```
// RSPI1_Mode3.c
/************************************************************************************/
/************************************************************************************/
/***                                                                              ***/
/***                シリアルペリフェラルインタフェースRSPI1                       ***/
/***                                                                              ***/
/************************************************************************************/
/************************************************************************************/
/* 注意 : (1)PCLK=48MHzであること。                                                 */
/*      : (2)半二重通信専用（送信中に受信したデータは破棄する）。                   */

#include    <machine.h>
#include    <stdlib.h>
#include    "typedefine.h"
#include    "iodefine.h"
#include    "RSPI1_Mode3.h"

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

static volatile BYTE    st_byRecvComple;    /* 送受信完了フラグ（正常時のみ） */
static volatile BYTE    st_byRSPI1_OVRF;    /* ステータスレジスタSPSRのOVRFビット（異常時） */

static DWORD    st_dwSendPtr;           /* 送信済みバイト数   （ダミー含む） */
static DWORD    st_dwSendSize;          /* 送信データバイト数（正味の送信データ、ダミー含まず） */
static BYTE*    st_pSendBuff;           /* 送信データアドレス */
static DWORD    st_dwRecvPtr;           /* 受信済みバイト数   （送信中の受信含む） */
static DWORD    st_dwRecvPtrR;          /*    〃       〃     （正味の受信データ） */
static DWORD    st_dwRecvSize;          /* 受信データバイト数（正味の受信データ、送信中データ含まず） */
static BYTE*    st_pRecvBuff;           /* 受信データアドレス */
static DWORD    st_dwTrcvSize;          /* 送信＋受信バイト数 */

/************************************************************************************/
/*                        RSPI1：タイマー割り込み処理                               */
/************************************************************************************/
/* ※10mS周期割り込みで呼ぶこと。                                                   */
static volatile WORD wTmrTxRxWait;

void Timer10mS_spi1(void)
{
    if(wTmrTxRxWait != 0)
        --wTmrTxRxWait;
}

/************************************************************************************/
/*                      SPIインタフェース(RSPI1)初期化                              */
/************************************************************************************/

static void Init_RSPI1(void)
{
    /* RSPI1のモジュールストップ解除 */
    MSTP(RSPI1) = 0;                    /* SYSTEM.MSTPCRB.BIT.MSTPB16 = 0; */
    /* RSPI1の動作停止 */
    RSPI1.SPCR.BYTE = 0;                /* SPCR.SPE=0で動作停止 */
    /* RSPI1の初期化 */
    RSPI1.SPPCR.BYTE = 0x00;            /* SPLP=0,SPLP2=0,SPOM=0,MOIFV=1,MOIFE=1 */
    RSPI1.SPBR.BYTE = 2;                /* RSPIビットレートレジスタ(=n)  (SPCMDn.BRDV=0で 8.00Mbps)     */
//  RSPI1.SPBR.BYTE = 3;                /* RSPIビットレートレジスタ(=n)  (SPCMDn.BRDV=0で 6.00Mbps)     */
                                        /* ビットレート= PCLK/(2x(n+1)x2^N) N=SPCMDn.BRDV */

    RSPI1.SPDCR.BYTE = 0x00;            /* SPFC=0,SLSEL=0,SPRDTD=0,SPLW=0 */

    RSPI1.SPCKD.BYTE = 0x00;            /* SCKDL=0 */

    RSPI1.SSLND.BYTE = 0x00;            /* SLNDL=0 */

    RSPI1.SPND.BYTE  = 0x00;            /* SPNDL=0 */

    RSPI1.SPCR2.BYTE = 0x00;            /* SPPE=0,SPOE=0,SPIIE=0,PTE=0 */

    RSPI1.SSLP.BYTE  = 0x00;            /* SSL0P=0,SSL1P=0,SSL2P=0,SSL3P=0 */

    RSPI1.SPSR.BYTE  = 0xA0;            /* OVRF=0,IDLNF=0,MODF=0,PERF=0,SPTEF=1,SPRF=1 */

    RSPI1.SPSCR.BYTE = 0x00;            /* SPSLN=0:シーケンス長1（0→0→…） */

