ChipChipArray

Generated by Doxygen 1.8.8

Fri Apr 22 2016 15:52:01

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 ChipChipArray Namespace Reference

Classes

- class Arm
- class Block
- class Grabber
- 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 Function Documentation

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

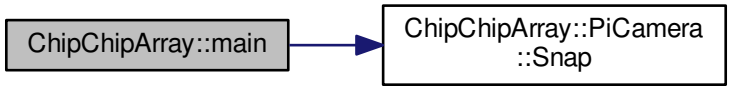
This program was used before cv_shape.cpp was written to find HSV ranges for the different color blocks. This is a slightly modified version of some code written by Shermal Fernando in the blog post "Color Detection & Object Tracking" at http://opencv-srf.blogspot.com/2010/09/object-detection-using-color-seperation.html".

Definition at line 27 of file cv_hue.cpp.

```
00027                                    {
00028         PiCamera cam;
00029         namedWindow("Control", CV_WINDOW_AUTOSIZE); //create a window called "Control"
00030
00031         int iLowH = 170;
00032         int iHighH = 179;
00033
00034         int iLowS = 150;
00035         int iHighS = 255;
00036
00037         int iLowV = 60;
```

```
00038         int iHighV = 255;
00039
00040         //Create trackbars in "Control" window
00041         createTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)
00042         createTrackbar("HighH", "Control", &iHighH, 179);
00043
00044         createTrackbar("LowS", "Control", &iLowS, 255); //Saturation (0 - 255)
00045         createTrackbar("HighS", "Control", &iHighS, 255);
00046
00047         createTrackbar("LowV", "Control", &iLowV, 255);//Value (0 - 255)
00048         createTrackbar("HighV", "Control", &iHighV, 255);
00049
00050         int iLastX = -1;
00051         int iLastY = -1;
00052
00053         //Capture a temporary image from the camera
00054         Mat imgTmp = cam.Snap();
00055
00056         //Create a black image with the size as the camera output
00057         Mat imgLines = Mat::zeros( imgTmp.size(), CV_8UC3 );;
00058
00059
00060         while (true)
00061         {
00062             Mat imgOriginal = cam.Snap(); // read a new frame from video
00063             Mat imgHSV;
00064
00065             cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV); //Convert the captured frame from BGR to HSV
00066
00067             Mat imgThresholded;
00068
00069             inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV), imgThresholded);
      //Threshold the image
00070
00071             //morphological opening (removes small objects from the foreground)
00072             erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
      Size(5, 5)) );
00073             dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
      Size(5, 5)) );
00074
00075             //morphological closing (removes small holes from the foreground)
00076             dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
      Size(5, 5)) );
00077             erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
      Size(5, 5)) );
00078
00079             //Calculate the moments of the thresholded image
00080             Moments oMoments = moments(imgThresholded);
00081
00082             /*double dM01 = oMoments.m01;
00083               double dM10 = oMoments.m10;
00084               double dArea = oMoments.m00;
00085
00086             // if the area <= 10000, I consider that the there are no object in the image and it's because
       of the noise, the area is not zero
00087             if (dArea > 50000)
00088             {
00089             //calculate the position of the ball
00090             int posX = dM10 / dArea;
00091             int posY = dM01 / dArea;
00092
00093             if (iLastX >= 0 && iLastY >= 0 && posX >= 0 && posY >= 0)
00094             {
00095             //Draw a red line from the previous point to the current point
00096             line(imgLines, Point(posX, posY), Point(iLastX, iLastY), Scalar(0,0,255), 2);
00097             }
00098
00099             iLastX = posX;
00100             iLastY = posY;
00101             }
00102             */
00103             imshow("Thresholded Image", imgThresholded); //show the thresholded image
00104
00105             //      imgOriginal = imgOriginal + imgLines;
00106             imshow("Original", imgOriginal); //show the original image
00107
00108             if (waitKey(30) == 27) //wait for 'esc' key press for 30ms. If 'esc' key is pressed, break loop
00109             {
00110                 cout << "esc key is pressed by user" << endl;
00111                 break;
00112             }
00113         }
00114
00115         return 0;
00116     }
```

Here is the call graph for this function:



4.1.1.2 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.

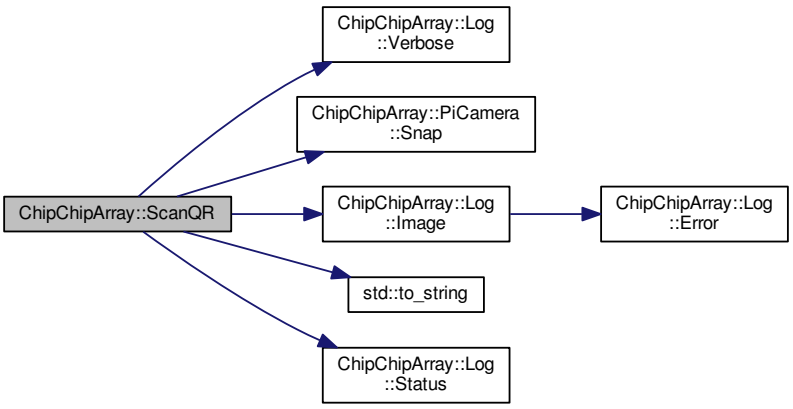
This function is based on code written by Michael Young (https://github.com/ayoungprogrammer/WebcamCodeScanner).

Definition at line 41 of file ScanQR.hpp.

```
00041                   {
00042         // 0. Initialize variables
00043         Color color;
00044         PiCamera cam(false);
00045 
00046         // 1. Position arm
00047         scanQrLog.Verbose("Positioning arm");
00048 
00049         // 2. Scan images from camera
00050         scanQrLog.Verbose("Scanning for QR code");
00051 
00052         // Nick's supposed to make sure this isn't an endles loop
00053         while(true) {
00054             // get image
00055             cv::Mat frame = cam.Snap();
00056             cv::Mat canvas;
00057             cv::cvtColor(frame, canvas, CV_BGR2GRAY);
00058             scanQrLog.Image(canvas, std::to_string(++
      qrInvokeCount)
00059                     + ".bmp");
00060 
00061             uint32 width = canvas.cols;
00062             uint32 height = canvas.rows;
00063             zbar::Image image(width, height, "Y800",
00064                     (uchar*)canvas.data, width * height);
00065 
00066             zbar::ImageScanner scanner;
00067             scanner.set_config(zbar::ZBAR_NONE, zbar::ZBAR_CFG_ENABLE, 1);
00068             scanner.scan(image);
00069             zbar::Image::SymbolIterator symbol = image.symbol_begin();
00070 
00071             if(symbol != image.symbol_end()) {
00072                 switch(symbol->get_data()[0]) {
00073                     case 'r':
00074                         color = Color::Red;
00075                         break;
00076 
00077                     case 'y':
00078                         color = Color::Yellow;
00079                         break;
00080 
00081                     case 'g':
00082                         color = Color::Green;
00083                         break;
00084 
00085                     case 'b':
00086                         color = Color::Blue;
00087                         break;
00088                 }
00089 
```

```
00090                 scanQrLog.Status("Detected " + std::to_string(color)
00091                         + " train car");
00092                 break;
00093             }
00094         }
00095
00096         // 3. Retract arm
00097         scanQrLog.Verbose("Retracting arm");
00098         return color;
00099     }
```

Here is the call graph for this function:



Here is the caller graph for this function:

4.1.2 Variable Documentation

4.1.2.1 uint8 ChipChipArray::qrInvokeCount = 0

The number of times ScanQR() has been called. Used for ScanQR log.

Definition at line 24 of file ScanQR.hpp.

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

The Log instance used by the ScanQR() function.

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

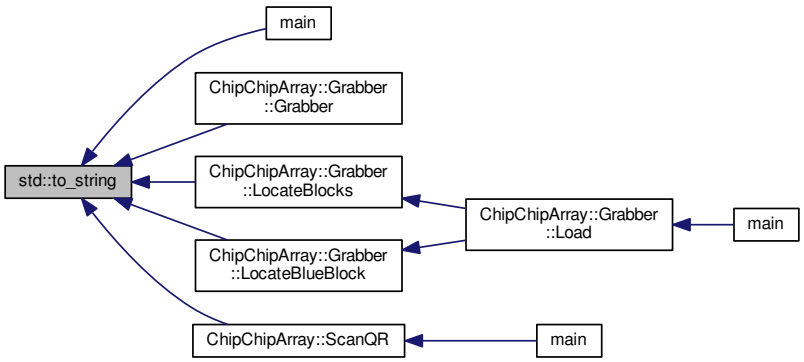
4.2.1.1 string std::to_string (BlockPosition *pos*)

Converts a BlockPosition to a string.

Definition at line 107 of file definitions.hpp.

```
00107                                       {
00108         if(pos == BlockPosition::Front) return string("Front");
00109         else return string("Back");
00110     }
```

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 115 of file definitions.hpp.

```
00115                                     {
00116         string ret;
00117 
00118         switch((ENUM)color) {
00119             case 0:
00120                 ret = "Red";
00121                 break;
00122 
00123             case 1:
00124                 ret = "Yellow";
00125                 break;
00126 
00127             case 2:
```

```
00128                 ret = "Green";
00129                 break;
00130
00131             case 3:
00132                 ret = "Blue";
00133                 break;
00134
00135             case 4:
00136                 ret = "All";
00137                 break;
00138         }
00139
00140         return ret;
00141     }
```

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

Converts a LogMode to a string.

Definition at line 146 of file definitions.hpp.

```
00146                                   {
00147         if(mode == LogMode::Multi) return string("Text");
00148         else return string("Multi");
00149     }
```

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

Converts a Result to a string.

Definition at line 154 of file definitions.hpp.

```
00154                                   {
00155         string ret;
00156
00157         switch((ENUM)res) {
00158             case -1:
00159                 ret = "No Blocks";
00160                 break;
00161
00162             case 0:
00163                 ret = "Two whole, no halves";
00164                 break;
00165
00166             case 2:
00167                 ret = "Two whole, two halves";
00168                 break;
00169
00170             case 4:
00171                 ret = "No whole, four halves";
00172                 break;
00173         }
00174
00175         return ret;
00176     }
```

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

Converts a Side to a string.

Definition at line 181 of file definitions.hpp.

```
00181                                   {
00182         if(side == Side::Left) return string("Left");
00183         else return string("Right");
00184     }
```

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

Converts a Size to a string.

Definition at line 189 of file definitions.hpp.

```
00189                                   {
00190         if(size == Size::Long) return string("Long");
00191         else return string("Short");
00192     }
```

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

Converts a Zone to a string.

Definition at line 197 of file definitions.hpp.

```
00197                                   {
00198         return string(1, (char)zone);
00199     }
```

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 73 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 37 of file Adafruit_PWMServoDriver.cpp.

```
00037                                                          {
00038     _i2caddr = addr;
00039     _i2cFD = -1;
00040 }
```

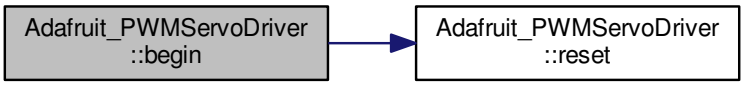
5.1.3 Member Function Documentation

5.1.3.1 void Adafruit_PWMServoDriver::begin (void)

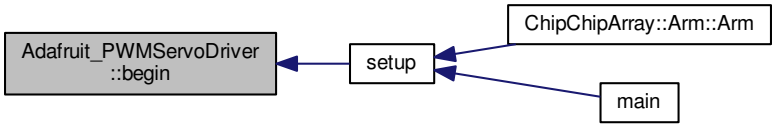
Definition at line 42 of file Adafruit_PWMServoDriver.cpp.

```
00042                                       {
00043     _i2cFD = wiringPiI2CSetup(_i2caddr);
00044     if (_i2cFD < 0) {
00045         //FIXME: error occurred
00046     }
00047     reset();
00048 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

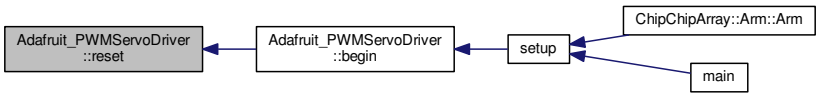


5.1.3.2 void Adafruit_PWMServoDriver::reset (void)

Definition at line 50 of file Adafruit_PWMServoDriver.cpp.

```
00050                                       {
00051     write8(PCA9685_MODE1, 0x0);
00052 }
```

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 116 of file Adafruit_PWMServoDriver.cpp.

```
00116                                                                       {
00117     // Clamp value between 0 and 4095 inclusive.
00118     val = min(val, (uint16_t)4095);
00119     if (invert) {
00120         if (val == 0) {
00121             // Special value for signal fully on.
00122             setPWM(num, 4096, 0);
00123         } else if (val == 4095) {
00124             // Special value for signal fully off.
00125             setPWM(num, 0, 4096);
```

```
00126         } else {
00127             setPWM(num, 0, 4095 - val);
00128         }
00129     } else {
00130         if (val == 4095) {
00131             // Special value for signal fully on.
00132             setPWM(num, 4096, 0);
00133         } else if (val == 0) {
00134             // Special value for signal fully off.
00135             setPWM(num, 0, 4096);
00136         } else {
00137             setPWM(num, 0, val);
00138         }
00139     }
00140 }
```

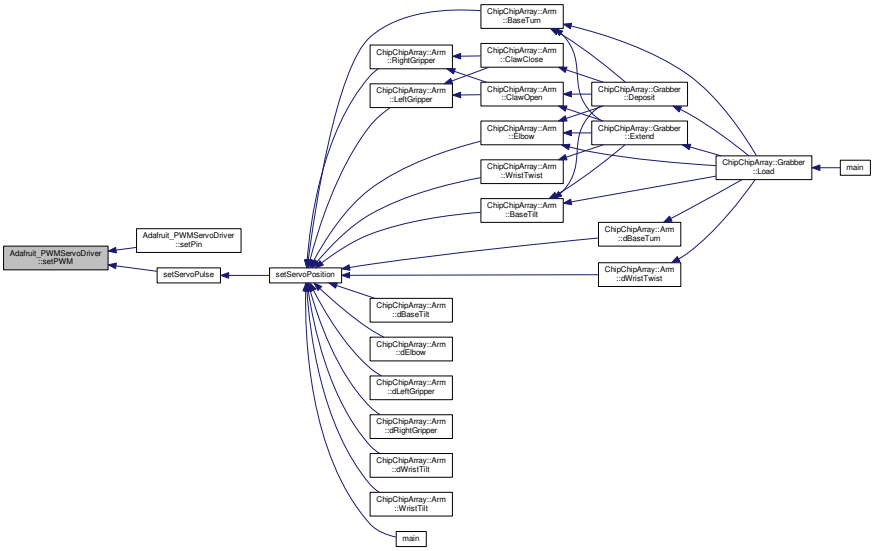
Here is the call graph for this function:

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

Definition at line 81 of file Adafruit_PWMServoDriver.cpp.

```
00081                                                                   {
00082     //Serial.print("Setting PWM "); Serial.print(num); Serial.print(": "); Serial.print(on);
      Serial.print("->"); Serial.println(off);
00083
00084     int result = wiringPiI2CWriteReg16(_i2cFD, LED0_ON_L + 4 * num, on);
00085     if (result < 0) {
00086         string s(strerror(errno));
00087         cout << "setPWM error: " << s.c_str() << endl;
00088     }
00089 //    result = wiringPiI2CWrite(_i2cFD, on);
00090 //    if (result < 0) {
00091 //        string s(strerror(errno));
00092 //        cout << "setPWM error: " << s.c_str() << endl;
00093 //    }
00094 //    result = wiringPiI2CWrite(_i2cFD, on >> 8);
00095 //    if (result < 0) {
00096 //        string s(strerror(errno));
00097 //        cout << "setPWM error: " << s.c_str() << endl;
00098 //    }
00099     result = wiringPiI2CWriteReg16(_i2cFD, LED0_OFF_L + 4 * num, off);
00100 //    result = wiringPiI2CWrite(_i2cFD, off);
00101     if (result < 0) {
00102         string s(strerror(errno));
00103         cout << "setPWM error: " << s.c_str() << endl;
00104     }
00105 //    result = wiringPiI2CWrite(_i2cFD, off >> 8);
00106 //    if (result < 0) {
00107 //        string s(strerror(errno));
00108 //        cout << "setPWM error: " << s.c_str() << endl;
00109 //    }
00110 }
```

Here is the caller graph for this function:

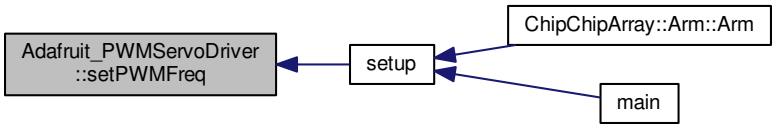


5.1.3.5 void Adafruit_PWMServoDriver::setPWMFreq (float *freq*)

Definition at line 54 of file Adafruit_PWMServoDriver.cpp.

```
00054                                             {
00055     //Serial.print("Attempting to set freq ");
00056     //Serial.println(freq);
00057     freq *= 0.9; // Correct for overshoot in the frequency setting (see issue #11).
00058     float prescaleval = 25000000;
00059     prescaleval /= 4096;
00060     prescaleval /= freq;
00061     prescaleval -= 1;
00062     if (ENABLE_DEBUG_OUTPUT) {
00063         cout << "Estimated pre-scale: " << prescaleval << endl;
00064     }
00065     uint8_t prescale = floor(prescaleval + 0.5);
00066     if (ENABLE_DEBUG_OUTPUT) {
00067         cout << "Final pre-scale: " << prescale << endl;
00068     }
00069
00070     uint8_t oldmode = read8(PCA9685_MODE1);
00071     uint8_t newmode = (oldmode & 0x7F) | 0x10; // sleep
00072     write8(PCA9685_MODE1, newmode); // go to sleep
00073     write8(PCA9685_PRESCALE, prescale); // set the prescaler
00074     write8(PCA9685_MODE1, oldmode);
00075     usleep(5000);
00076     write8(PCA9685_MODE1, oldmode | 0xa1); //  This sets the MODE1 register to turn on auto
    increment.
00077     // This is why the beginTransmission below was not working.
00078     //  Serial.print("Mode now 0x"); Serial.println(read8(PCA9685_MODE1), HEX);
00079 }
```

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 ClawOpen ()
- void ClawClose ()
- void dBaseTilt (sint16 a)
- void dBaseTurn (sint16 a)
- void dElbow (sint16 a)
- void dWristTilt (sint16 a)
- void dWristTwist (sint16 a)
- void Elbow (uint8 a)
- void Hover (Zone zone)
- void WristTilt (uint8 a)
- void WristTwist (uint8 a)

Public Attributes

- uint8 servoPos [7] = { 0, 0, 0, 0, 0, 0, 0 }

Protected Member Functions

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

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 23 of file Arm.hpp.

5.2.2 Constructor & Destructor Documentation

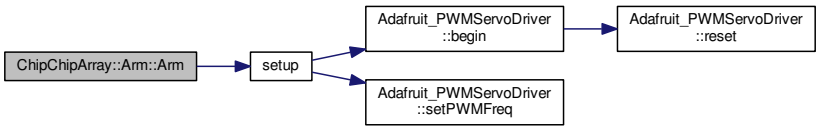
5.2.2.1 ChipChipArray::Arm::Arm ( )

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

Definition at line 173 of file Arm.hpp.

```
00173                {
00174          if(!init) {
00175              setup();
00176              init = true;
00177          }
00178      }
```

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.

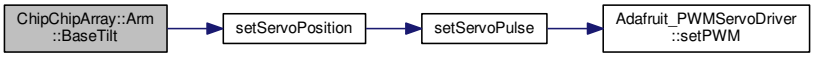
Parameters

a	desired servo position in degrees

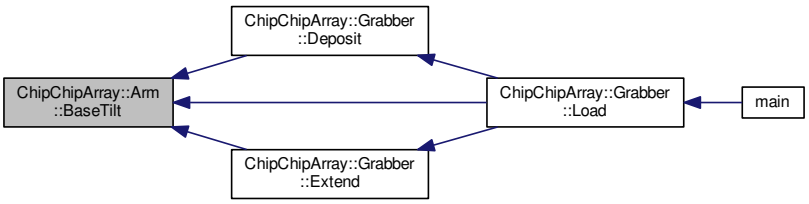
Definition at line 180 of file Arm.hpp.

```
00180                               {
00181          setServoPosition(BASE_TILT, a);
00182          servoPos[BASE_TILT] = a;
00183      }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Twists the entire arm.

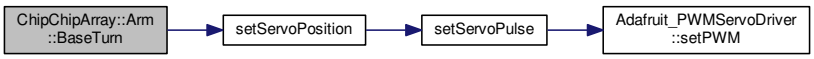
Parameters

a	desired servo position in degrees

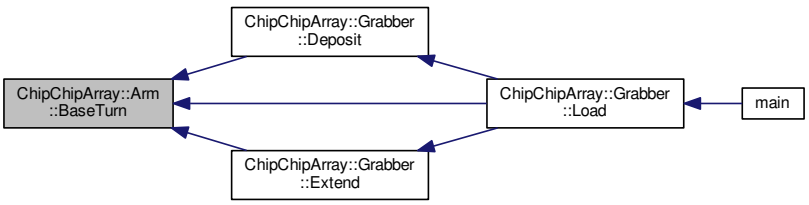
Definition at line 185 of file Arm.hpp.

```
00185                               {
00186         setServoPosition(BASE_TURN, a);
00187         servoPos[BASE_TURN] = a;
00188     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.3 void ChipChipArray::Arm::ClawClose ( )

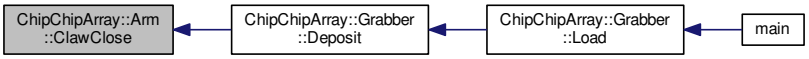
Closes the claw enough to hold a block in place during movement but does not attempt to completely close the claw in order ot prevent unnecessary strain on the servos.

Definition at line 195 of file Arm.hpp.

```
00195                         {
00196         LeftGripper(180);
00197         RightGripper(0);
00198     }
```

Here is the call graph for this function:

Here is the caller graph for this function:



5.2.3.4 void ChipChipArray::Arm::ClawOpen ( )

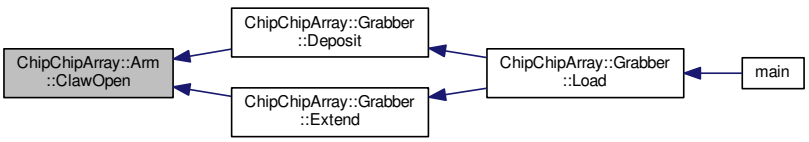
Opens the claw completely (within safe limits).

Definition at line 190 of file Arm.hpp.

```
00190                         {
00191         LeftGripper(0);
00192         RightGripper(180);
00193     }
```

Here is the call graph for this function:

Here is the caller graph for this function:



5.2.3.5 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 200 of file Arm.hpp.

```
00200                                {
00201         a += servoPos[BASE_TILT];
00202         setServoPosition(BASE_TILT, a);
00203         servoPos[BASE_TILT] = a;
00204     }
```

Here is the call graph for this function:

5.2.3.6 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 206 of file Arm.hpp.

```
00206                                {
00207         a += servoPos[BASE_TURN];
00208         setServoPosition(BASE_TURN, a);
00209         servoPos[BASE_TURN] = a;
00210     }
```

Here is the call graph for this function:

Here is the caller graph for this function:

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

Bend the elbow a certain number of degrees.

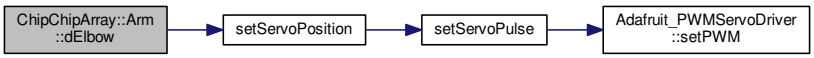
Parameters

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

Definition at line 212 of file Arm.hpp.

```
00212                           {
00213         a += servoPos[ELBOW];
00214         setServoPosition(ELBOW, a);
00215         servoPos[ELBOW] = a;
00216     }
```

Here is the call graph for this function:



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

Moves the left gripper servo a certain number of degrees.

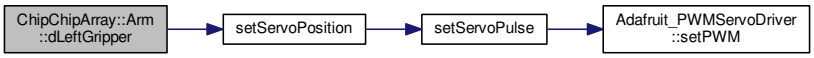
Parameters

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

Definition at line 218 of file Arm.hpp.

```
00218                                 {
00219         a += servoPos[GRIP_LEFT];
00220         setServoPosition(GRIP_LEFT, a);
00221         servoPos[GRIP_LEFT] = a;
00222     }
```

Here is the call graph for this function:



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

Moves the right gripper servo a certain number of degrees.

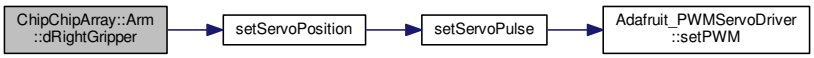
Parameters

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

Definition at line 224 of file Arm.hpp.

```
00224                                {
00225         a += servoPos[GRIP_RIGHT];
00226         setServoPosition(GRIP_RIGHT, a);
00227         servoPos[GRIP_RIGHT] = a;
00228     }
```

Here is the call graph for this function:



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

Tilt the wrist a certain number of degrees.

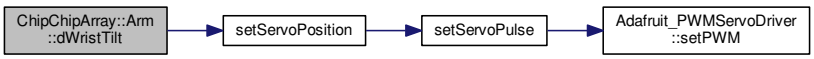
Parameters

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

Definition at line 230 of file Arm.hpp.

```
00230                               {
00231         a += servoPos[WRIST_TILT];
00232         setServoPosition(WRIST_TILT, a);
00233         servoPos[WRIST_TILT] = a;
00234     }
```

Here is the call graph for this function:



5.2.3.11 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.

Definition at line 236 of file Arm.hpp.

```
00236                                {
00237         a += servoPos[WRIST_TWIST];
00238         setServoPosition(WRIST_TWIST, a);
00239         servoPos[WRIST_TWIST] = a;
00240     }
```

Here is the call graph for this function:

Here is the caller graph for this function:

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

Bend the elbow to a specific position.

Parameters

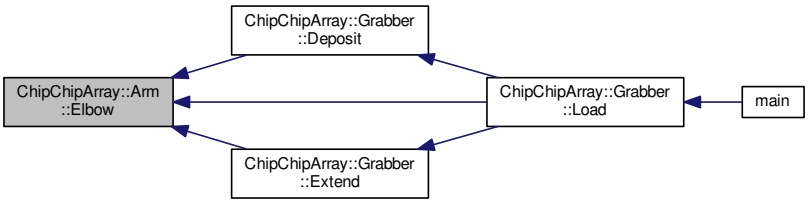
a	desired servo position in degrees

Definition at line 242 of file Arm.hpp.

```
00242                           {
00243         setServoPosition(ELBOW, a);
00244         servoPos[ELBOW] = a;
00245     }
```

Here is the call graph for this function:

Here is the caller graph for this function:



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

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

Moves the left gripper to a specific position.

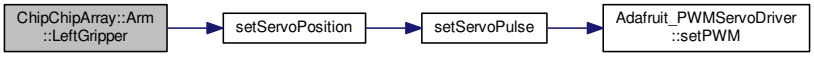
Parameters

a	desired servo position in degrees

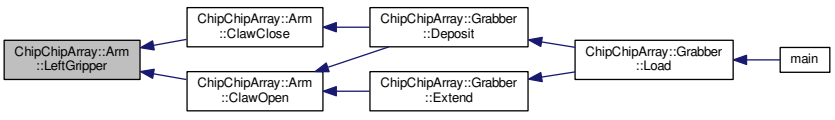
Definition at line 247 of file Arm.hpp.

```
00247                                {
00248         setServoPosition(GRIP_LEFT, a);
00249         servoPos[GRIP_LEFT] = a;
00250     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Moves the right gripper to a specific position.

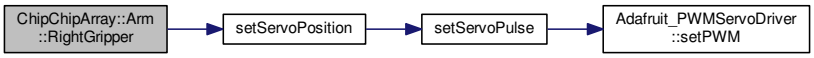
Parameters

a	desired servo position in degrees

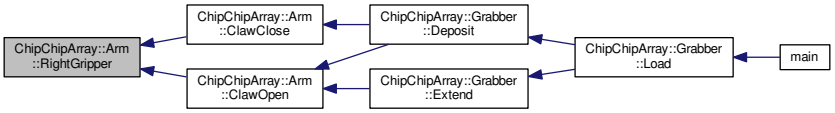
Definition at line 252 of file Arm.hpp.

```
00252                                {
00253         setServoPosition(GRIP_RIGHT, a);
00254         servoPos[GRIP_RIGHT] = a;
00255     }
```

Here is the call graph for this function:



Here is the caller 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 257 of file Arm.hpp.

```
00257                                {
00258         setServoPosition(WRIST_TILT, a);
00259         servoPos[WRIST_TILT] = a;
00260     }
```

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.

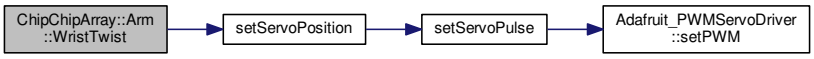
Parameters

a	desired servo position in degrees

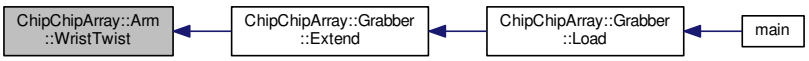
Definition at line 262 of file Arm.hpp.

```
00262                                {
00263         setServoPosition(WRIST_TWIST, a);
00264         servoPos[WRIST_TWIST] = a;
00265     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.4 Member Data Documentation

5.2.4.1 uint8 ChipChipArray::Arm::servoPos[7] = { 0, 0, 0, 0, 0, 0, 0 }

The instantaneous position of each arm servo.

Definition at line 34 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 20 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 139 of file Block.hpp.

