /*****************************************************************************/
275 /*                              周期割り込み処理                             */
276 /*****************************************************************************/
277 /* 1mS周期(CYCLE_1M)で実行                                                   */
278 static volatile WORD wTmrDwnCnt1_1mS;
279 static volatile WORD wTmrDwnCnt2_1mS;
280
281 void Timer01mS_LCDC(void)
282 {
283     if(wTmrDwnCnt1_1mS != 0){
284         --wTmrDwnCnt1_1mS;
285     }
286     if(wTmrDwnCnt2_1mS != 0){
287         --wTmrDwnCnt2_1mS;
288     }
289 }
290
291 /*****************************************************************************/
292 /* １秒周期で実行                                                            */
293 static volatile WORD wTmrDwnCnt_1sec;
294
295 void Timer1sec_LCDC(void)
296 {
297
298     if(wTmrDwnCnt_1sec != 0){
299         --wTmrDwnCnt_1sec;
300         if(wTmrDwnCnt_1sec == 0){
301             if(st_SetOption.BIT.BackLight != 0){
302                 LCD_BKLIGHT = _OFF;
303             }
```

```
304         }
305     }
306 }
307
308 /*****************************************************************************/
309 /*****************************************************************************/
310 /*                                                                           */
311 /*                                LCD表示関数                                */
312 /*                                                                           */
313 /*****************************************************************************/
314 /*****************************************************************************/
315
316 static const BYTE st_abyLocTable[LINE_SZ]={0x00, 0x40, 0x14, 0x54}; /* 各行の先頭DATA-RAMアドレス *
    /
317
318 static char st_abyFrameBuff[LINE_SZ][COLUMN_SZ];
319 static int  st_nCurPosX,st_nCurPosY;
320
321 /*****************************************************************************/
322 /*                      LCDのバックライトを点灯／消灯                        */
323 /*****************************************************************************/
324
325 void ON_BackLight(void)
326 {
327     wTmrDwnCnt_1sec = BKLIGHT_OFFTIME;
328     LCD_BKLIGHT = _ON;
329 }
330
331 BOOL Check_BackLight(void)
332 {
333     if(LCD_BKLIGHT == _OFF){
334         return FALSE;
335     }
336     return TRUE;
337 }
338
339 /*****************************************************************************/
340 /*                              指定時間待つ                                 */
341 /*****************************************************************************/
342 /* 注意 : 指定時間が経過するまで戻らない、ただし割り込みは受け付ける。       */
343 /* 引数 : wTimeMsec = 待ち時間 (CYCLE_1M単位)                                */
344 /* 戻値 : なし                                                               */
345
346 static void Wait_Simple(WORD wTimeMsec)
347 {
348     wTmrDwnCnt1_1mS = wTimeMsec;
349     while(wTmrDwnCnt1_1mS != 0);
350 }
351
352 /*****************************************************************************/
353 /*                          LCDに１バイト書き込み                            */
354 /*****************************************************************************/
355 /* 注意 : 書き込みバッファがフルの場合は何もしないでFALSEで戻る。            */
356 /* 引数 :   byRS = ATTR_BFCHK | ATTR_RAMの指定、指定なしでレジスタに書き込み  */
357 /*      :           ・ATTR_BFCHKの指定でBusyFlagチェック有り                 */