//  RSPI1.SPSSR.BYTE = 0x00;            /* SPCP=0,SPECM=0 */

    RSPI1.SPCMD0.WORD = 0xE783;         /* RSPIコマンドレジスタ0（モード３, データ長８, MSBファースト, バースト転送,
SSL0選択） */
                                        /* CPHA=1,CPOL=1,BRDV=0,SSLA=0,SSLKP=1,SPB=7,LSBF=0,SPNDEN=1,SLNDEN=1,SCKDEN=
1 */
    RSPI1.SPCMD1.WORD = 0x070D;         /* RSPIコマンドレジスタ1            */
    RSPI1.SPCMD2.WORD = 0x070D;         /* RSPIコマンドレジスタ2            */
    RSPI1.SPCMD3.WORD = 0x070D;         /* RSPIコマンドレジスタ3            */
    RSPI1.SPCMD4.WORD = 0x070D;         /* RSPIコマンドレジスタ4            */
```

```
    RSPI1.SPCMD5.WORD = 0x070D;         /* RSPIコマンドレジスタ5            */
    RSPI1.SPCMD6.WORD = 0x070D;         /* RSPIコマンドレジスタ6            */
    RSPI1.SPCMD7.WORD = 0x070D;         /* RSPIコマンドレジスタ7            */

    /* RSPI1のポート設定 */
    PORTE.DDR.BIT.B7 = 0;
    PORTE.ICR.BIT.B7 = 1;               /* PE7(MISOB)の入力バッファ有効     */
    PORTE.PCR.BIT.B7 = 1;               /* PE7(MISOB)のプルアップ抵抗有効   */
#if SPI_SLAVE_MAX == 4
    IOPORT.PFHSPI.BYTE = 0xFF;          /* RSPIS=1,RSPCKE=1,MOSIE=1,MISOE=1,SSL0E=1,SSL1E=1,SSL2E=1,SSL3E=1 (SSLB0～S
SLB3有効)  */
#elif SPI_SLAVE_MAX == 3;
    IOPORT.PFHSPI.BYTE = 0x7F;          /* RSPIS=1,RSPCKE=1,MOSIE=1,MISOE=1,SSL0E=1,SSL1E=1,SSL2E=1,SSL3E=0 (SSLB0～S
SLB2有効)  */
#elif SPI_SLAVE_MAX == 2;
    IOPORT.PFHSPI.BYTE = 0x3F;          /* RSPIS=1,RSPCKE=1,MOSIE=1,MISOE=1,SSL0E=1,SSL1E=1,SSL2E=0,SSL3E=0 (SSLB0～S
SLB1有効)  */
#else
    IOPORT.PFHSPI.BYTE = 0x1F;          /* RSPIS=1,RSPCKE=1,MOSIE=1,MISOE=1,SSL0E=1,SSL1E=0,SSL2E=0,SSL3E=0 (SSLB0の
み有効)     */
#endif
    /* RSPI1制御レジスタ設定 */
    RSPI1.SPCR.BYTE  = 0xB8;            /* SPMS=0,TXMD=0,MODFEN=0,MSTR=1,SPEIE=1,SPTIE=1,SPE=0,SPRIE=1 */
    RSPI1.SPCR.BYTE;                    /* SPCR空読み */

    /* RSPI1割り込み優先順位セット */
    IPR(RSPI1, ) = 4;                   /* SPEI1,SPRI1,SPTI1,SPII1共通 */
    /* RSPI1割り込み要求ステータスフラグクリア */
    IR(RSPI1, SPTI1) = 0;
    IR(RSPI1, SPRI1) = 0;
    IR(RSPI1, SPEI1) = 0;
    /* RSPI1割り込み禁止 */
    IEN(RSPI1, SPTI1) = 0;
    IEN(RSPI1, SPRI1) = 0;
    /* RSPI1エラー割り込み許可 */
    IEN(RSPI1, SPEI1) = 1;
}

/*****************************************************************************/
/*                          SPI通信開始／停止                                */
/*****************************************************************************/
/* nSlaveNum = スレーブ番号 (SPI_SLAVE0～SPI_SLAVE_MAX-1)                    */
/*             SPI_DISABLE指定で通信停止                                     */

void Enable_SPI1(int nSlaveNum)
{
    /* 有効なスレーブ番号の場合 (＝RSPI通信開始)  */
    if((SPI_SLAVE0 <= nSlaveNum) && (nSlaveNum < SPI_SLAVE_MAX)){
        /* スレーブ選択信号セット */
        RSPI1.SPCMD0.BIT.SSLA = (unsigned short)nSlaveNum;
        /* ダミーリード */
        RSPI1.SPDR.WORD.H;
        /* RSPIステータスクリア */
        RSPI1.SPSR.BYTE  = 0xA0;
        /* SPI動作許可 */
        RSPI1.SPCR.BIT.SPE   = 1;
    }
    /* 無効なスレーブ番号の場合 (＝RSPI通信停止)  */
    else{
        /* SPI動作停止 */
        RSPI1.SPCR.BIT.SPE   = 0;
        /* 送信・受信割り込み禁止 */
        IEN(RSPI1, SPTI1) = 0;
        IEN(RSPI1, SPRI1) = 0;
    }
}

/*****************************************************************************/
/*                    送信／受信／エラー割り込み処理(RSPI1)                  */
/*****************************************************************************/

/* 送信割り込み ***************************************************************/

