RBELib

2.0

Generated by Doxygen 1.8.5

Wed Aug 6 2014 15:19:46

Contents

Chapter 1

3001 RBELib

1.1 Welcome to RBE 3001

Here you will find all the documentation that you need for working with RBELib which will contain function prototypes for everything that you will need throughout the course. While it is possible to complete the course without using RBELib, using it will make your code easier to maintain as you get closer to the final project as well as follow proper coding practice as outlined in the syllabus.

As you progress through the course, it is encouraged that you keep your own version of RBELib with your SVN repository so that you can add to things such as the To Do list and function descriptions as you go.

Please note that all page numbers are for the

ATmega164P/V

ATmega324P/V

ATmega644P/V

datasheet.

1.2 RBELib Files

ADC **Page 240**

Here you will find everything relating to the Analog to Digital Converter (ADC) that you should need. You will have to make your own corresponding functions for these prototypes.

Ports **Page 72**

Ports contains the code for manipulating the ports on the chip that you can access via the breakouts. Keep in mind that some of these are also connected to components on the board such as the SPI line, if you have a problem using a pin and on-board components, **CHECK THE DATASHEET AND BOARD LAYOUT** to make sure that you are not using the same pin for your buttons that you are for the MOSI while using SPI or etc.

If you are not using a port, do not leave wires connected. After lab 1, instead of running code off buttons (if you decide to do that) instead use the UART to receive data and make a menu.

Printing

This contains initRBELib() which must be called if you want to use print statements with printf(). Whenever you have a problem and can't find out what it is, first try to print out all of your variables / registers and add delays so that you can see what is happening.

You need to have putCharDebug() written before you can use printf().

Peripherals **See respective datasheets for each peripheral.**

Here is where all code for peripheral devices such as the IR sensor and accelerometer go.

PID

This is where all PID code goes for the calculation. You will need to create your own calculation using the formulas that you used in class and optimize them for running on an embedded system.

This also contains a struct you can use for defining your constants.

RBELib Macros

Here are all of the includes for RBELib as well as some of the macros that you may find useful to use (such as INPUT/-OUTPUT instead of 0/1).

Reg_Structs Slave Selects

Reg_structs are a few useful shorthand notations that you can use in your coding for accessing the pins on ports. To go with this, SlaveSelects defines some of the SS lines for when using SPI.

Servo (Conveyer/Gripper)

This is used for moving the conveyer belt and opening/closing the gripper. You do not need to create anything additional.

SPI **Page 161**

For initializing the SPI and and sending/receiving data through it.

Timers **Pages 93, 111, 139**

Allows you to initialize any one of the timers and set their comparative values for when they reset if using CTC mode.

USART **Page 171**

This should be the very first thing that you work on and is the prelab assignement. This allows you to use serial printing within your code through the use of the USART.

Again, if you don't do putCharDebug() first, you can't use print statements and **RBELib may not compile unless you at least create a blank function.**

Potentiometers

These functions can be used to get the potentiometer values in degrees, voltage, or ADC counts. You need to make your own functions for the prototypes.

Motors

This contains declarations for controlling the motors on the arm. You need to create a way to drive to an (X,Y) coordinate as well as a way to drive to a desired angle for the links.

Potentiometers

These functions can be used to get the potentiometer values in degrees, voltage, or ADC counts. You need to make your own functions for the prototypes.

1.3 Other Helpful Links

Bug List

This is a place where any bugs in RBELib and your code should be documented.

To Do

Things that still need to be done in the RBELib and or your code.

Data Types

This lets you know the number of bytes in any given data type.

Chapter 2

Data Types

2.1 1 Byte

char

2.2 2 Bytes

short

int

2.3 4 Bytes

long

float

double

long double

2.4 8 Bytes

long long

Chapter 3

Todo List

Global calcPID (char link, int setPoint, int actPos)

Make a function to calculate the PID value for a link.

Global changeADC (int channel)

Create a way to switch ADC channels if you are using interrupts.

Global clearADC (int channel)

Create the corresponding function to clear the last ADC calculation register and disconnect the input to the ADC if desired.

Global debugUSARTInit (unsigned long baudrate)

Create the function that will initialize USART1 for debugging use.

Global driveLink (int link, int dir)

Create a way to drive either link in any direction.

Global encCount (int chan)

Find the current encoder ticks on a given channel.

Global encInit (int chan)

Make a function that can setup both encoder chips on the board.

Global getAccel (int axis)

Create a function that is able to find the acceleration of a given axis.

Global getADC (int channel)

Create the corresponding function to obtain the value of the last calculation if you are using polling.

Global getCharDebug (void)

Make the function that will listen for input on the USART1 RX line.

Global getPinsVal (char port, int numPins,...)

Create a way to read all given pins on a port.

Global gotoAngles (int lowerTheta, int upperTheta)

Make a way to drive the links to a desired angle.

Global gotoXY (int x, int y)

Use kinematic equations to move the end effector to the desired position.

Global homePos ()

Drive the arm to a known position using the potentiometers.

Global initADC (int channel)

Create the corresponding function to initialize the ADC using the channel parameter.

Global initSPI ()

Create the function that will allow you to initialize the SPI in a mode compatible with all devices. Do not forget to deassert all of your SS lines!

Global initTimer (int timer, int mode, unsigned int comp)

Create a function that initializes the desired timer in a given mode and set the compare value (as appropriate).

Global IRDist (int chan)

Make a function that is able to get the ADC value of the IR sensor.

Global pidConsts

Again, do not forget to use

Global potAngle (int pot)

Calculate the angle using the ADC reading.

Global potVolts (int pot)

Convert the ADC value into a voltage in mV (so no floats needed).

Global putCharDebug (char byteToSend)

Make the function that will put a character on the USART1 TX line.

Global resetEncCount (int chan)

Clear the encoder count (set to 0).

Global setCompValue (unsigned char timer, unsigned short int comp)

Create a function that will set a new compare value for the given timer.

Global setConst (char link, float Kp, float Ki, float Kd)

Create a function to the the PID constants for a given link.

Global setDAC (int DACn, int SPIVal)

Make the function that is able to set the DAC to a given value from 0 - 4095.

Global setPinsDir (char port, int dir, char numPins,...)

Create a way to set a port's pins to inputs or outputs.

Global setPinsVal (char port, int numPins,...)

Create a way to read all given pins on a port.

Global spiTransceive (BYTE data)

Make a function that will send a byte of data through the SPI and return whatever was sent back.

Global stopMotors ()

Create way to stop the motors using the DAC.

Chapter 4

Bug List

File SPI.h

While not a bug, some students have problems with some of the SPI devices where their code will not work after they send a command. To fix this, you may have to toggle the SS line once after sending your command and then disable it once more. This is because some of the devices need the toggle to load the registers and then execute the command.

While not a bug, the DAC SS may need to be toggled at startup. This is only something that matters during a soft reset but should be done anyway during your initialization for SPI.

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Data Structure Documentation

7.1 __8bitreg_t Struct Reference

Generic byte register.

```
#include <reg_structs.h>
```

Data Fields

- unsigned _P0:1
- unsigned _P1:1
- unsigned _P2:1
- unsigned _P3:1
- unsigned _P4:1
- unsigned _P5:1
- unsigned _P6:1
- unsigned _P7:1

7.1.1 Detailed Description

Generic byte register.

Used for controlling the pins on the ports.

Definition at line 26 of file reg_structs.h.

7.1.2 Field Documentation

7.1.2.1 unsigned __8bitreg_t::_P0

Definition at line 27 of file reg_structs.h.

7.1.2.2 unsigned __8bitreg_t::_P1

Definition at line 28 of file reg_structs.h.

