micrstepper_test

1.0.0.1

Generated by Doxygen 1.5.5

Mon Oct 20 07:46:36 2008

Contents

Chapter 1

Main Page

1.1 Introduction

This and related pages contain the documentation for all the header files, source files, functions, variables, enums, and defines related to the project 'Microstepper Motor Controller'.

1.2 Compilation Info

This software was written for the IAR Embedded Workbench 4.21a kickstart and AVR-GCC compiler.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

File Documentation

3.1 .dep/main.o.d File Reference

3.2 compiler.h File Reference

Common and compiler specific files.

```
#include <avr\io.h>
#include <avr\interrupt.h>
#include <avr\pgmspace.h>
#include "motor.h"
#include "config.h"
#include "tables.h"
#include "uart.h"
#include "motor.c"
#include "uart.c"
```

Defines

- #define **_AVR_GCC**
  Include Compiler specific code for ACR-GCC *compiler.*
- #define **_SIN_TABLE**(**i**) pgm_read_byte(&sinewave[**i**])
  Lookup sine table.
- #define **_COS_TABLE**(**i**) pgm_read_byte(&sinewave[(**i**+64)%256])
  Lookup cos table.
- #define **_TIMER1_OVERFLOW_INTERRUPT**() ISR(TIM1_OVF_vect)
  Compiler specific header for timer1 overflow ISR.

3.2.1 Detailed Description

Common and compiler specific files.

This file has been prepared for Doxygen automatic documentation generation.

This files allows to incorporate both, AVR-GCC and IAR-C compilers to compile the same piece of code.

1. Includes common and compiler specific header files.
2. Makes standard headers for some compiler specific ones.

Compiler specific include files and defines inclusion...

```
#ifdef _AVR_GCC
        #include <avr\io.h>
        #include <avr\interrupt.h>
        #include <avr\pgmspace.h>
        #define _SIN_TABLE(i)           pgm_read_byte(&sinewave[i])
        #define _COS_TABLE(i)           pgm_read_byte(&sinewave[(i+64)%256])
        #define _TIMER1_OVERFLOW_INTERRUPT()  ISR(TIM1_OVF_vect)
#endif
```

```
#ifdef _IAR_C
        #include <ioavr.h>
        #include <inavr.h>
        #include <pgmspace.h>
        #define _SIN_TABLE(i)           sinewave[i]
        #define _COS_TABLE(i)           sinewave[(i+64)%256]
        #define _TIMER1_OVERFLOW_INTERRUPT() (__interrupt void Timer1_Overflow_ISR(void))
#endif
```

Including files commpon to both compilers.

```
#include "motor.h"
#include "config.h"
#include "tables.h"
#include "uart.h"
#include "motor.c"
#include "uart.c"
```

Definition in file **compiler.h**.

3.2.2 Define Documentation

3.2.2.1 #define _AVR_GCC

Include Compiler specific code for *ACR-GCC* compiler.

If the code is to be compiled woth *AVR-GCC* compiler the this statement should be left uncommented, and `define _IAR_C` should be commented.

Definition at line 62 of file compiler.h.

3.2.2.2 #define _COS_TABLE(i) pgm_read_byte(&sinewave[(i+64)%256])

Lookup cos table.

Lookup the same sine table as in *define* ***_SIN_TABLE(i)*** *(*p. 7*)* stored in flash memory,convert it into an effective cos table and retrieve the value of the ith element in the effective cos table.

Definition at line 84 of file compiler.h.

Referenced by _TIMER1_OVERFLOW_INTERRUPT().

3.2.2.3 #define _SIN_TABLE(i) pgm_read_byte(&sinewave[i])

Lookup sine table.

Lookup the sine table stored in flash memory and access the ith element

Definition at line 75 of file compiler.h.

Referenced by _TIMER1_OVERFLOW_INTERRUPT().

3.2.2.4 #define _TIMER1_OVERFLOW_INTERRUPT() ISR(TIM1_OVF_vect)

Compiler specific header for timer1 overflow ISR.

This statement is neccecary for incorporating both AVR_GCC, and IAR_C compilers. Defines a common 'TIMER1 overflow interrupt' header.

Definition at line 92 of file compiler.h.

3.3 config.h File Reference

This file contains many important system parameters and statements.

Defines

- #define **_RESET_PORT**() DDRB=(1<<DDB5)|(1<<DDB4)|(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<

  Resets all pins in PORTB configuring each pin as an output pin.

- #define **_CONFIGURE_PWM1_CHANNEL**() OCR1A = 0x80
  Initialize PWM1 channel.

- #define **_CONFIGURE_PWM2_CHANNEL**() OCR1B = 0x80
  Initialize PWM2 channel.

- #define **_CONFIGURE_PWM_BASE_FREQ**() OCR1C=0xFF
  Set OCR1C to TOP.

- #define **_CONFIGURE_I_O_PINS**() DDRB=(1<<DDB5)|(1<<DDB4)|(0<<DDB3)|(0<<DDB2)|(1<

  Configure the GPIO pins on AtTiny45 according to the given system.

