```
/************************************************************************/
/***              リアルタイムクロック(RTC-8564NB)の制御                ***/
/************************************************************************/

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


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

typedef union {
    unsigned char Array[7];
    struct {
        unsigned char Week;         /* 0 */
        unsigned char Year;         /* 1 */
        unsigned char Month;        /* 2 */
        unsigned char Day;          /* 3 */
        unsigned char Hour;         /* 4 */
        unsigned char Min;          /* 5 */
        unsigned char Sec;          /* 6 */
    } DT;
}DTIME_T;

/************************************************************************/
/*                          BCD <--> BIN 変換                           */
/************************************************************************/

static BYTE BcdToBin1(BYTE byBCD)
{
    return (BYTE)((byBCD >> 4) * 10 + (byBCD & 0x0F));
}

static BYTE BinToBcd1(BYTE byBin)
{
    return (BYTE)(((byBin / 10) << 4) + (byBin % 10));
}

/************************************************************************/
/*                  RTCからの１秒割り込み(IRQ0)の処理                   */
/************************************************************************/
static BOOL bSecIntFlg;

__interrupt void Int_IRQ0(void)
{
    if(IRR1.BIT.IRRI0 == 0)
        return;
    IRR1.BIT.IRRI0 = 0;
    /* １秒フラグセット */
    bSecIntFlg = TRUE;
}

/************************************************************************/
/*                            RTCの初期化                               */
/************************************************************************/

static BOOL RTC_TimeSet(BYTE *pbyDtime);
static BOOL RTC_TimeRead(BYTE *pbyDtime);

BOOL InitRTC(void)
{
    static const BYTE aCtrReg[2]={0x00,     /* 0x00:制御レジスタ１設定：TEST=0, STOP=0 */
                                  0x11};    /* 0x01:制御レジスタ２設定：TI/IP=1:タイマリピートモード,AIE=0:アラーム割り
込み禁止,TIE=1:タイマ割り込み許可 */
    static const BYTE aAlmReg[8]={0x80,     /* 0x09:分アラーム設定：AE=1（アラーム無効） */
                                  0x80,     /* 0x0A:時アラーム設定：AE=1（アラーム無効） */
                                  0x80,     /* 0x0B:日アラーム設定：AE=1（アラーム無効） */
                                  0x80,     /* 0x0C:曜アラーム設定：AE=1（アラーム無効） */
                                  0x00,     /* 0x0D:CLKOUT端子設定：FE=0:出力せず */
                                  0x02,     /* 0x0E:タイマ制御設定：TE=0:タイマ割り込み停止、TD1=1,TD0=0:1Hz（1Hzにする
と秒に同期できる） */
                                  1,        /* 0x0F:タイマカウンタ初期値:1（TE=0のときにセットするのが良い） */
                                  0x82};    /* 0x0E:タイマ制御設定：TE=1:タイマカウント動作、TD1=1,TD0=0:1Hz（1Hzにする
と秒に同期できる） */
    static const BYTE DefDate[6]={13,1,1,12,0,0};   /* バックアップ無効時に設定されるデフォルト日時 */
    BYTE    byData;
    BOOL    bValid;

    /* IIC2初期化（スレーブアドレスセット） */
    iic2_init(0xA2);
    /* VLビットを読み込む */
    bValid = Read1_IIC(0x02, &byData);
    /* VLビットをチェック */
    if(((byData & 0x80) != 0) || (bValid == FALSE)){
        /* 制御レジスタ１，２設定 */
        if(!WriteP_IIC(0x00, 2, aCtrReg))
```

```
            return FALSE;
        /* RTCに日時設定 */
        if(!RTC_TimeSet(DefDate))
            return FALSE;
        /* アラーム・タイマ制御設定 */
        if(!WriteP_IIC(0x09, 7, aAlmReg))
            return FALSE;
        /* タイマカウント開始 */
        if(!WriteP_IIC(0x0E, 1, &aAlmReg[7]))
            return FALSE;
    }
    /* １秒フラグセット */
    bSecIntFlg = TRUE;

    IENR1.BIT.IEN0 = 1;                     /* IRQ0割り込みイネーブル */
    return TRUE;
}

/*****************************************************************************/
/*                  RTCにバイナリで日時を設定する                            */
