

PSE_2018_Gruppe1

1.0

Generated by Doxygen 1.8.14

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	AbstractModeController Class Reference	7
4.1.1	Detailed Description	8
4.1.2	Constructor & Destructor Documentation	8
4.1.2.1	<code>~AbstractModeController()</code>	8
4.1.2.2	<code>AbstractModeController()</code>	8
4.1.3	Member Function Documentation	8
4.1.3.1	<code>abort()</code>	8
4.1.3.2	<code>step()</code>	9
4.1.4	Member Data Documentation	9
4.1.4.1	<code>m_driveCommandHandler</code>	9
4.2	AutoModeController Class Reference	10
4.2.1	Detailed Description	11
4.2.2	Constructor & Destructor Documentation	11
4.2.2.1	<code>~AutoModeController()</code>	11
4.2.2.2	<code>AutoModeController()</code>	11

4.2.3	Member Function Documentation	11
4.2.3.1	setDistance()	12
4.2.3.2	step()	12
4.2.4	Member Data Documentation	13
4.2.4.1	DEFAULT_DISTANCE	13
4.2.4.2	m_findMarkerModule	13
4.2.4.3	m_followMarkerModule	13
4.2.4.4	m_obstacleAvoidanceModule	14
4.3	BoundingBox Class Reference	14
4.3.1	Detailed Description	15
4.3.2	Constructor & Destructor Documentation	15
4.3.2.1	BoundingBox() [1/2]	15
4.3.2.2	BoundingBox() [2/2]	16
4.3.3	Member Function Documentation	16
4.3.3.1	areaLt()	16
4.3.3.2	getArea()	16
4.3.3.3	getHeight()	17
4.3.3.4	getRegionID()	17
4.3.3.5	getWidth()	17
4.3.3.6	isOverlapping()	17
4.3.3.7	join()	17
4.3.3.8	setPen()	18
4.3.3.9	x()	18
4.3.3.10	xLt()	18
4.3.3.11	y()	19
4.3.3.12	yLt()	19
4.3.4	Member Data Documentation	19
4.3.4.1	m_Height	19
4.3.4.2	m_IsEmpty	20
4.3.4.3	m_PenColor	20

4.3.4.4	<code>m_RegionID</code>	20
4.3.4.5	<code>m_Width</code>	20
4.3.4.6	<code>m_X</code>	20
4.3.4.7	<code>m_Y</code>	20
4.4	Color Class Reference	21
4.4.1	Detailed Description	21
4.4.2	Constructor & Destructor Documentation	21
4.4.2.1	<code>Color() [1/2]</code>	21
4.4.2.2	<code>Color() [2/2]</code>	22
4.4.3	Member Function Documentation	22
4.4.3.1	<code>b()</code>	22
4.4.3.2	<code>g()</code>	22
4.4.3.3	<code>r()</code>	22
4.4.4	Member Data Documentation	22
4.4.4.1	<code>m_B</code>	23
4.4.4.2	<code>m_G</code>	23
4.4.4.3	<code>m_R</code>	23
4.5	DataProcessModule Class Reference	23
4.5.1	Detailed Description	24
4.5.2	Member Function Documentation	24
4.5.2.1	<code>processData()</code>	24
4.6	DriveCommand Struct Reference	24
4.6.1	Detailed Description	25
4.6.2	Constructor & Destructor Documentation	25
4.6.2.1	<code>DriveCommand()</code>	25
4.6.3	Member Data Documentation	25
4.6.3.1	<code>source</code>	25
4.6.3.2	<code>speed</code>	25
4.6.3.3	<code>steering</code>	26
4.7	DriveCommandHandler Class Reference	26

4.7.1	Detailed Description	27
4.7.2	Constructor & Destructor Documentation	27
4.7.2.1	DriveCommandHandler()	27
4.7.2.2	~DriveCommandHandler()	27
4.7.3	Member Function Documentation	27
4.7.3.1	forceDriveCommand()	27
4.7.3.2	trySetDriveCommand()	28
4.7.3.3	updateMotor()	29
4.7.4	Member Data Documentation	29
4.7.4.1	m_currentDriveCommand	29
4.8	DriveCommandPublisher Class Reference	30
4.8.1	Detailed Description	31
4.8.2	Constructor & Destructor Documentation	31
4.8.2.1	~DriveCommandPublisher()	31
4.8.2.2	DriveCommandPublisher()	31
4.8.3	Member Function Documentation	31
4.8.3.1	addSubscriber()	31
4.8.3.2	forceSubscribers()	32
4.8.3.3	updateSubscribers()	33
4.8.4	Member Data Documentation	33
4.8.4.1	m_driveCommandPublisherType	33
4.8.4.2	m_subscribers	33
4.9	DriveCommandSubscriber Class Reference	34
4.9.1	Detailed Description	34
4.9.2	Constructor & Destructor Documentation	34
4.9.2.1	DriveCommandSubscriber()	34
4.9.2.2	~DriveCommandSubscriber()	35
4.9.3	Member Function Documentation	35
4.9.3.1	forceDriveCommand()	35
4.9.3.2	trySetDriveCommand()	35

