

RJHS FRC Software Documentation

2011

Generated by Doxygen 1.7.3

Mon Sep 26 2011 11:25:56

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	RJFRC2011::Autonomous Class Reference	5
3.1.1	Detailed Description	7
3.1.2	Constructor & Destructor Documentation	8
3.1.2.1	Autonomous	8
3.1.3	Member Function Documentation	8
3.1.3.1	checkForLines	8
3.1.3.2	correctLeft	8
3.1.3.3	correctRight	8
3.1.3.4	flip	8
3.1.3.5	Go	9
3.1.3.6	initialState	9
3.1.3.7	move	9
3.1.3.8	placeTube	9
3.2	BuiltinDefaultCode Class Reference	9
3.2.1	Detailed Description	12
3.2.2	Constructor & Destructor Documentation	12
3.2.2.1	BuiltinDefaultCode	12
3.3	RJFRC2011::Controller Class Reference	12
3.3.1	Detailed Description	13
3.3.2	Constructor & Destructor Documentation	14
3.3.2.1	Controller	14
3.3.3	Member Function Documentation	14
3.3.3.1	abs	14
3.3.3.2	expo	14
3.3.3.3	getDriveSpeed	15
3.3.3.4	getDriveTurn	15
3.3.3.5	getManipulatorAction	15
3.3.3.6	getManipulatorElevation	15

3.3.3.7	getMinibotSwitches	16
3.3.3.8	normalize	16
3.4	RJFRC2011::Drive Class Reference	16
3.4.1	Detailed Description	17
3.4.2	Constructor & Destructor Documentation	18
3.4.2.1	Drive	18
3.4.3	Member Function Documentation	18
3.4.3.1	drive	18
3.4.3.2	drive_noramp	18
3.4.3.3	ramp	18
3.4.3.4	tank_drive	19
3.5	RJFRC2011::Manipulator Class Reference	19
3.5.1	Detailed Description	21
3.5.2	Constructor & Destructor Documentation	21
3.5.2.1	Manipulator	21
3.5.3	Member Function Documentation	21
3.5.3.1	ejectTube	21
3.5.3.2	elevate	21
3.5.3.3	GetBottomLimitSwitchAddress	22
3.5.3.4	GetTopLimitSwitchAddress	22
3.5.3.5	inputTube	22
3.5.3.6	rotateTube	22
3.5.3.7	stopManipulatorAction	22
3.5.3.8	stopManipulatorElevation	22
3.6	RJFRC2011::Teleoperated Class Reference	23
3.6.1	Detailed Description	25
3.6.2	Constructor & Destructor Documentation	25
3.6.2.1	Teleoperated	25
3.6.3	Member Function Documentation	25
3.6.3.1	Go	25
4	File Documentation	27
4.1	Autonomous.cpp File Reference	27
4.1.1	Detailed Description	28
4.2	Autonomous.h File Reference	29
4.2.1	Detailed Description	30
4.3	AxisCamera.cpp File Reference	30
4.3.1	Detailed Description	31
4.4	AxisCameraParams.cpp File Reference	31
4.4.1	Detailed Description	32
4.5	BuiltinDefaultCode.cpp File Reference	32
4.5.1	Detailed Description	33
4.6	Controller.cpp File Reference	34
4.6.1	Detailed Description	34
4.7	Controller.h File Reference	35
4.7.1	Detailed Description	36

4.8	Drive.cpp File Reference	36
4.8.1	Detailed Description	37
4.9	Drive.h File Reference	37
4.9.1	Detailed Description	39
4.10	macros.h File Reference	39
4.10.1	Detailed Description	43
4.10.2	Function Documentation	43
4.10.2.1	topchar	43
4.11	Manipulator.cpp File Reference	43
4.11.1	Detailed Description	44
4.12	Manipulator.h File Reference	44
4.12.1	Detailed Description	46
4.13	Teleoperated.cpp File Reference	46
4.13.1	Detailed Description	47
4.14	Teleoperated.h File Reference	48
4.14.1	Detailed Description	49

Chapter 1

Class Index

1.1 Class List

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

RJFRC2011::Autonomous (Class managing the Autonomous period of the competition)	5
BuiltinDefaultCode (Our big class that does everything. We didn't bother renaming it)	9
RJFRC2011::Controller (Class abstracting the controller(s) used by our driver(s))	12
RJFRC2011::Drive (Class abstracting the drive system)	16
RJFRC2011::Manipulator (Abstracts the manipulator)	19
RJFRC2011::Teleoperated (Class managing robot performance in Teleoperated mode)	23

Chapter 2

File Index

2.1 File List

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

Autonomous.cpp (File containing implementations of functions found in the <i>Autonomous</i> class (found in Autonomous.h))	27
Autonomous.h (File containing definition of <i>Autonomous</i> class, which is used within the <i>AutonomousPeriodic</i> function in the main program to simplify things)	29
AxisCamera.cpp (File patching buggy code for the <i>AxisCamera</i> class that causes the robot to freeze)	30
AxisCameraParams.cpp (File patching buggy code for the <i>AxisCameraParams</i> class that causes the robot to freeze. All documentation included was provided by Hurler and/or FIRST, though it was reformatted to be compatible with Doxygen)	31
BuiltinDefaultCode.cpp (The file containing the class <i>BuiltinDefaultCode</i> . We didn't bother changing the name..)	32
Controller.cpp (File containing implementations of functions found in the <i>Controller</i> class (found in Controller.h))	34
Controller.h (File containing definition of <i>Controller</i> class, which is used in both Autonomous and Teleoperated modes to get user input)	35
Drive.cpp (File containing definitions of functions declared in class <i>Drive</i> (declared in Drive.h))	36
Drive.h (File containing definition of <i>Drive</i> class, which is used in the main program to drive the robot in both Autonomous and Teleoperated modes with no waste of memory)	37
macros.h (File containing a bunch of macros to de-clutter our other files) . .	39
Manipulator.cpp (File containing implementations of functions found in the <i>Manipulator</i> class (found in Manipulator.h))	43

Manipulator.h (File containing definition of <i>Manipulator</i> class, which is used to simulate the manipulator in both Autonomous and Periodic modes)	44
Teleoperated.cpp (File containing definitions of functions declared in class <i>Teleoperated</i> (declared in Teleoperated.h))	46
Teleoperated.h (File containing definition of <i>Teleoperated</i> class, which is used in the main program during Teleoperated mode to simplify things)	48

Chapter 3

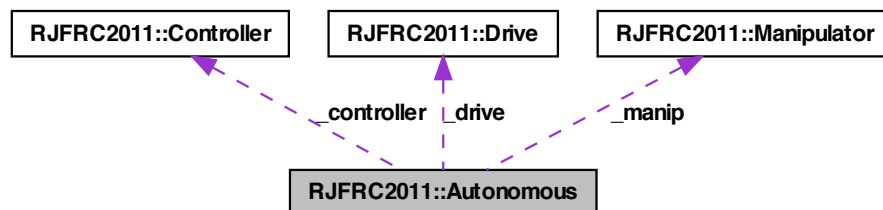
Class Documentation

3.1 RJFRC2011::Autonomous Class Reference

Class managing the [Autonomous](#) period of the competition.

```
#include <Autonomous.h>
```

Collaboration diagram for RJFRC2011::Autonomous:



Public Member Functions

- [Autonomous](#) ([Drive](#) *drive, [Manipulator](#) *manip, [Controller](#) *controller, Driver-StationLCD *screen)

Construct from the addresses of a drive system, a manipulator, a controller, and an LCD screen.

- `~Autonomous ()`
Destructor. Kill stuff.
- `void Go ()`
State machine function that decides which state to go into.
- `void Init ()`
Convenience method. Same as `initialState()`
- `void printLightSensors ()`
Debug info. Prints the values of `lightLeft`, `lightCenter`, and `lightRight` on the screen.

Private Member Functions

- `void initialState ()`
The initial state of the robot.
- `void checkForLines ()`
The state in which the robot is looking for tape lines on the floor.
- `void move ()`
State in which the robot moves forward.
- `void correctRight ()`
State in which the robot moves slightly to the right.
- `void correctLeft ()`
State in which the robot moves slightly to the left.
- `void placeTube ()`
State in which the robot ejects the tube (ideally onto a peg)
- `int flip (int x)`
If `x` is 0, return 1! If `x` is 1, return 0!

