My Project

Generated by Doxygen 1.8.11

Contents

Chapter 1

Todo List

Member b7

make a nice array or something

make for rest of buttons

Member clearADC (int channel)

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

Member degreesPerJoint2Val

change these to constants

Member gotoXY (int x, int y)

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

Member inPosition (int theta1, int theta2)

pass in the threshold to use

Member IRDist (int chan)

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

Member setFrequencyForPostScale (int Frequency)

make more generic

Member upperAngle

get rid of these globals

Member waitForButton7 ()

make more generic

Member y_coord

get rid of these globals

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

File Documentation

3.1 ADC.c File Reference

ADC setup function definitions.

```
#include "RBELib/RBELib.h"
#include "include/definitions.h"
```

Functions

- ISR (ADC_vect)

  ISR for updating ADC readings Runs when ADC reading completes based on configured ADC settings.
- void initADC (int channel)

  Initializes the ADC and makes one channel active.
- void clearADC (int channel)

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

  Get the analog value from the configured channel.
- void changeADC (int channel)

  Change the channel the ADC is sampling if using interrupts.

Variables

- volatile unsigned short adch
- volatile unsigned short adcl
- volatile unsigned short readNewChannel

3.1.1 Detailed Description

ADC setup function definitions.

This has everything needed to configure the ADC for 10 bit operation and reading the values.

Author

cpbove@wpi.edu

Date

28-Jan-2016

Version

1.0

3.1.2 Function Documentation

3.1.2.1 void changeADC (int *channel*)

Change the channel the ADC is sampling if using interrupts.

Parameters

channel	The ADC channel to switch to.

3.1.2.2 void clearADC (int *channel*)

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

This still needs to be tested.

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.

3.1.2.3 unsigned short getADC (int *channel*)

Get the analog value from the configured channel.

Parameters

channel	The ADC channel to run a conversion on.

Returns

adcVal The 10 bit value returned by the ADC conversion.

3.1.2.4 void initADC (int *channel*)

Initializes the ADC and makes one channel active.

Enables ADC to use interrupts

Parameters

channel	The ADC channel to initialize

3.1.2.5 ISR (ADC_vect)

ISR for updating ADC readings Runs when ADC reading completes based on configured ADC settings.

Parameters

ADC_vect	Interrupt vector for ADC on chip

3.1.3 Variable Documentation

3.1.3.1 adch

for storing the adch register value after an interrupt

3.1.3.2 adcl

for storing the adch register value after an interrupt

3.1.3.3 readNewChannel

used to make sure we read the new channel after we switch channels

3.2 arm.c File Reference

arm library

```
#include "include/arm.h"
#include "include/definitions.h"
#include "include/button.h"
#include "math.h"
```

Functions

- ISR (TIMER0_COMPA_vect)
  Timer ISR that runs at 100Hz.
- void initArm ()
  initialize the arm variables
- void setupTimer ()
  sets a 100Hz timer up on Timer 0
- void serviceArm ()
  runs functions critical to arm operation. Call as often as possible.
- void printHeaderForJointAngle ()
  prints the header for streaming joint angles
- void printHeaderForLogging ()
  prints the header for the main logging of step responses
- float getJointAngle (int joint)
  gets the current, calibrated joint angle of the passed joint number
- void printJointAngle (int joint)
  prints adcval,pot mV, and joint angle of the passed joint number
- void printLogLineJoint2 (int setPoint)
  prints time, setpoint, jointAngle, PID output, and motor current
- void printJointAnglesAndPos ()
  streams joint angles in degrees and (x,y) tip positions in mm
- int getCurrent (int joint)
  gets motor current of specified joint
- int convertVoltsToDACVal (float volts)
  converts a voltage value to a DAC value
- void setJointAngles (int lowerJoint, int upperJoint)
  updates globals with new desired ones
- void controlArmWithButtons ()
  changes desired joint angle of arm based on button presses for testing
- void controlPositionWithButtons ()
  changes desired joint angles of arm based on button presses
- void controlJoint1WithButtons ()
  Lab3 - control motor drive voltages with buttons.
- void drawTriangleWithButtons ()
  state machine for drawing triangles.
- float getTimeSeconds ()
  gets the time in seconds
- void calcXY ()
  updates 2 globals with the current xy in mm of the end effector
- BOOL inPosition (int theta1, int theta2)
  check if the arm is in the angle position specified
- float getGs (int axis)
  runs getAccel and converts to G's