7.1.2.3 unsigned __8bitreg_t::_P2

Definition at line 29 of file reg_structs.h.

7.1.2.4 unsigned __8bitreg_t::_P3

Definition at line 30 of file reg_structs.h.

7.1.2.5 unsigned __8bitreg_t::_P4

Definition at line 31 of file reg_structs.h.

7.1.2.6 unsigned __8bitreg_t::_P5

Definition at line 32 of file reg_structs.h.

7.1.2.7 unsigned __8bitreg_t::_P6

Definition at line 33 of file reg_structs.h.

7.1.2.8 unsigned __8bitreg_t::_P7

Definition at line 34 of file reg_structs.h.

The documentation for this struct was generated from the following file:

- include/RBELib/reg_structs.h

7.2 __SPIPORTbits_t Struct Reference

SPI port.

```
#include <reg_structs.h>
```

Data Fields

- unsigned __pad0__:5
- unsigned _MOSI:1
- unsigned _MISO:1
- unsigned _SCK:1

7.2.1 Detailed Description

SPI port.

MOSI, MISO and CLK pins.

Definition at line 42 of file reg_structs.h.

7.2.2 Field Documentation

7.2.2.1 unsigned __SPIPORTbits_t::__pad0__

Definition at line 43 of file reg_structs.h.

7.2.2.2 unsigned __SPIPORTbits_t::_MISO

Definition at line 45 of file reg_structs.h.

7.2.2.3 unsigned __SPIPORTbits_t::_MOSI

Definition at line 44 of file reg_structs.h.

7.2.2.4 unsigned __SPIPORTbits_t::_SCK

Definition at line 46 of file reg_structs.h.

The documentation for this struct was generated from the following file:

- include/RBELib/reg_structs.h

7.3 pidConst Struct Reference

PID constants.

```
#include <PID.h>
```

Data Fields

- float Kp_H

  Upper link Kp.
- float Ki_H

  Upper link Ki.
- float Kd_H

  Upper link Kd.
- float Kp_L

  Lower link Kp.
- float Ki_L

  Lower link Ki.
- float Kd_L

  Lower link Kd.

7.3.1 Detailed Description

PID constants.

Obtain value using:

```
pidConsts.Kp_H;
```

for the value desired.

Do not forget to use

```
pidConst pidConsts;
```

in any file you access them in!

Definition at line 21 of file PID.h.

7.3.2 Field Documentation

7.3.2.1 float pidConst::Kd_H

Upper link Kd.

Definition at line 33 of file PID.h.

7.3.2.2 float pidConst::Kd_L

Lower link Kd.

Definition at line 45 of file PID.h.

7.3.2.3 float pidConst::Ki_H

Upper link Ki.

Definition at line 29 of file PID.h.

7.3.2.4 float pidConst::Ki_L

Lower link Ki.

Definition at line 41 of file PID.h.

7.3.2.5 float pidConst::Kp_H

Upper link Kp.

Definition at line 25 of file PID.h.

7.3.2.6 float pidConst::Kp_L

Lower link Kp.

Definition at line 37 of file PID.h.

The documentation for this struct was generated from the following file:

- include/RBELib/PID.h

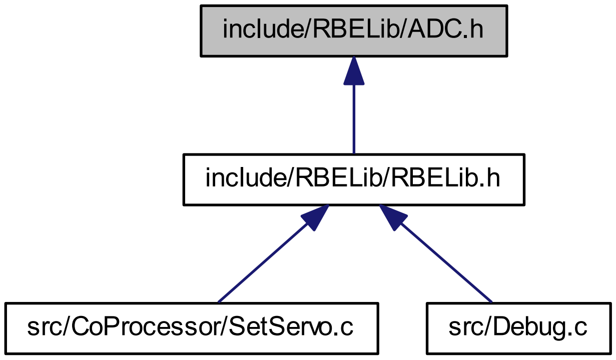
Chapter 8

File Documentation

8.1 include/RBELib/ADC.h File Reference

The header file and function prototypes for the ADC.

This graph shows which files directly or indirectly include this file:



Functions

- void initADC (int channel)

  Initializes the ADC and make one channel active. You can choose to use either interrupts or polling to read the desired channel.
- void clearADC (int channel)

  Disables ADC functionality and clears any saved values (globals).
- unsigned short getADC (int channel)

  Run a conversion on and get the analog value from one ADC channel if using polling.
- void changeADC (int channel)

  Change the channel the ADC is sampling if using interrupts.

8.1.1 Detailed Description

The header file and function prototypes for the ADC. For single ended conversion, the ADC value an be found from the voltage using:

$$\frac{V_{in} * 1024}{V_{ref}}$$

Author

Kevin Harrington

Date

February 11, 2010

Author

Justin Barrett

Date

August 23, 2011

Author

Eric Willcox

Date

August 19, 2013

Definition in file ADC.h.

8.1.2 Function Documentation

8.1.2.1 void changeADC (int *channel*)

Change the channel the ADC is sampling if using interrupts.

Parameters

channel	The ADC channel to switch to.

Todo Create a way to switch ADC channels if you are using interrupts.

8.1.2.2 void clearADC (int *channel*)

Disables ADC functionality and clears any saved values (globals).

Parameters

channel	The ADC channel to disable.

Todo Create the corresponding function to clear the last ADC calculation register and disconnect the input to the ADC if desired.

8.1.2.3 unsigned short getADC (int *channel*)

Run a conversion on and get the analog value from one ADC channel if using polling.

Parameters

channel	The ADC channel to run a conversion on.

Returns

adcVal The 8-10 bit value returned by the ADC conversion. The precision depends on your settings and how much accuracy you desire.

Todo Create the corresponding function to obtain the value of the last calculation if you are using polling.

8.1.2.4 void initADC (int *channel*)

Initializes the ADC and make one channel active. You can choose to use either interrupts or polling to read the desired channel.

Parameters

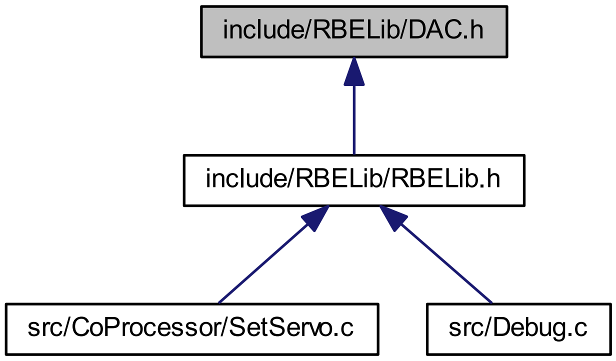
channel	The ADC channel to initialize.

Todo Create the corresponding function to initialize the ADC using the channel parameter.

8.2 include/RBELib/DAC.h File Reference

The header file and function prototypes for the DAC.

This graph shows which files directly or indirectly include this file:



Functions

- void setDAC (int DACn, int SPIVal)

  Set the DAC to the given value on the chosen channel.

8.2.1 Detailed Description

The header file and function prototypes for the DAC. Use these functions to control the behavior of the DAC.

Author

Eric Willcox

Date

August 21, 2013

Definition in file DAC.h.

8.2.2 Function Documentation

8.2.2.1 void setDAC (int *DACn*, int *SPIVal*)

Set the DAC to the given value on the chosen channel.

Parameters

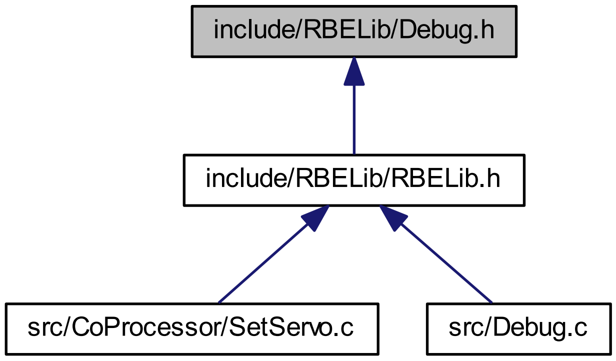
DACn	The channel that you want to set.
SPIVal	The value you want to set it to.