4.10 DriveCommandSubscriberMock Class Reference	35
4.10.1 Detailed Description	36
4.10.2 Constructor & Destructor Documentation	37
4.10.2.1 DriveCommandSubscriberMock()	37
4.10.3 Member Function Documentation	37
4.10.3.1 forceDriveCommand()	37
4.10.3.2 getLastDriveCommand()	37
4.10.3.3 trySetDriveCommand()	37
4.10.4 Member Data Documentation	38
4.10.4.1 lastDriveCommand	38
4.11 DriveController Class Reference	38
4.11.1 Detailed Description	39
4.11.2 Constructor & Destructor Documentation	39
4.11.2.1 ~DriveController()	39
4.11.3 Member Function Documentation	39
4.11.3.1 getInstance()	39
4.11.3.2 getSpeed()	40
4.11.3.3 getSteering()	40
4.11.3.4 setTestMode()	40
4.11.3.5 updateController()	41
4.11.3.6 updateDirect()	41
4.11.3.7 updateDirectMotor()	41
4.11.4 Member Data Documentation	41
4.11.4.1 motor_speed	41
4.11.4.2 motor_steering	42
4.12 FileLogger Class Reference	42
4.12.1 Detailed Description	42
4.12.2 Constructor & Destructor Documentation	43
4.12.2.1 ~FileLogger()	43
4.12.2.2 FileLogger() [1/2]	43

4.12.2.3 <code>FileLogger()</code> [2/2]	43
4.12.3 Member Function Documentation	43
4.12.3.1 <code>flushLogs()</code>	44
4.12.3.2 <code>getInstance()</code>	44
4.12.3.3 <code>log()</code>	44
4.12.3.4 <code>operator=()</code>	45
4.12.4 Member Data Documentation	45
4.12.4.1 <code>mLogFile</code>	45
4.12.4.2 <code>m_messageQueue</code>	45
4.13 FindMarkerModule Class Reference	46
4.13.1 Detailed Description	47
4.13.2 Constructor & Destructor Documentation	47
4.13.2.1 <code>FindMarkerModule()</code>	47
4.13.2.2 <code>~FindMarkerModule()</code>	47
4.13.3 Member Function Documentation	48
4.13.3.1 <code>abort()</code>	48
4.13.3.2 <code>foundMarker()</code>	48
4.13.3.3 <code>processData()</code>	49
4.13.3.4 <code>searchBlocking()</code>	50
4.13.3.5 <code>setMaxDriveDuration()</code>	50
4.13.4 Member Data Documentation	51
4.13.4.1 <code>m_maxDuration</code>	51
4.13.4.2 <code>m_stop</code>	51
4.13.4.3 <code>MINIMUM_CONFIDENT</code>	51
4.13.4.4 <code>SEARCH_SPEED</code>	51
4.14 FollowMarkerModule Class Reference	52
4.14.1 Detailed Description	53
4.14.2 Constructor & Destructor Documentation	53
4.14.2.1 <code>FollowMarkerModule()</code>	53
4.14.2.2 <code>~FollowMarkerModule()</code>	53

4.14.3 Member Function Documentation	53
4.14.3.1 processData()	54
4.14.3.2 setDistance()	54
4.14.3.3 setMarkerId()	55
4.14.4 Member Data Documentation	55
4.14.4.1 m_currentMarkerId	55
4.14.4.2 m_distanceController	55
4.14.4.3 m_followsMarker	56
4.14.4.4 m_setDistance	56
4.14.4.5 m_steeringController	56
4.15 Logging Class Reference	56
4.15.1 Detailed Description	56
4.15.2 Member Function Documentation	57
4.15.2.1 log()	57
4.15.2.2 setMaxLevel()	57
4.16 ManualDriveCommandModule Class Reference	58
4.16.1 Detailed Description	59
4.16.2 Constructor & Destructor Documentation	59
4.16.2.1 ManualDriveCommandModule()	60
4.16.2.2 ~ManualDriveCommandModule()	60
4.16.3 Member Function Documentation	60
4.16.3.1 abort()	60
4.16.3.2 driveBlocking()	60
4.16.3.3 setMaxDriveDuration()	61
4.16.4 Member Data Documentation	62
4.16.4.1 m_maxDuration	62
4.16.4.2 m_Stop	62
4.17 ManualModeController Class Reference	62
4.17.1 Detailed Description	63
4.17.2 Constructor & Destructor Documentation	64

4.17.2.1	<code>~ManualModeController()</code>	64
4.17.2.2	<code>ManualModeController()</code>	64
4.17.3	Member Function Documentation	64
4.17.3.1	<code>abort()</code>	64
4.17.3.2	<code>driveBlocking()</code>	65
4.17.3.3	<code>searchMarkerBlocking()</code>	66
4.17.3.4	<code>setMaxDriveDuration()</code>	66
4.17.3.5	<code>step()</code>	67
4.17.4	Member Data Documentation	67
4.17.4.1	<code>DEFAULT_MAX_DRIVE_DURATION</code>	67
4.17.4.2	<code>m_findMarkerModule</code>	67
4.17.4.3	<code>m_isInBlockingCall</code>	68
4.17.4.4	<code>m_manualDriveCommandModule</code>	68
4.18	ObstacleAvoidanceModule Class Reference	68
4.18.1	Detailed Description	69
4.18.2	Constructor & Destructor Documentation	69
4.18.2.1	<code>ObstacleAvoidanceModule()</code>	69
4.18.2.2	<code>~ObstacleAvoidanceModule()</code>	69
4.18.3	Member Function Documentation	69
4.18.3.1	<code>processData()</code>	70
4.18.4	Member Data Documentation	70
4.18.4.1	<code>MIN_DISTANCE</code>	71
4.19	PIDController Class Reference	71
4.19.1	Detailed Description	71
4.19.2	Constructor & Destructor Documentation	71
4.19.2.1	<code>PIDController()</code>	72
4.19.2.2	<code>~PIDController()</code>	73
4.19.3	Member Function Documentation	73
4.19.3.1	<code>calculate()</code>	73
4.19.4	Member Data Documentation	74

