```
/*****************************************************************************/
/*                          LEDのダイナミック点灯                            */
/*****************************************************************************/

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

注：７セグメントLEDの表示パターンSEGPTN_xxと、記号表示LEDの表示パターンLED_xxxxの
    宣言は省略してあります。

#pragma section
/* ７セグメント表示パターン ******************************************************/
/*                                                                           */
/*    _a_          .gfedcba                                                  */
/*  f|   |b      [76543210]                                                  */
/*   |_g_|        00111111 : 0                                               */
/*  e|   |c       00000110 : 1                                               */
/*   |_d_|        01011011 : 2                                               */
/*                01001111 : 3                                               */
/*                01100110 : 4                                               */
/*                01101101 : 5                                               */
/*                01111101 : 6                                               */
/*                00000111 : 7                                               */
/*                01111111 : 8                                               */
/*                01101111 : 9                                               */
/*                01011111 : a                                               */
/*                01111100 : b                                               */
/*                01011000 : c                                               */
/*                01011110 : d                                               */
/*                01111001 : e                                               */
/*                01110001 : f                                               */
/*                01111000 : t                                               */
/*                01101101 : S                                               */
/*                01010100 : n                                               */

/* ７セグメント表示バッファ ******************************************************/
#define DIGI_MAX (5)
typedef union {
    BYTE    BYTE;
    struct {
        BYTE    COLH:1;         /* b7:コロン（上）  */
        BYTE    COLL:1;         /* b6:コロン（下）  */
        BYTE    AM:1;           /* b5:AM            */
        BYTE    PM:1;           /* b4:PM            */
        BYTE    ALM:1;          /* b3:Alarm         */
        BYTE    TEMP:1;         /* b2:Temperature   */
        BYTE    ALMON:1;        /* b1:Alarm ON      */
        BYTE    YOBI:1;         /* b0:（予備）      */
    } BIT;
} COLON;
typedef union {
    BYTE    buff[DIGI_MAX];
    struct {
        BYTE    M01;            /* D0:分 1位 */
        BYTE    M10;            /* D1:分10位 */
        BYTE    H01;            /* D2:時 1位 */
        BYTE    H10;            /* D3:時10位 */
        COLON   COL;            /* D4:コロン・項目 */
    } Digi;
} FRMBUFF;

static FRMBUFF Segment;

/* 16進数→７セグパターン変換表  ***********************************************/
static const BYTE HexDecimalPtn[16]={
    SEGPTN_0,
    SEGPTN_1,
    SEGPTN_2,
    SEGPTN_3,
    SEGPTN_4,
    SEGPTN_5,
    SEGPTN_6,
    SEGPTN_7,
    SEGPTN_8,
    SEGPTN_9,
    SEGPTN_A,
    SEGPTN_b,
    SEGPTN_C,
    SEGPTN_d,
    SEGPTN_E,
    SEGPTN_F,
};

```

```
 85 /* バイナリ→10進数変換表 ***************************************************/
 86 static const WORD awDigiConst[5]={10000,1000,100,10,1};
 87
 88 /* ７セグメント表示スキャンビットパターン ***********************************/
 89 static const BYTE abyDigiPtn[DIGI_MAX]={0xFE, 0xFD, 0xFB, 0xF7, 0xEF};
 90
 91 /*****************************************************************************/