Todo Make the function that is able to set the DAC to a given value from 0 - 4095.

8.3 include/RBELib/Debug.h File Reference

Allows for the use of printf() on the 3001 board. Simply rebinds stdout to the UART.

This graph shows which files directly or indirectly include this file:



Functions

- int printfRBE (char var, FILE ∗stream)

  Calls the students function 'putCharDebug()' to output the stream to. You should not call this function directly, instead use the standard function printf().
- void initRBELib ()

  Rebinds stdout to call printfRBE() and initializes communication with the coprocessor. This function should be called once at the start of your code once you have putCharDebug() written in Lab 1. If you do not call this, printf() and SetServo() will not work.

8.3.1 Detailed Description

Allows for the use of printf() on the 3001 board. Simply rebinds stdout to the UART.

Author

Eric Willcox

Date

July 9, 2014

Definition in file Debug.h.

8.3.2 Function Documentation

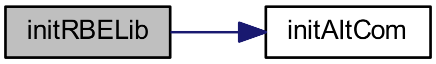
8.3.2.1 void initRBELib ( )

Rebinds stdout to call printfRBE() and initializes communication with the coprocessor. This function should be called once at the start of your code once you have putCharDebug() written in Lab 1. If you do not call this, printf() and SetServo() will not work.

Definition at line 20 of file Debug.c.

References initAltCom().

Here is the call graph for this function:



8.3.2.2 int printfRBE (char *var,* FILE ∗ *stream*)

Calls the students function 'putCharDebug()' to output the stream to. You should not call this function directly, instead use the standard function printf().

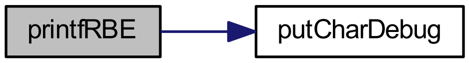
Parameters

var	Character to output
∗*stream*	Place to put the character (stdout)

Definition at line 14 of file Debug.c.

References putCharDebug().

Here is the call graph for this function:



8.4 include/RBELib/doxy_pages/datatypes.h File Reference

8.4.1 Detailed Description

Tells some of the common length of datatypes for your use. You can get this same data by doing

```
sizeOf(<datatype>);
```

and printing out the value.

Date

Aug 28, 2013

Author

Eric Willcox

Definition in file datatypes.h.

8.5 include/RBELib/doxy_pages/index.h File Reference

This is the index file for Doxygen that is used to link to everything.

8.5.1 Detailed Description

This is the index file for Doxygen that is used to link to everything.

Author

Kevin Harrington

Date

February 21, 2010

Author

Eric Willcox

Date

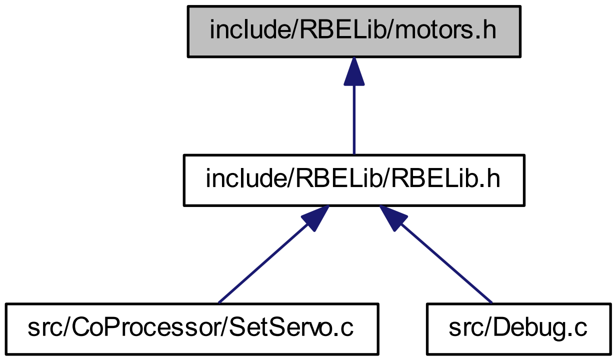
July 21, 2014

Definition in file index.h.

8.6 include/RBELib/motors.h File Reference

Motor driving functions for the arm.

This graph shows which files directly or indirectly include this file:



Functions

- void stopMotors ()

  Helper function to stop the motors on the arm.
- void gotoAngles (int lowerTheta, int upperTheta)

  Drive the arm to a desired angle.
- void gotoXY (int x, int y)

  Drive the end effector of the arm to a desired X and Y position in the workspace.
- void driveLink (int link, int dir)

  Drive a link (upper or lower) in a desired direction.
- void homePos ()

  Drive the arm to a "home" position using the potentiometers. This should be called before using the encoders and just goes to a default position. Once this has been called once, you can initialize/clear the encoders.

8.6.1 Detailed Description

Motor driving functions for the arm.

Author

Eric Willcox

Date

July 9, 2014

Definition in file motors.h.

8.6.2 Function Documentation

8.6.2.1 void driveLink (int *link,* int *dir*)

Drive a link (upper or lower) in a desired direction.

Parameters

link	Which link to control.
dir	Which way to drive the link.

Todo Create a way to drive either link in any direction.

8.6.2.2 void gotoAngles (int *lowerTheta,* int *upperTheta*)

Drive the arm to a desired angle.

Parameters

lowerTheta	The desired angle for the lower link.
upperTheta	The desired angle for the upper link.

Todo Make a way to drive the links to a desired angle.

8.6.2.3 void gotoXY (int *x,* int *y*)

Drive the end effector of the arm to a desired X and Y position in the workspace.

Parameters

x	The desired x position for the end effector.
y	The desired y position for the end effector.

Todo Use kinematic equations to move the end effector to the desired position.

8.6.2.4 void homePos ( )

Drive the arm to a "home" position using the potentiometers. This should be called before using the encoders and just goes to a default position. Once this has been called once, you can initialize/clear the encoders.

Todo Drive the arm to a known position using the potentiometers.

8.6.2.5 void stopMotors ( )

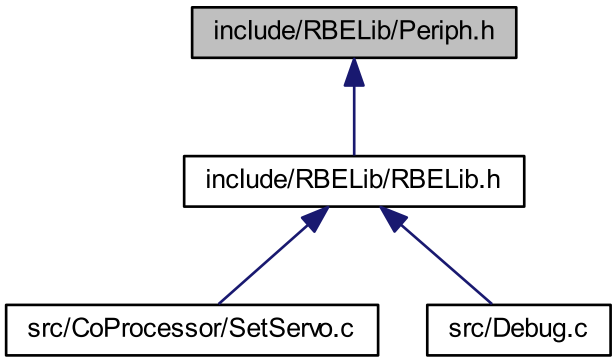
Helper function to stop the motors on the arm.

Todo Create way to stop the motors using the DAC.

8.7 include/RBELib/Periph.h File Reference

The header file and function prototypes for the peripherals (IR, encoder and accelerometer).

This graph shows which files directly or indirectly include this file:



Functions

- signed int getAccel (int axis)

  Find the acceleration in the given axis (X, Y, Z).
- int IRDist (int chan)

  Read an IR sensor and calculate the distance of the block.
- void encInit (int chan)

  Initialize the encoders with the desired settings.
- void resetEncCount (int chan)

  Reset the current count of the encoder ticks.
- signed long encCount (int chan)

  Finds the current count of one of the encoders.

8.7.1 Detailed Description

The header file and function prototypes for the peripherals (IR, encoder and accelerometer). Each of these functions is for controlling the peripheral devices of the arm.

Author

Eric Willcox

Date

August 21, 2013
July 18, 2014

Definition in file Periph.h.

8.7.2 Function Documentation

8.7.2.1 signed long encCount (int *chan*)

Finds the current count of one of the encoders.

Parameters

chan	Channel that the encoder is on that you would like to read.

Returns

count The current count of the encoder.

Todo Find the current encoder ticks on a given channel.

8.7.2.2 void encInit (int *chan*)

Initialize the encoders with the desired settings.

Parameters

chan	Channel to initialize.

Todo Make a function that can setup both encoder chips on the board.