4.19.4.1	m_d	74
4.19.4.2	m_dt	74
4.19.4.3	m_error	74
4.19.4.4	m_errorDot	75
4.19.4.5	m_i	75
4.19.4.6	m_integralError	75
4.19.4.7	m_maxVal	75
4.19.4.8	m_minVal	75
4.19.4.9	m_p	75
4.20	SensorManager Class Reference	76
4.20.1	Detailed Description	76
4.20.2	Constructor & Destructor Documentation	77
4.20.2.1	SensorManager() [1/2]	77
4.20.2.2	SensorManager() [2/2]	77
4.20.2.3	~SensorManager()	77
4.20.3	Member Function Documentation	77
4.20.3.1	getDepth()	77
4.20.3.2	getFps()	78
4.20.3.3	getInstance()	78
4.20.3.4	getMarkerList()	78
4.20.3.5	getSensorHeight()	79
4.20.3.6	getSensorWidth()	79
4.20.3.7	operator=()	79
4.20.3.8	runOnce()	80
4.20.3.9	setDebugMarkerInfoList()	80
4.20.3.10	setDepth()	81
4.20.4	Member Data Documentation	81
4.20.4.1	debugMarkerInfoList	81
4.20.4.2	depth	81
4.21	TerminalHandler Class Reference	81

4.21.1 Detailed Description	83
4.21.2 Constructor & Destructor Documentation	83
4.21.2.1 TerminalHandler() [1/2]	83
4.21.2.2 ~TerminalHandler() [1/2]	84
4.21.2.3 TerminalHandler() [2/2]	84
4.21.2.4 ~TerminalHandler() [2/2]	84
4.21.3 Member Function Documentation	85
4.21.3.1 abort() [1/2]	85
4.21.3.2 abort() [2/2]	85
4.21.3.3 evalUserInput() [1/2]	85
4.21.3.4 evalUserInput() [2/2]	86
4.21.3.5 handleDriveCommand() [1/2]	87
4.21.3.6 handleDriveCommand() [2/2]	87
4.21.3.7 handleModeCommand() [1/2]	88
4.21.3.8 handleModeCommand() [2/2]	89
4.21.3.9 handleScriptCommand() [1/2]	90
4.21.3.10 handleScriptCommand() [2/2]	90
4.21.3.11 handleSearchCommand() [1/2]	91
4.21.3.12 handleSearchCommand() [2/2]	91
4.21.3.13 handleSetCommand() [1/2]	92
4.21.3.14 handleSetCommand() [2/2]	93
4.21.3.15 isInAutoMode() [1/2]	94
4.21.3.16 isInAutoMode() [2/2]	94
4.21.3.17 splitString() [1/2]	94
4.21.3.18 splitString() [2/2]	95
4.21.3.19 startInputHandling() [1/2]	95
4.21.3.20 startInputHandling() [2/2]	96
4.21.3.21 tryRunSetupScript() [1/2]	96
4.21.3.22 tryRunSetupScript() [2/2]	97
4.21.4 Member Data Documentation	97

4.21.4.1	DEFAULT_DIAGNOSTICS_MAX_LEVEL	97
4.21.4.2	m_abortScript	98
4.21.4.3	m_autoModeController	98
4.21.4.4	m_handleInput	98
4.21.4.5	m_isInAutoMode	98
4.21.4.6	m_manualModeController	98
4.21.4.7	MAX_RECURSION_DEPTH	98
4.22	TerminalHandlerTest Class Reference	99
4.22.1	Detailed Description	99
4.22.2	Member Function Documentation	100
4.22.2.1	SetUp()	100
4.22.2.2	TearDown()	100
4.22.3	Member Data Documentation	100
4.22.3.1	autoModeController	100
4.22.3.2	driveCommandHandler	100
4.22.3.3	manualModeController	100
4.22.3.4	terminalHandler	100
5	File Documentation	101
5.1	PSE_2018_Gruppe1/src/AbstractModeController.cpp File Reference	101
5.2	PSE_2018_Gruppe1/src/AbstractModeController.hpp File Reference	101
5.3	PSE_2018_Gruppe1/src/AutoModeController.cpp File Reference	102
5.4	PSE_2018_Gruppe1/src/AutoModeController.hpp File Reference	103
5.5	PSE_2018_Gruppe1/src/DataProcessModule.hpp File Reference	104
5.6	PSE_2018_Gruppe1/src/DriveCommand.hpp File Reference	105
5.7	PSE_2018_Gruppe1/src/DriveCommandHandler.cpp File Reference	106
5.8	PSE_2018_Gruppe1/src/DriveCommandHandler.hpp File Reference	106
5.9	PSE_2018_Gruppe1/src/DriveCommandPublisher.cpp File Reference	107
5.10	PSE_2018_Gruppe1/src/DriveCommandPublisher.hpp File Reference	108
5.11	PSE_2018_Gruppe1/src/DriveCommandPublisherType.hpp File Reference	109
5.11.1	Enumeration Type Documentation	109