 92 /*                表示バッファSegmentの内容をダイナミック表示                */
 93 /*****************************************************************************/
 94 /* ※2mS周期割り込みから呼ばれる。                                           */
 95 /* ※割り込みマスク状態で実行すること。                                      */
 96 #pragma abs8(byDigCnt)
 97 static BYTE byDigCnt;
 98
 99 /* LED表示OFF */
100 void ScanLED_off(void)
101 {
102     /* 表示OFF (Source Driver OFF)  */
103     IO.PDR5.BYTE = 0x1F;                    /* P54～P50:Hi出力 (全消灯)  */
104     IO.PDR8.BYTE = 0x00;                    /* P87～P80:Low出力 (全消灯)  */
105 }
106
107 /* LED表示ON */
108 void ScanLED_on(void)
109 {
110     /* 表示桁位置を次へ */
111     byDigCnt++;
112     if(byDigCnt >= DIGI_MAX)
113         byDigCnt = 0;
114     /* 新しい桁位置のパターンを表示 */
115     IO.PDR8.BYTE = Segment.buff[byDigCnt];  /* P87～P80:セグメントパターン出力 */
116     /* 表示ON (Source Driver ON)  */
117     IO.PDR5.BYTE = abyDigiPtn[byDigCnt];    /* P54～P50:順次Low出力 */
118 }
119
120 /*****************************************************************************/
121 /*                            表示支援関数                                   */
122 /*****************************************************************************/
123
124 /*  ディスプレイ制御の初期化 *************************************************/
125
126 void InitDispLED(void)
127 {
128     /* I/O ポート５ */
129     IO.PUCR5.BYTE = 0x00;           /* P55～P50:プルアップ抵抗無効 */
130     IO.PDR5.BYTE = 0x1F;            /* P54～P50:Hi出力 (全消灯)  */
131     IO.PMR5.BYTE = 0x00;            /* P55～P50:汎用ポート */
132     IO.PCR5      = 0x1F;            /* P57,P56:0のみ可、P55:入力ポート、P54～P50:出力ポート */
133     /* I/O ポート８ */
134     IO.PDR8.BYTE = 0x00;            /* P87～P80:Low出力 (全消灯)  */
135     IO.PCR8      = 0xFF;            /* P87～P80:出力ポート */
136     /* タイマW */
137     TW.TIOR0.BYTE = 0x88;           /* P82:FTIOB出力無効、P81:FTIOA出力無効 */
138     TW.TIOR1.BYTE = 0x88;           /* P84:FTIOD出力無効、P83:FTIOC出力無効 */
139
140     byDigCnt = 0;                   /* スキャン桁＝０が初期値 */
141 }
142
143 /* 全LED点灯または消灯  *****************************************************/
144
145 void DispAllLED(BYTE bySW)
146 {
147     WORD    wLpcnt;
148
149     if(bySW != 0){
150         bySW = 0xFF;
151     }
152     wLpcnt = 0;
153     do{
154         Segment.buff[wLpcnt] = bySW;
155     }while(++wLpcnt < DIGI_MAX);
156 }
157
158 /* 記号表示LEDの表示  *******************************************************/
159
160 void DispIndicator(BYTE byLedPtn)
161 {
162     Segment.Digi.COL.BYTE = byLedPtn;
163 }
164
165 /* 記号表示LEDの点灯  *******************************************************/
166
167 void DispIndicatorON(BYTE byLedPtn)
168 {
```

```
    Segment.Digi.COL.BYTE |= byLedPtn;
}

/* 記号表示LEDの消灯  ********************************************************/

void DispIndicatorOFF(BYTE byLedPtn)
{
    Segment.Digi.COL.BYTE &= (BYTE)(~byLedPtn);
}

/* ７セグメント表示消去  ******************************************************/
/* ※小数点とコロンも消去。                                                   */

void DispClr7Seg(void)
{
    int     nLpcnt;

    /* ７セグメントを消去 */
    nLpcnt = 0;
    do{
        Segment.buff[nLpcnt] = SEGPTN_SP;
    }while(++nLpcnt < DIGI_MAX-1);
    /* コロン消去 */
    Segment.Digi.COL.BYTE &= (BYTE)(~(LED_COLH | LED_COLL));
}

/* 小数点の表示  **************************************************************/
/* 引数 : byDPptn = [76543210]                                                */
/*      :             ||||||||    '1'で点灯                                   */
/*      :             xxxx||||_0_ 最も右桁                                   */
/*      :                 |||__1_ 左から３桁目                               */
/*      :                 ||___2_ 左から２桁目                               */
/*      :                 |____3_ 最も左側                                   */
/* 戻値 : なし                                                                */

void DispDecPoint(BYTE byDPptn)
{