Variables

- float degreesPerJoint1Val
- float degreesPerJoint2Val
- float x_coord
- float y_coord
- int lowerAngle
- int upperAngle
- volatile BOOL servicePID
- volatile unsigned long timerCount

3.2.1 Detailed Description

arm library

Author

cpbove@wpi.edu

Date

30-Jan-2016

Version

1.0

3.2.2 Function Documentation

3.2.2.1 void controlArmWithButtons ( )

changes desired joint angle of arm based on button presses for testing

Note

This does not run the PID control, just sets setpoint

3.2.2.2 void controlJoint1WithButtons ( )

Lab3 - control motor drive voltages with buttons.

Sets first link to 0, -3, +3, or plus 6 Volts button 4 sets voltage to 0 button 5 sets voltage to -3 button 6 sets voltage to +3 button 7 sets voltage to +6

3.2.2.3 void controlPositionWithButtons ( )

changes desired joint angles of arm based on button presses

Note

This does not run the PID control, just sets setpoint

3.2.2.4 int convertVoltsToDACVal (float *volts*)

converts a voltage value to a DAC value

Parameters

volts	to convert to DAC -7.2 to 7.2 volts

Returns

DAC value 0-4095

3.2.2.5 void drawTriangleWithButtons ( )

state machine for drawing triangles.

button 4 draws a prescripted triangle button 5 just moves to a pose button 6 records a motion button 7 plays back the motion

Note

This does not run the PID control, just sets setpoint

3.2.2.6 int getCurrent (int *joint*)

gets motor current of specified joint

Parameters

joint	The joint to get the current for

Returns

current in mA

3.2.2.7 float getGs (int *axis*)

runs getAccel and converts to G's

Parameters

axis	0-2 for x,y,z axis to get g's on

Returns

-3.0 to 3.0 g's of force

3.2.2.8 float getJointAngle (int *joint*)

gets the current, calibrated joint angle of the passed joint number

Parameters

joint	1 or 2 of the joint to get the angle for

Returns

angle of joint in degrees (generally 0 to 180)

3.2.2.9 float getTimeSeconds ( )

gets the time in seconds

Returns

time in seconds

3.2.2.10 BOOL inPosition (int *theta1*, int *theta2*)

check if the arm is in the angle position specified

Parameters

theta1	angle of first joint
theta2	angle of second joint

Returns

TRUE if arm is between

Todo pass in the threshold to use

3.2.2.11 ISR (TIMER0_COMPA_vect)

Timer ISR that runs at 100Hz.

set flags for servicing at fixed intervals.

Parameters

TIMER0_COMPA_vect	Interrupt vector for timer0 vector on AVR

3.2.2.12 void printJointAngle (int *joint*)

prints adcval,pot mV, and joint angle of the passed joint number

Parameters

joint	1 or 2 of the joint to print data for

3.2.2.13 void printLogLineJoint2 (int *setPoint*)

prints time, setpoint, jointAngle, PID output, and motor current

Parameters

setPoint	the current command to the controller

3.2.2.14 void setJointAngles (int *lowerJoint,* int *upperJoint*)

updates globals with new desired ones

Parameters

lowerJoint	position for the lower joint 1
upperJoint	position for the upper joint 2

3.2.3 Variable Documentation

3.2.3.1 degreesPerJoint1Val

for storing the degrees per adc value for joint 1

3.2.3.2 degreesPerJoint2Val

for storing the degrees per adc value for joint 2

Todo change these to constants

3.2.3.3 lowerAngle

for storing the current lowerAngle of the arm

3.2.3.4 servicePID

flag - TRUE if PID controller needs to be serviced, FALSE otherwise

3.2.3.5 timerCount

for keeping time. increments in 0.01 seconds

3.2.3.6 upperAngle

for storing the current upperAngle of the arm

Todo get rid of these globals

3.2.3.7 x_coord

for storing the current x coordinate of the arm

3.2.3.8 y_coord

for storing the current y coordinate of the arm

Todo get rid of these globals

3.3 button.c File Reference

button setup function

```
#include "RBELib/RBELib.h"
#include "include/button.h"
```

