My Project

Generated by Doxygen 1.8.8

Tue Mar 8 2016 10:25:50

Contents

1 Namespace Index **1**

1.1 Namespace List . . . 1

2 Class Index **3**

2.1 Class List . . . 3

3 File Index **5**

3.1 File List . . . 5

4 Namespace Documentation **7**

4.1 ChipChipArray Namespace Reference . . . 7

4.1.1 Detailed Description . . . 7

4.1.2 Function Documentation . . . 7

4.1.2.1 main . . . 8

4.1.2.2 ScanQR . . . 8

4.1.3 Variable Documentation . . . 9

4.1.3.1 qrInvokeCount . . . 9

4.1.3.2 scanQrLog . . . 9

4.2 std Namespace Reference . . . 9

4.2.1 Function Documentation . . . 9

4.2.1.1 to_string . . . 9

4.2.1.2 to_string . . . 10

4.2.1.3 to_string . . . 10

4.2.1.4 to_string . . . 10

4.2.1.5 to_string . . . 10

4.2.1.6 to_string . . . 10

4.2.1.7 to_string . . . 10

5 Class Documentation **11**

5.1 Adafruit_PWMServoDriver Class Reference . . . 11

5.1.1 Detailed Description . . . 11

5.1.2 Constructor & Destructor Documentation . . . 11

5.1.2.1 Adafruit_PWMServoDriver . . . 11

5.1.3 Member Function Documentation . . . 11
5.1.3.1 begin . . . 11
5.1.3.2 reset . . . 12
5.1.3.3 setPin . . . 12
5.1.3.4 setPWM . . . 13
5.1.3.5 setPWMFreq . . . 13
5.2 ChipChipArray::Arm Class Reference . . . 14
5.2.1 Detailed Description . . . 14
5.2.2 Constructor & Destructor Documentation . . . 14
5.2.2.1 Arm . . . 14
5.2.3 Member Function Documentation . . . 15
5.2.3.1 BaseTilt . . . 15
5.2.3.2 BaseTurn . . . 15
5.2.3.3 dBaseTilt . . . 15
5.2.3.4 dBaseTurn . . . 16
5.2.3.5 dElbow . . . 16
5.2.3.6 dGrippers . . . 17
5.2.3.7 dLeftGripper . . . 18
5.2.3.8 dRightGripper . . . 18
5.2.3.9 dWristTilt . . . 18
5.2.3.10 dWristTwist . . . 19
5.2.3.11 Elbow . . . 19
5.2.3.12 Grippers . . . 19
5.2.3.13 Hover . . . 19
5.2.3.14 LeftGripper . . . 20
5.2.3.15 RightGripper . . . 20
5.2.3.16 WristTilt . . . 20
5.2.3.17 WristTwist . . . 21
5.2.4 Member Data Documentation . . . 21
5.2.4.1 servoPos . . . 21
5.3 ChipChipArray::Block Class Reference . . . 21
5.3.1 Detailed Description . . . 22
5.3.2 Constructor & Destructor Documentation . . . 22
5.3.2.1 Block . . . 22
5.3.3 Member Data Documentation . . . 22
5.3.3.1 area . . . 22
5.3.3.2 bottomLeft . . . 22
5.3.3.3 bottomRight . . . 22
5.3.3.4 color . . . 22
5.3.3.5 dBottom . . . 22

5.3.3.6 dLeft . . . 23
5.3.3.7 dRight . . . 23
5.3.3.8 dRightLeft . . . 23
5.3.3.9 dTop . . . 23
5.3.3.10 dTopBottom . . . 23
5.3.3.11 height . . . 23
5.3.3.12 offset . . . 23
5.3.3.13 size . . . 23
5.3.3.14 topLeft . . . 23
5.3.3.15 topRight . . . 24
5.3.3.16 width . . . 24
5.4 ChipChipArray::Log Class Reference . . . 24
5.4.1 Detailed Description . . . 24
5.4.2 Constructor & Destructor Documentation . . . 24
5.4.2.1 Log . . . 24
5.4.2.2 Log . . . 25
5.4.2.3 ∼Log . . . 25
5.4.3 Member Function Documentation . . . 25
5.4.3.1 Debug . . . 25
5.4.3.2 Error . . . 26
5.4.3.3 Image . . . 26
5.4.3.4 Open . . . 27
5.4.3.5 Status . . . 27
5.4.3.6 Variable . . . 27
5.4.3.7 Verbose . . . 28
5.5 ChipChipArray::PiCamera Class Reference . . . 28
5.5.1 Detailed Description . . . 28
5.5.2 Constructor & Destructor Documentation . . . 29
5.5.2.1 PiCamera . . . 29
5.5.2.2 PiCamera . . . 29
5.5.3 Member Function Documentation . . . 29
5.5.3.1 Close . . . 29
5.5.3.2 Snap . . . 29

6 File Documentation **31**

6.1 etc/doxygen.config File Reference . . . 31
6.2 makefile File Reference . . . 31
6.3 src/Adafruit_PWMServoDriver.cpp File Reference . . . 31
6.3.1 Macro Definition Documentation . . . 32
6.3.1.1 ENABLE_DEBUG_OUTPUT . . . 32

6.4 src/Adafruit_PWMServoDriver.h File Reference . . . 32
6.4.1 Macro Definition Documentation . . . 34
6.4.1.1 ALLLED_OFF_H . . . 34
6.4.1.2 ALLLED_OFF_L . . . 34
6.4.1.3 ALLLED_ON_H . . . 34
6.4.1.4 ALLLED_ON_L . . . 34
6.4.1.5 LED0_OFF_H . . . 34
6.4.1.6 LED0_OFF_L . . . 34
6.4.1.7 LED0_ON_H . . . 34
6.4.1.8 LED0_ON_L . . . 34
6.4.1.9 PCA9685_MODE1 . . . 34
6.4.1.10 PCA9685_PRESCALE . . . 35
6.4.1.11 PCA9685_SUBADR1 . . . 35
6.4.1.12 PCA9685_SUBADR2 . . . 35
6.4.1.13 PCA9685_SUBADR3 . . . 35
6.4.1.14 uint16_t . . . 35
6.4.1.15 uint8_t . . . 35
6.5 src/Arm.hpp File Reference . . . 35
6.5.1 Detailed Description . . . 36
6.6 src/Block.hpp File Reference . . . 36
6.7 src/cv_hue.cpp File Reference . . . 37
6.8 src/cv_shape.cpp File Reference . . . 38
6.8.1 Function Documentation . . . 38
6.8.1.1 main . . . 39
6.9 src/cv_test.cpp File Reference . . . 39
6.9.1 Function Documentation . . . 40
6.9.1.1 main . . . 40
6.10 src/definitions.hpp File Reference . . . 40
6.10.1 Macro Definition Documentation . . . 41
6.10.1.1 ENUM . . . 41
6.10.1.2 ERROR . . . 41
6.10.2 Typedef Documentation . . . 41
6.10.2.1 byte . . . 41
6.10.2.2 float32 . . . 42
6.10.2.3 float64 . . . 42
6.10.2.4 sint16 . . . 42
6.10.2.5 sint32 . . . 42
6.10.2.6 sint64 . . . 42
6.10.2.7 sint8 . . . 42
6.10.2.8 uint16 . . . 42

6.10.2.9 uint32 . . . 42
6.10.2.10 uint64 . . . 42
6.10.2.11 uint8 . . . 42
6.10.3 Enumeration Type Documentation . . . 42
6.10.3.1 BlockPosition . . . 42
6.10.3.2 Color . . . 43
6.10.3.3 LogMode . . . 43
6.10.3.4 Result . . . 43
6.10.3.5 Side . . . 43
6.10.3.6 Size . . . 43
6.10.3.7 Zone . . . 44
6.11 src/Grabber.hpp File Reference . . . 44
6.12 src/jacob_alg_test.cpp File Reference . . . 45
6.12.1 Function Documentation . . . 45
6.12.1.1 main . . . 45
6.13 src/loading_test.cpp File Reference . . . 45
6.13.1 Function Documentation . . . 46
6.13.1.1 main . . . 46
6.14 src/Log.hpp File Reference . . . 46
6.15 src/log_test.cpp File Reference . . . 47
6.15.1 Function Documentation . . . 48
6.15.1.1 main . . . 48
6.16 src/main.cpp File Reference . . . 48
6.16.1 Function Documentation . . . 49
6.16.1.1 main . . . 49
6.17 src/net_qr_test.cpp File Reference . . . 49
6.17.1 Function Documentation . . . 50
6.17.1.1 main . . . 50
6.18 src/old_cv_test.cpp File Reference . . . 50
6.18.1 Function Documentation . . . 51
6.18.1.1 main . . . 51
6.19 src/PiCamera.hpp File Reference . . . 51
6.20 src/qr_test.cpp File Reference . . . 52
6.20.1 Function Documentation . . . 52
6.20.1.1 main . . . 52
6.21 src/ScanQR.hpp File Reference . . . 53
6.22 src/Servo_Position_Shell.cpp File Reference . . . 54
6.22.1 Macro Definition Documentation . . . 55
6.22.1.1 SERVOMAX . . . 55
6.22.1.2 SERVOMIN . . . 55

6.22.2 Function Documentation . . . 56
6.22.2.1 setServoPosition . . . 56
6.22.2.2 setServoPulse . . . 57
6.22.2.3 setup . . . 59
6.22.3 Variable Documentation . . . 60
6.22.3.1 pwm . . . 60
6.22.3.2 servo_num . . . 60
6.23 src/Servo_Position_Shell.h File Reference . . . 61
6.23.1 Macro Definition Documentation . . . 62
6.23.1.1 uint8_t . . . 62
6.23.2 Enumeration Type Documentation . . . 62
6.23.2.1 Servo . . . 62
6.23.3 Function Documentation . . . 63
6.23.3.1 setServoPosition . . . 63
6.23.3.2 setServoPulse . . . 64
6.23.3.3 setup . . . 66

Index **68**

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

ChipChipArray
Block class 7
std 9

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Adafruit_PWMServoDriver 11
ChipChipArray::Arm 14
ChipChipArray::Block 21
ChipChipArray::Log 24
ChipChipArray::PiCamera 28

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

makefile . . . 31
etc/doxygen.config . . . 31
src/Adafruit_PWMServoDriver.cpp . . . 31
src/Adafruit_PWMServoDriver.h . . . 32
src/Arm.hpp
    Arm class used to control the robotic arm . . . 35
src/Block.hpp . . . 36
src/cv_hue.cpp . . . 37
src/cv_shape.cpp . . . 38
src/cv_test.cpp . . . 39
src/definitions.hpp . . . 40
src/Grabber.hpp . . . 44
src/jacob_alg_test.cpp . . . 45
src/loading_test.cpp . . . 45
src/Log.hpp . . . 46
src/log_test.cpp . . . 47
src/main.cpp . . . 48
src/net_qr_test.cpp . . . 49
src/old_cv_test.cpp . . . 50
src/PiCamera.hpp . . . 51
src/qr_test.cpp . . . 52
src/ScanQR.hpp . . . 53
src/Servo_Position_Shell.cpp . . . 54
src/Servo_Position_Shell.h . . . 61

Chapter 4

Namespace Documentation

4.1 ChipChipArray Namespace Reference

contains Block class

Classes

- class Arm
- class Block
- class Log
- class PiCamera

Functions

- int main (int argc, char **argv)
- Color ScanQR ()

Variables

- uint8 qrInvokeCount = 0
- Log scanQrLog ("logs/ScanQR", LogMode::Multi)

4.1.1 Detailed Description

contains Block class

contains ScanQR() function

contains PiCamera class

contains Log class

Author

Samuel Andrew Wisner, awisner94@gmail.com

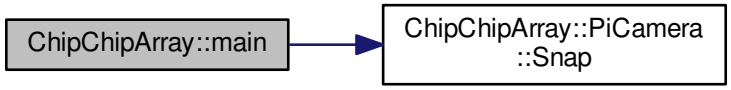
4.1.2 Function Documentation

4.1.2.1 int ChipChipArray::main (int *argc*, char ** *argv*)

This program was used before cv_shape.cpp was written to find HSV ranges for the different color blocks. This is a slightly modified version of some code written by Shermal Fernando in the blog post "Color Detection & Object Tracking" at http://opencv-srf.blogspot.com/2010/09/object-detection-using-color-seperation.html".