#pragma interrupt (Int_RSPI1_SPTI1)
void Int_RSPI1_SPTI1(void)
{
    /* 指定バイト数まで送信する */
    if(st_dwSendPtr < st_dwSendSize){
        RSPI1.SPDR.WORD.H = (WORD)*(st_pSendBuff + st_dwSendPtr);
    }
    /* 指定バイト数送信した後はダミー送信 */
    else{
        RSPI1.SPDR.WORD.H = 0x0000; /* ダミー値送信 */
    }
    /* 送信＋受信バイト数まで送信したら終了 */
    st_dwSendPtr++;
    if(st_dwSendPtr >= st_dwTrcvSize){
```

```
        /* RSPI1送信割り込み禁止 */
        IEN(RSPI1, SPTI1) = 0;
    }
}

/* 受信割り込み ***************************************************************/

#pragma interrupt (Int_RSPI1_SPRI1)
void Int_RSPI1_SPRI1(void)
{
    WORD    wRxData;

    /* 受信 */
    wRxData = RSPI1.SPDR.WORD.H;
    st_dwRecvPtr++;
    /* 半二重なので送信完了までは無視 */
    if(st_dwRecvPtr <= st_dwSendSize){
        /* 受信データがない場合 */
        if(st_dwRecvPtr >= st_dwTrcvSize){
            st_byRecvComple = 1;
            /* RSPI1割り込み禁止 */
            IEN(RSPI1, SPRI1) = 0;
            IEN(RSPI1, SPTI1) = 0;
        }
        return;
    }
    /* 受信データをバッファに格納 */
    *(st_pRecvBuff + st_dwRecvPtrR) = (BYTE)wRxData;
    st_dwRecvPtrR++;
    /* 指定バイト数受信したら終了 */
    if(st_dwRecvPtrR >= st_dwRecvSize){
        st_byRecvComple = 1;
        /* RSPI1割り込み禁止 */
        IEN(RSPI1, SPRI1) = 0;
        IEN(RSPI1, SPTI1) = 0;
    }
}

/* エラー割り込み *************************************************************/

#pragma interrupt (Int_RSPI1_SPEI1)
void Int_RSPI1_SPEI1(void)
{
    st_byRSPI1_OVRF = RSPI1.SPSR.BYTE;
    RSPI1.SPSR.BYTE = 0xA0;

    /* オーバランエラーフラグのみ残す（他は関係ない） */
    st_byRSPI1_OVRF &= 0x01;
    if(st_byRSPI1_OVRF != 0){   /* ※関数Tranceive_SPI1で異常処理される。 */
        /* ダミーリード */
        RSPI1.SPDR.WORD.H;
        /* RSPI1割り込み禁止 */
        IEN(RSPI1, SPRI1) = 0;
        IEN(RSPI1, SPTI1) = 0;
    }
}



/*****************************************************************************/
/*                          SPIブロック送受信                                */
/*****************************************************************************/

BOOL Tranceive_SPI1(WORD wSndSize, WORD wRcvSize, const BYTE* pSndData, BYTE* pRcvBuff)
{
    /* フラグ初期化 */
    st_byRecvComple = 0;                /* 送受信完了フラグ */
    st_byRSPI1_OVRF = 0;                /* ステータスレジスタSPSR */
    st_dwSendSize = (DWORD)wSndSize;    /* 指定の送信バイト数 */
    st_pSendBuff  = (BYTE*)pSndData;    /* 送信データポインタ */
    st_dwRecvPtr  = 0L;                 /* 全受信済みバイト数   */
    st_dwRecvPtrR = 0L;                 /* 正味の受信済みバイト数 */
    st_dwRecvSize = (DWORD)wRcvSize;    /* 指定の受信データバイト数 */
    st_pRecvBuff  = pRcvBuff;           /* 受信データアドレス */
    st_dwTrcvSize = (DWORD)(wSndSize + wRcvSize);   /* 送信＋受信バイト数 */
    /* RSPI1割り込み許可 */
    IEN(RSPI1, SPTI1) = 1;
    IEN(RSPI1, SPRI1) = 1;

    /* 送受信完了を待つ */
    wTmrTxRxWait = SPI_TXTIME;
    while(st_byRecvComple == 0){
        if((wTmrTxRxWait == 0) || (st_byRSPI1_OVRF != 0)){
            /* RSPI1割り込み禁止 */
            IEN(RSPI1, SPTI1) = 0;
            IEN(RSPI1, SPRI1) = 0;
            return FALSE;
```

```
        }
    }

    return TRUE;
}


/*****************************************************************************/
/*                          SPI転送の初期化(RSPI1)                           */
/*****************************************************************************/

void Init_SPI1(void)
{
    /* RSPI1の初期設定 */
    Init_RSPI1();
    /* フラグ初期化 */
    st_byRecvComple = 0;            /* 送受信完了フラグ */
    st_byRSPI1_OVRF = 0;            /* ステータスレジスタSPSR */
}
```