- #define **_SET_INTERNAL_PULLUPS**() PORTB|=(0<<PB5)|(0<<PB4)|(1<<PB3)|(1<<PB2)|(0<<

  Set the internal pullups of pins configured as 'Input'.

- #define **_ENABLE_PWM1_CHANNEL**() TCCR1=((1<<PWM1A)|(1<<COM1A1))

  Selects and Enables PWM output on **OC1A** *pin.*

- #define **_ENABLE_PWM2_CHANNEL**() GTCCR=(1<<PWM1B)|(1<<COM1B1)

  Selects and Enables PWM output on **OC1B** *pin.*

- #define **_ENABLE_PLL**() PLLCSR=(1<<PLLE)
  Enable the PLL by setting PLLE bit in PLLCSR.

- #define **_WAIT_TILL_PLL_LOCKED**() while(!(PLLCSR&(1<<PLOCK)));
  Wait till the PLL is locked.

- #define **_DELAY_1000_CYCLES**() for(unsigned int j=0;j<1000;j++)asm("NOP");
  A delay of 1000 clock cycles=125 microseconds at 8MHz.

- #define **_ENABLE_PLL_CLOCK_SOURCE**() PLLCSR|=(1<<PCKE)
- #define **_ENABLE_GLOBAL_INTERRUPTS**() asm("sei")
  Enable global interrupts.

- #define **_DISABLE_GLOBAL_INTERRUPTS**() asm("cli")
  Disable global interrupts.

3.3.1 Detailed Description

This file contains many important system parameters and statements.

This file has been prepared for Doxygen automatic documentation generation.

This file implements a set of parameters that allow the system to be cofigured. It contains a set of mnemonic parameters that make the system easy to understand.

Definition in file **config.h**.

3.3.2 Define Documentation

3.3.2.1 #define _CONFIGURE_I_O_PINS() DDRB=(1<<DDB5)|(1<<DDB4)|(0<<DDB3)|(0<<DDB2)|(1<<DDB1)|(0<<DDB0

Configure the GPIO pins on AtTiny45 according to the given system.

```
|Pin No.|___Description_________|___Config._|

-------------------------------------------
|__PB0__|___home position #1____|___Input___|
|__PB1__|___PWM output #1_______|___Output__|
|__PB2__|___RXD of soft UART____|___Input___|
|__PB3__|___home position #2____|___Input___|
|__PB4__|___PWM output #2_______|___Output__|
|__PB5__|___TXD of soft UART____|___Output__|

-------------------------------------------
```

Definition at line 65 of file config.h.

Referenced by main().

3.3.2.2 #define _CONFIGURE_PWM1_CHANNEL() OCR1A = 0x80

Initialize PWM1 channel.

Initialize PWM1 channel.(OC1A pin on the microcontroller) channel setting the output compare register OCR1A to 'some' value.

Definition at line 24 of file config.h.

Referenced by main().

3.3.2.3 #define _CONFIGURE_PWM2_CHANNEL() OCR1B = 0x80

Initialize PWM2 channel.

Initialize PWM2 channel.(OC1B pin on the microcontroller) channel setting the output compare register OCR1B to 'some' value.

Definition at line 33 of file config.h.

Referenced by main().

3.3.2.4 #define _CONFIGURE_PWM_BASE_FREQ() OCR1C=0xFF

Set OCR1C to TOP.

Set the the output compare register OCR1Cto 255 (TOP). This determines the PWM base frequency

The PWM base frequency can be calculated as:

$$f_w = \frac{(f_p)}{TOP + 1}$$

where,

f_w = The PWM base frequency,

f_p = Is the PLL clock frequency set by CS13/12/11/10 bits in TCCR1,

and, $TOP = 255$

Definition at line 47 of file config.h.

Referenced by main().

3.3.2.5 #define _DELAY_1000_CYCLES() for(unsigned int j=0;j<1000;j++)asm("NOP");

A delay of 1000 clock cycles=125 microseconds at 8MHz.

Definition at line 110 of file config.h.

Referenced by main().

3.3.2.6 #define _DISABLE_GLOBAL_INTERRUPTS() asm("cli")

Disable global interrupts.

Disables global interrupts by clearing the I bit in SREG register

Definition at line 134 of file config.h.

3.3.2.7 #define _ENABLE_GLOBAL_INTERRUPTS() asm("sei")

Enable global interrupts.

Enables global interrupts by setting the I bit in SREG register

Definition at line 127 of file config.h.

Referenced by main().

3.3.2.8 #define _ENABLE_PLL() PLLCSR=(1<<PLLE)

Enable the PLL by setting PLLE bit in PLLCSR.

The PLL is started and if needed internal RC-oscillator is started as a PLL reference clock.

Definition at line 96 of file config.h.

Referenced by main().

3.3.2.9 #define _ENABLE_PLL_CLOCK_SOURCE() PLLCSR|=(1<<PCKE)

Definition at line 120 of file config.h.

Referenced by main().