```
00139                                       {
00140         // basic geometric properties
00141         area = rect.area();
00142         height = rect.height;
00143         width = rect.width;
00144 
00145         // assigning corners
00146         topLeft = rect.tl();
00147         bottomRight = rect.br();
00148         topRight = cv::Point(topLeft.x + width, topLeft.y);
00149         bottomLeft = cv::Point(topLeft.x, topLeft.y +
    height);
00150         offset = (sint16)(topLeft.x + width / 2) - IMG_WIDTH / 2;
00151 
00152         // calculating offsets (opencv low coordinates start top left)
00153         dLeft = topLeft.x;
00154         dRight = IMG_WIDTH - topRight.x;
00155         dTop = topLeft.y;
00156         dBottom = IMG_HEIGHT - bottomRight.y;
00157         dTopBottom = dTop - dBottom;
00158         dRightLeft = dRight - dLeft;
00159 
00160         // set color and size
00161         this->color = color;
00162         size = area > MIN_WHOLE_BLOCK_SIZE ? Size::Long :
    Size::Short;
00163     }
```

5.3.3 Member Data Documentation

5.3.3.1 uint32 ChipChipArray::Block::area

The area of the block in pixels

Definition at line 25 of file Block.hpp.

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

Point of the block's bottom-left corner

Definition at line 30 of file Block.hpp.

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

Point of the block's bottom-right corner

Definition at line 35 of file Block.hpp.

5.3.3.4 Color ChipChipArray::Block::color

The detected color of the block

Definition at line 107 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 41 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 47 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 53 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 75 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 59 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 67 of file Block.hpp.

5.3.3.11 uint16 ChipChipArray::Block::height

The height of the block in pixels

Definition at line 87 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 82 of file Block.hpp.

5.3.3.13 Size ChipChipArray::Block::size

The size of the block (half or whole)

Definition at line 112 of file Block.hpp.

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

Point of the block's top-left corner

Definition at line 92 of file Block.hpp.

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

Point of the block's top-right corner

Definition at line 97 of file Block.hpp.

5.3.3.16 uint16 ChipChipArray::Block::width

The width of the block in pixels

Definition at line 102 of file Block.hpp.

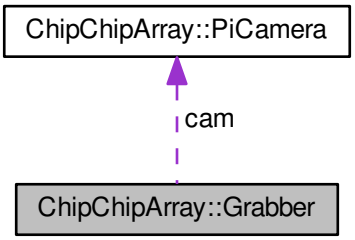
The documentation for this class was generated from the following file:

- src/Block.hpp

5.4 ChipChipArray::Grabber Class Reference

```
#include <Grabber.hpp>
```

Collaboration diagram for ChipChipArray::Grabber:



Public Member Functions

- Grabber (Zone zone, Side side)
- void Close ()
- Result Load ()

Protected Member Functions

- void Deposit (Color color=Color::Blue)
- void Extend ()
- Block LocateBlocks (Color color=Color::Perrywinkle)
- Block LocateBlueBlock ()

Protected Attributes

- PiCamera cam
- Side side
- Zone zone

5.4.1 Detailed Description

This class finds blocks, identifies them, and sorts them according to color, size, and zone.

Definition at line 30 of file Grabber.hpp.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ChipChipArray::Grabber::Grabber (Zone *zone*, Side *side*)

Initializes the class according to the side and zone and extends the robotic arm into position.

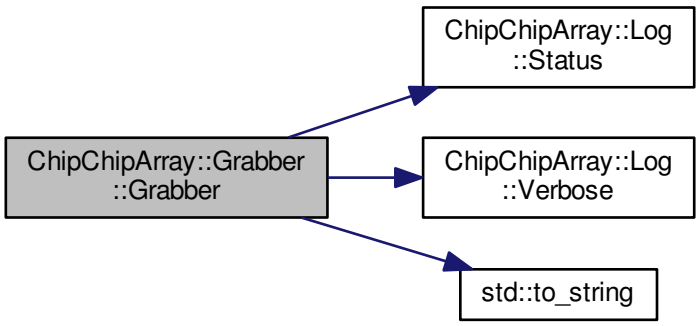
Parameters

zone	the zone (A, B, or C) for which to pick up blocks.
side	the side from which the robot is moving and the position of the blocks (right or left) in the view of the camera to pick up first

Definition at line 143 of file Grabber.hpp.

```
00143                                       {
00144         log.Status("Opening Grabber");
00145         log.Verbose("Zone: " + std::to_string(zone));
00146         log.Verbose("Side: " + std::to_string(side));
00147 
00148         this->zone = zone;
00149         this->side = side;
00150 
00151         log.Verbose("Setting HSV threshold values");
00152 
00153         rangeVals[Color::Red] = { cv::Scalar(0, 20, 60),
00154             cv::Scalar(12, 255, 255) };
00155         rangeVals[Color::Green] = { cv::Scalar(49, 41, 17),
00156             cv::Scalar(63, 255, 255) };
00157         rangeVals[Color::Blue] = { cv::Scalar(70, 0, 0),
00158             cv::Scalar(100, 255, 255) };
00159 
00160         /* Remember, we're only pretending this color's image is in HSV space.
00161          * It's really in YUV, as required by Jacob yellow-detection algorithm. */
00162         rangeVals[Color::Yellow] = { cv::Scalar(0, 0, 0),
00163             cv::Scalar(255, 255, 20)};
00164     }
```

Here is the call graph for this function:



5.4.3 Member Function Documentation

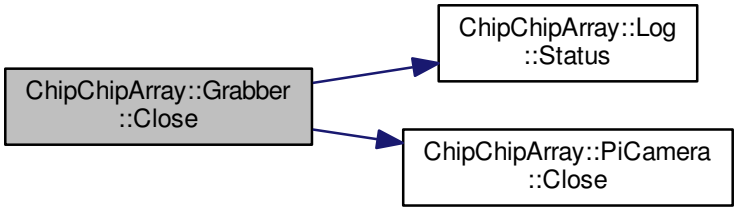
5.4.3.1 void ChipChipArray::Grabber::Close ( )

Closes the Grabber. Retracts the arm and closes the camera.

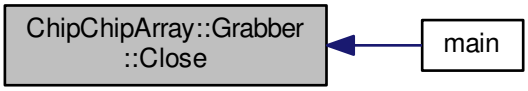
Definition at line 166 of file Grabber.hpp.

```
00166                           {
00167         log.Status("Closing Grabber");
00168         cam.Close();
00169     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



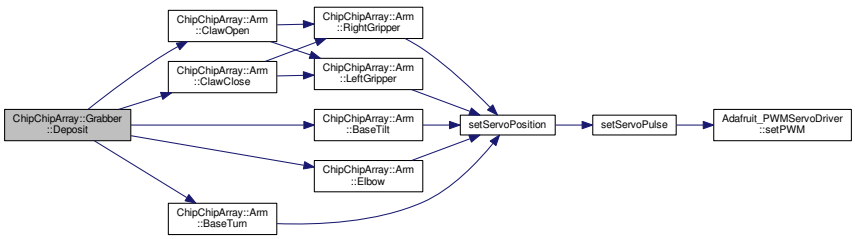
5.4.3.2 void ChipChipArray::Grabber::Deposit (Color *color* = Color::Blue) [protected]

Deposits blocks in the storage/unloading unit.

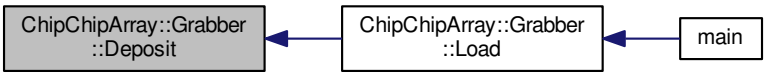
Definition at line 171 of file Grabber.hpp.

```
00171                                   {
00172         if(color == Color::Blue) {
00173             arm.ClawClose();
00174             sleep(1);
00175             arm.BaseTilt(160);
00176             sleep(1);
00177             arm.Elbow(130);
00178             sleep(1);
00179             arm.BaseTurn(47);
00180             sleep(1);
00181             arm.ClawOpen();
00182             sleep(1);
00183         } else {
00184             throw std::runtime_error("Du Idiot! Die Armbewegungen für diese "
00185                     "Farbe sind noch nicht implementiert. Vielleicht sollst "
00186                     "du die englische Phrase lernen 'Would you like fries "
00187                     "with that?");
00188         }
00189     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



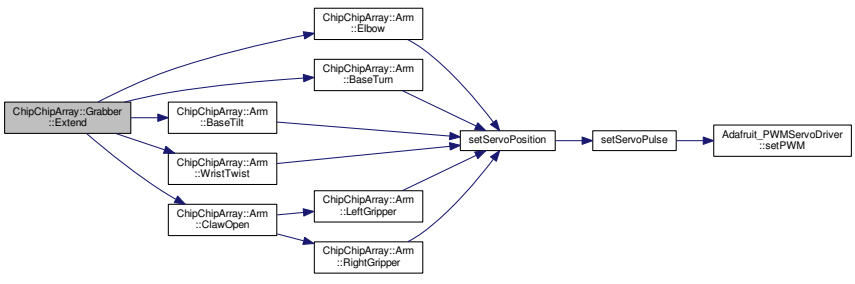
5.4.3.3 void ChipChipArray::Grabber::Extend ( ) [protected]

Sets arm to generic position roughly right above a stack of blocks.

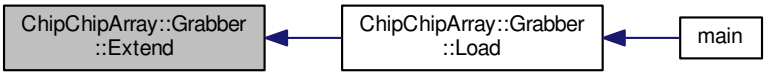
Definition at line 191 of file Grabber.hpp.

```
00191                           {
00192         arm.Elbow(180);
00193         usleep(500000);
00194         arm.BaseTurn(132);
00195         arm.BaseTilt(125);
00196         arm.Elbow(150);
00197         arm.WristTwist(90);
00198         arm.ClawOpen();
00199         sleep(2);
00200     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.4 Result ChipChipArray::Grabber::Load ( )

Loads a block(s) (if possible) at the robot's current position.

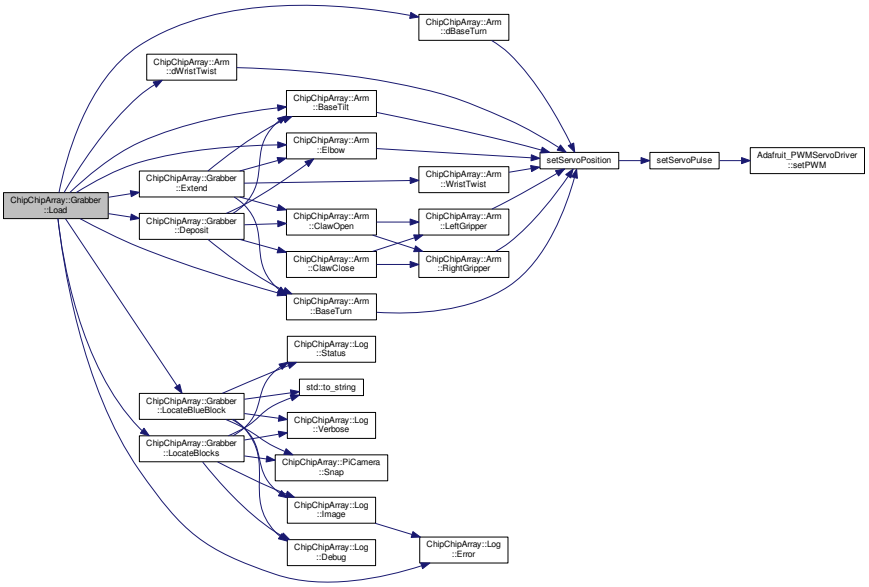
Returns

the number of half and whole blocks loaded

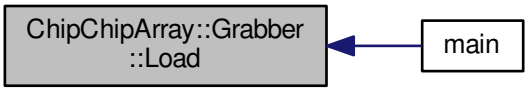
Definition at line 202 of file Grabber.hpp.

```
00202                           {
00203         for(uint8 i = 0; i < 2; i++) {
00204             Extend();
00205 
00206             try {
00207                 Block block = (zone == Zone::A)
00208                     ? LocateBlocks(Color::Blue) :
      LocateBlueBlock();
00209 
00210                 float32 baseKonstant = 0.5;
00211                 if(block.dRightLeft > 0) baseKonstant *= -1;
00212                 float32 degree = baseKonstant * std::sqrt(block.dRightLeft);
00213                 arm.dBaseTurn(degree);
00214                 arm.dWristTwist(-degree);
00215                 sleep(1);
00216                 arm.BaseTilt(140);
00217                 sleep(1);
00218 
00219                 uint8 bend = (i == 0 ? 100 : 90);
00220 
00221                 // lower claw over block
00222                 for(uint8 j = 140; j >= bend; j -= 10) {
00223                     arm.Elbow(j);
00224                     sleep(1);
00225                 }
00226 
00227                 // deposit in bin
00228                 sleep(1);
00229                 Deposit();
00230             } catch(std::exception ex) {
00231                 log.Error(std::string("An exception occured attempting "
00232                           "to load the blocks in function Grabber::Load(): ")
00233                         + ex.what());
00234             }
00235 
00236 
00237             if(i == 0) {
00238                 arm.BaseTurn(132);
00239             } else {
00240                 arm.BaseTurn(135);
00241                 sleep(1);
00242                 arm.BaseTilt(180);
00243                 sleep(1);
00244                 arm.Elbow(90);
00245                 sleep(1);
00246                 arm.Elbow(45);
00247                 sleep(1);
00248                 arm.Elbow(0);
00249             }
00250         }
00251     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.5 Block ChipChipArray::Grabber::LocateBlocks (Color *color* = Color::Perrywinkle) [protected]

Loads a stack of blocks for all zones. Does multiple colors and both sizes.

Parameters

color	the color block for which to search. Perrywinkle denotes searching for all colors (because who actually knows what color perrywinkle is?).

Returns

Block instance representing the block found

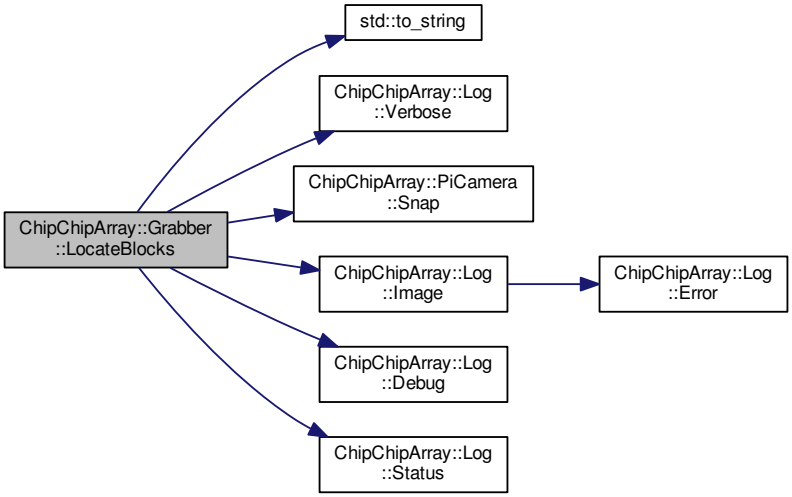
Definition at line 254 of file Grabber.hpp.

```
00254                                       {
00255         invokeCount++;
00256         std::string logstr = "Locating blocks";
00257 
00258         if(color == Color::Perrywinkle) {
00259             logstr += " (" + std::to_string(color) + ")";
```

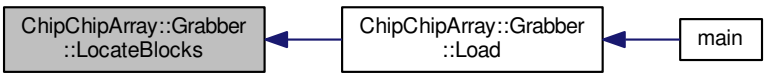
```
00260         }
00261
00262         log.Verbose(logstr);
00263
00264         cv::Mat imgOrig;
00265         cv::transpose(cam.Snap(), imgOrig);
00266
00267         cv::Mat imgHSV;
00268         cv::Mat imgThresh;
00269         std::vector<cv::Rect> blocks;
00270         std::vector<Color> colors;
00271
00272         uint8 loopNum = (color == Color::Perrywinkle ? rangeVals.size() : 1);
00273
00274         for(int i = 0; i < loopNum; i++) {
00275             if(loopNum > 1) {
00276                 switch(i) {
00277                     case 0:
00278                         color = Color::Red;
00279                         break;
00280
00281                     case 1:
00282                         color = Color::Green;
00283                         break;
00284
00285                     case 2:
00286                         color = Color::Blue;
00287                         break;
00288
00289                         /* Must be last, because it changes imgHSV from HSV space
00290                          * to YUV space. */
00291                     case 3:
00292                         color = Color::Yellow;
00293                         break;
00294                 }
00295             }
00296
00297             log.Verbose("Searching: " + std::to_string(color));
00298
00299             if(color == Color::Yellow) {
00300                 cv::Mat temp;
00301                 imgOrig.copyTo(temp);
00302                 cv::cvtColor(imgOrig, imgHSV, CV_BGR2YUV);
00303                 cv::cvtColor(imgHSV, temp, CV_HSV2BGR);
00304                 cv::cvtColor(temp, imgHSV, cv::COLOR_BGR2HSV);
00305                 log.Image(temp, "yuv_yellow_" + std::to_string(color)
00306                         + "_" + std::to_string(zone)
00307                         + std::to_string(invokeCount)
00308                         + ".bmp");
00309             } else {
00310                 cv::cvtColor(imgOrig, imgHSV, cv::COLOR_BGR2HSV);
00311             }
00312
00313             cv::inRange(imgHSV, rangeVals[color][0],
00314                     rangeVals[color][1], imgThresh);
00315
00316             /*
00317              * Not quite sure what all this does, but it seems to
00318              * relate to smoothing the image
00319              */
00320             cv::erode(imgThresh, imgThresh,
00321                     cv::getStructuringElement(
00322                         cv::MORPH_ELLIPSE,
00323                         cv::Size(5, 5)));
00324             cv::dilate(imgThresh, imgThresh,
00325                     cv::getStructuringElement(
00326                         cv::MORPH_ELLIPSE,
00327                         cv::Size(5, 5)));
00328             cv::dilate(imgThresh, imgThresh,
00329                     cv::getStructuringElement(
00330                         cv::MORPH_ELLIPSE,
00331                         cv::Size(5, 5)));
00332             cv::erode(imgThresh, imgThresh,
00333                     cv::getStructuringElement(
00334                         cv::MORPH_ELLIPSE,
00335                         cv::Size(5, 5)));
00336
00337             log.Image(imgThresh, "thresh_" + std::to_string(color)
00338                     + "_" + std::to_string(zone)
00339                     + std::to_string(invokeCount)
00340                     + ".bmp");
00341
00342             // calculate contours
00343             std::vector<std::vector<cv::Point>> contours;
00344             cv::findContours(imgThresh, contours, CV_RETR_TREE,
00345                     CV_CHAIN_APPROX_SIMPLE,
00346                     cv::Point(0, 0));
```

```
00347             std::vector<std::vector<cv::Point>>
00348                 contours_poly(contours.size());
00349             std::vector<cv::Rect> bounds(contours.size());
00350 
00351             // find rectangle around polygon-ish shapes
00352             for(int i = 0; i < contours.size(); i++) {
00353                 uint32 area = cv::contourArea(contours[i]);
00354 
00355                 // determine if block and add to blocks vector
00356                 if(area > MIN_HALF_BLOCK_SIZE) {
00357                     cv::approxPolyDP(cv::Mat(contours[i]),
00358                             contours_poly[i], 20,
00359                             false);
00360                     cv::Rect rect = cv::boundingRect(
00361                             cv::Mat(contours_poly[i]));
00362                     log.Debug(std::to_string(color)
00363                             + " block detected "
00364                             "with area "
00365                             + std::to_string(
00366                                 area));
00367                     blocks.push_back(rect);
00368                     colors.push_back(color);
00369                 }
00370             }
00371         }
00372 
00373         if(blocks.size() == 0) {
00374             log.Image(imgOrig, "original_" + std::to_string(
      zone)
00375                     + std::to_string(invokeCount)
00376                     + "_no_blocks.bmp");
00377             throw std::runtime_error("No blocks found!");
00378         } else {
00379             log.Status(std::to_string(blocks.size())
00380                     + " blocks found");
00381         }
00382 
00383         // coordinates start in top right
00384         Block block = Block(blocks[0], colors[0]);
00385 
00386         if(blocks.size() > 1) {
00387             for(int i = 1; i < blocks.size(); i++) {
00388                 if((side == Side::Right && blocks[i].x
00389                         > block.topLeft.x)
00390                         || (side == Side::Left
00391                             && blocks[i].x
00392                             < block.topLeft.x)) {
00393                     block = Block(blocks[i], colors[i]);
00394                 }
00395             }
00396         }
00397 
00398         log.Status(std::to_string(block.color) + " block is located");
00399 
00400         log.Debug("Block properties => area: " + std::to_string(block.area)
00401                 + ", height: " + std::to_string(block.height) + ", width: "
00402                 + std::to_string(block.width) + ", offset: "
00403                 + std::to_string(block.offset) + ", color: "
00404                 + std::to_string(block.color) + ", size: "
00405                 + std::to_string(block.size));
00406 
00407         /*
00408          * Draw surrounding rectangles from above on original
00409          * image.
00410          */
00411         cv::rectangle(imgOrig, block.topLeft , block.bottomRight,
00412                 cv::Scalar(255, 0, 0), 4, 8);
00413         log.Image(imgOrig, "original_" + std::to_string(zone)
00414                 + std::to_string(invokeCount)
00415                 + ".bmp");
00416 
00417         return block;
00418     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.6 Block ChipChipArray::Grabber::LocateBlueBlock ( ) [protected]

Finds whole, blue blocks.

Returns

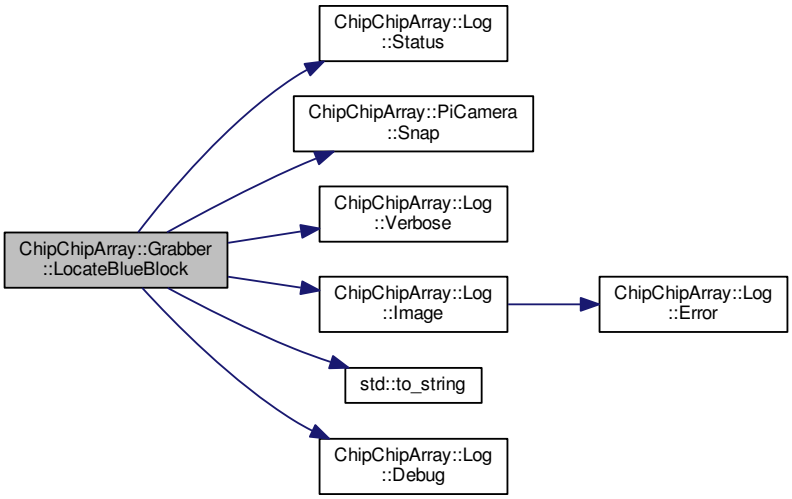
Block instance representing the block found

Definition at line 420 of file Grabber.hpp.

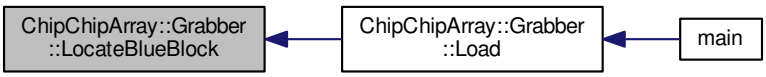
```
00420                                   {
00421         std::vector<cv::Mat> channels;
00422         std::vector<cv::Rect> blocks;
00423 
00424         invokeCount++;
00425         log.Status("Locating blue blocks");
00426 
00427         cv::Mat img;
00428         cv::Mat imgThresh;
00429         cv::split(cam.Snap(), channels);
00430         cv::transpose(channels[0], img);
00431 
00432         log.Verbose("Searching: Blue block");
00433         cv::inRange(img, 30, 255, imgThresh);
00434         log.Image(imgThresh, "thresh_blue" + std::to_string(
      zone)
```

```
00435                 + std::to_string(invokeCount) + ".bmp");
00436
00437         // calculate contours
00438         std::vector<std::vector<cv::Point>> contours;
00439         cv::findContours(imgThresh, contours, CV_RETR_TREE,
00440                 CV_CHAIN_APPROX_SIMPLE,
00441                 cv::Point(0, 0));
00442         std::vector<std::vector<cv::Point>>
00443             contours_poly(contours.size());
00444         std::vector<cv::Rect> bounds(contours.size());
00445
00446         // find rectangle around polygon-ish shapes
00447         for(int i = 0; i < contours.size(); i++) {
00448             uint32 area = cv::contourArea(contours[i]);
00449
00450             // determine if block and add to blocks vector
00451             if(area > MIN_HALF_BLOCK_SIZE) {
00452                 cv::approxPolyDP(cv::Mat(contours[i]),
00453                         contours_poly[i], 20,
00454                         false);
00455                 cv::Rect rect = cv::boundingRect(
00456                         cv::Mat(contours_poly[i]));
00457                 log.Debug("Blue block detected with area "
00458                         + std::to_string(area));
00459                 blocks.push_back(rect);
00460             }
00461         }
00462
00463
00464         if(blocks.size() == 0) {
00465             log.Image(img, "original_" + std::to_string(zone)
00466                     + std::to_string(invokeCount)
00467                     + "_no_blocks.bmp");
00468             throw std::runtime_error("No blocks found!");
00469         } else {
00470             log.Status(std::to_string(blocks.size())
00471                     + " blocks found");
00472         }
00473
00474         // coordinates start in top right
00475         Block block = Block(blocks[0], Color::Blue);
00476
00477         if(blocks.size() > 1) {
00478             for(int i = 1; i < blocks.size(); i++) {
00479                 if((side == Side::Right && blocks[i].x
00480                         > block.topLeft.x)
00481                         || (side == Side::Left
00482                             && blocks[i].x
00483                             < block.topLeft.x)) {
00484                     block = Block(blocks[i], Color::Blue);
00485                 }
00486             }
00487         }
00488
00489         log.Status(std::to_string(block.color) + " block is located");
00490
00491         log.Debug("Block properties => area: " + std::to_string(block.area)
00492                 + ", height: " + std::to_string(block.height) + ", width: "
00493                 + std::to_string(block.width) + ", offset: "
00494                 + std::to_string(block.offset) + ", color: "
00495                 + std::to_string(block.color) + ", size: "
00496                 + std::to_string(block.size));
00497
00498         /*
00499          * Draw surrounding rectangles from above on original
00500          * image.
00501          */
00502         cv::rectangle(img, block.topLeft , block.bottomRight,
00503                 cv::Scalar(255, 0, 0), 4, 8);
00504         log.Image(img, "original_" + std::to_string(zone)
00505                 + std::to_string(invokeCount)
00506                 + ".bmp");
00507
00508         return block;
00509     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.4 Member Data Documentation

5.4.4.1 PiCamera ChipChipArray::Grabber::cam [protected]

The Raspicam

Definition at line 64 of file Grabber.hpp.

5.4.4.2 Side ChipChipArray::Grabber::side [protected]

The side from which the robot is coming (i.e., the side where the higher priority blocks are to be picked up.

Definition at line 71 of file Grabber.hpp.

5.4.4.3 Zone ChipChipArray::Grabber::zone [protected]

The zone in which blocks are being loaded.

Definition at line 76 of file Grabber.hpp.

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

- src/Grabber.hpp

5.5 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)

5.5.1 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 35 of file Log.hpp.

5.5.2 Constructor & Destructor Documentation

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

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

Definition at line 41 of file Log.hpp.

```
00041 {};
```

5.5.2.2 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.

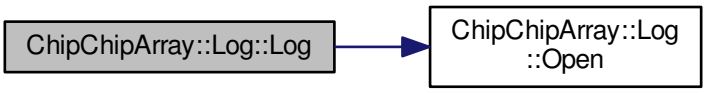
Parameters

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

Definition at line 187 of file Log.hpp.

```
00187                                     {
00188 #ifdef LOG
00189         Open(dir, mode);
00190 #endif
00191     }
```

Here is the call graph for this function:



5.5.2.3 ChipChipArray::Log::~Log ( )

Destroys the Log and closes the logfile.

Definition at line 193 of file Log.hpp.

```
00193               {
00194 #ifdef LOG
00195         try {
00196             file.flush();
00197             file.close();
00198         } catch (std::ofstream::failure f) {
00199             LogError("Gosh dang it! A fatal error has occured "
00200                     "closing the logfile.", f);
00201         }
00202 #endif
00203     }
```

5.5.3 Member Function Documentation

5.5.3.1 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.

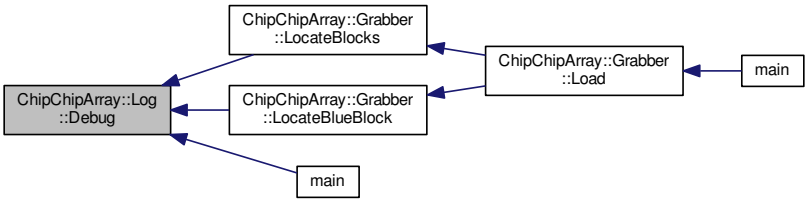
Parameters

mesg	the message to record in the logfile

Definition at line 205 of file Log.hpp.

```
00205                               {
00206 #ifdef LOG
00207         try {
00208             file << "DEBUG: " << mesg << std::endl;
00209             file.flush();
00210         } catch(std::ofstream::failure f) {
00211             LogError("Debug() write error", f);
00212         }
00213 #endif
00214     }
```

Here is the caller graph for this function:



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

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

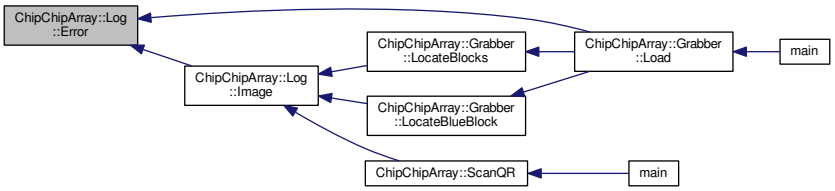
Parameters

mesg	the message to record in the log

Definition at line 216 of file Log.hpp.

```
00216                               {
00217 #ifdef LOG
00218         try {
00219             file << "ERROR: " << mesg << std::endl;
00220             file.flush();
00221         } catch(std::ofstream::failure f) {
00222             LogError("Error() write error", f);
00223         }
00224 #endif
00225     }
```

Here is the caller graph for this function:



5.5.3.3 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

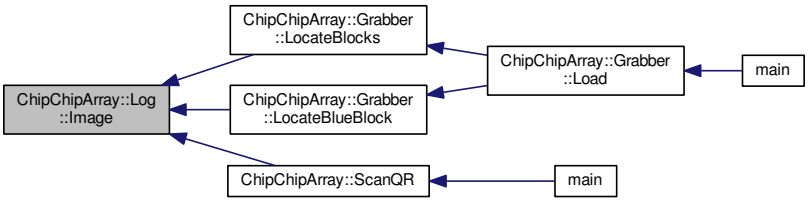
image	the image to save
filename	the filename for the saved image

Definition at line 227 of file Log.hpp.

```
00227                                                {
00228 #ifdef LOG
00229         try {
00230             cv::imwrite(dir + std::string(filename), image);
00231         } catch(std::ofstream::failure f) {
00232             LogError("Image() write error", f);
00233         } catch(std::exception ex) {
00234             Error("Error writing image " + std::string(filename));
00235         }
00236 #endif
00237     }
```

Here is the call graph for this function:

Here is the caller graph for this function:



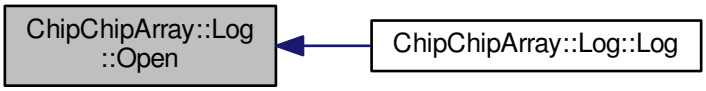
5.5.3.4 void ChipChipArray::Log::Open (auto *dir*, LogMode *mode* = LogMode::Text)

Definition at line 247 of file Log.hpp.

```
00247                                            {
00248 #ifdef LOG
00249         // format date and time
00250         char date[32];
00251         time_t sec = time(nullptr);
00252         struct tm * loctime = localtime(&sec);
00253         strftime(date, 32, "%m-%d_%H-%M-%S", loctime);
00254
00255         // create temperary strings
00256         this->dir = std::string(dir);
00257         std::string datestr = std::string(date);
00258
00259         // add path separator if necessary
00260         if(this->dir[this->dir.length() - 1] != PATH_SEP) {
00261             this->dir += PATH_SEP;
00262         }
```

```
00263
00264         // add directory for log and images if necessary
00265         if(mode == LogMode::Multi) this->dir += datestr + PATH_SEP;
00266
00267         int ret = mkdir(this->dir.c_str(), S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP
00268                 | S_IROTH | S_IWOTH | S_IXUSR | S_IXGRP
00269                 | S_IXOTH);
00270
00271         filename = this->dir + datestr + ".log";
00272
00273         // set class mode
00274         this->mode = mode;
00275
00276         // Initializing file
00277         file.exceptions(std::ofstream::failbit
00278                 | std::ofstream::badbit);
00279
00280         try {
00281             file.open(filename, std::ofstream::out
00282                     | std::ofstream::app);
00283         } catch(std::ofstream::failure ex) {
00284             LogError("Oh, no! An error has occurred opening the "
00285                     "log file.", ex);
00286         }
00287 #endif
00288     }
```

Here is the caller graph for this function:



5.5.3.5 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.

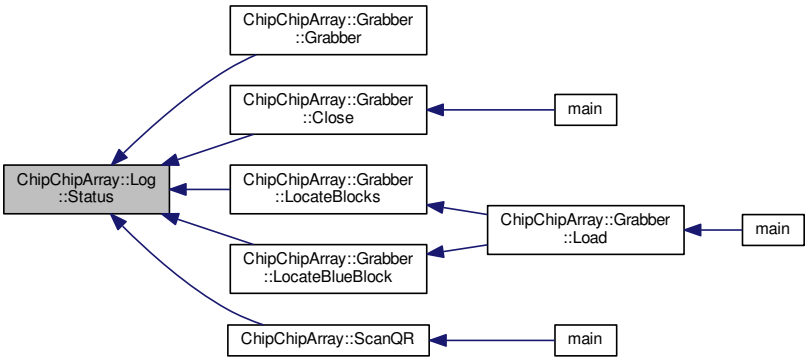
Parameters

mesg	the message to record in the logfile

Definition at line 290 of file Log.hpp.

