```
#ifndef DS1306_RTC_DRIVER_H_
#define DS1306_RTC_DRIVER_H_

/* Standard Library Inlcude Files */
#include <iom128.h>

#define _BV(bit) (1 << bit)

//*****************************************************************
**********
// Function : void spi_rtc_ds1306_config (void)
// Date and version : 031118, version 1.0
// Target MCU : ATmega128 @ 16MHz
// Author : Ken Short
// DESCRIPTION
// This function unselects the ds_1306 and configures an ATmega128
operated at
// 16 MHz to communicate with the ds1306. Pin PA1 of the ATmega128 is
used to
// select the ds_1306. SCLK is operated a the maximum possible
frequency for
// the ds1306.
//*****************************************************************
**********
void spi_rtc_ds1306_config(void);

//*****************************************************************
*******
// Function :
// void write_RTC (unsigned char reg_RTC, unsigned char data_RTC)
//
// Target MCU : ATmega128 @ 16MHz
// Target Hardware ;
// Author : Ken Short
// DESCRIPTION
// This function writes data to a register in the RTC. To accomplish
this, it
// must first write the register's address (reg_RTC) followed by
writing the
// data (data_RTC). In the DS1306 data sheet this operation is called
an SPI
// single-byte write.
//*****************************************************************
******
void write_rtc(unsigned char reg_RTC, unsigned char data_RTC);

//*****************************************************************
*******
// Function Name :
// unsigned char read_rtc (unsigned char reg_RTC)
// Target MCU : ATmega128 @ 16MHz
// Author : Ken Short
// DESCRIPTION
```

```
// This function reads data from a register in the RTC. To accomplish
this, it
// must first write the register's address (reg_RTC) followed by
writing a dummy
// byte to generate the SCLKs to read the data (data_RTC). In the
DS1306 data
// sheet this operaration is called am SPI single-byte read.
//*****************************************************************
******
unsigned char read_rtc(unsigned char reg_RTC);

//*****************************************************************
*******
// Function Name : "block_write_RTC"
// void block_write_RTC (volatile unsigned char *array_ptr,
// unsigned char strt_addr, unsigned char count)
// Target MCU : ATmega128 @ 16MHz
// Author : Ken Short
// DESCRIPTION
// This function writes a block of data from an array to the DS1306.
strt_addr
// is the starting address in the DS1306. count is the number of data
bytes to
// be transferred and array_ptr is the address of the source array.
//*****************************************************************
******
void block_write_rtc (volatile unsigned char *array_ptr, unsigned char
strt_addr, unsigned char count);

//*****************************************************************
*******
// Function Name : "block_read_RTC"
// void block_read_RTC (volatile unsigned char *array_ptr,
// unsigned char strt_addr, unsigned char count)
// Target MCU : ATmega128 @ 16MHz
// Author : Ken Short
// DESCRIPTION
// This function reads a block of data from the DS1306 and transfers
it to an
// array. strt_addr is the starting address in the DS1306. count is
the number
// of data bytes to be transferred and array_ptr is the address of the
// destination array.
//*****************************************************************
******
void block_read_rtc (volatile unsigned char *array_ptr, unsigned char
strt_addr, unsigned char count);

/**
 * Here, add a little description.
 * @param   list all parameters
 * @return  explain what is returned
 */
void unlock_rtc();

/**
 * Here, add a little description.
```

```
 * @param   list all parameters
 * @return  explain what is returned
 */
void lock_rtc();

#endif /* DS1306_RTC_DRIVER_H_ */
```

```
/*****************************************************************
 * File:        fsm.h
 * Author:      Bryant Gonzaga
 * Created:     4/12/2018
 * Modified:    4/12/2018
 *
 * Notes:
 *  Inteneded for the ATmega128
 *
 * Description:
 *  This file contains type defs and structures relevent to our fsm.