358 /*      :           ・ATTR_RAMの指定でRAM-DATAに書き込み                     */
359 /*      : byData = レジスタまたはDATA-RAMに書き込むデータ                     */
360 /* 戻値 : WriteData1() ==  TRUE : 正常                                       */
361 /*      :                == FALSE : 異常 (書き込みバッファフル)              */
362
363 static BOOL WriteData1(BYTE byRS, BYTE byData)
364 {
365     WORD        wWPtr;
366
367     wWPtr = (WORD)(st_wDWritePtr + 1);
368     if(wWPtr >= LCDBUFFSZ){
369         wWPtr = 0;
370     }
371     wTmrDwnCnt2_1mS = 100;
372     while(wWPtr == st_wDReadPtr){
373         if(wTmrDwnCnt2_1mS == 0){
374             return FALSE;
375         }
376     }
377     st_astAttrBuff[st_wDWritePtr].BYTE = (BYTE)(byRS & (ATTR_BFCHK | ATTR_RAM));
378     st_abyCodeBuff[st_wDWritePtr] = byData;
379     st_wDWritePtr = wWPtr;
380
```

```
381     return TRUE;
382 }
383 
384 /*****************************************************************************/
385 /*                      LCDにユーザー定義文字をセットする                    */
386 /*****************************************************************************/
387 /* 注意 : ST7066Uでは、コード0x00～0x07の８文字がユーザー定義可能。          */
388 /* 引数 : なし                                                               */
389 /* 戻値 : Set_UserCG() ==  TRUE : 正常                                       */
390 /*      :              == FALSE : 異常（書き込みバッファフル）               */
391 
392 static BOOL Set_UserCG(void)
393 {
394     int     nCode,nLine;
395     BYTE    byAddr;
396 
397     byAddr = 0;
398     nCode  = 0;
399     do{
400         /* Set CGRAM Address */
401         if(WriteData1(ATTR_BFCHK, (BYTE)(byAddr | 0x40)) == FALSE){
402             return FALSE;
403         }
404         nLine = 0;
405         do{
406             /* Write CG-DATA */
407             if(WriteData1(ATTR_BFCHK | ATTR_RAM, UserCGpatten[nCode][nLine]) == FALSE){
408                 return FALSE;
409             }
410         }while(++nLine < 8);
411         byAddr += 8;
412     }while(++nCode < 8);
413 
414     return TRUE;
415 }
416 
417 /*****************************************************************************/
418 /*                            ディスプレイ初期化                             */
419 /*****************************************************************************/
420 /* 注意 : ・割り込み許可状態で実行すること。                                 */
421 /*      : ・表示バッファ書き込みに関するエラーは無視される。                 */
422 /*      : ・スクロールなしにセットされる。                                   */
423 /* 引数 : なし                                                               */
424 /* 戻値 : なし                                                               */
425 
426 void Init_Display(void)
427 {
428     static const BYTE LcdCtrlParam[]={
429         0x38,       /* Function set           (DL=1,N=0,F=0 : ８ビットバス、１行表示モード、５×８
    ドットフォント)  */
430         0x38,       /* Function set           (DL=1,N=0,F=0 : )  */
431         0x38,       /* Function set           (DL=1,N=0,F=0 : )  */
432         0x0C,       /* Display ON/OFF control (D=1,C=0,B=0  : ディスプレイオン、カーソル表示なし、
    カーソル点滅オフ)   */
433     };
434     const BYTE* pbyParams;
435     int     nLpcnt;
436 
437     /* LCDコントローラ初期化 */
438     pbyParams = LcdCtrlParam;
439     nLpcnt = 0;
440     do{
441         /* LCDにコマンド送信 */
442         WriteData1(0, *pbyParams);          /* Function set */
443         pbyParams++;
444         /* 5mS待つ */
