```
// RIIC0_Master.c
/************************************************************************************/
/************************************************************************************/
/***                                                                              ***/
/***                    IICバス(RIIC0:Inter-IC-Bus)通信制御                       ***/
/***                                                                              ***/
/************************************************************************************/
/************************************************************************************/

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

#include    "RIIC0_Master.h"


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

/* RIIC_ICSR2はRIICのICSR2と同じビット構成のフラグ（Int_RIIC0_ICEEI0で使用）   */
typedef union {
    BYTE BYTE;
    struct {
        BYTE TMOF:1;                        /* b0:TMOF  */
        BYTE AL:1;                          /* b1:AL    */
        BYTE START:1;                       /* b2:START */
        BYTE STOP:1;                        /* b3:STOP  */
        BYTE NACKF:1;                       /* b4:NACKF */
        BYTE RDRF:1;                        /* b5:RDRF  */
        BYTE TEND:1;                        /* b6:TEND  */
        BYTE TDRE:1;                        /* b7:TDRE  */
    } BIT;
} RIIC_ICSR2;
/* RIIC_STATは通信状態を示すステータスフラグ（iic0_stat用） */
typedef union {
    BYTE    Byte;
    struct {
        BYTE    MODE_WR:1;                  /* b0:書き込みモード                    */
        BYTE    MODE_RD:1;                  /* b1:読み込みモード                    */
        BYTE    ERROR:1;                    /* b2:異常発生                          */
        BYTE    NACK:1;                     /* b3:NACK受信                          */
        BYTE    STOP:1;                     /* b4:STOPシーケンス検知                */
        BYTE    START:1;                    /* b5:STARTシーケンス検知（リスタート） */
        BYTE    B6:1;                       /* b6:                                  */
        BYTE    END:1;                      /* b7:シーケンス終了                    */
    } BIT;
} RIIC_STAT;


static const BYTE*  s_pSndDataPtr1;
static const BYTE*  s_pSndDataPtr2;
static BYTE*    s_pRcvBuffPtr;
static WORD     s_wSndSize1;
static WORD     s_wSndSize2;
static WORD     s_wRcvSize;
static WORD     s_wSndCount;
static WORD     s_wRcvCount;
static BYTE     s_bySlaveID;

static volatile RIIC_STAT   iic0_stat;

/************************************************************************************/
/*                        RIIC0：タイマー割り込み処理                               */
/************************************************************************************/
/* ※10mS周期（MCLK0単位）割り込みで呼ぶこと。                                      */
static volatile WORD wTmrEndWait;
static volatile WORD wTmrBusyWait;

void Timer10mS_riic0(void)
{
    if(wTmrEndWait != 0)
        --wTmrEndWait;
    if(wTmrBusyWait != 0)
        --wTmrBusyWait;
}

/************************************************************************************/
/*                      IICインタフェースの初期化(RIIC0)                            */