3.3.2.10 #define _ENABLE_PWM1_CHANNEL() TCCR1=((1<<PWM1A)|(1<<COM1A1))

Selects and Enables PWM output on **OC1A** pin.

This statement selects and enables PWM output on the OC1A pin on Attiny45. Additionally it sets the default frequency of the TCNT1 timer counter register.

Definition at line 82 of file config.h.

Referenced by main().

3.3.2.11 #define _ENABLE_PWM2_CHANNEL() GTCCR=(1<<PWM1B)|(1<<COM1B1)

Selects and Enables PWM output on **OC1B** pin.

This statement selects and enables PWM output on the OC1B pin on Attiny45.

Definition at line 89 of file config.h.

Referenced by main().

3.3.2.12 #define _RESET_PORT() DDRB=(1<<DDB5)|(1<<DDB4)|(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDI

Resets all pins in PORTB configuring each pin as an output pin.

Definition at line 16 of file config.h.

Referenced by main().

3.3.2.13 #define _SET_INTERNAL_PULLUPS() PORTB|=(0<<PB5)|(0<<PB4)|(1<<PB3)|(1<<PB2)|(0<<PB1)|(1<<PB0)

Set the internal pullups of pins configured as 'Input'.

This statement configures the internal pullup resistors of the GPIO pins configured as 'Input' by **setting** them.

This statement should be preceded by the **_CONFIGURE_I_O_PINS()** (p. 10) statement

When the PLOCK bit is set, the PLL is locked to the reference clock. The PLOCK bit should be ignored during initial PLL lock-in sequence when PLL frequency overshoots and undershoots, before reaching steady state. The steady state is obtained within 100 s. After PLL lock-in it is recommended to check the PLOCK bit before enabling PCK for Timer/Counter1.

Referenced by main().

3.3.2.14 #define _WAIT_TILL_PLL_LOCKED() while(!(PLLCSR&(1<<PLOCK)));

Wait till the PLL is locked.

This statement configures the internal pullup resistors of the GPIO pins configured as 'Input' by **setting** them.

This statement should be preceded by the **_CONFIGURE_I_O_PINS()** (p. 10) statement

Definition at line 105 of file config.h.

Referenced by main().

3.4 main.c File Reference

```
#include "compiler.h"
```

Functions

- **_TIMER1_OVERFLOW_INTERRUPT** ()

  Timer1 overflow interrupt subroutine.

- int **main** (void)

  The main function.

Variables

- unsigned char **count_down** = 1

  Used to count down from an initial value (= speed) to zero.

- unsigned char **r1** = 0

  This determines which action will be performed.

3.4.1 Detailed Description

Atmel Corporation

- File : **main.c** (p. 14)
- Compiler : IAR EWAVR 4.21a kickstart and AVR-GCC
- Support mail : `avr@atmel.com`
- Supported devices : ATtiny25, ATtiny 45, ATtiny85
- AppNote : Microstepper motor controller
- Description : The full implementation is included in this file.

Revision

1.4

Date

Monday, February 19,2007 UTC

Definition in file **main.c**.

3.4.2 Function Documentation

3.4.2.1 _TIMER1_OVERFLOW_INTERRUPT ()

Timer1 overflow interrupt subroutine.

This is the the subroutine called when timer1 overflows. Many critical functions pertaining to the functioning of the code are performed here. Since this is a time critical interrupt, only few number of statements must be executed within the interrupt which is fulfilled with a number of if-then-else statements and switch-case constructs.

Case 'H': Move to home

If currently moving to Home 1, and a low voltage is recieved on PB0 Sends DH1 on UART, sets current command to 'N' (representing No Operation).

Else

Moves forward by one minor step If currently moving to Home 2, and a low voltage is recieved on PB3 Sends DH2 on UART, sets current command to 'N' (representing No Operation)

Else

Moves forward by one minor step

```
case 'H':       if(((PINB&0x01)==0) && (home==1))
                        {
                                curr_command='N';
                                return;
                        }

                        if(((PINB&0x80)==0) && (home==2))
                        {
                                curr_command='N';
                                return;
                        }

                        OCR1A=_SIN_TABLE(i);
                        OCR1B=_COS_TABLE(i);
                        i+=step;
                        if(i==256)
                                i=0;
                        break;
```

Case 'R' & 'F': Move forward/reverse

As long as there are any revolutions, major steps, or minor steps left to move, move forward in the sine table by one step

(step = 1 for forward

step = 1 for backward)

