/*
 * File: Stepper.h
 * Author: Elkaim
 *
 * Software module to drive a stepper motor through a normal H-bridge in
 * full-step drive
 * mode. The module uses TIMER3 and is capable of generated 1/2 to 20,000
 * steps per second.
 * The nominal port used is PORTZ and can be changed by changing the
 * appropriate #defines
 * below.
 *
 * NOTE: Module uses TIMER3 for its interrupts. When using this module to
 * directly drive
 * the DRV8811 stepper board, it will be stepping at 1/2 the rate
 * specified due to
 * toggling the pin on every entry to the ISR (board steps on
 * rising edge).
 *
 * STEPPER_TEST (in the .c file) conditionally compiles the test harness
 * for the code.
 * Make sure it is commented out for module useage.
 *
 * Created on January 2, 2012, 9:36 PM
 */

#ifndef Stepper_H
#define Stepper_H
#include <IO_Ports.h>

/**************************************************************************
***** *
* PUBLIC #DEFINES
*  
**************************************************************************

#ifndef SUCCESS
#define SUCCESS 0
#define ERROR -1
#endif

#ifndef TRUE
#define TRUE 1
#define FALSE 0
#endif

#define FORWARD 1
#define REVERSE 0

//There are four types of drive
//Hbridge works with full wave and half;
//Stepper board uses stepperboard
const enum drive{
    full, wave, half, stepperboard
}drive;

/**************************************************************************
*****
PUBLIC VARIABLES
***************************************************************************/

// H Bridges motors: (uint16_t[]){Enable_APin, Enable_BPin, DirAPin, DirBPin}
// Stepper motors: (uint16_t[]){EnablePin, DirectionPin, StepPin}
#define STEPPER_PINS {PIN5, PIN4, PIN3}
#define TYPE_OF_DRIVE stepperboard //one of the four drive states above
#define STEPPER_PORT PORTZ

/**************************************************************************
*****
PUBLIC FUNCTION PROTOTYPES
***************************************************************************/

/**
 * @Function: Stepper_Init(unsigned short int stepper, char direction,
 * unsigned int steps, unsigned int rate);
 * @param direction  REVERSE or FORWARD
 * @param steps Number of steps stepper takes before stopping
 * @param rate - steps per second (1Hz-2kHz)
 * @return SUCCESS or ERROR
 * @brief sets up the stepper and starts the timer.
 * @brief Stepper does not step until the Stepper_Start()
 * @author Gabriel Hugh Elkaim, 2012.01.28 23:21
 * @note returns ERROR if stepper is in stepping state when inited*/
char Stepper_Init(char direction, unsigned int steps, unsigned int rate);

/**
 * @Function: Stepper_ChangeStepRate(unsigned short int rate);
 * @return SUCCESS or ERROR
 * @brief Changes the step rate of the stepper motor
 * @note can be changed mid-stepping or while stepper is off
 * @author Gabriel Hugh Elkaim, 2012.01.28 23:21
 * @revised Soja Morgens, Max Lichtenstein, 2014.05.21 */
char Stepper_ChangeStepRate(unsigned short int rate);

/**
 * @Function: Stepper_SetSteps(char direction, unsigned int steps);
 * @param stepper - which stepper in int
 * @param direction – stepper direction (FORWARD or REVERSE)
 * @param steps - number of steps to take
 * @return SUCCESS or ERROR
 * @brief Sets the number of steps and direction for the stepper drive.
 */
* @note Wipes out the current steps.
* @note Does not change the state of the motor: ie if its stepping or halted
* @author Gabriel Hugh Elkaim, 2012.01.28 23:21 */

char Stepper_SetSteps(char direction, unsigned int steps);

/**
* @Function: Stepper_GetRemainingCount(un);
* @param stepper -which stepper
* @return number of remaining steps
* @brief Returns the number of remaining steps
* @author Gabriel Hugh Elkaim, 2012.01.28 23:21 */
int Stepper_GetRemainingCount(void);

/**
* @Function: Stepper_GetRate(void);
* @param stepper -which stepper
* @return number of remaining steps
* @brief Returns the number of remaining steps
* @author Gabriel Hugh Elkaim, 2012.01.28 23:21 */
unsigned int Stepper_GetRate(void);

/**
* @Function: Stepper_Start(unsigned short int stepper);
* @param stepper which stepper you are using
* @return SUCCESS or ERROR
* @brief Starts the stepper stepping. Stepper will not step if no steps left.
* @author Gabriel Hugh Elkaim, 2012.01.28 23:21 */
char Stepper_Start(void);

/**
* @Function: char Stepper_Pause(unsigned short int stepper);
* @param stepper which stepper
* @return SUCCESS or ERROR
* @brief Halts the stepper driver, but does not affect the step count or rate.
* @note Stepper resumes by calling Stepper_Start
* @author Gabriel Hugh Elkaim, 2012.01.28 23:21 */
char Stepper_Pause(void);

/**
* @Function: Stepper_End(void);
* @return SUCCESS or ERROR
* @brief Shuts down the stepper driver software module and timer
* @brief Releases the pins
* @brief Stepper cannot be used until re-init
* @author Gabriel Hugh Elkaim, 2012.01.28 23:21 */
char Stepper_End(void);
#endif