

```

#include <htc.h>
#include "i2c.h"

/*
 * マスターモード専用I2C関数 教育・ホビー用
 * 営利目的・商用への利用は禁止
 */

/*
 * 詳しいタイミングはNXPの資料参照のこと
 * PORTアドレスはi2c.hでdefineする
 */

/*
 * 関数の説明
 * ストップ状態にします：クロックをHにしてデータをL->Hします
 */

void i2c_stop(void)
{
    SDA_LOW(); /* 初めはデータをL確認 */
    SCL_HIGH(); /* SCLをHにする */

    // __delay_us(I2C_TM_DATA_SU); /* セットアップ時間待つ */
    // SCL_DIR = I2C_TNPUT; /* クロックをHにフロート */
    // __delay_us(I2C_TM_STOP_SU); /* ホールド時間待つ */
    SDA_HIGH(); /* SDAをLからHにデータ遷移 */
    // __delay_us(I2C_TM_BUS_FREE); /* バス開放時間 */
    // SDA_DIR = I2C_TNPUT; /* データ線をHにフロート */
}

/*
 * スタート状態にします：まずはデータ線を確実にHにして
 * スタート状態にします。i2c_Start()マクロも同一定義です
 */

void
i2c_restart(void)
{
    SCL_HIGH(); /* クロックは確実にHとする */
    SDA_HIGH(); /* データ線をHとする */

    __delay_us(I2C_TM_DATA_SU); /* セットアップ時間待つ */

    SCL_DIR = I2C_INPUT; /* クロックパルスをHに */
    __delay_us(I2C_TM_SCL_HIGH);

    SDA_LOW(); /* SDAをHからLに変化させる */
    __delay_us(I2C_TM_START_HD);
}

/*
 * バイトをスレーブに送ります
 * エラー時にはTRUEを返します
 */
unsigned char
i2c_sendbyte(unsigned char byte)
{
    signed char i;

    for(i=7; i>=0; i--) /* 8ビット分処理します */
    {
        SCL_LOW(); /* クロックをLに */

        /* データのホールドタイムは0で、すぐデータ送る */

        SDA_DIR = ((byte>>i)&0x01); /* データ線のフロートするか */
        if ((byte>>i)&0x01) { /* 送るビットは */
            SDA_HIGH();
        } else {
            SDA_LOW();
        }
    }
}

```

```

        }
        __delay_us(I2C_TM_DATA_SU);
        SCL_DIR = I2C_INPUT; /* クロック線をHにフロート */
        if(i2c_waitforSCL()) /* クロック開放しているか */
            return TRUE; /* バスエラー */
        __delay_us(I2C_TM_SCL_HIGH); /* クロックのHレベル時間を確保 */
    }
    return FALSE;
}

/*
 * スレーブアドレスとデータ方向指定ビットを送ります
 * 7ビットアドレス（最下位無視）、方向（read:1,write:0）
 */
unsigned char
i2c_sendaddress(unsigned char address, unsigned char rw)
{
    return i2c_sendbyte(address | (rw?1:0));
}

/*
 * スレーブ側からのアックをチェックします
 * アックを返すか、アックなしか、バスエラーならERRORです
 */
signed char
i2c_readack(void)
{
    unsigned char ack;

    SCL_LOW(); /* クロック線をLにする */
    SDA_DIR = I2C_INPUT; /* データ線を開放 - アック応答をみる */
    __delay_us(I2C_TM_SCL_TO_DATA); /* データ出力を確定のためSCLをLにする */
    SCL_HIGH(); /* クロック線をH */
    __delay_us(I2C_TM_DATA_SU); /* セットアップ時間待ち */
    ack = SDA; /* アック応答よみ */

    /* バイト処理の後クロック線をスレーブが開放したか確認 */

    if(i2c_waitforSCL())
        return I2C_ERROR; /* スレーブ動作エラー */
    SCL_LOW(); // append
    return ack;
}

/*
 * スレーブからバイトリードします
 * 読んだバイトを返すか、もしバスエラーだったらERRORを返します
 */
int
i2c_readbyte(void)
{
    unsigned char i;
    unsigned char byte = 0;

    for(i=0; i<8; i++)
    {
        SCL_LOW(); /* クロック線をLにする */
        __delay_us(I2C_TM_SCL_LOW); /* クロックのL時間を保証 */
        SDA_DIR = I2C_INPUT; /* rデータ線を開放 */

        SCL_DIR = I2C_INPUT; /* クロック線をフロートしてH */
        if(i2c_waitforSCL())
            return I2C_ERROR;
        __delay_us(I2C_TM_SCL_HIGH);
        byte = byte << 1; /* 次のビットを読む */
        byte |= SDA;
    }
    return (int)byte;
}

/*
 * スレーブ側にアックあるいは非アックを送ります。