If not, sends DF/DR and sets the current command to N

```
case 'R':
case 'F':       if(revolutions>0&&major_step==0)
                        {
                                major_step=50;
                                revolutions--;
                                //if(revolutions==0)
                                        //rev_flag=1;
                        }

                        if(major_step>0)
```

```
                        {
                                if(step_started==0)
                                {
                                        pwmstate = i;
                                        i+=step;
                                        step_started=1;
                                }
                                else
                                {
                                        if(i==pwmstate)
                                        {
                                                step_started=0;
                                                major_step--;
                                        }
                                        else
                                                i+=step;
                                }
                        }
                        else
                        {
                                pwmstate=minor_step;
                                if(i!=pwmstate)
                                        i+=step;
                        }

                        if(curr_command=='R')
                        {
                                if(i==256)
                                        i=0;

                                OCR1B=_SIN_TABLE(i);
                                OCR1A=_COS_TABLE(i);
                        }
                        else
                        {
                                if(i==256)
                                        i=0;
                                OCR1A=_SIN_TABLE(i);
                                OCR1B=_COS_TABLE(i);
                        }

                if((major_step==0)&&(i==minor_step)&&(revolutions==0))
                {
                        if(curr_command=='R')
                        {
                                reverse_effect();
                                minor_step=i;
                        }
                        curr_command='N';
                }
                break;


        default:
                OCR1A=_SIN_TABLE(i);
                OCR1B=_COS_TABLE(i);
```

Case default:

Mostly invoked in case the current command is N (No Operation) Doesn't move either forward or backward in the sine table Maintains the current PWM values indefinitely

Definition at line 161 of file main.c.

References _COS_TABLE, _SIN_TABLE, count_down, curr_command, home, i, major_step, minor_step, pwmstate, reverse_effect(), revolutions, speed, step, and step_started.

3.4.2.2 int main (void)

The main function.

1. Configures hardware PWM on PB1 and PB5
2. Configures PB0 and PB3 to allow external interrupts to occur.
3. Enables all interrupts
4. Enters an infinite loop, waiting for a valid command from UART:

A : Abort

F : Move Forward

R : Move reverse

H : Move to home position

S : Status

V : Set velocity

On receiving a valid command, the corresponding function is called for further processing.

NOTE: If a character other than the valid ones is received, a E is sent, representing error.

Returns:

int

Definition at line 292 of file main.c.

References _CONFIGURE_I_O_PINS, _CONFIGURE_PWM1_CHANNEL, _CONFIGURE_PWM2_CHANNEL, _CONFIGURE_PWM_BASE_FREQ, _DELAY_1000_CYCLES, _ENABLE_GLOBAL_INTERRUPTS, _ENABLE_PLL, _ENABLE_PLL_CLOCK_SOURCE, _ENABLE_PWM1_CHANNEL, _ENABLE_PWM2_CHANNEL, _ENABLE_TIMER1_OVERFLOW_INTERRUPT, _RESET_PORT, _SET_INTERNAL_PULLUPS, _SET_TIMER1_FREQUENCY, _WAIT_TILL_PLL_LOCKED, abort(), forward(), home, init_usart(), r1, recv_char(), reverse(), send_char(), status(), to_home(), and velocity().

3.4.3 Variable Documentation

3.4.3.1 unsigned char count_down = 1

Used to count down from an initial value (= speed) to zero.

Definition at line 41 of file main.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT().

3.4.3.2 unsigned char r1 = 0

This determines which action will be performed.

Definition at line 270 of file main.c.

Referenced by main().

3.5 motor.c File Reference

Functions

- void **velocity** ()
  velocity/speed/PWM_frequency control
- void **forward** ()
  Move the motor in forward direction.
- void **reverse_effect** ()
  Make index conversions to map 'sin to cos' for same value.
- void **reverse** ()
  Move the motor in reverse direction.
- void **to_home** ()
  Move the motor to home position #1 or #2.
- void **abort** ()
  Immediately abort the current operation.
- void **status** ()
  Previous command executed.

Variables

- unsigned char **speed** = '1'
  Prescaling of PLL clock.
- unsigned char **pwmstate** = 0
  current value of PWM duty cycle reffered from the sine table
- int **i** = 0
- unsigned char **home** = '0'
  Status of home position.
- unsigned char **in** = 0
- unsigned char **curr_command** = 'N'
  Current command being executed.
- unsigned char **major_step** = 0
  major steps left to move
- unsigned char **revolutions** = 0
  revolutions left to move
- unsigned char **minor_step** = 0

minor steps left to move

- unsigned char **step_started** = 0
- unsigned char **major_recv** = 0
- char **step** = 1

  traverse by 'step' in sin table at next timer1 overflow interrupt.

- unsigned char **temp** = 0
- unsigned char **diff** = 0

3.5.1 Function Documentation

3.5.1.1 static void abort (void)

Immediately abort the current operation.

Aborts whichever command was last given.

Returns a D followed by an alphabet representing the command that was aborted.

e.g. If the last command was to move forward(F) then on executing abort DF would be sent.

Returns:

void

Definition at line 242 of file motor.c.

References curr_command, major_step, and send_char().

Referenced by main().

3.5.1.2 static void forward (void)

Move the motor in forward direction.

Accepts 3 values from UART:

a) No. of complete revolutions to move forward.

b) No. of major steps to move forward.

c) No. of minor steps to move forward.

Note: 1 complete revolution = 50 major steps

1 major step = 256 minor steps

1 step of stepper = 64 minor steps = 1/4 major step

Each of these 3 values is accepted as an 8 bit unsigned value

Returns:

void

Definition at line 102 of file motor.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_-OVERFLOW_INTERRUPT, curr_command, i, major_recv, major_step, minor_step, recv_char(), revolutions, step, step_started, and temp.