Functions

- void setupButtons (__8bitreg_t *DDxn, __8bitreg_t *PORTxn, __8bitreg_t *PINxbits, BOOL enablePullup)
  sets registers to configure the buttons as inputs
- void serviceButtons ()
  runs functions critical to button operation. Call as often as possible.
- unsigned char lastButtonPressed ()
  determines what button was pressed last

Variables

- __8bitreg_t * PINybits
- unsigned char b4
- unsigned char b5
- unsigned char b6
- unsigned char b7

3.3.1 Detailed Description

button setup function

To read buttons, check PINxbits._Py

Author

cpbove@wpi.edu

Date

28-Jan-2016

Version

1.0

3.3.2 Function Documentation

3.3.2.1 unsigned char lastButtonPressed ( )

determines what button was pressed last

Returns

number of last button pressed

3.3.2.2 void setupButtons (__8bitreg_t * *DDxn*, __8bitreg_t * *PORTxn*, __8bitreg_t * *PINxbits*, BOOL *enablePullup*)

sets registers to configure the buttons as inputs

Parameters

**DDxn*	Pointer to the digital port (DDxn) that buttons are connected to
**PORTxn*	Pointer to the PORTxn register for the buttons
**PINxbits*	Pointer to the PINx register for the buttons
enablePullUp	Set true to enable pull up resistors on the buttons

3.3.3 Variable Documentation

3.3.3.1 b4

storing last state of button 4

3.3.3.2 b5

storing last state of button 5

3.3.3.3 b6

storing last state of button 6

3.3.3.4 b7

storing last state of button 7

Todo make a nice array or something
make for rest of buttons

3.3.3.5 * PINybits

pointer to the PINbits the buttons are connected to

3.4 DAC.c File Reference

DAC library.

```
#include "RBELib/RBELib.h"
```

Functions

- void setDAC (int DACn, int SPIVal)
  Set the DAC to the given value on the chosen channel.

3.4.1 Detailed Description

DAC library.

Author

cpbove@wpi.edu

Date

30-Jan-2016

Version

1.0

3.4.2 Function Documentation

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

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

Parameters

DACn	The channel (0-3) that you want to set.
SPIVal	The value you want to set it to.

3.5 definitions.c File Reference

defines for our team

```
#include "RBELib/RBELib.h"
#include "include/definitions.h"
```

Functions

- int abs (int a)

  take absolute value of passed integer
- BOOL betweenTwoVals (int value, int lower, int upper)

  check if passed value is between lower and upper bounds

3.5.1 Detailed Description

defines for our team

some basic useful functions that our team uses.

Author

cpbove@wpi.edu

Date

3-Feb-2016

Version

1.0

3.5.2 Function Documentation

3.5.2.1 int abs (int *a*)

take absolute value of passed integer

Parameters

a	The number to take the absolute value of

3.5.2.2 BOOL betweenTwoVals (int *value*, int *lower*, int *upper*)

check if passed value is between lower and upper bounds

Parameters

value	the number to check with
lower	the lower bound to check against
upper	the upper bound to check against

Returns

true if value is between upper and lower numbers

3.6 encoder.c File Reference

encoder library

```
#include "include/encoder.h"
#include "include/definitions.h"
```

Functions

- void singleByteWrite (unsigned char op_code, unsigned char data, int joint)
  write a single byte to encoder
- void slaveSelect (int joint)
  sets the chip select low for the passed joint
- void slaveDeselect (int joint)
  sets the chip select high for the passed joint
- void waitForTransmissionEnd ()
  holds processor until transmission has finished

3.6.1 Detailed Description

encoder library

Author

LSI Computer Systems, modified by cpbove@wpi.edu

Date

16-Feb-2016

Version

1.0

3.6.2 Function Documentation

3.6.2.1 void singleByteWrite (unsigned char *op_code,* unsigned char *data,* int *joint*)

write a single byte to encoder

Parameters

op_code	one of the defined operation codes
data	data to write to the encoder
joint	1 or 2 for the encoder joint to read

3.6.2.2 void slaveDeselect (int *joint*)

sets the chip select high for the passed joint

Parameters

joint	1 or 2 to deselect

3.6.2.3 void slaveSelect (int *joint*)

sets the chip select low for the passed joint

Parameters

joint	1 or 2 to select

3.7 lab1.c File Reference

Lab1 functions.

```
#include "RBELib/RBELib.h"
#include "include/definitions.h"
#include "include/lab1.h"
```

Functions

- void initPart2and7Timer ()

  ISR to increment global timer0Count, used for timestamps on Part 2 and 7 Also toggles an LED each run.
- void initSignalGeneratorTimer ()

  Configures Timer1 for Part 4 of lab(signal generation), prints header for logging.
- void outputSetup ()

Configures outputs for generating signals on pins.

