```
/*****************************************************************************/
/***                                                                       ***/
/***                    リアルタイムクロックRX-8025の制御                  ***/
/***                  （定周期割り込みとアラームW機能を使う）              ***/
/***                                                                       ***/
/*****************************************************************************/
/* 備考 :                                                                    */
/*  (1) この例では、RX-8025の割り込み出力を次のように接続し検知する。        */
/*      /INTA → IRQ0 立下りエッジ                                            */
/*      /INTB → IRQ1 立下りエッジ                                            */
/*  (2) 指定時刻に割り込みを発生するアラーム機能は、RX-8025のアラームW機能を */
/*      使って実現している。アラームD機能は使用していない。                  */
/*  (3) RX-8025からのデータ読み出しは標準モードのみで、短縮モードおよび      */
/*      アドレスを指定しないモードは使用していない。                         */
/*  (4) ここから呼んでいるI2C通信関数を下記に示す。                          */
/*       ("riic0_i2c.h"にプロトタイプがあるものとする)                       */
/*      SetSlaveAddr_IIC()...通信相手のスレーブアドレスセット                */
/*      WriteByte_A8_IIC()...指定レジスタに、１バイトのデータを書き込む      */
/*      WritePage_A8_IIC()...指定レジスタから順に、指定バイト数のデータを    */
/*                           書き込む                                        */
/*      ReadByte_A8_IIC()....指定レジスタから、１バイトのデータを読み込む    */
/*      ReadPage_A8_IIC()....指定レジスタから順に、指定バイト数のデータを    */
/*                           読み込む                                        */

#include    <machine.h>
#include    <stdlib.h>
#include    "typedefine.h"  // ←BYTE,WORD,DWORD,BOOLなどの型宣言あり
#include    "iodefine.h"

#include    "riic0_i2c.h"

/*****************************************************************************/
/* RX-8025レジスタアドレス */
#define RTCREG_SEC  0x00    /* 時計カウンタ：秒(Seconds)  */
#define RTCREG_MIN  0x10    /* 時計カウンタ：分(Minutes)  */
#define RTCREG_HOUR 0x20    /* 時計カウンタ：時(Hours)    */
#define RTCREG_WEEK 0x30    /* 時計カウンタ：曜(Weekdays) */
#define RTCREG_DAY  0x40    /* 日付カウンタ：日(Days)     */
#define RTCREG_MON  0x50    /* 日付カウンタ：月(Months)   */
#define RTCREG_YEAR 0x60    /* 日付カウンタ：年(Years)    */
#define RTCREG_ADJ  0x70    /* 時計精度調整レジスタ       */
#define RTCREG_AW_M 0x80    /* ：アラームW(Minute)        */
#define RTCREG_AW_H 0x90    /* ：アラームW(Hour)          */
#define RTCREG_AW_W 0xA0    /* ：アラームW(Week)          */
#define RTCREG_AD_M 0xB0    /* ：アラームD(Minute)        */
#define RTCREG_AD_H 0xC0    /* ：アラームD(Hour)          */
#define RTCREG_CTR1 0xE0    /* ：制御レジスタ１(Control1) */
#define RTCREG_CTR2 0xF0    /* ：制御レジスタ２(Control2) */

/*****************************************************************************/
#define IICADDR_RTC (0x64)          /* RX-8025のデバイスアドレス */

// 参考 : このDTIME_T型はヘッダファイルに移し、他のファイルからも使用できるようにする必要あり
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;

#pragma section
/*****************************************************************************/
/*                            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));
}

/*****************************************************************************/
/*                    割り込み処理：RTCから日時を読み出す                    */
/*****************************************************************************/
static BOOL bIntFlag_A;
static BOOL bIntFlag_B;

/* INTA : 定周期割り込み */

#pragma interrupt (Int_IRQ0)
void Int_IRQ0(void)
{
    /* １秒フラグセット */
    bIntFlag_A = TRUE;
}

/* INTB : アラームW割り込み */

#pragma interrupt (Int_IRQ1)