8.7.2.3 signed int getAccel (int *axis*)

Find the acceleration in the given axis (X, Y, Z).

Parameters

axis	The axis that you want to get the measurement of.

Returns

gVal Value of acceleration.

Todo Create a function that is able to find the acceleration of a given axis.

8.7.2.4 int IRDist (int *chan*)

Read an IR sensor and calculate the distance of the block.

Parameters

chan	The port that the IR sensor is on.

Returns

value The distance the block is from the sensor.

Todo Make a function that is able to get the ADC value of the IR sensor.

8.7.2.5 void resetEncCount (int *chan*)

Reset the current count of the encoder ticks.

Parameters

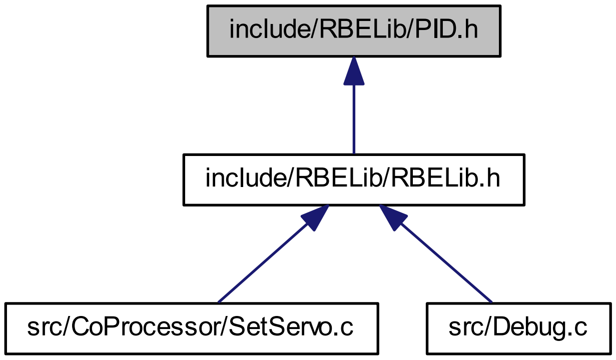
chan	The channel to clear.

Todo Clear the encoder count (set to 0).

8.8 include/RBELib/PID.h File Reference

The header file for PID constants and calculations.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct pidConst

  PID constants.

Functions

- void setConst (char link, float Kp, float Ki, float Kd)

  Sets the Kp, Ki, and Kd values for 1 link.
- signed int calcPID (char link, int setPoint, int actPos)

  Calculate the PID value.

Variables

- pidConst pidConsts

  Declaration for use in other files.

8.8.1 Detailed Description

The header file for PID constants and calculations. Sets the PID constants and calculate the PID value.

Author

Eric Willcox

Date

August 17 2013

Definition in file PID.h.

8.8.2 Function Documentation

8.8.2.1 signed int calcPID (char *link*, int *setPoint*, int *actPos*)

Calculate the PID value.

Parameters

link	Which link to calculate the error for (Use 'U' and 'L').
setPoint	The desired position of the link.
actPos	The current position of the link.

Todo Make a function to calculate the PID value for a link.

8.8.2.2 void setConst (char *link*, float *Kp*, float *Ki*, float *Kd*)

Sets the Kp, Ki, and Kd values for 1 link.

to set the values, use the following style

```
pidConst.Kp = 1.3;
```

Parameters

link	The link you want to set the values for (H or L).
Kp	Proportional value.
Ki	Integral value.
Kd	Derivative value.

Todo Create a function to the the PID constants for a given link.

8.8.3 Variable Documentation

8.8.3.1 pidConst pidConsts

Declaration for use in other files.

Todo Again, do not forget to use

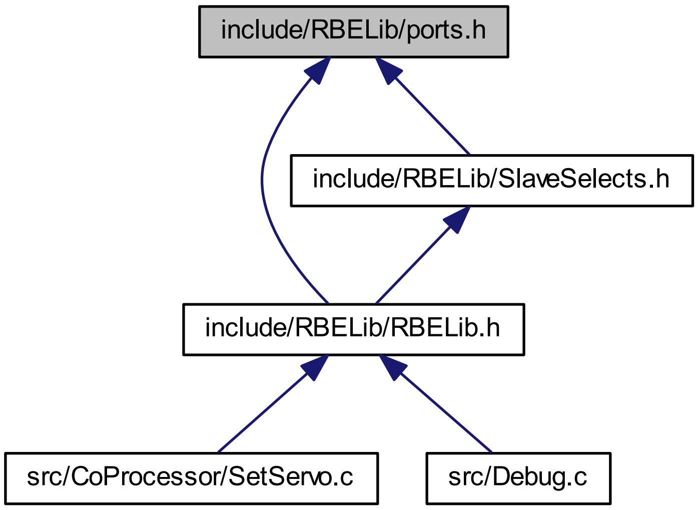
```
pidConst pidConsts;
```

in any file you access them in!

8.9 include/RBELib/ports.h File Reference

Controls ports A - D to be able to set direction, read a value and set a value for any pins desired.

This graph shows which files directly or indirectly include this file:



Functions

- void setPinsDir (char port, int dir, char numPins,...)

    Sets the direction (Input/Output) of the specified pins.
- unsigned char getPinsVal (char port, int numPins,...)

    Sets the given value on the specified pins of a port.
- void setPinsVal (char port, int numPins,...)

    Sets the value on the specified pins of a port.

8.9.1 Detailed Description

Controls ports A - D to be able to set direction, read a value and set a value for any pins desired. All functions here can take in a variable number of pins to set/read from. The way this is accomplished in C is by using the functions provided by stdarg.h and is the way that printf() works in the standard library.

While an array could be used just as well, a variable number of arguments is used to give exposure to variable arguments.

An example of how to call a function with variable arguments can look like this for calling setPinsDir().

```
setPinsDir('A', INPUT, 2, 0, 5)
```

which sets 2 pins on Port A as inputs: 0 and 5.

An example has been created on the RBE wiki and can be found `here.`

Author

Eric Willcox

Date

July 18, 2014

Definition in file ports.h.

8.9.2 Function Documentation

8.9.2.1 unsigned char getPinsVal (char *port,* int *numPins,* *...*)

Sets the given value on the specified pins of a port.

Parameters

port	Port to read (A/B/C/D).
numPins	The number of pins that you are reading.
...	The pins one after another.

Returns

value The value of the specified pins on the port.

Todo Create a way to read all given pins on a port.

8.9.2.2 void setPinsDir (char *port,* int *dir,* char *numPins,* *...*)

Sets the direction (Input/Output) of the specified pins.

Parameters

port	Port to set (A/B/C/D).
dir	The pin on PORTA to set the direction of.
numPins	The number of pins that you are setting the direction of.
...	Pins one after another

Todo Create a way to set a port's pins to inputs or outputs.

8.9.2.3 void setPinsVal (char *port,* int *numPins,* *...*)

Sets the value on the specified pins of a port.

Parameters

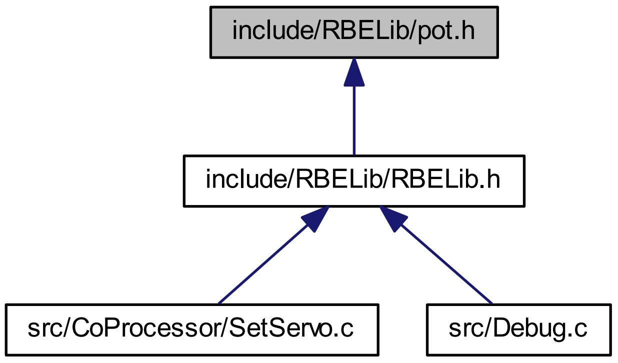
port	Port to read (A/B/C/D).
numPins	The number of pins that you are setting.
...	The pins one after another.

Todo Create a way to read all given pins on a port.

8.10 include/RBELib/pot.h File Reference

The header file and function prototypes for the potentiometers.

This graph shows which files directly or indirectly include this file:



Functions

- int potAngle (int pot)

  Find the angle of the given potentiometer.
- int potVolts (int pot)

  Find the voltage value of the given potentiometer.

8.10.1 Detailed Description

The header file and function prototypes for the potentiometers. Use these functions to read the values from the pots.

Author

Eric Willcox

Date

August 17 2013

Definition in file pot.h.

8.10.2 Function Documentation

8.10.2.1 int potAngle (int *pot*)

Find the angle of the given potentiometer.