Definition at line 26 of file cv_hue.cpp.

Here is the call graph for this function:



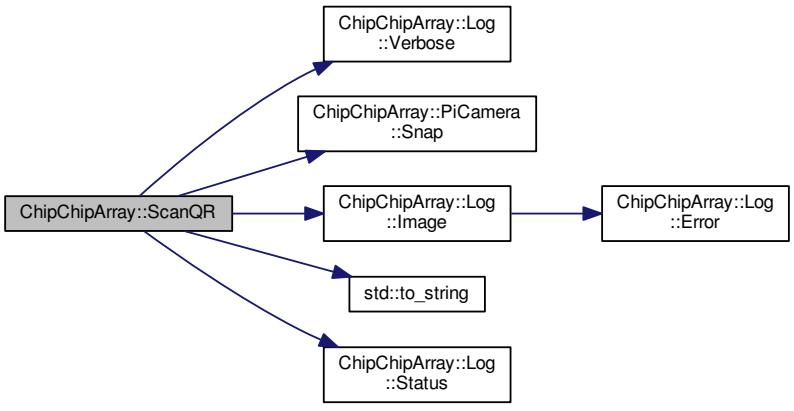
4.1.2.2 Color ChipChipArray::ScanQR ( )

This function manuvers arm to look at the QR code on a train car as the robot is backing up to the car. It attempts to find the code in multiple images before finally throwing an exeption if a code is not found. If multiple codes are found, it returns a single Color by (seemingly) arbitrary decision.

This function is based on code written by Michael Young (https://github.com/ayoungprogrammer/WebcamCodeScanner).

Definition at line 33 of file ScanQR.hpp.

Here is the call graph for this function:



Here is the caller graph for this function:

4.1.3 Variable Documentation

4.1.3.1 uint8 ChipChipArray::qrInvokeCount = 0

Definition at line 20 of file ScanQR.hpp.

4.1.3.2 Log ChipChipArray::scanQrLog("logs/ScanQR", LogMode::Multi)

4.2 std Namespace Reference

Functions

- string to_string (BlockPosition pos)
- string to_string (Color color)
- string to_string (LogMode mode)
- string to_string (Result res)
- string to_string (Side side)
- string to_string (Size size)
- string to_string (Zone zone)

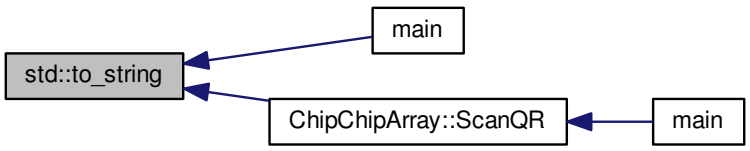
4.2.1 Function Documentation

4.2.1.1 string std::to_string (BlockPosition *pos*)

Converts a BlockPosition to a string.

Definition at line 88 of file definitions.hpp.

Here is the caller graph for this function:



4.2.1.2 string std::to_string (Color *color*)

Converts a Color to a string.

Definition at line 96 of file definitions.hpp.

4.2.1.3 string std::to_string (LogMode *mode*)

Converts a LogMode to a string.

Definition at line 123 of file definitions.hpp.

4.2.1.4 string std::to_string (Result *res*)

Converts a Result to a string.

Definition at line 131 of file definitions.hpp.

4.2.1.5 string std::to_string (Side *side*)

Converts a Side to a string.

Definition at line 158 of file definitions.hpp.

4.2.1.6 string std::to_string (Size *size*)

Converts a Size to a string.

Definition at line 166 of file definitions.hpp.

4.2.1.7 string std::to_string (Zone *zone*)

Converts a Zone to a string.

Definition at line 174 of file definitions.hpp.

Chapter 5

Class Documentation

5.1 Adafruit_PWMServoDriver Class Reference

`#include <Adafruit_PWMServoDriver.h>`

Public Member Functions

- Adafruit_PWMServoDriver (uint8_t addr=0x41)
- void begin (void)
- void reset (void)
- void setPWMFreq (float freq)
- void setPWM (uint8_t num, uint16_t on, uint16_t off)
- void setPin (uint8_t num, uint16_t val, bool invert=false)

5.1.1 Detailed Description

Definition at line 65 of file Adafruit_PWMServoDriver.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Adafruit_PWMServoDriver::Adafruit_PWMServoDriver (uint8_t *addr* = `0x41`)

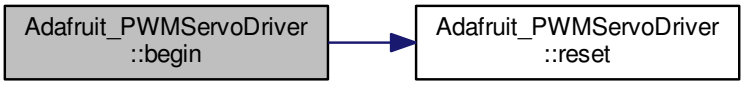
Definition at line 29 of file Adafruit_PWMServoDriver.cpp.

5.1.3 Member Function Documentation

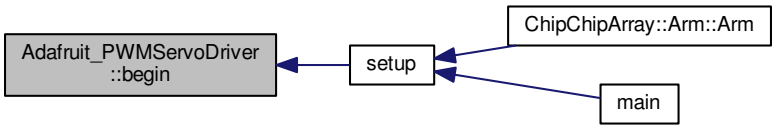
5.1.3.1 void Adafruit_PWMServoDriver::begin (void)

Definition at line 34 of file Adafruit_PWMServoDriver.cpp.

Here is the call graph for this function:



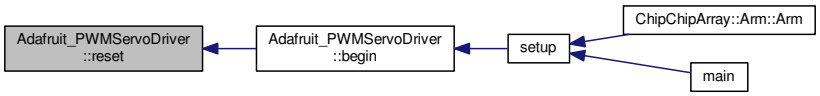
Here is the caller graph for this function:



5.1.3.2 void Adafruit_PWMServoDriver::reset (void)

Definition at line 42 of file Adafruit_PWMServoDriver.cpp.

Here is the caller graph for this function:



5.1.3.3 void Adafruit_PWMServoDriver::setPin (uint8_t *num*, uint16_t *val*, bool *invert* = `false`)

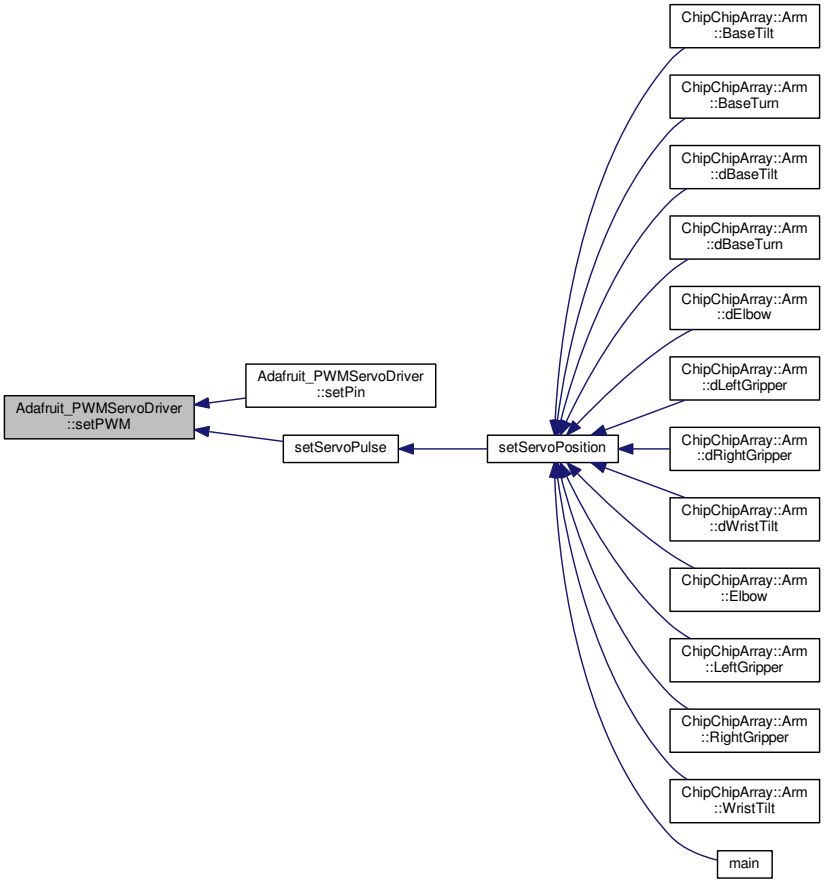
Definition at line 108 of file Adafruit_PWMServoDriver.cpp.

Here is the call graph for this function:

5.1.3.4 void Adafruit_PWMServoDriver::setPWM (uint8_t *num*, uint16_t *on*, uint16_t *off*)

Definition at line 73 of file Adafruit_PWMServoDriver.cpp.

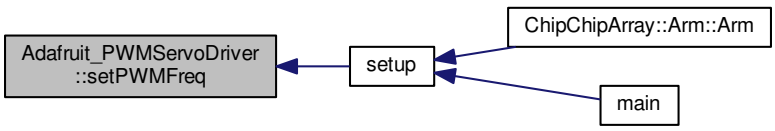
Here is the caller graph for this function:



5.1.3.5 void Adafruit_PWMServoDriver::setPWMFreq (float *freq*)

Definition at line 46 of file Adafruit_PWMServoDriver.cpp.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- src/Adafruit_PWMServoDriver.h
- src/Adafruit_PWMServoDriver.cpp

5.2 ChipChipArray::Arm Class Reference

`#include <Arm.hpp>`

Public Member Functions

- Arm ()
- void BaseTilt (uint8 a)
- void BaseTurn (uint8 a)
- void dBaseTilt (sint16 a)
- void dBaseTurn (sint16 a)
- void dElbow (sint16 a)
- void dGrippers (sint16 a)
- void dWristTilt (sint16 a)
- void dWristTwist (sint16 a)
- void Elbow (uint8 a)
- void Grippers (uint8 a)
- void Hover (Zone zone)
- void WristTilt (uint8 a)
- void WristTwist (uint8 a)

Protected Member Functions

- void dLeftGripper (sint16 a)
- void dRightGripper (sint16 a)
- void LeftGripper (uint8 a)
- void RightGripper (uint8 a)

Protected Attributes

- uint8 servoPos = { 0, 0, 0, 0, 0, 0, 0 }

5.2.1 Detailed Description

This class provides a layer of abstraction from the existing servo interface. It is designed to make more sense programmatically and to be easier to use.

Definition at line 19 of file Arm.hpp.

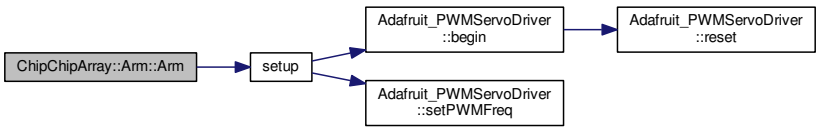
5.2.2 Constructor & Destructor Documentation

5.2.2.1 ChipChipArray::Arm::Arm ( )

Initializes the I2C interface for the arm if another instance of the Arm class has not already.

Definition at line 174 of file Arm.hpp.

Here is the call graph for this function:



5.2.3 Member Function Documentation

5.2.3.1 void ChipChipArray::Arm::BaseTilt (uint8 *a*)

Tilts the base of the arm.

Parameters

a	desired servo position in degrees

Definition at line 181 of file Arm.hpp.

Here is the call graph for this function:

5.2.3.2 void ChipChipArray::Arm::BaseTurn (uint8 *a*)

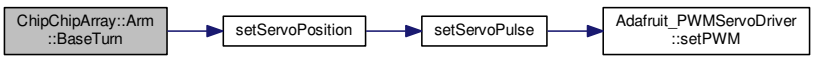
Twists the entire arm.

Parameters

a	desired servo position in degrees

Definition at line 186 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.3 void ChipChipArray::Arm::dBaseTilt (sint16 *a*)

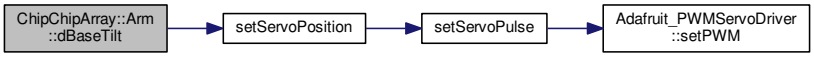
Tilts the base a certain number of degrees.

Parameters

degrees	to move servo. Positive values add to the servo angle, and negative values subtract from the servo angle.

Definition at line 191 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.4 void ChipChipArray::Arm::dBaseTurn (sint16 *a*)

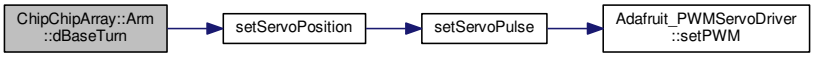
Turn the base a certain number of degrees.