/************************************************************************************/

void Init_RIIC0(void)
{
    /* ポート１の設定 */
    PORT1.DDR.BIT.B2 = 0;               /* b2:入力ポートに設定(SCL0)  */
    PORT1.ICR.BIT.B2 = 1;               /* b2:入力バッファ有効(SCL0)  */
    PORT1.DDR.BIT.B3 = 0;               /* b3:入力ポートに設定(SDA0)  */
```

```
    PORT1.ICR.BIT.B3 = 1;               /* b3:入力バッファ有効(SDA0)  */

    /* RIIC0のモジュールストップ解除 */
    MSTP(RIIC0) = 0;                    /* SYSTEM.MSTPCRB.BIT.MSTPB21 = 0; */
    /* チャンネル0 (RIIC0)  */
    RIIC0.ICCR1.BIT.ICE    = 0;         /* RIICは機能停止 */
    RIIC0.ICCR1.BIT.IICRST = 1;         /* RIIC/内部リセット */
    RIIC0.ICCR1.BIT.IICRST = 0;         /* RIIC/内部リセット解除 */
    RIIC0.ICSER.BYTE = 0x00;

#if IIC0_SPEED == IIC0_400kbps          // 実測で328kHz
    RIIC0.ICMR1.BYTE = 0x28;            /* BC=0(9bit),BCWP=1,CKS=2(PCLK/4),MTWP=0 */
    RIIC0.ICBRH.BYTE = 0xE7;            /* BRH=7:ビットレートHigh幅 (=400kbps,マスタモード用)  */
    RIIC0.ICBRL.BYTE = 0xF0;            /* BRL=16ビットレートLow幅  (=400kbps)  */
#else   // 100kbpsモード                // 実測で96kHz
    RIIC0.ICMR1.BYTE = 0x38;            /* CKS=3:PCLK/8 */
    RIIC0.ICBRH.BYTE = 0xF8;            /* BRH=24 */
    RIIC0.ICBRL.BYTE = 0xFD;            /* BRL=29 */
#endif
    RIIC0.ICMR2.BYTE = 0x06;            /* TMOS=0,TMOL=1,TMOH=1,SDDL=0,DLCS=0 */
    RIIC0.ICMR3.BYTE = 0x10;            /* NF=0,ACKBR=0,ACKBT=0,ACKWP=1,RDRFS=0,WAIT=0,SMBS=0 */
    RIIC0.ICFER.BYTE = 0x72;            /* TMOE=0,MALE=1,NALE=0,SALE=0,NACKE=1,NFE=1,SCLE=1,FMPE=0 */
    RIIC0.ICIER.BYTE = 0xB8;            /* TMOIE=0,ALIE=0,STIE=0,SPIE=1,NAKIE=1,RIE=1,TEIE=0,TIE=1 */

    /* 割り込み優先順位セット */
    IPR(RIIC0, ICEEI0) = 3;
    IPR(RIIC0, ICRXI0) = 3;
    IPR(RIIC0, ICTXI0) = 3;
    IPR(RIIC0, ICTEI0) = 3;
    /* 割り込み要求ステータスフラグクリア */
    IR(RIIC0, ICEEI0) = 0;
    IR(RIIC0, ICRXI0) = 0;
    IR(RIIC0, ICTXI0) = 0;
    IR(RIIC0, ICTEI0) = 0;
    /* 全割り込み許可 */
    IEN(RIIC0, ICEEI0) = 1;             /* 通信エラー/イベント発生（レベル）  */
    IEN(RIIC0, ICRXI0) = 1;             /* 受信データフル          （エッジ）  */
    IEN(RIIC0, ICTXI0) = 1;             /* 送信データエンプティ    （エッジ）  */
    IEN(RIIC0, ICTEI0) = 1;             /* 送信終了                （レベル）  */

    /* 作業用変数を初期化 */
    s_wSndSize1 = 0;
    s_wSndSize2 = 0;
    s_wRcvSize  = 0;
    s_wSndCount = 0;
    s_wRcvCount = 0;
    s_bySlaveID  = 0;
    iic0_stat.Byte = 0;

    /* RIIC0の動作許可 */
    RIIC0.ICCR1.BIT.ICE = 1;
}

/*****************************************************************************/
/*                    受信／イベント発生割り込み処理(RIIC0)                  */
/*****************************************************************************/

/* 通信エラー/イベント発生割り込み(RIIC0)  ***********************************/

#pragma interrupt (Int_RIIC0_ICEEI0(enable))
void Int_RIIC0_ICEEI0(void)
{
    RIIC_ICSR2  sICSR2;

    /* I2Cバスステータスレジスタ2読み込み  */
    sICSR2.BYTE = RIIC0.ICSR2.BYTE;
    /* TMOF,AL,START,STOP,NACKFフラグをクリア */
    RIIC0.ICSR2.BYTE = 0xE0;
    /* 割り込み許可ビットのみ抽出  */
    sICSR2.BYTE = (BYTE)(sICSR2.BYTE & RIIC0.ICIER.BYTE);

    /* スタートコンディション検出（＝リスタートコンディション検出）  */
    if(sICSR2.BIT.START == 1){
        /* スタートコンディション検出禁止 */
        RIIC0.ICIER.BIT.STIE = 0;
        /* 読み込みモード指定(R/W=1)でスレーブアドレス送信 */