```
00290                               {
00291 #ifdef LOG
00292         try {
00293             file << "STATUS: " << mesg << std::endl;
00294             file.flush();
00295         } catch (std::ofstream::failure f) {
00296             LogError("Status() write error", f);
00297         }
00298 #endif
00299     }
```

Here is the caller graph for this function:



5.5.3.6 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 301 of file Log.hpp.

```
00301                                           {
00302 #ifdef LOG
00303         try {
00304             file << "VARIABLE: " << name << " = " << value
00305                 << std::endl;
00306             file.flush();
00307         } catch(std::ofstream::failure f) {
00308             LogError("Variable() write error", f);
00309         }
00310 #endif
00311     }
```

Here is the caller graph for this function:

5.5.3.7 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.

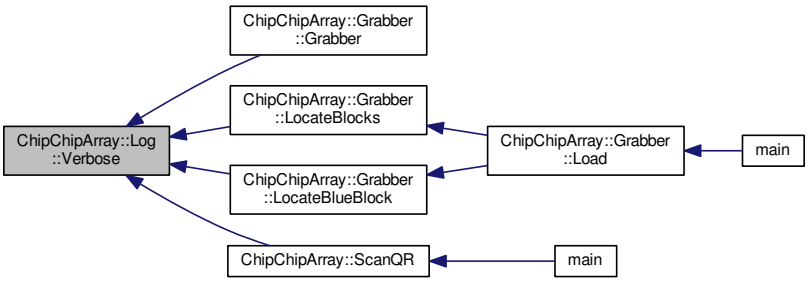
Parameters

mesg	the message to record in the logfile

Definition at line 313 of file Log.hpp.

```
00313                                {
00314 #ifdef LOG
00315         try {
00316             file << "VERBOSE: " << mesg << std::endl;
00317             file.flush();
00318         } catch(std::ofstream::failure f) {
00319             LogError("Verbose() write error", f);
00320         }
00321 #endif
00322     }
```

Here is the caller graph for this function:



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

- src/Log.hpp

5.6 ChipChipArray::PiCamera Class Reference

`#include <PiCamera.hpp>`

Public Member Functions

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

5.6.1 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 24 of file PiCamera.hpp.

5.6.2 Constructor & Destructor Documentation

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

Opens the camera and configures it for color images.

Definition at line 29 of file PiCamera.hpp.

```
00029 : PiCamera(true) {};
```

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

Opens the camera.

Parameters

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

Definition at line 59 of file PiCamera.hpp.

```
00059                                   {
00060         cam.set(CV_CAP_PROP_FORMAT, (useColor ? CV_16UC3 : CV_16UC1));
00061         cam.open();
00062         usleep(500000);  // required to allow camera time to adjust!
00063     }
```

5.6.3 Member Function Documentation

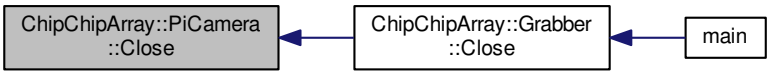
5.6.3.1 void ChipChipArray::PiCamera::Close ( )

Closes connection to camera.

Definition at line 65 of file PiCamera.hpp.

```
00065                           {
00066         cam.release();
00067     }
```

Here is the caller graph for this function:



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

Makes picture.

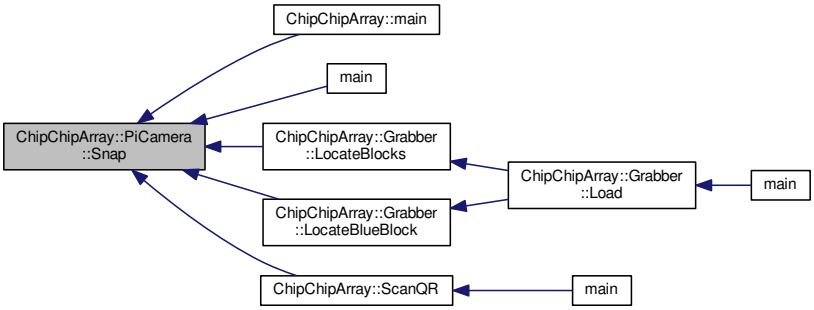
Returns

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

Definition at line 69 of file PiCamera.hpp.

```
00069                           {
00070         if(!cam.isOpened()) throw std::runtime_error("Camera "
00071                 "is not open!");
00072
00073         cv::Mat image;
00074         cam.grab();
00075         cam.retrieve(image);
00076         return image;
00077     }
```

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

Doxygen configuration file.

6.1.1 Detailed Description

Doxygen configuration file.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file doxygen.config.

6.2 doxygen.config

```
00001 PROJECT_NAME = "ChipChipArray"
00002
00003 INPUT = src/ etc/doxygen.config makefile
00004 OUTPUT_DIRECTORY = doc/
00005
00006 GENERATE_HTML = YES
00007 GENERATE_RTF = YES
00008 GENERATE_LATEX = YES
00009 GENERATE_MAN = YES
00010 GENERATE_XML = NO
00011 GENERATE_DOCBOOK = NO
00012
00013 USE_PDF_LATEX = YES
00014 USE_PDF_HYPERLINKS = YES
00015
00016 RECURSIVE = YES
00017 SOURCE_BROWSER = YES
00018 SOURCE_TOOLTIPS = YES
00019 EXTRACT_ALL = YES
00020 DISABLE_INDEX = NO
00021 GENERATE_TREEVIEW = YES
00022 SEARCHENGINE = YES
00023 SERVER_BASED_SEARCH = NO
00024 USE_MDFILE_AS_MAINPAGE = README.md
00025
00026 LATEX_SOURCE_CODE = YES
00027 STRIP_CODE_COMMENTS = YES
00028 INLINE_SOURCES = YES
00029
00030 HAVE_DOT = YES
00031 CALL_GRAPH = YES
00032 CALLER_GRAPH = YES
```

6.3 makefile File Reference

Project makefile containing recipes for compiling the actual application, test applications, and generating documentation.

6.3.1 Detailed Description

Project makefile containing recipes for compiling the actual application, test applications, and generating documentation.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file makefile.

6.4 makefile

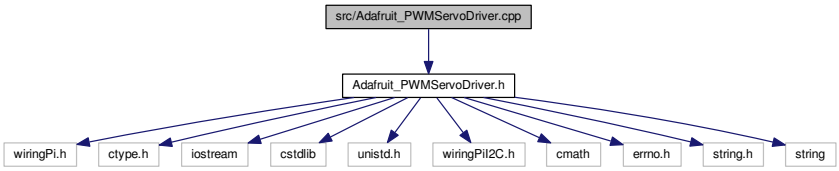
```
00001 GCC = g++-4.9
00002 ARM = -L/usr/local/lib -lwiringPi
00003 CPPFLAGS = -g -std=gnu++14
00004 CVFLAGS = -lraspicam -lraspicam_cv -lmmal -lmmal_core -lmmal_util -lzbar -lopencv_core
       -lopencv_highgui -lopencv_imgproc
00005 LOG = -DLOG
00006
00007 export LIBRARY_PATH=/opt/vc/lib:/usr/lib/arm-linux-gnueabihf
00008
00009 arm:
00010    $(GCC) src/dark_magic.cpp -o bin/arm $(ARM) $(CPPFLAGS)
00011
00012 block-test:
00013    $(GCC) src/cv_shape.cpp -o bin/cvshape $(CVFLAGS) $(CPPFLAGS)
00014
00015 channel-test:
00016    $(GCC) src/cv_channel_test.cpp -o bin/channeltest $(CVFLAGS) $(CPPFLAGS)
00017
00018
00019 comp:
00020    $(GCC) src/main.cpp -o bin/main $(CVFLAGS) $(CPPFLAGS) $(ARM) -lpthread
00021
00022 configure:
00023    sudo apt-get install -y libopencv-dev libzbar-dev cmake doxygen libgl1-meda-dri
00024    git clone https://github.com/cedricve/raspicam
00025    cd raspicam; mkdir build; cd build; cmake ..; make; sudo make install; sudo ldconfig;
00026    sudo rm -r raspicam
00027    mkdir docs
00028
00029 cv-test:
00030    $(GCC) src/cv_test.cpp -o bin/cvtest $(CVFLAGS) $(CPPFLAGS) $(LOG)
00031
00032 img:
00033    $(GCC) src/img.cpp -o bin/img $(CVFLAGS) $(CPPFLAGS)
00034
00035 jacob-algorithm-test:
00036    $(GCC) src/jacob_alg_test.cpp -o bin/jacobalgtest $(CVFLAGS) $(CPPFLAGS)
00037
00038 loading-test:
00039    $(GCC) src/loading_test.cpp -o bin/loadingtest $(CPPFLAGS) $(LOG) $(ARM) $(CVFLAGS)
00040
00041 log-test:
00042    $(GCC) src/log_test.cpp -o bin/logtest $(CPPFLAGS) $(LOG)
00043
00044 net-cv-hue-test:
00045    $(GCC) src/cv_hue.cpp -o bin/cvhue $(CPPFLAGS) $(CVFLAGS)
00046
00047 net-qr-test:
00048    $(GCC) src/net_qr_test.cpp -o bin/netqrtest $(CPPFLAGS) $(CVFLAGS)
00049
00050 qr-test:
00051    $(GCC) src/qr_test.cpp -o bin/qrtest $(CPPFLAGS) $(CVFLAGS) $(LOG)
00052
00053 servotrip:
00054    $(GCC) src/main.cpp -o bin/servotrip $(CPPFLAGS) $(ARM)
00055
00056 docs:
```

```
00057    rm -r doc/
00058    doxygen etc/doxygen.config
00059    cd doc/latex; make pdf;
00060    git reset
00061    git add doc/.
00062    git commit -m "Updated documentation."
00063    git push
00064
00065 count:
00066    grep -r "src/" -e "Samuel Andrew Wisner" -l | xargs wc -l  # works assuming there's no
       subdirectories
```

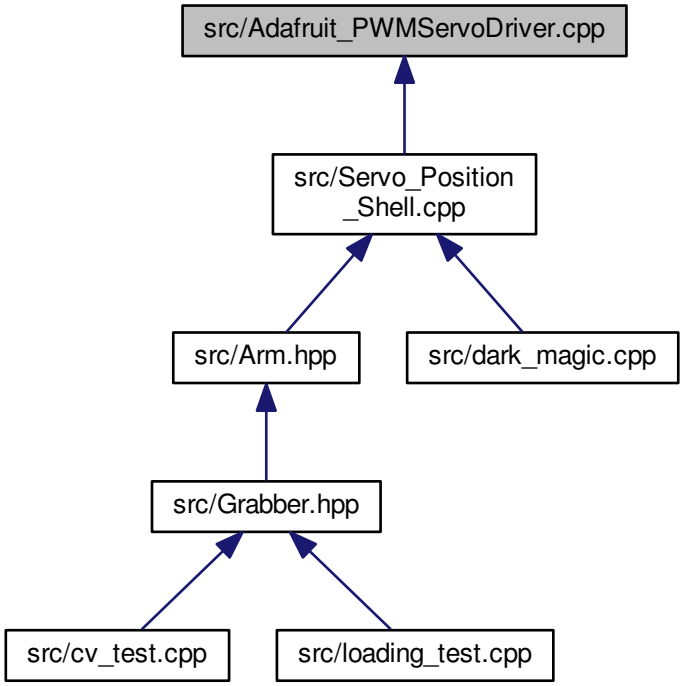
6.5 src/Adafruit_PWMServoDriver.cpp File Reference

Contains the function and class definitions necessary for the PWM servo driver.

`#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

6.5.1 Detailed Description

Contains the function and class definitions necessary for the PWM servo driver.

Author

Limor Frief/Ladyadda
Nickolas Neely

Definition in file Adafruit_PWMServoDriver.cpp.

6.5.2 Macro Definition Documentation

6.5.2.1 #define ENABLE_DEBUG_OUTPUT false

Definition at line 35 of file Adafruit_PWMServoDriver.cpp.

6.6 Adafruit_PWMServoDriver.cpp

```
00001
00009 /* File: Adafruit_PWMServoDriver.cpp
00010  * Credit to Limor Fried/Ladyada for PWM servo driver code.
00011  * Editted by: Nickolas Neely for ChipChipArray Raspberry Pi
00012  */
00013
00014 /*************************************************
00015   This is a library for our Adafruit 16-channel PWM & Servo driver
00016
00017   Pick one up today in the adafruit shop!
00018   ------> http://www.adafruit.com/products/815
00019
00020   These displays use I2C to communicate, 2 pins are required to
00021   interface. For Arduino UNOs, thats SCL -> Analog 5, SDA -> Analog 4
00022
00023   Adafruit invests time and resources providing this open source code,
00024   please support Adafruit and open-source hardware by purchasing
00025   products from Adafruit!
00026
00027   Written by Limor Fried/Ladyada for Adafruit Industries.
00028   BSD license, all text above must be included in any redistribution
00029  *************************************************/
00030
00031 #include "Adafruit_PWMServoDriver.h"
00032
00033
00034 // Set to true to print some debug messages, or false to disable them.
00035 #define ENABLE_DEBUG_OUTPUT false
00036
00037 Adafruit_PWMServoDriver::Adafruit_PWMServoDriver(
      uint8_t addr) {
00038     _i2caddr = addr;
00039     _i2cFD = -1;
00040 }
00041
00042 void Adafruit_PWMServoDriver::begin(void) {
00043     _i2cFD = wiringPiI2CSetup(_i2caddr);
00044     if (_i2cFD < 0) {
00045         //FIXME: error occurred
00046     }
00047     reset();
00048 }
00049
00050 void Adafruit_PWMServoDriver::reset(void) {
00051     write8(PCA9685_MODE1, 0x0);
00052 }
00053
00054 void Adafruit_PWMServoDriver::setPWMFreq(float freq) {
```

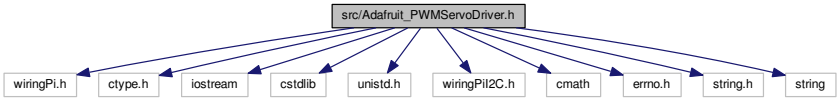
```
00055     //Serial.print("Attempting to set freq ");
00056     //Serial.println(freq);
00057     freq *= 0.9; // Correct for overshoot in the frequency setting (see issue #11).
00058     float prescaleval = 25000000;
00059     prescaleval /= 4096;
00060     prescaleval /= freq;
00061     prescaleval -= 1;
00062     if (ENABLE_DEBUG_OUTPUT) {
00063         cout << "Estimated pre-scale: " << prescaleval << endl;
00064     }
00065     uint8_t prescale = floor(prescaleval + 0.5);
00066     if (ENABLE_DEBUG_OUTPUT) {
00067         cout << "Final pre-scale: " << prescale << endl;
00068     }
00069
00070     uint8_t oldmode = read8(PCA9685_MODE1);
00071     uint8_t newmode = (oldmode & 0x7F) | 0x10; // sleep
00072     write8(PCA9685_MODE1, newmode); // go to sleep
00073     write8(PCA9685_PRESCALE, prescale); // set the prescaler
00074     write8(PCA9685_MODE1, oldmode);
00075     usleep(5000);
00076     write8(PCA9685_MODE1, oldmode | 0xa1); //  This sets the MODE1 register to turn on auto
      increment.
00077     // This is why the beginTransmission below was not working.
00078     //  Serial.print("Mode now 0x"); Serial.println(read8(PCA9685_MODE1), HEX);
00079 }
00080
00081 void Adafruit_PWMServoDriver::setPWM(uint8_t num,
     uint16_t on, uint16_t off) {
00082     //Serial.print("Setting PWM "); Serial.print(num); Serial.print(": "); Serial.print(on);
      Serial.print("->"); Serial.println(off);
00083
00084     int result = wiringPiI2CWriteReg16(_i2cFD, LED0_ON_L + 4 * num, on);
00085     if (result < 0) {
00086         string s(strerror(errno));
00087         cout << "setPWM error: " << s.c_str() << endl;
00088     }
00089 //    result = wiringPiI2CWrite(_i2cFD, on);
00090 //    if (result < 0) {
00091 //        string s(strerror(errno));
00092 //        cout << "setPWM error: " << s.c_str() << endl;
00093 //    }
00094 //    result = wiringPiI2CWrite(_i2cFD, on >> 8);
00095 //    if (result < 0) {
00096 //        string s(strerror(errno));
00097 //        cout << "setPWM error: " << s.c_str() << endl;
00098 //    }
00099     result = wiringPiI2CWriteReg16(_i2cFD, LED0_OFF_L + 4 * num, off);
00100 //    result = wiringPiI2CWrite(_i2cFD, off);
00101     if (result < 0) {
00102         string s(strerror(errno));
00103         cout << "setPWM error: " << s.c_str() << endl;
00104     }
00105 //    result = wiringPiI2CWrite(_i2cFD, off >> 8);
00106 //    if (result < 0) {
00107 //        string s(strerror(errno));
00108 //        cout << "setPWM error: " << s.c_str() << endl;
00109 //    }
00110 }
00111
00112 // Sets pin without having to deal with on/off tick placement and properly handles
00113 // a zero value as completely off.  Optional invert parameter supports inverting
00114 // the pulse for sinking to ground.  Val should be a value from 0 to 4095 inclusive.
00115
00116 void Adafruit_PWMServoDriver::setPin(uint8_t num,
     uint16_t val, bool invert) {
00117     // Clamp value between 0 and 4095 inclusive.
00118     val = min(val, (uint16_t)4095);
00119     if (invert) {
00120         if (val == 0) {
00121             // Special value for signal fully on.
00122             setPWM(num, 4096, 0);
00123         } else if (val == 4095) {
00124             // Special value for signal fully off.
00125             setPWM(num, 0, 4096);
00126         } else {
00127             setPWM(num, 0, 4095 - val);
00128         }
00129     } else {
00130         if (val == 4095) {
00131             // Special value for signal fully on.
00132             setPWM(num, 4096, 0);
00133         } else if (val == 0) {
00134             // Special value for signal fully off.
00135             setPWM(num, 0, 4096);
00136         } else {
00137             setPWM(num, 0, val);
```

```
00138         }
00139     }
00140 }
00141
00142 uint8_t Adafruit_PWMServoDriver::read8(uint8_t addr) {
00143     int result = wiringPiI2CReadReg8(_i2cFD, addr);
00144     if (result < 0) {
00145         string s(strerror(errno));
00146         cout << "Error read8: " << std::dec << (unsigned int)addr << " -> " << s.c_str() << endl;
00147     }
00148     return result;
00149 }
00150
00151 void Adafruit_PWMServoDriver::write8(uint8_t addr, uint8_t d) {
00152     int result = wiringPiI2CWriteReg8(_i2cFD, addr, d);
00153     if (result < 0) {
00154         string s(strerror(errno));
00155         cout << "Error write8: " << std::dec << (unsigned int)addr << " -> " << (unsigned int)d << " -> " <
      < s.c_str() << endl;
00156     }
00157 }
```

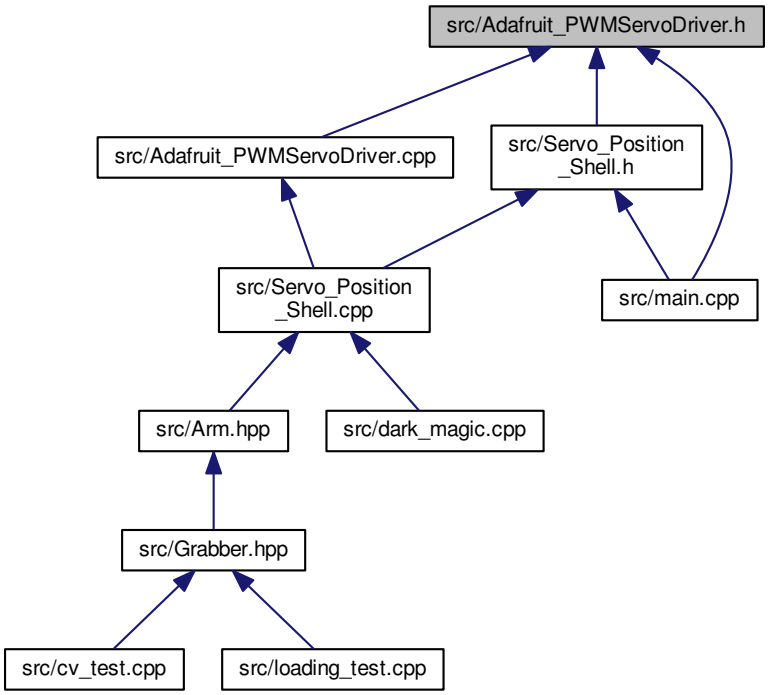
6.7 src/Adafruit_PWMServoDriver.h File Reference

```
#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

6.7.1 Detailed Description

Author

Limor Fried/Ladyada
Nickolas Neely Contains the function and class headers necessary for the PWM servo driver.

Definition in file Adafruit_PWMServoDriver.h.

6.7.2 Macro Definition Documentation

6.7.2.1 #define ALLLED_OFF_H 0xFD

Definition at line 62 of file Adafruit_PWMServoDriver.h.

6.7.2.2 #define ALLLED_OFF_L 0xFC

Definition at line 61 of file Adafruit_PWMServoDriver.h.

6.7.2.3 #define ALLLED_ON_H 0xFB

Definition at line 60 of file Adafruit_PWMServoDriver.h.

6.7.2.4 #define ALLLED_ON_L 0xFA

Definition at line 59 of file Adafruit_PWMServoDriver.h.

6.7.2.5 #define LED0_OFF_H 0x9

Definition at line 57 of file Adafruit_PWMServoDriver.h.

6.7.2.6 #define LED0_OFF_L 0x8

Definition at line 56 of file Adafruit_PWMServoDriver.h.

6.7.2.7 #define LED0_ON_H 0x7

Definition at line 55 of file Adafruit_PWMServoDriver.h.

6.7.2.8 #define LED0_ON_L 0x6

Definition at line 54 of file Adafruit_PWMServoDriver.h.

6.7.2.9 #define PCA9685_MODE1 0x0

Definition at line 51 of file Adafruit_PWMServoDriver.h.

6.7.2.10 #define PCA9685_PRESCALE 0xFE

Definition at line 52 of file Adafruit_PWMServoDriver.h.

6.7.2.11 #define PCA9685_SUBADR1 0x2

Definition at line 47 of file Adafruit_PWMServoDriver.h.

6.7.2.12 #define PCA9685_SUBADR2 0x3

Definition at line 48 of file Adafruit_PWMServoDriver.h.

6.7.2.13 #define PCA9685_SUBADR3 0x4

Definition at line 49 of file Adafruit_PWMServoDriver.h.

6.7.2.14 #define uint16_t unsigned short int

Definition at line 69 of file Adafruit_PWMServoDriver.h.

6.7.2.15 #define uint8_t unsigned char

Definition at line 65 of file Adafruit_PWMServoDriver.h.

6.8 Adafruit_PWMServoDriver.h

```
00001
00009 /* File: Adafruit_PWMServoDriver.h
00010  * Credit to Limor Fried/Ladyada for PWM servo driver code.
00011  * Editted by: Nickolas Neely for ChipChipArray Raspberry Pi
00012  */
00013
00014 /*************************************************
00015   This is a library for our Adafruit 16-channel PWM & Servo driver
00016
00017   Pick one up today in the adafruit shop!
00018   ------> http://www.adafruit.com/products/815
00019
00020   These displays use I2C to communicate, 2 pins are required to
00021   interface. For Arduino UNOs, thats SCL -> Analog 5, SDA -> Analog 4
00022
00023   Adafruit invests time and resources providing this open source code,
00024   please support Adafruit and open-source hardware by purchasing
00025   products from Adafruit!
00026
00027   Written by Limor Fried/Ladyada for Adafruit Industries.
00028   BSD license, all text above must be included in any redistribution
00029  *************************************************/
00030
00031 #ifndef _ADAFRUIT_PWMServoDriver_H
00032 #define _ADAFRUIT_PWMServoDriver_H
00033
00034 #include <wiringPi.h>
00035 #include <ctype.h>
00036 #include <iostream>
00037 #include <cstdlib>
00038 #include <unistd.h>
00039 #include <wiringPiI2C.h>
00040 #include <cmath>
00041 #include <errno.h>
00042 #include <string.h>
00043 #include <string>
00044
00045 using namespace std;
00046
00047 #define PCA9685_SUBADR1 0x2
00048 #define PCA9685_SUBADR2 0x3
00049 #define PCA9685_SUBADR3 0x4
00050
00051 #define PCA9685_MODE1 0x0
00052 #define PCA9685_PRESCALE 0xFE
00053
00054 #define LED0_ON_L 0x6
```

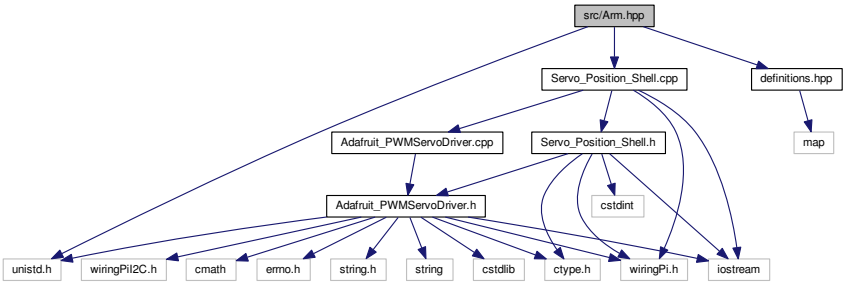
```
00055 #define LED0_ON_H 0x7
00056 #define LED0_OFF_L 0x8
00057 #define LED0_OFF_H 0x9
00058
00059 #define ALLLED_ON_L 0xFA
00060 #define ALLLED_ON_H 0xFB
00061 #define ALLLED_OFF_L 0xFC
00062 #define ALLLED_OFF_H 0xFD
00063
00064 #ifndef uint8_t
00065 #define uint8_t unsigned char
00066 #endif
00067
00068 #ifndef uint16_t
00069 #define uint16_t unsigned short int
00070 #endif
00071
00072
00073 class Adafruit_PWMServoDriver {
00074  public:
00075   Adafruit_PWMServoDriver(uint8_t addr = 0x41);
00076   void begin(void);
00077   void reset(void);
00078   void setPWMFreq(float freq);
00079   void setPWM(uint8_t num, uint16_t on, uint16_t off);
00080   void setPin(uint8_t num, uint16_t val, bool invert=false);
00081
00082  private:
00083   uint8_t _i2caddr;
00084   int _i2cFD;
00085
00086   uint8_t read8(uint8_t addr);
00087   void write8(uint8_t addr, uint8_t d);
00088 };
00089
00090 #endif
```

6.9 src/Arm.hpp File Reference

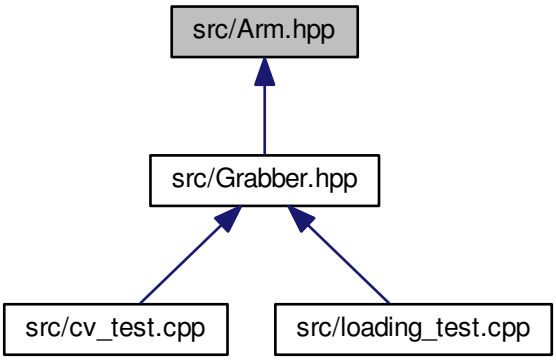
Contains the Arm class used to control the robotic arm.

```
#include <unistd.h>
#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

Macros

- #define WRIST_TWIST WRIST_PAN

6.9.1 Detailed Description

Contains the Arm class used to control the robotic arm.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file Arm.hpp.

6.9.2 Macro Definition Documentation

6.9.2.1 #define WRIST_TWIST WRIST_PAN

Definition at line 15 of file Arm.hpp.

6.10 Arm.hpp

```
00001
```

```
00007 #ifndef Arm_H
00008 #define Arm_H
00009
00010 #include <unistd.h>
00011
00012 #include "definitions.hpp"
00013 #include "Servo_Position_Shell.cpp"
00014
00015 #define WRIST_TWIST WRIST_PAN
00016
00017 namespace ChipChipArray {
00023     class Arm {
00024         public:
00029             Arm();
00030
00034             uint8 servoPos[7] = { 0, 0, 0, 0, 0, 0, 0 };
00035
00041             void BaseTilt(uint8 a);
00042
00048             void BaseTurn(uint8 a);
00049
00053             void ClawOpen();
00054
00060             void ClawClose();
00061
00068             void dBaseTilt(sint16 a);
00069
00076             void dBaseTurn(sint16 a);
00077
00084             void dElbow(sint16 a);
00085
00092             void dWristTilt(sint16 a);
00093
00100             void dWristTwist(sint16 a);
00101
00107             void Elbow(uint8 a);
00108
00115             void Hover(Zone zone);
00116
00122             void WristTilt(uint8 a);
00123
00129             void WristTwist(uint8 a);
00130
00131         protected:
00132
00139             void dLeftGripper(sint16 a);
00140
00147             void dRightGripper(sint16 a);
00148
00154             void LeftGripper(uint8 a);
00155
00161             void RightGripper(uint8 a);
00162
00163         private:
00168             static bool init;
00169     };
00170
00171     bool Arm::init = false;
00172
00173     Arm::Arm() {
00174         if(!init) {
00175             setup();
00176             init = true;
00177         }
00178     }
00179
00180     void Arm::BaseTilt(uint8 a) {
00181         setServoPosition(BASE_TILT, a);
00182         servoPos[BASE_TILT] = a;
00183     }
00184
00185     void Arm::BaseTurn(uint8 a) {
00186         setServoPosition(BASE_TURN, a);
00187         servoPos[BASE_TURN] = a;
00188     }
00189
00190     void Arm::ClawOpen() {
00191         LeftGripper(0);
00192         RightGripper(180);
00193     }
00194
00195     void Arm::ClawClose() {
00196         LeftGripper(180);
00197         RightGripper(0);
00198     }
00199
00200     void Arm::dBaseTilt(sint16 a) {
```

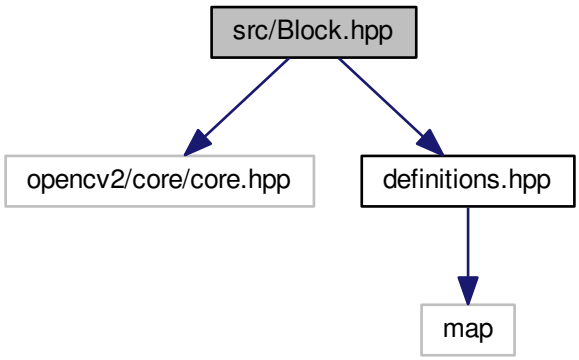
```
00201         a += servoPos[BASE_TILT];
00202         setServoPosition(BASE_TILT, a);
00203         servoPos[BASE_TILT] = a;
00204     }
00205
00206     void Arm::dBaseTurn(sint16 a) {
00207         a += servoPos[BASE_TURN];
00208         setServoPosition(BASE_TURN, a);
00209         servoPos[BASE_TURN] = a;
00210     }
00211
00212     void Arm::dElbow(sint16 a) {
00213         a += servoPos[ELBOW];
00214         setServoPosition(ELBOW, a);
00215         servoPos[ELBOW] = a;
00216     }
00217
00218     void Arm::dLeftGripper(sint16 a) {
00219         a += servoPos[GRIP_LEFT];
00220         setServoPosition(GRIP_LEFT, a);
00221         servoPos[GRIP_LEFT] = a;
00222     }
00223
00224     void Arm::dRightGripper(sint16 a) {
00225         a += servoPos[GRIP_RIGHT];
00226         setServoPosition(GRIP_RIGHT, a);
00227         servoPos[GRIP_RIGHT] = a;
00228     }
00229
00230     void Arm::dWristTilt(sint16 a) {
00231         a += servoPos[WRIST_TILT];
00232         setServoPosition(WRIST_TILT, a);
00233         servoPos[WRIST_TILT] = a;
00234     }
00235
00236     void Arm::dWristTwist(sint16 a) {
00237         a += servoPos[WRIST_TWIST];
00238         setServoPosition(WRIST_TWIST, a);
00239         servoPos[WRIST_TWIST] = a;
00240     }
00241
00242     void Arm::Elbow(uint8 a) {
00243         setServoPosition(ELBOW, a);
00244         servoPos[ELBOW] = a;
00245     }
00246
00247     void Arm::LeftGripper(uint8 a) {
00248         setServoPosition(GRIP_LEFT, a);
00249         servoPos[GRIP_LEFT] = a;
00250     }
00251
00252     void Arm::RightGripper(uint8 a) {
00253         setServoPosition(GRIP_RIGHT, a);
00254         servoPos[GRIP_RIGHT] = a;
00255     }
00256
00257     void Arm::WristTilt(uint8 a) {
00258         setServoPosition(WRIST_TILT, a);
00259         servoPos[WRIST_TILT] = a;
00260     }
00261
00262     void Arm::WristTwist(uint8 a) {
00263         setServoPosition(WRIST_TWIST, a);
00264         servoPos[WRIST_TWIST] = a;
00265     }
00266 }
00267
00268 #endif
```

6.11 src/Block.hpp File Reference

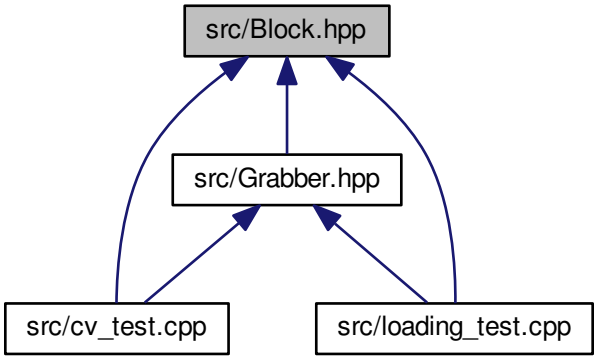
Contains Block class.