Referenced by main().

3.5.1.3 static void reverse (void)

Move the motor in reverse direction.

Accepts 3 values from UART:

a) No. of complete revolutions to move reverse.

b) No. of major steps to move reverse.

c) No. of minor steps to move reverse.

Note: 1 complete revolution = 50 major steps

1 major step = 256 minor steps

1 step of stepper = 64 minor steps = 1/4 major step

Each of these 3 values is accepted as an 8 bit unsigned value

Returns:

void

Definition at line 182 of file motor.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_-OVERFLOW_INTERRUPT, curr_command, i, major_recv, major_step, minor_step, recv_char(), reverse_effect(), revolutions, step, step_started, and temp.

Referenced by main().

3.5.1.4 static void reverse_effect (void)

Make index conversions to map 'sin to cos' for same value.

Adjusts the current position of the index into the sine table so that if it is currently in the increasing part of the curve, the new value is in the decreasing part , and vice versa. This is required to be done whenever the motor has to reverse its direction of motion.

Returns:

void

Definition at line 138 of file motor.c.

References diff, and i.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), and reverse().

3.5.1.5 static void status (void)

Previous command executed.

Returns the command which is currently being processed. If no command is being executed, a N is sent.

e.g. If the motor is currently moving in the reverse direction, an R is sent

Returns:

void

Definition at line 260 of file motor.c.

References curr_command, and send_char().

Referenced by main().

3.5.1.6 static void to_home (void)

Move the motor to home position #1 or #2.

Accepts a value from USART, which can be either '1' or '2':

1. '1' represents home position 1 : Stops moving when a low voltage is received on PB0.

2. '2' represents home position 2 : Stops moving when a low voltage is received on PB3.

NOTE: In both home 1 and home 2, the stepper moves in forward direction only.

Returns:

void

Definition at line 222 of file motor.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_OVERFLOW_INTERRUPT, curr_command, home, recv_char(), and step.

Referenced by main().

3.5.1.7 static void velocity (void)

velocity/speed/PWM_frequency control

Accepts a number(from UART) between '1' to '9' '1' being the highest and '9' the slowest speed Each speed is half its immediately lesser value so '9' is 1/256th of the fastest available(which is'1')

Returns:

void

Accepts a number(from UART) between '1' to '7' '1' being the highest and '7' the slowest speed Each speed is half its immediately lesser value so '7' is 1/64th of the fastest available(which is'1')

Returns:

void

Definition at line 76 of file motor.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_OVERFLOW_INTERRUPT, recv_char(), and speed.

Referenced by main().

3.5.2 Variable Documentation

3.5.2.1 unsigned char curr_command = 'N'

Current command being executed.

Definition at line 35 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), abort(), forward(), reverse(), status(), and to_home().

3.5.2.2 unsigned char diff = 0

Definition at line 64 of file motor.c.

Referenced by reverse_effect().

3.5.2.3 unsigned char home = '0'

Status of home position.

Definition at line 28 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), main(), and to_home().

3.5.2.4 int i = 0

Definition at line 23 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), forward(), recv_char(), reverse(), reverse_effect(), and send_char().

3.5.2.5 unsigned char in = 0

Definition at line 30 of file motor.c.

3.5.2.6 unsigned char major_recv = 0

Definition at line 54 of file motor.c.

Referenced by forward(), and reverse().

3.5.2.7 unsigned char major_step = 0

major steps left to move

Definition at line 40 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), abort(), forward(), and reverse().

3.5.2.8 unsigned char minor_step = 0

minor steps left to move

Definition at line 50 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), forward(), and reverse().

3.5.2.9 unsigned char pwmstate = 0

current value of PWM duty cycle reffered from the sine table

Definition at line 20 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT().

3.5.2.10 unsigned char revolutions = 0

revolutions left to move

Definition at line 45 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), forward(), and reverse().

3.5.2.11 unsigned char speed = '1'

Prescaling of PLL clock.

Definition at line 15 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), and velocity().

3.5.2.12 char step = 1

traverse by 'step' in sin table at next timer1 overflow interrupt.

Definition at line 60 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), forward(), reverse(), and to_home().

3.5.2.13 unsigned char step_started = 0

Definition at line 52 of file motor.c.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), forward(), and reverse().

3.5.2.14 unsigned char temp = 0

Definition at line 62 of file motor.c.

Referenced by forward(), recv_char(), and reverse().

3.6 motor.h File Reference

Utilities and functions for controlling the stepper motor.

Defines

- #define **_ENABLE_TIMER1_OVERFLOW_INTERRUPT**() TIMSK|=(1<<TOIE1)

  Enable TCCR1.

- #define **_DISABLE_TIMER1_OVERFLOW_INTERRUPT**() TIMSK&=(0<<TOIE1)

  Disable TCCR1.