        RIIC0.ICDRT = (BYTE)(s_bySlaveID | 0x01);
        /* IICステータスセット */
        iic0_stat.BIT.START = 1;
    }

    /* ストップコンディション検出 */
    if(sICSR2.BIT.STOP == 1){
        /* ストップコンディション検出以外の割り込み禁止 */
        /* TIE（送信データエンプティ）、RIE（受信データフル）、NAKIE（NACK受信）、SPIE（ストップコンディション検出）
割り込み許可 */
        RIIC0.ICIER.BYTE = 0xB8;
```

```
        /* 書き込みモードで送信未完了があれば異常とする */
        if(iic0_stat.BIT.MODE_WR != 0){
            if((s_wSndSize1 != 0) || (s_wSndSize2 != 0)){
                iic0_stat.BIT.ERROR = 1;
            }
        }
        /* 読み込みモードで受信途中があれば異常とする */
        else if(iic0_stat.BIT.MODE_RD != 0){
            if(s_wRcvSize != 0){
                iic0_stat.BIT.ERROR = 1;
            }
        }
        /* IICステータスセット */
        iic0_stat.BIT.STOP = 1;
        iic0_stat.BIT.END  = 1;
    }

    /* NACK検出フラグ（送信モードで動作） */
    if(sICSR2.BIT.NACKF == 1){
        /* NACK受信割り込み禁止 */
        RIIC0.ICIER.BIT.NAKIE = 0;
        /* ストップコンディション発行 */
        RIIC0.ICSR2.BIT.STOP = 0;
        RIIC0.ICCR2.BIT.SP   = 1;
        /* IICステータスセット */
        iic0_stat.BIT.NACK = 1;
        iic0_stat.BIT.END  = 1;
    }

    /* タイムアウト検出（未使用） */
    if(sICSR2.BIT.TMOF == 1){
        /* タイムアウト検出禁止 */
        RIIC0.ICIER.BIT.TMOIE = 0;
    }

    /* アービトレーションロスト発生（未使用） */
    if(sICSR2.BIT.AL == 1){
        /* アービトレーションロスト検出禁止 */
        RIIC0.ICIER.BIT.ALIE = 0;
    }
}

/*  受信データフル割り込み(RIIC0)  ******************************************/

#pragma interrupt (Int_RIIC0_ICRXI0(enable))
void Int_RIIC0_ICRXI0(void)
{
    /* 受信数をカウント */
    s_wRcvCount++;
    /* 最初の受信は特殊な処理 */
    if(s_wRcvCount == 1){
        /* １バイトのみの場合 */
        if(s_wRcvSize == 1){
            /* WAITありを設定 */
            RIIC0.ICMR3.BIT.WAIT = 1;
            /* NAK送信を設定 */
            RIIC0.ICMR3.BIT.ACKBT = 1;
        }
        /* ダミーリード */
        RIIC0.ICDRR;
    }
    /* 最終－１番目の受信 */
    else if(s_wRcvSize == 2){
        /* WAITありを設定 */
        RIIC0.ICMR3.BIT.WAIT = 1;
        /* NAK送信を設定 */
        RIIC0.ICMR3.BIT.ACKBT = 1;
        /* 受信データ格納 */
        *(s_pRcvBuffPtr++) = RIIC0.ICDRR;
        s_wRcvSize--;
    }
    /* 最終の受信 */
    else if(s_wRcvSize == 1){
        /* ストップコンディション発行 */
        RIIC0.ICSR2.BIT.STOP = 0;
        RIIC0.ICCR2.BIT.SP   = 1;
        /* 次回のためACKビットをセット */
        RIIC0.ICMR3.BIT.ACKBT = 0;
        /* 受信データ格納 */
        *(s_pRcvBuffPtr++) = RIIC0.ICDRR;
        s_wRcvSize--;
        /* 次回のためWAITなしを設定 */
        RIIC0.ICMR3.BIT.WAIT = 0;
    }
    /* 上記以外で受信データ格納 */
    else{
        /* 受信データ格納 */
```

```
        *(s_pRcvBuffPtr++) = RIIC0.ICDRR;
        s_wRcvSize--;
    }
}

/*  送信データエンプティ割り込み(RIIC0)  ************************************/

#pragma interrupt (Int_RIIC0_ICTXI0(enable))
void Int_RIIC0_ICTXI0(void)
{
    /* 送信数をカウント */
    s_wSndCount++;
    /* 最初にスレーブアドレスを送信 */
    if(s_wSndCount == 1){
        /* 書き込みモード指定(R/W=0)でスレーブアドレス送信 */
        RIIC0.ICDRT = (BYTE)(s_bySlaveID & 0xFE);
        return;
    }
    /* 書き込みモードの場合 */
    if(iic0_stat.BIT.MODE_WR != 0){
        /* １ブロック目の送信データがある場合 */
        if(s_wSndSize1 != 0){
            /* データ送信 */
            RIIC0.ICDRT = *(s_pSndDataPtr1++);
            s_wSndSize1--;
        }