Parameters

pot	The pot to check.

Returns

angle Angle of the potentiometer.

Todo Calculate the angle using the ADC reading.

8.10.2.2 int potVolts (int *pot*)

Find the voltage value of the given potentiometer.

Parameters

pot	The pot to get the value of.

Returns

volts Voltage of potentiometer.

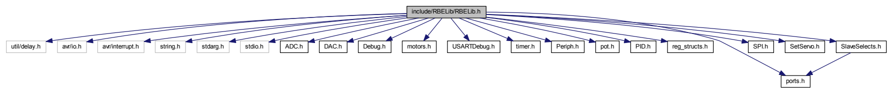
Todo Convert the ADC value into a voltage in mV (so no floats needed).

8.11 include/RBELib/RBELib.h File Reference

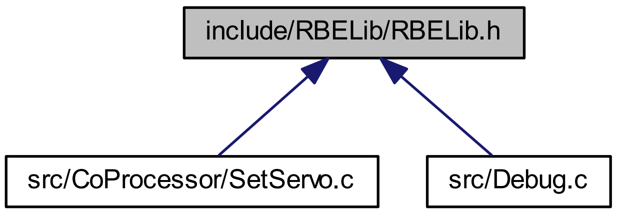
This is a meta-header. It includes all the other header files that are needed for RBELib as well as some macros.

```
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <string.h>
#include <stdarg.h>
#include <stdio.h>
#include "ADC.h"
#include "DAC.h"
#include "Debug.h"
#include "motors.h"
#include "USARTDebug.h"
#include "timer.h"
#include "Periph.h"
#include "pot.h"
#include "PID.h"
#include "reg_structs.h"
#include "ports.h"
#include "SPI.h"
#include "SetServo.h"
#include "SlaveSelects.h"
```

Include dependency graph for RBELib.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define OUTPUT 1
- #define INPUT 0
- #define ON 1
- #define OFF 0
- #define HIGH 1
- #define LOW 0
- #define TRUE 1
- #define FALSE 0
- #define BOOL unsigned char
- #define BYTE unsigned char
- #define UINT32 unsigned long
- #define INT32 long
- #define UINT16 unsigned short int
- #define WORD unsigned short int

8.11.1 Detailed Description

This is a meta-header. It includes all the other header files that are needed for RBELib as well as some macros.

Author

Kevin Harrington

Date

February 21, 2010

Author

Justin Barrett

Date

August 23, 2011

Author

Eric Willcox

Date

August 19, 2013

Definition in file RBELib.h.

8.11.2 Macro Definition Documentation

8.11.2.1 #define BOOL unsigned char

Definition at line 39 of file RBELib.h.

8.11.2.2 #define BYTE unsigned char

Definition at line 40 of file RBELib.h.

8.11.2.3 #define FALSE 0

Definition at line 36 of file RBELib.h.

8.11.2.4 #define HIGH 1

Definition at line 33 of file RBELib.h.

8.11.2.5 #define INPUT 0

Definition at line 30 of file RBELib.h.

8.11.2.6 #define INT32 long

Definition at line 42 of file RBELib.h.

8.11.2.7 #define LOW 0

Definition at line 34 of file RBELib.h.

8.11.2.8 #define OFF 0

Definition at line 32 of file RBELib.h.

8.11.2.9 #define ON 1

Definition at line 31 of file RBELib.h.

8.11.2.10 #define OUTPUT 1

Definition at line 29 of file RBELib.h.

8.11.2.11 #define TRUE 1

Definition at line 35 of file RBELib.h.

8.11.2.12 #define UINT16 unsigned short int

Definition at line 43 of file RBELib.h.

8.11.2.13 #define UINT32 unsigned long

Definition at line 41 of file RBELib.h.

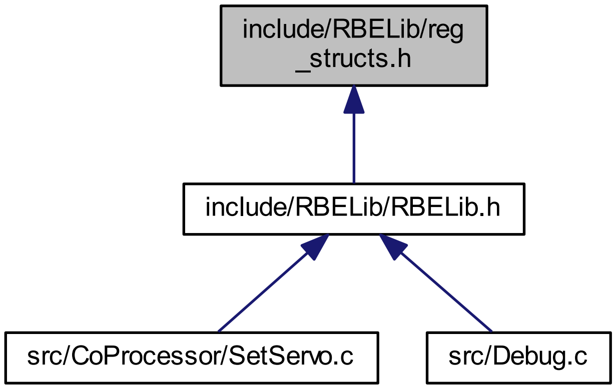
8.11.2.14 #define WORD unsigned short int

Definition at line 44 of file RBELib.h.

8.12 include/RBELib/reg_structs.h File Reference

This file redefines some of the registers of the ATmega644p as structs to allow for easy access to individual bit fields in each register. The general syntax is <register name>bits._<bitfield name>.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct __8bitreg_t

  Generic byte register.
- struct __SPIPORTbits_t

  SPI port.

Functions

- volatile __8bitreg_t PINAbits __asm__ ("0x20") __attribute__((section("sfr")))
- volatile __8bitreg_t DDRAbits __asm__ ("0x21") __attribute__((section("sfr")))
- volatile __8bitreg_t PORTAbits __asm__ ("0x22") __attribute__((section("sfr")))
- volatile __8bitreg_t PINBbits __asm__ ("0x23") __attribute__((section("sfr")))
- volatile __8bitreg_t DDRBbits __asm__ ("0x24") __attribute__((section("sfr")))

- volatile __8bitreg_t PORTBbits __asm__ ("0x25") __attribute__((section("sfr")))
- volatile __8bitreg_t PINCbits __asm__ ("0x26") __attribute__((section("sfr")))
- volatile __8bitreg_t DDRCbits __asm__ ("0x27") __attribute__((section("sfr")))
- volatile __8bitreg_t PORTCbits __asm__ ("0x28") __attribute__((section("sfr")))
- volatile __8bitreg_t PINDbits __asm__ ("0x29") __attribute__((section("sfr")))
- volatile __8bitreg_t DDRDbits __asm__ ("0x2A") __attribute__((section("sfr")))
- volatile __8bitreg_t PORTDbits __asm__ ("0x2B") __attribute__((section("sfr")))

8.12.1 Detailed Description

This file redefines some of the registers of the ATmega644p as structs to allow for easy access to individual bit fields in each register. The general syntax is <register name>bits._<bitfield name>.

Author

Peter Alley

Date

January 26, 2010

Author

Justin Barrett

Date

August 23, 2011

Author

Eric Willcox

Date

July 21, 2014

Definition in file reg_structs.h.

8.12.2 Function Documentation

8.12.2.1 volatile __8bitreg_t PINAbits __asm__ ("0x20")

8.12.2.2 volatile __8bitreg_t DDRAbits __asm__ ("0x21")

8.12.2.3 volatile __8bitreg_t PORTAbits __asm__ ("0x22")

8.12.2.4 volatile __8bitreg_t PINBbits __asm__ ("0x23")

8.12.2.5 volatile __SPIPORTbits_t SPIDDRbits __asm__ ("0x24")

8.12.2.6 volatile __SPIPORTbits_t SPIPORTbits __asm__ ("0x25")

8.12.2.7 volatile __8bitreg_t PINCbits __asm__ ("0x26")

8.12.2.8 volatile __8bitreg_t DDRCbits __asm__ ("0x27")

8.12.2.9 volatile __8bitreg_t PORTCbits __asm__ ("0x28")

8.12.2.10 volatile __8bitreg_t PINDbits __asm__ ("0x29")

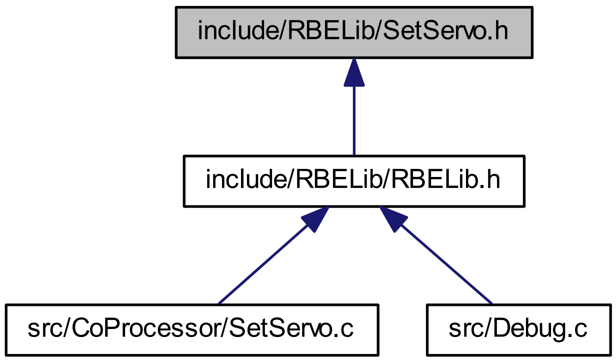
8.12.2.11 volatile __8bitreg_t DDRDbits __asm__ ("0x2A")

8.12.2.12 volatile __8bitreg_t PORTDbits __asm__ ("0x2B")

8.13 include/RBELib/SetServo.h File Reference

This file allows for using the SerSevo function to move the conveyor and gripper.