```
#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

6.11.1 Detailed Description

Contains Block class.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file Block.hpp.

6.12 Block.hpp

```
00001
00007 #ifndef Block_H
00008 #define Block_H
00009
00010 #include <opencv2/core/core.hpp>
00011
00012 #include "definitions.hpp"
00013
00014 namespace ChipChipArray {
00020     class Block {
00021         public:
00025             uint32 area;
00026
00030             cv::Point bottomLeft;
00031
00035             cv::Point bottomRight;
00036
00041             sint16 dBottom;
00042
00047             sint16 dLeft;
00048
00053             sint16 dRight;
00054
00059             sint16 dTop;
00060
00067             sint16 dTopBottom;
00068
00075             sint16 dRightLeft;
00076
00082             sint16 offset;
00083
00087             uint16 height;
00088
00092             cv::Point topLeft;
00093
00097             cv::Point topRight;
00098
00102             uint16 width;
00103
00107             Color color;
00108
00112             Size size;
00113
00119             Block(cv::Rect rect, Color color);
00120
00121         private:
00125             static const uint16 IMG_HEIGHT = 1280;
00126
00130             static const uint16 IMG_WIDTH = 720;
00131
00136             static const uint32 MIN_WHOLE_BLOCK_SIZE = 50000;
00137     };
00138
00139     Block::Block(cv::Rect rect, Color color) {
00140         // basic geometric properties
00141         area = rect.area();
00142         height = rect.height;
00143         width = rect.width;
00144
00145         // assigning corners
00146         topLeft = rect.tl();
00147         bottomRight = rect.br();
00148         topRight = cv::Point(topLeft.x + width, topLeft.y);
00149         bottomLeft = cv::Point(topLeft.x, topLeft.y +
      height);
00150         offset = (sint16)(topLeft.x + width / 2) - IMG_WIDTH / 2;
00151
00152         // calculating offsets (opencv low coordinates start top left)
00153         dLeft = topLeft.x;
00154         dRight = IMG_WIDTH - topRight.x;
00155         dTop = topLeft.y;
00156         dBottom = IMG_HEIGHT - bottomRight.y;
00157         dTopBottom = dTop - dBottom;
00158         dRightLeft = dRight - dLeft;
```

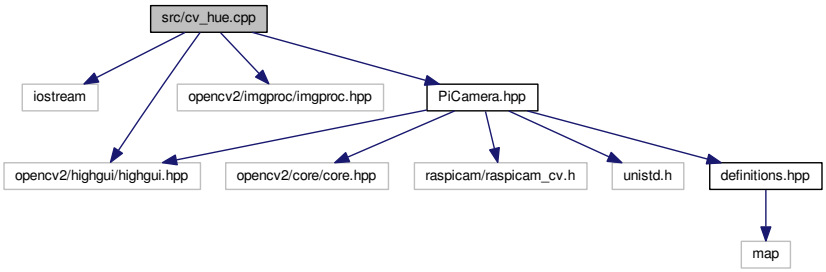
```
00159
00160         // set color and size
00161         this->color = color;
00162         size = area > MIN_WHOLE_BLOCK_SIZE ? Size::Long :
      Size::Short;
00163     }
00164 }
00165 #endif
```

6.13 src/cv_hue.cpp File Reference

Contains program formally used to find HSV values for blocks.

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

Include dependency graph for cv_hue.cpp:



Namespaces

- ChipChipArray

Functions

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

6.13.1 Detailed Description

Contains program formally used to find HSV values for blocks.

Author

Shermal Fernando
Samuel Andrew Wisner, awisner94@gmail.com

Definition in file cv_hue.cpp.

6.14 cv_hue.cpp

```
00001
00008 #include <iostream>
```

```
00009 #include "opencv2/highgui/highgui.hpp"
00010 #include "opencv2/imgproc/imgproc.hpp"
00011
00012 #include "PiCamera.hpp"
00013
00014 using namespace cv;
00015 using namespace std;
00016 using namespace ChipChipArray;
00017
00018 namespace ChipChipArray {
00019
00027     int main( int argc, char** argv ) {
00028         PiCamera cam;
00029         namedWindow("Control", CV_WINDOW_AUTOSIZE); //create a window called "Control"
00030
00031         int iLowH = 170;
00032         int iHighH = 179;
00033
00034         int iLowS = 150;
00035         int iHighS = 255;
00036
00037         int iLowV = 60;
00038         int iHighV = 255;
00039
00040         //Create trackbars in "Control" window
00041         createTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)
00042         createTrackbar("HighH", "Control", &iHighH, 179);
00043
00044         createTrackbar("LowS", "Control", &iLowS, 255); //Saturation (0 - 255)
00045         createTrackbar("HighS", "Control", &iHighS, 255);
00046
00047         createTrackbar("LowV", "Control", &iLowV, 255);//Value (0 - 255)
00048         createTrackbar("HighV", "Control", &iHighV, 255);
00049
00050         int iLastX = -1;
00051         int iLastY = -1;
00052
00053         //Capture a temporary image from the camera
00054         Mat imgTmp = cam.Snap();
00055
00056         //Create a black image with the size as the camera output
00057         Mat imgLines = Mat::zeros( imgTmp.size(), CV_8UC3 );;
00058
00059
00060         while (true)
00061         {
00062             Mat imgOriginal = cam.Snap(); // read a new frame from video
00063             Mat imgHSV;
00064
00065             cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV); //Convert the captured frame from BGR to HSV
00066
00067             Mat imgThresholded;
00068
00069             inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV), imgThresholded);
    //Threshold the image
00070
00071             //morphological opening (removes small objects from the foreground)
00072             erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
    Size(5, 5)) );
00073             dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
    Size(5, 5)) );
00074
00075             //morphological closing (removes small holes from the foreground)
00076             dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
    Size(5, 5)) );
00077             erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
    Size(5, 5)) );
00078
00079             //Calculate the moments of the thresholded image
00080             Moments oMoments = moments(imgThresholded);
00081
00082             /*double dM01 = oMoments.m01;
00083               double dM10 = oMoments.m10;
00084               double dArea = oMoments.m00;
00085
00086             // if the area <= 10000, I consider that the there are no object in the image and it's because
     of the noise, the area is not zero
00087             if (dArea > 50000)
00088             {
00089             //calculate the position of the ball
00090             int posX = dM10 / dArea;
00091             int posY = dM01 / dArea;
00092
00093             if (iLastX >= 0 && iLastY >= 0 && posX >= 0 && posY >= 0)
00094             {
00095             //Draw a red line from the previous point to the current point
00096             line(imgLines, Point(posX, posY), Point(iLastX, iLastY), Scalar(0,0,255), 2);
```

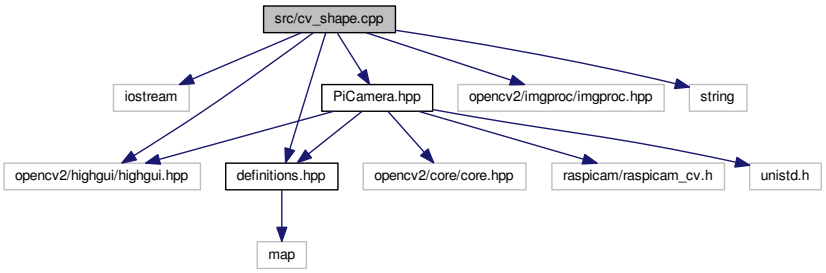
```
00097             }
00098
00099             iLastX = posX;
00100             iLastY = posY;
00101             }
00102             */
00103             imshow("Thresholded Image", imgThresholded); //show the thresholded image
00104
00105             //      imgOriginal = imgOriginal + imgLines;
00106             imshow("Original", imgOriginal); //show the original image
00107
00108             if (waitKey(30) == 27) //wait for 'esc' key press for 30ms. If 'esc' key is pressed, break loop
00109             {
00110                 cout << "esc key is pressed by user" << endl;
00111                 break;
00112             }
00113         }
00114
00115         return 0;
00116     }
00117 }
```

6.15 src/cv_shape.cpp File Reference

Contains a program to aid in determining HSV ranges.

```
#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 ()

6.15.1 Detailed Description

Contains a program to aid in determining HSV ranges.

Author

Shermal Fernando
Samuel Andrew Wisner, awisner94@gmail.com

Definition in file cv_shape.cpp.

6.15.2 Function Documentation

6.15.2.1 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 37 of file cv_shape.cpp.

```
00037           {
00038     PiCamera cam;
00039
00040     // window names
00041     string control = "Settings";
00042     string winThresh = "Image Threshold";
00043     string winContours = "Contours Detected";
00044
00045     // control (trackbar) variables
00046     int lowH = 0;  // hue
00047     int highH = 255;
00048     int lowS = 0;  // saturation
00049     int highS = 255;
00050     int lowV = 0;  // value
00051     int highV = 255;
00052     int polyEps = 3;  // max dif b/t bin shape edge & est poly edge
00053
00054     // opening windows
00055     namedWindow(control, CV_WINDOW_AUTOSIZE);
00056     namedWindow(winThresh, CV_WINDOW_AUTOSIZE);
00057     namedWindow(winContours, CV_WINDOW_AUTOSIZE);
00058
00059     // creating control trackbars
00060     createTrackbar("Hue Min", control, &lowH, highH);
00061     createTrackbar("Hue Max", control, &highH, highH);
00062     createTrackbar("Sat Min", control, &lowS, highS);
00063     createTrackbar("Sat Max", control, &highS, highS);
00064     createTrackbar("Val Min", control, &lowV, highV);
00065     createTrackbar("Val Max", control, &highV, highV);
00066     createTrackbar("Polygon Epsilon", control, &polyEps, 20);
00067
00068     while(true) {
00069         Mat imgOrig = cam.Snap();  // real iage
00070         Mat imgHSV;  // RGB image converted to HSV space
00071         Mat imgThresh;  // binary threshold image
00072         //cvtColor(imgOrig, imgHSV, CV_BGR2YUV);
00073         //cvtColor(imgHSV, imgOrig, CV_HSV2BGR);
00074         cvtColor(imgOrig, imgHSV, COLOR_BGR2HSV);
00075
00076         // create binary image
00077         inRange(imgHSV, Scalar(lowH, lowS, lowV), Scalar(highH, highS,
00078                 highV), imgThresh);
00079
00080         /*
00081          * Not quite sure what all this does, but it seems to
00082          * relate to smoothing the image
00083          */
00084         erode(imgThresh, imgThresh, getStructuringElement(
00085                 MORPH_ELLIPSE, cv::Size(5, 5)));
00086         dilate(imgThresh, imgThresh, getStructuringElement(
00087                 MORPH_ELLIPSE, cv::Size(5, 5)));
00088         dilate(imgThresh, imgThresh, getStructuringElement(
00089                 MORPH_ELLIPSE, cv::Size(5, 5)));
00090         erode(imgThresh, imgThresh, getStructuringElement(
00091                 MORPH_ELLIPSE, cv::Size(5, 5)));
00092
00093         // show binary image in threshold window
```

```
00094         imshow(winThresh, imgThresh);
00095
00096         // calculate contours
00097         vector<vector<Point>> contours;
00098         findContours(imgThresh, contours, CV_RETR_TREE,
00099                 CV_CHAIN_APPROX_SIMPLE, Point(0, 0));
00100        vector<vector<Point>> contours_poly(contours.size());
00101        vector<Rect> bounds(contours.size());
00102        int maxArea = 0;
00103        int offset;
00104
00105        // find rectangle around polygon-ish shapes
00106        for(int i = 0; i < contours.size(); i++) {
00107            approxPolyDP(Mat(contours[i]), contours_poly[i],
00108                    polyEps, false);
00109            bounds[i] = boundingRect(Mat(contours_poly[i]));
00110        }
00111
00112        /*
00113         * Draw surrounding rectangles from above on original
00114         * image.
00115         */
00116        for(int i = 0; i < contours.size(); i++) {
00117            rectangle(imgOrig, bounds[i].tl(), bounds[i].br(),
00118                    Scalar(255, 0, 0), 2, 8, 0);
00119            //drawContours(imgOrig, contours_poly, i,
00120            //      Scalar(255, 0, 0), 4, 8);
00121            int area = bounds[i].width * bounds[i].height;
00122
00123            if(area > maxArea) {
00124                offset = abs(640 - (bounds[i].tl().x + bounds[i].width / 2));
00125                maxArea = area;
00126            }
00127        }
00128
00129        cout << "Block area: " << maxArea << " pixels\t\t"
00130            << "Center offset: " << offset << endl;
00131        imshow(winContours, imgOrig);  // show original image with rectangles
00132        waitKey(50);  // has to be here :(
00133    }
00134 }
```

Here is the call graph for this function:

6.16 cv_shape.cpp

```
00001
00008 #include <iostream>
00009 #include <opencv2/highgui/highgui.hpp>
00010 #include <opencv2/imgproc/imgproc.hpp>
00011 #include <string>
00012
00013 #include "definitions.hpp"
00014 #include "PiCamera.hpp"
00015
00016 using namespace ChipChipArray;
00017 using namespace cv;
00018 using namespace std;
00019
00037 int main() {
00038     PiCamera cam;
00039
00040     // window names
00041     string control = "Settings";
00042     string winThresh = "Image Threshold";
```

```
00043     string winContours = "Contours Detected";
00044
00045     // control (trackbar) variables
00046     int lowH = 0;  // hue
00047     int highH = 255;
00048     int lowS = 0;  // saturation
00049     int highS = 255;
00050     int lowV = 0;  // value
00051     int highV = 255;
00052     int polyEps = 3;  // max dif b/t bin shape edge & est poly edge
00053
00054     // opening windows
00055     namedWindow(control, CV_WINDOW_AUTOSIZE);
00056     namedWindow(winThresh, CV_WINDOW_AUTOSIZE);
00057     namedWindow(winContours, CV_WINDOW_AUTOSIZE);
00058
00059     // creating control trackbars
00060     createTrackbar("Hue Min", control, &lowH, highH);
00061     createTrackbar("Hue Max", control, &highH, highH);
00062     createTrackbar("Sat Min", control, &lowS, highS);
00063     createTrackbar("Sat Max", control, &highS, highS);
00064     createTrackbar("Val Min", control, &lowV, highV);
00065     createTrackbar("Val Max", control, &highV, highV);
00066     createTrackbar("Polygon Epsilon", control, &polyEps, 20);
00067
00068     while(true) {
00069         Mat imgOrig = cam.Snap();  // real iage
00070         Mat imgHSV;  // RGB image converted to HSV space
00071         Mat imgThresh;  // binary threshold image
00072         //cvtColor(imgOrig, imgHSV, CV_BGR2YUV);
00073         //cvtColor(imgHSV, imgOrig, CV_HSV2BGR);
00074         cvtColor(imgOrig, imgHSV, COLOR_BGR2HSV);
00075
00076         // create binary image
00077         inRange(imgHSV, Scalar(lowH, lowS, lowV), Scalar(highH, highS,
00078                     highV), imgThresh);
00079
00080         /*
00081          * Not quite sure what all this does, but it seems to
00082          * relate to smoothing the image
00083          */
00084         erode(imgThresh, imgThresh, getStructuringElement(
00085                     MORPH_ELLIPSE, cv::Size(5, 5)));
00086         dilate(imgThresh, imgThresh, getStructuringElement(
00087                     MORPH_ELLIPSE, cv::Size(5, 5)));
00088         dilate(imgThresh, imgThresh, getStructuringElement(
00089                     MORPH_ELLIPSE, cv::Size(5, 5)));
00090         erode(imgThresh, imgThresh, getStructuringElement(
00091                     MORPH_ELLIPSE, cv::Size(5, 5)));
00092
00093         // show binary image in threshold window
00094         imshow(winThresh, imgThresh);
00095
00096         // calculate contours
00097         vector<vector<Point>> contours;
00098         findContours(imgThresh, contours, CV_RETR_TREE,
00099                 CV_CHAIN_APPROX_SIMPLE, Point(0, 0));
00100         vector<vector<Point>> contours_poly(contours.size());
00101         vector<Rect> bounds(contours.size());
00102         int maxArea = 0;
00103         int offset;
00104
00105         // find rectangle around polygon-ish shapes
00106         for(int i = 0; i < contours.size(); i++) {
00107             approxPolyDP(Mat(contours[i]), contours_poly[i],
00108                     polyEps, false);
00109             bounds[i] = boundingRect(Mat(contours_poly[i]));
00110         }
00111
00112         /*
00113          * Draw surrounding rectangles from above on original
00114          * image.
00115          */
00116         for(int i = 0; i < contours.size(); i++) {
00117             rectangle(imgOrig, bounds[i].tl(), bounds[i].br(),
00118                     Scalar(255, 0, 0), 2, 8, 0);
00119             //drawContours(imgOrig, contours_poly, i,
00120             //      Scalar(255, 0, 0), 4, 8);
00121             int area = bounds[i].width * bounds[i].height;
00122
00123             if(area > maxArea) {
00124                 offset = abs(640 - (bounds[i].tl().x + bounds[i].width / 2));
00125                 maxArea = area;
00126             }
00127         }
00128
00129         cout << "Block area: " << maxArea << " pixels\t\t"
```

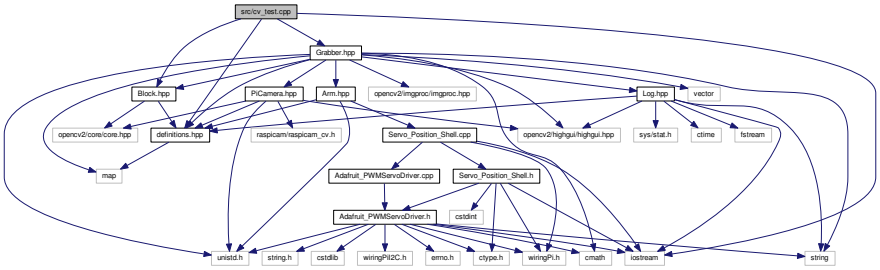
```
00130             << "Center offset: " << offset << endl;
00131         imshow(winContours, imgOrig);  // show original image with rectangles
00132         waitKey(50);  // has to be here :(
00133     }
00134 }
```

6.17 src/cv_test.cpp File Reference

Contains program used to test PiCamera class.

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

Include dependency graph for cv_test.cpp:



Functions

- int main ()

6.17.1 Detailed Description

Contains program used to test PiCamera class.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file cv_test.cpp.

6.17.2 Function Documentation

6.17.2.1 int main ( )

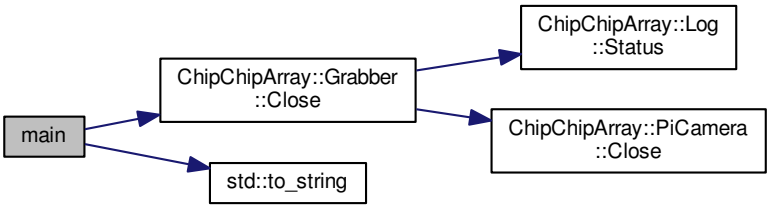
This program was used solely to test the PiCamera wrapper class and its compatibility with the raspicam wrapper and ultimately OpenCV.

Definition at line 19 of file cv_test.cpp.

```
00019           {
00020     Grabber g(Zone::C, Side::Left);
00021     Block block = g.LocateBlock();
00022     g.Close();
00023     std::cout << std::to_string(block.color) << std::endl;
00024     std::cout << "Offset: " << block.offset << std::endl;
00025
00026     Grabber g2(Zone::B, Side::Right);
```

```
00027     Block block2 = g2.LocateBlock();
00028     g2.Close();
00029     std::cout << std::to_string(block2.color) << std::endl;
00030     std::cout << "Offset: " << block2.offset << std::endl;
00031
00032 }
```

Here is the call graph for this function:



6.18 cv_test.cpp

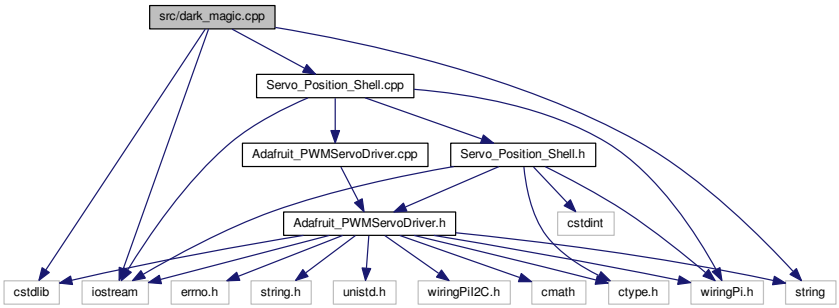
```
00001
00007 #include <iostream>
00008
00009 #include "definitions.hpp"
00010 #include "Block.hpp"
00011 #include "Grabber.hpp"
00012
00013 using namespace ChipChipArray;
00014
00019 int main() {
00020     Grabber g(Zone::C, Side::Left);
00021     Block block = g.LocateBlock();
00022     g.Close();
00023     std::cout << std::to_string(block.color) << std::endl;
00024     std::cout << "Offset: " << block.offset << std::endl;
00025
00026     Grabber g2(Zone::B, Side::Right);
00027     Block block2 = g2.LocateBlock();
00028     g2.Close();
00029     std::cout << std::to_string(block2.color) << std::endl;
00030     std::cout << "Offset: " << block2.offset << std::endl;
00031
00032 }
```

6.19 src/dark_magic.cpp File Reference

Contains test code for arm.

```
#include <cstdlib>
#include <iostream>
#include <string>
#include "Servo_Position_Shell.cpp"
```

Include dependency graph for dark_magic.cpp:



Functions

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

6.19.1 Detailed Description

Contains test code for arm.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file dark_magic.cpp.

6.19.2 Function Documentation

6.19.2.1 int main (int *argc*, char ** *argv*)

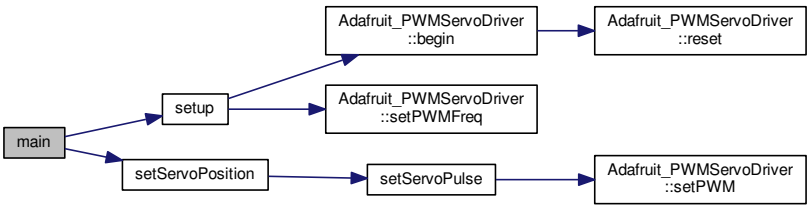
Controls the positions of the arm servos. Will likely work for other servos on the robot, as well.

Usage: arm [SERVO NUMBER] [ANGULAR POSITION DEGREES]

Definition at line 21 of file dark_magic.cpp.

```
00021                                {
00022     setup();
00023 
00024     if(argc != 3) {
00025         cout << "Usage: arm [SERVO] [VALUE]" << endl;
00026     } else {
00027         setServoPosition((Servo)atoi(argv[1]), atoi(argv[2]));
00028     }
00029 }
```

Here is the call graph for this function:



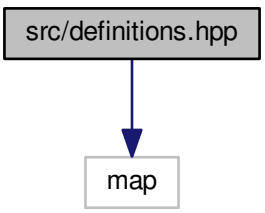
6.20 dark_magic.cpp

```
00001
00007 #include <cstdlib>
00008 #include <iostream>
00009 #include <string>
00010
00011 #include "Servo_Position_Shell.cpp"
00012
00013 using namespace std;
00014
00021 int main(int argc, char** argv) {
00022     setup();
00023
00024     if(argc != 3) {
00025         cout << "Usage: arm [SERVO] [VALUE]" << endl;
00026     } else {
00027         setServoPosition((Servo)atoi(argv[1]), atoi(argv[2]));
00028     }
00029 }
```

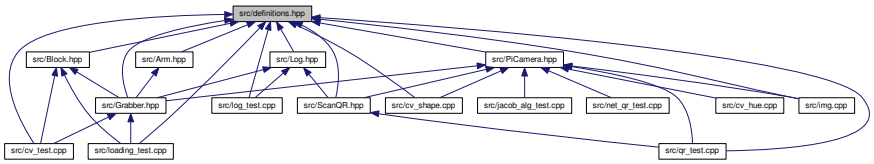
6.21 src/definitions.hpp File Reference

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

`#include <map>`
Include dependency graph for definitions.hpp:



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



Namespaces

- std

Macros

- #define ENUM signed char
- #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
- typedef std::map< Zone, std::map< BlockPosition, uint8 > > PosMap

Enumerations

- enum BlockPosition : ENUM { BlockPosition::Front, BlockPosition::Back, BlockPosition::Middle }
- enum Color : ENUM { Color::Red, Color::Yellow, Color::Green, Color::Blue, Color::Perrywinkle }
- enum Layer : ENUM { Layer::Top, Layer::Bottom }
- enum LogMode : ENUM { LogMode::Text, LogMode::Multi }
- enum Result : ENUM { Result::NoBlocks = -1, Result::NoHalves = 0, Result::TwoHalves = 2, Result::FourHalves = 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)

6.21.1 Detailed Description

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 in file definitions.hpp.

6.21.2 Macro Definition Documentation

6.21.2.1 #define ENUM signed char

Definition at line 12 of file definitions.hpp.

6.21.2.2 #define ERROR -1

Definition at line 13 of file definitions.hpp.

6.21.3 Typedef Documentation

6.21.3.1 typedef unsigned char byte

Definition at line 17 of file definitions.hpp.

6.21.3.2 typedef float float32

Definition at line 30 of file definitions.hpp.

6.21.3.3 typedef double float64

Definition at line 31 of file definitions.hpp.

6.21.3.4 typedef std::map<Zone, std::map<BlockPosition, uint8> > PosMap

Definition at line 101 of file definitions.hpp.

6.21.3.5 typedef signed short sint16

Definition at line 22 of file definitions.hpp.

6.21.3.6 typedef signed int sint32

Definition at line 25 of file definitions.hpp.

6.21.3.7 typedef signed long long sint64

Definition at line 28 of file definitions.hpp.

6.21.3.8 typedef signed char sint8

Definition at line 19 of file definitions.hpp.

6.21.3.9 typedef unsigned short uint16

Definition at line 21 of file definitions.hpp.

6.21.3.10 typedef unsigned int uint32

Definition at line 24 of file definitions.hpp.

6.21.3.11 typedef unsigned long long uint64

Definition at line 27 of file definitions.hpp.

6.21.3.12 typedef unsigned char uint8

Definition at line 18 of file definitions.hpp.

6.21.4 Enumeration Type Documentation

6.21.4.1 enum BlockPosition : ENUM [strong]

The position of the block relative to the arm.

Enumerator

Front

Back

Middle

Definition at line 37 of file definitions.hpp.

```
00037                           : ENUM {
00038     Front,
00039     Back,
00040     Middle
00041 };
```

6.21.4.2 enum Color : ENUM [strong]

The color of a block or train car. Perrywinkle represents all colors.

Enumerator

Red

Yellow

Green

Blue

Perrywinkle

Definition at line 46 of file definitions.hpp.

```
00046                 : ENUM {
00047     Red,
00048     Yellow,
00049     Green,
00050     Blue,
00051     Perrywinkle  // essentially, no color
00052 };
```

6.21.4.3 enum Layer : ENUM [strong]

Defines the location of a block in its stack.

Enumerator

Top

Bottom

Definition at line 57 of file definitions.hpp.

```
00057                 : ENUM {
00058     Top,
00059     Bottom
00060 };
```

6.21.4.4 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 66 of file definitions.hpp.

```
00066                   : ENUM {
00067     Text,
00068     Multi
00069 };
```

6.21.4.5 enum Result : ENUM [strong]

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

Enumerator

NoBlocks

NoHalves

TwoHalves

FourHalves

Definition at line 75 of file definitions.hpp.

```
00075                   : ENUM {
00076     NoBlocks = -1,
00077     NoHalves = 0,
00078     TwoHalves = 2,
00079     FourHalves = 4
00080 };
```

6.21.4.6 enum Side : ENUM [strong]

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

Enumerator

Left

Right

Definition at line 83 of file definitions.hpp.

```
00083                 : ENUM {
00084     Left,
00085     Right
00086 };
```

6.21.4.7 enum Size : ENUM [strong]

The block size, either 2.5" or 5".

Enumerator

Short

Long

Definition at line 89 of file definitions.hpp.

```
00089                 : ENUM {
00090     Short,
00091     Long
00092 };
```

6.21.4.8 enum Zone : ENUM [strong]

Zone A, B, or C

Enumerator

A

B

C

Definition at line 95 of file definitions.hpp.