Private Attributes

- `UINT8 lane`
The starting lane of the robot; 1 for center, 0 for left or right.

- char [forkDirection](#)
If we're in the center lane: do we go left or right at the fork?
- Drive * [_drive](#)
Pointer to object abstracting drive mechanism.
- Manipulator * [_manip](#)
Pointer to object abstracting manipulator.
- Controller * [_controller](#)
Pointer to object abstracting controller.
- DriverStationLCD * [_screen](#)
Pointer to object representing LCD screen on the Driver Station. Used to print feedback to the screen.
- DigitalInput * [lightSensorLeft](#)
Pointer to object interfacing with leftmost light sensor.
- DigitalInput * [lightSensorCenter](#)
Pointer to object interfacing with center light sensor.
- DigitalInput * [lightSensorRight](#)
Pointer to object interfacing with rightmost light sensor.
- DigitalInput * [autonomousLaneSwitch](#)
Physical switch controlling which lane our robot starts out in.
- DigitalInput * [autonomousForkSwitch](#)
Physical switch controlling which direction our robot is to take if it's in the center lane.

3.1.1 Detailed Description

Class managing the [Autonomous](#) period of the competition. Manages a "state machine" (essentially a *switch* block) that defines which of a number of "states" to execute based on inputs from three light sensors. These states are: check light sensors, move, correct left, correct right, place tube.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 RJFRC2011::Autonomous::Autonomous (Drive * *drive*, Manipulator * *manip*, Controller * *controller*, DriverStationLCD * *screen*)

Construct from the addresses of a drive system, a manipulator, a controller, and an LCD screen.

Parameters

<i>drive</i>	Pointer to a Drive variable that will be used in both Autonomous and Teleoperated periods.
<i>manip</i>	Pointer to a Manipulator variable that will be used in both Autonomous and Teleoperated periods.
<i>controller</i>	Pointer to a Controller variable that will be used in both Autonomous and Teleoperated periods.
<i>screen</i>	Pointer to a DriverStationLCD variable that will be used in Autonomous , Teleoperated , and Disabled periods to write to the screen.

3.1.3 Member Function Documentation

3.1.3.1 void RJFRC2011::Autonomous::checkForLines () [private]

The state in which the robot is looking for tape lines on the floor.

Read input from the 3 light sensors and store the results in three variables.

3.1.3.2 void RJFRC2011::Autonomous::correctLeft () [private]

State in which the robot moves slightly to the left.

Print to the screen that we're in [correctLeft\(\)](#) and turn left by driving the left wheels at -.15 and the right wheels at [AUTO_TURN_SPEED](#).

3.1.3.3 void RJFRC2011::Autonomous::correctRight () [private]

State in which the robot moves slightly to the right.

Print to the screen that we're in [correctRight\(\)](#) and turn right by driving the left wheels at [AUTO_TURN_SPEED](#) and the right wheels at -.15.

3.1.3.4 int RJFRC2011::Autonomous::flip (int x) [private]

If x is 0, return 1! If x is 1, return 0!

Parameters

x	The value to be flipped
-----	-------------------------

Returns

The flipped value

3.1.3.5 void RJFRC2011::Autonomous::Go ()

State machine function that decides which state to go into.

Operates on the *lightX* variables to tell where the line is, with 0 indicating that the sensor saw nothing and 1 indicating that the sensor detected something (ideally the tape). We then use a series of *if* statements to change to the appropriate state.

3.1.3.6 void RJFRC2011::Autonomous::initialState () [private]

The initial state of the robot.

Read input from the switches and set variables accordingly. If lane switch is off, we're in a side lane; if it's on, we're in the center one. If we're in the center, read the second fork switch: if it's on, go left; if not, go right.

3.1.3.7 void RJFRC2011::Autonomous::move () [private]

State in which the robot moves forward.

Print to the screen that we're in *move()* and drive forward at *AUTO_DRIVE_SPEED*

3.1.3.8 void RJFRC2011::Autonomous::placeTube () [private]

State in which the robot ejects the tube (ideally onto a peg)

Brake the robot to a stop, print that we're in *placeTube()*, and run the ejection wheels for one second.

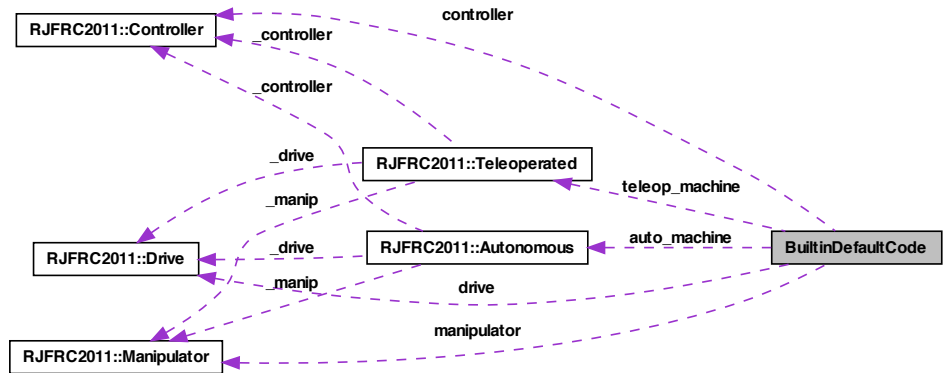
The documentation for this class was generated from the following files:

- [Autonomous.h](#)
- [Autonomous.cpp](#)

3.2 BuiltinDefaultCode Class Reference

Our big class that does everything. We didn't bother renaming it.

Collaboration diagram for BuiltinDefaultCode:



Public Member Functions

- **BuiltinDefaultCode** (void)
Constructor.
- **~BuiltinDefaultCode** ()
Destructor. Called when a class object expires. Used so we don't waste memory.
- void **RobotInit** (void)
Actions which would be performed once (and only once) upon initialization of the robot.
- void **DisabledInit** (void)
Code called at the beginning of Disabled Mode.
- void **AutonomousInit** (void)
Code called at the beginning of Autonomous Mode.
- void **TeleopInit** (void)
Code called at the beginning of Teleoperated Mode.
- void **DisabledPeriodic** (void)
The code called in a loop during Disabled Mode. Left as the default, minus the loop counter.

- void [AutonomousPeriodic](#) (void)
The code called in a loop during Autonomous Mode.
- void [TeleopPeriodic](#) (void)
The code called in a loop during Teleoperated Mode.

Private Attributes

- [Drive](#) * [drive](#)
Class abstracting drive system.
- [Manipulator](#) * [manipulator](#)
Class abstracting manipulator.
- [Controller](#) * [controller](#)
Class abstracting the controller.
- DriverStationLCD * [screen](#)
The screen. Yup.
- DriverStation * [ds](#)
Driver Station object; used so we can send info back to the user (at least, in theory)
- AxisCamera * [cam](#)
Our rear-mounted camera, used to see stuff in style.
- [Autonomous](#) * [auto_machine](#)
Class running the Autonomous period.
- [Teleoperated](#) * [teleop_machine](#)
Class running the Teleoperated period.
- Timer * [timer](#)
Used to time movements in Autonomous.

3.2.1 Detailed Description

Our big class that does everything. We didn't bother renaming it. [BuiltinDefaultCode](#) is technically the name of the example class that we first opened, but once we started coding we realized that we couldn't rename the project and didn't want to reconfigure all the build settings from scratch. So here we are.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 [BuiltinDefaultCode::BuiltinDefaultCode \(void \)](#) `[inline]`

Constructor.

Set up things before robot does anything; mostly this allocates memory for things.

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

- [BuiltinDefaultCode.cpp](#)

3.3 RJFRC2011::Controller Class Reference

Class abstracting the controller(s) used by our driver(s)

```
#include <Controller.h>
```

Public Member Functions

- [Controller \(\)](#)
Constructor.
- [~Controller \(\)](#)
Destructor. Kill stuff.
- float [getManipulatorElevation \(\)](#)
Get user-requested manipulator elevation.
- float [getDriveSpeed \(\)](#)
Get user-requested drive speed.
- float [getDriveTurn \(\)](#)
Get user-requested drive turn speed.
- int [getMinibotSwitches \(\)](#)

Get user-requested minibot shelf action (in/out).