        /* ２ブロック目の送信データがある場合 */
        else if(s_wSndSize2 != 0){
            /* データ送信 */
            RIIC0.ICDRT = *(s_pSndDataPtr2++);
            s_wSndSize2--;
        }
        /* すべての送信データを送信した場合 */
        else{
            /* 送信終了割り込み許可 */
            RIIC0.ICIER.BIT.TEIE = 1;
        }
    }
    /* 読み込みモードの場合 */
    else if(iic0_stat.BIT.MODE_RD != 0){
        /* １ブロック目の送信データがある場合 */
        if(s_wSndSize1 != 0){
            /* データ送信 */
            RIIC0.ICDRT = *(s_pSndDataPtr1++);
            s_wSndSize1--;
        }
        /* すべての送信データを送信した場合 */
        else{
            /* 送信終了割り込み許可 */
            RIIC0.ICIER.BIT.TEIE = 1;
        }
    }
    /* 書き込みでも読み込みでもない場合 */
    else{
        /* ストップコンディション発行 */
        RIIC0.ICSR2.BIT.STOP = 0;
        RIIC0.ICCR2.BIT.SP = 1;
        /* IICステータスセット */
        iic0_stat.BIT.ERROR = 1;
    }
}

/*  送信終了割り込み(RIIC0)  ************************************************/
/* ※シフトレジスタ(ICDRS)から送信が完了したことを示す。                       */

#pragma interrupt (Int_RIIC0_ICTEI0(enable))
void Int_RIIC0_ICTEI0(void)
{
    /* 送信終了フラグをクリア */
    RIIC0.ICSR2.BIT.TEND = 0;

    /* 書き込みモードの場合 */
    if(iic0_stat.BIT.MODE_WR != 0){
        /* ストップコンディション発行 */
        RIIC0.ICSR2.BIT.STOP = 0;
        RIIC0.ICCR2.BIT.SP   = 1;
    }
    /* 読み込みモードの場合 */
    else if(iic0_stat.BIT.MODE_RD != 0){
        /* スタートコンディション検出割り込み許可 */
        RIIC0.ICIER.BIT.STIE  = 1;
        /* リスタートコンディション発行 */
        RIIC0.ICSR2.BIT.START = 0;
        RIIC0.ICCR2.BIT.RS = 1;
    }
    /* 書き込みでも読み込みでもない場合 */
    else{
```

```
        /* ストップコンディション発行 */
        RIIC0.ICSR2.BIT.STOP = 0;
        RIIC0.ICCR2.BIT.SP = 1;
        /* IICステータスセット */
        iic0_stat.BIT.ERROR = 1;
    }
    /* 送信終了割り込み禁止 */
    RIIC0.ICIER.BIT.TEIE = 0;

}

/******************************************************************************/
/*                              RIIC通信スタート                              */
/******************************************************************************/

static BOOL Start_RIIC0(void)
{
    /* バスの開放を100mSまで待つ */
    wTmrBusyWait = 10;
    while(RIIC0.ICCR2.BIT.BBSY != 0){   /*BUSYなら待つ */
        if(wTmrBusyWait == 0){
            return FALSE;
        }
    }

    /* IICステータス初期化 */
    iic0_stat.Byte = 0x00;
    if(s_wRcvSize == 0){
        iic0_stat.BIT.MODE_WR = 1;
    }
    else{
        iic0_stat.BIT.MODE_RD = 1;
    }
    /* 送信・受信カウンタをクリア */
    s_wSndCount = 0;
    s_wRcvCount = 0;
    /* スタートコンディション発行 */
    RIIC0.ICSR2.BIT.START = 0;
    RIIC0.ICCR2.BIT.ST    = 1;

    return TRUE;
}

static void Stop_RIIC0(void)
{
    /* IICステータス初期化 */
    iic0_stat.Byte = 0;
    /* ストップコンディション発行 */
    RIIC0.ICSR2.BIT.STOP = 0;
    RIIC0.ICCR2.BIT.SP   = 1;
}

static void Reset_RIIC0(void)
{
    /* ICMR1.BC[2:0]ビット、ICSR1、ICSR2、ICDRSレジスタおよび内部状態をリセット */
    RIIC0.ICCR1.BIT.IICRST = 1;
    RIIC0.ICCR1.BIT.IICRST = 0;
    /* I2Cバスモードレジスタ1を再セット  */
#if IIC0_SPEED == IIC0_400kbps
    RIIC0.ICMR1.BYTE = 0x28;        /* 基準クロック：PCLK/4、ビットカウンタ：９ビット */
    RIIC0.ICBRH.BYTE = 0xE7;
    RIIC0.ICBRL.BYTE = 0xF0;
#else   // 100kbpsモード
    RIIC0.ICMR1.BYTE = 0x38;        /* 基準クロック：PCLK/8、ビットカウンタ：９ビット */
    RIIC0.ICBRH.BYTE = 0xF8;