```
00095                 : ENUM {
00096     A = 'A',
00097     B = 'B',
00098     C = 'C'
00099 };
```

6.22 definitions.hpp

```
00001
00009 #ifndef definitions_H
00010 #define definitions_H
00011
00012 #define ENUM signed char
00013 #define ERROR -1
00014
00015 #include <map>
00016
00017 typedef unsigned char byte;
00018 typedef unsigned char uint8;
00019 typedef signed char sint8;
00020
00021 typedef unsigned short uint16;
00022 typedef signed short sint16;
00023
00024 typedef unsigned int uint32;
00025 typedef signed int sint32;
00026
00027 typedef unsigned long long uint64;
00028 typedef signed long long sint64;
00029
00030 typedef float float32;
00031 typedef double float64;
00032
00033
00037 enum class BlockPosition : ENUM {
00038     Front,
00039     Back,
00040     Middle
00041 };
00042
00046 enum class Color : ENUM {
00047     Red,
00048     Yellow,
00049     Green,
00050     Blue,
00051     Perrywinkle  // essentially, no color
00052 };
00053
00057 enum class Layer : ENUM {
00058     Top,
00059     Bottom
00060 };
00061
00066 enum class LogMode : ENUM {
00067     Text,
00068     Multi
00069 };
00070
00075 enum class Result : ENUM {
00076     NoBlocks = -1,
00077     NoHalves = 0,
00078     TwoHalves = 2,
00079     FourHalves = 4
00080 };
00081
00083 enum class Side : ENUM {
00084     Left,
00085     Right
00086 };
00087
00089 enum class Size : ENUM {
00090     Short,
00091     Long
00092 };
00093
```

```
00095 enum class Zone : ENUM {
00096     A = 'A',
00097     B = 'B',
00098     C = 'C'
00099 };
00100
00101 typedef std::map<Zone, std::map<BlockPosition, uint8>> PosMap;
00102
00103 namespace std {
00107     string to_string(BlockPosition pos) {
00108         if(pos == BlockPosition::Front) return string("Front");
00109         else return string("Back");
00110     }
00111
00115     string to_string(Color color) {
00116         string ret;
00117
00118         switch((ENUM)color) {
00119             case 0:
00120                 ret = "Red";
00121                 break;
00122
00123             case 1:
00124                 ret = "Yellow";
00125                 break;
00126
00127             case 2:
00128                 ret = "Green";
00129                 break;
00130
00131             case 3:
00132                 ret = "Blue";
00133                 break;
00134
00135             case 4:
00136                 ret = "All";
00137                 break;
00138         }
00139
00140         return ret;
00141     }
00142
00146     string to_string(LogMode mode) {
00147         if(mode == LogMode::Multi) return string("Text");
00148         else return string("Multi");
00149     }
00150
00154     string to_string(Result res) {
00155         string ret;
00156
00157         switch((ENUM)res) {
00158             case -1:
00159                 ret = "No Blocks";
00160                 break;
00161
00162             case 0:
00163                 ret = "Two whole, no halves";
00164                 break;
00165
00166             case 2:
00167                 ret = "Two whole, two halves";
00168                 break;
00169
00170             case 4:
00171                 ret = "No whole, four halves";
00172                 break;
00173         }
00174
00175         return ret;
00176     }
00177
00181     string to_string(Side side) {
00182         if(side == Side::Left) return string("Left");
00183         else return string("Right");
00184     }
00185
00189     string to_string(Size size) {
00190         if(size == Size::Long) return string("Long");
00191         else return string("Short");
00192     }
00193
00197     string to_string(Zone zone) {
00198         return string(1, (char)zone);
00199     }
00200 }
00201
00202 // OTHER FILES
```

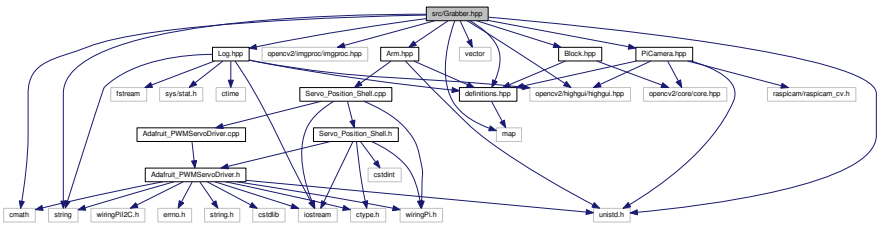
```
00203
00223 #endif
```

6.23 src/Grabber.hpp File Reference

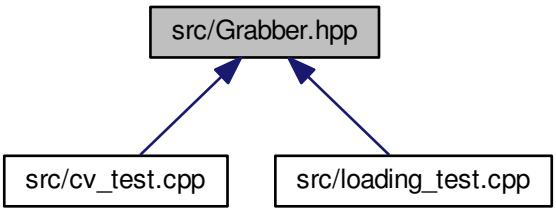
Contains the Grabber class.

```
#include <cmath>
#include <map>
#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 dependency graph for Grabber.hpp:



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



Classes

- class ChipChipArray::Grabber

Namespaces

- ChipChipArray

6.23.1 Detailed Description

Contains the Grabber class.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file Grabber.hpp.

6.24 Grabber.hpp

```
00001
00007 #ifndef Grabber_H
00008 #define Grabber_H
00009
00010 #include <cmath>
00011 #include <map>
00012 #include <opencv2/highgui/highgui.hpp>
00013 #include <opencv2/imgproc/imgproc.hpp>
00014 #include <string>
00015 #include <unistd.h>
00016 #include <vector>
00017
00018 #include "Arm.hpp"
00019 #include "definitions.hpp"
00020 #include "Block.hpp"
00021 #include "Log.hpp"
00022 #include "PiCamera.hpp"
00023
00024 namespace ChipChipArray {
00025
00030     class Grabber {
00031         public:
00043             Grabber(Zone zone, Side side);
00044
00049             void Close();
00050
00051
00058             Result Load();
00059
00060         protected:
00064             PiCamera cam;
00065
00071             Side side;
00072
00076             Zone zone;
00077
00081             void Deposit(Color color = Color::Blue);
00082
00087             void Extend();
00088
00099             Block LocateBlocks(Color color =
      Color::Perrywinkle);
00100
00106             Block LocateBlueBlock();
00107
00108         private:
00112             static const uint32 MIN_HALF_BLOCK_SIZE = 50000;
00113
00118             static Log log;
00119
00124             uint8 invokeCount = 0;
00125
00130             std::map<Color, std::array<cv::Scalar, 2>> rangeVals;
00131
00135             Arm arm;
00136     };
00137
00141     Log Grabber::log("logs/Grabber", LogMode::Multi);
00142
00143     Grabber::Grabber(Zone zone, Side side) {
00144         log.Status("Opening Grabber");
00145         log.Verbose("Zone: " + std::to_string(zone));
00146         log.Verbose("Side: " + std::to_string(side));
00147
00148         this->zone = zone;
00149         this->side = side;
00150
00151         log.Verbose("Setting HSV threshold values");
```

```
00152
00153         rangeVals[Color::Red] = { cv::Scalar(0, 20, 60),
00154             cv::Scalar(12, 255, 255) };
00155         rangeVals[Color::Green] = { cv::Scalar(49, 41, 17),
00156             cv::Scalar(63, 255, 255) };
00157         rangeVals[Color::Blue] = { cv::Scalar(70, 0, 0),
00158             cv::Scalar(100, 255, 255) };
00159
00160         /* Remember, we're only pretending this color's image is in HSV space.
00161          * It's really in YUV, as required by Jacob yellow-detection algorithm. */
00162         rangeVals[Color::Yellow] = { cv::Scalar(0, 0, 0),
00163             cv::Scalar(255, 255, 20)};
00164     }
00165
00166     void Grabber::Close() {
00167         log.Status("Closing Grabber");
00168         cam.Close();
00169     }
00170
00171     void Grabber::Deposit(Color color) {
00172         if(color == Color::Blue) {
00173             arm.ClawClose();
00174             sleep(1);
00175             arm.BaseTilt(160);
00176             sleep(1);
00177             arm.Elbow(130);
00178             sleep(1);
00179             arm.BaseTurn(47);
00180             sleep(1);
00181             arm.ClawOpen();
00182             sleep(1);
00183         } else {
00184             throw std::runtime_error("Du Idiot! Die Armbewegungen für diese "
00185                     "Farbe sind noch nicht implementiert. Vielleicht sollst "
00186                     "du die englische Phrase lernen 'Would you like fries "
00187                     "with that?");
00188         }
00189     }
00190
00191     void Grabber::Extend() {
00192         arm.Elbow(180);
00193         usleep(500000);
00194         arm.BaseTurn(132);
00195         arm.BaseTilt(125);
00196         arm.Elbow(150);
00197         arm.WristTwist(90);
00198         arm.ClawOpen();
00199         sleep(2);
00200     }
00201
00202     Result Grabber::Load() {
00203         for(uint8 i = 0; i < 2; i++) {
00204             Extend();
00205
00206             try {
00207                 Block block = (zone == Zone::A)
00208                     ? LocateBlocks(Color::Blue) :
      LocateBlueBlock();
00209
00210                 float32 baseKonstant = 0.5;
00211                 if(block.dRightLeft > 0) baseKonstant *= -1;
00212                 float32 degree = baseKonstant * std::sqrt(block.
      dRightLeft);
00213                 arm.dBaseTurn(degree);
00214                 arm.dWristTwist(-degree);
00215                 sleep(1);
00216                 arm.BaseTilt(140);
00217                 sleep(1);
00218
00219                 uint8 bend = (i == 0 ? 100 : 90);
00220
00221                 // lower claw over block
00222                 for(uint8 j = 140; j >= bend; j -= 10) {
00223                     arm.Elbow(j);
00224                     sleep(1);
00225                 }
00226
00227                 // deposit in bin
00228                 sleep(1);
00229                 Deposit();
00230             } catch(std::exception ex) {
00231                 log.Error(std::string("An exception occured attempting "
00232                         "to load the blocks in function Grabber::Load(): ")
00233                         + ex.what());
00234             }
00235
00236
```

```
00237             if(i == 0) {
00238                 arm.BaseTurn(132);
00239             } else {
00240                 arm.BaseTurn(135);
00241                 sleep(1);
00242                 arm.BaseTilt(180);
00243                 sleep(1);
00244                 arm.Elbow(90);
00245                 sleep(1);
00246                 arm.Elbow(45);
00247                 sleep(1);
00248                 arm.Elbow(0);
00249             }
00250         }
00251     }
00252
00253
00254     Block Grabber::LocateBlocks(Color color) {
00255         invokeCount++;
00256         std::string logstr = "Locating blocks";
00257
00258         if(color == Color::Perrywinkle) {
00259             logstr += " (" + std::to_string(color) + ")";
00260         }
00261
00262         log.Verbose(logstr);
00263
00264         cv::Mat imgOrig;
00265         cv::transpose(cam.Snap(), imgOrig);
00266
00267         cv::Mat imgHSV;
00268         cv::Mat imgThresh;
00269         std::vector<cv::Rect> blocks;
00270         std::vector<Color> colors;
00271
00272         uint8 loopNum = (color == Color::Perrywinkle ? rangeVals.size() : 1);
00273
00274         for(int i = 0; i < loopNum; i++) {
00275             if(loopNum > 1) {
00276                 switch(i) {
00277                     case 0:
00278                         color = Color::Red;
00279                         break;
00280
00281                     case 1:
00282                         color = Color::Green;
00283                         break;
00284
00285                     case 2:
00286                         color = Color::Blue;
00287                         break;
00288
00289                         /* Must be last, because it changes imgHSV from HSV space
00290                          * to YUV space. */
00291                     case 3:
00292                         color = Color::Yellow;
00293                         break;
00294                 }
00295             }
00296
00297             log.Verbose("Searching: " + std::to_string(color));
00298
00299             if(color == Color::Yellow) {
00300                 cv::Mat temp;
00301                 imgOrig.copyTo(temp);
00302                 cv::cvtColor(imgOrig, imgHSV, CV_BGR2YUV);
00303                 cv::cvtColor(imgHSV, temp, CV_HSV2BGR);
00304                 cv::cvtColor(temp, imgHSV, cv::COLOR_BGR2HSV);
00305                 log.Image(temp, "yuv_yellow_" + std::to_string(color)
00306                         + "_" + std::to_string(zone)
00307                         + std::to_string(invokeCount)
00308                         + ".bmp");
00309             } else {
00310                 cv::cvtColor(imgOrig, imgHSV, cv::COLOR_BGR2HSV);
00311             }
00312
00313             cv::inRange(imgHSV, rangeVals[color][0],
00314                     rangeVals[color][1], imgThresh);
00315
00316             /*
00317              * Not quite sure what all this does, but it seems to
00318              * relate to smoothing the image
00319              */
00320             cv::erode(imgThresh, imgThresh,
00321                     cv::getStructuringElement(
00322                         cv::MORPH_ELLIPSE,
00323                         cv::Size(5, 5)));
```

```
00324             cv::dilate(imgThresh, imgThresh,
00325                     cv::getStructuringElement(
00326                         cv::MORPH_ELLIPSE,
00327                         cv::Size(5, 5)));
00328             cv::dilate(imgThresh, imgThresh,
00329                     cv::getStructuringElement(
00330                         cv::MORPH_ELLIPSE,
00331                         cv::Size(5, 5)));
00332             cv::erode(imgThresh, imgThresh,
00333                     cv::getStructuringElement(
00334                         cv::MORPH_ELLIPSE,
00335                         cv::Size(5, 5)));
00336 
00337             log.Image(imgThresh, "thresh_" + std::to_string(color)
00338                     + "_" + std::to_string(zone)
00339                     + std::to_string(invokeCount)
00340                     + ".bmp");
00341 
00342             // calculate contours
00343             std::vector<std::vector<cv::Point>> contours;
00344             cv::findContours(imgThresh, contours, CV_RETR_TREE,
00345                     CV_CHAIN_APPROX_SIMPLE,
00346                     cv::Point(0, 0));
00347             std::vector<std::vector<cv::Point>>
00348                 contours_poly(contours.size());
00349             std::vector<cv::Rect> bounds(contours.size());
00350 
00351             // find rectangle around polygon-ish shapes
00352             for(int i = 0; i < contours.size(); i++) {
00353                 uint32 area = cv::contourArea(contours[i]);
00354 
00355                 // determine if block and add to blocks vector
00356                 if(area > MIN_HALF_BLOCK_SIZE) {
00357                     cv::approxPolyDP(cv::Mat(contours[i]),
00358                             contours_poly[i], 20,
00359                             false);
00360                     cv::Rect rect = cv::boundingRect(
00361                             cv::Mat(contours_poly[i]));
00362                     log.Debug(std::to_string(color)
00363                             + " block detected "
00364                             "with area "
00365                             + std::to_string(
00366                                 area));
00367                     blocks.push_back(rect);
00368                     colors.push_back(color);
00369                 }
00370             }
00371         }
00372 
00373         if(blocks.size() == 0) {
00374             log.Image(imgOrig, "original_" + std::to_string(
      zone)
00375                     + std::to_string(invokeCount)
00376                     + "_no_blocks.bmp");
00377             throw std::runtime_error("No blocks found!");
00378         } else {
00379             log.Status(std::to_string(blocks.size())
00380                     + " blocks found");
00381         }
00382 
00383         // coordinates start in top right
00384         Block block = Block(blocks[0], colors[0]);
00385 
00386         if(blocks.size() > 1) {
00387             for(int i = 1; i < blocks.size(); i++) {
00388                 if((side == Side::Right && blocks[i].x
00389                         > block.topLeft.x)
00390                     || (side == Side::Left
00391                         && blocks[i].x
00392                         < block.topLeft.x)) {
00393                     block = Block(blocks[i], colors[i]);
00394                 }
00395             }
00396         }
00397 
00398         log.Status(std::to_string(block.color) + " block is located");
00399 
00400         log.Debug("Block properties => area: " + std::to_string(block.
      area)
00401                 + ", height: " + std::to_string(block.height) + ", width: "
00402                 + std::to_string(block.width) + ", offset: "
00403                 + std::to_string(block.offset) + ", color: "
00404                 + std::to_string(block.color) + ", size: "
00405                 + std::to_string(block.size));
00406 
00407         /*
00408          * Draw surrounding rectangles from above on original
```

```
00409          * image.
00410          */
00411         cv::rectangle(imgOrig, block.topLeft , block.bottomRight,
00412                 cv::Scalar(255, 0, 0), 4, 8);
00413         log.Image(imgOrig, "original_" + std::to_string(zone)
00414                 + std::to_string(invokeCount)
00415                 + ".bmp");
00416
00417         return block;
00418     }
00419
00420     Block Grabber::LocateBlueBlock() {
00421         std::vector<cv::Mat> channels;
00422         std::vector<cv::Rect> blocks;
00423
00424         invokeCount++;
00425         log.Status("Locating blue blocks");
00426
00427         cv::Mat img;
00428         cv::Mat imgThresh;
00429         cv::split(cam.Snap(), channels);
00430         cv::transpose(channels[0], img);
00431
00432         log.Verbose("Searching: Blue block");
00433         cv::inRange(img, 30, 255, imgThresh);
00434         log.Image(imgThresh, "thresh_blue" + std::to_string(
      zone)
00435                 + std::to_string(invokeCount) + ".bmp");
00436
00437         // calculate contours
00438         std::vector<std::vector<cv::Point>> contours;
00439         cv::findContours(imgThresh, contours, CV_RETR_TREE,
00440                 CV_CHAIN_APPROX_SIMPLE,
00441                 cv::Point(0, 0));
00442         std::vector<std::vector<cv::Point>>
00443             contours_poly(contours.size());
00444         std::vector<cv::Rect> bounds(contours.size());
00445
00446         // find rectangle around polygon-ish shapes
00447         for(int i = 0; i < contours.size(); i++) {
00448             uint32 area = cv::contourArea(contours[i]);
00449
00450             // determine if block and add to blocks vector
00451             if(area > MIN_HALF_BLOCK_SIZE) {
00452                 cv::approxPolyDP(cv::Mat(contours[i]),
00453                         contours_poly[i], 20,
00454                         false);
00455                 cv::Rect rect = cv::boundingRect(
00456                         cv::Mat(contours_poly[i]));
00457                 log.Debug("Blue block detected with area "
00458                         + std::to_string(area));
00459                 blocks.push_back(rect);
00460             }
00461         }
00462
00463
00464         if(blocks.size() == 0) {
00465             log.Image(img, "original_" + std::to_string(zone)
00466                     + std::to_string(invokeCount)
00467                     + "_no_blocks.bmp");
00468             throw std::runtime_error("No blocks found!");
00469         } else {
00470             log.Status(std::to_string(blocks.size())
00471                     + " blocks found");
00472         }
00473
00474         // coordinates start in top right
00475         Block block = Block(blocks[0], Color::Blue);
00476
00477         if(blocks.size() > 1) {
00478             for(int i = 1; i < blocks.size(); i++) {
00479                 if((side == Side::Right && blocks[i].x
00480                         > block.topLeft.x)
00481                         || (side == Side::Left
00482                             && blocks[i].x
00483                             < block.topLeft.x)) {
00484                     block = Block(blocks[i], Color::Blue);
00485                 }
00486             }
00487         }
00488
00489         log.Status(std::to_string(block.color) + " block is located");
00490
00491         log.Debug("Block properties => area: " + std::to_string(block.
      area)
00492                 + ", height: " + std::to_string(block.height) + ", width: "
00493                 + std::to_string(block.width) + ", offset: "
```

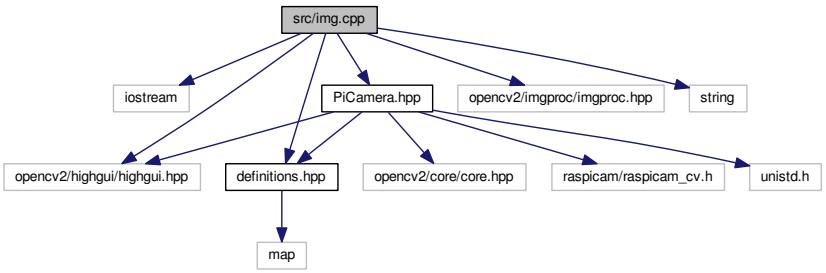
```
00494                 + std::to_string(block.offset) + ", color: "
00495                 + std::to_string(block.color) + ", size: "
00496                 + std::to_string(block.size));
00497
00498         /*
00499          * Draw surrounding rectangles from above on original
00500          * image.
00501          */
00502         cv::rectangle(img, block.topLeft , block.bottomRight,
00503                 cv::Scalar(255, 0, 0), 4, 8);
00504         log.Image(img, "original_" + std::to_string(zone)
00505                 + std::to_string(invokeCount)
00506                 + ".bmp");
00507
00508         return block;
00509     }
00510 }
00511
00512 #endif
```

6.25 src/img.cpp File Reference

Contains a program to display the current camera image.

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

Include dependency graph for img.cpp:



Functions

- int main ()

6.25.1 Detailed Description

Contains a program to display the current camera image.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file img.cpp.

6.25.2 Function Documentation

6.25.2.1 int main ( )

This program displays the current camera image.

Definition at line 22 of file img.cpp.

```
00022           {
00023     string window = "Current";
00024     PiCamera cam;
00025
00026     namedWindow(window, CV_WINDOW_NORMAL | CV_WINDOW_KEEPRATIO);
00027
00028     while(true) {
00029         Mat image;
00030         transpose(cam.Snap(), image);
00031         imshow(window, image);
00032         waitKey(10);  // has to be here :(
00033     }
00034 }
```

Here is the call graph for this function:

6.26 img.cpp

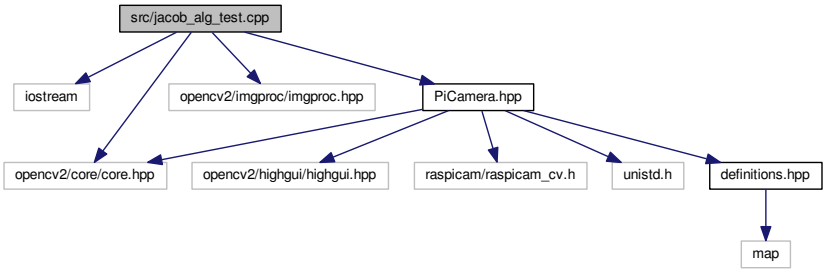
```
00001
00007 #include <iostream>
00008 #include <opencv2/highgui/highgui.hpp>
00009 #include <opencv2/imgproc/imgproc.hpp>
00010 #include <string>
00011
00012 #include "definitions.hpp"
00013 #include "PiCamera.hpp"
00014
00015 using namespace ChipChipArray;
00016 using namespace cv;
00017 using namespace std;
00018
00022 int main() {
00023     string window = "Current";
00024     PiCamera cam;
00025
00026     namedWindow(window, CV_WINDOW_NORMAL | CV_WINDOW_KEEPRATIO);
00027
00028     while(true) {
00029         Mat image;
00030         transpose(cam.Snap(), image);
00031         imshow(window, image);
00032         waitKey(10);  // has to be here :(
00033     }
00034 }
```

6.27 src/jacob_alg_test.cpp File Reference

Contains a program that tests Jacob's yellow-detection algorithm.

```
#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 ()

6.27.1 Detailed Description

Contains a program that tests Jacob's yellow-detection algorithm.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file jacob_alg_test.cpp.

6.27.2 Function Documentation

6.27.2.1 int main ( )

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

Definition at line 21 of file jacob_alg_test.cpp.

```
00021           {
00022     PiCamera cam;
00023     namedWindow("window", CV_WINDOW_NORMAL);
00024     Mat orig = cam.Snap();
00025     Mat yuv;
00026     Mat fin;
00027     cvtColor(orig, yuv, CV_BGR2YUV);
00028     cvtColor(yuv, fin, CV_HSV2BGR);
00029     imshow("window", fin);
00030     waitKey(-1);
00031 }
```

Here is the call graph for this function:

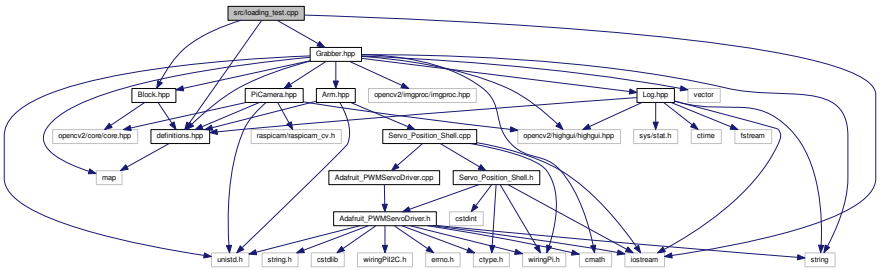
6.28 jacob_alg_test.cpp

```
00001
00007 #include <iostream>
00008 #include <opencv2/core/core.hpp>
00009 #include <opencv2/imgproc/imgproc.hpp>
00010
00011 #include "PiCamera.hpp"
00012
00013 using namespace std;
00014 using namespace cv;
00015 using namespace ChipChipArray;
00016
00021 int main() {
00022     PiCamera cam;
00023     namedWindow("window", CV_WINDOW_NORMAL);
00024     Mat orig = cam.Snap();
00025     Mat yuv;
00026     Mat fin;
00027     cvtColor(orig, yuv, CV_BGR2YUV);
00028     cvtColor(yuv, fin, CV_HSV2BGR);
00029     imshow("window", fin);
00030     waitKey(-1);
00031 }
```

6.29 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 ()

6.29.1 Function Documentation

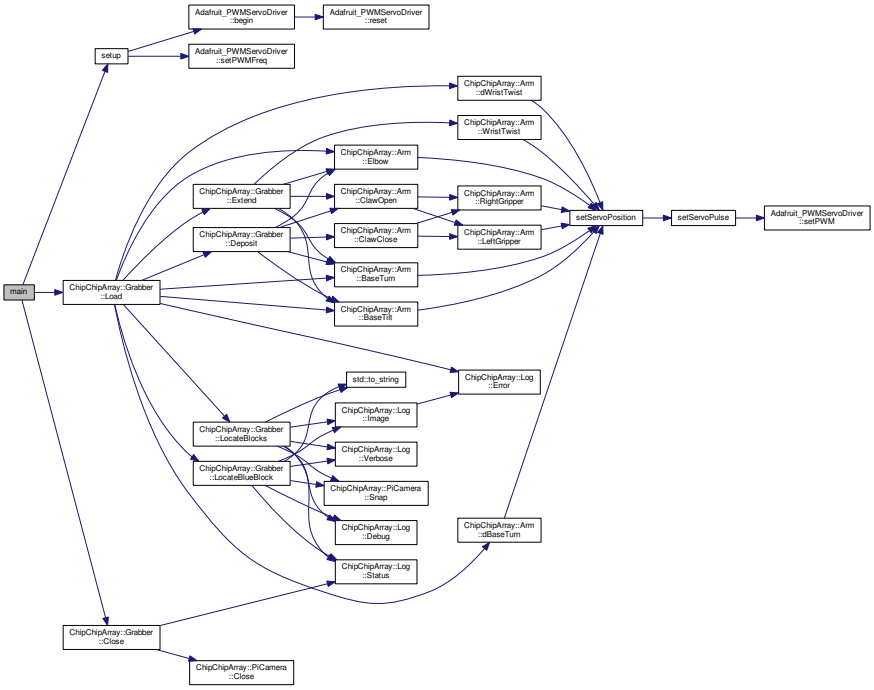
6.29.1.1 int main ( )

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

Definition at line 19 of file loading_test.cpp.

```
00019           {
00020     setup();
00021     Grabber g(Zone::A, Side::Left);
00022     g.Load();
00023     g.Close();
00024 }
```

Here is the call graph for this function:



6.30 loading_test.cpp

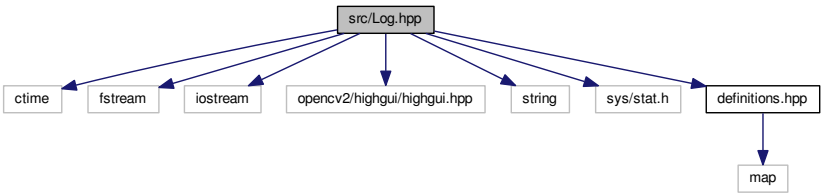
```
00001
00007 #include <iostream>
00008
00009 #include "definitions.hpp"
00010 #include "Block.hpp"
00011 #include "Grabber.hpp"
00012
00013 using namespace ChipChipArray;
00014
00019 int main() {
00020     setup();
00021     Grabber g(Zone::A, Side::Left);
00022     g.Load();
00023     g.Close();
00024 }
```

6.31 src/Log.hpp File Reference

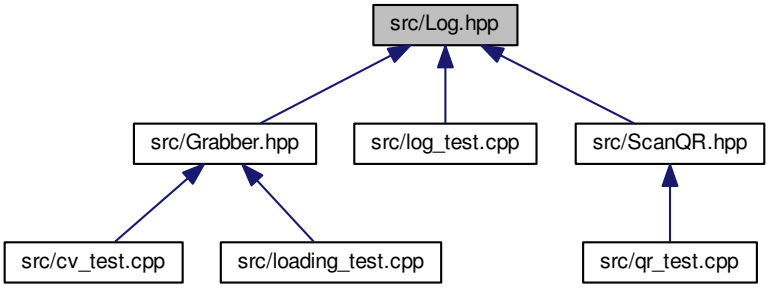
Contains Log class.

```
#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

6.31.1 Detailed Description

Contains Log class.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file Log.hpp.