This graph shows which files directly or indirectly include this file:



Macros

- #define CLK 18432000

Functions

- void setServo (int Pin, int Value)

  Set a servo to a desired value.
- void initAltCom (unsigned long baudrate)

  Used to initialize UART0 for communication with the coprocessor. It should never be called manually.
- void setCharDebug (char byteToSend)

Used to put a char on UART0. It should never be called manually and is used by printf().

- void coPrintf (char ∗str)

  String to send to the coprocessor via UART0.

8.13.1 Detailed Description

This file allows for using the SerSevo function to move the conveyor and gripper.

Author

cwrus

Date

Jun 28, 2012

Author

Eric Willcox

Date

July 9, 2014

Definition in file SetServo.h.

8.13.2 Macro Definition Documentation

8.13.2.1 #define CLK 18432000

Definition at line 16 of file SetServo.h.

Referenced by initAltCom().

8.13.3 Function Documentation

8.13.3.1 void coPrintf (char ∗ *str*)

String to send to the coprocessor via UART0.

Parameters

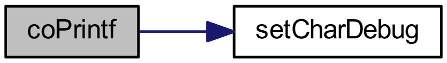
∗*str*	String to send.

Definition at line 71 of file SetServo.c.

References setCharDebug().

Referenced by setServo().

Here is the call graph for this function:



8.13.3.2 void initAltCom (unsigned long *baudrate*)

Used to initialize UART0 for communication with the coprocessor. It should never be called manually.

Parameters

baudrate	Baud rate of the communication line in bps.

Definition at line 28 of file SetServo.c.

References CLK.

Referenced by initRBELib().

8.13.3.3 void setCharDebug (char *byteToSend*)

Used to put a char on UART0. It should never be called manually and is used by printf().

Parameters

byteToSend	Character to send

Definition at line 57 of file SetServo.c.

Referenced by coPrintf().

8.13.3.4 void setServo (int *Pin,* int *Value*)

Set a servo to a desired value.

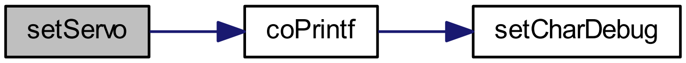
Parameters

Pin	Pin number.
Value	Value to set the pin to.

Definition at line 17 of file SetServo.c.

References coPrintf().

Here is the call graph for this function:

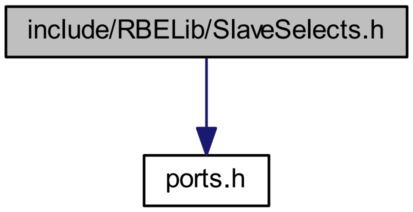


8.14 include/RBELib/SlaveSelects.h File Reference

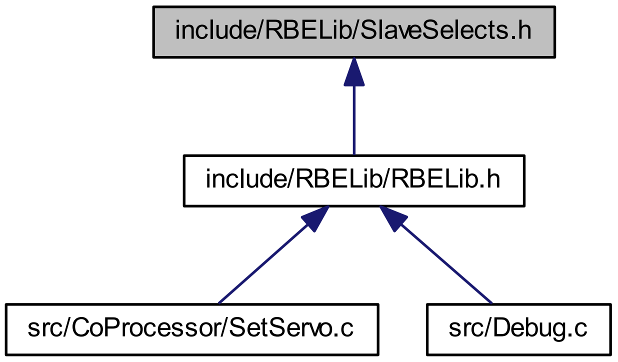
Here are all of the SPI line constants such as select lines and direction registers that can be called easily by the user instead of looking up the pins manually.

`#include "ports.h"`

Include dependency graph for SlaveSelects.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define SPI_MISO PORTBbits._P6

  SPI MISO.
- #define SPI_MISO_DDR DDRBbits._P6

  SPI MISO.
- #define SPI_MOSI PORTBbits._P5

  SPI MOSI.
- #define SPI_MOSI_DDR DDRBbits._P5

  SPI MOSI.
- #define SPI_SCK PORTBbits._P7

  SPI Clock.
- #define SPI_SCK_DDR DDRBbits._P7

  SPI Clock.
- #define SPI_MASTER_SS DDRBbits._P4

  SPI master SS.
- #define ENCODER_SS_0 PORTCbits._P5

  SPI Slave Select Encoder 0.
- #define ENCODER_SS_1 PORTCbits._P4

  SPI Slave Select Encoder 1.
- #define ENCODER_SS_2 PORTCbits._P3

  SPI Slave Select Encoder 2.
- #define ENCODER_SS_3 PORTCbits._P2

  SPI Slave Select Encoder 3.
- #define DAC_SS PORTDbits._P4

  SPI Slave Select DAC 1.
- #define AUX_DAC_SS PORTCbits._P6

  SPI Slave Select DAC 2.

- #define ENCODER_SS_0_ddr DDRCbits._P5

  SPI Slave Select Encoder 0.
- #define ENCODER_SS_1_ddr DDRCbits._P4

  SPI Slave Select Encoder 1.
- #define ENCODER_SS_2_ddr DDRCbits._P3

  SPI Slave Select Encoder 2.
- #define ENCODER_SS_3_ddr DDRCbits._P2

  SPI Slave Select Encoder 3.
- #define DAC_SS_ddr DDRDbits._P4

  SPI Slave Select DAC 1.
- #define AUX_DAC_SS_ddr DDRCbits._P6

  SPI Slave Select DAC 2.
- #define SPARE_SS_ddr DDRCbits._P0

  SPI Slave Select Unused.
- #define ENCODER_IRQ PORTBbitd._P2

  Interrupt line for all encoders.
- #define ENCODER_IRQ_ddr DDRBbitd._P2

  Interrupt line for all encoders.

8.14.1 Detailed Description

Here are all of the SPI line constants such as select lines and direction registers that can be called easily by the user instead of looking up the pins manually.

Author

cwrus

Date

Jun 28, 2012

Author

Eric Willcox

Date

July 9, 2014

Definition in file SlaveSelects.h.

8.14.2 Macro Definition Documentation

8.14.2.1 #define AUX_DAC_SS PORTCbits._P6

SPI Slave Select DAC 2.

Definition at line 75 of file SlaveSelects.h.

8.14.2.2 #define AUX_DAC_SS_ddr DDRCbits._P6

SPI Slave Select DAC 2.

Definition at line 101 of file SlaveSelects.h.

8.14.2.3 #define DAC_SS PORTDbits._P4

SPI Slave Select DAC 1.

Definition at line 71 of file SlaveSelects.h.

8.14.2.4 #define DAC_SS_ddr DDRDbits._P4

SPI Slave Select DAC 1.

Definition at line 97 of file SlaveSelects.h.

8.14.2.5 #define ENCODER_IRQ PORTBbitd._P2

Interrupt line for all encoders.

Definition at line 110 of file SlaveSelects.h.