    BYTE    *pbyTgt;
    int     nLpcnt;

    /* 表示 */
    pbyTgt = &Segment.buff[0];
    nLpcnt = 0;
    do{
        if((byDPptn & 0x01) != 0)
            *pbyTgt |= 0x80;
        else
            *pbyTgt &= 0x7F;
        pbyTgt++;
        byDPptn >>= 1;
    }while(++nLpcnt < DIGI_MAX-1);
}

/* 指定の表示部に４桁の任意パターン表示  **************************************/
/* ※小数点は上書きされる。                                                   */

void DispSegP4(BYTE *pbyPtn)
{
    BYTE    *pbyTgt;
    WORD    wLpcnt;

    /* 表示 */
    pbyTgt = &Segment.buff[4];
    wLpcnt = 0;
    do{
        *(--pbyTgt) = *(pbyPtn++);          /* 小数点はすべて上書き */
    }while(++wLpcnt < DIGI_MAX-1);
}

/* 指定の表示部に指定桁の任意パターン表示  ************************************/
/* ※小数点は保存される。                                                     */

void DispSegPattern(BYTE byDigit, BYTE byWidth, BYTE *pbyPtn)
{
    BYTE    *pbyTgt;
    WORD    wLpcnt;

    /* 指定桁のチェック */
    if(byDigit > (DIGI_MAX-2))  byDigit = (DIGI_MAX-2);
    if((byDigit + byWidth) > (DIGI_MAX-1))
        byWidth = (BYTE)((DIGI_MAX-1) - byDigit);
    /* 表示 */
    pbyTgt = &Segment.buff[(DIGI_MAX-1) - byDigit];
    wLpcnt = 0;
```

```
253     do{
254         --pbyTgt;
255         *pbyTgt = (BYTE)((*pbyTgt & 0x80) | *(pbyPtn++));
256     }while(++wLpcnt < (WORD)byWidth);
257 }
258 
259 /* 指定の表示部に16進４桁で数値表示  ****************************************/
260 /* ※小数点は保存されない。                                                 */
261 
262 void DispSegH4Word(WORD wData)
263 {
264     BYTE    *pbyTgt;
265     WORD    wHex;
266     int     nLpcnt;
267 
268     /* 16進変換→表示バッファにコピー */
269     pbyTgt = &Segment.buff[0];
270     nLpcnt = 0;
271     do{
272         wHex = (WORD)(wData & 0x000F);
273         *(pbyTgt++) = HexDecimalPtn[wHex];
274         wData >>= 4;
275     }while(++nLpcnt < DIGI_MAX-1);
276 }
277 
278 /* 指定の表示部に10進で数値４桁表示  ****************************************/
279 /* ※小数点は保存される。                                                   */
280 
281 void DispSegD4Word(BYTE bySuppress, WORD wData)
282 {
283     BYTE    abyBuff[6];
284     WORD    *pwConst;
285     BYTE    *pbyData,*pbyTgt;
286     WORD    wDigit,wAns;
287 
288     /* 10進数変換＆セグメントパターン変換（５桁） */
289     pbyData = &abyBuff[0];
290     pwConst = &awDigiConst[0];
291     wDigit = 0;
292     do{
293         wAns  = wData / *pwConst;
294         wData = wData % *pwConst;
295         pwConst++;
296         *(pbyData++) = HexDecimalPtn[wAns];
297     }while(++wDigit < 4);
298     *pbyData = HexDecimalPtn[wData];
299     /* ゼロサプレス */
300     if(bySuppress != 0){
301         pbyData = &abyBuff[0];
302         wDigit = 0;
303         do{
304             if(*pbyData != SEGPTN_0)    break;
305             *(pbyData++) = SEGPTN_SP;
306         }while(++wDigit < 4);
307     }
308     /* 対象の表示バッファアドレスセット */
309     pbyTgt = &Segment.buff[DIGI_MAX-1];
310     /* ７セグパターンで表示バッファにコピー（下位４ケタ） */
311     pbyData = &abyBuff[1];
312     wDigit = 0;
313     do{
314         --pbyTgt;
315         *pbyTgt = (BYTE)((*pbyTgt & 0x80) | *(pbyData++));
316     }while(++wDigit < DIGI_MAX-1);
317 }
318 
319 /* 指定の表示部に10進で２桁数値表示  ****************************************/
320 /* ※小数点は保存される。                                                   */
321 
322 void DispSegD2Byte(BYTE byDigit, BYTE bySuppress, BYTE byData)
323 {
324     BYTE    abyBuff[4];
325     WORD    *pwConst;
326     BYTE    *pbyData,*pbyTgt;
327     WORD    wDigit;
328     BYTE    byAns;
329 
330     /* 桁位置チェック */
331     if(byDigit > 2) byDigit = 2;
332     /* 10進数変換＆セグメントパターン変換（３桁） */
333     pbyData = &abyBuff[0];
334     pwConst = &awDigiConst[2];
335     wDigit = 0;
336     do{
```

```
337         byAns  = (BYTE)(byData / *pwConst);
338         byData = (BYTE)(byData % *pwConst);
339         pwConst++;
340         *(pbyData++) = HexDecimalPtn[byAns];
341     }while(++wDigit < 2);
342     *pbyData = HexDecimalPtn[byData];
