```
/************************************************************************/
/*                  タイマWによるインターバルタイマ                    */
/************************************************************************/

#include <machine.h>
#include "iodefine.h"
#include "typedefine.h"
#include "ADConvert.h"
#include "Timer.h"

#pragma section
/************************************************************************/
/*                      インターバルタイマ起動                          */
/*                                                                      */
/************************************************************************/
/* ※割込み禁止状態で実行してください。                                 */
/* ※φ=20.0MHz (50.0nS)                                                 */
/* ※φ/8=400nS                                                          */
#define CNT_LEDON    (  100000/400)      /* LEDの ONタイミングmin (100uS/400nS=   250) */
#define CNT_LEDOFF   ( 2000000/400)      /* LEDのOFFタイミング    (  2mS/400nS= 5,000) */
#define CNT_1MS      ( 1000000/400)      /*  1mS周期発生タイミング(  1mS/400nS= 2,500) */
#define CNT_10MS     (10000000/400)      /* 10mS周期発生タイミング( 10mS/400nS=25,000) */

void start_timer_int(void)
{
    /* インターバルタイマスタート（タイマW） */
    TW.TCNT = 0x0000;                    /* カウンタクリア          */
    TW.TCRW.BYTE  = 0x30;                /* クロック＝φ／8(=400nS)を指定、カウンタクリア禁止      */
    TW.TMRW.BYTE  = 0x48;                /* GRC,GRDはアウトプットコンペアマッチレジスタとする      */
    TW.TIERW.BYTE = 0x7F;                /* TSRWのIMFA～IMFDで割り込み許可                         */
    TW.GRA = CNT_LEDON;                  /* GRAコンペアマッチレジスタセット （ディスプレイリフレッシュ用） */
    TW.GRB = CNT_LEDOFF;                 /* GRBコンペアマッチレジスタセット （ディスプレイリフレッシュ用） */
    TW.GRC = CNT_1MS;                    /* GRCコンペアマッチレジスタセット （ 1mS周期割り込み発生用） */
    TW.GRD = CNT_10MS;                   /* GRDコンペアマッチレジスタセット （10mS周期割り込み発生用） */
    TW.TIOR0.BYTE = 0x88;                /* FTIOA,FTIOB端子出力禁止  */
    TW.TIOR1.BYTE = 0x88;                /* FTIOC,FTIOD端子出力禁止  */
    TW.TSRW.BYTE  = 0x70;                /* タイマステータスクリア   */
    TW.TMRW.BYTE  = 0xC8;                /* タイマWスタート          */

    /*  0.5mSタイマスタート（タイマV） ※ブザー発音用  */
    TV.TCNTV = 0x00;                     /* カウンタクリア          */
    TV.TCRV0.BYTE = 0x0B;                /* TCSRVのCMFA,CMFB,OVF割り込み禁止、コンペアマッチＡでカウンタクリア、クロック＝φ/128 (=6.4uS) を指定 */
    TV.TCRV1.BYTE = 0x03;                /* TRGV入力禁止、TCRV0と組でクロック＝φ/128 (=6.4uS) を指定 */
    TV.TCSRV.BYTE = 0x10;                /* タイマステータスクリア、コンペアマッチＡでTMOV端子を変化しない */
//  TV.TCSRV.BYTE = 0x13;                /* タイマステータスクリア、コンペアマッチＡでTMOV端子をトグル出力（PCR76の設定に無関係） */
    TV.TCORA = 26;                       /* 1/φ×128×26 = 0.166mS（約3kHzを出力） */
    TV.TCORB = 0;                        /* コンペアマッチＢレジスタ不使用 */
}

/************************************************************************/
/*                    明るさを0～3000にして保存する                     */
/************************************************************************/
/* 引数 : wBright = 明るさ : 0（明）～1023（暗）                        */
/* 戻値 : なし                                                          */
/*      : (wBrightness) = 0（明）～3000（暗）                           */

static WORD wBrightness;                 /* 0（明）～3000（暗） */

void StoreBrightness(WORD wBright)
{
    WORD    wWork;

    if(wBright < 500){
        wBrightness = 0;                 /* 500未満は0とする */
        return;
    }
    wBright = wBright - 524;

    wWork = wBright << 1;
    wWork = wWork + (wBright << 2);      /* ６倍（0～3138）にする */
    if(wWork > 3000)
        wBrightness = 3000;              /* 0～3000に制限する */
    else
        wBrightness = wWork;
}

```

```
 77
 78 /*****************************************************************************/
 79 /*                タイマWコンペアマッチＡ～Ｄ割り込み                        */
 80 /*                  （汎用インターバルタイマ割込み）                         */