Parameters

degrees	to move servo. Positive values add to the servo angle, and negative values subtract from the servo angle.

Definition at line 197 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.5 void ChipChipArray::Arm::dElbow (sint16 *a*)

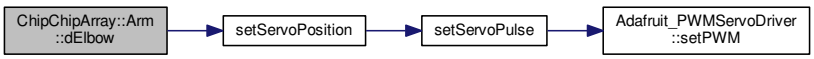
Bend the elbow a certain number of degrees.

Parameters

degrees	to move servo. Positive values add to the servo angle, and negative values subtract from the servo angle.

Definition at line 203 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.6 void ChipChipArray::Arm::dGrippers (sint16 *a*)

Move the grippers a certain number of degrees. Note that they will both move inward or outward; one will never move inward and the other outward.

Parameters

degrees	to move servo.

Definition at line 209 of file Arm.hpp.

5.2.3.7 void ChipChipArray::Arm::dLeftGripper (sint16 *a*) [protected]

Moves the left gripper servo a certain number of degrees.

Parameters

degrees	to move servo. Positive values add to the servo angle, and negative values subtract from the servo angle.

Definition at line 216 of file Arm.hpp.

Here is the call graph for this function:

5.2.3.8 void ChipChipArray::Arm::dRightGripper (sint16 *a*) [protected]

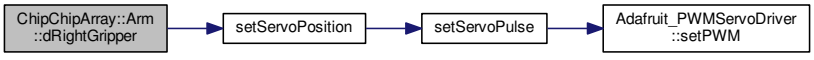
Moves the right gripper servo a certain number of degrees.

Parameters

degrees	to move servo. Positive values add to the servo angle, and negative values subtract from the servo angle.

Definition at line 222 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.9 void ChipChipArray::Arm::dWristTilt (sint16 *a*)

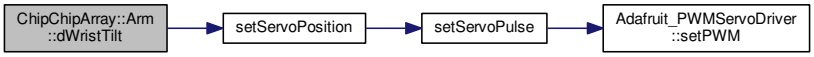
Tilt the wrist a certain number of degrees.

Parameters

degrees	to move servo. Positive values add to the servo angle, and negative values subtract from the servo angle.

Definition at line 228 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.10 void ChipChipArray::Arm::dWristTwist (sint16 *a*)

Twist the wrist a certain number of degrees.

Parameters

degrees	to move servo. Positive values add to the servo angle, and negative values subtract from the servo angle.

5.2.3.11 void ChipChipArray::Arm::Elbow (uint8 *a*)

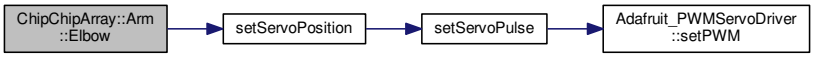
Bend the elbow to a specific position.

Parameters

a	desired servo position in degrees

Definition at line 240 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.12 void ChipChipArray::Arm::Grippers (uint8 *a*)

Move the grippers to a specific position. Note that they will both move inward or outward; one will never move inward and the other outward.

Parameters

a	desired servo position in degrees

Definition at line 245 of file Arm.hpp.

5.2.3.13 void ChipChipArray::Arm::Hover (Zone *zone*)

Moves arm into its "hovering" position over the blocks. The position changes with the zone.

Parameters

zone	the zone for which the arm should position itself

Definition at line 249 of file Arm.hpp.

5.2.3.14 void ChipChipArray::Arm::LeftGripper (uint8 *a*) [protected]

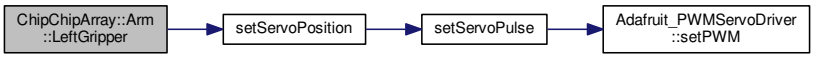
Moves the left gripper to a specific position.

Parameters

a	desired servo position in degrees

Definition at line 253 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.15 void ChipChipArray::Arm::RightGripper (uint8 *a*) [protected]

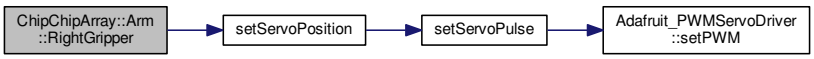
Moves the right gripper to a specific position.

Parameters

a	desired servo position in degrees

Definition at line 258 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.16 void ChipChipArray::Arm::WristTilt (uint8 *a*)

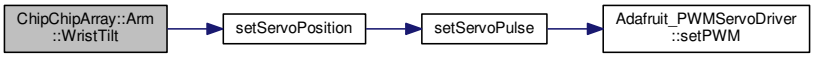
Tilt the wrist to a specific position.

Parameters

a	desired servo position in degrees

Definition at line 263 of file Arm.hpp.

Here is the call graph for this function:



5.2.3.17 void ChipChipArray::Arm::WristTwist (uint8 *a*)

Twist the wrist to a specific position.

Parameters

a	desired servo position in degrees

5.2.4 Member Data Documentation

5.2.4.1 uint8 ChipChipArray::Arm::servoPos = { 0, 0, 0, 0, 0, 0 } [protected]

The instantaneous position of each arm servo.

Definition at line 132 of file Arm.hpp.

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

- src/Arm.hpp

5.3 ChipChipArray::Block Class Reference

`#include <Block.hpp>`

Public Member Functions

- Block (cv::Rect rect, Color color)

Public Attributes

- uint32 area
- cv::Point bottomLeft
- cv::Point bottomRight
- sint16 dBottom
- sint16 dLeft
- sint16 dRight
- sint16 dTop
- sint16 dTopBottom
- sint16 dRightLeft
- sint16 offset
- uint16 height
- cv::Point topLeft
- cv::Point topRight

- uint16 width
- Color color
- Size size

5.3.1 Detailed Description

This class represents a block. It only works for blocks found with the "boundingRect" algorithm (i.e., it doesn't work for blocks that are skewed on the image).

Definition at line 19 of file Block.hpp.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ChipChipArray::Block::Block (cv::Rect *rect,* Color *color*)

Creates a new Block using the Points in the cv::Rect and the color. Also determines the size based on the area of the Block.

Definition at line 138 of file Block.hpp.

5.3.3 Member Data Documentation

5.3.3.1 uint32 ChipChipArray::Block::area

The area of the block in pixels

Definition at line 24 of file Block.hpp.

5.3.3.2 cv::Point ChipChipArray::Block::bottomLeft

Point of the block's bottom-left corner

Definition at line 29 of file Block.hpp.

5.3.3.3 cv::Point ChipChipArray::Block::bottomRight

Point of the block's bottom-right corner

Definition at line 34 of file Block.hpp.

5.3.3.4 Color ChipChipArray::Block::color

The detected color of the block

Definition at line 106 of file Block.hpp.

5.3.3.5 sint16 ChipChipArray::Block::dBottom

Number of pixels from the block's bottom edge to the bottom edge of the image frame.

Definition at line 40 of file Block.hpp.

5.3.3.6 sint16 ChipChipArray::Block::dLeft

Number of pixels from the block's left edge to the left edge of the image frame.

Definition at line 46 of file Block.hpp.

5.3.3.7 sint16 ChipChipArray::Block::dRight

Number of pixels from the block's right edge to the right edge of the image frame.

Definition at line 52 of file Block.hpp.

5.3.3.8 sint16 ChipChipArray::Block::dRightLeft

The difference between dRight and dLeft. It indicates the relative vertical positioning of the block regardless of the block's area. A positive value indicates the block is off-center towards the left.

Definition at line 74 of file Block.hpp.

5.3.3.9 sint16 ChipChipArray::Block::dTop

Number of pixels from the block's top edge to the top edge of the image frame.

Definition at line 58 of file Block.hpp.

5.3.3.10 sint16 ChipChipArray::Block::dTopBottom

The difference between dTop and dBottom. It indicates the relative vertical positioning of the block regardless of the block's area. A positive value indicates the block is off-center towards the bottom.

Definition at line 66 of file Block.hpp.

5.3.3.11 uint16 ChipChipArray::Block::height

The height of the block in pixels

Definition at line 86 of file Block.hpp.

5.3.3.12 sint16 ChipChipArray::Block::offset

The difference in pixels between the vertical center of the image and the vertical center of the block. Assumes image is 1280 pixels wide (like the Raspicam images).

Definition at line 81 of file Block.hpp.

5.3.3.13 Size ChipChipArray::Block::size

The size of the block (half or whole)

Definition at line 111 of file Block.hpp.

5.3.3.14 cv::Point ChipChipArray::Block::topLeft

Point of the block's top-left corner

Definition at line 91 of file Block.hpp.

5.3.3.15 cv::Point ChipChipArray::Block::topRight

Point of the block's top-right corner

Definition at line 96 of file Block.hpp.

5.3.3.16 uint16 ChipChipArray::Block::width

The width of the block in pixels

Definition at line 101 of file Block.hpp.

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

- src/Block.hpp

5.4 ChipChipArray::Log Class Reference

`#include <Log.hpp>`

Public Member Functions

- Log ()
- Log (auto dir, LogMode mode=LogMode::Text)
- ~Log ()
- void Debug (auto mesg)
- void Error (auto mesg)
- void Image (cv::Mat image, auto filename)
- void Open (auto dir, LogMode mode=LogMode::Text)
- void Status (auto mesg)
- void Variable (auto name, auto value)
- void Verbose (auto mesg)

5.4.1 Detailed Description

This class logs the text and images passed to it to specificed directory.

A "container" directory to which the class can write is passed in the constructor. When the Log is initialized with LogMode::Text, a new log file is created with a filename based on the time of initialization in the given directory. When initialized in LogMode::Multi, it will create a subdirectory in the given directory with a name based on time. In this new directory, a log file will be created. Images may later be stored in this directory with names based on the order in which they were saved.

This class DOES NOT WORK without compiling without a "LOG" definition (#define LOG or -DLOG).

Definition at line 34 of file Log.hpp.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ChipChipArray::Log::Log ( ) [inline]

Initializes Log object but does not open log. Open() must be called.

Definition at line 40 of file Log.hpp.

5.4.2.2 ChipChipArray::Log::Log (auto *dir,* LogMode *mode* = LogMode::Text)

Initializes the Log.

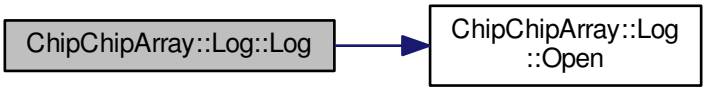
A new log file is created in dir if LogMode::Text is given. The file will have a name based on the current date and time. If LogMode::Multi is given, a new directory is created, and a log file with a name based on the current date and time is created inside it.

Parameters

dir	the directory for the newly created logfile/folder
mode	the LogMode

Definition at line 186 of file Log.hpp.

Here is the call graph for this function:



5.4.2.3 ChipChipArray::Log::~Log ( )

Destroys the Log and closes the logfile.

Definition at line 192 of file Log.hpp.

5.4.3 Member Function Documentation

5.4.3.1 void ChipChipArray::Log::Debug (auto *mesg*)

Writes "DEBUG: " to the log file along with the message passed. Should be used for generic debugging information. If recording the value of a variable in the Log is desired, use the function Variable() instead.

Parameters

mesg	the message to record in the logfile

Definition at line 204 of file Log.hpp.

Here is the caller graph for this function:

5.4.3.2 void ChipChipArray::Log::Error (auto *mesg*)

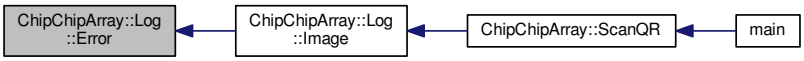
Writes "ERROR: " to the log file. Should only be use when an exception is thrown.

Parameters

mesg	the message to record in the log

Definition at line 215 of file Log.hpp.

Here is the caller graph for this function:



5.4.3.3 void ChipChipArray::Log::Image (cv::Mat *image,* auto *filename*)

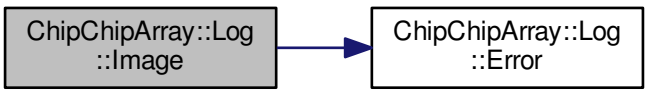
Creates a bitmap image in the subdirectory created by the Log during initialization. Does nothing if LogMode::Text was passed in the constructor.