5.11.1.1	DriveCommandPublisherType	109
5.12	PSE_2018_Gruppe1/src/DriveCommandSubscriber.hpp File Reference	109
5.13	PSE_2018_Gruppe1/src/FileLogger.hpp File Reference	111
5.14	PSE_2018_Gruppe1/src/FindMarkerModule.cpp File Reference	111
5.15	PSE_2018_Gruppe1/src/FindMarkerModule.hpp File Reference	111
5.16	PSE_2018_Gruppe1/src/FollowMarkerModule.cpp File Reference	112
5.17	PSE_2018_Gruppe1/src/FollowMarkerModule.hpp File Reference	113
5.18	PSE_2018_Gruppe1/src/Logging.cpp File Reference	114
5.18.1	Variable Documentation	115
5.18.1.1	s_logLevel	115
5.19	PSE_2018_Gruppe1/src/Logging.hpp File Reference	115
5.19.1	Macro Definition Documentation	116
5.19.1.1	DIAG_DEBUG	116
5.19.1.2	DIAG_ERROR	116
5.19.1.3	DIAG_INFO	117
5.19.1.4	DIAG_VERBOSE	117
5.19.1.5	DIAG_WARNING	117
5.20	PSE_2018_Gruppe1/src/main.cpp File Reference	117
5.20.1	Function Documentation	118
5.20.1.1	main()	118
5.20.1.2	motorStep()	119
5.20.2	Variable Documentation	120
5.20.2.1	g_appRunning	120
5.20.2.2	g_doStep	120
5.20.2.3	s_flushLogInstant	120
5.21	PSE_2018_Gruppe1/src/ManualDriveCommandModule.cpp File Reference	120
5.22	PSE_2018_Gruppe1/src/ManualDriveCommandModule.hpp File Reference	121
5.23	PSE_2018_Gruppe1/src/ManualModeController.cpp File Reference	121
5.24	PSE_2018_Gruppe1/src/ManualModeController.hpp File Reference	122
5.25	PSE_2018_Gruppe1/src/MathFunctions.hpp File Reference	123

5.25.1	Function Documentation	123
5.25.1.1	CheckLimits()	124
5.25.1.2	LimitBetween()	125
5.26	PSE_2018_Gruppe1/src/ObstacleAvoidanceModule.cpp File Reference	126
5.27	PSE_2018_Gruppe1/src/ObstacleAvoidanceModule.hpp File Reference	126
5.28	PSE_2018_Gruppe1/src/PIDController.cpp File Reference	127
5.29	PSE_2018_Gruppe1/src/PIDController.hpp File Reference	128
5.30	PSE_2018_Gruppe1/src/TerminalHandler.cpp File Reference	128
5.31	PSE_2018_Gruppe1/src/TerminalHandler.hpp File Reference	129
5.32	PSE_2018_Gruppe1/src/test/mocks/TerminalHandler.hpp File Reference	130
5.33	PSE_2018_Gruppe1/src/test/DriveCommandHandlerTest.cpp File Reference	131
5.33.1	Function Documentation	132
5.33.1.1	TEST()	132
5.34	PSE_2018_Gruppe1/src/test/FindMarkerModuleTest.cpp File Reference	132
5.34.1	Function Documentation	133
5.34.1.1	TEST() [1/2]	133
5.34.1.2	TEST() [2/2]	134
5.35	PSE_2018_Gruppe1/src/test/FollowMarkerModuleTest.cpp File Reference	134
5.35.1	Function Documentation	135
5.35.1.1	TEST() [1/5]	135
5.35.1.2	TEST() [2/5]	136
5.35.1.3	TEST() [3/5]	136
5.35.1.4	TEST() [4/5]	137
5.35.1.5	TEST() [5/5]	138
5.36	PSE_2018_Gruppe1/src/test/ManualDriveCommandModuleTest.cpp File Reference	139
5.36.1	Function Documentation	140
5.36.1.1	TEST()	140
5.37	PSE_2018_Gruppe1/src/test/MathFunctionsTest.cpp File Reference	140
5.37.1	Function Documentation	141
5.37.1.1	TEST()	141