/*****************************************************************************/
/* 注意：曜日は設定できない。                                                */
/* ※ pbyDtime = YYMMDD-hhmmssの先頭アドレス（各バイナリ）                   */

static BOOL RTC_TimeSet(BYTE *pbyDtime)
{
    static const BYTE byStop =0x20;     /* 0x00:制御レジスタ１：STOP=1 */
    static const BYTE byStart=0x00;     /* 0x00:制御レジスタ１：STOP=0 */
    BYTE    abyBuff[8];
    int     lpcnt;
    BOOL    result;

    /* ssmmhhDDwwMMYYの順に並び替える */
    pbyDtime += 6;
    lpcnt = 0;
    do{
        --pbyDtime;
        abyBuff[lpcnt] = BinToBcd1(*pbyDtime);
    }while(++lpcnt < 4);
    lpcnt++;                /* skip WEEK area */
    do{
        --pbyDtime;
        abyBuff[lpcnt] = BinToBcd1(*pbyDtime);
    }while(++lpcnt < 7);
    /* 曜日は０にする */
    abyBuff[4] = 0;
    /* 時計カウント停止 */
    Write1_IIC(0x00, &byStop);
    /* 日付・時刻設定 */
    result = WriteP_IIC(0x02, 7, &abyBuff[0]);
    /* 時計カウント開始 */
    Write1_IIC(0x00, &byStart);

    return result;
}

/*****************************************************************************/
/*                          RTCから日時を読み出す                            */
/*****************************************************************************/
/* ※ (pbyDtime) = WW-YYMMDD-hhmmssの順に格納する（各BCD、WWは無効値）       */

static BOOL RTC_TimeRead(BYTE *pbyDtime)
{                               /* 0秒, 1分, 2時, 3日, 4曜, 5月, 6年 */
    static const BYTE MaskBit[7]={0x7F,0x7F,0x3F,0x3F,0x07,0x1F,0xFF};
    BYTE    abyBuff[8];
    int     lpcnt;

    /* 時刻, 曜日, 日付読み出し */
    if(ReadP_IIC(0x02, 7, &abyBuff[0]) == FALSE)
        return FALSE;
    /* 時刻を逆順に並び替える（不要ビットは削除） */
    pbyDtime += 7;
    lpcnt = 0;
    do{
        *(--pbyDtime) = (BYTE)(abyBuff[lpcnt] & MaskBit[lpcnt]);
    }while(++lpcnt < 4);
    /* skip WEEK area */
    lpcnt++;
    /* 日付を逆順に並び替える（不要ビットは削除） */
    do{
        *(--pbyDtime) = (BYTE)(abyBuff[lpcnt] & MaskBit[lpcnt]);
    }while(++lpcnt < 7);
    /* 曜日（無効な値） */
    *(--pbyDtime) = (BYTE)(abyBuff[4] & MaskBit[4]);

    return TRUE;
}
```

```
/*****************************************************************************/
/*                                  現在日時を得る                           */
/*****************************************************************************/
/* ※ (pDtime) = WW-YYMMDD-hhmmssの順に格納する（各バイナリ、WWは無効値）     */

void GetDtime(DTIME_T *pDtime)
{
    BYTE    abyDtime[7];
    BYTE    byBCD;
    int     lpcnt;

    /* RTCから日時読み込み */
    if(!RTC_TimeRead(abyDtime))
        return;

    /* バイナリに変換しながら読み込む */
    lpcnt = 0;
    do{
        byBCD = abyDtime[lpcnt];
        pDtime->Array[lpcnt] = BcdToBin1(byBCD);
    }while(++lpcnt < 7);
}

/*****************************************************************************/
/*                                  日時を設定する                           */
/*****************************************************************************/
/* 注意 : 曜日は設定できない。                                               */
/* ※ pDtime = YYMMDD-hhmmssの先頭アドレス（各バイナリ）                      */

void PutDtime(DTIME_T *pDtime)
{
    /* RTCに日時設定 */
    RTC_TimeSet(&pDtime->Array[1]);
}

/*****************************************************************************/
/*                      RTCからアラーム設定「時分」を読み出す                */
/*****************************************************************************/
/* 引数 : pbyHour = アラーム設定「時分」を格納するアドレス                   */
/* 戻値 : なし                                                               */

void GetAlarmTime(BYTE* pbyHour)
{
    BYTE    abyBuff[4];

    /* アラーム設定読み出し */
    if(ReadP_IIC(0x09, 4, &abyBuff[0]) == FALSE)
        return;
    /* 設定の'時分'を取り出す */