Static Public Member Functions

- static int [getManipulatorAction](#) ()

Get user-requested manipulator action (input, rotate, or eject).

Private Member Functions

- float [abs](#) (float initial)

Absolute value of a float, since I'm not sure if we can import the `<cmath>` library onto the cRIO. EDIT: turns out we can, but I'll leave this here to conserve memory.

- float [expo](#) (float x, float a)

An exponential function used to make joysticks less sensitive near the center and more sensitive towards the edges.

- float [normalize](#) (float joyVal, float min, float max)

A function that "normalizes" inputs from the joysticks (because they don't give perfect -1.0 to 1.0 values).

Private Attributes

- Joystick * [_controller](#)

Object abstracting our InterLink Elite controller, which WIPLib seems to think is a joystick.

- Joystick * [_joystick](#)

Object abstracting our legitimate Logitech Attack3 joystick.

3.3.1 Detailed Description

Class abstracting the controller(s) used by our driver(s) Get input from a pair of joysticks (one of which is physically a flight simulator controller) and return those inputs. A class like this is useful because we can modify a few lines of code to change the inputs for functions throughout the entire code.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 RJFRC2011::Controller::Controller ()

Constructor.

Basically, initialize the controllers to USB ports 1 and 2.

3.3.3 Member Function Documentation

3.3.3.1 float RJFRC2011::Controller::abs (float *initial*) [private]

Absolute value of a float, since I'm not sure if we can import the `<cmath>` library onto the cRIO. EDIT: turns out we can, but I'll leave this here to conserve memory.

Parameters

<i>initial</i>	the initial value
----------------	-------------------

Returns

the absolute value of the passed value; if it's negative, make it positive

Author

Matthew Haney

3.3.3.2 float RJFRC2011::Controller::expo (float *x*, float *a*) [private]

An exponential function used to make joysticks less sensitive near the center and more sensitive towards the edges.

Basically, plug the value requested by the user and a predefined constant into an exponential equation and return the result.

Parameters

<i>x</i>	the value to be exponentiated
<i>a</i>	a predefined exponential factor

Returns

the "expo-ed" value

Author

Adam Bryant

3.3.3.3 float RJFRC2011::Controller::getDriveSpeed ()

Get user-requested drive speed.

Reads input from the y-axis on the right stick, normalizes it to a value between -1.0 and 1.0, and then exponentiates it (so the stick's less sensitive in the center and more sensitive at the edges).

Returns

The received, normalized, and expo-ed value (inverted [if the inversion switch {button #2} is thrown]).

3.3.3.4 float RJFRC2011::Controller::getDriveTurn ()

Get user-requested drive turn speed.

Reads input from the x-axis on the right stick, normalizes it to a value between -1.0 and 1.0, and then exponentiates it (so the stick's less sensitive in the center and more sensitive at the edges).

Returns

The received, normalized, and expo-ed value (inverted [if the inversion switch {button #2} is thrown]).

3.3.3.5 Controller::getManipulatorAction () [static]

Get user-requested manipulator action (input, rotate, or eject).

Read input from three buttons on the joystick: the trigger (1) and two thumb buttons (2 and 3). If any one of them is pressed, return that button's ID. If multiple are pressed, or none are pressed, return 0.

Returns

The ID of the received button.

3.3.3.6 float RJFRC2011::Controller::getManipulatorElevation ()

Get user-requested manipulator elevation.

Reads input from the y-axis on the joystick.

Returns

The received value (inverted).

3.3.3.7 int RJFRC2011::Controller::getMinibotSwitches ()

Get user-requested minibot shelf action (in/out).

Read input from two thumb buttons on the joystick (buttons 4 and 5) and return their states in a single variable.

Returns

An integer with the last two bits being the right and left button inputs, respectively.

3.3.3.8 float RJFRC2011::Controller::normalize (float *joyVal*, float *min*, float *max*) [private]

A function that "normalizes" inputs from the joysticks (because they don't give perfect -1.0 to 1.0 values).

If the requested value is negative, return its percentage of the minimum possible value; if it's possible, do the same with the max. If it's zero, of course, return zero.

Parameters

<i>joyVal</i>	the input from the joystick
<i>min</i>	the minimum joystick value
<i>max</i>	the maximum joystick value

Returns

the normalized value

Author

Adam Bryant

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

- [Controller.h](#)
- [Controller.cpp](#)

3.4 RJFRC2011::Drive Class Reference

Class abstracting the drive system.

```
#include <Drive.h>
```

Public Member Functions

- [Drive](#) ()
Constructor.
- void [drive](#) (float speed, float turn)
Drive the robot, ramping as you go.
- void [drive_noramp](#) (float speed, float turn)
Drive without ramping. Used ONLY in [Autonomous](#), since we don't trust our drivers ;)
- void [tank_drive](#) (float left, float right)
Drive robot from values given for the speed of each wheel; used in autonomous. Regrettably, no ramping here; however, since it won't be sporadic, we don't need it.
- float [ramp](#) (float desired_output, float current_output)
A function that "ramps" input from a joystick to motors so that an overzealous driver doesn't tear up the chassis.

Private Attributes

- RobotDrive * [_drive](#)
WPILib-defined class for abstracting the drive mechanism. Our "Drive" class basically clarifies and ramps the inputs provided by RobotDrive.
- float [_speed_prev](#)
The last sent speed value (used for ramping).
- float [_turn_prev](#)
The last sent turn value (used for ramping).

3.4.1 Detailed Description

Class abstracting the drive system. Basically a wrapper for the WPILib-defined *RobotDrive* class, with our own fancy ramping capabilities put in.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 RJFRC2011::Drive::Drive ()

Constructor.

Initialize the drive system to work with Jaguars on ports *DRIVE_FRONT_LEFT_JAGUAR_PORT*, *DRIVE_FRONT_RIGHT_JAGUAR_PORT*, *DRIVE_BACK_LEFT_JAGUAR_PORT*, and *DRIVE_BACK_RIGHT_JAGUAR_PORT*.

3.4.3 Member Function Documentation

3.4.3.1 void RJFRC2011::Drive::drive (float *speed*, float *turn*)

[Drive](#) the robot, ramping as you go.

Parameters

<i>speed</i>	The user-requested speed
<i>turn</i>	The user-requested turn

[Drive](#) the robot with the *RobotDrive.Drive(speed, turn)* function UNLESS the user wants to do a dead turn (by pushing a stick directly to the left or right), in which case we use the *RobotDrive.TankDrive(left, right)* function.

3.4.3.2 void RJFRC2011::Drive::drive_noramp (float *speed*, float *turn*)

[Drive](#) without ramping. Used ONLY in [Autonomous](#), since we don't trust our drivers ;)

Parameters

<i>speed</i>	The requested speed
<i>turn</i>	The requested turn

[Drive](#) the robot with the *RobotDrive.Drive(speed, turn)* function UNLESS we want to do a dead turn, in which case we use the *RobotDrive.TankDrive(left, right)* function.

3.4.3.3 float RJFRC2011::Drive::ramp (float *desired_output*, float *current_output*)

A function that "ramps" input from a joystick to motors so that an overzealous driver doesn't tear up the chassis.

Increase or decrease the sent value gradually based on operator response. With values close to zero, go even more gradually than normal.

Parameters

<i>desired_-output</i>	The output that the operator is trying to send
<i>current_-output</i>	The current output
<i>increment</i>	The amount by which to increment ramping. Defaults to .005

Returns

the ramped value

Author

Drew Lazzeri

3.4.3.4 void RJFRC2011::Drive::tank_drive (float *left*, float *right*)

[Drive](#) robot from values given for the speed of each wheel; used in autonomous. Regrettably, no ramping here; however, since it won't be sporadic, we don't need it.

Parameters

<i>left</i>	Speed of the left motor
<i>right</i>	Speed of the right motor

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

- [Drive.h](#)
- [Drive.cpp](#)

3.5 RJFRC2011::Manipulator Class Reference

Abstracts the manipulator.

```
#include <Manipulator.h>
```

Public Member Functions

- [Manipulator \(\)](#)
Constructor.
- [~Manipulator \(\)](#)

Destructor. Kill stuff.