5.38 PSE_2018_Gruppe1/src/test/mocks/BoundingBox.hpp File Reference	141
5.39 PSE_2018_Gruppe1/src/test/mocks/Diagnostics.cpp File Reference	141
5.39.1 Variable Documentation	142
5.39.1.1 maxLevel	142
5.40 PSE_2018_Gruppe1/src/test/mocks/DriveCommandSubscriberMock.hpp File Reference	142
5.41 PSE_2018_Gruppe1/src/test/mocks/DriveController.hpp File Reference	144
5.42 PSE_2018_Gruppe1/src/test/mocks/SensorManager.hpp File Reference	144
5.43 PSE_2018_Gruppe1/src/test/ObstacleAvoidanceModuleTest.cpp File Reference	145
5.43.1 Function Documentation	146
5.43.1.1 TEST()	146
5.44 PSE_2018_Gruppe1/src/test/PIDControllerTest.cpp File Reference	146
5.44.1 Function Documentation	147
5.44.1.1 TEST()	147
5.45 PSE_2018_Gruppe1/src/test/TerminalHandlerTest.cpp File Reference	148
5.45.1 Function Documentation	149
5.45.1.1 TEST_F() [1/11]	149
5.45.1.2 TEST_F() [2/11]	149
5.45.1.3 TEST_F() [3/11]	149
5.45.1.4 TEST_F() [4/11]	150
5.45.1.5 TEST_F() [5/11]	150
5.45.1.6 TEST_F() [6/11]	150
5.45.1.7 TEST_F() [7/11]	150
5.45.1.8 TEST_F() [8/11]	151
5.45.1.9 TEST_F() [9/11]	151
5.45.1.10 TEST_F() [10/11]	151
5.45.1.11 TEST_F() [11/11]	151
5.46 PSE_2018_Gruppe1/src/test/test.cpp File Reference	152
5.46.1 Function Documentation	152
5.46.1.1 main()	152
5.46.1.2 TEST()	152
Index	153

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AbstractModeController	7
AutoModeController	10
ManualModeController	62
BoundingBox	14
Color	21
DataProcessModule	23
FindMarkerModule	46
FollowMarkerModule	52
ObstacleAvoidanceModule	68
DriveCommand	24
DriveCommandPublisher	30
FindMarkerModule	46
FollowMarkerModule	52
ManualDriveCommandModule	58
ObstacleAvoidanceModule	68
DriveCommandSubscriber	34
DriveCommandHandler	26
DriveCommandSubscriberMock	35
DriveController	38
FileLogger	42
Logging	56
PIDController	71
SensorManager	76
TerminalHandler	81
Test	
TerminalHandlerTest	99

Chapter 2

Class Index

2.1 Class List

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

AbstractModeController	Abstract base class for AutoModeController and ManualModeController	7
AutoModeController	Controls the execution of all three submodules in automatic driving mode	10
BoundingBox	Helper class to track and draw bounding boxes on screen	14
Color	RGB Color	21
DataProcessModule	All automatic mode modules that have to work with data from the SensorManager use this base class as their interface	23
DriveCommand	This struct is used during communication between DriveCommandPublishers and DriveCommandSubscribers	24
DriveCommandHandler	This class can receive DriveCommand structs, prioritizes them by their source and forwards the values to the DriveController once updateMotor() gets called	26
DriveCommandPublisher	Base class for all classes that are able to publish DriveCommands to a DriveCommandSubscriber	30
DriveCommandSubscriber	DriveCommandSubscribers can receive DriveCommand structs from objects that are derived from DriveCommandPublisher	34
DriveCommandSubscriberMock		35
DriveController		38
FileLogger	Class to write log messages to a file called "events.log"	42
FindMarkerModule	This module can be active in manual and in auto mode	46
FollowMarkerModule	This module retrieves the markerList from the SensorManager , checks for markers and tries to follow the one with the best confidence until it loses track of it	52
Logging	This is a wrapper class for Diagnostics	56

ManualDriveCommandModule	
This module is used for blocking drive operations via terminal command or script and can only be invoked via terminal	58
ManualModeController	
Child of AbstractModeController that is used for manual operation	62
ObstacleAvoidanceModule	
Used in automatic mode to detect and avoid obstacles during automatic driving operations	68
PIDController	
Basic universal PID controller implementation	71
SensorManager	
Sensor and image processing manager	76
TerminalHandler	
.	81
TerminalHandlerTest	
.	99

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

PSE_2018_Gruppe1/src/AbstractModeController.cpp	101
PSE_2018_Gruppe1/src/AbstractModeController.hpp	101
PSE_2018_Gruppe1/src/AutoModeController.cpp	102
PSE_2018_Gruppe1/src/AutoModeController.hpp	103
PSE_2018_Gruppe1/src/DataProcessModule.hpp	104
PSE_2018_Gruppe1/src/DriveCommand.hpp	105
PSE_2018_Gruppe1/src/DriveCommandHandler.cpp	106
PSE_2018_Gruppe1/src/DriveCommandHandler.hpp	106
PSE_2018_Gruppe1/src/DriveCommandPublisher.cpp	107
PSE_2018_Gruppe1/src/DriveCommandPublisher.hpp	108
PSE_2018_Gruppe1/src/DriveCommandPublisherType.hpp	109
PSE_2018_Gruppe1/src/DriveCommandSubscriber.hpp	109
PSE_2018_Gruppe1/src/FileLogger.hpp	111
PSE_2018_Gruppe1/src/FindMarkerModule.cpp	111
PSE_2018_Gruppe1/src/FindMarkerModule.hpp	111
PSE_2018_Gruppe1/src/FollowMarkerModule.cpp	112
PSE_2018_Gruppe1/src/FollowMarkerModule.hpp	113
PSE_2018_Gruppe1/src/Logging.cpp	114
PSE_2018_Gruppe1/src/Logging.hpp	115
PSE_2018_Gruppe1/src/main.cpp	117
PSE_2018_Gruppe1/src/ManualDriveCommandModule.cpp	120
PSE_2018_Gruppe1/src/ManualDriveCommandModule.hpp	121
PSE_2018_Gruppe1/src/ManualModeController.cpp	121
PSE_2018_Gruppe1/src/ManualModeController.hpp	122
PSE_2018_Gruppe1/src/MathFunctions.hpp	123
PSE_2018_Gruppe1/src/ObstacleAvoidanceModule.cpp	126
PSE_2018_Gruppe1/src/ObstacleAvoidanceModule.hpp	126
PSE_2018_Gruppe1/src/PIDController.cpp	127
PSE_2018_Gruppe1/src/PIDController.hpp	128
PSE_2018_Gruppe1/src/TerminalHandler.cpp	128
PSE_2018_Gruppe1/src/TerminalHandler.hpp	129
PSE_2018_Gruppe1/src/test/DriveCommandHandlerTest.cpp	131
PSE_2018_Gruppe1/src/test/FindMarkerModuleTest.cpp	132
PSE_2018_Gruppe1/src/test/FollowMarkerModuleTest.cpp	134
PSE_2018_Gruppe1/src/test/ManualDriveCommandModuleTest.cpp	139