    RIIC0.ICBRL.BYTE = 0xFD;
#endif
    RIIC0.ICCR2.BYTE = 0x00;
}


/******************************************************************************/
/*                            スレーブアドレスの設定                          */
/******************************************************************************/
/* IN  : bySlaveAddr = 相手のスレーブアドレス                                 */
/* OUT : なし                                                                 */

void SetSlaveAddr_RIIC0(BYTE bySlaveAddr)
{
    /* （相手）スレーブアドレス保存 */
    s_bySlaveID = bySlaveAddr;
}


/******************************************************************************/
/*                          送信してレスポンスを受信する                      */
/******************************************************************************/
```

```
/* スレーブアドレスはあらかじめ設定しておき、１バイト目に送信される。          */
/* 【読み／書きモードとリスタートコンディションの有無】                        */
/*  .wSndDataSz?|.wRcvDataSz |                                               */
/*  ------------|------------|---------------|---------------------------- */
/*    == 0      |  == 0      | 書き込みのみ  | リスタートコンディションなし  */
/*    == 0      |  != 0      | 読み込みのみ  | リスタートコンディションなし  */
/*    != 0      |  == 0      | 書き込みのみ  | リスタートコンディションなし  */
/*    != 0      |  != 0      | 書き→読み込み | リスタートコンディションあり  */


int TranceiveData_RIIC0(RIIC_TXRX* TxRxParam)
{
    /* 送信パラメータセット */
    s_pSndDataPtr1 = TxRxParam->pSndData1;
    s_wSndSize1    = TxRxParam->wSndDataSz1;
    s_pSndDataPtr2 = TxRxParam->pSndData2;
    s_wSndSize2    = TxRxParam->wSndDataSz2;
    /* 受信パラメータセット */
    s_pRcvBuffPtr = TxRxParam->pRcvBuff;
    s_wRcvSize    = TxRxParam->wRcvDataSz;


    /* RIIC通信スタート */
    if(Start_RIIC0() == FALSE){         /* バスが空かない（BUSY状態） */
        return RIIC0_ERR_BUSY;
    }

    /* 通信終了を100mSまで待つ */
    wTmrEndWait = 10;
    while(iic0_stat.BIT.END == 0){
        if(wTmrEndWait == 0){
            /* RIIC通信ストップ */
            Stop_RIIC0();
            /* RIICリセット */
            Reset_RIIC0();
            return RIIC0_ERR_TMOUT;
        }
    }
    if(iic0_stat.BIT.NACK != 0){
        return RIIC0_ERR_NACK;
    }
    if(iic0_stat.BIT.ERROR != 0){
        return RIIC0_ERR_OTHER;
    }

    return RIIC0_ERR_OK;
}

/*****************************************************************************/
/*****************************************************************************/
/***                                                                       ***/
/***                        IICデバイス通信支援関数                        ***/
/***                                                                       ***/
/*****************************************************************************/
/*****************************************************************************/
/* 注意 : TRYMAX_IICはEEPROMの書き込み中(=BUSY)を待つ時間をつくるリトライ数。 */
/*      : その時間(Write Cycle Time)はEEPROMによって異なるが、一律に10mSとみ  */
/*      : なしている。また、余裕を持って２倍程度(20mS)の値になっている。      */
/*      : EEPROM以外のデバイスには関係ないが、簡略化のため共通に作用する。    */
#if IIC0_SPEED == IIC0_100kbps
#define TRYMAX_IIC  (60)        /* 100Kbpsのとき */
#else
#define TRYMAX_IIC  (230)       /* 400Kbpsのとき */
#endif