- void [inputTube](#) ()
Suck in the tube.
- void [rotateTube](#) ()
Rotate the tube downward.
- void [ejectTube](#) ()
Spit out the tube.
- void [elevate](#) (float val)
Elevate or lower the manipulator.
- void [stopManipulatorAction](#) ()
Stop all movement of the manipulator.
- void [stopManipulatorElevation](#) ()
Stop all elevation of the manipulator.
- DigitalInput * [GetTopLimitSwitchAddress](#) () const
Safely return address of the top limit switch.
- DigitalInput * [GetBottomLimitSwitchAddress](#) () const
Safely return address of the bottom limit switch.

Private Attributes

- Relay * [manipulatorTop](#)
Forwards/backwards relay controlling motion of the top two wheels of the manipulator.
- Relay * [manipulatorBottom](#)
Forwards/backwards relay controlling motion of the bottom two wheels of the manipulator.
- Relay * [manipulatorElevation](#)
Forwards/backwards relay controlling position of the manipulator.
- DigitalInput * [manipulatorElevationBottomLimitSwitch](#)
Limit switch at the bottom of the manipulator elevator.

- DigitalInput * [manipulatorElevationTopLimitSwitch](#)

Limit switch at the top of the manipulator elevator.

3.5.1 Detailed Description

Abstracts the manipulator. Custom class interfacing with the manipulator on our robot through various relays and digitl inputs. Provides multiple methods for easily manipulating the... err... manipulator.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 RJFRC2011::Manipulator::Manipulator ()

Constructor.

Initialize the relays and limit switches to their appropriate ports: see *MANIPULATOR_TOP_RELAY_PORT*, *MANIPULATOR_BOTTOM_RELAY_PORT*, *MANIPULATOR_ELEVATION_RELAY_PORT*, *MANIPULATOR_ELEVATION_BOTTOM_LIMIT_SWITCH_PORT*, and *MANIPULATOR_ELEVATION_TOP_LIMIT_SWITCH_PORT*.

3.5.3 Member Function Documentation

3.5.3.1 void RJFRC2011::Manipulator::ejectTube ()

Spit out the tube.

Set both manipulator relays forward to spit out the tube.

3.5.3.2 void RJFRC2011::Manipulator::elevate (float *val*)

Elevate or lower the manipulator.

If the user is pushing the joystick forward, go down until the lower limit switch is tripped. If they're pulling it backward, go up until the upper limit switch is triggered.

Parameters

<i>val</i>	The user input.
------------	-----------------

3.5.3.3 `DigitalInput* RJFRC2011::Manipulator::GetBottomLimitSwitchAddress () const`
[inline]

Safely return address of the bottom limit switch.

Returns

The address, as a pointer-to-DigitalInput

3.5.3.4 `DigitalInput* RJFRC2011::Manipulator::GetTopLimitSwitchAddress () const`
[inline]

Safely return address of the top limit switch.

Returns

The address, as a pointer-to-DigitalInput

3.5.3.5 `void RJFRC2011::Manipulator::inputTube ()`

Suck in the tube.

Set both manipulator relays to reverse to suck in the tube.

3.5.3.6 `void RJFRC2011::Manipulator::rotateTube ()`

Rotate the tube downward.

Set the top manipulator relay to reverse and the bottom one forward.

3.5.3.7 `void RJFRC2011::Manipulator::stopManipulatorAction ()`

Stop all movement of the manipulator.

Turn off the manipulator relays.

3.5.3.8 `void RJFRC2011::Manipulator::stopManipulatorElevation ()`

Stop all elevation of the manipulator.

Turn off the manipulator elevation relays.

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

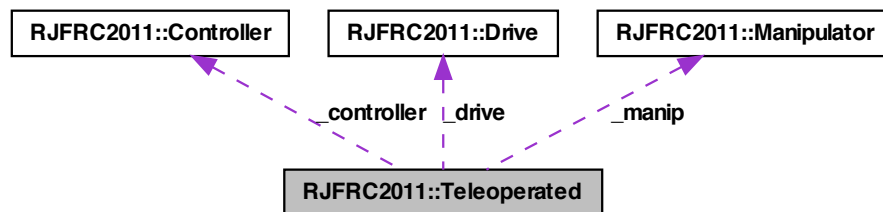
- [Manipulator.h](#)
- [Manipulator.cpp](#)

3.6 RJFRC2011::Teleoperated Class Reference

Class managing robot performance in [Teleoperated](#) mode.

```
#include <Teleoperated.h>
```

Collaboration diagram for RJFRC2011::Teleoperated:



Public Member Functions

- [Teleoperated](#) ([Drive](#) *drive, [Manipulator](#) *manip, [Controller](#) *controller, Driver-StationLCD *screen)

Construct from the addresses of a drive system, a manipulator, a controller, and the driver station LCD screen.

- [~Teleoperated](#) ()

Destructor. Kill stuff.

- void [Init](#) ()

Set up for [Teleoperated](#) mode.

- void [Go](#) ()

This is where the magic happens. Get driver input, then figure out what to do with it.

- void [testLimitSwitches](#) ()

Print the values we're getting from the limit switches to the screen.

Private Attributes

- [Drive](#) * [_drive](#)
Pointer to object abstracting drive system.
- [Manipulator](#) * [_manip](#)
Pointer to object abstracting manipulator.
- [Controller](#) * [_controller](#)
Pointer to object abstracting controller.
- [DriverStationLCD](#) * [_screen](#)
The screen of the driver station.
- [Relay](#) * [minibotShelf](#)
The minibot shelf relay.
- [DigitalInput](#) * [top](#)
Top manipulator limit switch.
- [DigitalInput](#) * [bottom](#)
Bottom manipulator limit switch.
- [float](#) [manipulatorElevation](#)
User input: requested manipulator elevation.
- [float](#) [driveSpeed](#)
User input: requested drive speed.
- [float](#) [driveTurn](#)
User input: requested drive turn.
- [int](#) [manipulatorAction](#)
User input: requested manipulator action.
- [int](#) [minibotSwitches](#)
User input: requested minibot switches.
- [int](#) [_nextState](#)
Used internally (in genericCondition()) to manage state switching.
- [bool](#) [rotateTubeDownFlag](#)
Used to flag that we want to rotate the tube down (after inputting it)

3.6.1 Detailed Description

Class managing robot performance in [Teleoperated](#) mode. Gets user input from two controllers (abstracted as instances of class *Joystick*), processes said inputs (normalizing joystick values, dead space, etc.), and feeds them to a series of outputs (mainly the [Drive](#) class).

3.6.2 Constructor & Destructor Documentation

3.6.2.1 RJFRC2011::Teleoperated::Teleoperated ([Drive](#) * *drive*, [Manipulator](#) * *manip*, [Controller](#) * *controller*, [DriverStationLCD](#) * *screen*)

Construct from the addresses of a drive system, a manipulator, a controller, and the driver station LCD screen.

Set up pointers to passed addresses. Initialize state and condition pointers.

3.6.3 Member Function Documentation

3.6.3.1 void RJFRC2011::Teleoperated::Go ()

This is where the magic happens. Get driver input, then figure out what to do with it.

Get input from the controller (see [Controller](#) class). Based on this, decide what values to send to the [Drive](#) class and [Manipulator](#) class, as well as how to move the relay (*Relay* class) controlling the minibot deployment system.