6.32 Log.hpp

```
00001
00007 #ifndef Log_H
00008 #define Log_H
00009
00010 #include <ctime>
00011 #include <fstream>
00012 #include <iostream>
00013 #include <opencv2/highgui/highgui.hpp>
00014 #include <string>
00015 #include <sys/stat.h>
00016
00017 #include "definitions.hpp"
00018
00019 namespace ChipChipArray {
00020
00035     class Log {
00036         public:
00041             Log() {};
00042
00058             Log(auto dir, LogMode mode = LogMode::Text);
00059
00063             ~Log();
00064
00074             void Debug(auto mesg);
00075
00082             void Error(auto mesg);
00083
00092             void Image(cv::Mat image, auto filename);
00093
00097             void Open(auto dir, LogMode mode = LogMode::Text);
00098
00107             void Status(auto mesg);
00108
00117             void Variable(auto name, auto value);
00118
00127             void Verbose(auto mesg);
00128
00129         private:
00130             // CONSTANTS
00131
00136             const char PATH_SEP = '/';
00137
00142             static const uint8 LEN = 255;
00143
00144
00145             // OTHER VARIABLES
00146
00151             std::string dir;
00152
00156             std::string filename;
00157
00162             uint8 imgCount = 0;
00163
00167             LogMode mode;
00168
00172             std::ofstream file;
00173
00174
00175             // FUNCTIONS
00176
00184             void LogError(auto mesg, std::ofstream::failure f);
00185     };
00186
00187     Log::Log(auto dir, LogMode mode) {
00188 #ifdef LOG
00189         Open(dir, mode);
00190 #endif
00191     }
00192
00193     Log::~Log() {
00194 #ifdef LOG
00195         try {
00196             file.flush();
00197             file.close();
```

```
00198         } catch (std::ofstream::failure f) {
00199             LogError("Gosh dang it! A fatal error has occured "
00200                     "closing the logfile.", f);
00201         }
00202 #endif
00203     }
00204
00205     void Log::Debug(auto mesg) {
00206 #ifdef LOG
00207         try {
00208             file << "DEBUG: " << mesg << std::endl;
00209             file.flush();
00210         } catch(std::ofstream::failure f) {
00211             LogError("Debug() write error", f);
00212         }
00213 #endif
00214     }
00215
00216     void Log::Error(auto mesg) {
00217 #ifdef LOG
00218         try {
00219             file << "ERROR: " << mesg << std::endl;
00220             file.flush();
00221         } catch(std::ofstream::failure f) {
00222             LogError("Error() write error", f);
00223         }
00224 #endif
00225     }
00226
00227     void Log::Image(cv::Mat image, auto filename) {
00228 #ifdef LOG
00229         try {
00230             cv::imwrite(dir + std::string(filename), image);
00231         } catch(std::ofstream::failure f) {
00232             LogError("Image() write error", f);
00233         } catch(std::exception ex) {
00234             Error("Error writing image " + std::string(filename));
00235         }
00236 #endif
00237     }
00238
00239     void Log::LogError(auto mesg, std::ofstream::failure f) {
00240 #ifdef LOG
00241         std::cerr << mesg << std::endl;
00242         std::cerr << "MESSAGE: " << f.what() << std::endl;
00243         exit(ERROR);
00244 #endif
00245     }
00246
00247     void Log::Open(auto dir, LogMode mode) {
00248 #ifdef LOG
00249         // format date and time
00250         char date[32];
00251         time_t sec = time(nullptr);
00252         struct tm * loctime = localtime(&sec);
00253         strftime(date, 32, "%m-%d_%H-%M-%S", loctime);
00254
00255         // create temperary strings
00256         this->dir = std::string(dir);
00257         std::string datestr = std::string(date);
00258
00259         // add path separator if necessary
00260         if(this->dir[this->dir.length() - 1] != PATH_SEP) {
00261             this->dir += PATH_SEP;
00262         }
00263
00264         // add directory for log and images if necessary
00265         if(mode == LogMode::Multi) this->dir += datestr + PATH_SEP;
00266
00267         int ret = mkdir(this->dir.c_str(), S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP
00268                 | S_IROTH | S_IWOTH | S_IXUSR | S_IXGRP
00269                 | S_IXOTH);
00270
00271         filename = this->dir + datestr + ".log";
00272
00273         // set class mode
00274         this->mode = mode;
00275
00276         // Initializing file
00277         file.exceptions(std::ofstream::failbit
00278                 | std::ofstream::badbit);
00279
00280         try {
00281             file.open(filename, std::ofstream::out
00282                     | std::ofstream::app);
00283         } catch(std::ofstream::failure ex) {
00284             LogError("Oh, no! An error has occurred opening the "
```

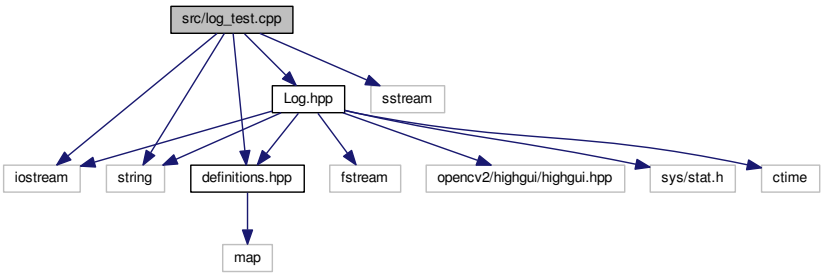
```
00285                     "log file.", ex);
00286         }
00287 #endif
00288     }
00289
00290     void Log::Status(auto mesg) {
00291 #ifdef LOG
00292         try {
00293             file << "STATUS: " << mesg << std::endl;
00294             file.flush();
00295         } catch (std::ofstream::failure f) {
00296             LogError("Status() write error", f);
00297         }
00298 #endif
00299     }
00300
00301     void Log::Variable(auto name, auto value) {
00302 #ifdef LOG
00303         try {
00304             file << "VARIABLE: " << name << " = " << value
00305                 << std::endl;
00306             file.flush();
00307         } catch(std::ofstream::failure f) {
00308             LogError("Variable() write error", f);
00309         }
00310 #endif
00311     }
00312
00313     void Log::Verbose(auto mesg) {
00314 #ifdef LOG
00315         try {
00316             file << "VERBOSE: " << mesg << std::endl;
00317             file.flush();
00318         } catch(std::ofstream::failure f) {
00319             LogError("Verbose() write error", f);
00320         }
00321 #endif
00322     }
00323 }
00324 #endif
```

6.33 src/log_test.cpp File Reference

contains a program to test partially the Log class

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

Include dependency graph for log_test.cpp:



Functions

- int main ()

6.33.1 Detailed Description

contains a program to test partially the Log class

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file log_test.cpp.

6.33.2 Function Documentation

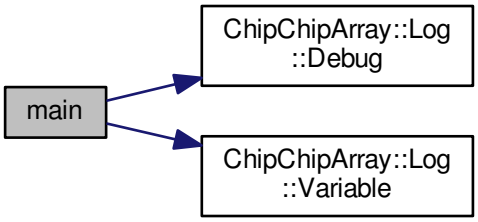
6.33.2.1 int main ( )

This program partially tests the Log class.

Definition at line 18 of file log_test.cpp.

```
00018           {
00019     Log log("logs", LogMode::Text);
00020     log.Debug("yolo!");
00021     log.Variable("mymom","toldme");
00022 }
```

Here is the call graph for this function:



6.34 log_test.cpp

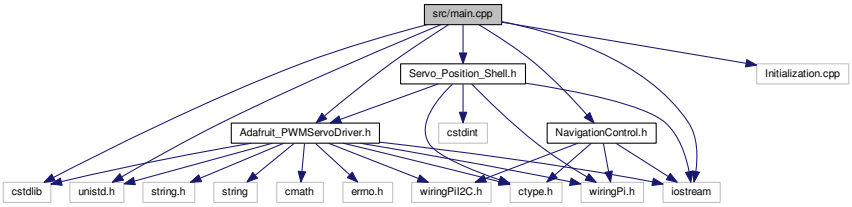
```
00001
00007 #include <iostream>
00008 #include <sstream>
00009 #include <string>
00010
00011 #include "definitions.hpp"
00012 #include "Log.hpp"
00013
00014 using namespace ChipChipArray;
00018 int main() {
00019     Log log("logs", LogMode::Text);
00020     log.Debug("yolo!");
00021     log.Variable("mymom","toldme");
00022 }
```

6.35 src/main.cpp File Reference

Contains the main() function to the whole project.

```
#include <cstdlib>
#include <unistd.h>
#include "Servo_Position_Shell.h"
#include "Adafruit_PWMServoDriver.h"
#include <iostream>
#include "NavigationControl.h"
#include "Initialization.cpp"
```

Include dependency graph for main.cpp:



Macros

- #define READTHINKDO

Functions

- int main ()

Variables

- int startstatus
- int startroboth
- int stoproboth
- int track
- int navigationbusy
- int cleartonavigate
- int cleartoload
- int cleartounload
- int robotismoving
- int whereistherobot
- int loadcounter
- int unloadcounter
- int loadtheblocks
- int unloadtheblocks
- int giveitasec
- int startloadingthread

6.35.1 Detailed Description

Contains the main() function to the whole project.

Author

Nickolas Neely

Date

1. February 2016, 8:20 AM

Definition in file main.cpp.

6.35.2 Macro Definition Documentation

6.35.2.1 #define READTHINKDO

Definition at line 27 of file main.cpp.

6.35.3 Function Documentation

6.35.3.1 int main ( )

The main() project program function.

Definition at line 51 of file main.cpp.

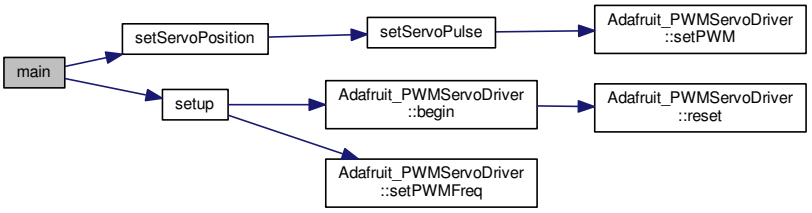
```
00051           {
00052
00053 initialization();
00054 #ifdef READTHINKDO

00055 /*Track selection initialization will allow track select until start condition is true*/
00056 while(1){
00057     while(track == 1 && startroboth == 0){
00058         startstatus = digitalRead(0);
00059         track = digitalRead(1);
00060         cout<<endl<<"I am in left track mode";
00061         if(startstatus == 1){
00062             startroboth = 1;
00063         }
00064
00065         /* RUN MODE FOR LEFT TRACK GAME START!*/
00066         while(startroboth == 1 && stoproboth == 0){
00067  //READ for left track
00068
00069             startstatus = digitalRead(0);
00070             navigationbusy = digitalRead(2);
00071
00072 //THINK for left track
00073
00074             //clearance to do an action;
00075             if(cleartoload == 1 || cleartounload == 1){
00076                 cleartonavigate = 0;
00077             }
00078             else if(cleartonavigate == 1 || cleartounload == 1){
00079                 cleartoload = 0;
00080             }
00081             else if(cleartonavigate == 1 || cleartoload == 1){
00082                 cleartounload = 0;
00083             }
00084
00085
00086             // latch to figure out robot position in sequence
00087             if(navigationbusy == 1 && robotismoving == 0){
00088                 robotismoving = 1;
00089                 whereistherobot++;
00090             }else if(navigationbusy == 0 && robotismoving == 1){
00091                 robotismoving = 0;
00092             }
00093
00094             //working actions
00095             if(whereistherobot > 0 && whereistherobot <=4 &&
      navigationbusy == 0){
00096                 cleartoload = 1;
00097             }else if(whereistherobot > 0 && whereistherobot <=4 &&
      navigationbusy == 1){
00098                 cleartoload = 0;
00099             }else if(whereistherobot == 5 && navigationbusy == 0){
00100                 cleartounload = 1;
00101             }else if(whereistherobot == 5 && navigationbusy == 1){
00102                 cleartounload = 0;
00103             }
```

```
00104
00105
00106             //controls load action timing
00107             if(cleartoload == 1 && loadcounter < 2000){
00108                 loadtheblocks = 1;
00109                 loadcounter ++;
00110                 startloadingthread = 1;
00111
00112             }else if(loadcounter >= 2000){
00113                 cleartoload = 0;
00114                 loadcounter = 1;
00115             }
00116
00117             //controls the unload action timing
00118             if(cleartounload ==1 && unloadcounter < 150){
00119                 unloadcounter++;
00120                 unloadtheblocks = 1;
00121             }else if(loadcounter >= 150){
00122                 cleartounload = 0;
00123                 unloadcounter = 1;
00124                 unloadtheblocks = 0;
00125             }
00126
00127 //DO for the left track
00128
00129             //Navigate to next position in sequence
00130             if(cleartounload == 0 && cleartoload == 0 &&
    robotismoving == 0){
00131                 digitalWrite(3,HIGH);
00132             }else if(robotismoving == 1){
00133                 digitalWrite(3,LOW);
00134             }
00135
00136             //load blocks
00137             if(loadtheblocks ==1){
00138                 pthread_t andrewthread;
00139                 pthread_create(&andrewthread,NULL,grabCall,NULL);
00140
00141                 loadtheblocks =0;
00142
00143             }
00144
00145             // unload the blocks
00146             if(unloadtheblocks ==1 && unloadcounter < 100){
00147                 setServoPosition(Servo(10),90);
00148                 setServoPosition(Servo(11),90);
00149                 setServoPosition(Servo(12),90);
00150                 setServoPosition(Servo(13),90);
00151             }else if(unloadtheblocks == 1 && unloadcounter >= 100 &&
    unloadcounter <140){
00152                 setServoPosition(Servo(10),0);
00153                 setServoPosition(Servo(11),0);
00154                 setServoPosition(Servo(12),0);
00155                 setServoPosition(Servo(13),0);
00156             }else if(unloadtheblocks == 1 && unloadcounter >= 140){
00157                 setServoPosition(Servo(10),-1);
00158                 setServoPosition(Servo(11),-1);
00159                 setServoPosition(Servo(12),-1);
00160                 setServoPosition(Servo(13),-1);
00161             }
00162
00163             cout<<endl<<"I am in run mode of left track";
00164
00165             /* stops Robot if ever hit on a RTD loop*/
00166             if(startstatus == 0){
00167                 digitalWrite(3,LOW);
00168                 cout<<endl<<"Halting all function stop engaged after a start";
00169                 giveitasec++;
00170                 if(giveitasec >= 50){
00171                     stoproboth = 1;
00172                     giveitasec = 0;
00173                 }
00174             }
00175             cout<<endl<<"The current position of robot is:"<<whereistherobot<<endl;
00176
00177             // delay for 20 milliseconds per loop
00178             usleep(20000);
00179
00180
00181         }
00182     }
00183     /*Track selection initialization will allow track select until start condition is true*/
00184     while(track == 0 && startroboth ==0){
00185         startstatus = digitalRead(0);
00186         track = digitalRead(1);
00187         if(startstatus == 1){
00188             startroboth = 1;
```

```
00189         }
00190         cout<<endl<<"I am in right track mode.";
00191         cout<<"Not yet implemented. Danger. Danger. Danger.";
00192     }
00193 }
00194 #endif

00195 #ifdef SWITCHTEST

00196     int cat;
00197     int dog;
00198     wiringPiSetup();
00199     while(1){
00200
00201         //test each pin if needed for setup and checking
00202         /*
00203         cout<<endl;

00204         cout<<"Input a pin to check:"<<endl;

00205         cin>>dog;

00206         cat =digitalRead(dog);

00207         cout<<"I am reading:"<<cat<<endl;

00208         */
00209
00210         if(digitalRead(0)==1){
00211             cout<<"Heavy is in stop or idle mode."<<endl;
00212
00213         }else if(digitalRead(0)==0){
00214             cout<<"Heavy is in run mode!"<<endl;
00215         }
00216         if(digitalRead(1)==1){
00217             cout<<"Heavy is set for left track."<<endl;
00218         }else if(digitalRead(1)==0){
00219             cout<<"Heavy is set for right track."<<endl;
00220         }
00221
00222         cout<<endl<<"Press enter to continue";
00223         cin.ignore();
00224     }
00225 #endif

00226
00227 #ifdef ARMTEST

00228
00229     Servo whichservo;
00230     int tmpServo = -1;
00231     int position;
00232     setup();
00233     while(1){
00234         cout<<endl;
00235         cout<<"Pick a servo to use: BASE_TURN = 0, BASE_TILT = 1, ELBOW = 2, WRIST_TILT = 3,";
00236         cout<<endl<<"WRIST_PAN = 4, GRIP_LEFT = 5, GRIP_RIGHT = 6";
00237         cout<<endl;
00238         cin>>tmpServo;
00239         if(tmpServo > 6 || tmpServo < 0){
00240             cout<<"Please choose again:"<<endl;
00241             continue;
00242         }
00243         whichservo = (Servo)tmpServo;
00244         cout<<endl;
00245         cout<<"Pick a position (set position to -1 to disengage servo and set pwm to 0):";
00246         cin>>position;
00247         cout<<endl;
00248         setServoPosition(whichservo,position);
00249
00250     }
00251 #endif

00252
00253 #ifdef NAVTEST

00254
00255     int cat = 0;
00256     pinMode(2,INPUT);
00257     while(1){
00258         cout<<"High or Low?:"<<endl;
00259         cin>>cat;
00260         digitalWrite(3,cat);
00261         cout<<endl;
00262         if(digitalRead(2)==1){
00263             cout<<"I am getting a high from Micah"<<endl;
00264
```

```
00265         }else if(digitalRead(2)==0){
00266             cout<<"I am getting a low from Micah"<<endl;
00267         }
00268     }
00269     cin.ignore();
00270
00271
00272
00273 #endif

00274
00275 #ifdef UNLOADTEST

00276
00277     Servo whichservo;
00278     int tmpServo = -1;
00279     int position;
00280
00281     setup();
00282     while(1){
00283         /*
00284         cout<<endl;

00285         cout<<"Pick a servo to use: ";

00286         cout<<endl<<"UNLOAD_1= 10, UNLOAD_2 = 11, UNLOAD_3 = 12, UNLOAD_4 = 13";

00287         cout<<endl;

00288         cin>>tmpServo;

00289         if(tmpServo < 10 || tmpServo > 14){

00290             cout<<"Please choose again:"<<endl;

00291             continue;

00292         }

00293         whichservo = (Servo)tmpServo;

00294         */
00295         cout<<endl;
00296         cout<<"Pick a position (set position to -1 to disengage servo and set pwm to 0):";
00297         cin>>position;
00298         cout<<endl;
00299         //setServoPosition((Servo)tmpServo,position);
00300         setServoPosition((Servo)10,position);
00301         setServoPosition((Servo)11,position);
00302         setServoPosition((Servo)12,position);
00303         setServoPosition((Servo)13,position);
00304
00305     }
00306
00307 #endif

00308     return 0;
00309 }
```

Here is the call graph for this function:



6.35.4 Variable Documentation

6.35.4.1 int cleartoload

Definition at line 37 of file main.cpp.

6.35.4.2 int cleartonavigate

Definition at line 36 of file main.cpp.

6.35.4.3 int cleartounload

Definition at line 38 of file main.cpp.

6.35.4.4 int giveitasec

Definition at line 45 of file main.cpp.

6.35.4.5 int loadcounter

Definition at line 41 of file main.cpp.

6.35.4.6 int loadtheblocks

Definition at line 43 of file main.cpp.

6.35.4.7 int navigationbusy

Definition at line 35 of file main.cpp.

6.35.4.8 int robotismoving

Definition at line 39 of file main.cpp.

6.35.4.9 int startloadingthread

Definition at line 46 of file main.cpp.

6.35.4.10 int startroboth

Definition at line 32 of file main.cpp.

6.35.4.11 int startstatus

Definition at line 31 of file main.cpp.

6.35.4.12 int stoproboth

Definition at line 33 of file main.cpp.

6.35.4.13 int track

Definition at line 34 of file main.cpp.

6.35.4.14 int unloadcounter

Definition at line 42 of file main.cpp.

6.35.4.15 int unloadtheblocks

Definition at line 44 of file main.cpp.

6.35.4.16 int whereistherobot

Definition at line 40 of file main.cpp.

6.36 main.cpp

```
00001
00008 /*
00009  * File:   main.cpp
00010  * Author: Nick
00011  *
00012  * Created on February 1, 2016, 8:20 AM
00013  */
00014
00015 #include <cstdlib>
00016 #include <unistd.h>
00017 #include "Servo_Position_Shell.h"
00018 #include "Adafruit_PWMServoDriver.h"
00019 #include <iostream>
00020 #include "NavigationControl.h"
00021 #include "Initialization.cpp"
00022
00023 //#define UNLOADTEST
00024 //#define ARMTEST
00025 //#define NAVTEST
00026 //#define SWITCHTEST
00027 #define READTHINKDO
00028
00029 using namespace std;
00030
00031 int startstatus;
00032 int startroboth;
00033 int stoproboth;
00034 int track;
00035 int navigationbusy;
00036 int cleartonavigate;
00037 int cleartoload;
00038 int cleartounload;
00039 int robotismoving;
00040 int whereistherobot;
00041 int loadcounter;
00042 int unloadcounter;
00043 int loadtheblocks;
00044 int unloadtheblocks;
00045 int giveitasec;
00046 int startloadingthread;
00047
00051 int main() {
00052
00053 initialization();
00054 #ifdef READTHINKDO
00055 /*Track selection initialization will allow track select until start condition is true*/
00056 while(1){
00057     while(track == 1 && startroboth == 0){
00058         startstatus = digitalRead(0);
00059         track = digitalRead(1);
00060         cout<<endl<<"I am in left track mode";
00061         if(startstatus == 1){
00062             startroboth = 1;
00063         }
```

```
00064
00065         /* RUN MODE FOR LEFT TRACK GAME START!*/
00066         while(startroboth == 1 && stoproboth == 0){
00067  //READ for left track
00068
00069             startstatus = digitalRead(0);
00070             navigationbusy = digitalRead(2);
00071
00072 //THINK for left track
00073
00074             //clearance to do an action;
00075             if(cleartoload == 1 || cleartounload == 1){
00076                 cleartonavigate = 0;
00077             }
00078             else if(cleartonavigate == 1 || cleartounload == 1){
00079                 cleartoload = 0;
00080             }
00081             else if(cleartonavigate == 1 || cleartoload == 1){
00082                 cleartounload = 0;
00083             }
00084
00085
00086             // latch to figure out robot position in sequence
00087             if(navigationbusy == 1 && robotismoving == 0){
00088                 robotismoving = 1;
00089                 whereistherobot++;
00090             }else if(navigationbusy == 0 && robotismoving == 1){
00091                 robotismoving = 0;
00092             }
00093
00094             //working actions
00095             if(whereistherobot > 0 && whereistherobot <=4 &&
      navigationbusy == 0){
00096                 cleartoload = 1;
00097             }else if(whereistherobot > 0 && whereistherobot <=4 &&
      navigationbusy == 1){
00098                 cleartoload = 0;
00099             }else if(whereistherobot == 5 && navigationbusy == 0){
00100                 cleartounload = 1;
00101             }else if(whereistherobot == 5 && navigationbusy == 1){
00102                 cleartounload = 0;
00103             }
00104
00105
00106             //controls load action timing
00107             if(cleartoload == 1 && loadcounter < 2000){
00108                 loadtheblocks = 1;
00109                 loadcounter ++;
00110                 startloadingthread = 1;
00111
00112             }else if(loadcounter >= 2000){
00113                 cleartoload = 0;
00114                 loadcounter = 1;
00115             }
00116
00117             //controls the unload action timing
00118             if(cleartounload ==1 && unloadcounter < 150){
00119                 unloadcounter++;
00120                 unloadtheblocks = 1;
00121             }else if(loadcounter >= 150){
00122                 cleartounload = 0;
00123                 unloadcounter = 1;
00124                 unloadtheblocks = 0;
00125             }
00126
00127 //DO for the left track
00128
00129             //Navigate to next position in sequence
00130             if(cleartounload == 0 && cleartoload == 0 &&
      robotismoving == 0){
00131                 digitalWrite(3,HIGH);
00132             }else if(robotismoving == 1){
00133                 digitalWrite(3,LOW);
00134             }
00135
00136             //load blocks
00137             if(loadtheblocks ==1){
00138                 pthread_t andrewthread;
00139                 pthread_create(&andrewthread,NULL,grabCall,NULL);
00140
00141                 loadtheblocks =0;
00142
00143             }
00144
00145             // unload the blocks
00146             if(unloadtheblocks ==1 && unloadcounter < 100){
00147                 setServoPosition(Servo(10),90);
```

```
00148                 setServoPosition(Servo(11),90);
00149                 setServoPosition(Servo(12),90);
00150                 setServoPosition(Servo(13),90);
00151             }else if(unloadtheblocks == 1 && unloadcounter >= 100 &&
    unloadcounter <140){
00152                 setServoPosition(Servo(10),0);
00153                 setServoPosition(Servo(11),0);
00154                 setServoPosition(Servo(12),0);
00155                 setServoPosition(Servo(13),0);
00156             }else if(unloadtheblocks == 1 && unloadcounter >= 140){
00157                 setServoPosition(Servo(10),-1);
00158                 setServoPosition(Servo(11),-1);
00159                 setServoPosition(Servo(12),-1);
00160                 setServoPosition(Servo(13),-1);
00161             }
00162 
00163             cout<<endl<<"I am in run mode of left track";
00164 
00165             /* stops Robot if ever hit on a RTD loop*/
00166             if(startstatus == 0){
00167                 digitalWrite(3,LOW);
00168                 cout<<endl<<"Halting all function stop engaged after a start";
00169                 giveitasec++;
00170                 if(giveitasec >= 50){
00171                     stoproboth = 1;
00172                     giveitasec = 0;
00173                 }
00174             }
00175             cout<<endl<<"The current position of robot is:"<<whereistherobot<<endl;
00176 
00177             // delay for 20 milliseconds per loop
00178             usleep(20000);
00179 
00180 
00181         }
00182     }
00183     /*Track selection initialization will allow track select until start condition is true*/
00184     while(track == 0 && startroboth ==0){
00185         startstatus = digitalRead(0);
00186         track = digitalRead(1);
00187         if(startstatus == 1){
00188             startroboth = 1;
00189         }
00190         cout<<endl<<"I am in right track mode.";
00191         cout<<"Not yet implemented. Danger. Danger. Danger.";
00192     }
00193 }
00194 #endif
00195 #ifdef SWITCHTEST
00196     int cat;
00197     int dog;
00198     wiringPiSetup();
00199     while(1){
00200 
00201         //test each pin if needed for setup and checking
00202         /*
00203         cout<<endl;
00204         cout<<"Input a pin to check:"<<endl;
00205         cin>>dog;
00206         cat =digitalRead(dog);
00207         cout<<"I am reading:"<<cat<<endl;
00208         */
00209 
00210         if(digitalRead(0)==1){
00211             cout<<"Heavy is in stop or idle mode."<<endl;
00212 
00213         }else if(digitalRead(0)==0){
00214             cout<<"Heavy is in run mode!"<<endl;
00215         }
00216         if(digitalRead(1)==1){
00217             cout<<"Heavy is set for left track."<<endl;
00218         }else if(digitalRead(1)==0){
00219             cout<<"Heavy is set for right track."<<endl;
00220         }
00221 
00222         cout<<endl<<"Press enter to continue";
00223         cin.ignore();
00224     }
00225 #endif
00226 
00227 #ifdef ARMTEST
00228 
00229     Servo whichservo;
00230     int tmpServo = -1;
00231     int position;
00232     setup();
00233     while(1){
```

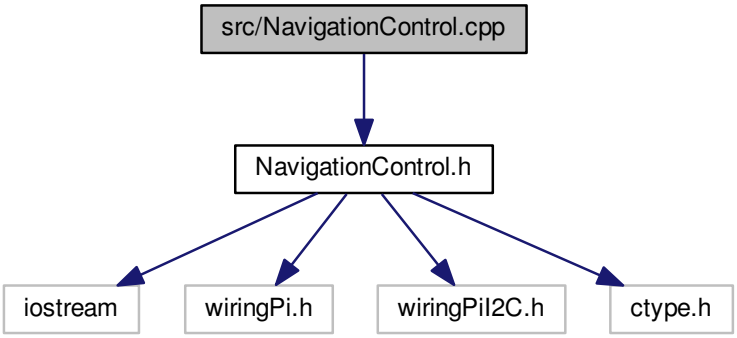
```
00234         cout<<endl;
00235         cout<<"Pick a servo to use: BASE_TURN = 0, BASE_TILT = 1, ELBOW = 2, WRIST_TILT = 3,";
00236         cout<<endl<<"WRIST_PAN = 4, GRIP_LEFT = 5, GRIP_RIGHT = 6";
00237         cout<<endl;
00238         cin>>tmpServo;
00239         if(tmpServo > 6 || tmpServo < 0){
00240             cout<<"Please choose again:"<<endl;
00241             continue;
00242         }
00243         whichservo = (Servo)tmpServo;
00244         cout<<endl;
00245         cout<<"Pick a position (set position to -1 to disengage servo and set pwm to 0):";
00246         cin>>position;
00247         cout<<endl;
00248         setServoPosition(whichservo,position);
00249 
00250     }
00251 #endif
00252 
00253 #ifdef NAVTEST
00254 
00255     int cat = 0;
00256     pinMode(2,INPUT);
00257     while(1){
00258         cout<<"High or Low?:"<<endl;
00259         cin>>cat;
00260         digitalWrite(3,cat);
00261         cout<<endl;
00262         if(digitalRead(2)==1){
00263             cout<<"I am getting a high from Micah"<<endl;
00264 
00265         }else if(digitalRead(2)==0){
00266             cout<<"I am getting a low from Micah"<<endl;
00267         }
00268     }
00269     cin.ignore();
00270 
00271 
00272 
00273 #endif
00274 
00275 #ifdef UNLOADTEST
00276 
00277     Servo whichservo;
00278     int tmpServo = -1;
00279     int position;
00280 
00281     setup();
00282     while(1){
00283         /*
00284         cout<<endl;
00285         cout<<"Pick a servo to use: ";
00286         cout<<endl<<"UNLOAD_1= 10, UNLOAD_2 = 11, UNLOAD_3 = 12, UNLOAD_4 = 13";
00287         cout<<endl;
00288         cin>>tmpServo;
00289         if(tmpServo < 10 || tmpServo > 14){
00290             cout<<"Please choose again:"<<endl;
00291             continue;
00292         }
00293         whichservo = (Servo)tmpServo;
00294         */
00295         cout<<endl;
00296         cout<<"Pick a position (set position to -1 to disengage servo and set pwm to 0):";
00297         cin>>position;
00298         cout<<endl;
00299         //setServoPosition((Servo)tmpServo,position);
00300         setServoPosition((Servo)10,position);
00301         setServoPosition((Servo)11,position);
00302         setServoPosition((Servo)12,position);
00303         setServoPosition((Servo)13,position);
00304 
00305     }
00306 
00307 #endif
00308     return 0;
00309 }
```

6.37 src/NavigationControl.cpp File Reference

Contains the navigation control function definitions.

```
#include "NavigationControl.h"
```

Include dependency graph for NavigationControl.cpp:



Functions

- void navigationSetup (uint8_t navigation_add)
- void commandNavigation (uint8_t navigationcommand)

Variables

- uint8_t navigationcmd
- uint8_t nav_fd

6.37.1 Detailed Description

Contains the navigation control function definitions.

Author

Nickolas Neely

Definition in file NavigationControl.cpp.

6.37.2 Function Documentation

6.37.2.1 void commandNavigation (uint8_t *navigationcommand*)

Definition at line 19 of file NavigationControl.cpp.

```
00019                                                {
00020     uint8_t data = navigationcommand;
00021     wiringPiI2CWrite(nav_fd, data);
00022 }
```

6.37.2.2 void navigationSetup (uint8_t *navigation_add*)

Definition at line 14 of file NavigationControl.cpp.

```
00014                                           {
00015     nav_fd=wiringPiI2CSetup(navigation_add);
00016 
00017 }
```

6.37.3 Variable Documentation

6.37.3.1 uint8_t nav_fd

Definition at line 11 of file NavigationControl.cpp.

6.37.3.2 uint8_t navigationcmd

Definition at line 10 of file NavigationControl.cpp.

6.38 NavigationControl.cpp

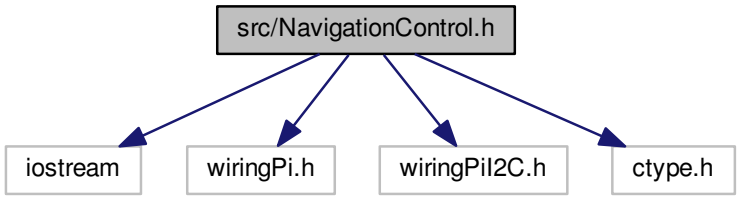
```
00001 
00007 #include "NavigationControl.h"
00008 
00009 //uint8_t navigation_address = 10u;
00010 uint8_t navigationcmd;
00011 uint8_t nav_fd;
00012 
00013 
00014 void navigationSetup(uint8_t navigation_add){
00015     nav_fd=wiringPiI2CSetup(navigation_add);
00016 
00017 }
00018 
00019 void commandNavigation(uint8_t navigationcommand){
00020     uint8_t data = navigationcommand;
00021     wiringPiI2CWrite(nav_fd, data);
00022 }
```

6.39 src/NavigationControl.h File Reference

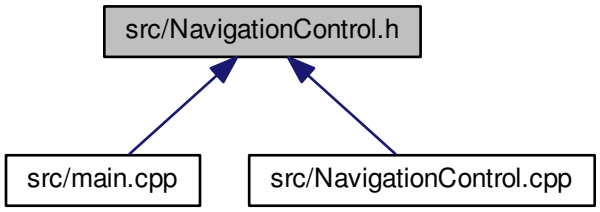
Contains the function definitions for navigation control.

```
#include <iostream>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <ctype.h>
```

Include dependency graph for NavigationControl.h:



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



Macros

- #define uint8_t unsigned char

Functions

- void navigationSetup (uint8_t navigation_add)
- void commandNavigation (uint8_t navigationcommand)

6.39.1 Detailed Description

Contains the function definitions for navigation control.

