

```

/*****
/* Program      : 1306SPI.C                               */
/* Function     : SPI Real Time Clock Utility control routines */
/* Author      : John F. Fitter B.E.                       */
/* Reference    : Dallas Data Sheet 032598 1/20 DS1306     */
/*                                                     */
/* Rev No.     Rev date   Test date   Test platform   Description           */
/* -----     - - - - -   - - - - -   - - - - -   - - - - -           */
/*      00      6jun98                Original in HiTech C           */
/*                                                     */
/*      Copyright © 1997 Eagle Air Australia Pty. Ltd. All rights reserved */
/*****

#define _1306SPI_C

#include <commdefs.h>
#include "main.h"
#include "1306spi.h"
#include "spi.h"
#include "25cxxspi.h"
#include "delays.h"

// Procedure to return data at 1306 address.

unsigned char read_1306(unsigned char address) {
    unsigned char n;

    select_1306();
    spi_write(address);
    n = spi_read(0);
    deselect_1306();
    return n;
}

// Procedure to write data to 1306 at address - as far as possible this procedure uses all
// inline code to conserve stack space.

void write_1306(unsigned char address, unsigned char data) {
    unsigned char n;

    select_1306();
    spi_write(CONTROL_1306);
    n = spi_read(0);
    deselect_1306();
    n &= ~WP_1306;
    select_1306();
    spi_write(CONTROL_1306 | 0x80);
    spi_write(n);
    deselect_1306();
    select_1306();
    spi_write(address | 0x80);
    spi_write(data);
    deselect_1306();
    select_1306();
    spi_write(CONTROL_1306 | 0x80);
    n = spi_read(0);
    deselect_1306();
    n |= WP_1306;
    select_1306();
    spi_write(CONTROL_1306 | 0x80);
    spi_write(n);
    deselect_1306();

// Procedure to initialize the 1306 for reading. Leaves the 1306 write disabled.
// contr_1306 is the initial value of the control register bits. trk_1306 is the initial
// value of the trickle charge register bits. The 1306 is set to the 24hr mode.

void init_1306(unsigned char contr_1306, unsigned char trk_1306) {

    DS1306_ce_dir = B_OUT;
    write_1306(CONTROL_1306, contr_1306);
    write_1306(TRICKLE_1306, trk_1306);
    write_1306(HOURS_1306, read_1306(HOURS_1306)&0xbf);
}

// Procedure to retrieve a floating point double value from nvram and return the value.

```

```
// address is the location of the double.
```

```
double read_double_1306(unsigned char address) {
    double dbl;
    *(unsigned char*)&dbl      = read_1306(address++);
    *((unsigned char*)&dbl+1) = read_1306(address++);
    *((unsigned char*)&dbl+2) = read_1306(address++);
    *((unsigned char*)&dbl+3) = read_1306(address);
    return dbl;
}
```

```
// Procedure to write a floating point double value to nvram at address.
```

```
void write_double_1306(double value, unsigned char address) {
    write_1306(address++, *(unsigned char*)&value);
    write_1306(address++, *((unsigned char*)&value+1));
    write_1306(address++, *((unsigned char*)&value+2));
    write_1306(address,  *((unsigned char*)&value+3));
}
```

```
// Procedures to increment/decrement the various fields of the date/time group. The direction
// argument is true to increment and false to decrement.
// Note: due to stack limitations, call these only from the main loop.
```

```
void id_dt_1306(unsigned char dt_field, unsigned char dirn) {
    unsigned char mth, yr;

    dirn = dirn ? 2 : 1;
    mth  = bcd_to_dec(read_1306(MONTH_1306));
    yr   = bcd_to_dec(read_1306(YEAR_1306));

    switch(dt_field) { // *** the order of the cases is really important, not the values ***

        // Increment/decrement/wrap the day. The day can range from 1 to 7.
        case 3: write_1306(DAY_1306, dec_to_bcd(inc_dec(bcd_to_dec(
            read_1306(DAY_1306)), 1, 7, dirn)));
            break;

        // Increment/decrement/wrap the year. The year can range from 0 to 99.
        case 6: write_1306(YEAR_1306, dec_to_bcd(yr = inc_dec(yr, 0, 99, dirn)));
            dirn = 0;

        // Increment/decrement/wrap the month. The month can range from 1 to 12.
        case 5: write_1306(MONTH_1306, dec_to_bcd(mth = inc_dec(mth, 1, 12, dirn)));
            dirn = 0;

        // Increment/decrement/wrap the date. The limits of the date are determined by
        // leap/clear years and the month. The date can range from 1 to 28, 29, 30, or 31
        // depending on the year and the month.
        case 4: if(mth == 2) yr = (yr & 3) ? 28 : 29; // yr is used as a temporary var
            else if((mth == 4) || (mth == 6) || (mth == 9) || (mth == 11)) yr = 30;
            else yr = 31;
            write_1306(1306, dec_to_bcd(inc_dec(bcd_to_dec(
                read_1306(1306)), 1, yr, dirn)));
            break;

        // Increment/decrement/wrap the hours. The hours can range from 0 to 23.
        case 2: write_1306(HOURS_1306, dec_to_bcd(inc_dec(bcd_to_dec(
            read_1306(HOURS_1306)), 0, 23, dirn)));
            break;

        // Increment/decrement/wrap the minutes. The minutes can range from 0 to 59.
        case 1: write_1306(MINUTES_1306, dec_to_bcd(inc_dec(bcd_to_dec(
            read_1306(MINUTES_1306)), 0, 59, dirn)));
            break;
    }
}
```

```
// Procedure to increment/decrement a variable by one. If dirn is 1 the variable is
// decremented, if dirn is 2 the variable is incremented, and if dirn is 0 the variable is
// just clipped to max/min. Rollover from min to max and from max to min is supported.
```

```
unsigned char inc_dec(unsigned char val, unsigned char min,
    unsigned char max, unsigned char dirn) {
```

```
if(dirn & 1) {
    if(val == min) val = max;
    else --val;
}else if(dirn & 2) {
    if(val = max) val = min;
    else ++val;
}
if(val > max) return max; // also returns max when val < 0
if(val < min) return min; // only makes sense when min >= 1
return val;
}
```

```
// ***** EOF 1306SPI.C *****
```