void Int_IRQ1(void)
{
    bIntFlag_B = TRUE;
}

/*****************************************************************************/
/*                              RTCの初期化                                  */
/*****************************************************************************/
/* 注意 : (1)IICインタフェースはあらかじめ初期化してあること。               */
/*      : (2)ここでは、RTCスレーブアドレスはセットされない。                 */
static BOOL SetTime_RTC(const BYTE *pbyDtime);
static BOOL ReadBCDTime_RTC(BYTE *pbyDtime);

BOOL Default_RTC(void)
{
    static const BYTE aCtrReg[2]={0x23,      /* 0xE0:制御レジスタ１設定：WALE=0,DALE=0,24Hmode,C
LEN2=0,TEST=0, CT2:0=3(1Hz-puls) */
                                  0x20};     /* 0xF0:制御レジスタ２設定：VDSL=0,VDET=0,XST=1,PON
=0,CLEN1=0,CTFG=0,WAFG=0,DAFG=0  */
    static const BYTE aAlmReg[5]={0x00,      /* 0x80:アラームW(Minute) */
                                  0x00,      /* 0x90:アラームW(Hour)   */
                                  0x00,      /* 0xA0:アラームW(Week)   */
                                  0x00,      /* 0xB0:アラームＤ(Minute) */
                                  0x00};     /* 0xC0:アラームＤ(Hour)   */
    static const BYTE DefDate[6]={19,1,1,12,0,0};   /* バックアップ無効時に設定されるデフォルト
日時 */
    BYTE    byError;

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    byError = 0;
    /* 制御レジスタ１，２を設定（アラームW，Ｄは無効化） */
    if(!WritePage_A8_IIC(RTCREG_CTR1, 2, aCtrReg)){
        byError |= 0x01;
    }
    /* RTCにデフォルトの日時を設定 */
    if(!SetTime_RTC(DefDate)){
        byError |= 0x02;
    }
    /* アラームW，Ｄの無効を設定 */
    if(!WritePage_A8_IIC(RTCREG_AW_M, 5, aAlmReg)){
        byError |= 0x04;
    }

    return (byError == 0);
}

BOOL Init_RTC(void)
{
```

```
    BYTE    byData;
    BOOL    bValid,bError;

    /* IRQ0割り込み禁止 */
    IEN(ICU, IRQ0) = 0;                 /* IRQ0=１秒周期とアラームＤによる割り込み */
    /* IRQ1割り込み禁止 */
    IEN(ICU, IRQ1) = 0;                 /* IRQ1=アラームＷによる割り込み */

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    /* VDET,XST,PONビットを読み込む */
    bValid = ReadByte_A8_IIC(RTCREG_CTR2, &byData);
    /* VDET,XST,PONビットをチェック */
    bError = TRUE;
    if(((byData & 0x50) != 0)           /* VDET,PONビットによる電源電圧低下検知 */
    || ((byData & 0x20) == 0)           /* XSTビットによる発振停止検知 */
    || (bValid == FALSE)){
        /* RTCにデフォルトの日時などを設定 */
        bError = Default_RTC();
    }
    /* １秒フラグセット */
    bIntFlag_A = TRUE;

/*** ↓以下はRX220の場合のIRQ0,IRQ1の設定 ***/
    /* IRQ0,IRQ1 (RTCの#INTA,#intB) のポート設定 */
    PORTH.PMR.BIT.B1 = 0;               /* PH1をI/Oポートとして使う設定 */
    PORTH.PMR.BIT.B2 = 0;               /* PH2をI/Oポートとして使う設定 */
    /* マルチファンクションピンコントローラ (MPC) */
    MPC.PWPR.BYTE = 0x00;               /* 書き込みプロテクトレジスタ (先にBOWI=0にすること)
 */
    MPC.PWPR.BYTE = 0x40;               /* 書き込みプロテクトレジスタ (次にPFSWE=1にして解除)
 */
    /* 端子機能制御レジスタ */
    MPC.PH1PFS.BYTE = 0x40;             /* PH1端子をIRQ0入力として使用する
 */
    MPC.PH2PFS.BYTE = 0x40;             /* PH2端子をIRQ1入力として使用する
 */
    /* マルチファンクションピンコントローラ (MPC) */
    MPC.PWPR.BYTE = 0x80;               /* 書き込みプロテクトレジスタ (PFSWE=0,BOWI=1)
 */
    /* IRQ0,IRQ1 (RTCの#INTA,#intB) のポート設定 */
    PORTH.PMR.BIT.B1 = 1;               /* PH1を周辺機能として使う設定 */
    PORTH.PMR.BIT.B2 = 1;               /* PH2を周辺機能として使う設定 */
    /* IRQ0,IRQ1端子デジタルフィルタ設定 */
    ICU.IRQFLTE0.BIT.FLTEN0 = 0;        /* IRQ0端子デジタルフィルタ無効 */
    ICU.IRQFLTE0.BIT.FLTEN1 = 0;        /* IRQ1端子デジタルフィルタ無効 */
    ICU.IRQFLTC0.BIT.FCLKSEL0 = 1;      /* IRQ0端子デジタルフィルタサンプリングクロック：PCLK/8
 */
    ICU.IRQFLTC0.BIT.FCLKSEL1 = 1;      /* IRQ1端子デジタルフィルタサンプリングクロック：PCLK/8