445         Wait_Simple(5);
446     }while(++nLpcnt < 2);
447     do{
448         /* LCDにコマンド送信 */
449         WriteData1(ATTR_BFCHK, *pbyParams); /* Display Control set */
450         pbyParams++;
451         /* 5mS待つ */
452         Wait_Simple(5);
453     }while(++nLpcnt < 4);
454 
455     /* LCDにコマンド送信 */
456     WriteData1(ATTR_BFCHK, 0x01);       /* Display clear */
```

```
457     Wait_Simple(5);                     /* 5mS待つ (DisplayClearは最大1.52mSかかるため) */
458     /* LCDにコマンド送信 */
459     WriteData1(ATTR_BFCHK, 0x06);       /* Entry mode set (I/D=1,S=0 : カーソルは右移動、DDRAMアドレスは＋１、ディスプレイシフト無し) */
460
461     /* ユーザー定義文字をセット */
462     Set_UserCG();
463
464     /* フレームメモリ消去 */
465     st_nCurPosY = LINE_SZ;
466     do{
467         --st_nCurPosY;
468         st_nCurPosX = COLUMN_SZ;
469         do{
470             --st_nCurPosX;
471             st_abyFrameBuff[st_nCurPosY][st_nCurPosX] = ' ';
472         }while(st_nCurPosX != 0);
473     }while(st_nCurPosY != 0);
474
475     /* スクロールしないをセット */
476     st_bScrollOn = FALSE;
477     /* LCDバックライトをONにする */
478     ON_BackLight();
479 }
480
481
482 /*****************************************************************************/
483 /*                          スクロールの有無を設定                           */
484 /*****************************************************************************/
485 /* 注意 : デフォルトではスクロールなしになっている。                         */
486 /* 引数 : bScrollSW = スクロール設定 (FALSE:なし、TRUE:あり)                 */
487 /* 戻値 : なし                                                               */
488
489 void Set_DispScroll(BOOL bScrollSW)
490 {
491     /* スクロールスイッチセット */
492     st_bScrollOn = bScrollSW;
493 }
494
495
496 /*****************************************************************************/
497 /*                              ディスプレイ消去                             */
498 /*****************************************************************************/
499 /* 注意 : 表示位置(st_nCurPosX,st_nCurPosY)はそれぞれ０になる。              */
500 /* 引数 : なし                                                               */
501 /* 戻値 : Clear_Display() ==  TRUE : 正常                                    */
502 /*      :                 == FALSE : 異常 (書き込みバッファフル)             */
503
504 BOOL Clear_Display(void)
505 {
506     /* Display clear */
507     if(WriteData1(ATTR_BFCHK, 0x01) == FALSE){
508         return FALSE;
509     }
510     /* 5mSタイマセット */
511     wTmrDwnCnt1_1mS = 5;
512     /* フレームメモリ消去 */
513     st_nCurPosY = LINE_SZ;
514     do{
515         --st_nCurPosY;
516         st_nCurPosX = COLUMN_SZ;
517         do{
518             --st_nCurPosX;
519             st_abyFrameBuff[st_nCurPosY][st_nCurPosX] = ' ';
520         }while(st_nCurPosX != 0);
521     }while(st_nCurPosY != 0);
522     /* 最大5mS待つ (DisplayClearは最大1.52mSかかるため) */
523     while(wTmrDwnCnt1_1mS != 0);
524
525     return TRUE;
526 }
527
528 /*****************************************************************************/
529 /*                    現在の表示位置から右側をクリアする                     */
530 /*****************************************************************************/
531 /* 機能 : 現在表示行の、現在の桁位置から右側を空白で埋める。                 */
532 /* 引数 : なし                                                               */
533 /* 戻値 : なし                                                               */
```

```
534
535 BOOL Clear_RightSide(void)
536 {