- #define **_CONFIGURE_TIMER1_SET_SPEED_MODE**() TCCR1=(1<<PWM1A)|(1<<COM1A0)

  Configure the TCCR1 control register for PWM output.

- #define **_SET_TIMER1_FREQUENCY**(n) TCCR1 |= n+3

  Configure the TCCR1 control register to set the timer frequency(n=1 to 7).

Functions

- static void **velocity** (void)

  velocity/speed/PWM_frequency control

- static void **forward** (void)

  Move the motor in forward direction.

- static void **reverse_effect** (void)

  Make index conversions to map 'sin to cos' for same value.

- static void **reverse** (void)

  Move the motor in reverse direction.

- static void **to_home** (void)

  Move the motor to home position #1 or #2.

- static void **abort** (void)

  Immediately abort the current operation.

- static void **status** (void)

  Previous command executed.

3.6.1 Detailed Description

Utilities and functions for controlling the stepper motor.

This file has been prepared for Doxygen automatic documentation generation.

This file contains statements and functions to control the behaviour of the stepper motor.

Definition in file **motor.h**.

3.6.2 Define Documentation

3.6.2.1 #define _CONFIGURE_TIMER1_SET_SPEED_MODE() TCCR1=(1<<PWM1A)|(1<<COM1A0)

Configure the TCCR1 control register for PWM output.

_CONFIGURE_TIMER1_SET_SPEED_MODE() (p. 25)

Configures the TCCR1 control register to enable the PWM mode based on comparator OCR1A in Timer/Counter1.

Clears the OC1A output line after a compare match with OCR1C register value.

Definition at line 38 of file motor.h.

3.6.2.2 #define _DISABLE_TIMER1_OVERFLOW_INTERRUPT() TIMSK&=(0<<TOIE1)

Disable TCCR1.

Disable the Timer/Counter1 Overflow Interrupt.

Definition at line 27 of file motor.h.

Referenced by forward(), init_usart(), recv_char(), reverse(), send_char(), to_home(), and velocity().

3.6.2.3 #define _ENABLE_TIMER1_OVERFLOW_INTERRUPT() TIMSK|=(1<<TOIE1)

Enable TCCR1.

Enable the Timer/Counter1 Overflow Interrupt.

Definition at line 19 of file motor.h.

Referenced by forward(), init_usart(), main(), recv_char(), reverse(), send_char(), to_home(), and velocity().

3.6.2.4 #define _SET_TIMER1_FREQUENCY(n) TCCR1 |= n+3

Configure the TCCR1 control register to set the timer frequency(n=1 to 7).

Configures the TCCR1 control register to define the prescaling source of Timer/Counter1. It determines the speed at which the steeper motor will operate. A lower value of n indicates a higher speed of operation and vice versa.

|__n___|_CS13__|_CS12__|_CS11__|_CS10__|____Prescaling_|

---------- |__1___|___0___|___1___|___0___|___0___|_____CK/8____-
__|
__2___	___0___	___1___	___0___	___1___	_____CK/16_____
__3___	___0___	___1___	___1___	___0___	_____CK/32_____
__4___	___0___	___1___	___1___	___1___	_____CK/64_____
__5___	___1___	___0___	___0___	___0___	_____CK/128____
__6___	___1___	___0___	___0___	___1___	_____CK/256____
__7___	___1___	___0___	___1___	___0___	_____CK/512____

IMPORTANT: For smooth operation of the stepper motor, n must lie within this range (1-7).

Parameters:

n TCCR1 timer frequency.

Definition at line 66 of file motor.h.

Referenced by main().

3.6.3 Function Documentation

3.6.3.1 static void abort (void) [static]

Immediately abort the current operation.

Aborts whichever command was last given.

Returns a D followed by an alphabet representing the command that was aborted.

e.g. If the last command was to move forward(F) then on executing abort DF would be sent.

Returns:

void

Definition at line 242 of file motor.c.

References curr_command, major_step, and send_char().

Referenced by main().

3.6.3.2 static void forward (void) [static]

Move the motor in forward direction.

Accepts 3 values from UART:

a) No. of complete revolutions to move forward.

b) No. of major steps to move forward.

c) No. of minor steps to move forward.

Note: 1 complete revolution = 50 major steps

1 major step = 256 minor steps

1 step of stepper = 64 minor steps = 1/4 major step

Each of these 3 values is accepted as an 8 bit unsigned value

Returns:

void

Definition at line 102 of file motor.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_OVERFLOW_INTERRUPT, curr_command, i, major_recv, major_step, minor_step, recv_char(), revolutions, step, step_started, and temp.

Referenced by main().

3.6.3.3 static void reverse (void) [static]

Move the motor in reverse direction.

Accepts 3 values from UART:

a) No. of complete revolutions to move reverse.

b) No. of major steps to move reverse.

c) No. of minor steps to move reverse.