Parameters

image	the image to save
filename	the filename for the saved image

Definition at line 226 of file Log.hpp.

Here is the call graph for this function:

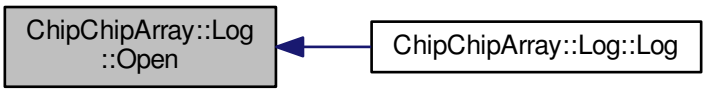


Here is the caller graph for this function:

5.4.3.4 void ChipChipArray::Log::Open (auto *dir,* LogMode *mode* = LogMode::Text)

Definition at line 246 of file Log.hpp.

Here is the caller graph for this function:



5.4.3.5 void ChipChipArray::Log::Status (auto *mesg*)

Writes "STATUS: " to the log file. Should be used when recording the status or state of the program. It should not be used to record microalgorithmic changes. Use Verbose() for these instead.

Parameters

mesg	the message to record in the logfile

Definition at line 289 of file Log.hpp.

Here is the caller graph for this function:

5.4.3.6 void ChipChipArray::Log::Variable (auto *name,* auto *value*)

Writes "VARIABLE: " to the log file. Should be used whenever recording the value of a variable is desired.

Parameters

name	the variable name to record
value	the variable value to record

Definition at line 300 of file Log.hpp.

Here is the caller graph for this function:

5.4.3.7 void ChipChipArray::Log::Verbose (auto *mesg*)

Writes "VERBOSE: " to the log file. Should only be used for recording small, specific portions of code. To record a change in the more general state of the program, use Status() instead.

Parameters

mesg	the message to record in the logfile

Definition at line 312 of file Log.hpp.

Here is the caller graph for this function:

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

- src/Log.hpp

5.5 ChipChipArray::PiCamera Class Reference

`#include <PiCamera.hpp>`

Public Member Functions

- PiCamera ()
- PiCamera (bool useColor)
- void Close ()
- cv::Mat Snap ()

5.5.1 Detailed Description

This class is a basic wrapper to allow the Raspicam to interface with OpenCV. It uses another wrapper class, Raspicam, provided by Cédric Verstraeten (https://github.com/cedricve/raspicam).

Definition at line 23 of file PiCamera.hpp.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 ChipChipArray::PiCamera::PiCamera ( ) [inline]

Opens the camera and configures it for color images.

Definition at line 28 of file PiCamera.hpp.

5.5.2.2 ChipChipArray::PiCamera::PiCamera (bool *useColor*)

Opens the camera.

Parameters

useColor	Specifices whether camera should make color images. TRUE = color, FALSE = grayscale.

Definition at line 58 of file PiCamera.hpp.

5.5.3 Member Function Documentation

5.5.3.1 void ChipChipArray::PiCamera::Close ( )

Closes connection to camera.

Definition at line 64 of file PiCamera.hpp.

5.5.3.2 cv::Mat ChipChipArray::PiCamera::Snap ( )

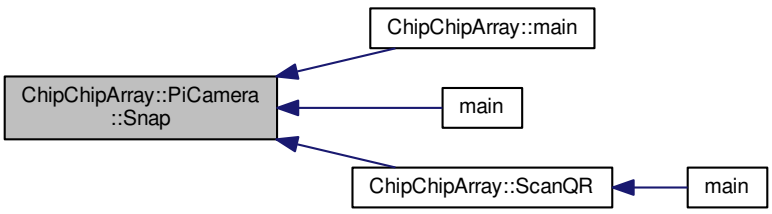
Makes picture.

Returns

OpenCV Mat object (i.e., an image) from the camera

Definition at line 68 of file PiCamera.hpp.

Here is the caller graph for this function:



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

- src/PiCamera.hpp

Chapter 6

File Documentation

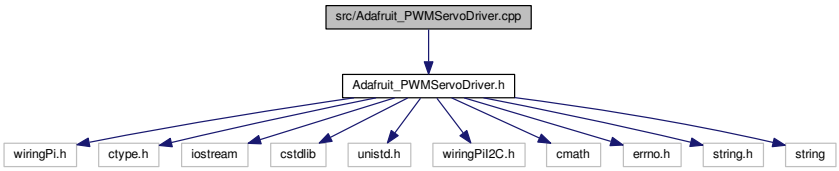
6.1 etc/doxygen.config File Reference

6.2 makefile File Reference

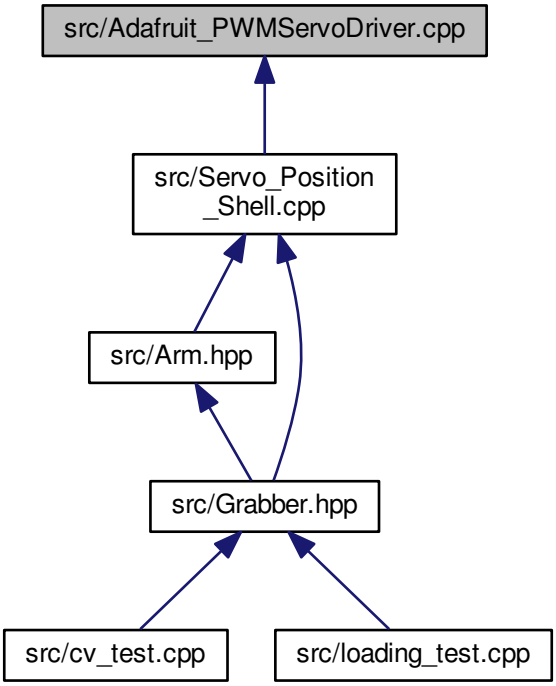
6.3 src/Adafruit_PWMServoDriver.cpp File Reference

```
#include "Adafruit_PWMServoDriver.h"
```

Include dependency graph for Adafruit_PWMServoDriver.cpp:



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



Macros

- #define ENABLE_DEBUG_OUTPUT false

6.3.1 Macro Definition Documentation

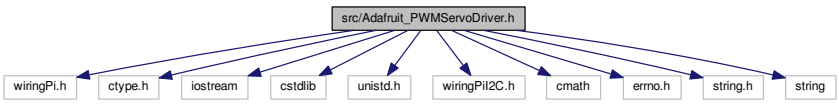
6.3.1.1 #define ENABLE_DEBUG_OUTPUT false

Definition at line 27 of file Adafruit_PWMServoDriver.cpp.

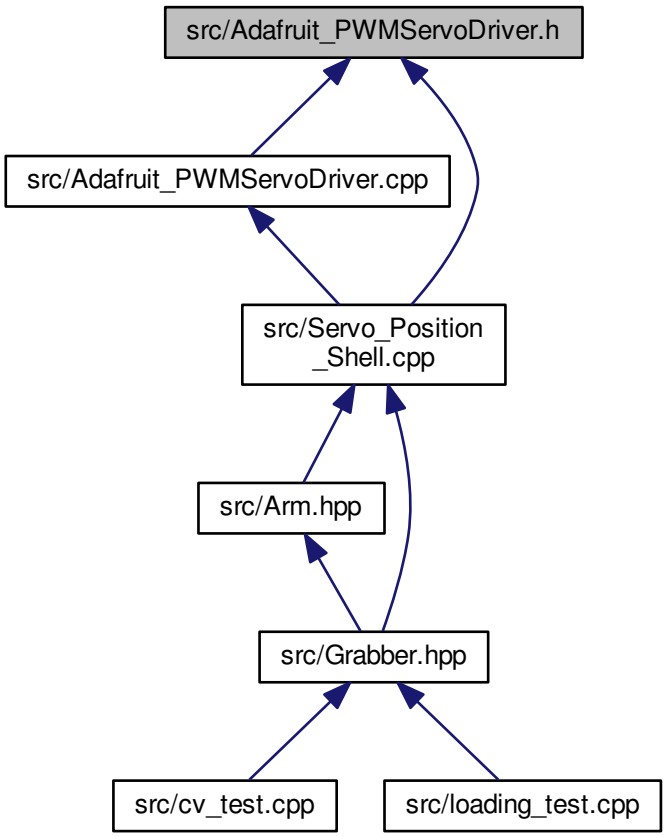
6.4 src/Adafruit_PWMServoDriver.h File Reference

```
#include <wiringPi.h>
#include <ctype.h>
#include <iostream>
#include <cstdlib>
#include <unistd.h>
#include <wiringPiI2C.h>
#include <cmath>
#include <errno.h>
#include <string.h>
#include <string>
```

Include dependency graph for Adafruit_PWMServoDriver.h:



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



Classes

- class Adafruit_PWMServoDriver

Macros

- #define PCA9685_SUBADR1 0x2
- #define PCA9685_SUBADR2 0x3
- #define PCA9685_SUBADR3 0x4

- #define PCA9685_MODE1 0x0
- #define PCA9685_PRESCALE 0xFE
- #define LED0_ON_L 0x6
- #define LED0_ON_H 0x7
- #define LED0_OFF_L 0x8
- #define LED0_OFF_H 0x9
- #define ALLLED_ON_L 0xFA
- #define ALLLED_ON_H 0xFB
- #define ALLLED_OFF_L 0xFC
- #define ALLLED_OFF_H 0xFD
- #define uint8_t unsigned char
- #define uint16_t unsigned short int

6.4.1 Macro Definition Documentation

6.4.1.1 #define ALLLED_OFF_H 0xFD

Definition at line 54 of file Adafruit_PWMServoDriver.h.

6.4.1.2 #define ALLLED_OFF_L 0xFC

Definition at line 53 of file Adafruit_PWMServoDriver.h.

6.4.1.3 #define ALLLED_ON_H 0xFB

Definition at line 52 of file Adafruit_PWMServoDriver.h.

6.4.1.4 #define ALLLED_ON_L 0xFA

Definition at line 51 of file Adafruit_PWMServoDriver.h.

6.4.1.5 #define LED0_OFF_H 0x9

Definition at line 49 of file Adafruit_PWMServoDriver.h.

6.4.1.6 #define LED0_OFF_L 0x8

Definition at line 48 of file Adafruit_PWMServoDriver.h.

6.4.1.7 #define LED0_ON_H 0x7

Definition at line 47 of file Adafruit_PWMServoDriver.h.

6.4.1.8 #define LED0_ON_L 0x6

Definition at line 46 of file Adafruit_PWMServoDriver.h.

6.4.1.9 #define PCA9685_MODE1 0x0

Definition at line 43 of file Adafruit_PWMServoDriver.h.

6.4.1.10 #define PCA9685_PRESCALE 0xFE

Definition at line 44 of file Adafruit_PWMServoDriver.h.

6.4.1.11 #define PCA9685_SUBADR1 0x2

Definition at line 39 of file Adafruit_PWMServoDriver.h.

6.4.1.12 #define PCA9685_SUBADR2 0x3

Definition at line 40 of file Adafruit_PWMServoDriver.h.

6.4.1.13 #define PCA9685_SUBADR3 0x4

Definition at line 41 of file Adafruit_PWMServoDriver.h.

6.4.1.14 #define uint16_t unsigned short int

Definition at line 61 of file Adafruit_PWMServoDriver.h.

6.4.1.15 #define uint8_t unsigned char

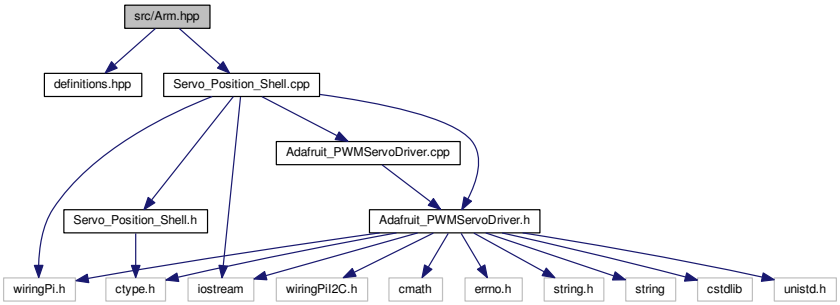
Definition at line 57 of file Adafruit_PWMServoDriver.h.