537     int     nPosX;
538
539     nPosX = st_nCurPosX;
540     while(nPosX < COLUMN_SZ){
541         if(Disp_Character(' ') == FALSE){
542             return FALSE;
543         }
544         nPosX++;
545     }
546
547     return TRUE;
548 }
549
550 /*****************************************************************************/
551 /*                      スクロールアップ（表示シフトアップ）                 */
552 /*****************************************************************************/
553 /* 注意 : 表示位置(st_nCurPosX,st_nCurPosY)は変更されない。                  */
554 /* 引数 : なし                                                               */
555 /* 戻値 : Scroll_Up() ==  TRUE : 正常                                        */
556 /*      :             == FALSE : 異常（書き込みバッファフル）               */
557
558 BOOL Scroll_Up(void)
559 {
560     int     nPosX,nPosY;
561     BYTE    byAddr;
562
563     /* フレームバッファst_abyFrameBuff[]の内容をシフトアップ */
564     for(nPosY=0;nPosY < LINE_SZ-1;nPosY++){
565         for(nPosX=0;nPosX < COLUMN_SZ;nPosX++){
566             st_abyFrameBuff[nPosY][nPosX] = st_abyFrameBuff[nPosY+1][nPosX];
567         }
568     }
569     /* フレームバッファの最下行クリア */
570     for(nPosX=0;nPosX < COLUMN_SZ;nPosX++){
571         st_abyFrameBuff[nPosY][nPosX] = ' ';
572     }
573     /* フレームバッファを全画面再表示 */
574     for(nPosY=0;nPosY < LINE_SZ;nPosY++){
575         /* DATA-RAMのアドレスを指定 */
576         byAddr = (BYTE)(st_abyLocTable[nPosY] + 0x80);
577         if(WriteData1(ATTR_BFCHK, byAddr) == FALSE){
578             return FALSE;
579         }
580         /* フレームバッファの内容をDATA-RAMに転送して表示 */
581         for(nPosX=0;nPosX < COLUMN_SZ;nPosX++){
582             if(WriteData1(ATTR_BFCHK | ATTR_RAM, (BYTE)st_abyFrameBuff[nPosY][nPosX]) == FALSE){
583                 return FALSE;
584             }
585         }
586     }
587     /* 表示位置（DATA-RAMのアドレス）を元に戻す */
588 //  byAddr = (BYTE)(st_nCurPosX + st_abyLocTable[st_nCurPosY] + 0x80);
589 //  WriteData1(ATTR_BFCHK, byAddr);
590
591     return TRUE;
592 }
593
594
595 /*****************************************************************************/
596 /*                      スクロールダウン（表示シフトダウン）                 */
597 /*****************************************************************************/
598 /* 注意 : 表示位置(st_nCurPosX,st_nCurPosY)は変更されない。                  */
599 /* 引数 : なし                                                               */
600 /* 戻値 : Scroll_Down() ==  TRUE : 正常                                      */
601 /*      :               == FALSE : 異常（書き込みバッファフル）              */
602
603 BOOL Scroll_Down(void)
604 {
605     int     nPosX,nPosY;
606     BYTE    byAddr;
607
608     /* フレームバッファst_abyFrameBuff[]の内容をシフトダウン */
609     for(nPosY=LINE_SZ-1;nPosY > 0;nPosY--){
610         for(nPosX=0;nPosX < COLUMN_SZ;nPosX++){
611             st_abyFrameBuff[nPosY][nPosX] = st_abyFrameBuff[nPosY-1][nPosX];
```

```
612         }
613     }
614     /* フレームバッファの先頭行クリア */
615     for(nPosX=0;nPosX < COLUMN_SZ;nPosX++){
616         st_abyFrameBuff[nPosY][nPosX] = ' ';
617     }
618     /* フレームバッファを全画面再表示 */
619     for(nPosY=0;nPosY < LINE_SZ;nPosY++){
620         /* DATA-RAMのアドレスを指定 */
621         byAddr = (BYTE)(st_abyLocTable[nPosY] + 0x80);
622         if(WriteData1(ATTR_BFCHK, byAddr) == FALSE){
623             return FALSE;
624         }
625         /* フレームバッファの内容をDATA-RAMに転送して表示 */
626         for(nPosX=0;nPosX < COLUMN_SZ;nPosX++){