Note: 1 complete revolution = 50 major steps

1 major step = 256 minor steps

1 step of stepper = 64 minor steps = 1/4 major step

Each of these 3 values is accepted as an 8 bit unsigned value

Returns:

void

Definition at line 182 of file motor.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_OVERFLOW_INTERRUPT, curr_command, i, major_recv, major_step, minor_step, recv_char(), reverse_effect(), revolutions, step, step_started, and temp.

Referenced by main().

3.6.3.4 static void reverse_effect (void) [static]

Make index conversions to map 'sin to cos' for same value.

Adjusts the current position of the index into the sine table so that if it is currently in the increasing part of the curve, the new value is in the decreasing part , and vice versa. This is required to be done whenever the motor has to reverse its direction of motion.

Returns:

void

Definition at line 138 of file motor.c.

References diff, and i.

Referenced by _TIMER1_OVERFLOW_INTERRUPT(), and reverse().

3.6.3.5 static void status (void) [static]

Previous command executed.

Returns the command which is currently being processed. If no command is being executed, a N is sent.

e.g. If the motor is currently moving in the reverse direction, an R is sent

Returns:

void

Definition at line 260 of file motor.c.

References curr_command, and send_char().

Referenced by main().

3.6.3.6 static void to_home (void) [static]

Move the motor to home position #1 or #2.

Accepts a value from USART, which can be either '1' or '2':

1. '1' represents home position 1 : Stops moving when a low voltage is received on PB0.

2. '2' represents home position 2 : Stops moving when a low voltage is received on PB3.

NOTE: In both home 1 and home 2, the stepper moves in forward direction only.

Returns:

void

Definition at line 222 of file motor.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_OVERFLOW_INTERRUPT, curr_command, home, recv_char(), and step.

Referenced by main().

3.6.3.7 static void velocity (void) [static]

velocity/speed/PWM_frequency control

Accepts a number(from UART) between '1' to '9' '1' being the highest and '9' the slowest speed Each speed is half its immediately lesser value so '9' is 1/256th of the fastest available(which is'1')

Returns:

void

Accepts a number(from UART) between '1' to '7' '1' being the highest and '7' the slowest speed Each speed is half its immediately lesser value so '7' is 1/64th of the fastest available(which is'1')

Returns:

void

Definition at line 76 of file motor.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_-OVERFLOW_INTERRUPT, recv_char(), and speed.

Referenced by main().

3.7 tables.h File Reference

Contains a 256 value sine table to be stored in the flash memory.

3.7.1 Detailed Description

Contains a 256 value sine table to be stored in the flash memory.

This file has been prepared for Doxygen automatic documentation generation.

Definition in file **tables.h**.

3.8 uart.c File Reference

Functions

- void **bit_delay** (void)

  Function to implement a one bit delay.

- void **half_bit_delay** (void)

  Function to implement a half bit delay.

- void **init_usart** (void)

  Function to initialize the software UART.

- void **send_char** (unsigned char value)

  Function to transmit an eight bit unsigned value from TXPIN.

- unsigned char **recv_char** (void)

  Function to recieve an eight bit unsigned value from RXPIN.

3.8.1 Function Documentation

3.8.1.1 void bit_delay (void)

Function to implement a one bit delay.

Returns:

void

Definition at line 18 of file uart.c.

References _TICKS2WAITONE.

Referenced by recv_char(), and send_char().

3.8.1.2 void half_bit_delay (void)

Function to implement a half bit delay.

Returns:

void

Definition at line 32 of file uart.c.

References _TICKS2WAITHALF.

Referenced by recv_char().

3.8.1.3 void init_usart (void)

Function to initialize the software UART.

This function will set up pins to transmit and receive on, and initialize control register of Timer0.

Returns:

void

Definition at line 49 of file uart.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_OVERFLOW_INTERRUPT, _RX_PIN, _RX_PIN_DDR, _TX_PIN, _TX_PIN_DDR, IDLE, and state.

Referenced by main(), recv_char(), and send_char().

3.8.1.4 unsigned char recv_char (void)

Function to recieve an eight bit unsigned value from RXPIN.

This function will recieve an eight bit unsigned value through the soft-UART from the microcontroller's RXPIN

Returns:

unsigned char

Definition at line 130 of file uart.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_OVERFLOW_INTERRUPT, bit_delay(), half_bit_delay(), i, IDLE, init_usart(), RXING, state, and temp.

Referenced by forward(), main(), reverse(), to_home(), and velocity().

3.8.1.5 void send_char (unsigned char *value*)

Function to transmit an eight bit unsigned value from TXPIN.

This function will transmit an eight bit unsigned value through the soft-UART from the microcontroller's TXPIN

Parameters:

value

Returns:

void

This function will transmit an eight bit unsigned value through the soft-UART from the microcontroller's TXPIN

Parameters:

unsigned char value

Returns:

void

Definition at line 74 of file uart.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_OVERFLOW_INTERRUPT, bit_delay(), i, IDLE, init_usart(), state, and TXING.

Referenced by abort(), main(), and status().

3.9 uart.h File Reference

This file implements a half duplex software driven uart.