Author

Nickolas Neely

Date

22. Febrary 2016, 12:00 PM

Definition in file NavigationControl.h.

6.39.2 Macro Definition Documentation

6.39.2.1 #define uint8_t unsigned char

Definition at line 24 of file NavigationControl.h.

6.39.3 Function Documentation

6.39.3.1 void commandNavigation (uint8_t *navigationcommand*)

Definition at line 19 of file NavigationControl.cpp.

```
00019                                                {
00020     uint8_t data = navigationcommand;
00021     wiringPiI2CWrite(nav_fd, data);
00022 }
```

6.39.3.2 void navigationSetup (uint8_t *navigation_add*)

Definition at line 14 of file NavigationControl.cpp.

```
00014                                              {
00015     nav_fd=wiringPiI2CSetup(navigation_add);
00016
00017 }
```

6.40 NavigationControl.h

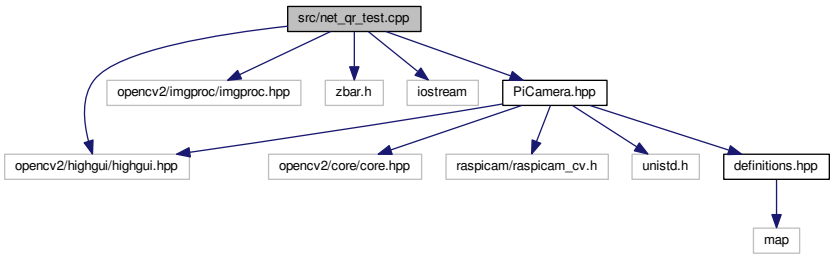
```
00001
00008 /*
00009  * File:   NavigationControl.h
00010  * Author: Nick
00011  *
00012  * Created on February 22, 2016, 12:00 PM
00013  */
00014
00015
00016 #ifndef NAVIGATIONCONTROL_H
00017 #define NAVIGATIONCONTROL_H
00018 #include <iostream>
00019 #include <wiringPi.h>
00020 #include <wiringPiI2C.h>
00021 #include <ctype.h>
00022
00023 #ifndef uint8_t
00024 #define uint8_t unsigned char
00025 #endif
00026
00027 #ifdef __cplusplus
00028 extern "C"{
00029 #endif
00030
00031
00032 void navigationSetup(uint8_t navigation_add);
00033
00034 void commandNavigation(uint8_t navigationcommand);
00035
00036 #ifdef __cplusplus
00037 }
00038 #endif
00039
00040 #endif /*NAVIGATIONCONTROL_H*/
00041
```

6.41 src/net_qr_test.cpp File Reference

Contains test program for reading QR codes.

```
#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[])

6.41.1 Detailed Description

Contains test program for reading QR codes.

Author

Michael Young
Samuel Andrew Wisner, awisner94@gmail.com

Definition in file net_qr_test.cpp.

6.41.2 Function Documentation

6.41.2.1 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 25 of file net_qr_test.cpp.

```
00026 {
00027     PiCamera cam; // open the video camera no. 0
00028     ImageScanner scanner;
00029     scanner.set_config(ZBAR_NONE, ZBAR_CFG_ENABLE, 1);
00030
00031     namedWindow("MyVideo",CV_WINDOW_AUTOSIZE); //create a window called "MyVideo"
00032
00033     while (1)
00034     {
00035         Mat frame = cam.Snap();
00036         Mat grey;
00037         cvtColor(frame,grey,CV_BGR2GRAY);
00038
00039         int width = frame.cols;
00040         int height = frame.rows;
00041         uchar *raw = (uchar *)grey.data;
00042         // wrap image data
00043         zbar::Image image(width, height, "Y800", raw, width * height);
```

```
00044         // scan the image for barcodes
00045         int n = scanner.scan(image);
00046         // extract results
00047         for(Image::SymbolIterator symbol = image.symbol_begin();
00048                 symbol != image.symbol_end();
00049                 ++symbol) {
00050             vector<Point> vp;
00051             // do something useful with results
00052             cout << "decoded " << symbol->get_type_name()  << " symbol \"" << symbol->get_data() << '"' <<"
      "<< endl;
00053             int n = symbol->get_location_size();
00054             for(int i=0;i<n;i++){
00055                 vp.push_back(Point(symbol->get_location_x(i),symbol->get_location_y(i)));
00056             }
00057             RotatedRect r = minAreaRect(vp);
00058             Point2f pts[4];
00059             r.points(pts);
00060             for(int i=0;i<4;i++){
00061                 line(frame,pts[i],pts[(i+1)%4],Scalar(255,0,0),3);
00062             }
00063         }
00064
00065         imshow("MyVideo", frame); //show the frame in "MyVideo" window
00066
00067         if (waitKey(30) == 27) //wait for 'esc' key press for 30ms. If 'esc' key is pressed, break loop
00068         {
00069             cout << "esc key is pressed by user" << endl;
00070             break;
00071         }
00072     }
00073     return 0;
00074
00075 }
```

Here is the call graph for this function:

6.42 net_qr_test.cpp

```
00001
00008 #include <opencv2/highgui/highgui.hpp>
00009 #include <opencv2/imgproc/imgproc.hpp>
00010 #include <zbar.h>
00011 #include <iostream>
00012
00013 #include "PiCamera.hpp"
00014
00015 using namespace cv;
00016 using namespace std;
00017 using namespace zbar;
00018 using namespace ChipChipArray;
00019
00025 int main(int argc, char* argv[])
00026 {
00027     PiCamera cam; // open the video camera no. 0
00028     ImageScanner scanner;
00029     scanner.set_config(ZBAR_NONE, ZBAR_CFG_ENABLE, 1);
00030
00031     namedWindow("MyVideo",CV_WINDOW_AUTOSIZE); //create a window called "MyVideo"
00032
00033     while (1)
00034     {
00035         Mat frame = cam.Snap();
00036         Mat grey;
00037         cvtColor(frame,grey,CV_BGR2GRAY);
00038
```

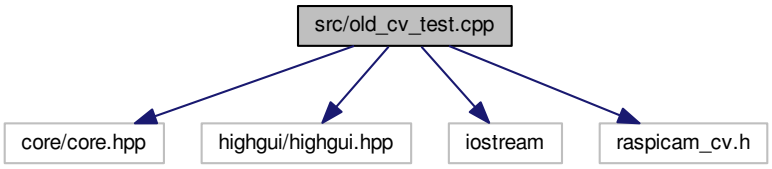
```
00039         int width = frame.cols;
00040         int height = frame.rows;
00041         uchar *raw = (uchar *)grey.data;
00042         // wrap image data
00043         zbar::Image image(width, height, "Y800", raw, width * height);
00044         // scan the image for barcodes
00045         int n = scanner.scan(image);
00046         // extract results
00047         for(Image::SymbolIterator symbol = image.symbol_begin();
00048                 symbol != image.symbol_end();
00049                 ++symbol) {
00050             vector<Point> vp;
00051             // do something useful with results
00052             cout << "decoded " << symbol->get_type_name()  << " symbol \"" << symbol->get_data() << '"' <<"
      "<< endl;
00053             int n = symbol->get_location_size();
00054             for(int i=0;i<n;i++){
00055                 vp.push_back(Point(symbol->get_location_x(i),symbol->get_location_y(i)));
00056             }
00057             RotatedRect r = minAreaRect(vp);
00058             Point2f pts[4];
00059             r.points(pts);
00060             for(int i=0;i<4;i++){
00061                 line(frame,pts[i],pts[(i+1)%4],Scalar(255,0,0),3);
00062             }
00063         }
00064
00065         imshow("MyVideo", frame); //show the frame in "MyVideo" window
00066
00067         if (waitKey(30) == 27) //wait for 'esc' key press for 30ms. If 'esc' key is pressed, break loop
00068         {
00069             cout << "esc key is pressed by user" << endl;
00070             break;
00071         }
00072     }
00073     return 0;
00074
00075 }
```

6.43 src/old_cv_test.cpp File Reference

Contains old test program for the RaspiCam_Cv class.

```
#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 ()

6.43.1 Detailed Description

Contains old test program for the RaspiCam_Cv class.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file old_cv_test.cpp.

6.43.2 Function Documentation

6.43.2.1 int main ( )

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 17 of file old_cv_test.cpp.

```
00017            {
00018     cv::namedWindow("Test", CV_WINDOW_AUTOSIZE);
00019
00020     raspicam::RaspiCam_Cv cam;
00021     cv::Mat image;
00022
00023     bool s = cam.set(CV_CAP_PROP_FORMAT, CV_16UC3);
00024     bool o = cam.open();
00025
00026     while(true) {
00027         cam.grab();
00028         cam.retrieve(image);
00029         cv::imshow("Test", image);
00030         cv::waitKey(1000);
00031
00032     }
00033
00034     cam.release();
00035 }
```

6.44 old_cv_test.cpp

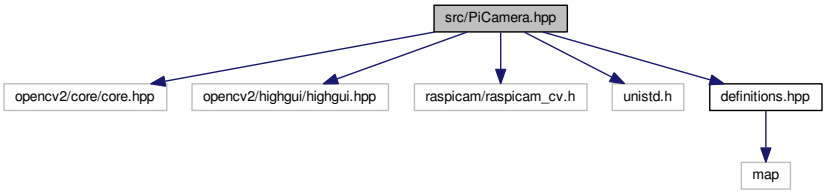
```
00001
00007 #include <core/core.hpp>
00008 #include <highgui/highgui.hpp>
00009 #include <iostream>
00010
00011 #include "raspicam_cv.h"
00012
00017 int main() {
00018     cv::namedWindow("Test", CV_WINDOW_AUTOSIZE);
00019
00020     raspicam::RaspiCam_Cv cam;
00021     cv::Mat image;
00022
00023     bool s = cam.set(CV_CAP_PROP_FORMAT, CV_16UC3);
00024     bool o = cam.open();
00025
00026     while(true) {
00027         cam.grab();
00028         cam.retrieve(image);
00029         cv::imshow("Test", image);
00030         cv::waitKey(1000);
00031
00032     }
00033
00034     cam.release();
00035 }
```

6.45 src/PiCamera.hpp File Reference

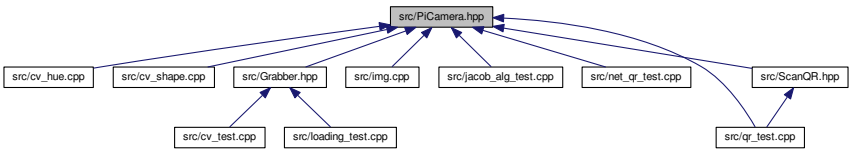
Contains PiCamera class.

```
#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

6.45.1 Detailed Description

Contains PiCamera class.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file PiCamera.hpp.

6.46 PiCamera.hpp

```
00001
00007 #ifndef PiCamera_H
00008 #define PiCamera_H
00009
00010 #include <opencv2/core/core.hpp>
00011 #include <opencv2/highgui/highgui.hpp>
00012 #include <raspicam/raspicam_cv.h>
```

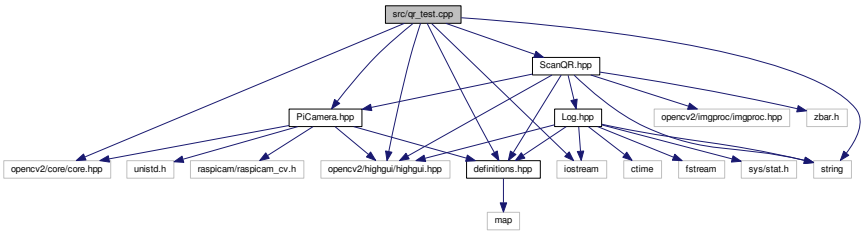
```
00013 #include <unistd.h>
00014
00015 #include "definitions.hpp"
00016
00017 namespace ChipChipArray {
00018
00024     class PiCamera {
00025         public:
00029             PiCamera() : PiCamera(true) {};
00030
00037             PiCamera(bool useColor);
00038
00042             void Close();
00043
00050             cv::Mat Snap();
00051
00052         private:
00056             raspicam::RaspiCam_Cv cam;
00057     };
00058
00059     PiCamera::PiCamera(bool useColor) {
00060         cam.set(CV_CAP_PROP_FORMAT, (useColor ? CV_16UC3 : CV_16UC1));
00061         cam.open();
00062         usleep(500000);  // required to allow camera time to adjust!
00063     }
00064
00065     void PiCamera::Close() {
00066         cam.release();
00067     }
00068
00069     cv::Mat PiCamera::Snap() {
00070         if(!cam.isOpened()) throw std::runtime_error("Camera "
00071                 "is not open!");
00072
00073         cv::Mat image;
00074         cam.grab();
00075         cam.retrieve(image);
00076         return image;
00077     }
00078 }
00079
00080 #endif
```

6.47 src/qr_test.cpp File Reference

Contains test program for ScanQR() function.

```
#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 ()

6.47.1 Detailed Description

Contains test program for ScanQR() function.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file qr_test.cpp.

6.47.2 Function Documentation

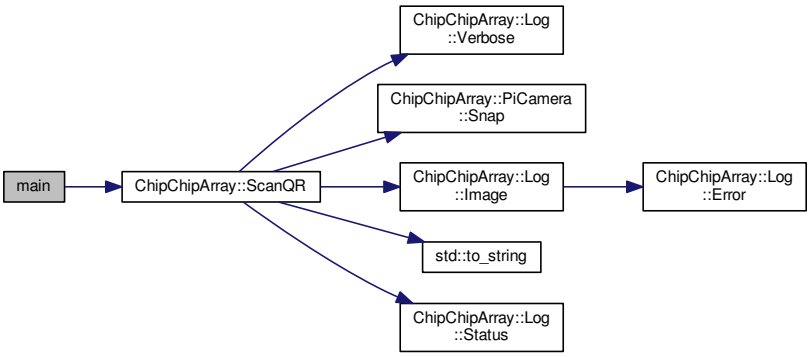
6.47.2.1 int main ( )

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

Definition at line 22 of file qr_test.cpp.

```
00022           {
00023     while(true) {
00024         Color color = ScanQR();
00025         std::string colstr;
00026 
00027         switch(color) {
00028             case Color::Red:
00029                 colstr = "RED";
00030                 break;
00031 
00032             case Color::Yellow:
00033                 colstr = "YELLOW";
00034                 break;
00035 
00036             case Color::Green:
00037                 colstr = "GREEN";
00038                 break;
00039 
00040             case Color::Blue:
00041                 colstr = "BLUE";
00042                 break;
00043 
00044         }
00045 
00046         std::cout << colstr << std::endl;
00047     }
00048 }
```

Here is the call graph for this function:



6.48 qr_test.cpp

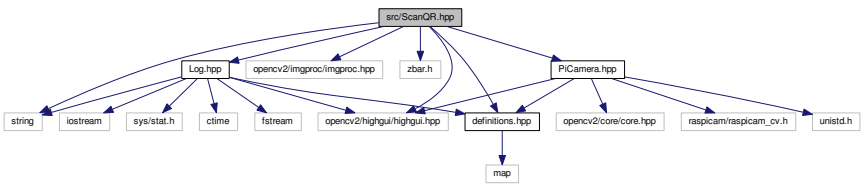
```
00001
00007 #include <iostream>
00008 #include <opencv2/core/core.hpp>
00009 #include <opencv2/highgui/highgui.hpp>
00010 #include <string>
00011
00012 #include "definitions.hpp"
00013 #include "PiCamera.hpp"
00014 #include "ScanQR.hpp"
00015
00016 using namespace ChipChipArray;
00017
00022 int main() {
00023     while(true) {
00024         Color color = ScanQR();
00025         std::string colstr;
00026
00027         switch(color) {
00028             case Color::Red:
00029                 colstr = "RED";
00030                 break;
00031
00032             case Color::Yellow:
00033                 colstr = "YELLOW";
00034                 break;
00035
00036             case Color::Green:
00037                 colstr = "GREEN";
00038                 break;
00039
00040             case Color::Blue:
00041                 colstr = "BLUE";
00042                 break;
00043
00044         }
00045
00046         std::cout << colstr << std::endl;
00047     }
00048 }
```

6.49 src/ScanQR.hpp File Reference

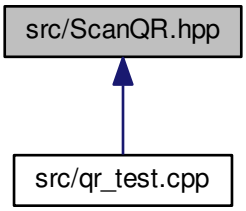
Contains ScanQR() function.

```
#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

Functions

- Color ChipChipArray::ScanQR ()

Variables

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

6.49.1 Detailed Description

Contains ScanQR() function.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file ScanQR.hpp.

6.50 ScanQR.hpp

```
00001
00007 #ifndef ScanQR_H
00008 #define ScanQR_H
00009
00010 #include <string>
00011 #include <opencv2/highgui/highgui.hpp>
00012 #include <opencv2/imgproc/imgproc.hpp>
00013 #include <zbar.h>
00014
00015 #include "definitions.hpp"
00016 #include "Log.hpp"
00017 #include "PiCamera.hpp"
00018
00019 namespace ChipChipArray {
00020
00024     uint8 qrInvokeCount = 0;
00025
```

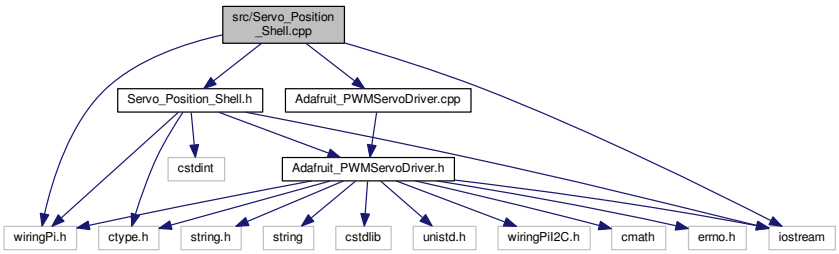
```
00029     Log scanQrLog("logs/ScanQR", LogMode::Multi);
00030
00041     Color ScanQR() {
00042         // 0. Initialize variables
00043         Color color;
00044         PiCamera cam(false);
00045
00046         // 1. Position arm
00047         scanQrLog.Verbose("Positioning arm");
00048
00049         // 2. Scan images from camera
00050         scanQrLog.Verbose("Scanning for QR code");
00051
00052         // Nick's supposed to make sure this isn't an endles loop
00053         while(true) {
00054             // get image
00055             cv::Mat frame = cam.Snap();
00056             cv::Mat canvas;
00057             cv::cvtColor(frame, canvas, CV_BGR2GRAY);
00058             scanQrLog.Image(canvas, std::to_string(++qrInvokeCount)
00059                     + ".bmp");
00060
00061             uint32 width = canvas.cols;
00062             uint32 height = canvas.rows;
00063             zbar::Image image(width, height, "Y800",
00064                     (uchar*)canvas.data, width * height);
00065
00066             zbar::ImageScanner scanner;
00067             scanner.set_config(zbar::ZBAR_NONE, zbar::ZBAR_CFG_ENABLE, 1);
00068             scanner.scan(image);
00069             zbar::Image::SymbolIterator symbol = image.symbol_begin();
00070
00071             if(symbol != image.symbol_end()) {
00072                 switch(symbol->get_data()[0]) {
00073                     case 'r':
00074                         color = Color::Red;
00075                         break;
00076
00077                     case 'y':
00078                         color = Color::Yellow;
00079                         break;
00080
00081                     case 'g':
00082                         color = Color::Green;
00083                         break;
00084
00085                     case 'b':
00086                         color = Color::Blue;
00087                         break;
00088                 }
00089
00090                 scanQrLog.Status("Detected " + std::to_string(color)
00091                         + " train car");
00092                 break;
00093             }
00094         }
00095
00096         // 3. Retract arm
00097         scanQrLog.Verbose("Retracting arm");
00098         return color;
00099     }
00100 }
00101
00102 #endif
```

6.51 src/Servo_Position_Shell.cpp File Reference

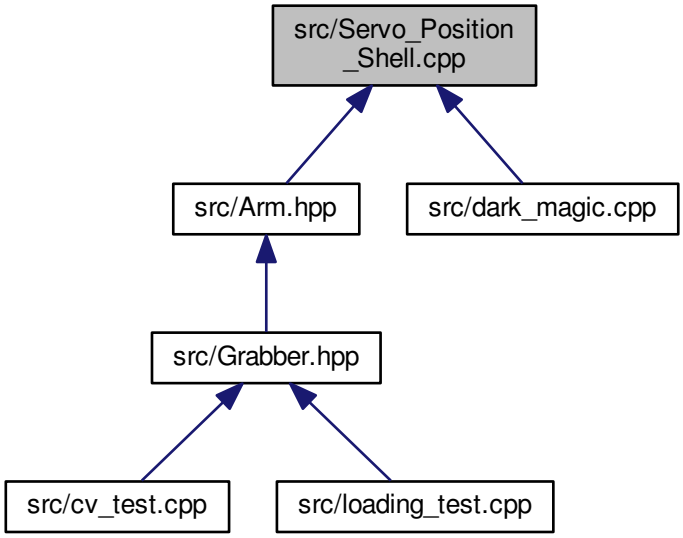
COntains the function definitions for the servo position shell.

```
#include <wiringPi.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

6.51.1 Detailed Description

COntains the function definitions for the servo position shell.

Author

Nickolas Neely

Date

8. February 2016, 12:05 PM

Definition in file Servo_Position_Shell.cpp.

6.51.2 Macro Definition Documentation

6.51.2.1 #define SERVOMAX 600

Definition at line 31 of file Servo_Position_Shell.cpp.

6.51.2.2 #define SERVOMIN 150

Definition at line 30 of file Servo_Position_Shell.cpp.

6.51.3 Function Documentation

6.51.3.1 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 71 of file Servo_Position_Shell.cpp.

```
00071                                                   {
00072     // works for servo 0, 3, 4
00073     double dividedconstant = 180.0;
00074     double highservo = 2.4;
00075     double lowservo = 0.6;
00076     // To fix the magical digital servo on LIFT 1
00077     double highservoweird = 1.9;
00078     double lowservoweird = 0.6;
00079     // To compensate for the bent servo spline on LIFT 2
00080     double highservospline = 2.25;
00081     double lowservospline = 0.6;
00082     // works for servo 1, 2
00083     double digitalservohigh = 2.45;
00084     double digitalservolow = 0.9;
00085     // left gripper servo 5
00086     double gripleftopen = 2.2;
00087     double gripleftclose = 1.3;
00088     // right gripper servo 6
00089     double griprightopen = 2.2;
00090     double griprightclose = 1.3;
00091     double pulse;
00092
00093     switch (whichservo) {
```

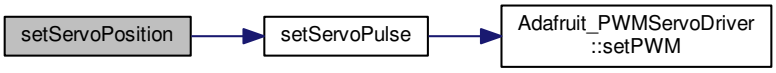
```
00094
00095             // BASE TURN
00096         case 0:
00097         {
00098             if (position == -1) {
00099                 pulse = 0.0;
00100             }else if (position < 0){
00101                 position = 20;
00102                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00103             }else if (position > 179){
00104                 position = 179;
00105                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00106             }else{
00107                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00108             }
00109
00110         }
00111
00112             break;
00113
00114             // BASE TILT
00115         case 1:
00116         {
00117
00118
00119             if (position == -1) {
00120                 pulse = 0.0;
00121             } else if (position < 90){
00122                 position = 90;
00123                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00124             } else if (position > 172){
00125                 position = 172;
00126                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00127             } else {
00128                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00129             }
00130         }
00131             break;
00132
00133             // ELBOW
00134         case 2:
00135         {
00136             if (position == -1) {
00137                 pulse = 0.0;
00138             } else if (position < 43){
00139                 position = 43;
00140                 pulse = ((((digitalservohigh - digitalservolow) / dividedconstant)*((double) position)) +
      digitalservolow);
00141             } else if (position > 179){
00142                 position = 179;
00143                 pulse = ((((digitalservohigh - digitalservolow) / dividedconstant)*((double) position)) +
      digitalservolow);
00144             } else {
00145                 pulse = ((((digitalservohigh - digitalservolow) / dividedconstant)*((double) position)) +
      digitalservolow);
00146             }
00147         }
00148             break;
00149
00150             // WRIST TURN
00151         case 3:
00152         {
00153             if (position == -1) {
00154                 pulse = 0.0;
00155             } else {
00156                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00157             }
00158         }
00159             break;
00160
00161             // WRIST PAN
00162         case 4:
00163         {
00164             if (position == -1) {
00165                 pulse = 0.0;
00166             } else if (position < 0){
00167                 position = 0;
00168                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00169             } else if (position > 180){
00170                 position = 180;
00171                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00172             } else {
00173                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00174             }
00175         }
00176             break;
00177
```

```
00178             // GRIP LEFT
00179         case 5:
00180         {
00181             if (position == -1) {
00182                 pulse = 0.0;
00183             } else if(position < 0){
00184                 position = 0;
00185                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00186             }else if(position > 90){
00187                 position = 90;
00188                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00189             } else {
00190                 pulse = ((((gripleftopen - gripleftclose) / dividedconstant)*((double) position)) +
      gripleftclose);
00191             }
00192         }
00193             break;
00194
00195             // GRIP RIGHT
00196         case 6:
00197         {
00198             if (position == -1) {
00199                 pulse = 0.0;
00200             } else if(position < 90){
00201                 position = 90;
00202                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00203             } else if(position > 180){
00204                 position = 180;
00205                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00206             }else{
00207                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00208             }
00209         }
00210             break;
00211
00212             // Michael Yellow Gate
00213         case 7:
00214         {
00215             if (position == -1) {
00216                 pulse = 0.0;
00217             } else if(position < 0){
00218                 position = 0;
00219                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00220             } else if(position > 90){
00221                 position = 90;
00222                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00223             } else {
00224                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00225             }
00226         }
00227             break;
00228
00229             // Michael Green Gate
00230         case 8:
00231         {
00232             if (position == -1) {
00233                 pulse = 0.0;
00234             } else if(position < 0){
00235                 position = 0;
00236                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00237             } else if(position > 90){
00238                 position = 90;
00239                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00240             } else {
00241                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00242             }
00243         }
00244             break;
00245
00246             // Michael Blue Gate
00247         case 9:
00248         {
00249             if (position == -1) {
00250                 pulse = 0.0;
00251             } else if(position < 0){
00252                 position = 0;
00253                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00254             } else if(position > 90){
00255                 position = 90;
00256                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00257             } else {
00258                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
```

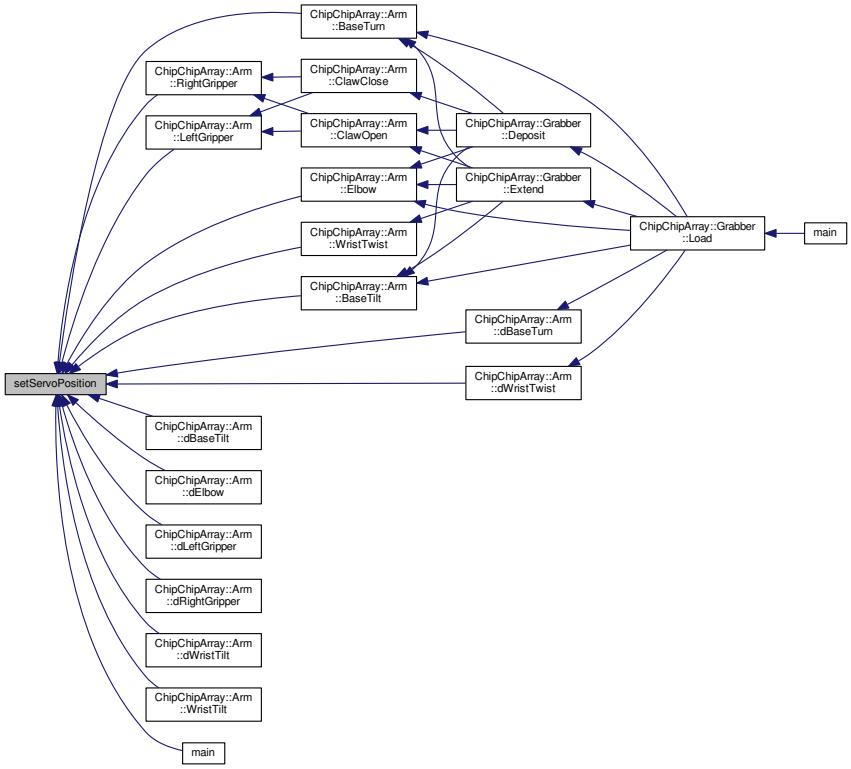
```
00259             }
00260         }
00261             break;
00262 
00263             // Michael Lift 1
00264         case 10:
00265         {
00266             if (position == -1) {
00267                 pulse = 0.0;
00268             } else if(position < 0){
00269                 position = 0;
00270                 pulse = ((((highservoweird - lowservoweird) / dividedconstant)*((double) position)) +
      lowservoweird);
00271             } else if(position > 105){
00272                 position = 105;
00273                 pulse = ((((highservoweird - lowservoweird) / dividedconstant)*((double) position)) +
      lowservoweird);
00274             } else {
00275                 pulse = ((((highservoweird - lowservoweird) / dividedconstant)*((double) position)) +
      lowservoweird);
00276             }
00277         }
00278             break;
00279 
00280             //Michael Lift 2
00281         case 11:
00282         {
00283             if (position == -1) {
00284                 pulse = 0.0;
00285             } else if(position < 0){
00286                 position = 0;
00287                 pulse = ((((highservospline - lowservospline) / dividedconstant)*((double) position)) +
      lowservospline);
00288             } else if(position > 105){
00289                 position = 105;
00290                 pulse = ((((highservospline - lowservospline) / dividedconstant)*((double) position)) +
      lowservospline);
00291             } else {
00292                 pulse = ((((highservospline - lowservospline) / dividedconstant)*((double) position)) +
      lowservospline);
00293             }
00294         }
00295             break;
00296 
00297             //Michael lift 3
00298         case 12:
00299         {
00300             if (position == -1) {
00301                 pulse = 0.0;
00302             } else if(position < 0){
00303                 position = 0;
00304                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00305             } else if(position > 105){
00306                 position = 105;
00307                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00308             } else {
00309                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00310             }
00311         }
00312             break;
00313 
00314             // Michael lift 4
00315         case 13:
00316         {
00317             if (position == -1) {
00318                 pulse = 0.0;
00319             } else if(position < 0){
00320                 position = 0;
00321                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00322             } else if(position > 105){
00323                 position = 105;
00324                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00325             } else {
00326                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00327             }
00328         }
00329             break;
00330 
00331             // Michael RED GATE
00332         case 14:
00333         {
00334             if (position == -1) {
00335                 pulse = 0.0;
00336             } else if(position < 0){
00337                 position = 0;
00338                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00339             } else if(position > 105){
```

```
00340                 position = 105;
00341                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00342             } else {
00343                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00344             }
00345         }
00346             break;
00347     }
00348     setServoPulse(whichservo, pulse);
00349
00350 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

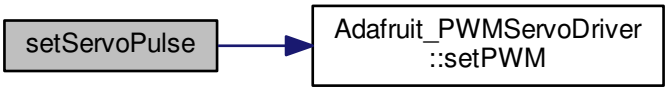
Parameters

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.

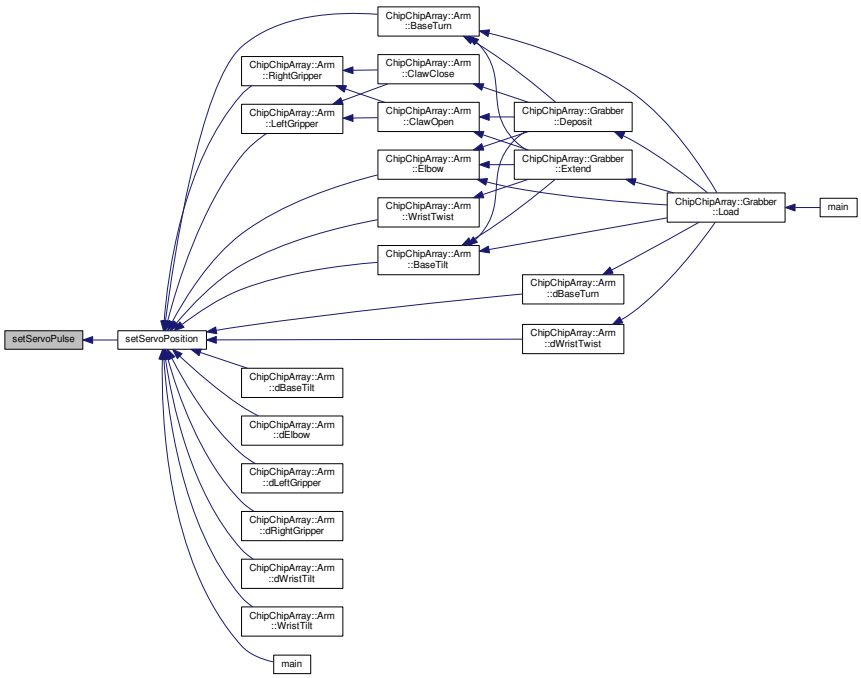
Definition at line 50 of file Servo_Position_Shell.cpp.