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

- [Teleoperated.h](#)
- [Teleoperated.cpp](#)

Chapter 4

File Documentation

4.1 Autonomous.cpp File Reference

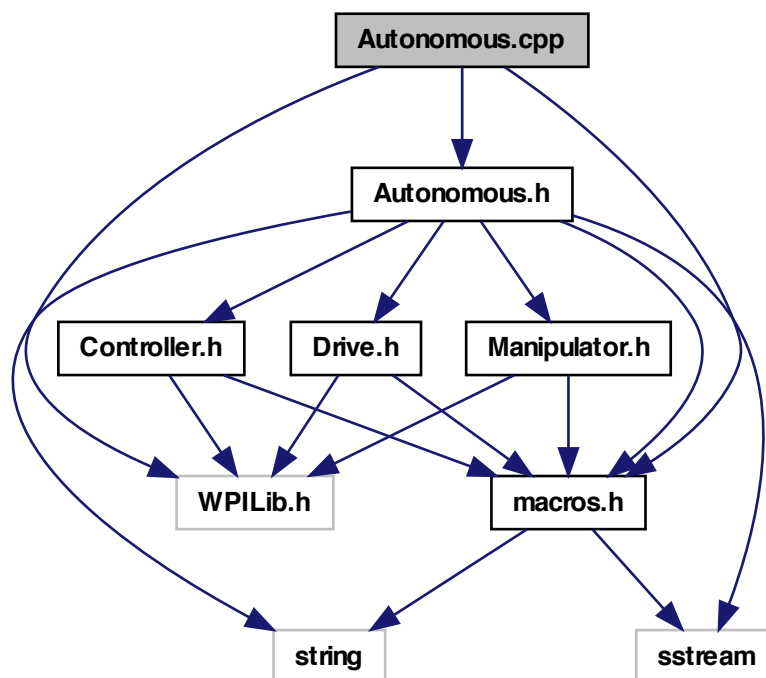
File containing implementations of functions found in the *Autonomous* class (found in [Autonomous.h](#))

```
#include "macros.h"
```

```
#include "WPILib.h"
```

```
#include "Autonomous.h"
```

Include dependency graph for Autonomous.cpp:



4.1.1 Detailed Description

File containing implementations of functions found in the *Autonomous* class (found in [Autonomous.h](#))

Authors

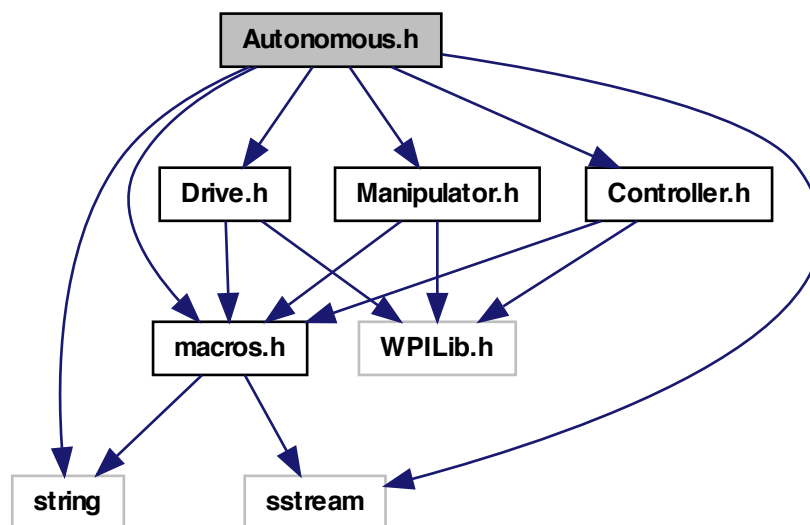
Matthew Haney, Drew Lazzeri

4.2 Autonomous.h File Reference

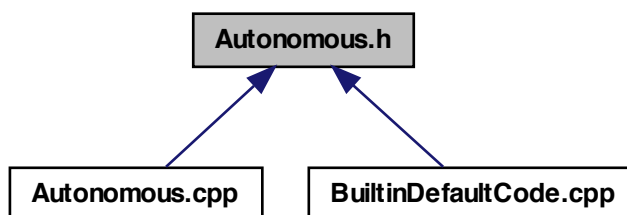
File containing definition of *Autonomous* class, which is used within the *Autonomous-Periodic* function in the main program to simplify things.

```
#include "macros.h"
#include "Drive.h"
#include "Manipulator.h"
#include "Controller.h"
#include <string>
#include <sstream>
```

Include dependency graph for Autonomous.h:



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



Classes

- class [RJFRC2011::Autonomous](#)

Class managing the [Autonomous](#) period of the competition.

4.2.1 Detailed Description

File containing definition of *Autonomous* class, which is used within the *Autonomous-Periodic* function in the main program to simplify things.

Authors

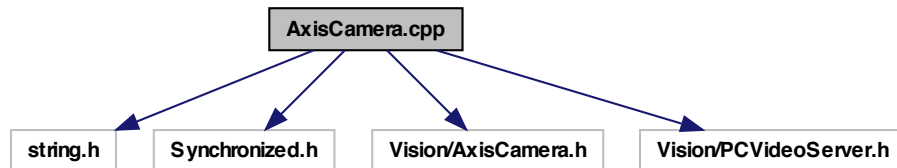
Matthew Haney, Drew Lazzeri

4.3 AxisCamera.cpp File Reference

File patching buggy code for the AxisCamera class that causes the robot to freeze.

```
#include <string.h>
#include "Synchronized.h"
#include "Vision/AxisCamera.h"
#include "Vision/PCVideoServer.h"
```

Include dependency graph for AxisCamera.cpp:



4.3.1 Detailed Description

File patching buggy code for the `AxisCamera` class that causes the robot to freeze.

Authors

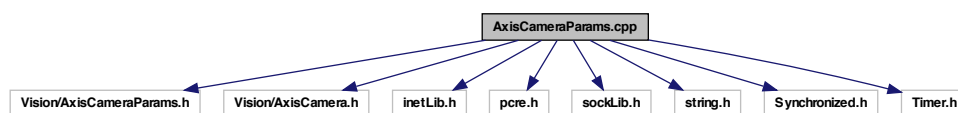
FIRST people, Joe Hurler

4.4 AxisCameraParams.cpp File Reference

File patching buggy code for the `AxisCameraParams` class that causes the robot to freeze. All documentation included was provided by Hurler and/or FIRST, though it was reformatted to be compatible with Doxygen.

```
#include "Vision/AxisCameraParams.h"
#include "Vision/AxisCamera.h"
#include <inetLib.h>
#include "pcre.h"
#include <sockLib.h>
#include <string.h>
#include "Synchronized.h"
#include "Timer.h"
```

Include dependency graph for AxisCameraParams.cpp:



4.4.1 Detailed Description

File patching buggy code for the `AxisCameraParams` class that causes the robot to freeze. All documentation included was provided by Hurler and/or FIRST, though it was reformatted to be compatible with Doxygen.

Authors

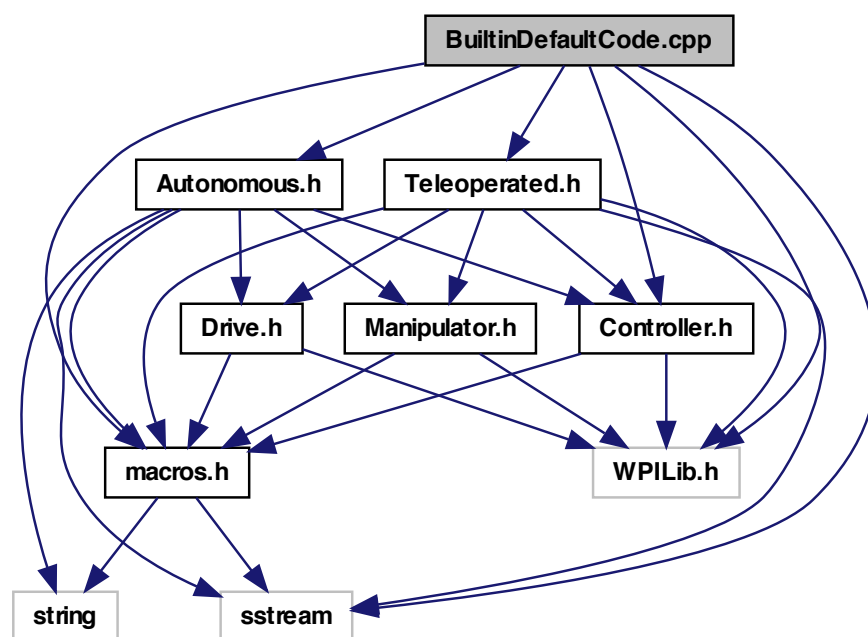
FIRST people, Joe Hurler

4.5 BuiltinDefaultCode.cpp File Reference

The file containing the class [BuiltinDefaultCode](#). We didn't bother changing the name...

```
#include "macros.h"
#include "WPILib.h"
#include "Autonomous.h"
#include "Teleoperated.h"
#include "Controller.h"
#include <sstream>
```

Include dependency graph for BuiltinDefaultCode.cpp:



Classes

- class [BuiltinDefaultCode](#)

Our big class that does everything. We didn't bother renaming it.