- void waitForButton7 ()

  Holds processor until button 7 is pressed.

- void waitForChar (char c)

  Holds the processor until the right character is received from USART.

- void ADCToSerial (int channel)

  Streams timestamp, adcval, potmv, and potangle to PC w/ USART Transmits timestamps, ADC value in counts, pot output in millivolts, and pot angle in degrees over the serial port to the screen in the Terminal.

- void ADCToSerialPart7 (int channel)

  Streams timestamp and ADC values to the serial port for 1 second Transmits timestamps, ADC value in counts in degrees over the serial port to the screen in the Terminal.

- void setFrequencyForPostScale (int Frequency)

  Configures postscaler in ISR based on inputted frequency.

- void signalGeneratorMain (int channel)

  Generates signals w/particular duty cycles set by pot.

- void **triangleSignalGengerator** ()

Variables

- volatile unsigned long **timer0Count**
- volatile unsigned long **timer2Count**
- volatile unsigned **timer2CountSub**
- unsigned **postScale**

3.7.1 Detailed Description

Lab1 functions.

This includes all the functions needed for Lab1.

Authors

cpbove@wpi.edu jlhonicker@wpi.edu dmmurray@wpi.edu

Date

28-Jan-2016

Version

1.0

3.7.2 Function Documentation

3.7.2.1 void ADCToSerial (int *channel*)

Streams timestamp, adcval, potmv, and potangle to PC w/ USART Transmits timestamps, ADC value in counts, pot output in millivolts, and pot angle in degrees over the serial port to the screen in the Terminal.

Parameters

channel	The ADC channel to read from

3.7.2.2 void ADCToSerialPart7 (int *channel*)

Streams timestamp and ADC values to the serial port for 1 second Transmits timestamps, ADC value in counts in degrees over the serial port to the screen in the Terminal.

Parameters

channel	The ADC channel to read

3.7.2.3 void initPart2and7Timer ( )

ISR to increment global timer0Count, used for timestamps on Part 2 and 7 Also toggles an LED each run.

Parameters

TIMER0_COMPA_vect	Vector for Timer0 ISR ISR to increment global timer2Count, used for signal generation

Uses a sub counter to get a slower rate for some of the signal generations.

Parameters

TIMER2_COMPA_vect	Vector for Timer2 ISR Configures Timer0 for Parts 2 and 7 of lab, prints header for logging

3.7.2.4 void setFrequencyForPostScale (int *Frequency*)

Configures postscaler in ISR based on inputted frequency.

Only handles a few frequencies: 1,20,100

Todo make more generic

Parameters

frequency	The frequency to change to (1, 20, or 100)

3.7.2.5 void signalGeneratorMain (int *channel*)

Generates signals w/particular duty cycles set by pot.

Also reads buttons 5-7 to determine which frequency to output

Parameters

channel	The ADC channel to read the pot from

3.7.2.6 void waitForButton7 ( )

Holds processor until button 7 is pressed.

Todo make more generic

3.7.2.7 void waitForChar (char *c*)

Holds the processor until the right character is received from USART.

Parameters

c	The character to wait for.

3.8 led.c File Reference

LED setup functions.

```
#include "include/led.h"
```

Functions

- void setupLEDs (__8bitreg_t *DDxn, __8bitreg_t *PORTxn)
  configures registers as outputs for the LEDs

3.8.1 Detailed Description

LED setup functions.

Author

cpbove@wpi.edu

Date

28-Jan-2016

Version

1.0

3.8.2 Function Documentation

3.8.2.1 void setupLEDs (__8bitreg_t * *DDxn*, __8bitreg_t * *PORTxn*)

configures registers as outputs for the LEDs

Sets all the ports as outputs and then puts them all low to prevent them from turning on unintentionally.