627             if(WriteData1(ATTR_BFCHK | ATTR_RAM, (BYTE)st_abyFrameBuff[nPosY][nPosX]) == FALSE){
628                 return FALSE;
629             }
630         }
631     }
632     /* 表示位置（DATA-RAMのアドレス）を元に戻す */
633 //  byAddr = (BYTE)(st_nCurPosX + st_abyLocTable[st_nCurPosY] + 0x80);
634 //  WriteData1(ATTR_BFCHK, byAddr);
635
636     return TRUE;
637 }
638
639
640 /*****************************************************************************/
641 /*                                １文字表示                                 */
642 /*****************************************************************************/
643 /* 注意 : ・フォントがないコードの選別は省略している、使用側で注意。         */
644 /*      : ・ユーザー定義文字のCG_0は0x00=NUL（終端）と同じコードのためプログ  */
645 /*      :   ラム中ではCG_REVS=0x0B)として扱い、ここで0x00に変換する。        */
646 /* 機能 : (st_nCurPosY)(st_nCurPosX)が示す位置に表示する。                    */
647 /*      : 画面がいっぱいになるとスクロールする。                              */
648 /* 引数 : chChar = 表示文字（ASCII、ユーザー定義文字CG_??が使用可能）         */
649 /*      :           ※ユーザ定義文字はヘッダファイル LCD_Controlh を参照      */
650 /* 戻値 : Disp_Character() == TRUE : 正常                                     */
651 /*      :                       == FALSE : 異常（書き込みバッファフル）       */
652 /*      : (st_nCurPosY)(st_nCurPosX) : 新しい表示位置                         */
653
654 BOOL Disp_Character(char chChar)
655 {
656     BYTE    byAddr;
657
658     switch(chChar){
659     case _CR:
660         st_nCurPosX = 0;
661         break;
662     case _LF:
663         st_nCurPosY++;
664         if(st_nCurPosY >= LINE_SZ){
665             /* スクロールしない場合 */
666             if(!st_bScrollOn)
667                 break;
668             st_nCurPosY = LINE_SZ-1;
669             /* シフトアップ */
670             if(Scroll_Up() == FALSE){
671                 return FALSE;
672             }
673         }
674         /* DATA-RAMのアドレスを指定（b7='1'はDDRAMのアドレスセット） */
675         byAddr = (BYTE)(st_nCurPosX + st_abyLocTable[st_nCurPosY] + 0x80);
676         if(WriteData1(ATTR_BFCHK, byAddr) == FALSE){
677             return FALSE;
678         }
679         break;
680     case _BS:
681         /* 表示位置を１行上に戻す */
682         if(st_nCurPosX == 0){
683             --st_nCurPosY;
684             st_nCurPosX = COLUMN_SZ;
685         }
686         /* 表示位置を１つ左に戻す */
687         --st_nCurPosX;
688         /* 表示領域外では何もしない */
689         if((st_nCurPosX < 0) || (COLUMN_SZ <= st_nCurPosX))
```

```
690             break;
691         if((st_nCurPosY < 0) || (LINE_SZ   <= st_nCurPosY))
692             break;
693         /* 空白をフレームバッファに格納 */
694         st_abyFrameBuff[st_nCurPosY][st_nCurPosX] = ' ';
695         /* DATA-RAMのアドレスを指定 (b7='1'はDDRAMのアドレスセット)  */
696         byAddr = (BYTE)(st_nCurPosX + st_abyLocTable[st_nCurPosY] + 0x80);
697         if(WriteData1(ATTR_BFCHK, byAddr) == FALSE){
698             return FALSE;
699         }
700         /* DATA-RAMに転送して表示 */
701         if(WriteData1(ATTR_BFCHK | ATTR_RAM, ' ') == FALSE){
702             return FALSE;
703         }
704         /* DATA-RAMのアドレスを再度指定 (b7='1'はDDRAMのアドレスセット)  */
705         byAddr = (BYTE)(st_nCurPosX + st_abyLocTable[st_nCurPosY] + 0x80);
706         if(WriteData1(ATTR_BFCHK, byAddr) == FALSE){
707             return FALSE;
708         }
709         break;
710     case CG_REVS:                   /* CG_REVS(=0x0B)はCG_0(=0x00)に変換する */
711         chChar = 0;
712     default:                        /* ※フォントがないコードの選別は省略する、使用側で注意。 */