 *****************************************************************/

#ifndef FSM_H_
#define FSM_H_

/* FSM States */
typedef enum
{
    set_alarm_state = 0,
    set_time_state  = 1,
    display_state   = 2,
}  state;

/* Input Keys */
typedef enum
{
    set_alarm_key,
    set_time_key,
    confirm_key,
    rtc_1hz_key,
    cancel_key,
    num_keys,
    eol
} key;

/* Key Values */
#define SET_ALARM_KEY   15
#define SET_TIME_KEY    14
#define CONFIRM_KEY     12
#define CANCEL_KEY      10

/* Function Pointer (For Output of FSM) */
typedef void (* task_ptr) ();

/* Transition Struct */
typedef struct
{
    key key_val;
    state next_state;
    task_ptr task_ptr;
} transition;

#endif /* FSM_H_ */
```

```
/*****************************************************************
 * File:        fsm_state_tables.h
 * Author:      Bryant Gonzaga
 * Created:     4/12/2018
 * Modified:    4/12/2018
 *
 * Notes:
 *  Intended for the ATmega128.
 *
 * Description:
 *  Contains the state tables for our fsm.
 *****************************************************************/

#ifndef FSM_STATE_TABLES_H_
#define FSM_STATE_TABLES_H_

#include "fsm_defs.h"
#include "fsm_tasks.h"

/* Transitions from the Display State */
const transition display_transitions[] = {
    // INPUT          NEXT_STATE          TASK
    {set_time_key   , set_time_state    , time_input_init},
    {set_alarm_key  , set_alarm_state   , alarm_input_init},
    {rtc_1hz_key    , display_state     , dsp_all_fn},
    {eol            , display_state     , error_fn}
};

// TODO: confirm always transfers to the display state so its like
cancel_key. i want to find a way where its ignored if not valid.

/* Transitions from the Set-Time State */
const transition set_time_transitions[] = {
//  INPUT       NEXT_STATE          TASK
    {num_keys   , set_time_state    , time_input_handler_fn},
    {confirm_key, display_state     , confirm_time_fn},
    {cancel_key , display_state     , dsp_all_fn},
    {eol        , set_time_state    , error_fn}
};

/* Transitions from the Set-Alarm State */
const transition set_alarm_transitions[] = {
    // INPUT       NEXT_STATE          TASK
    {num_keys   , set_alarm_state   , alarm_input_handler_fn},
    {confirm_key, display_state     , confirm_alarm_fn},
    {cancel_key , display_state     , dsp_all_fn},
    {eol        , set_alarm_state   , error_fn}
};

/* All Transitions */
const transition* ps_transitions_ptr[3] = {
    set_alarm_transitions,
    set_time_transitions,
    display_transitions
};
```

```
#endif /* FSM_STATE_TABLES_H_ */
```

```
/*****************************************************************
 * File:        fsm_tasks.h
 * Author:      Bryant Gonzaga
 * Created:     4/12/2018
 * Modified:    4/12/2018
 *
 * Notes:
 *  Processor specific, libraries need
 *
 * Description:
 *  A full description of what can be found in this file
 *
 * How To:
 *  If necessary add some instructions on how to use the file.
 *****************************************************************/

#ifndef FSM_TASKS_H_
#define FSM_TASKS_H_

#include <stdio.h>

#include "ds1306_rtc_driver.h"
#include "fsm_defs.h"
#include "lcd.h"


void time_input_init();
void alarm_input_init();

void alarm_input_handler_fn();
void time_input_handler_fn();
void dsp_alarm_setter_fn();
void dsp_time_setter_fn();
void confirm_alarm_fn();
void confirm_time_fn();
void dsp_all_fn();
void error_fn();
void nop_fn();          // not sure if needed yet

#endif /* FSM_TASKS_H_ */
```

```
/*********************************************
 * File:    humidicon.h
 * Author:  Bryant Gonzaga
 * Date:    3/6/2018
 *********************************************/

#ifndef HUMIDICON_H_
#define HUMIDICON_H_

#include <iom128.h>

#define _BV(bit) (1 << (bit))

//*********************************************************************
**********
// Function : void SPI_humidicon_config (void)
// Date and version : version 1.0
// Target MCU : ATmega128A @ 16MHz
// Author :
// DESCRIPTION
// This function unselects the HumidIcon and configures it for
operation with
// an ATmega128A operated a 16 MHz. Pin PA0 of the ATmega128A is used
to select
// the HumidIcon. SPI for humidicon has a max
// slave clock frequency of 800 kHz
//
// Modified
//*********************************************************************
**********
void spi_humidicon_config();

//*********************************************************************
**********
// Function : unsigned char read_humidicon_byte(void)
// Date and version : version 1.0
// Target MCU : ATmega128A
// Author : Ken Short
// DESCRIPTION
// This function reads a data byte from the HumidIcon sensor and
returns it as
// an unsigned char. The function does not return until the SPI
transfer is
// completed. The function determines whether the SPI transfer is
complete
// by polling the appropriate SPI status flag.