 */
    /* IRQ0,IRQ1のエッジ設定 */
    ICU.IRQCR[0].BIT.IRQMD = 1;         /* IRQ0を立下りエッジ検出に     */
    ICU.IRQCR[1].BIT.IRQMD = 1;         /* IRQ1を立下りエッジ検出に     */
    /* IRQ0,IRQ1ステータスフラグクリア */
    IR(ICU, IRQ0) = 0;
    IR(ICU, IRQ1) = 0;
    /* IRQ0,IRQ1の優先順位設定 */
    IPR(ICU, IRQ0) = 4;                 /* IRQ0優先順位設定 */
    IPR(ICU, IRQ1) = 4;                 /* IRQ1優先順位設定 */
    /* IRQ0,IRQ1端子デジタルフィルタ設定 */
    ICU.IRQFLTE0.BIT.FLTEN0 = 1;        /* IRQ0端子デジタルフィルタ有効 */
    ICU.IRQFLTE0.BIT.FLTEN1 = 1;        /* IRQ1端子デジタルフィルタ有効 */
    /* IRQ0割り込み許可 */
    IEN(ICU, IRQ0) = 1;
    /* IRQ1割り込み許可 */
    IEN(ICU, IRQ1) = 1;
/*** ↑ここまではRX220の場合のIRQ0,IRQ1の設定 ***/

    return bError;
}

/*****************************************************************************/
/*                  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。                                                    */
/* 引数 : pbyDtime = YYMMDD-hhmmssの順（各バイナリ）                            */
/* 戻値 : SetTime_RTC() =  TRUE : 正常終了                                      */
/*      :                = FALSE : 異常あり                                     */

BYTE aWeek(BYTE year,BYTE month,BYTE day);

static BOOL SetTime_RTC(const BYTE *pbyDtime)
{
    BYTE    abyBuff[8];
    int     nLpcnt;
    BOOL    bResult;

    /* BCD変換し"ssmmhhwwDDMMYY"の順に並び替える */
    /* まず"ssmmhh"の3byteをセット */
    pbyDtime += 6;
    nLpcnt = 0;
    do{
        --pbyDtime;
        abyBuff[nLpcnt] = BinToBcd1(*pbyDtime);
    }while(++nLpcnt < 3);
    /* 曜日"ww"を計算してセット */
    abyBuff[nLpcnt++] = aWeek(*(pbyDtime-3), *(pbyDtime-2), *(pbyDtime-1));
    /* 次に"DDMMYY"の3byteをセット */
    do{
        --pbyDtime;
        abyBuff[nLpcnt] = BinToBcd1(*pbyDtime);
    }while(++nLpcnt < 7);

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    /* 日付・時刻設定 */
    bResult = WritePage_A8_IIC(RTCREG_SEC, 7, &abyBuff[0]);

    /* コントロールレジスタ２を読み込む */
    if(ReadByte_A8_IIC(RTCREG_CTR2, &abyBuff[0]) == FALSE)
        return FALSE;
    /* VDETフラグを'0'にする */
    abyBuff[0] &= 0xBF;             /* ←VDET(b6)='0' */
    /* 制御レジスタ２を設定 */
    WriteByte_A8_IIC(RTCREG_CTR2, &abyBuff[0]);

    return bResult;
}

/*****************************************************************************/
/*                            RTCから日時を読み出す                          */
/*****************************************************************************/
/* 引数 : pbyDtime = 日時を格納するアドレス                                     */
/*      :              WW-YYMMDD-hhmmssの順に格納する（各BCD）                 */
/* 戻値 : ReadBCDTime_RTC() =  TRUE : 正常終了                                  */
/*      :                   = FALSE : 異常あり                                  */

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

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    /* 時刻, 曜日, 日付読み出し */
    bResult = ReadPage_A8_IIC(RTCREG_SEC, 7, &abyBuff[0]);
    /* 時刻を逆順に並び替える（不要ビットは削除） */
    pbyDtime += 7;
    nLpcnt = 0;
    do{
        *(--pbyDtime) = (BYTE)(abyBuff[nLpcnt] & MaskBit[nLpcnt]);
    }while(++nLpcnt < 3);
    /* skip WEEK area */
    nLpcnt++;
    /* 日付を逆順に並び替える（不要ビットは削除） */
    do{
```

```
        *(--pbyDtime) = (BYTE)(abyBuff[nLpcnt] & MaskBit[nLpcnt]);
    }while(++nLpcnt < 7);
    /* 曜日 */
    *(--pbyDtime) = (BYTE)(abyBuff[3] & MaskBit[3]);

    return bResult;
}


/*****************************************************************************/
/*                              現在日時を得る                               */