Parameters

**DDxn*	Pointer to the digital outputs (DDxn) that buttons will be connected to
**PORTxn*	Pointer to the PORTxn that actually writes the outputs
enablePullUp	Set true to enable pull ups on the buttons

3.9 main.c File Reference

Main loop running on board. Runs Lab1 functions.

```
#include "RBELib/RBELib.h"
#include "include/definitions.h"
#include "include/button.h"
#include "include/lab1.h"
#include "include/led.h"
#include "include/arm.h"
```

Functions

- int main (void)
  main loop for AVR chip

3.9.1 Detailed Description

Main loop running on board. Runs Lab1 functions.

This code runs a collection of Lab 1 functions to complete the lab. Some functions need to be uncommented or commented depending on which section they complete. This should be more streamlined in the future.

Author

cpbove@wpi.edu

Date

28-Jan-2016

Version

1.0

3.10 motors.c File Reference

Motor driving functions for the arm.

```
#include "RBELib/RBELib.h"
#include "include/definitions.h"
#include "include/arm.h"
```

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.

3.10.1 Detailed Description

Motor driving functions for the arm.

Author

Chris Bove

Date

Feb 3, 2016

3.10.2 Function Documentation

3.10.2.1 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. Between -2048 and +2048

3.10.2.2 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.

3.10.2.3 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.

3.11 Periph.c File Reference

peripheral library for encoder and accel

```
#include "RBELib/RBELib.h"
#include "include/encoder.h"
```

Macros

- #define ENC_CNTR 0x20
- #define ENC_CLR_CMD 0x00
- #define ENC_RD_CMD 0x40

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.

3.11.1 Detailed Description

peripheral library for encoder and accel

Author

cpbove@wpi.edu with help from Joe St. Germain

Date

15-02-2016

Version

1.0

3.11.2 Macro Definition Documentation

3.11.2.1 #define ENC_CLR_CMD 0x00

hex value for clearing the encoder count

3.11.2.2 #define ENC_CNTR 0x20

hex value for addressing the CNTR register on the encoder

3.11.2.3 #define ENC_RD_CMD 0x40

hex value command for reading the encoder count

3.11.3 Function Documentation

3.11.3.1 signed long encCount (int *chan*)

Finds the current count of one of the encoders.

Parameters

chan	Channel of the encoder (change: Joint 1 or 2)

Returns

count The current count of the encoder.

3.11.3.2 void encInit (int *chan*)

Initialize the encoders with the desired settings.

Parameters

chan	Channel to initialize (change: Joint 1 or 2)

Note

leaves the encoder in Byte 4, Quad X1 default state

3.11.3.3 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.

3.11.3.4 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.

3.11.3.5 void resetEncCount (int *chan*)

Reset the current count of the encoder ticks.

Parameters

chan	The channel to clear. (change: Joint 1 or 2)

3.12 PID.c File Reference

The source file for PID constants and calculations.

```
#include "RBELib/RBELib.h"
#include "include/definitions.h"
#include "math.h"
```

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.

3.12.1 Detailed Description

The source file for PID constants and calculations.

Sets the PID constants and calculate the PID value.

Author

Chris Bove

Date

2-3-16

3.12.2 Function Documentation

3.12.2.1 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.

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

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

Parameters

link	The link you want to set the values for (2 or 3).
Kp	Proportional value.
Ki	Integral value.
Kd	Derivative value.

3.13 pot.c File Reference

The potentiometer functions.

```
#include "RBELib/RBELib.h"
```

Functions

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

3.13.1 Detailed Description

The potentiometer functions.

Use these functions to read the values from the pots.

Author

Chris Bove

Date

2-3-16

3.13.2 Function Documentation

3.13.2.1 int potAngle (int *pot*)

Find the angle of the given potentiometer.

Parameters

pot	The pot to check.

Returns

angle Angle of the potentiometer.

3.13.2.2 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.

3.14 SPI.c File Reference

arm library

```
#include "RBELib/RBELib.h"
#include "include/definitions.h"
```

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.

3.14.1 Detailed Description

arm library

Author

cpbove@wpi.edu

Date

30-Jan-2016

Version

1.0

3.14.2 Function Documentation

3.14.2.1 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

Index