6.5 src/Arm.hpp File Reference

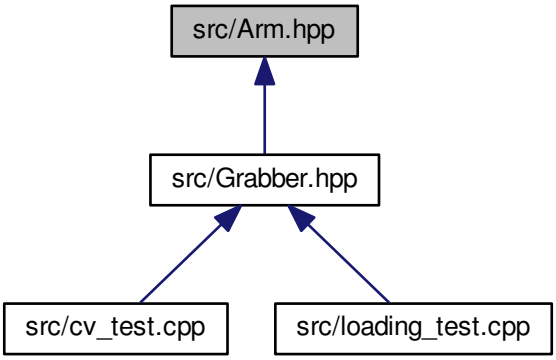
contains the Arm class used to control the robotic arm

```
#include "definitions.hpp"
#include "Servo_Position_Shell.cpp"
```

Include dependency graph for Arm.hpp:



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



Classes

- class ChipChipArray::Arm

Namespaces

- ChipChipArray

  contains Block class

6.5.1 Detailed Description

contains the Arm class used to control the robotic arm

Author

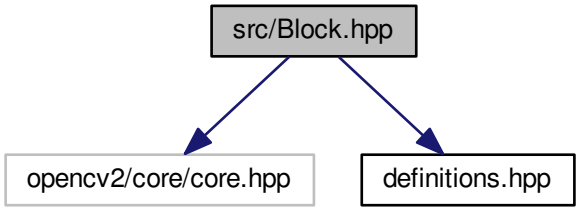
Samuel Andrew Wisner, awisner94@gmail.com

Definition in file Arm.hpp.

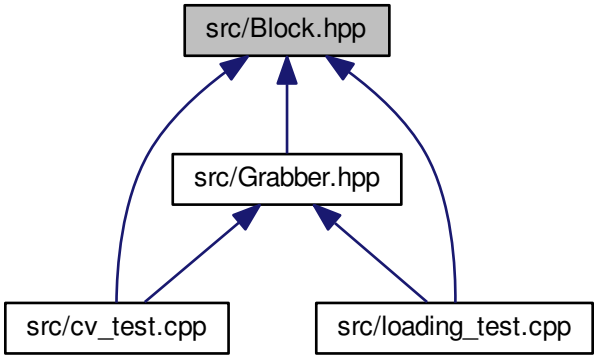
6.6 src/Block.hpp File Reference

```
#include <opencv2/core/core.hpp>
#include "definitions.hpp"
```

Include dependency graph for Block.hpp:



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



Classes

- class ChipChipArray::Block

Namespaces

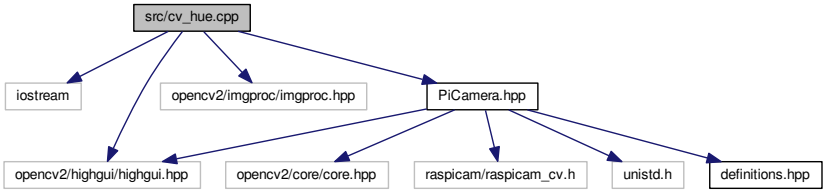
- ChipChipArray

  contains Block class

6.7 src/cv_hue.cpp File Reference

```
#include <iostream>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "PiCamera.hpp"
```

Include dependency graph for cv_hue.cpp:



Namespaces

- ChipChipArray

  contains Block class

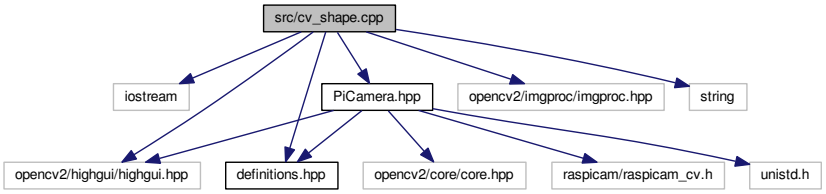
Functions

- int ChipChipArray::main (int argc, char **argv)

6.8 src/cv_shape.cpp File Reference

```
#include <iostream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <string>
#include "definitions.hpp"
#include "PiCamera.hpp"
```

Include dependency graph for cv_shape.cpp:



Functions

- int main ()

6.8.1 Function Documentation

6.8.1.1 int main ( )

This program (a single function) is a test of the computer vision algorithms for loading the blocks. It will likely be in development for some time to come. The plan currently is to develop and test all CV algorithms for block loading here before moving it all into class functions and testing again.

This code is based on several online articles:

- "Color Detectionn & Object Tracking" by Shermal Fernando (http://opencv-srf.blogspot.com/2010/09/object-detection-using-color-seperation.html)
- "Shape Detection & Tracking using Contours" by Shermal Fernando (http://opencv-srf.blogspot.com/2011/09/object-detection-tracking-using-contours.html)
- "Creating Bounding boxes and circles for contours" in the OpenCV 2.4 Tutorials (http://opencv-srf.blogspot.com/2011/09/object-detection-tracking-using-contours.html)

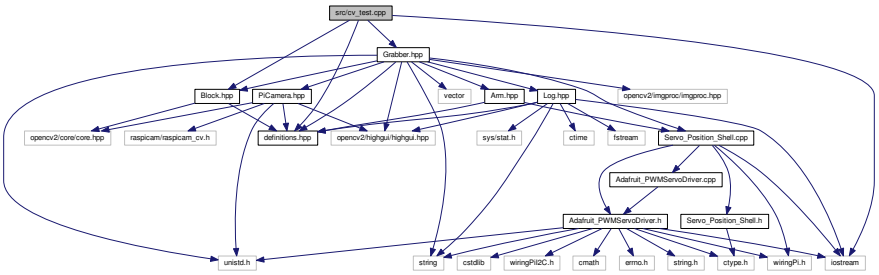
Definition at line 36 of file cv_shape.cpp.

Here is the call graph for this function:

6.9 src/cv_test.cpp File Reference

```
#include <iostream>
#include "definitions.hpp"
#include "Block.hpp"
#include "Grabber.hpp"
```

Include dependency graph for cv_test.cpp:



Functions

- int main ()

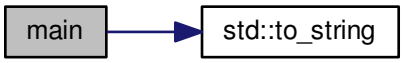
6.9.1 Function Documentation

6.9.1.1 int main ( )

This program was used solely to test the PiCamera wrapper class and its compatibility with the raspicam wrapper and ultimately OpenCV.

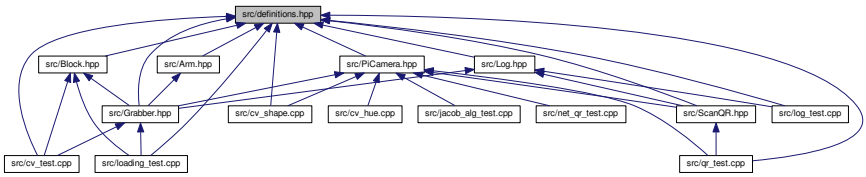
Definition at line 18 of file cv_test.cpp.

Here is the call graph for this function:



6.10 src/definitions.hpp File Reference

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



Namespaces

- std

Macros

- #define ENUM signed char

  contains definitions for architecture-independant numeric variables, enumerations and enumerated classes, and #define'd constants, and to_sting() overloads for the enumerated classes.
- #define ERROR -1

Typedefs

- typedef unsigned char byte
- typedef unsigned char uint8
- typedef signed char sint8
- typedef unsigned short uint16
- typedef signed short sint16
- typedef unsigned int uint32
- typedef signed int sint32

- typedef unsigned long long uint64
- typedef signed long long sint64
- typedef float float32
- typedef double float64

Enumerations

- enum BlockPosition : ENUM { BlockPosition::Front, BlockPosition::Back }
- enum Color : ENUM {
  Color::Red, Color::Yellow, Color::Green, Color::Blue,
  Color::Perrywinkle }
- enum LogMode : ENUM { LogMode::Text, LogMode::Multi }
- enum Result : ENUM { Result::No_Blocks = -1, Result::No_Halves = 0, Result::Two_Halves = 2, Result::Four_Halves = 4 }
- enum Side : ENUM { Side::Left, Side::Right }
- enum Size : ENUM { Size::Short, Size::Long }
- enum Zone : ENUM { Zone::A = 'A', Zone::B = 'B', Zone::C = 'C' }

Functions

- string std::to_string (BlockPosition pos)
- string std::to_string (Color color)
- string std::to_string (LogMode mode)
- string std::to_string (Result res)
- string std::to_string (Side side)
- string std::to_string (Size size)
- string std::to_string (Zone zone)

6.10.1 Macro Definition Documentation

6.10.1.1 #define ENUM signed char

contains definitions for architecture-independant numeric variables, enumerations and enumerated classes, and #define'd constants, and to_sting() overloads for the enumerated classes.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition at line 11 of file definitions.hpp.

6.10.1.2 #define ERROR -1

Definition at line 12 of file definitions.hpp.

6.10.2 Typedef Documentation

6.10.2.1 typedef unsigned char byte

Definition at line 14 of file definitions.hpp.

6.10.2.2 typedef float float32

Definition at line 27 of file definitions.hpp.

6.10.2.3 typedef double float64

Definition at line 28 of file definitions.hpp.

6.10.2.4 typedef signed short sint16

Definition at line 19 of file definitions.hpp.

6.10.2.5 typedef signed int sint32

Definition at line 22 of file definitions.hpp.

6.10.2.6 typedef signed long long sint64

Definition at line 25 of file definitions.hpp.

6.10.2.7 typedef signed char sint8

Definition at line 16 of file definitions.hpp.

6.10.2.8 typedef unsigned short uint16

Definition at line 18 of file definitions.hpp.

6.10.2.9 typedef unsigned int uint32

Definition at line 21 of file definitions.hpp.

6.10.2.10 typedef unsigned long long uint64

Definition at line 24 of file definitions.hpp.

6.10.2.11 typedef unsigned char uint8

Definition at line 15 of file definitions.hpp.

6.10.3 Enumeration Type Documentation

6.10.3.1 enum BlockPosition : ENUM [strong]

The position of the block relative to the arm.

Enumerator

Front

Back

Definition at line 31 of file definitions.hpp.

6.10.3.2 enum Color : ENUM [strong]

The color of a block or train car.

Enumerator

Red
Yellow
Green
Blue
Perrywinkle

Definition at line 37 of file definitions.hpp.

6.10.3.3 enum LogMode : ENUM [strong]

The mode in which the Log should prepare (i.e., text only or text and images).

Enumerator

Text
Multi

Definition at line 49 of file definitions.hpp.

6.10.3.4 enum Result : ENUM [strong]

The number of half blocks picked up in a stack. The integer value of the

Enumerator

No_Blocks
No_Halves
Two_Halves
Four_Halves

Definition at line 58 of file definitions.hpp.

6.10.3.5 enum Side : ENUM [strong]

Represents which block to pick up when multiple blocks are visible.

Enumerator

Left
Right

Definition at line 66 of file definitions.hpp.

6.10.3.6 enum Size : ENUM [strong]

The block size, either 2.5" or 5".

Enumerator

Short
Long

Definition at line 72 of file definitions.hpp.

6.10.3.7 enum Zone : ENUM [strong]

Zone A, B, or C

Enumerator

A

B

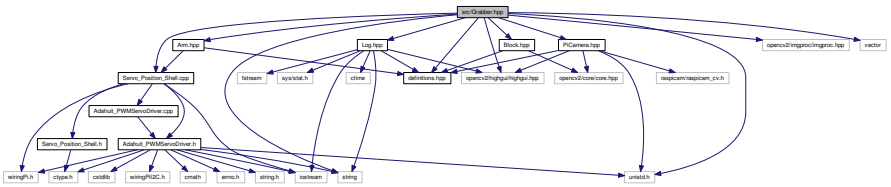
C

Definition at line 78 of file definitions.hpp.