/*****************************************************************************/
/*                              １バイトデータリード                         */
/*****************************************************************************/
/* １バイトのアドレス指定（RTC用） ******************************************/

BOOL ReadByte_A8_IIC(BYTE byRdAddr, BYTE *pRxBuff)
{
    RIIC_TXRX RIICParam;
    int     nTryCntr;

    /* 送受信パラメータセット */
    RIICParam.wSndDataSz1 = 1;
    RIICParam.pSndData1   = (const BYTE*)&byRdAddr;
    RIICParam.wSndDataSz2 = 0;
    RIICParam.pSndData2   = NULL;
    RIICParam.wRcvDataSz  = 1;
    RIICParam.pRcvBuff    = pRxBuff;

    nTryCntr = 0;
    do{
        if(TranceiveData_RIIC0(&RIICParam) == RIIC0_ERR_OK){
            return TRUE;
```

```
        }
    }while(++nTryCntr < TRYMAX_IIC);

    return FALSE;
}

/* ２バイトのアドレス指定（EEPROM用） ****************************************/

BOOL ReadByte_A16_IIC(WORD wRdAddr, BYTE *pRxBuff)
{
    RIIC_TXRX RIICParam;
    int     nTryCntr;
    BYTE    abyTxBuff[2];

    /* 16bitアドレスを8bitずつにする */
    abyTxBuff[0] = (BYTE)(wRdAddr >> 8);
    abyTxBuff[1] = (BYTE)wRdAddr;

    /* 送受信パラメータセット */
    RIICParam.wSndDataSz1 = 2;
    RIICParam.pSndData1   = (const BYTE*)abyTxBuff;
    RIICParam.wSndDataSz2 = 0;
    RIICParam.pSndData2   = NULL;
    RIICParam.wRcvDataSz  = 1;
    RIICParam.pRcvBuff    = pRxBuff;

    /* 読み込みシーケンス */
    nTryCntr = 0;
    do{
        if(TranceiveData_RIIC0(&RIICParam) == RIIC0_ERR_OK){
            return TRUE;
        }
    }while(++nTryCntr < TRYMAX_IIC);

    return FALSE;
}

/*****************************************************************************/
/*                              ページデータリード                           */
/*****************************************************************************/
/* １バイトのアドレス指定（RTC用） ********************************************/

BOOL ReadPage_A8_IIC(BYTE byRdAddr, WORD wSize, BYTE *pRxBuff)
{
    RIIC_TXRX RIICParam;
    int     nTryCntr;

    /* 送受信パラメータセット */
    RIICParam.wSndDataSz1 = 1;
    RIICParam.pSndData1   = (const BYTE*)&byRdAddr;
    RIICParam.wSndDataSz2 = 0;
    RIICParam.pSndData2   = NULL;
    RIICParam.wRcvDataSz  = wSize;
    RIICParam.pRcvBuff    = pRxBuff;

    nTryCntr = 0;
    do{
        if(TranceiveData_RIIC0(&RIICParam) == RIIC0_ERR_OK){
            return TRUE;
        }
    }while(++nTryCntr < TRYMAX_IIC);

    return FALSE;
}

/* ２バイトのアドレス指定（EEPROM用） ****************************************/

BOOL ReadPage_A16_IIC(WORD wRdAddr, WORD wSize, BYTE *pRxBuff)
{
    RIIC_TXRX RIICParam;
    int     nTryCntr;
    BYTE    abyTxBuff[2];

    /* 16bitアドレスを8bitずつにする */
    abyTxBuff[0] = (BYTE)(wRdAddr >> 8);
    abyTxBuff[1] = (BYTE)wRdAddr;

    /* 送受信パラメータセット */
    RIICParam.wSndDataSz1 = 2;
    RIICParam.pSndData1   = (const BYTE*)abyTxBuff;
    RIICParam.wSndDataSz2 = 0;
    RIICParam.pSndData2   = NULL;
    RIICParam.wRcvDataSz  = wSize;
    RIICParam.pRcvBuff    = pRxBuff;

    /* 読み込みシーケンス */
    nTryCntr = 0;
```

```
    do{
        if(TranceiveData_RIIC0(&RIICParam) == RIIC0_ERR_OK){
            return TRUE;
        }
    }while(++nTryCntr < TRYMAX_IIC);

    return FALSE;
}

/*****************************************************************************/