 81 /*****************************************************************************/
 82 #pragma abs8(byCnt100mS)
 83 static BYTE byCnt100mS;
 84
 85 void startAD(void);
 86 void KeyChatCancel(void);
 87 void ScanLED_off(void);
 88 void ScanLED_on(void);
 89 void BuzzIntermit(void);
 90 void MainTimer_10mS(void);
 91 static WORD AdjBrightness(void);
 92
 93 __interrupt void Int_TMRW(void)
 94 {
 95     /* コンペアマッチＡ割り込み（LED_ON用） */
 96     if(TW.TSRW.BIT.IMFA == 1){
 97         TW.TSRW.BIT.IMFA = 0;              /* コンペアマッチフラグＡクリア  */
 98         TW.GRA = (WORD)(TW.GRB + CNT_LEDON + wBrightness);  /* 次のLED_ONタイミングセット    */
 99         /* LEDディスプレイ点灯（ダイナミック用） */
100         ScanLED_on();
101     }
102     /* コンペアマッチＢ割り込み（LED_OFF用） */
103     if(TW.TSRW.BIT.IMFB == 1){
104         TW.TSRW.BIT.IMFB = 0;              /* コンペアマッチフラグＢクリア  */
105         TW.GRB += CNT_LEDOFF;              /* 次のLED_OFFタイミングをセット */
106         /* LEDディスプレイ消灯（ダイナミック用） */
107         ScanLED_off();
108     }
109     /* コンペアマッチＣ割り込み（ 1mS周期） */
110     if(TW.TSRW.BIT.IMFC == 1){
111         TW.TSRW.BIT.IMFC = 0;              /* コンペアマッチフラグＣクリア */
112         TW.GRC += CNT_1MS;                 /* 次の1mSをセット */
113         /* ここに1mS周期の関数呼び出しを記述する */
114
115     }
116     /* コンペアマッチＤ割り込み（10mS周期） */
117     if(TW.TSRW.BIT.IMFD == 1){
118         TW.TSRW.BIT.IMFD = 0;              /* コンペアマッチフラグＤクリア */
119         TW.GRD += CNT_10MS;                /* 次の10mSをセット */
120         /* ここに10mS周期の関数呼び出しを記述する */
121         KeyChatCancel();
122         MainTimer_10mS();
123
124         if(++byCnt100mS >= 10){
125             byCnt100mS = 0;
126             /* ここに100mS周期の関数呼び出しを記述する */
127             StartAD();
128             BuzzIntermit();
129         }
130     }
131 }
132
133 /*****************************************************************************/
134 /*             100mS周期割り込みで呼び、ブザー音の断続を行う                 */
135 /*****************************************************************************/
136
137 #pragma abs8(BuzzValid,BuzzValid2,BuzzPattern,BuzzCount)
138 static BYTE BuzzValid,BuzzValid2;
139 static BYTE BuzzPattern;
140 static BYTE BuzzCount;
141 static const BYTE BitPos[8] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80};
142
143 static void BuzzIntermit(void)
144 {
145     if(BuzzValid == 0){
146         /* 発音停止（コンペアマッチＡでTMOV端子を変化しない） */
147         if(BuzzValid2 != 0){                    /* 発音中か？ */
148             BuzzValid2 = 0;
149             TV.TCSRV.BYTE = 0x10;
150         }
151     }
152     else{
153         /* 発音中を示す */
154         BuzzValid2 = 1;
155         /* 発音開始（コンペアマッチＡでTMOV端子をトグル出力） */
156         if((BuzzPattern & BitPos[BuzzCount]) != 0){
157             TV.TCSRV.BYTE = 0x13;
158
159         }
160         /* 発音停止（コンペアマッチＡでTMOV端子を変化しない） */
```

```
161         else{
162             TV.TCSRV.BYTE = 0x10;
163         }
164         /* ビット位置カウンタ更新 */
165         BuzzCount++;
166         if(BuzzCount >= 8)  BuzzCount = 0;
167     }
168 }
169 
170 /*****************************************************************************/
171 /*                                  ブザー制御                               */
172 /*****************************************************************************/
173 /* 引数 : なし                                                               */
174 /* 戻値 :  BuzzON() = なし                                                   */
175 /*      : BuzzOFF() = なし                                                   */
176 
177 void BuzzON(void)
178 {
179     BYTE    byCCR;
180 
181     /* 割り込みマスク */
182     byCCR = get_ccr();
183     set_imask_ccr(1);
184     /* ピピピピ…という音発生 */
185     BuzzPattern = 0x55;
186     BuzzCount = 0;
187     BuzzValid = 1;
188     /* 割り込みマスク復帰 */
189     set_ccr(byCCR);
190 }
191 
192 void BuzzOFF(void)
193 {
194     BuzzValid = 0;
195 }
196 
197 /* End of File */
```