```
00050                                                 {
00051     double pulselength;
00052
00053     pulselength = 1000000; // 1,000,000 us per second
00054     pulselength /= 60; // 60 Hz
00055     //cout << pulselength << " us per period" << endl;
00056     pulselength /= 4096; // 12 bits of resolution
00057     //cout << pulselength << "us per bit" << endl;
00058     pulse *= 1000;
00059     pulse /= pulselength;
00060     //cout << (uint16_t) pulse << endl;
00061     pwm.setPWM(servo_num, 0, (uint16_t) pulse);
00062     //cout << endl;
00063 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



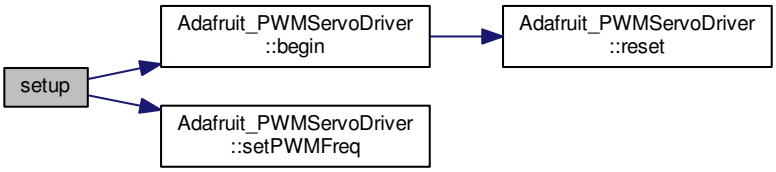
6.51.3.3 void setup ( )

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.

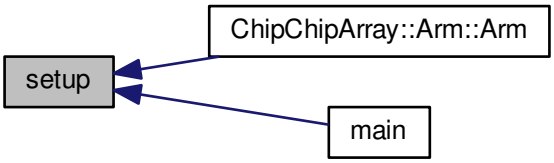
Definition at line 41 of file Servo_Position_Shell.cpp.

```
00041             {
00042     //cout << "Testing Servos" << endl;
00043     pwm.begin();
00044     pwm.setPWMFreq(60.0); // Analog servos run at ~60 Hz updates
00045 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.51.4 Variable Documentation

6.51.4.1 Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver()

Definition at line 22 of file Servo_Position_Shell.cpp.

6.51.4.2 uint8_t servo_num

Definition at line 34 of file Servo_Position_Shell.cpp.

6.52 Servo_Position_Shell.cpp

```
00001
00008 /*
```

```
00009  * File:   Servo_Position_Shell.cpp
00010  * Author: Nickolas Neely
00011  *
00012  * Created on February 8, 2016, 12:05 PM
00013  */
00014
00015 #include <wiringPi.h>
00016 #include "Adafruit_PWMServoDriver.cpp"
00017 #include <iostream>
00018 #include "Servo_Position_Shell.h"
00019
00020
00021 // called this way, it uses the default address 0x40
00022 Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
00023 // you can also call it with a different address you want
00024 //Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x41);
00025
00026 // Depending on your servo make, the pulse width min and max may vary, you
00027 // want these to be as small/large as possible without hitting the hard stop
00028 // for max range. You'll have to tweak them as necessary to match the servos you
00029 // have!
00030 #define SERVOMIN  150 // this is the 'minimum' pulse length count (out of 4096)
00031 #define SERVOMAX  600 // this is the 'maximum' pulse length count (out of 4096)
00032
00033 // our servo # counter
00034 uint8_t servo_num;
00035
00041 void setup() {
00042     //cout << "Testing Servos" << endl;
00043     pwm.begin();
00044     pwm.setPWMFreq(60.0); // Analog servos run at ~60 Hz updates
00045 }
00046
00047 // you can use this function if you'd like to set the pulse length in seconds
00048 // e.g. setServoPulse(0, 0.001) is a ~1 millisecond pulse width. its not precise!
00049
00050 void setServoPulse(uint8_t servo_num, double pulse) {
00051     double pulselength;
00052
00053     pulselength = 1000000; // 1,000,000 us per second
00054     pulselength /= 60; // 60 Hz
00055     //cout << pulselength << " us per period" << endl;
00056     pulselength /= 4096; // 12 bits of resolution
00057     //cout << pulselength << "us per bit" << endl;
00058     pulse *= 1000;
00059     pulse /= pulselength;
00060     //cout << (uint16_t) pulse << endl;
00061     pwm.setPWM(servo_num, 0, (uint16_t) pulse);
00062     //cout << endl;
00063 }
00064
00071 void setServoPosition(Servo whichservo, int position) {
00072     // works for servo 0, 3, 4
00073     double dividedconstant = 180.0;
00074     double highservo = 2.4;
00075     double lowservo = 0.6;
00076     // To fix the magical digital servo on LIFT 1
00077     double highservoweird = 1.9;
00078     double lowservoweird = 0.6;
00079     // To compensate for the bent servo spline on LIFT 2
00080     double highservospline = 2.25;
00081     double lowservospline = 0.6;
00082     // works for servo 1, 2
00083     double digitalservohigh = 2.45;
00084     double digitalservolow = 0.9;
00085     // left gripper servo 5
00086     double gripleftopen = 2.2;
00087     double gripleftclose = 1.3;
00088     // right gripper servo 6
00089     double griprightopen = 2.2;
00090     double griprightclose = 1.3;
00091     double pulse;
00092
00093     switch (whichservo) {
00094
00095             // BASE TURN
00096         case 0:
00097         {
00098             if (position == -1) {
00099                 pulse = 0.0;
00100             }else if (position < 0){
00101                 position = 20;
00102                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00103             }else if (position > 179){
00104                 position = 179;
00105                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00106             }else{
```

```
00107                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00108             }
00109 
00110         }
00111 
00112             break;
00113 
00114             // BASE TILT
00115         case 1:
00116         {
00117 
00118 
00119             if (position == -1) {
00120                 pulse = 0.0;
00121             } else if (position < 90){
00122                 position = 90;
00123                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00124             } else if (position > 172){
00125                 position = 172;
00126                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00127             } else {
00128                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00129             }
00130         }
00131             break;
00132 
00133             // ELBOW
00134         case 2:
00135         {
00136             if (position == -1) {
00137                 pulse = 0.0;
00138             } else if (position < 43){
00139                 position = 43;
00140                 pulse = ((((digitalservohigh - digitalservolow) / dividedconstant)*((double) position)) +
      digitalservolow);
00141             } else if (position > 179){
00142                 position = 179;
00143                 pulse = ((((digitalservohigh - digitalservolow) / dividedconstant)*((double) position)) +
      digitalservolow);
00144             } else {
00145                 pulse = ((((digitalservohigh - digitalservolow) / dividedconstant)*((double) position)) +
      digitalservolow);
00146             }
00147         }
00148             break;
00149 
00150             // WRIST TURN
00151         case 3:
00152         {
00153             if (position == -1) {
00154                 pulse = 0.0;
00155             } else {
00156                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00157             }
00158         }
00159             break;
00160 
00161             // WRIST PAN
00162         case 4:
00163         {
00164             if (position == -1) {
00165                 pulse = 0.0;
00166             } else if (position < 0){
00167                 position = 0;
00168                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00169             } else if (position > 180){
00170                 position = 180;
00171                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00172             } else {
00173                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00174             }
00175         }
00176             break;
00177 
00178             // GRIP LEFT
00179         case 5:
00180         {
00181             if (position == -1) {
00182                 pulse = 0.0;
00183             } else if(position < 0){
00184                 position = 0;
00185                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00186             }else if(position > 90){
00187                 position = 90;
00188                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
```

```
00189             } else {
00190                 pulse = ((((gripleftopen - gripleftclose) / dividedconstant)*((double) position)) +
      gripleftclose);
00191             }
00192         }
00193             break;
00194
00195             // GRIP RIGHT
00196         case 6:
00197         {
00198             if (position == -1) {
00199                 pulse = 0.0;
00200             } else if(position < 90){
00201                 position = 90;
00202                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00203             } else if(position > 180){
00204                 position = 180;
00205                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00206             }else{
00207                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00208             }
00209         }
00210             break;
00211
00212             // Michael Yellow Gate
00213         case 7:
00214         {
00215             if (position == -1) {
00216                 pulse = 0.0;
00217             } else if(position < 0){
00218                 position = 0;
00219                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00220             } else if(position > 90){
00221                 position = 90;
00222                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00223             } else {
00224                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00225             }
00226         }
00227             break;
00228
00229             // Michael Green Gate
00230         case 8:
00231         {
00232             if (position == -1) {
00233                 pulse = 0.0;
00234             } else if(position < 0){
00235                 position = 0;
00236                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00237             } else if(position > 90){
00238                 position = 90;
00239                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00240             } else {
00241                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00242             }
00243         }
00244             break;
00245
00246             // Michael Blue Gate
00247         case 9:
00248         {
00249             if (position == -1) {
00250                 pulse = 0.0;
00251             } else if(position < 0){
00252                 position = 0;
00253                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00254             } else if(position > 90){
00255                 position = 90;
00256                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00257             } else {
00258                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00259             }
00260         }
00261             break;
00262
00263             // Michael Lift 1
00264         case 10:
00265         {
00266             if (position == -1) {
00267                 pulse = 0.0;
00268             } else if(position < 0){
00269                 position = 0;
00270                 pulse = ((((highservoweird - lowservoweird) / dividedconstant)*((double) position)) +
      lowservoweird);
```

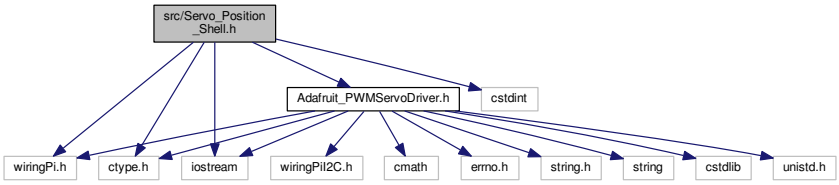
```
00271             } else if(position > 105){
00272                 position = 105;
00273                 pulse = ((((highservoweird - lowservoweird) / dividedconstant)*((double) position)) +
      lowservoweird);
00274             } else {
00275                 pulse = ((((highservoweird - lowservoweird) / dividedconstant)*((double) position)) +
      lowservoweird);
00276             }
00277         }
00278             break;
00279
00280             //Michael Lift 2
00281         case 11:
00282         {
00283             if (position == -1) {
00284                 pulse = 0.0;
00285             } else if(position < 0){
00286                 position = 0;
00287                 pulse = ((((highservospline - lowservospline) / dividedconstant)*((double) position)) +
      lowservospline);
00288             } else if(position > 105){
00289                 position = 105;
00290                 pulse = ((((highservospline - lowservospline) / dividedconstant)*((double) position)) +
      lowservospline);
00291             } else {
00292                 pulse = ((((highservospline - lowservospline) / dividedconstant)*((double) position)) +
      lowservospline);
00293             }
00294         }
00295             break;
00296
00297             //Michael lift 3
00298         case 12:
00299         {
00300             if (position == -1) {
00301                 pulse = 0.0;
00302             } else if(position < 0){
00303                 position = 0;
00304                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00305             } else if(position > 105){
00306                 position = 105;
00307                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00308             } else {
00309                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00310             }
00311         }
00312             break;
00313
00314             // Michael lift 4
00315         case 13:
00316         {
00317             if (position == -1) {
00318                 pulse = 0.0;
00319             } else if(position < 0){
00320                 position = 0;
00321                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00322             } else if(position > 105){
00323                 position = 105;
00324                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00325             } else {
00326                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00327             }
00328         }
00329             break;
00330
00331             // Michael RED GATE
00332         case 14:
00333         {
00334             if (position == -1) {
00335                 pulse = 0.0;
00336             } else if(position < 0){
00337                 position = 0;
00338                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00339             } else if(position > 105){
00340                 position = 105;
00341                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00342             } else {
00343                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00344             }
00345         }
00346             break;
00347     }
00348     setServoPulse(whichservo, pulse);
00349
00350 }
```

6.53 src/Servo_Position_Shell.h File Reference

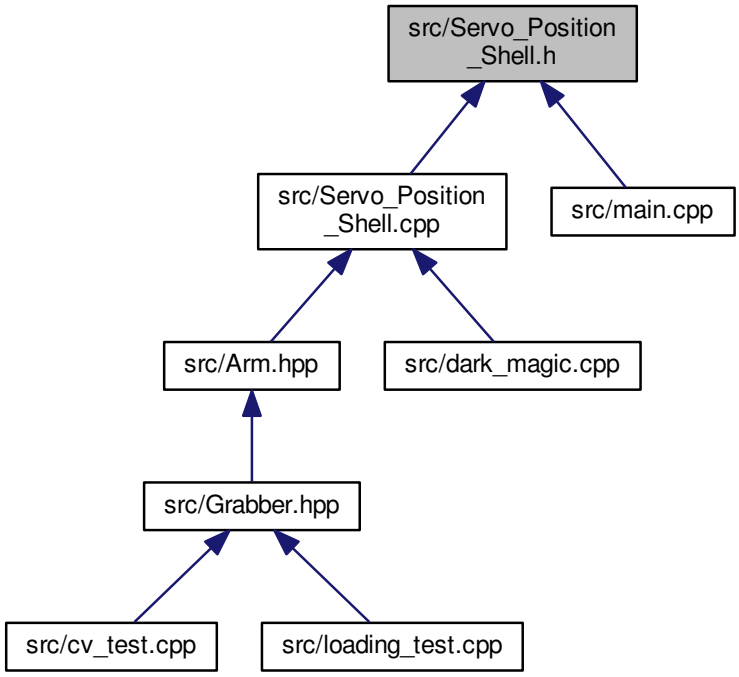
Contains the function prototypes for the servo position shell.

```
#include <wiringPi.h>
#include "Adafruit_PWMServoDriver.h"
#include <iostream>
#include <ctype.h>
#include <cstdint>
```

Include dependency graph for Servo_Position_Shell.h:



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



Enumerations

- enum Servo {
  BASE_TURN = 0, BASE_TILT = 1, ELBOW = 2, WRIST_TILT = 3,
  WRIST_PAN = 4, GRIP_RIGHT = 5, GRIP_LEFT = 6, GATE_1 = 7,
  GATE_2 = 8, GATE_3 = 9, LIFT_1 = 10, LIFT_2 = 11,
  LIFT_3 = 12, LIFT_4 = 13, GATE_4 = 14 }

Functions

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

6.53.1 Detailed Description

Contains the function prototypes for the servo position shell.

Author

Nickolas Neely

Date

8. February 2016, 12:05 PM

Definition in file Servo_Position_Shell.h.

6.53.2 Enumeration Type Documentation

6.53.2.1 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_1

GATE_2

GATE_3

LIFT_1

LIFT_2

LIFT_3

LIFT_4

GATE_4

Definition at line 28 of file Servo_Position_Shell.h.

```
00028           {
00029     BASE_TURN = 0,
00030     BASE_TILT = 1,
00031     ELBOW = 2,
00032     WRIST_TILT = 3,
00033     WRIST_PAN = 4,
00034     GRIP_RIGHT = 5,
00035     GRIP_LEFT = 6,
00036     GATE_1 = 7,
00037     GATE_2 = 8,
00038     GATE_3 = 9,
00039     LIFT_1 = 10,
00040     LIFT_2 = 11,
00041     LIFT_3 = 12,
00042     LIFT_4 = 13,
00043     GATE_4 = 14
00044 };
```

6.53.3 Function Documentation

6.53.3.1 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 71 of file Servo_Position_Shell.cpp.

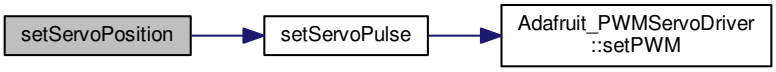
```
00071                                                   {
00072     // works for servo 0, 3, 4
00073     double dividedconstant = 180.0;
00074     double highservo = 2.4;
00075     double lowservo = 0.6;
00076     // To fix the magical digital servo on LIFT 1
00077     double highservoweird = 1.9;
00078     double lowservoweird = 0.6;
00079     // To compensate for the bent servo spline on LIFT 2
00080     double highservospline = 2.25;
00081     double lowservospline = 0.6;
00082     // works for servo 1, 2
00083     double digitalservohigh = 2.45;
00084     double digitalservolow = 0.9;
00085     // left gripper servo 5
00086     double gripleftopen = 2.2;
00087     double gripleftclose = 1.3;
00088     // right gripper servo 6
00089     double griprightopen = 2.2;
00090     double griprightclose = 1.3;
00091     double pulse;
00092
00093     switch (whichservo) {
00094
00095             // BASE TURN
00096         case 0:
00097         {
00098             if (position == -1) {
00099                 pulse = 0.0;
00100             }else if (position < 0){
00101                 position = 20;
00102                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00103             }else if (position > 179){
00104                 position = 179;
00105                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00106             }else{
00107                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00108             }
00109
00110         }
00111
00112             break;
00113
00114             // BASE TILT
00115         case 1:
```

```
00116         {
00117
00118
00119             if (position == -1) {
00120                 pulse = 0.0;
00121             } else if (position < 90){
00122                 position = 90;
00123                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00124             } else if (position > 172){
00125                 position = 172;
00126                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00127             } else {
00128                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00129             }
00130         }
00131             break;
00132
00133             // ELBOW
00134         case 2:
00135         {
00136             if (position == -1) {
00137                 pulse = 0.0;
00138             } else if (position < 43){
00139                 position = 43;
00140                 pulse = ((((digitalservohigh - digitalservolow) / dividedconstant)*((double) position)) +
      digitalservolow);
00141             } else if (position > 179){
00142                 position = 179;
00143                 pulse = ((((digitalservohigh - digitalservolow) / dividedconstant)*((double) position)) +
      digitalservolow);
00144             } else {
00145                 pulse = ((((digitalservohigh - digitalservolow) / dividedconstant)*((double) position)) +
      digitalservolow);
00146             }
00147         }
00148             break;
00149
00150             // WRIST TURN
00151         case 3:
00152         {
00153             if (position == -1) {
00154                 pulse = 0.0;
00155             } else {
00156                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00157             }
00158         }
00159             break;
00160
00161             // WRIST PAN
00162         case 4:
00163         {
00164             if (position == -1) {
00165                 pulse = 0.0;
00166             } else if (position < 0){
00167                 position = 0;
00168                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00169             } else if (position > 180){
00170                 position = 180;
00171                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00172             } else {
00173                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00174             }
00175         }
00176             break;
00177
00178             // GRIP LEFT
00179         case 5:
00180         {
00181             if (position == -1) {
00182                 pulse = 0.0;
00183             } else if(position < 0){
00184                 position = 0;
00185                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00186             }else if(position > 90){
00187                 position = 90;
00188                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00189             } else {
00190                 pulse = ((((gripleftopen - gripleftclose) / dividedconstant)*((double) position)) +
      gripleftclose);
00191             }
00192         }
00193             break;
00194
00195             // GRIP RIGHT
00196         case 6:
```

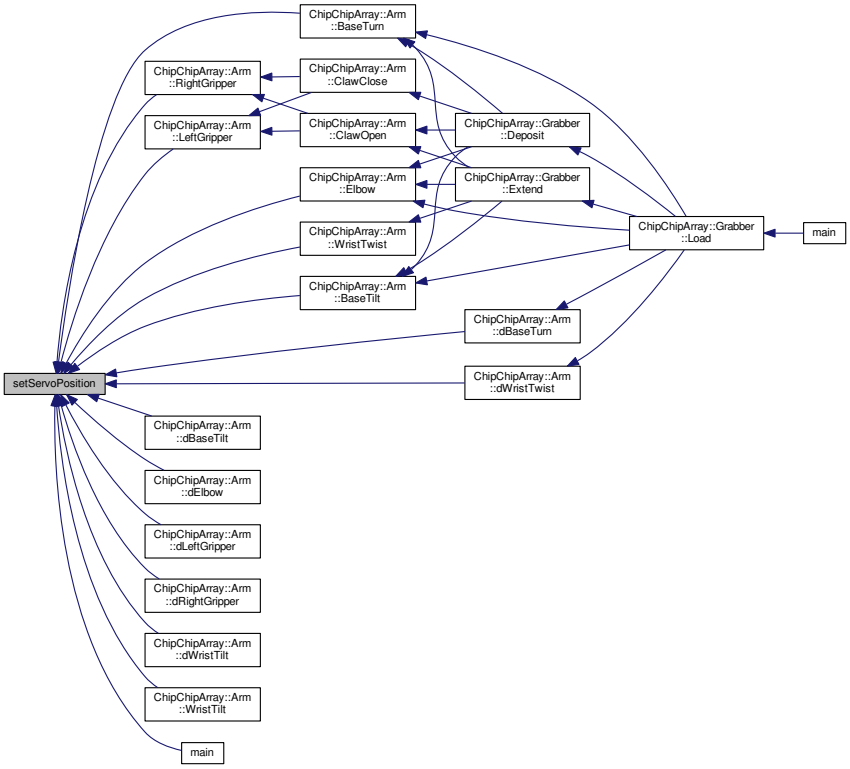
```
00197         {
00198             if (position == -1) {
00199                 pulse = 0.0;
00200             } else if(position < 90){
00201                 position = 90;
00202                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00203             } else if(position > 180){
00204                 position = 180;
00205                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00206             }else{
00207                 pulse = ((((griprightopen - griprightclose) / dividedconstant)*((double) position)) +
      griprightclose);
00208             }
00209         }
00210             break;
00211
00212             // Michael Yellow Gate
00213         case 7:
00214         {
00215             if (position == -1) {
00216                 pulse = 0.0;
00217             } else if(position < 0){
00218                 position = 0;
00219                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00220             } else if(position > 90){
00221                 position = 90;
00222                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00223             } else {
00224                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00225             }
00226         }
00227             break;
00228
00229             // Michael Green Gate
00230         case 8:
00231         {
00232             if (position == -1) {
00233                 pulse = 0.0;
00234             } else if(position < 0){
00235                 position = 0;
00236                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00237             } else if(position > 90){
00238                 position = 90;
00239                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00240             } else {
00241                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00242             }
00243         }
00244             break;
00245
00246             // Michael Blue Gate
00247         case 9:
00248         {
00249             if (position == -1) {
00250                 pulse = 0.0;
00251             } else if(position < 0){
00252                 position = 0;
00253                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00254             } else if(position > 90){
00255                 position = 90;
00256                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00257             } else {
00258                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00259             }
00260         }
00261             break;
00262
00263             // Michael Lift 1
00264         case 10:
00265         {
00266             if (position == -1) {
00267                 pulse = 0.0;
00268             } else if(position < 0){
00269                 position = 0;
00270                 pulse = ((((highservoweird - lowservoweird) / dividedconstant)*((double) position)) +
      lowservoweird);
00271             } else if(position > 105){
00272                 position = 105;
00273                 pulse = ((((highservoweird - lowservoweird) / dividedconstant)*((double) position)) +
      lowservoweird);
00274             } else {
00275                 pulse = ((((highservoweird - lowservoweird) / dividedconstant)*((double) position)) +
      lowservoweird);
00276             }
00277         }
```

```
00278             break;
00279
00280             //Michael Lift 2
00281         case 11:
00282         {
00283             if (position == -1) {
00284                 pulse = 0.0;
00285             } else if(position < 0){
00286                 position = 0;
00287                 pulse = ((((highservospline - lowservospline) / dividedconstant)*((double) position)) +
      lowservospline);
00288             } else if(position > 105){
00289                 position = 105;
00290                 pulse = ((((highservospline - lowservospline) / dividedconstant)*((double) position)) +
      lowservospline);
00291             } else {
00292                 pulse = ((((highservospline - lowservospline) / dividedconstant)*((double) position)) +
      lowservospline);
00293             }
00294         }
00295             break;
00296
00297             //Michael lift 3
00298         case 12:
00299         {
00300             if (position == -1) {
00301                 pulse = 0.0;
00302             } else if(position < 0){
00303                 position = 0;
00304                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00305             } else if(position > 105){
00306                 position = 105;
00307                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00308             } else {
00309                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00310             }
00311         }
00312             break;
00313
00314             // Michael lift 4
00315         case 13:
00316         {
00317             if (position == -1) {
00318                 pulse = 0.0;
00319             } else if(position < 0){
00320                 position = 0;
00321                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00322             } else if(position > 105){
00323                 position = 105;
00324                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00325             } else {
00326                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00327             }
00328         }
00329             break;
00330
00331             // Michael RED GATE
00332         case 14:
00333         {
00334             if (position == -1) {
00335                 pulse = 0.0;
00336             } else if(position < 0){
00337                 position = 0;
00338                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00339             } else if(position > 105){
00340                 position = 105;
00341                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00342             } else {
00343                 pulse = ((((highservo - lowservo) / dividedconstant)*((double) position)) + lowservo);
00344             }
00345         }
00346             break;
00347     }
00348     setServoPulse(whichservo, pulse);
00349
00350 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.53.3.2 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

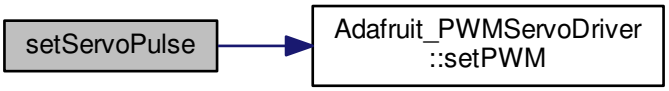
n	which servo on the breakout board am I calling. Starting with 0.
pulse	what is the pulse length (in micro seconds) the pwm of the servo is set to.

Definition at line 50 of file Servo_Position_Shell.cpp.

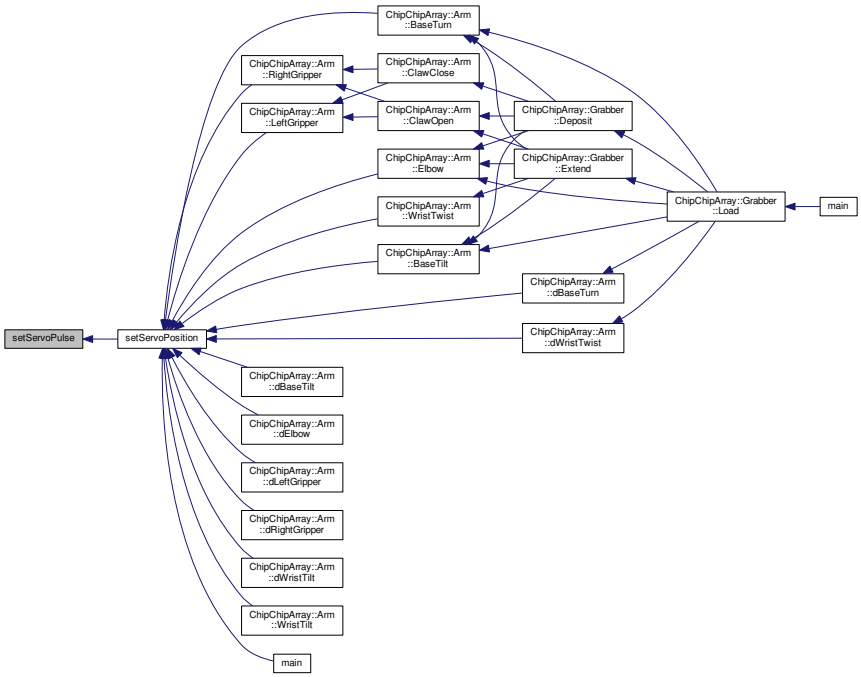
```
00050                                                {
00051     double pulselength;
00052
00053     pulselength = 1000000; // 1,000,000 us per second
00054     pulselength /= 60; // 60 Hz
00055     //cout << pulselength << " us per period" << endl;
00056     pulselength /= 4096; // 12 bits of resolution
```

```
00057     //cout << pulselength << "us per bit" << endl;
00058     pulse *= 1000;
00059     pulse /= pulselength;
00060     //cout << (uint16_t) pulse << endl;
00061     pwm.setPWM(servo_num, 0, (uint16_t) pulse);
00062     //cout << endl;
00063 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



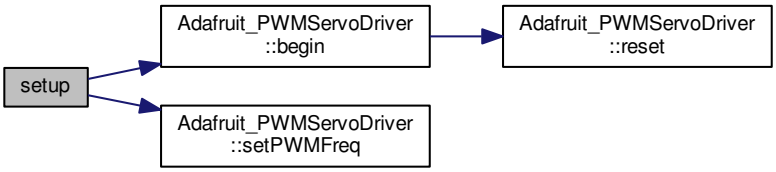
6.53.3.3 void setup ( )

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.

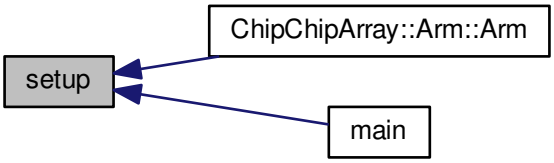
Definition at line 41 of file Servo_Position_Shell.cpp.

```
00041             {
00042     //cout << "Testing Servos" << endl;
00043     pwm.begin();
00044     pwm.setPWMFreq(60.0); // Analog servos run at ~60 Hz updates
00045 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.54 Servo_Position_Shell.h

```
00001
00008 /*
00009  * File: Servo_Position_Shell.h
00010  * Author: Nickolas Neely
00011  *
00012  * Created on February 8, 2016, 12:05 PM
00013  */
00014
00015 #ifndef SERVO_POSITION_SHELL_H
00016 #define SERVO_POSITION_SHELL_H
00017
00018
00019
00020 #include <wiringPi.h>
00021 #include "Adafruit_PWMServoDriver.h"
00022 #include <iostream>
00023 #include <ctype.h>
00024 #include <cstdint>
00025
00028 enum Servo{
00029     BASE_TURN = 0,
00030     BASE_TILT = 1,
00031     ELBOW = 2,
00032     WRIST_TILT = 3,
00033     WRIST_PAN = 4,
00034     GRIP_RIGHT = 5,
00035     GRIP_LEFT = 6,
00036     GATE_1 = 7,
00037     GATE_2 = 8,
00038     GATE_3 = 9,
00039     LIFT_1 = 10,
00040     LIFT_2 = 11,
00041     LIFT_3 = 12,
00042     LIFT_4 = 13,
00043     GATE_4 = 14
00044 };
00045
```

```
00046 #ifdef __cplusplus
00047 extern "C"{
00048 #endif
00049
00057 void setServoPulse(uint8_t n, double pulse);
00063 void setup();
00070 void setServoPosition(Servo whichservo, int position);
00071
00072 #ifdef __cplusplus
00073 }
00074 #endif
00075
00076 #endif  /* SERVO_POSITION_SHELL_H */
00077
```

Index