713         /* 表示領域外では何もしない */
714         if((st_nCurPosX < 0) || (COLUMN_SZ <= st_nCurPosX))
715             break;
716         if((st_nCurPosY < 0) || (LINE_SZ   <= st_nCurPosY))
717             break;
718         /* 表示文字をフレームバッファに格納 */
719         st_abyFrameBuff[st_nCurPosY][st_nCurPosX] = chChar;
720         /* DATA-RAMに転送して表示 */
721         if(WriteData1(ATTR_BFCHK | ATTR_RAM, (BYTE)chChar) == FALSE){
722             return FALSE;
723         }
724 
725         /* 表示位置（カーソル位置）更新 */
726         st_nCurPosX++;
727         if(st_nCurPosX >= COLUMN_SZ){
728             st_nCurPosX = 0;
729             st_nCurPosY++;
730             if(st_nCurPosY >= LINE_SZ){
731                 /* スクロールしない場合 */
732                 if(!st_bScrollOn)
733                     break;
734                 st_nCurPosY = LINE_SZ-1;
735                 /* シフトアップ */
736                 if(Scroll_Up() == FALSE){
737                     return FALSE;
738                 }
739             }
740             /* DATA-RAMのアドレスを指定 */
741             byAddr = (BYTE)(st_abyLocTable[st_nCurPosY] + 0x80);
742             if(WriteData1(ATTR_BFCHK, byAddr) == FALSE){
743                 return FALSE;
744             }
745         }
746         break;
747     }
748 
749     return TRUE;
750 }
751 
752 
753 /*****************************************************************************/
754 /*                        表示位置（カーソル位置）セット                     */
755 /*****************************************************************************/
756 /* 注意 : 表示位置が範囲外の場合は何もしない。                               */
757 /* 引数 : nPosX = 横（桁）位置（0～COLUMN_SZ-1)                              */
758 /*      : nPosY = 縦（行）位置（0～LINE_SZ-1)                                */
759 /* 戻値 : Set_DispPos() ==  TRUE : 正常                                      */
760 /*      :                == FALSE : 異常（書き込みバッファフル）            */
761 /*      : (st_nCurPosY)(st_nCurPosX) : 新しい表示位置                        */
762 
763 BOOL Set_DispPos(int nPosX, int nPosY)
764 {
765     BYTE    byAddr;
766 
767     /* 格納 */
```

```
768     st_nCurPosX = nPosX;
769     st_nCurPosY = nPosY;
770     /* 表示領域外では何もしない */
771     if((nPosX < 0) || (COLUMN_SZ <= nPosX))
772         return TRUE;
773     if((nPosY < 0) || (LINE_SZ   <= nPosY))
774         return TRUE;
775     /* DATA-RAMのアドレスを指定（b7='1'はDDRAMのアドレスセット） */
776     byAddr = (BYTE)(nPosX + st_abyLocTable[nPosY] + 0x80);
777     return WriteData1(ATTR_BFCHK, byAddr);
778 }
779
780 /*****************************************************************************/
781 /*                    表示位置（カーソル位置）読み込み                       */
782 /*****************************************************************************/
783 /* 引数 : pnPosX = 横（桁）位置格納アドレス                                  */
784 /*      : pnPosY = 縦（行）位置格納アドレス                                  */
785 /*      : (st_nCurPosY)(st_nCurPosX) : 現在の表示位置                        */
786 /* 戻値 : なし                                                               */
787
788 void Get_DispPos(int* pnPosX, int* pnPosY)
789 {
790     /* 格納 */
791     *pnPosX = st_nCurPosX;
792     *pnPosY = st_nCurPosY;
793 }
794
795 /*****************************************************************************/
796 /*                          表示位置指定で文字列表示                         */
797 /*****************************************************************************/