8.14.2.6 #define ENCODER_IRQ_ddr DDRBbitd._P2

Interrupt line for all encoders.

Definition at line 114 of file SlaveSelects.h.

8.14.2.7 #define ENCODER_SS_0 PORTCbits._P5

SPI Slave Select Encoder 0.

Definition at line 54 of file SlaveSelects.h.

8.14.2.8 #define ENCODER_SS_0_ddr DDRCbits._P5

SPI Slave Select Encoder 0.

Definition at line 80 of file SlaveSelects.h.

8.14.2.9 #define ENCODER_SS_1 PORTCbits._P4

SPI Slave Select Encoder 1.

Definition at line 58 of file SlaveSelects.h.

8.14.2.10 #define ENCODER_SS_1_ddr DDRCbits._P4

SPI Slave Select Encoder 1.

Definition at line 84 of file SlaveSelects.h.

8.14.2.11 #define ENCODER_SS_2 PORTCbits._P3

SPI Slave Select Encoder 2.

Definition at line 62 of file SlaveSelects.h.

8.14.2.12 #define ENCODER_SS_2_ddr DDRCbits._P3

SPI Slave Select Encoder 2.

Definition at line 88 of file SlaveSelects.h.

8.14.2.13 #define ENCODER_SS_3 PORTCbits._P2

SPI Slave Select Encoder 3.

Definition at line 66 of file SlaveSelects.h.

8.14.2.14 #define ENCODER_SS_3_ddr DDRCbits._P2

SPI Slave Select Encoder 3.

Definition at line 92 of file SlaveSelects.h.

8.14.2.15 #define SPARE_SS_ddr DDRCbits._P0

SPI Slave Select Unused.

Definition at line 105 of file SlaveSelects.h.

8.14.2.16 #define SPI_MASTER_SS DDRBbits._P4

SPI master SS.

Definition at line 50 of file SlaveSelects.h.

8.14.2.17 #define SPI_MISO PORTBbits._P6

SPI MISO.

Definition at line 24 of file SlaveSelects.h.

8.14.2.18 #define SPI_MISO_DDR DDRBbits._P6

SPI MISO.

Definition at line 28 of file SlaveSelects.h.

8.14.2.19 #define SPI_MOSI PORTBbits._P5

SPI MOSI.

Definition at line 33 of file SlaveSelects.h.

8.14.2.20 #define SPI_MOSI_DDR DDRBbits._P5

SPI MOSI.

Definition at line 37 of file SlaveSelects.h.

8.14.2.21 #define SPI_SCK PORTBbits._P7

SPI Clock.

Definition at line 42 of file SlaveSelects.h.

8.14.2.22 #define SPI_SCK_DDR DDRBbits._P7

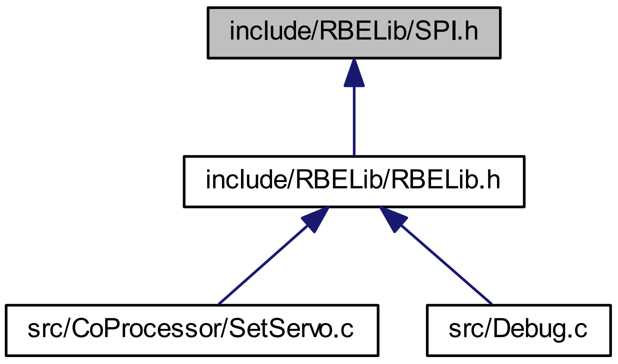
SPI Clock.

Definition at line 46 of file SlaveSelects.h.

8.15 include/RBELib/SPI.h File Reference

The header file and function prototypes for the SPI.

This graph shows which files directly or indirectly include this file:



Functions

- void initSPI ()

  Initializes the SPI bus for communication with all of your SPI devices.
- unsigned char spiTransceive (BYTE data)

  Send and receive a byte out of the MOSI line.

8.15.1 Detailed Description

The header file and function prototypes for the SPI.

Bug While not a bug, some students have problems with some of the SPI devices where their code will not work after they send a command. To fix this, you may have to toggle the SS line once after sending your command and then disable it once more. This is because some of the devices need the toggle to load the registers and then execute the command.

Bug While not a bug, the DAC SS may need to be toggled at startup. This is only something that matters during a soft reset but should be done anyway during your initialization for SPI.

Author

kamiro87

Date

August 31, 2010

Author

Justin Barrett

Date

August 23, 2011

Author

Eric Willcox

Date

August 20, 2013

Definition in file SPI.h.

8.15.2 Function Documentation

8.15.2.1 void initSPI ( )

Initializes the SPI bus for communication with all of your SPI devices.

Todo Create the function that will allow you to initialize the SPI in a mode compatible with all devices. Do not forget to deassert all of your SS lines!

8.15.2.2 unsigned char spiTransceive (BYTE *data*)

Send and receive a byte out of the MOSI line.

Please note that even if you do not want to receive any data back from a SPI device, the SPI standard requires you still receive something back even if it is junk data.

Parameters

data	The byte to send down the SPI bus.

Returns

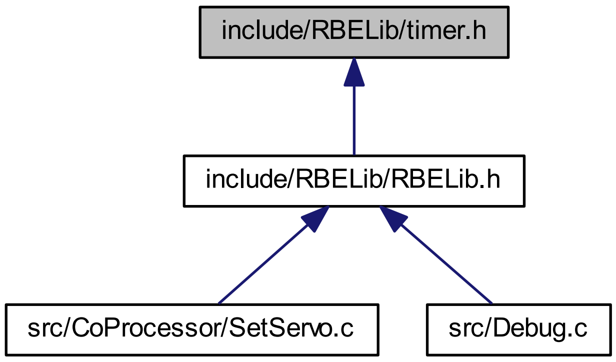
value The byte shifted in during transmit

Todo Make a function that will send a byte of data through the SPI and return whatever was sent back.

8.16 include/RBELib/timer.h File Reference

The header file and function prototypes for Timers 0-2.

This graph shows which files directly or indirectly include this file:



Macros

- #define NORMAL 0

  Timer normal mode.
- #define CTC 1

  Timer Clear Timer on Compare (CTC) mode.

Functions

- void initTimer (int timer, int mode, unsigned int comp)

  Initializes the specified timer in the specified mode. This header file provides constants for NORMAL operation mode and CTC operation mode, however there are many more modes of operation for the Timers that can be experimented with.
- void setCompValue (unsigned char timer, unsigned short int comp)

  Only used when the specified timer is in CTC mode. Changes the value of the comparison register to the new specified value.

8.16.1 Detailed Description

The header file and function prototypes for Timers 0-2.

Author

Justin Barrett

Date

August 25, 2011

Author

Eric Willcox

Date

August 20, 2013

Definition in file timer.h.

8.16.2 Macro Definition Documentation

8.16.2.1 #define CTC 1

Timer Clear Timer on Compare (CTC) mode.

Definition at line 22 of file timer.h.

8.16.2.2 #define NORMAL 0

Timer normal mode.

Definition at line 18 of file timer.h.

8.16.3 Function Documentation

8.16.3.1 void initTimer (int *timer,* int *mode,* unsigned int *comp*)

Initializes the specified timer in the specified mode. This header file provides constants for NORMAL operation mode and CTC operation mode, however there are many more modes of operation for the Timers that can be experimented with.

Parameters

timer	The number of the timer to be initialized (0-2).
mode	The mode to initialize the specified timer in.

comp	Only used in CTC mode. The value that the timer counts to before it triggers an interrupt and resets.

Todo Create a function that initializes the desired timer in a given mode and set the compare value (as appropriate).

8.16.3.2 void setCompValue (unsigned char *timer*, unsigned short int *comp*)

Only used when the specified timer is in CTC mode. Changes the value of the comparison register to the new specified value.