/*****************************************************************************/
/* 引数 :  nTop = 日時の先頭格納位置 (0=曜, 1=年, 2=月, …6=秒)               */
/*       : pBuff = 日時格納アドレス                                          */
/* 戻値 : GetDTbyte() = pBuff+(7-nTop)                                       */
/*       :  (pBuff+0) = 曜                                                   */
/*       :  (pBuff+1) = 年                                                   */
/*       :  (pBuff+2) = 月                                                   */
/*       :  (pBuff+3) = 日                                                   */
/*       :  (pBuff+4) = 時                                                   */
/*       :  (pBuff+5) = 分                                                   */
/*       :  (pBuff+6) = 秒 (各バイナリ)                                      */

BYTE *GetDTbyte(int nTop, BYTE *pBuff)
{
    BYTE    abyDtime[7];
    int     nLpcnt;
    BYTE    byBCD;

    /* RTCから日時読み込み */
    if(!ReadBCDTime_RTC(abyDtime)){
        return pBuff;
    }
    /* バイナリに変換しながら読み込む */
    for(nLpcnt=nTop;nLpcnt < 7;nLpcnt++){
        byBCD = abyDtime[nLpcnt];
        *(pBuff++) = BcdToBin1(byBCD);
    }

    return pBuff;
}

/*****************************************************************************/
/* 引数 : pDtime = 日時格納アドレス                                          */
/* 戻値 : なし、ただし (pDtime) = WW-YYMMDD-hhmmss (各バイナリ)              */

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

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

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


/*****************************************************************************/
/*                              日時を設定する                               */
/*****************************************************************************/
/* 注意 : 曜日は設定できない。                                               */
/* 引数 : pDtime = 設定する日時 (YYMMDD-hhmmss) のアドレス                   */
/* 戻値 : なし                                                               */

void PutDtime(DTIME_T *pDtime)
{
```

```
    /* RTCに日時設定 */
    SetTime_RTC(&pDtime->Array[1]);
}

/*****************************************************************************/
/*                        アラームW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];

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    /* アラームW設定読み出し */
    if(ReadPage_A8_IIC(RTCREG_AW_M, 3, &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));
}

/*****************************************************************************/
/*                        アラームW設定「時分」を設定する                    */
/*****************************************************************************/
/* 引数 : bEnable =  TRUE : アラーム割り込みを許可する                       */
/*      :         = FALSE : アラーム割り込みの設定を現状維持する             */
/*      : pbyHour = アラーム設定「時分」のアドレス                           */
/* 戻値 : なし                                                               */

void PutAlarmTime(BOOL bEnable, BYTE* pbyHour)
{
    BYTE    abyBuff[4];
    BYTE    abyCTRL[2];

    /* 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] = 0x7F;                          /* 曜（全て指定） */
    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    /* WALE保存とWAFG操作のためコントロールレジスタ１～２を読み込む */
    if(ReadPage_A8_IIC(RTCREG_CTR1, 2, &abyCTRL[0]) == FALSE)
        return;
    /* 一時WALEビットを'0'にしてアラームを止める */
    abyCTRL[0] &= 0x7F;                         /* WALE(b7)='0' */
    WriteByte_A8_IIC(RTCREG_CTR1, &abyCTRL[0]);