4.5.1 Detailed Description

The file containing the class [BuiltinDefaultCode](#). We didn't bother changing the name...
The main source code file for Regis Jesuit High School FRC Team #3729.

Authors

Matthew Haney, Drew Lazzerri

4.6 Controller.cpp File Reference

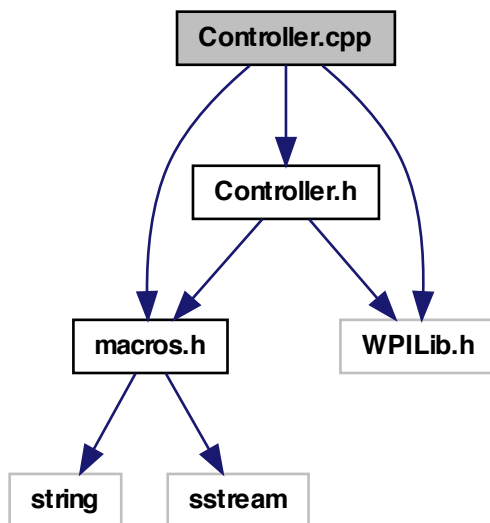
File containing implementations of functions found in the *Controller* class (found in [Controller.h](#))

```
#include "Controller.h"
```

```
#include "WPILib.h"
```

```
#include "macros.h"
```

Include dependency graph for Controller.cpp:



4.6.1 Detailed Description

File containing implementations of functions found in the *Controller* class (found in [Controller.h](#))

Authors

Matthew Haney, Drew Lazzeri

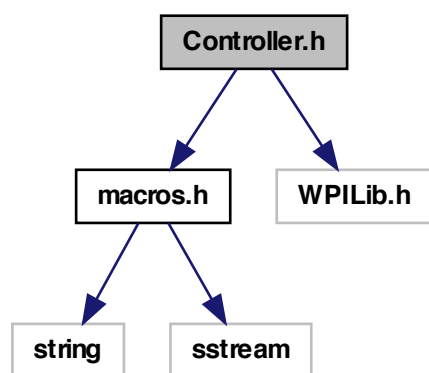
4.7 Controller.h File Reference

File containing definition of *Controller* class, which is used in both Autonomous and Teleoperated modes to get user input.

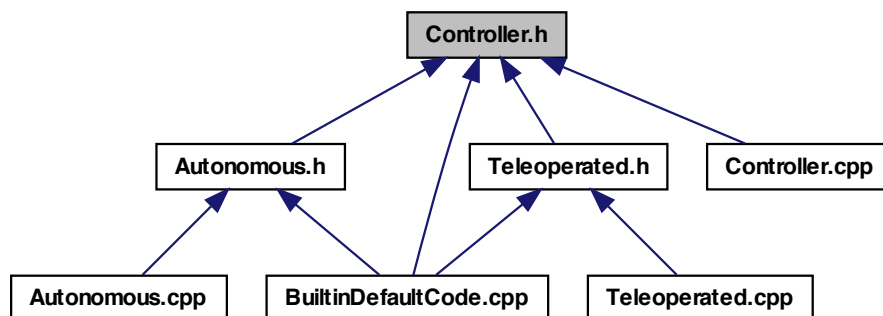
```
#include "macros.h"
```

```
#include "WPILib.h"
```

Include dependency graph for Controller.h:



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



Classes

- class [RJFRC2011::Controller](#)

Class abstracting the controller(s) used by our driver(s)

4.7.1 Detailed Description

File containing definition of *Controller* class, which is used in both Autonomous and Teleoperated modes to get user input.

Authors

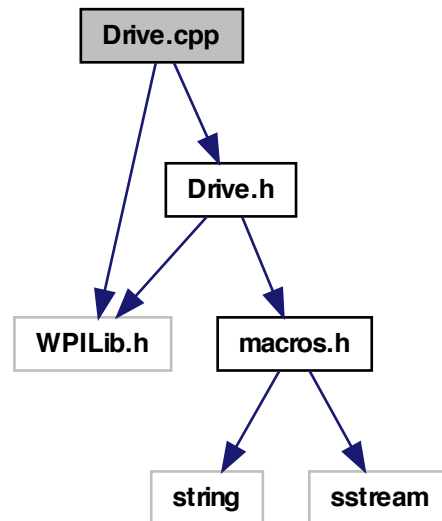
Matthew Haney, Drew Lazzeri

4.8 Drive.cpp File Reference

File containing definitions of functions declared in class *Drive* (declared in [Drive.h](#)).

```
#include "Drive.h"  
#include "WPILib.h"
```

Include dependency graph for Drive.cpp:



4.8.1 Detailed Description

File containing definitions of functions declared in class *Drive* (declared in [Drive.h](#)).

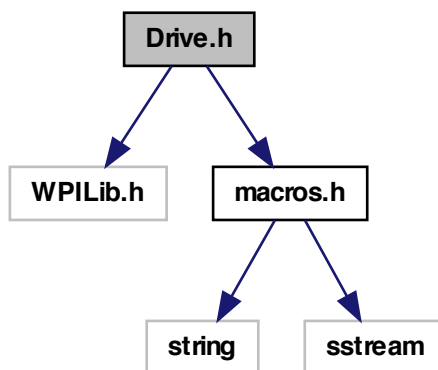
4.9 Drive.h File Reference

File containing definition of *Drive* class, which is used in the main program to drive the robot in both Autonomous and Teleoperated modes with no waste of memory.

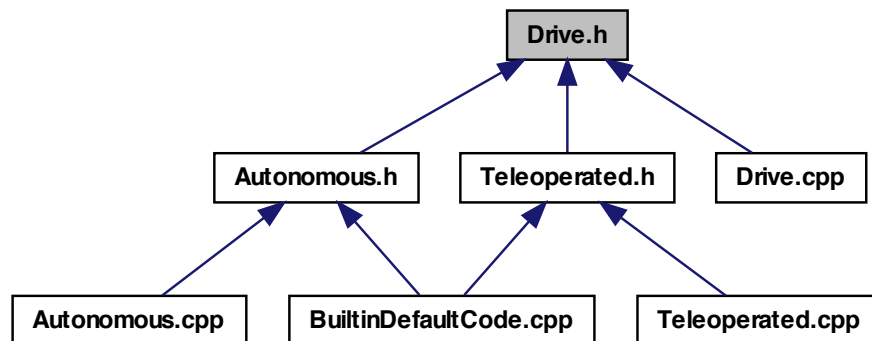
```
#include "WPILib.h"
```

```
#include "macros.h"
```

Include dependency graph for Drive.h:



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



Classes

- class [RJFRC2011::Drive](#)

Class abstracting the drive system.

4.9.1 Detailed Description

File containing definition of *Drive* class, which is used in the main program to drive the robot in both Autonomous and Teleoperated modes with no waste of memory.

Authors

Matthew Haney, Drew Lazzeri

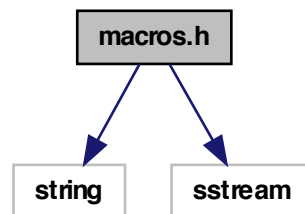
4.10 macros.h File Reference

File containing a bunch of macros to de-clutter our other files.

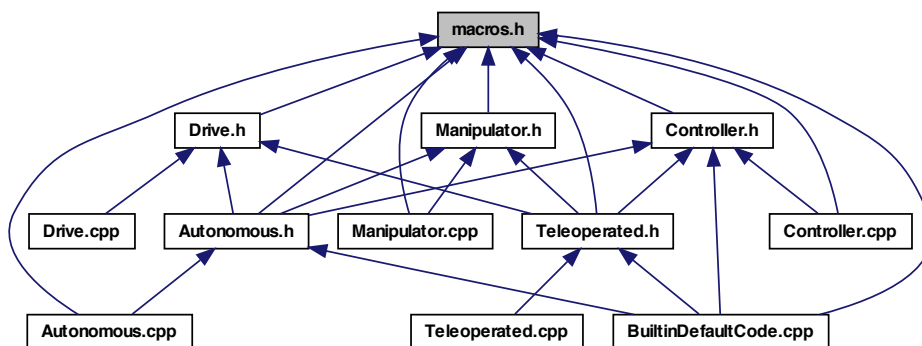
```
#include <string>
```

```
#include <sstream>
```

Include dependency graph for macros.h:



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



Defines

- #define **YCENTER** (0.03125)
Adjustment for the fact that the joystick is slightly off-center.
- #define **ROTCENTER** (0.0156)
Adjustment for the fact that the joystick is slightly off-center.
- #define **XMIN** -0.641
Minimum possible X value.
- #define **XMAX** 0.648
Maximum possible X value.
- #define **YMIN** (-0.57-YCENTER)
Minimum possible Y value.
- #define **YMAX** (0.641-YCENTER)
Maximum possible Y value.
- #define **ZMIN** (-0.54)
Minimum possible Z value.
- #define **ZMAX** (0.63)

Maximum possible Z value.