```

```

    *           i2c
    *   I2C_LASTというstatusを送ると、これで最後バイトを送りますという意味です
    */
void
i2c_sendack(unsigned char status)
{
    SCL_LOW();                               /* クロック線をLにする */
    if ( status & 0x01) {
        SDA_LOW();                             /* drive line low -> more to come */
    } else {
        SDA_HIGH();
    }
    __delay_us(I2C_TM_DATA_SU);
    SCL_DIR = I2C_INPUT;                       /* クロック線をHIに */
    __delay_us(I2C_TM_SCL_HIGH);
    return;
}

/*
 *   スレーブに1バイトを送ります。I2C_ERROR、アック、非アックを返します
 */
signed char
i2c_putbyte(unsigned char data)
{
    if(i2c_sendbyte(data))
        return I2C_ERROR;
    return i2c_readack();                       /* アック、非アックを読み取り返します */
}

/*
 *   スレーブから1バイト読み取り、転送のアックを確認
 *   戻り値はI2C_ERRORならtrueであり、それ以外はbyte
 */
int
i2c_getbyte(unsigned char more)
{
    int byte;

    if((byte = i2c_readbyte()) == I2C_ERROR)
        return I2C_ERROR;

    i2c_sendack(more);

    return byte;
}

/*
 *   スレーブにバイト列を送り、転送のアックを確認する
 *   もし転送不成功なら転送残バイト数を返す
 */
int
i2c_putstring(const unsigned char *str, unsigned char length)
{
    signed char error;

    while(length)
    {
        if((error = i2c_putbyte(*str)) == I2C_ERROR)
            return -(int)length;               /* バスエラー時の戻り値 */
        else
            if(error)
                return (int)length;           /* アックがありません */
        str++;
        length--;
    }

    return FALSE;                               /* 処理すべてOK */
}

/*
 *   スレーブから指定されたバイト数をstr文字列に格納、転送のアックを返す
 *   読み込みに成功しなかった文字数を返す
 */
unsigned char
i2c_getstring(unsigned char *str, unsigned char number)

```

```

                                i2c
{
    int byte;
    while(number)
    {
        if((byte = i2c_getbyte(number-1)) == I2C_ERROR) /* バスエラーなので残バイト数返す */
            return number;
        else
            *str = (unsigned char)byte;
        str++;
        number--;
    }
    return FALSE; /* 処理すべてOK */
}

/*
 * 指定アドレスのデバイスと通信を開始する。モードは
 * I2C_READあるいはI2C_WRITEで指定される。
 * もし指定アドレスに対してアック無いとTRUEを返す
 */
unsigned char
i2c_open(unsigned char address, unsigned char mode)
{
    i2c_start();
    i2c_sendaddress(address, mode);
    if(!i2c_readack())
        return TRUE;

    return FALSE; /* 処理すべてOK */
}

/*
 * 遅いスレーブの場合用にクロック線が開放されるのをまつ
 * タイムアウト時間の後もSCLが開放されないときはTRUEを返す
 * もしそうでないときはSCLが開放された時にFALSEを返す
 */
unsigned char
i2c_waitforSCL(void)
{
    /* SCL_DIRをここでは入力にする */
    if(!SCL)
    {
        __delay_us(I2C_TM_SCL_TMO);
        /* もしまだクロックがLのままならバスエラー */
        if(!SCL)
            return TRUE;
    }
    return FALSE;
}

/*
 * バスを開放する
 */
void
i2c_free()
{
    unsigned char ucl;

    SDA_DIR=I2C_INPUT;
    for(ucl=0;ucl!=9;ucl++)
    {
        SCL_HIGH();
        __delay_us(5);
        SCL_LOW();
        __delay_us(5);
    }
}

/*
 * 1文字読んでucAdrが0ならば読み取り終了する
 */
unsigned char i2c_read(unsigned char ucAdr)
{
    unsigned char ucDat;

```

```
if (i2c_readfrom(ucAdr)==0)          i2c
{
    ucDat=i2c_getbyte(I2C_MORE);
    i2c_stop();
}
return(ucDat);
}
```