Parameters

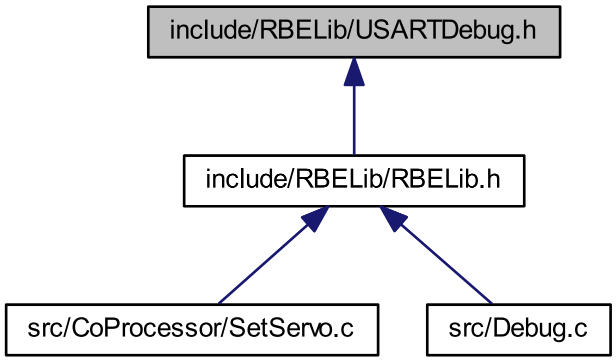
timer	The timer comparison value to change (0-2).
comp	The value to set the comparison register to.

Todo Create a function that will set a new compare value for the given timer.

8.17 include/RBELib/USARTDebug.h File Reference

The header file and function prototypes for USART1.

This graph shows which files directly or indirectly include this file:



Macros

- #define DEFAULT_BAUD = 115200;

Functions

- void debugUSARTInit (unsigned long baudrate)

Initializes USART1 as a print terminal to the PC. This function must check the incoming baudrate against the valid baudrates from the data-sheet. If the baudrate is invalid, then the DEFAULT_BAUD constant must be used instead.

- void putCharDebug (char byteToSend)

  Sends one byte to the USART1 Tx pin (Transmits one byte).

- unsigned char getCharDebug (void)

  Recieves one byte of data from the serial port (i.e. from the PC).

8.17.1 Detailed Description

The header file and function prototypes for USART1.

Author

Kevin Harrington

Date

August 20, 2010

Author

Justin Barrett

Date

August 25, 2011

Author

Eric Willcox

Date

August 20, 2013

Definition in file USARTDebug.h.

8.17.2 Macro Definition Documentation

8.17.2.1 #define DEFAULT_BAUD = 115200;

Definition at line 19 of file USARTDebug.h.

8.17.3 Function Documentation

8.17.3.1 void debugUSARTInit (unsigned long *baudrate*)

Initializes USART1 as a print terminal to the PC. This function must check the incoming baudrate against the valid baudrates from the data-sheet. If the baudrate is invalid, then the DEFAULT_BAUD constant must be used instead.

Parameters

baudrate	The desired baudrate to set for USART1.

Todo Create the function that will initialize USART1 for debugging use.

8.17.3.2 unsigned char getCharDebug (void)

Recieves one byte of data from the serial port (i.e. from the PC).

Returns

byteReceived Character that was received on the USART.

Todo Make the function that will listen for input on the USART1 RX line.

8.17.3.3 void putCharDebug (char *byteToSend*)

Sends one byte to the USART1 Tx pin (Transmits one byte).

Parameters

byteToSend	The byte that is to be transmitted through USART1.

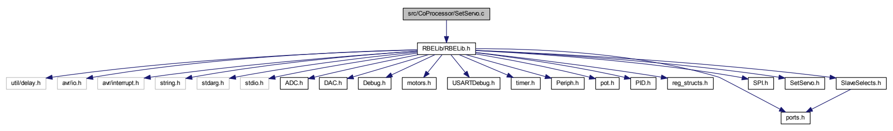
Todo Make the function that will put a character on the USART1 TX line.

Referenced by printfRBE().

8.18 src/CoProcessor/SetServo.c File Reference

Sending setServo() command to the coprocessor.

```
#include "RBELib/RBELib.h"
```
Include dependency graph for SetServo.c:



Functions

- void setServo (int pin, int value)

  Set a servo to a desired value.
- void initAltCom (unsigned long baudrate)

  Used to initialize UART0 for communication with the coprocessor. It should never be called manually.
- void setCharDebug (char byteToSend)

Used to put a char on UART0. It should never be called manually and is used by printf().

- void coPrintf (char ∗str)

  String to send to the coprocessor via UART0.

8.18.1 Detailed Description

Sending setServo() command to the coprocessor.

Author

Eric Willcox

Date

July 9, 2014

Definition in file SetServo.c.

8.18.2 Function Documentation

8.18.2.1 void coPrintf (char ∗ *str*)

String to send to the coprocessor via UART0.

Parameters

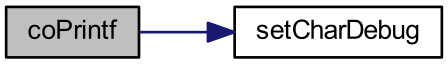
∗*str*	String to send.

Definition at line 71 of file SetServo.c.

References setCharDebug().

Referenced by setServo().

Here is the call graph for this function:



8.18.2.2 void initAltCom (unsigned long *baudrate*)

Used to initialize UART0 for communication with the coprocessor. It should never be called manually.

Parameters

baudrate	Baud rate of the communication line in bps.

Definition at line 28 of file SetServo.c.

References CLK.

Referenced by initRBELib().

8.18.2.3 void setCharDebug (char *byteToSend*)

Used to put a char on UART0. It should never be called manually and is used by printf().

Parameters

byteToSend	Character to send

Definition at line 57 of file SetServo.c.

Referenced by coPrintf().

8.18.2.4 void setServo (int *Pin,* int *Value*)

Set a servo to a desired value.

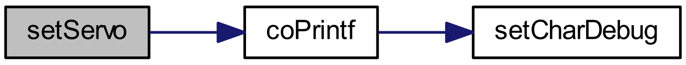
Parameters

Pin	Pin number.
Value	Value to set the pin to.

Definition at line 17 of file SetServo.c.

References coPrintf().

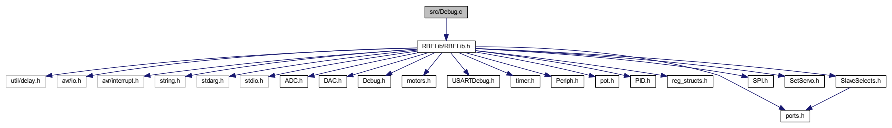
Here is the call graph for this function:



8.19 src/Debug.c File Reference

Allows for printf() and setServo() capability.

```
#include "RBELib/RBELib.h"
```
Include dependency graph for Debug.c:



Functions

- int printfRBE (char var, FILE ∗stream)

    Calls the students function 'putCharDebug()' to output the stream to. You should not call this function directly, instead use the standard function printf().

- void initRBELib ()

    Rebinds stdout to call printfRBE() and initializes communication with the coprocessor. This function should be called once at the start of your code once you have putCharDebug() written in Lab 1. If you do not call this, printf() and SetServo() will not work.

8.19.1 Detailed Description

Allows for printf() and setServo() capability.

Author

Eric Willcox

Date

July 9, 2014

Definition in file Debug.c.

8.19.2 Function Documentation

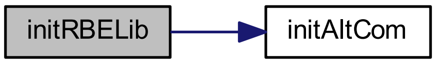
8.19.2.1 void initRBELib ( )

Rebinds stdout to call printfRBE() and initializes communication with the coprocessor. This function should be called once at the start of your code once you have putCharDebug() written in Lab 1. If you do not call this, printf() and SetServo() will not work.

Definition at line 20 of file Debug.c.

References initAltCom().

Here is the call graph for this function:



8.19.2.2 int printfRBE (char *var,* FILE ∗ *stream*)

Calls the students function 'putCharDebug()' to output the stream to. You should not call this function directly, instead use the standard function printf().

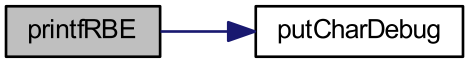
Parameters

var	Character to output
∗*stream*	Place to put the character (stdout)

Definition at line 14 of file Debug.c.

References putCharDebug().

Here is the call graph for this function:



Index