PSE_2018_Gruppe1/src/test/MathFunctionsTest.cpp	140
PSE_2018_Gruppe1/src/test/ObstacleAvoidanceModuleTest.cpp	145
PSE_2018_Gruppe1/src/test/PIDControllerTest.cpp	146
PSE_2018_Gruppe1/src/test/TerminalHandlerTest.cpp	148
PSE_2018_Gruppe1/src/test/test.cpp	152
PSE_2018_Gruppe1/src/test/mocks/BoundingBox.hpp	141
PSE_2018_Gruppe1/src/test/mocks/Diagnostics.cpp	141
PSE_2018_Gruppe1/src/test/mocks/DriveCommandSubscriberMock.hpp	142
PSE_2018_Gruppe1/src/test/mocks/DriveController.hpp	144
PSE_2018_Gruppe1/src/test/mocks/SensorManager.hpp	144
PSE_2018_Gruppe1/src/test/mocks/TerminalHandler.hpp	130

Chapter 4

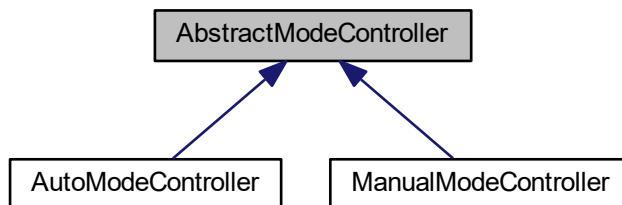
Class Documentation

4.1 AbstractModeController Class Reference

Abstract base class for [AutoModeController](#) and [ManualModeController](#).

```
#include <AbstractModeController.hpp>
```

Inheritance diagram for AbstractModeController:



Public Member Functions

- virtual [~AbstractModeController \(\)](#)
- [AbstractModeController \(std::shared_ptr< DriveCommandHandler > driveCommandHandler\)](#)
Constructor for AbstractModeController.
- virtual void [abort \(\)](#)
Aborts the execution of the AbstractModeController.
- virtual void [step \(\)=0](#)
has to be called in a loop for continuous execution of the controller.

Protected Attributes

- const std::shared_ptr< [DriveCommandHandler](#) > [m_driveCommandHandler](#)

4.1.1 Detailed Description

Abstract base class for [AutoModeController](#) and [ManualModeController](#).

Definition at line 19 of file AbstractModeController.hpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ~AbstractModeController()

```
AbstractModeController::~AbstractModeController ( ) [virtual], [default]
```

4.1.2.2 AbstractModeController()

```
AbstractModeController::AbstractModeController ( std::shared_ptr< DriveCommandHandler > driveCommandHandler ) [explicit]
```

Constructor for [AbstractModeController](#).

All ModeControllers need a [DriveCommandHandler](#) in order to wire up their submodules (publish-subscribe)

Parameters

<code>driveCommandHandler</code>	a pointer to the DriveCommandHandler that was instantiated in main.c
----------------------------------	--

Definition at line 12 of file AbstractModeController.cpp.

4.1.3 Member Function Documentation

4.1.3.1 abort()

```
void AbstractModeController::abort ( ) [virtual]
```

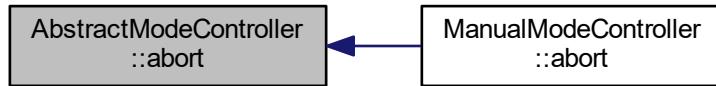
Aborts the execution of the [AbstractModeController](#).

All ModeControllers have to be abortable in case that the user presses ctrl + c

Reimplemented in [ManualModeController](#).

Definition at line 17 of file AbstractModeController.cpp.

Here is the caller graph for this function:



4.1.3.2 step()

```
virtual void AbstractModeController::step ( ) [pure virtual]
```

has to be called in a loop for continuous execution of the controller.

Implemented in [ManualModeController](#), and [AutoModeController](#).

4.1.4 Member Data Documentation

4.1.4.1 m_driveCommandHandler

```
const std::shared_ptr<DriveCommandHandler> AbstractModeController::m_driveCommandHandler  
[protected]
```

Definition at line 43 of file [AbstractModeController.hpp](#).