343     /* ゼロサプレス */
344     if(bySuppress != 0){
345         pbyData = &abyBuff[0];
346         wDigit = 0;
347         do{
348             if(*pbyData != SEGPTN_0)    break;
349             *(pbyData++) = SEGPTN_SP;
350         }while(++wDigit < 2);
351     }
352 
353     /* 対象の表示バッファアドレスセット */
354     pbyTgt = &Segment.buff[(DIGI_MAX-1) - byDigit - 2];
355     /* ７セグパターンで表示バッファに下位からコピー（下位２ケタ） */
356     pbyData = &abyBuff[3];
357     wDigit = 0;
358     do{
359         --pbyData;
360         *pbyTgt = (BYTE)((*pbyTgt & 0x80) | *pbyData);
361         pbyTgt++;
362     }while(++wDigit < 2);
363 }
364 
365 /* 指定の表示部に符号付き10進で数値表示  ************************************/
366 
367 void DispSegF4Word(BYTE byPoint, short nData)
368 {
369     BYTE    abyBuff[DIGI_MAX+1];
370     WORD    *pwConst;
371     BYTE    *pbyData,*pbyTgt;
372     WORD    wDigit,wPoint,wAns,wData;
373     BOOL    bMinus;
374 
375     /* 小数点位置指定の変換 */
376     wPoint = (DIGI_MAX-1) - (WORD)byPoint;
377     /* 符号を取り出して正数化する */
378     if(bMinus = (nData < 0))
379         wData = (WORD)(0 - nData);
380     else
381         wData = (WORD)nData;
382     /* 10進数変換＆セグメントパターン変換 */
383     pbyData = &abyBuff[0];
384     pwConst = &awDigiConst[0];
385     wDigit = 0;
386     do{
387         wAns  = wData / *pwConst;
388         wData = wData % *pwConst;
389         pwConst++;
390         *(pbyData++) = HexDecimalPtn[wAns];
391     }while(++wDigit < 4);
392     *pbyData = HexDecimalPtn[wData];
393     /* ゼロサプレス */
394     pbyData = &abyBuff[0];
395     for(wDigit = 0;(wDigit < (DIGI_MAX-1)) && (wDigit < wPoint);wDigit++){
396         if(*pbyData != SEGPTN_0)    break;
397         *(pbyData++) = SEGPTN_SP;
398     }
399     /* マイナス表示 */
400     if((wDigit != 0) && (bMinus != FALSE)){
401         *(pbyData-1) = SEGPTN_MN;
402     }
403     /* 対象の表示バッファアドレスセット */
404     pbyTgt = &Segment.buff[DIGI_MAX-1];
405     /* ７セグパターンで表示バッファにコピー */
406     pbyData = &abyBuff[1];
407     wDigit = 1;
408     do{
409         --pbyTgt;
410         if(wDigit == wPoint)
411             *pbyTgt = (BYTE)(*pbyData | 0x80);  /* 小数点を追加 */
412         else
413             *pbyTgt = *pbyData;
414         pbyData++;
415     }while(++wDigit < 5);
416 }
417 
418 /* 指定の表示部にオーバーフロー／アンダーフロー表示  *************************/
419 
420 void DispOverFlow(BOOL bOver)
```

```
{
    static const BYTE  ptnOver[4]={SEGPTN_SP,SEGPTN_OV,SEGPTN_OV,SEGPTN_OV};
    static const BYTE ptnUnder[4]={SEGPTN_SP,SEGPTN_UN,SEGPTN_UN,SEGPTN_UN};

    /* バーを表示 */
    if(bOver)
        DispSegP4(ptnOver);
    else
        DispSegP4(ptnUnder);
}

/* 指定の表示部にMM.DDまたはhh:mmを10進で表示  ********************************/

void DispSegD4DTime(BOOL bDate, BYTE *pData)
{
    BYTE    byDat,byDec;
    BYTE    abyBuff[DIGI_MAX-1];
    BYTE    *pbyData,*pbyTgt;
    int     nLpcnt;

    /* 10進数→セグメントパターンに変換 */
    pbyData = &abyBuff[0];
    nLpcnt = 0;
    do{
        /* 10進変換 */
        byDat = *(pData++);
        if(byDat > 99)  byDat = 99;
        byDec = (BYTE)(byDat / 10);
        byDat = (BYTE)(byDat % 10);
        /* 10位 */
        if(byDec == 0 && (nLpcnt == 0 || bDate != FALSE))
            *(pbyData++) = 0;       /* blank */
        else
            *(pbyData++) = HexDecimalPtn[byDec];
        /* 1位 */
        *(pbyData++) = HexDecimalPtn[byDat];
    }while(++nLpcnt < 2);
    /* 対象の表示バッファアドレスセット */
    pbyTgt = &Segment.buff[DIGI_MAX-1];
    /* 表示バッファにコピー */
    pbyData  = &abyBuff[0];
    nLpcnt = 0;
    do{
        --pbyTgt;
        if((nLpcnt == 1) && (bDate != FALSE))
            *pbyTgt = (BYTE)(*pbyData | 0x80);  /* 小数点を追加 */
        else
            *pbyTgt = *pbyData;
        pbyData++;
    }while(++nLpcnt < DIGI_MAX-1);
    /* コロンの表示・消去 */
    if(bDate)
        Segment.Digi.COL.BYTE &= (BYTE)~(LED_COLL|LED_COLH);
    else
        Segment.Digi.COL.BYTE |= (LED_COLL|LED_COLH);
}
```