    /* アラームW設定書き込み */
    WritePage_A8_IIC(RTCREG_AW_M, 3, &abyBuff[0]);
    /* WAFGビットを'0'にする */
    abyCTRL[1] &= 0xFD;                         /* WAFG(b1)='0'  */
    WriteByte_A8_IIC(RTCREG_CTR2, &abyCTRL[1]);
    /* 指示あればWALEビットを'1'にする */
    if(bEnable){
        abyCTRL[0] |= 0x80;                     /* WALE(b7)='1' */
    }
    WriteByte_A8_IIC(RTCREG_CTR1, &abyCTRL[0]);
}

/*****************************************************************************/
/*                  アラームW割り込みの許可・禁止状態を読み込む              */
/*****************************************************************************/
/* 引数 : なし                                                               */
/* 戻値 : GetAlarmInt() == FALSE : アラーム割り込み禁止状態                  */
/*      :               ==  TRUE : アラーム割り込み許可状態                  */

BOOL GetAlarmInt(void)
{
    BYTE    byCTR1;

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
```

```
    /* コントロールレジスタ１を読み込む */
    if(ReadByte_A8_IIC(RTCREG_CTR1, &byCTR1) == FALSE)
        return FALSE;

    return (byCTR1 & 0x80);                     /* WALE(b7)='1' */
}

/*****************************************************************************/
/*                  アラームW割り込みを許可・禁止する                        */
/*****************************************************************************/
/* 引数 : bEnable = FALSE : アラーム割り込み禁止                             */
/*      :         =  TRUE : アラーム割り込み許可                             */
/* 戻値 : SetAlarmInt() == FALSE : 処理中（ビジー、異常）                    */
/*      :               ==  TRUE : 正常受付                                  */

BOOL SetAlarmInt(BOOL bEnable)
{
    BYTE    byCTR1;

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    /* コントロールレジスタ１を読み込む */
    if(ReadByte_A8_IIC(RTCREG_CTR1, &byCTR1) == FALSE)
        return FALSE;
    /* WALEビットを'0'または'1'にする */
    if(bEnable) byCTR1 |= 0x80;                 /* WALE(b7)='1' */
    else        byCTR1 &= 0x7F;
    /* コントロールレジスタ１を書き込む */
    return WriteByte_A8_IIC(RTCREG_CTR1, &byCTR1);
}

/*****************************************************************************/
/*                        時計精度調整量を読み込む                           */
/*****************************************************************************/
/* 引数 : pValue = 調整量（-64～+63）格納アドレス                            */
/* 戻値 : GetTimeAdjustment() == FALSE : 処理中（ビジー、異常）              */
/*      :                     ==  TRUE : 正常受付                            */

BOOL GetTimeAdjustment(short* pValue)
{
    signed char cData;

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    /* 時計精度調整量レジスタを読み込む */
    if(ReadByte_A8_IIC(RTCREG_ADJ, (BYTE*)&cData) == FALSE)
        return FALSE;
    if((cData & 0x40) != 0)
        *pValue = (short)(cData | 0x80);
    else
        *pValue = (short)(cData & 0x3F);

    return TRUE;
}

/*****************************************************************************/
/*                          時計精度調整量を設定する                         */
/*****************************************************************************/
/* 引数 : nValue = 調整量（-64～+63）                                        */
/* 戻値 : SetTimeAdjustment() == FALSE : 処理中（ビジー、異常）              */
/*      :                     ==  TRUE : 正常受付                            */

BOOL SetTimeAdjustment(short nValue)
{
    BYTE    byData;

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    /* 時計精度調整量レジスタに書き込む */
    byData = (BYTE)(nValue & 0x7F);