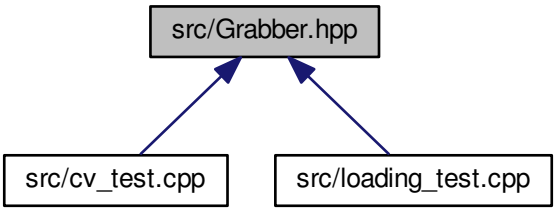
6.11 src/Grabber.hpp File Reference

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <string>
#include <unistd.h>
#include <vector>
#include "Arm.hpp"
#include "definitions.hpp"
#include "Block.hpp"
#include "Log.hpp"
#include "PiCamera.hpp"
#include "Servo_Position_Shell.cpp"
```

Include dependency graph for Grabber.hpp:



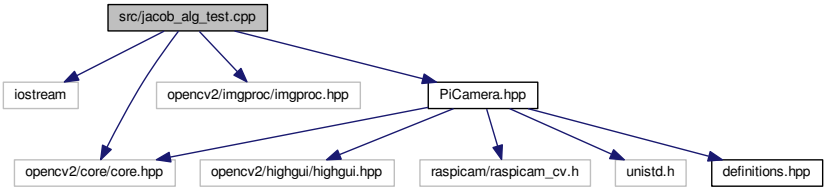
This graph shows which files directly or indirectly include this file:



6.12 src/jacob_alg_test.cpp File Reference

```
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include "PiCamera.hpp"
```

Include dependency graph for jacob_alg_test.cpp:



Functions

- int main ()

6.12.1 Function Documentation

6.12.1.1 int main ( )

This program tests Jacob Laurel's algorithm for detecting yellow blocks (RGB=>YUV, HSV=>RGB).

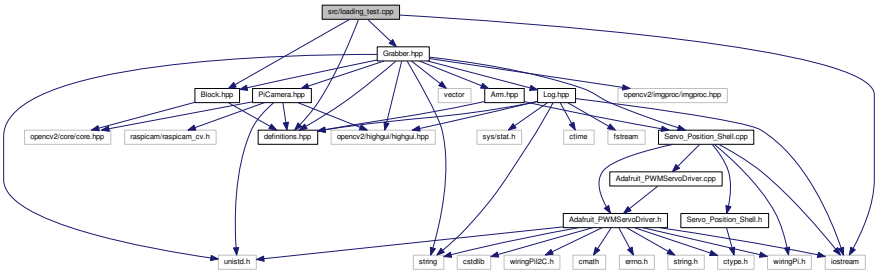
Definition at line 20 of file jacob_alg_test.cpp.

Here is the call graph for this function:

6.13 src/loading_test.cpp File Reference

```
#include <iostream>
#include "definitions.hpp"
#include "Block.hpp"
#include "Grabber.hpp"
```

Include dependency graph for loading_test.cpp:



Functions

- int main ()

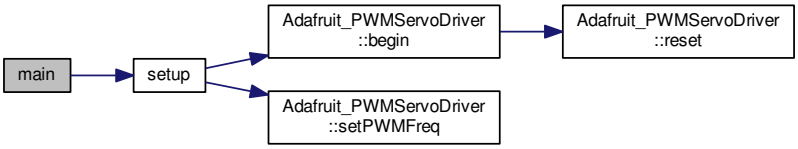
6.13.1 Function Documentation

6.13.1.1 int main ( )

This program was used solely to test the Grabber class. It moves the arm and picks up blocks.

Definition at line 18 of file loading_test.cpp.

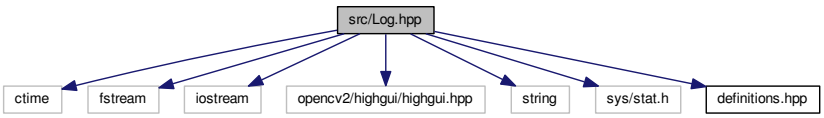
Here is the call graph for this function:



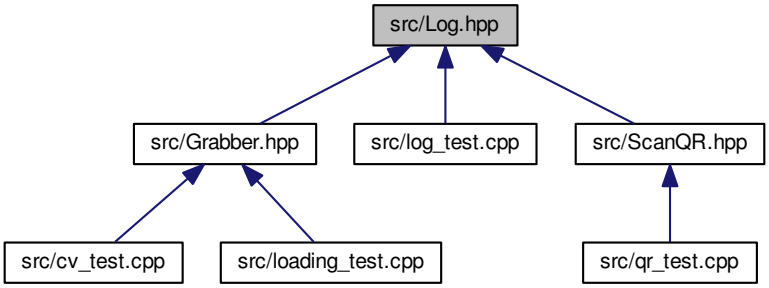
6.14 src/Log.hpp File Reference

```
#include <ctime>
#include <fstream>
#include <iostream>
#include <opencv2/highgui/highgui.hpp>
#include <string>
#include <sys/stat.h>
#include "definitions.hpp"
```

Include dependency graph for Log.hpp:



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



Classes

- class ChipChipArray::Log

Namespaces

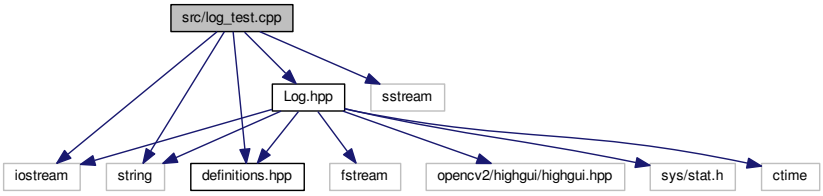
- ChipChipArray

  contains Block class

6.15 src/log_test.cpp File Reference

```
#include <iostream>
#include <sstream>
#include <string>
#include "definitions.hpp"
#include "Log.hpp"
```

Include dependency graph for log_test.cpp:



Functions

- int main ()

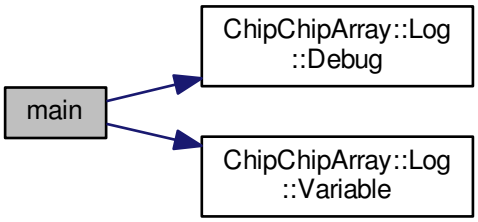
6.15.1 Function Documentation

6.15.1.1 int main ( )

This program partially tests the Log class.

Definition at line 17 of file log_test.cpp.

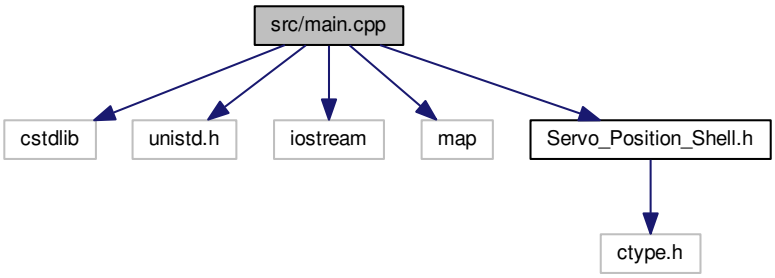
Here is the call graph for this function:



6.16 src/main.cpp File Reference

```
#include <cstdlib>
#include <unistd.h>
#include <iostream>
#include <map>
#include "Servo_Position_Shell.h"
```

Include dependency graph for main.cpp:



Functions

- int main ()

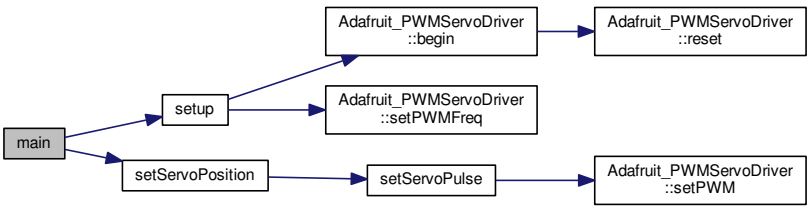
6.16.1 Function Documentation

6.16.1.1 int main ( )

This program moves a specified servo to a specified position. It doubles as a test program for the functional servo interface (i.e., Servo_Position_Shell.cpp).

Definition at line 23 of file main.cpp.

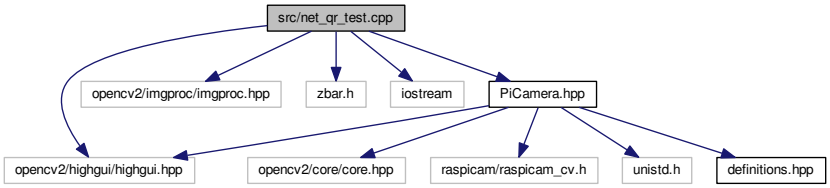
Here is the call graph for this function:



6.17 src/net_qr_test.cpp File Reference

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <zbar.h>
#include <iostream>
#include "PiCamera.hpp"
```

Include dependency graph for net_qr_test.cpp:



Functions

- int main (int argc, char ∗argv[])

6.17.1 Function Documentation

6.17.1.1 int main (int *argc*, char ∗ *argv[]*)

This is a (modified) test program written by Michael Young (https://github.com/ayoungprogrammer/↩WebcamCodeScanner). It was modified to work with the Raspicam.

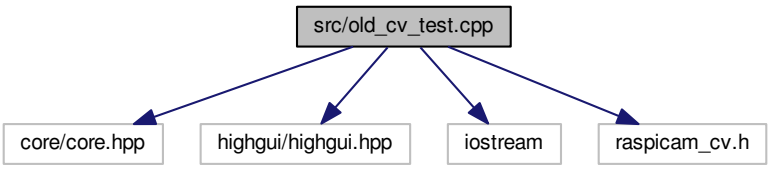
Definition at line 24 of file net_qr_test.cpp.

Here is the call graph for this function:

6.18 src/old_cv_test.cpp File Reference

```
#include <core/core.hpp>
#include <highgui/highgui.hpp>
#include <iostream>
#include "raspicam_cv.h"
```

Include dependency graph for old_cv_test.cpp:



Functions

- int main ()

  contains old test program for the RaspiCam_Cv class

6.18.1 Function Documentation

6.18.1.1 int main ( )

contains old test program for the RaspiCam_Cv class

Author

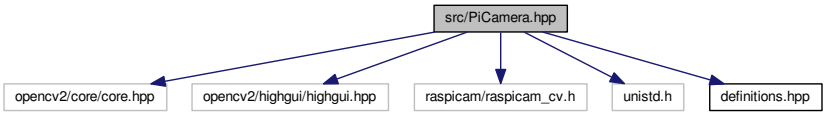
Samuel Andrew Wisner, awisner94@gmail.com This program was used to test the raspicam wrapper for OpenCV before implementing it in a more projet-friendly form as the PiCamera class.

Definition at line 16 of file old_cv_test.cpp.

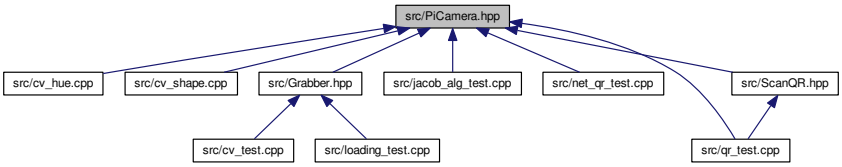
6.19 src/PiCamera.hpp File Reference

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <raspicam/raspicam_cv.h>
#include <unistd.h>
#include "definitions.hpp"
```

Include dependency graph for PiCamera.hpp:



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



Classes

- class ChipChipArray::PiCamera

Namespaces

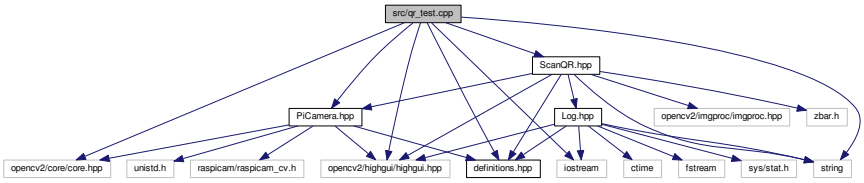
- ChipChipArray

  contains Block class

6.20 src/qr_test.cpp File Reference

```
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <string>
#include "definitions.hpp"
#include "PiCamera.hpp"
#include "ScanQR.hpp"
```

Include dependency graph for qr_test.cpp:



Functions

- int main ()

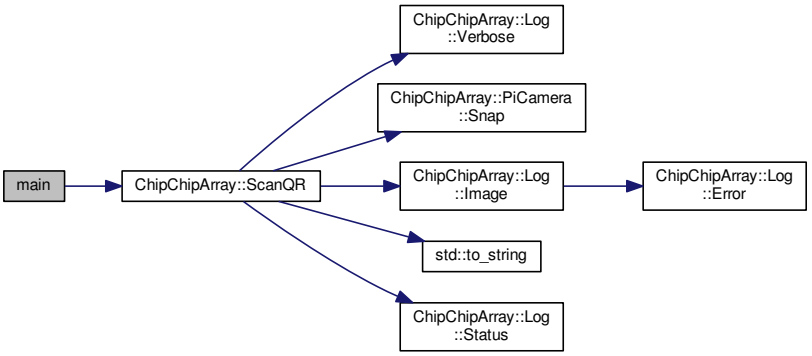
6.20.1 Function Documentation

6.20.1.1 int main ( )

This program tests the ScanQR() function in terms of reading QR codes (not moving the arm).

Definition at line 21 of file qr_test.cpp.

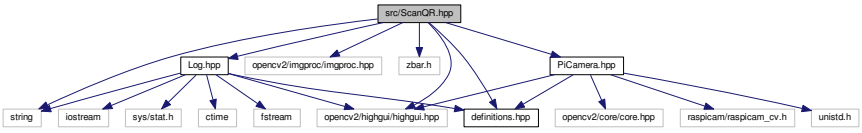
Here is the call graph for this function:



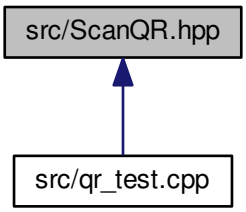
6.21 src/ScanQR.hpp File Reference