798 /* 注意 : nPosX<0またはnPosY<0のときは現在の表示位置から表示。               */
799 /* 引数 :  nPosX = 横（桁）表示位置（0～COLUMN_SZ-1）                        */
800 /*      :  nPosY = 縦（行）表示位置（0～LINE_SZ-1）                          */
801 /*      : pchStr = 表示文字列の先頭アドレス（0x00で終端する文字列）          */
802 /* 戻値 : Disp_String() ==  TRUE : 正常                                      */
803 /*      :                == FALSE : 異常（書き込みバッファフル）           */
804 /*      : (st_nCurPosY)(st_nCurPosX) : 新しい表示位置                        */
805
806 BOOL Disp_String(int nPosX, int nPosY, const char* pchStr)
807 {
808     char    chChar;
809
810     /* 表示位置指定有効のとき */
811     if((nPosX >= 0) && (nPosY >= 0)){
812         Set_DispPos(nPosX, nPosY);
813     }
814     /* １文字ずつ順に表示 */
815     while((chChar = *(pchStr++)) != 0){
816         if(Disp_Character(chChar) == FALSE){
817             return FALSE;
818         }
819     }
820
821     return TRUE;
822 }
823
824
825 /*****************************************************************************/
826 /*                  表示位置指定で文字列表示（文字数指定）                   */
827 /*****************************************************************************/
828 /* 注意 : nPosX<0またはnPosY<0のときは現在の表示位置から表示。               */
829 /* 引数 :  nPosX = 横（桁）表示位置（0～COLUMN_SZ-1）                        */
830 /*      :  nPosY = 縦（行）表示位置（0～LINE_SZ-1）                          */
831 /*      :  nSize = 表示文字数（終端0x00があれば終端前まで表示）              */
832 /*      : pchStr = 表示文字列の先頭アドレス                                  */
833 /* 戻値 : Disp_BlockChar() ==  TRUE : 正常                                   */
834 /*      :                  == FALSE : 異常（書き込みバッファフル）         */
835 /*      : (st_nCurPosY)(st_nCurPosX) : 新しい表示位置                        */
836
837 BOOL Disp_BlockChar(int nPosX, int nPosY, int nSize, const char* pchBlock)
838 {
839     char    chChar;
840
841     /* 表示位置指定有効のとき */
842     if((nPosX >= 0) && (nPosY >= 0)){
843         Set_DispPos(nPosX, nPosY);
844     }
845     /* １文字ずつ順に表示 */
```

```
846     while(nSize-- != 0){
847         chChar = *(pchBlock++);
848         if(chChar == 0x00)  break;
849         if(Disp_Character(chChar) == FALSE){
850             return FALSE;
851         }
852     }
853 
854     return TRUE;
855 }
856 
857 /*****************************************************************************/
858 /*                        表示位置指定で指定数の空白を表示                   */
859 /*****************************************************************************/
860 /* 注意 : nPosX<0またはnPosY<0のときは現在の表示位置から表示。                */
861 /* 引数 :    nPosX = 横（桁）表示位置（0～COLUMN_SZ-1）                      */
862 /*      :    nPosY = 縦（行）表示位置（0～LINE_SZ-1）                        */
863 /*      : nSpcSize = 空白数                                                  */
864 /* 戻値 : Disp_Space() ==  TRUE : 正常                                       */
865 /*      :                  == FALSE : 異常（書き込みバッファフル）          */
866 /*      : ※LCDコントローラ内に次の新しい表示位置が保持される。             */
867 
868 BOOL Disp_Space(int nPosX, int nPosY, int nSpcSize)
869 {
870     /* 表示位置指定有効のとき */
871     if((nPosX >= 0) && (nPosY >= 0)){
872         Set_DispPos(nPosX, nPosY);
873     }
874     /* １文字ずつ順に表示 */
875     while(nSpcSize-- != 0){
876         if(Disp_Character(' ') == FALSE){
877             return FALSE;
878         }
879     }
880 
881     return TRUE;
882 }
883 
884 
885 /* End of File */
```