- #define **ROTMIN** (-0.64-ROTCENTER)
Minimum possible rotation value.
- #define **ROTMAX** (0.68-ROTCENTER)
Maximum possible rotation value.
- #define **XEXPO** 0.4
Exponential constant for modifying input from the x-axis.
- #define **YEXPO** 0.4
Exponential constant for modifying input from the y-axis.
- #define **ROTEXPO** 0.6
Exponential constant for modifying input from the rotational axis.
- #define **AUTONOMOUS_LANE_SWITCH_PORT** 6
Port # for the physical lane choosing switch in Autonomous.
- #define **AUTONOMOUS_FORK_SWITCH_PORT** 7
Port # for the physical fork choosing switch in Autonomous.
- #define **MANIPULATOR_ELEVATION_TOP_LIMIT_SWITCH_PORT** 4
Port # for the limit switch at the top of the manipulator elevator's reach.
- #define **MANIPULATOR_ELEVATION_BOTTOM_LIMIT_SWITCH_PORT** 5
Port # for the limit switch at the bottom of the manipulator elevator's reach.
- #define **LIGHT_SENSOR_LEFT_PORT** 1
Port # of the leftmost line-following light sensor.
- #define **LIGHT_SENSOR_CENTER_PORT** 2
Port # of the center line-following light sensor.
- #define **LIGHT_SENSOR_RIGHT_PORT** 3
Port # of the rightmost line-following light sensor.
- #define **MINIBOT_SHELF_RELAY_PORT** 1
Port # for the relay controlling the minbot shelf.
- #define **MANIPULATOR_TOP_RELAY_PORT** 2

Port # for the relay controlling the top wheels of the manipulator.

- #define MANIPULATOR_BOTTOM_RELAY_PORT 3

Port # for the relay controlling the bottom wheels of the manipulator.

- #define MANIPULATOR_ELEVATION_RELAY_PORT 4

Port # for the relay controlling elevation of the manipulator.

- #define DRIVE_FRONT_LEFT_JAGUAR_PORT 5

Port # of the front left Jaguar on the drive train.

- #define DRIVE_FRONT_RIGHT_JAGUAR_PORT 7

Port # of the front right Jaguar on the drive train.

- #define DRIVE_BACK_LEFT_JAGUAR_PORT 6

Port # of the back left Jaguar on the drive train.

- #define DRIVE_BACK_RIGHT_JAGUAR_PORT 8

Port # of the back right Jaguar on the drive train.

- #define DEFAULT_WATCHDOG_TIME 3.0

The default expiration time of the Watchdog timer, in seconds.

- #define AUTO_DRIVE_SPEED 0.35

Speed at which we drive in Autonomous.

- #define AUTO_TURN_SPEED 0.65

Speed at which we turn in Autonomous.

- #define AUTO_BRAKE_SPEED -0.6

Speed at which we brake in Autonomous.

Functions

- template<typename T >
const char * topchar (T val)

Convert an arbitrary type to a printable string, since this isn't Python.

4.10.1 Detailed Description

File containing a bunch of macros to de-clutter our other files. NOTE: All adjustment macros experimentally made by Adam Bryant.

4.10.2 Function Documentation

4.10.2.1 `template<typename T> const char* topchar (T val)`

Convert an arbitrary type to a printable string, since this isn't Python.

Parameters

<i>val</i>	The number to be converted
------------	----------------------------

Returns

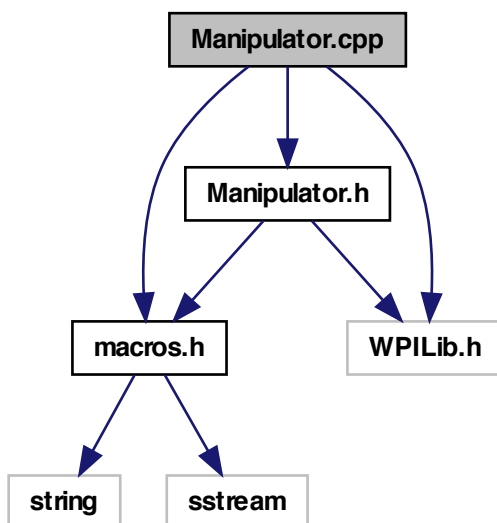
The string representation of the passed number

4.11 Manipulator.cpp File Reference

File containing implementations of functions found in the *Manipulator* class (found in [Manipulator.h](#)).

```
#include "Manipulator.h"
#include "WPILib.h"
#include "macros.h"
```

Include dependency graph for Manipulator.cpp:



4.11.1 Detailed Description

File containing implementations of functions found in the *Manipulator* class (found in [Manipulator.h](#)).

Authors

Matthew Haney, Drew Lazzeri

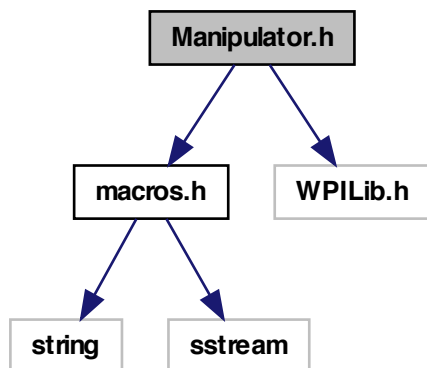
4.12 Manipulator.h File Reference

File containing definition of *Manipulator* class, which is used to simulate the manipulator in both Autonomous and Periodic modes.

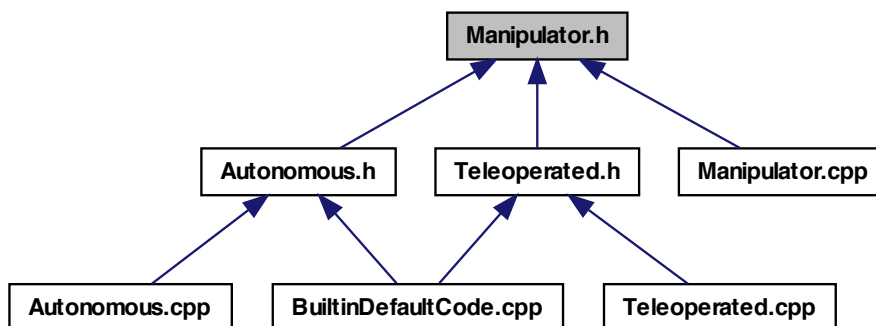
```
#include "macros.h"
```

```
#include "WPILib.h"
```

Include dependency graph for Manipulator.h:



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



Classes

- class [RJFRC2011::Manipulator](#)

Abstracts the manipulator.

4.12.1 Detailed Description

File containing definition of *Manipulator* class, which is used to simulate the manipulator in both Autonomous and Periodic modes.