    *pbyHour     = BcdToBin1((BYTE)(abyBuff[1] & 0x3F));
    *(pbyHour+1) = BcdToBin1((BYTE)(abyBuff[0] & 0x7F));
}

/*****************************************************************************/
/*                      RTCにアラーム設定「時分」を設定する                  */
/*****************************************************************************/

void PutAlarmTime(BOOL bEnable, BYTE* pbyHour)
{
    BYTE    abyBuff[4];
    BYTE    byCTR2,byWrk;

    /* バッファに'時分'のみをセットする */
    abyBuff[0] = BinToBcd1(*(pbyHour+1));       /* 分 */
    abyBuff[1] = BinToBcd1(*pbyHour);           /* 時 */
    abyBuff[2] = 0x80;                          /* 日（不問）*/
    abyBuff[3] = 0x80;                          /* 曜（不問）*/
    /* AIE保存のためコントロールレジスタ２を読み込む */
    if(Read1_IIC(0x01, &byCTR2) == FALSE)
        return;
    /* 一時AIEビットを'0'にする */
    byWrk = (BYTE)(byCTR2 & 0xFD);
    Write1_IIC(0x01, &byWrk);
    /* アラーム設定書き込み */
    WriteP_IIC(0x09, 4, &abyBuff[0]);
    /* AFビットを'0'にし、指示あればAIEビットを'1'にする */
    byCTR2 &= 0xF7;
    if(bEnable) byCTR2 |= 0x02;
    Write1_IIC(0x01, &byCTR2);
}

/*****************************************************************************/
/*                  アラーム割り込みの許可・禁止状態を読み込む               */
/*****************************************************************************/
```

```
BOOL GetAlarmInt(void)
{
    BYTE    byCTR2;

    /* コントロールレジスタ２を読み込む */
    if(Read1_IIC(0x01, &byCTR2) == FALSE)
        return FALSE;

    return (byCTR2 & 0x02);
}

/*****************************************************************************/
/*                      アラーム割り込みを許可・禁止する                     */
/*****************************************************************************/

BOOL SetAlarmInt(BOOL bEnable)
{
    BYTE    byCTR2;

    /* コントロールレジスタ２を読み込む */
    if(Read1_IIC(0x01, &byCTR2) == FALSE)
        return FALSE;
    /* AIEビットを'0'または'1'にする */
    if(bEnable) byCTR2 |= 0x02;
    else        byCTR2 &= 0xFD;
    /* コントロールレジスタ２を書き込む */
    return Write1_IIC(0x01, &byCTR2);
}

/*****************************************************************************/
/*                        割り込み発生フラグを読み込む                       */
/*****************************************************************************/
/* 注意 : AF,TFはAIE,TIEの設定にかかわらず'1'になるのでペアで確認すること。    */
/* 引数 : なし                                                               */
/* 戻値 : GetRTCIntFlag() = [76543210]                                       */
/*      :                    ||||||||                                        */
/*      :                    0000||||                                        */
/*      :                        ||||_ TIE:周期割り込み有効                  */
/*      :                        |||__ AIE:アラーム割り込み有効              */
/*      :                        ||___  TF:周期割り込みあり                  */
/*      :                        |____  AF:アラーム割り込みあり              */

BYTE GetRTCIntFlag(void)
{
    BYTE    byCTR2,byWrk;

    /* コントロールレジスタ２を読み込む */
    if(Read1_IIC(0x01, &byCTR2) == FALSE)
        return 0;
    /* AF,TFビットを'0'にする */
    byWrk = (BYTE)(byCTR2 & 0xF3);
    /* コントロールレジスタ２を書き込む */
    Write1_IIC(0x01, &byWrk);

    return (BYTE)(byCTR2 & 0x0F);
}

/*****************************************************************************/
/*                            １秒フラグを読み込む                           */
/*****************************************************************************/

BOOL ChkSecIntFlg(void)
{
    BOOL bFlag;
    BYTE byCCR;

    /* 割り込みマスク */
    byCCR = get_ccr();
    set_imask_ccr(1);
    /* １秒フラグ読み込み＆クリア */
    bFlag = bSecIntFlg;
    bSecIntFlg = FALSE;
    /* 割り込みマスク復帰 */
    set_ccr(byCCR);

    return bFlag;
}
```