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

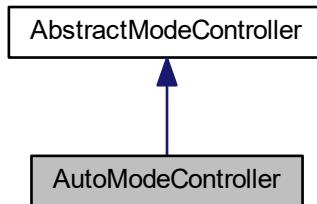
- PSE_2018_Gruppe1/src/[AbstractModeController.hpp](#)
- PSE_2018_Gruppe1/src/[AbstractModeController.cpp](#)

4.2 AutoModeController Class Reference

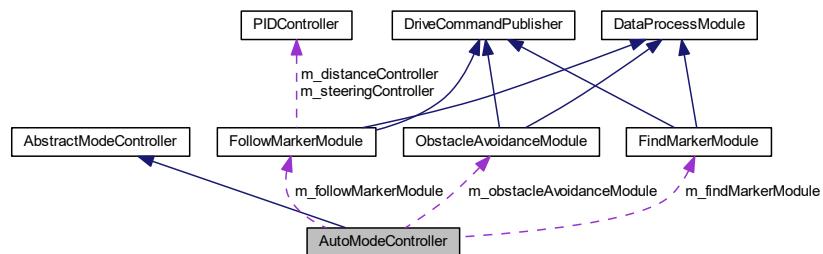
Controls the execution of all three submodules in automatic driving mode.

```
#include <AutoModeController.hpp>
```

Inheritance diagram for AutoModeController:



Collaboration diagram for AutoModeController:



Public Member Functions

- `~AutoModeController () override`
- `AutoModeController (std::shared_ptr< DriveCommandHandler > driveCommandHandler)`
Constructs a new AutoModeController.
- `void setDistance (double distance)`
Sets the desired distance for the MarkerFollowModule.
- `void step () override`
has to be called in a loop for continuous execution of the controller.

Protected Attributes

- `FindMarkerModule m_findMarkerModule`
- `FollowMarkerModule m_followMarkerModule`
- `ObstacleAvoidanceModule m_obstacleAvoidanceModule`

Static Protected Attributes

- static constexpr double `DEFAULT_DISTANCE` = 0.3

4.2.1 Detailed Description

Controls the execution of all three submodules in automatic driving mode.

Definition at line 19 of file AutoModeController.hpp.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 `~AutoModeController()`

```
AutoModeController::~AutoModeController () [override], [default]
```

4.2.2.2 `AutoModeController()`

```
AutoModeController::AutoModeController (
    std::shared_ptr< DriveCommandHandler > driveCommandHandler ) [explicit]
```

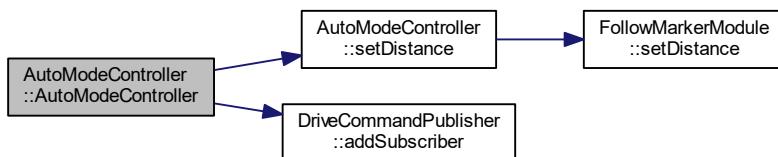
Constructs a new `AutoModeController`.

Parameters

<code>driveCommandHandler</code>	is needed to wire up the three submodules to publish DriveCommands
----------------------------------	--

Definition at line 12 of file AutoModeController.cpp.

Here is the call graph for this function:



4.2.3 Member Function Documentation

4.2.3.1 setDistance()

```
void AutoModeController::setDistance (
    double distance )
```

Sets the desired distance for the MarkerFollowModule.

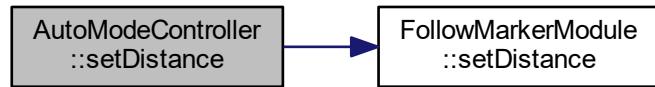
Sets the desired distance to the followed marker.

Parameters

<i>distance</i>	desired distance in [m]
<i>distance</i>	

Definition at line 29 of file AutoModeController.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3.2 step()

```
void AutoModeController::step ( ) [override], [virtual]
```

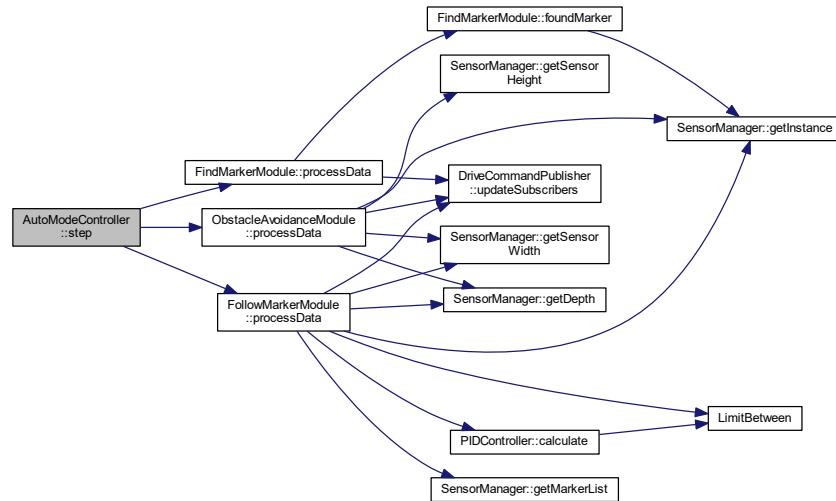
has to be called in a loop for continuous execution of the controller.

Triggers all DriveCommandModules to process the Sensor-Data.