Authors

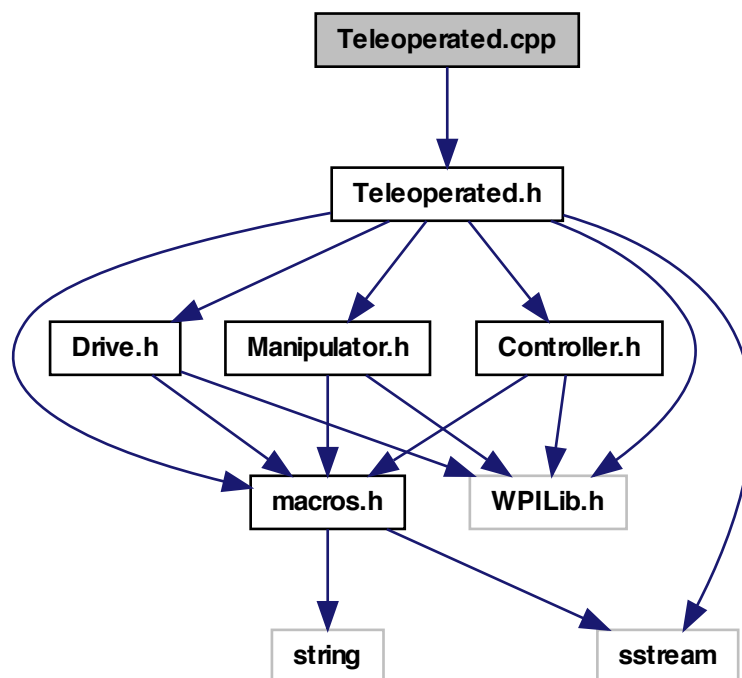
Matthew Haney, Drew Lazzeri

4.13 Teleoperated.cpp File Reference

File containing definitions of functions declared in class *Teleoperated* (declared in [Teleoperated.h](#)).

```
#include "Teleoperated.h"
```

Include dependency graph for Teleoperated.cpp:



4.13.1 Detailed Description

File containing definitions of functions declared in class *Teleoperated* (declared in [Teleoperated.h](#)).

Authors

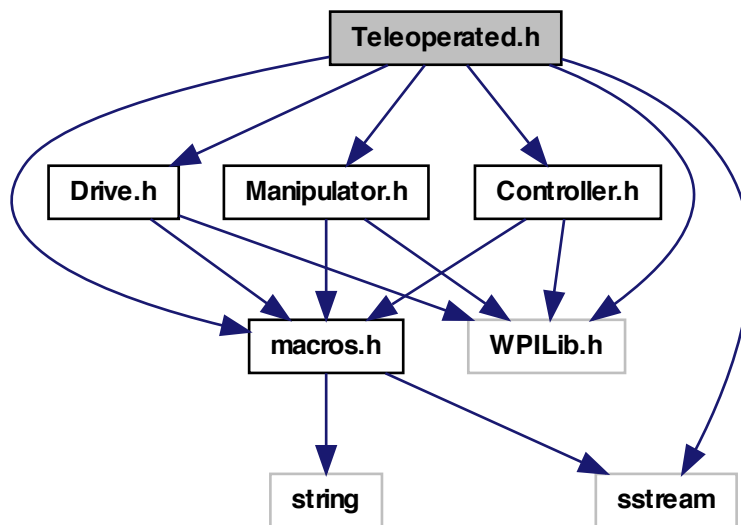
Matthew Haney, Drew Lazzeri

4.14 Teleoperated.h File Reference

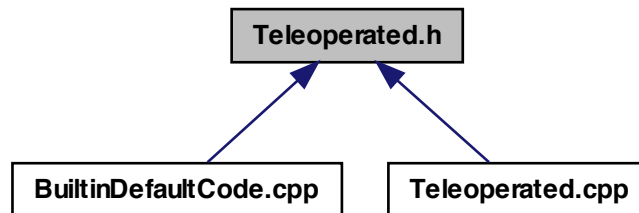
File containing definition of *Teleoperated* class, which is used in the main program during Teleoperated mode to simplify things.

```
#include "macros.h"  
#include "WPILib.h"  
#include "Drive.h"  
#include "Manipulator.h"  
#include "Controller.h"  
#include <sstream>
```

Include dependency graph for Teleoperated.h:



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



Classes

- class [RJFRC2011::Teleoperated](#)
Class managing robot performance in [Teleoperated](#) mode.

4.14.1 Detailed Description

File containing definition of *Teleoperated* class, which is used in the main program during Teleoperated mode to simplify things.

Authors

Matthew Haney, Drew Lazzeri

Index

- abs
 - RJFRC2011::Controller, [14](#)
- Autonomous
 - RJFRC2011::Autonomous, [8](#)
 - Autonomous.cpp, [27](#)
 - Autonomous.h, [29](#)
 - AxisCamera.cpp, [30](#)
 - AxisCameraParams.cpp, [31](#)
- BuiltinDefaultCode, [9](#)
 - BuiltinDefaultCode, [12](#)
- BuiltinDefaultCode.cpp, [32](#)
- checkForLines
 - RJFRC2011::Autonomous, [8](#)
- Controller
 - RJFRC2011::Controller, [14](#)
- Controller.cpp, [34](#)
- Controller.h, [35](#)
- correctLeft
 - RJFRC2011::Autonomous, [8](#)
- correctRight
 - RJFRC2011::Autonomous, [8](#)
- Drive
 - RJFRC2011::Drive, [18](#)
- drive
 - RJFRC2011::Drive, [18](#)
- Drive.cpp, [36](#)
- Drive.h, [37](#)
- drive_noramp
 - RJFRC2011::Drive, [18](#)
- ejectTube
 - RJFRC2011::Manipulator, [21](#)
- elevate
 - RJFRC2011::Manipulator, [21](#)
- expo
 - RJFRC2011::Controller, [14](#)
- flip
 - RJFRC2011::Autonomous, [8](#)
- GetBottomLimitSwitchAddress
 - RJFRC2011::Manipulator, [21](#)
- getDriveSpeed
 - RJFRC2011::Controller, [14](#)
- getDriveTurn
 - RJFRC2011::Controller, [15](#)
- getManipulatorAction
 - RJFRC2011::Controller, [15](#)
- getManipulatorElevation
 - RJFRC2011::Controller, [15](#)
- getMinibotSwitches
 - RJFRC2011::Controller, [15](#)
- GetTopLimitSwitchAddress
 - RJFRC2011::Manipulator, [22](#)
- Go
 - RJFRC2011::Autonomous, [9](#)
 - RJFRC2011::Teleoperated, [25](#)
- initialState
 - RJFRC2011::Autonomous, [9](#)
- inputTube
 - RJFRC2011::Manipulator, [22](#)
- macros.h, [39](#)
 - topchar, [43](#)
- Manipulator
 - RJFRC2011::Manipulator, [21](#)
- Manipulator.cpp, [43](#)
- Manipulator.h, [44](#)
- move
 - RJFRC2011::Autonomous, [9](#)
- normalize

- RJFRC2011::Controller, [16](#)
- placeTube
 - RJFRC2011::Autonomous, [9](#)
- ramp
 - RJFRC2011::Drive, [18](#)
- RJFRC2011::Autonomous, [5](#)
 - Autonomous, [8](#)
 - checkForLines, [8](#)
 - correctLeft, [8](#)
 - correctRight, [8](#)
 - flip, [8](#)
 - Go, [9](#)
 - initialState, [9](#)
 - move, [9](#)
 - placeTube, [9](#)
- RJFRC2011::Controller, [12](#)
 - abs, [14](#)
 - Controller, [14](#)
 - expo, [14](#)
 - getDriveSpeed, [14](#)
 - getDriveTurn, [15](#)
 - getManipulatorAction, [15](#)
 - getManipulatorElevation, [15](#)
 - getMinibotSwitches, [15](#)
 - normalize, [16](#)
- RJFRC2011::Drive, [16](#)
 - Drive, [18](#)
 - drive, [18](#)
 - drive_noramp, [18](#)
 - ramp, [18](#)
 - tank_drive, [19](#)
- RJFRC2011::Manipulator, [19](#)
 - ejectTube, [21](#)
 - elevate, [21](#)
 - GetBottomLimitSwitchAddress, [21](#)
 - GetTopLimitSwitchAddress, [22](#)
 - inputTube, [22](#)
 - Manipulator, [21](#)
 - rotateTube, [22](#)
 - stopManipulatorAction, [22](#)
 - stopManipulatorElevation, [22](#)
- RJFRC2011::Teleoperated, [23](#)
 - Go, [25](#)
 - Teleoperated, [25](#)
- rotateTube
 - RJFRC2011::Manipulator, [22](#)
- stopManipulatorAction
 - RJFRC2011::Manipulator, [22](#)
- stopManipulatorElevation
 - RJFRC2011::Manipulator, [22](#)
- tank_drive
 - RJFRC2011::Drive, [19](#)
- Teleoperated
 - RJFRC2011::Teleoperated, [25](#)
- Teleoperated.cpp, [46](#)
- Teleoperated.h, [48](#)
- topchar
 - macros.h, [43](#)