```
#include <string>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <zbar.h>
#include "definitions.hpp"
#include "Log.hpp"
#include "PiCamera.hpp"
```

Include dependency graph for ScanQR.hpp:



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



Namespaces

- ChipChipArray

  contains Block class

Functions

- Color ChipChipArray::ScanQR ()

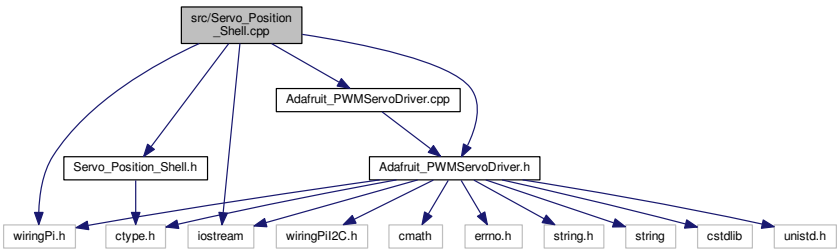
Variables

- uint8 ChipChipArray::qrInvokeCount = 0
- Log ChipChipArray::scanQrLog ("logs/ScanQR", LogMode::Multi)

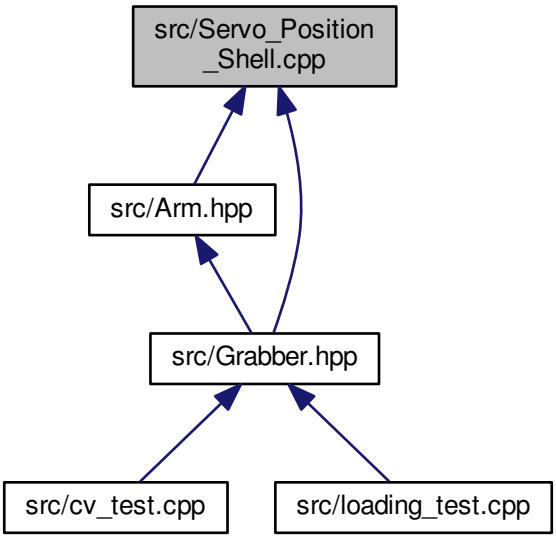
6.22 src/Servo_Position_Shell.cpp File Reference

```
#include <wiringPi.h>
#include "Adafruit_PWMServoDriver.h"
#include "Adafruit_PWMServoDriver.cpp"
#include <iostream>
#include "Servo_Position_Shell.h"
```

Include dependency graph for Servo_Position_Shell.cpp:



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



Macros

- #define SERVOMIN 150
- #define SERVOMAX 600

Functions

- void setup ()
- void setServoPulse (uint8_t servo_num, double pulse)
- void setServoPosition (Servo whichservo, int position)

Variables

- Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver()
- uint8_t servo_num

6.22.1 Macro Definition Documentation

6.22.1.1 #define SERVOMAX 600

Definition at line 25 of file Servo_Position_Shell.cpp.

6.22.1.2 #define SERVOMIN 150

Definition at line 24 of file Servo_Position_Shell.cpp.

6.22.2 Function Documentation

6.22.2.1 void setServoPosition (Servo *whichservo,* int *position*)

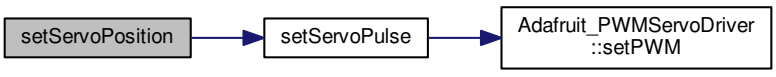
Desc: This function sets which servo to use using whichservo and what position out of 180 degrees for each servo (with limits).

Parameters

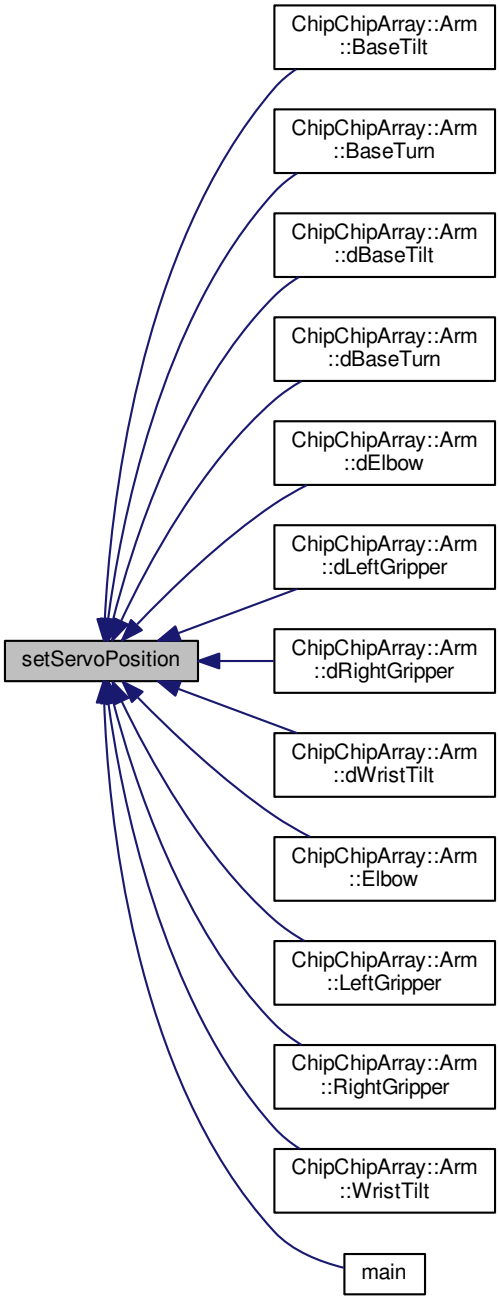
whichservo	which servo would you like to use on the board
position	what position do you want to set the servo selected at

Definition at line 65 of file Servo_Position_Shell.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.22.2.2 void setServoPulse (uint8_t *n*, double *pulse*)

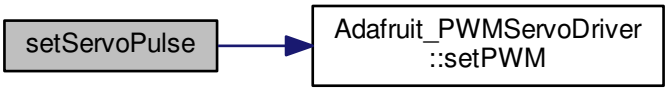
Desc: This function sets which servo to use and what pulse to set that servos pwm to.

Parameters

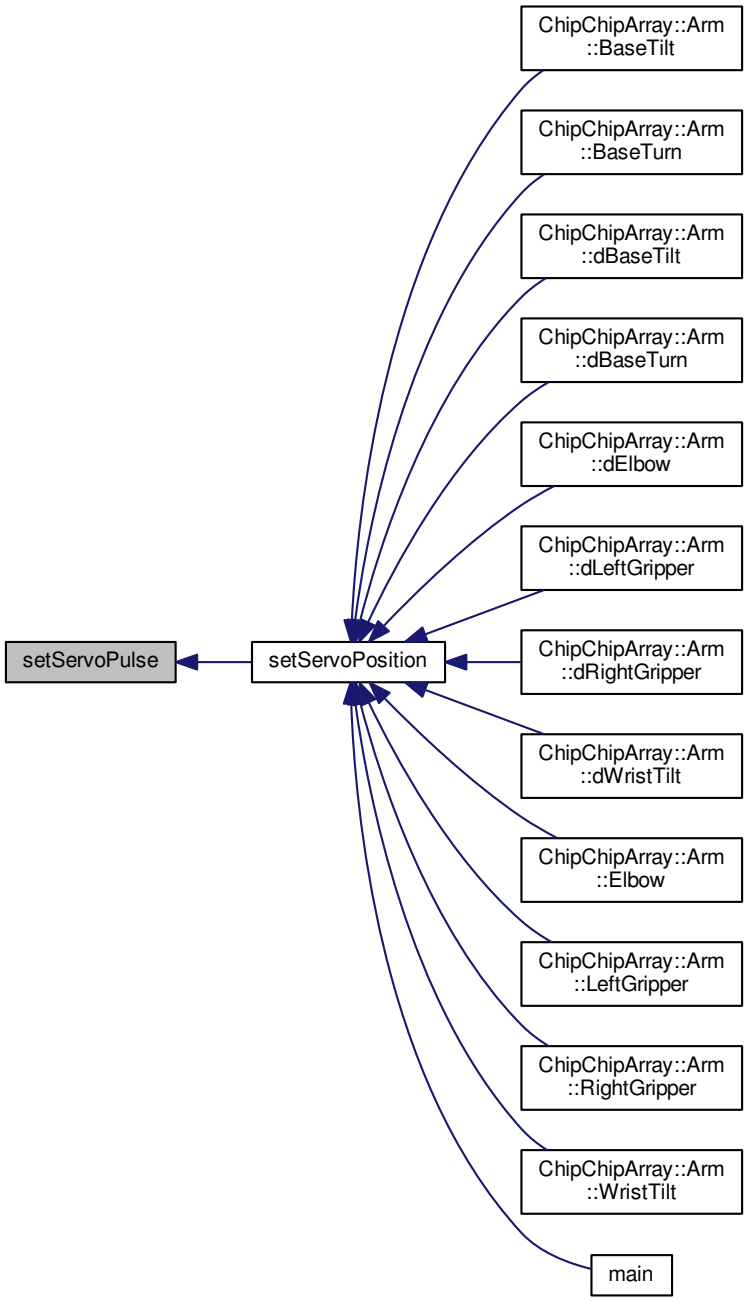
n	which servo on the breakout board am I calling. Starting with 0.
pulse	what is the pulse length (in micro seconds) the pwm of the servo is set to.

Definition at line 44 of file Servo_Position_Shell.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

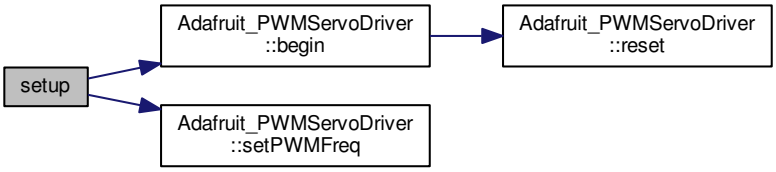


6.22.2.3 void setup ( )

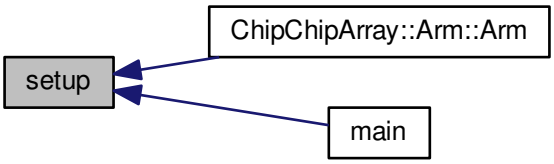
Desc: This function sets up the breakout board communication with I2C using Adafruits_PWMServoDriver.cpp and to set the frequency of the servos to 60Hz.

Definition at line 35 of file Servo_Position_Shell.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.22.3 Variable Documentation

6.22.3.1 Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver()

Definition at line 16 of file Servo_Position_Shell.cpp.

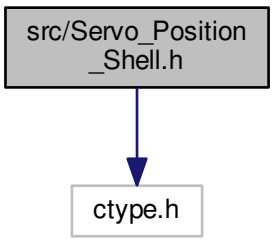
6.22.3.2 uint8_t servo_num

Definition at line 28 of file Servo_Position_Shell.cpp.

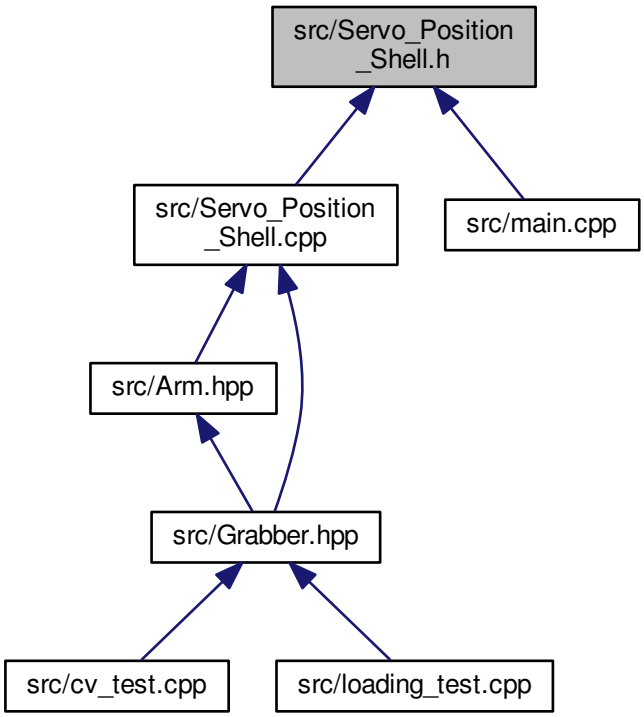
6.23 src/Servo_Position_Shell.h File Reference