    return WriteByte_A8_IIC(RTCREG_ADJ, &byData);
}

/*****************************************************************************/
```

```
/*                          割り込み発生フラグを読み込む                                */
/*******************************************************************************/
/* 注意 : WAFGはControl1のWALEが'1'のとき'1'になる。                                */
/*      : CTFGはレベルモードのとき'0'を書き込むことができる。                        */
/* 引数 : なし                                                                 */
/* 戻値 : GetRTCIntFlag() = [76543210]                                         */
/*      :                    ||||||||                                          */
/*      :                    0000||||                                          */
/*      :                        ||||_ b0:CT20:定周期割り込み有効               */
/*      :                        |||__ b1:WALE:アラームW割り込み有効             */
/*      :                        ||___ b2:CTFG:定周期割り込みあり               */
/*      :                        |____ b3:WAFG:アラームW割り込みあり             */

BYTE GetRTCIntFlag(void)
{
    BYTE    abyCTRL[2];
    BYTE    byWrk;

    /* RTCのスレーブアドレスセット */
    SetSlaveAddr_IIC(IICADDR_RTC);
    /* コントロールレジスタ１～２を読み込む */
    if(ReadPage_A8_IIC(RTCREG_CTR1, 2, &abyCTRL[0]) == FALSE)
        return 0;
    /* アラームW発生フラグを'0'にする */
    byWrk = (BYTE)(abyCTRL[1] & 0xF9);  /* ←CTFG(b2)='0'は無効だがやっておく */
    /* 制御レジスタ２を設定 */
    WriteByte_A8_IIC(RTCREG_CTR2, &byWrk);

    byWrk = 0;
    if((abyCTRL[0] & 0x07) != 0){
        byWrk |= 0x01;
    }
    if((abyCTRL[0] & 0x80) != 0){
        byWrk |= 0x02;
    }
    if((abyCTRL[1] & 0x04) != 0){
        byWrk |= 0x04;
    }
    if((abyCTRL[1] & 0x02) != 0){
        byWrk |= 0x08;
    }

    return byWrk;
}

/*******************************************************************************/
/*                            １秒フラグを読み込む                             */
/*******************************************************************************/
/* 注意 : 割り込みマスクの操作はマイコンによって異なる場合がある。                */
/* 引数 : なし                                                                 */
/* 戻値 : ChkSecIntFlg() =  TRUE : １秒フラグON                                */
/*      :                = FALSE : １秒フラグOFF                               */

BOOL ChkSecIntFlg(void)
{
    DWORD   dwPSW;
    BOOL    bFlag;

    /* 割り込みマスク */
    dwPSW = get_psw();
    clrpsw_i();
    /* １秒フラグ読み込み＆クリア */
    bFlag = (bIntFlag_A | bIntFlag_B);
    bIntFlag_A = FALSE;
    bIntFlag_B = FALSE;
    /* 割り込みマスク復帰 */
    set_psw(dwPSW);

    return bFlag;
}

/*******************************************************************************/
/*                              曜日を計算する                                 */
/*******************************************************************************/
```

```
/* 構文 : BYTE aWeek(BYTE year,BYTE month,BYTE day);                          */
/* 引数 :  year = 年 (00～99、バイナリ) ※20XX年代とする                      */
/*      : month = 月 (1～12、バイナリ)                                        */
/*      :   day = 日 (1～31、バイナリ)                                        */
/* 戻値 : 曜日 (0:日、1:月、2:火、3:水、4:木、5:金、6:土)                     */

BYTE aWeek(BYTE year, BYTE month, BYTE day)
{
    int nYear,nDif;

    nYear = 2000 + year;
    if(month <= 2){
        --nYear;
        month += 12;
    }
    nDif = nYear/4 - nYear/100 + nYear/400;
    return (BYTE)((nYear + nDif + (13 * (int)month + 8) / 5 + (int)day) % 7);
}


/* End of File */
```