Defines

- #define **_BR_9600**
  Baud Rate.
- #define **_TICKS2COUNT** 100
  Ticks between two bits.
- #define **_TICKS2WAITONE** 100
  Wait one bit period.
- #define **_TICKS2WAITHALF** 50
  Wait half bit period.
- #define **_SYS_FREQ** 8
  System clock frequency. This is the system clock frequency in MHz.
- #define **_TX_PIN** PB5
  The pin which is to be configured as the soft-uART TXD(transmit) pin.
- #define **_TX_PIN_DDR** DDB5
- #define **_RX_PIN** PB2
  The pin which is to be configured as the soft-uART RXD(receive) pin.
- #define **_RX_PIN_DDR** DDB2

Enumerations

- enum **uartstate_t** { **IDLE**, **TXING**, **RXING** }
  Type defined enumeration holding software UART's state.

Functions

- void **init_usart** (void)
  Function to initialize the software UART.
- void **send_char** (unsigned char value)
  Function to transmit an eight bit unsigned value from TXPIN.
- unsigned char **recv_char** (void)
  Function to recieve an eight bit unsigned value from RXPIN.

Variables

- static volatile **uartstate_t state**

  Holds the state of the UART.

3.9.1 Detailed Description

This file implements a half duplex software driven uart.

This file has been prepared for Doxygen automatic documentation generation.

If other operating voltages and/or temperatures than 5 Volts and 25C are desired, consider calibrating the internal oscillator.

Definition in file **uart.h**.

3.9.2 Define Documentation

3.9.2.1 #define _BR_9600

Baud Rate.

This is the baudrate at which the software UART will operate. Please choose one, and comment the others. Desired baudrate...choose one, comment the others.

Definition at line 17 of file uart.h.

3.9.2.2 #define _RX_PIN PB2

The pin which is to be configured as the soft-uART RXD(receive) pin.

Definition at line 54 of file uart.h.

Referenced by init_usart().

3.9.2.3 #define _RX_PIN_DDR DDB2

Definition at line 56 of file uart.h.

Referenced by init_usart().

3.9.2.4 #define _SYS_FREQ 8

System clock frequency. This is the system clock frequency in MHz.

Definition at line 42 of file uart.h.

3.9.2.5 #define _TICKS2COUNT 100

Ticks between two bits.

Definition at line 23 of file uart.h.

3.9.2.6 #define _TICKS2WAITHALF 50

Wait half bit period.

Definition at line 25 of file uart.h.

Referenced by half_bit_delay().

3.9.2.7 #define _TICKS2WAITONE 100

Wait one bit period.

Definition at line 24 of file uart.h.

Referenced by bit_delay().

3.9.2.8 #define _TX_PIN PB5

The pin which is to be configured as the soft-uART TXD(transmit) pin.

Definition at line 47 of file uart.h.

Referenced by init_usart().

3.9.2.9 #define _TX_PIN_DDR DDB5

Definition at line 49 of file uart.h.

Referenced by init_usart().

3.9.3 Enumeration Type Documentation

3.9.3.1 enum uartstate_t

Type defined enumeration holding software UART's state.

Enumerator:

IDLE Idle state, both transmit and receive possible.

TXING Transmitting byte.

RXING Receiving byte.

Definition at line 61 of file uart.h.

3.9.4 Function Documentation

3.9.4.1 void init_usart (void)

Function to initialize the software UART.

This function will set up pins to transmit and receive on, and initialize control register of Timer0.

Returns:

void

Definition at line 49 of file uart.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_-OVERFLOW_INTERRUPT, _RX_PIN, _RX_PIN_DDR, _TX_PIN, _TX_PIN_DDR, IDLE, and state.

Referenced by main(), recv_char(), and send_char().

3.9.4.2 unsigned char recv_char (void)

Function to recieve an eight bit unsigned value from RXPIN.

This function will recieve an eight bit unsigned value through the soft-UART from the microcontroller's RXPIN

Returns:

unsigned char

Definition at line 130 of file uart.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_-OVERFLOW_INTERRUPT, bit_delay(), half_bit_delay(), i, IDLE, init_usart(), RXING, state, and temp.

Referenced by forward(), main(), reverse(), to_home(), and velocity().

3.9.4.3 void send_char (unsigned char *value*)

Function to transmit an eight bit unsigned value from TXPIN.

This function will transmit an eight bit unsigned value through the soft-UART from the microcontroller's TXPIN

Parameters:

value

Returns:

void

This function will transmit an eight bit unsigned value through the soft-UART from the microcontroller's TXPIN

Parameters:

unsigned char value

Returns:

void

Definition at line 74 of file uart.c.

References _DISABLE_TIMER1_OVERFLOW_INTERRUPT, _ENABLE_TIMER1_-OVERFLOW_INTERRUPT, bit_delay(), i, IDLE, init_usart(), state, and TXING.

Referenced by abort(), main(), and status().

3.9.5 Variable Documentation

3.9.5.1 volatile uartstate_t state [static]

Holds the state of the UART.

Definition at line 68 of file uart.h.

Referenced by init_usart(), recv_char(), and send_char().