```
#include <ctype.h>
```

Include dependency graph for Servo_Position_Shell.h:



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



Macros

- #define uint8_t unsigned char

Created on February 8, 2016, 12:05 PM.

Enumerations

- enum Servo {
  BASE_TURN = 0, BASE_TILT = 1, ELBOW = 2, WRIST_TILT = 3,
  WRIST_PAN = 4, GRIP_RIGHT = 5, GRIP_LEFT = 6, GATE_YELLOW = 7,
  GATE_GREEN = 8, GATE_BLUE = 9, GATE_RED = 10, LIFT_YELLOW = 11,
  LIFT_GREEN = 12, LIFT_BLUE = 13, LIFT_RED = 14 }

Functions

- void setServoPulse (uint8_t n, double pulse)
- void setup ()
- void setServoPosition (Servo whichservo, int position)

6.23.1 Macro Definition Documentation

6.23.1.1 #define uint8_t unsigned char

Created on February 8, 2016, 12:05 PM.

File: Servo_Position_Shell.h

Author

Nickolas Neely

Definition at line 14 of file Servo_Position_Shell.h.

6.23.2 Enumeration Type Documentation

6.23.2.1 enum Servo

Defines each of the servos on the robot.

Enumerator

BASE_TURN
BASE_TILT
ELBOW
WRIST_TILT
WRIST_PAN
GRIP_RIGHT
GRIP_LEFT
GATE_YELLOW
GATE_GREEN
GATE_BLUE
GATE_RED
LIFT_YELLOW
LIFT_GREEN
LIFT_BLUE
LIFT_RED

Definition at line 21 of file Servo_Position_Shell.h.

6.23.3 Function Documentation

6.23.3.1 void setServoPosition (Servo *whichservo,* int *position*)

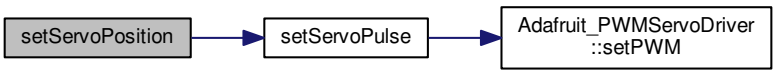
Desc: This function sets which servo to use using whichservo and what position out of 180 degrees for each servo (with limits).

Parameters

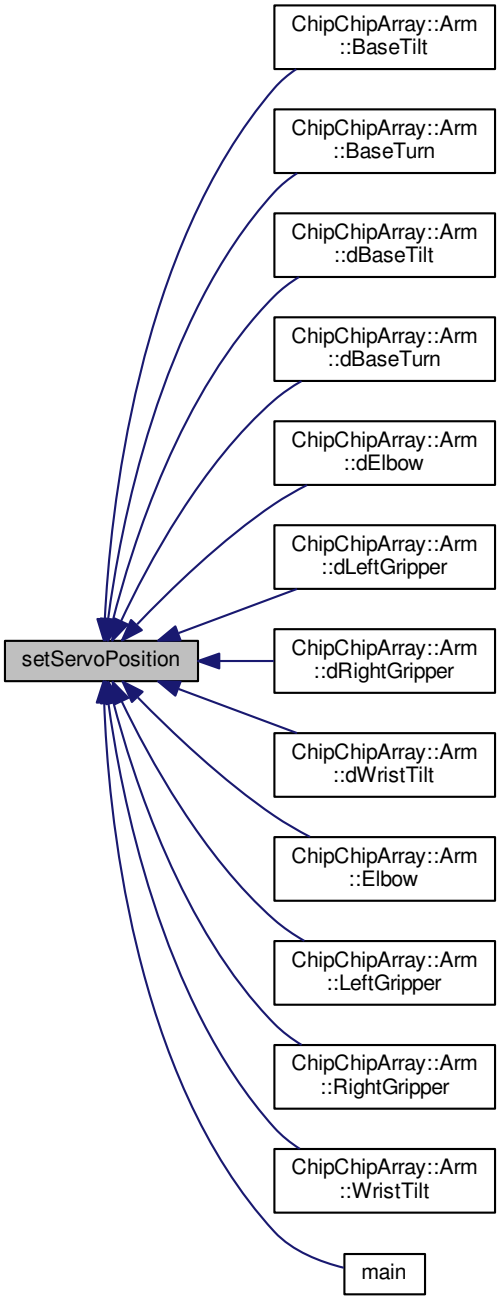
whichservo	which servo would you like to use on the board
position	what position do you want to set the servo selected at

Definition at line 65 of file Servo_Position_Shell.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.23.3.2 void setServoPulse (uint8_t *n*, double *pulse*)

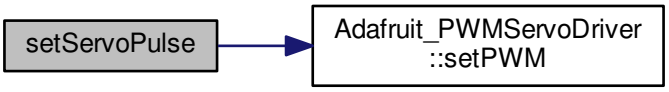
Desc: This function sets which servo to use and what pulse to set that servos pwm to.

Parameters

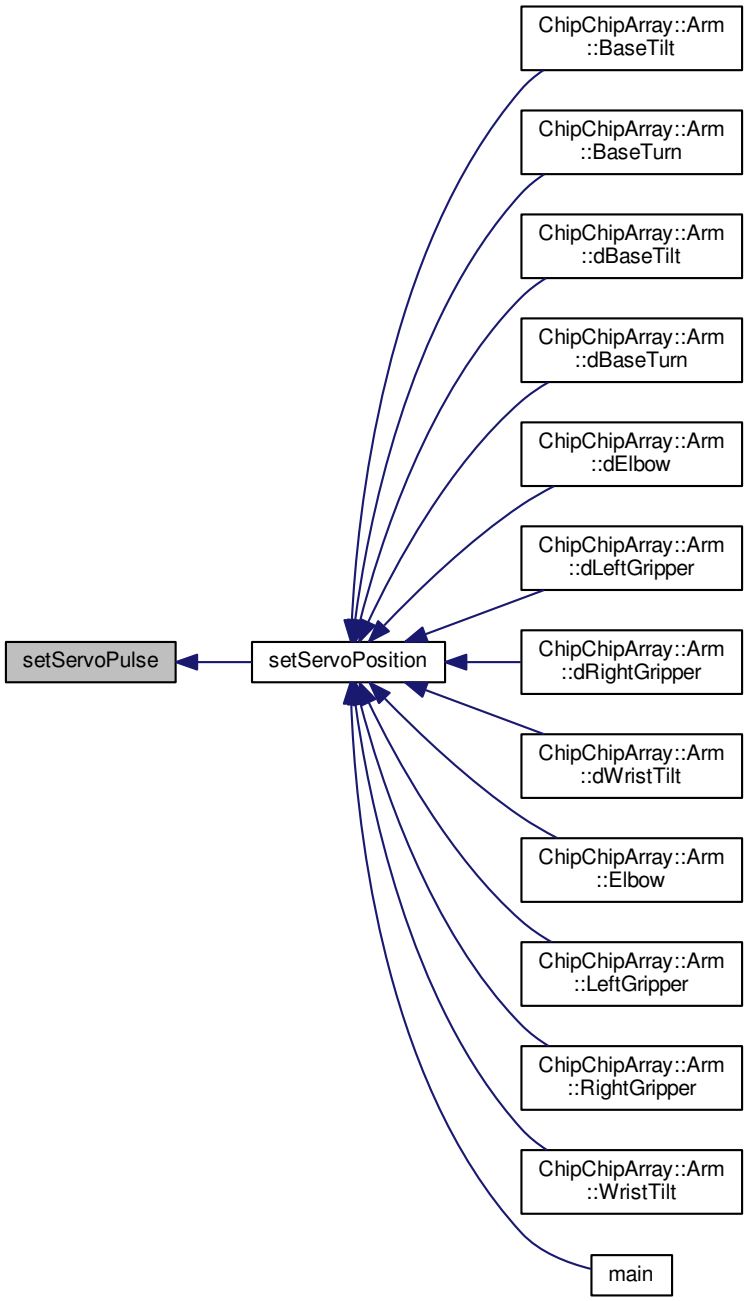
n	which servo on the breakout board am I calling. Starting with 0.
pulse	what is the pulse length (in micro seconds) the pwm of the servo is set to.

Definition at line 44 of file Servo_Position_Shell.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

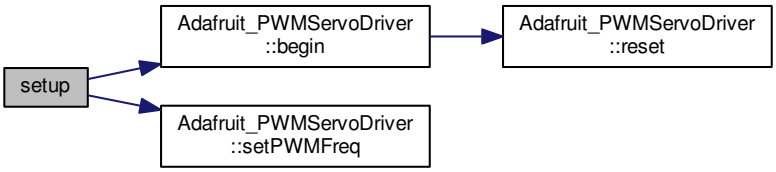


6.23.3.3 void setup ( )

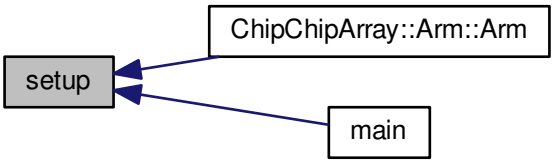
Desc: This function sets up the breakout board communication with I2C using Adafruits_PWMServoDriver.cpp and to set the frequency of the servos to 60Hz.

Definition at line 35 of file Servo_Position_Shell.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



Index

A
definitions.hpp, 44

B
definitions.hpp, 44
BASE_TILT
Servo_Position_Shell.h, 62
BASE_TURN
Servo_Position_Shell.h, 62
Back
definitions.hpp, 42
Blue
definitions.hpp, 43

C
definitions.hpp, 44

definitions.hpp
A, 44
B, 44
Back, 42
Blue, 43
C, 44
Four_Halves, 43
Front, 42
Green, 43
Left, 43
Long, 43
Multi, 43
No_Blocks, 43
No_Halves, 43
Perrywinkle, 43
Red, 43
Right, 43
Short, 43
Text, 43
Two_Halves, 43
Yellow, 43

ELBOW
Servo_Position_Shell.h, 62

Four_Halves
definitions.hpp, 43
Front
definitions.hpp, 42

GATE_BLUE
Servo_Position_Shell.h, 62
GATE_GREEN
Servo_Position_Shell.h, 62
GATE_RED
Servo_Position_Shell.h, 62
GATE_YELLOW
Servo_Position_Shell.h, 62
GRIP_LEFT
Servo_Position_Shell.h, 62
GRIP_RIGHT
Servo_Position_Shell.h, 62
Green
definitions.hpp, 43

LIFT_BLUE
Servo_Position_Shell.h, 62
LIFT_GREEN
Servo_Position_Shell.h, 62
LIFT_RED
Servo_Position_Shell.h, 62
LIFT_YELLOW
Servo_Position_Shell.h, 62
Left
definitions.hpp, 43
Long
definitions.hpp, 43

makefile, 31
Multi
definitions.hpp, 43

No_Blocks
definitions.hpp, 43
No_Halves
definitions.hpp, 43

Perrywinkle
definitions.hpp, 43

Red
definitions.hpp, 43
Right
definitions.hpp, 43

Servo_Position_Shell.h
BASE_TILT, 62
BASE_TURN, 62
ELBOW, 62
GATE_BLUE, 62
GATE_GREEN, 62
GATE_RED, 62
GATE_YELLOW, 62
GRIP_LEFT, 62
GRIP_RIGHT, 62

LIFT_BLUE, 62
LIFT_GREEN, 62
LIFT_RED, 62
LIFT_YELLOW, 62
WRIST_PAN, 62
WRIST_TILT, 62
Short
definitions.hpp, 43
std, 9

Text
definitions.hpp, 43
Two_Halves
definitions.hpp, 43

WRIST_PAN
Servo_Position_Shell.h, 62
WRIST_TILT
Servo_Position_Shell.h, 62

Yellow
definitions.hpp, 43