//
// Modified
//*********************************************************************
**********
unsigned char read_humidicon_byte();

//*********************************************************************
**********
// Function : void read_humidicon (void)
// Date and version : version 1.0
```

```
// Target MCU : ATmega128A
// Author :
// DESCRIPTION
// This function selects the Humidicon by asserting PA0. It then calls
// read_humidicon_byte() four times to read the temperature and
humidity
// information. Is assigns the values read to the global unsigned ints
humidicon_byte1,
// humidion_byte2, humidion_byte3, and humidion_byte4, respectively.
The
// function then deselects the HumidIcon.
//
// The function then extracts the fourteen bits corresponding to the
humidity
// information and stores them right justified in the global unsigned
int humidity_raw.
// Next if extracts the fourteen bits corresponding to the temperature
// information and stores them in the global unsigned int
temperature_raw. The function
// then returns
//
// Modified
//*****************************************************************
**********
void read_humidicon();

long int compute_scaled_rh(unsigned int rh);
long int compute_scaled_temp(unsigned int temp);

#endif /* HUMIDICON_H_ */
```

```
#ifndef LCD_DOG_C_DRIVER_H_
#define LCD_DOG_C_DRIVER_H_

#include <iom128.h>
#include <intrinsics.h>

// Bit Value
#define _BV(bit) (1 << (bit))

// PINs for LCD
#define SCK     1
#define MISO    3
#define MOSI    2
#define SS_bar  0
#define RS      4
#define BLC     5

#define LCD_PORT PORTB

char dsp_buff_1[16];
char dsp_buff_2[16];
char dsp_buff_3[16];

void delay_30uS();

void delay_40mS();

void init_spi_lcd();

int lcd_spi_transmit_CMD(char cmd);

int lcd_spi_transmit_DATA(char data);

void init_lcd_dog(void);

void update_lcd_dog(void);

#endif /* LCD_DOG_C_DRIVER_H_*/
```

```
//*********************************************************************
*******
//
// File Name            : lcd.h
// Title                : Header file for LCD module
// Date                 : 02/07/10
// Version              : 1.0
// Target MCU           : ATmega128 @  MHz
// Target Hardware      ;
// Author               : Ken Short
// DESCRIPTION
// This file includes all the declaration the compiler needs to
// reference the functions and variables written in the files
lcd_ext.c.
// lcd.asm and lcd_dog_iar_driver.asm
//
// Warnings             : none
// Restrictions         : none
// Algorithms           : none
// References           : none
//
// Revision History     : Initial version
//
//
//*********************************************************************
******

#ifndef LDC_H_
#define LDC_H_

/**
 *  To use the lcd functions in lcd_dog_iar_driver.asm lcd_ext.c all
that is
 *  needed is to include this file.
 */

/**
 *  This declaration tells the compiler to look for dsp_buff_x in
 *  another module. It is used by lcd_ext.c and main.c to locate the
buffers.
 */
extern char dsp_buff_1[16];
extern char dsp_buff_2[16];
extern char dsp_buff_3[16];

/**
 *  Declaratios of low level lcd functions located in
lcd_dog_iar_driver.asm
 *  Note that these are external.
 */
__version_1 extern void init_lcd_dog(void);
__version_1 extern void update_lcd_dog(void);

/**
 *  These functions are located in lcd_ext.c
 */
```

```
extern void clear_dsp(void);
extern int putchar(int);

#endif /* LCD_H_ */
```