/*                              １バイトデータライト                         */
/*****************************************************************************/
/* １バイトのアドレス指定（RTC用） ***********************************************/

BOOL WriteByte_A8_IIC(BYTE byWrAddr, const BYTE *pTxData)
{
    RIIC_TXRX RIICParam;
    int     nTryCntr;

    /* 送受信パラメータセット */
    RIICParam.wSndDataSz1 = 1;
    RIICParam.pSndData1   = (const BYTE*)&byWrAddr;
    RIICParam.wSndDataSz2 = 1;
    RIICParam.pSndData2   = pTxData;
    RIICParam.wRcvDataSz  = 0;
    RIICParam.pRcvBuff    = NULL;

    nTryCntr = 0;
    do{
        if(TranceiveData_RIIC0(&RIICParam) == RIIC0_ERR_OK){
            return TRUE;
        }
    }while(++nTryCntr < TRYMAX_IIC);

    return FALSE;
}

/* ２バイトのアドレス指定（EEPROM用） ********************************************/

BOOL WriteByte_A16_IIC(WORD wWrAddr, const BYTE *pTxData)
{
    RIIC_TXRX RIICParam;
    int     nTryCntr;
    BYTE    abyTxBuff[2];

    /* 16bitアドレスを8bitずつにする */
    abyTxBuff[0] = (BYTE)(wWrAddr >> 8);
    abyTxBuff[1] = (BYTE)wWrAddr;

    /* 送受信パラメータセット */
    RIICParam.wSndDataSz1 = 2;
    RIICParam.pSndData1   = (const BYTE*)abyTxBuff;
    RIICParam.wSndDataSz2 = 1;
    RIICParam.pSndData2   = pTxData;
    RIICParam.wRcvDataSz  = 0;
    RIICParam.pRcvBuff    = NULL;

    /* 書き込みシーケンス開始 */
    nTryCntr = 0;
    do{
        if(TranceiveData_RIIC0(&RIICParam) == RIIC0_ERR_OK){
            return TRUE;
        }
    }while(++nTryCntr < TRYMAX_IIC);

    return FALSE;
}


/*****************************************************************************/
/*                              ページデータライト                           */
/*****************************************************************************/
/* １バイトのアドレス指定（RTC用） ***********************************************/

BOOL WritePage_A8_IIC(BYTE byWrAddr, WORD wSize, const BYTE *pTxData)
{
    RIIC_TXRX RIICParam;
    int     nTryCntr;

    /* 送受信パラメータセット */
    RIICParam.wSndDataSz1 = 1;
    RIICParam.pSndData1   = (const BYTE*)&byWrAddr;
    RIICParam.wSndDataSz2 = wSize;
    RIICParam.pSndData2   = pTxData;
    RIICParam.wRcvDataSz  = 0;
    RIICParam.pRcvBuff    = NULL;
```

```
    nTryCntr = 0;
    do{
        if(TranceiveData_RIIC0(&RIICParam) == RIIC0_ERR_OK){
            return TRUE;
        }
    }while(++nTryCntr < TRYMAX_IIC);

    return FALSE;
}

/* ２バイトのアドレス指定（EEPROM用） ****************************************/

BOOL WritePage_A16_IIC(WORD wWrAddr, WORD wSize, const BYTE *pTxData)
{
    RIIC_TXRX_RIICParam;
    int     nTryCntr;
    BYTE    abyTxBuff[2];

    /* 16bitアドレスを8bitずつにする */
    abyTxBuff[0] = (BYTE)(wWrAddr >> 8);
    abyTxBuff[1] = (BYTE)wWrAddr;
    /* 送受信パラメータセット */
    RIICParam.wSndDataSz1 = 2;
    RIICParam.pSndData1   = (const BYTE*)abyTxBuff;
    RIICParam.wSndDataSz2 = wSize;
    RIICParam.pSndData2   = pTxData;
    RIICParam.wRcvDataSz  = 0;
    RIICParam.pRcvBuff    = NULL;

    /* 書き込みシーケンス開始 */
    nTryCntr = 0;
    do{
        if(TranceiveData_RIIC0(&RIICParam) == RIIC0_ERR_OK){
            return TRUE;
        }
    }while(++nTryCntr < TRYMAX_IIC);

    return FALSE;
}

/* End of File */
```