Implements [AbstractModeController](#).

Definition at line 34 of file `AutoModeController.cpp`.

Here is the call graph for this function:



4.2.4 Member Data Documentation

4.2.4.1 DEFAULT_DISTANCE

```
constexpr double AutoModeController::DEFAULT_DISTANCE = 0.3 [static], [protected]
```

Definition at line 46 of file `AutoModeController.hpp`.

4.2.4.2 m_findMarkerModule

```
FindMarkerModule AutoModeController::m_findMarkerModule [protected]
```

Definition at line 42 of file `AutoModeController.hpp`.

4.2.4.3 m_followMarkerModule

```
FollowMarkerModule AutoModeController::m_followMarkerModule [protected]
```

Definition at line 43 of file `AutoModeController.hpp`.

4.2.4.4 m_obstacleAvoidanceModule

`ObstacleAvoidanceModule` AutoModeController::m_obstacleAvoidanceModule [protected]

Definition at line 44 of file AutoModeController.hpp.

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

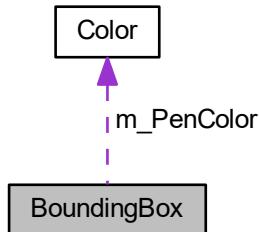
- PSE_2018_Gruppe1/src/AutoModeController.hpp
- PSE_2018_Gruppe1/src/AutoModeController.cpp

4.3 BoundingBox Class Reference

Helper class to track and draw bounding boxes on screen.

```
#include <BoundingBox.hpp>
```

Collaboration diagram for BoundingBox:



Public Member Functions

- `BoundingBox ()`
Default constructor (empty box).
- `BoundingBox (int x, int y, int width, int height, int regionID)`
Parametrized constructor.
- `void join (const BoundingBox &other)`
Expand this box to also bound the other box.
- `bool isOverlapping (const BoundingBox &other) const`
Check if this box overlaps the other box.
- `int x () const`
Get x (left) coordinate.
- `int y () const`
Get y (top) coordinate.
- `int getWidth () const`
Get width.

- int [getHeight \(\) const](#)
Get height.
- int [getArea \(\) const](#)
Get area.
- int [getRegionID \(\) const](#)
Get region id.
- void [setPen \(Color color\)](#)
Set outline color.

Static Public Member Functions

- static bool [xLt \(const BoundingBox &first, const BoundingBox &second\)](#)
- static bool [yLt \(const BoundingBox &first, const BoundingBox &second\)](#)
- static bool [areaLt \(const BoundingBox &first, const BoundingBox &second\)](#)

Protected Attributes

- int [m_X](#)
- int [m_Y](#)
- int [m_Width](#)
- int [m_Height](#)
- int [m_RegionID](#)
- Color [m_PenColor](#)
- bool [m_IsEmpty](#)

4.3.1 Detailed Description

Helper class to track and draw bounding boxes on screen.

Definition at line 37 of file BoundingBox.hpp.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 BoundingBox() [1/2]

```
BoundingBox::BoundingBox ( ) [inline]
```

Default constructor (empty box).

Definition at line 40 of file BoundingBox.hpp.

4.3.2.2 BoundingBox() [2/2]

```
BoundingBox::BoundingBox (
    int x,
    int y,
    int width,
    int height,
    int regionID ) [inline]
```

Parametrized constructor.

Definition at line 45 of file BoundingBox.hpp.

4.3.3 Member Function Documentation

4.3.3.1 areaLt()

```
static bool BoundingBox::areaLt (
    const BoundingBox & first,
    const BoundingBox & second ) [inline], [static]
```

Definition at line 87 of file BoundingBox.hpp.

Here is the call graph for this function:



4.3.3.2 getArea()

```
int BoundingBox::getArea ( ) const [inline]
```

Get area.

Definition at line 73 of file BoundingBox.hpp.

Here is the caller graph for this function:



4.3.3.3 getHeight()

```
int BoundingBox::getHeight ( ) const [inline]
```

Get height.

Definition at line 70 of file BoundingBox.hpp.

4.3.3.4 getRegionID()

```
int BoundingBox::getRegionID ( ) const [inline]
```

Get region id.

Definition at line 78 of file BoundingBox.hpp.

4.3.3.5 getWidth()

```
int BoundingBox::getWidth ( ) const [inline]
```

Get width.

Definition at line 67 of file BoundingBox.hpp.

4.3.3.6 isOverlapping()

```
bool BoundingBox::isOverlapping (
    const BoundingBox & other ) const [inline]
```

Check if this box overlaps the other box.

Definition at line 56 of file BoundingBox.hpp.

4.3.3.7 join()

```
void BoundingBox::join (
    const BoundingBox & other ) [inline]
```

Expand this box to also bound the other box.

Definition at line 51 of file BoundingBox.hpp.

4.3.3.8 setPen()

```
void BoundingBox::setPen (
    Color color ) [inline]
```

Set outline color.

Definition at line 81 of file BoundingBox.hpp.

4.3.3.9 x()

```
int BoundingBox::x ( ) const [inline]
```

Get x (left) coordinate.

Definition at line 61 of file BoundingBox.hpp.

Here is the caller graph for this function:



4.3.3.10 xLt()

```
static bool BoundingBox::xLt (
    const BoundingBox & first,
    const BoundingBox & second ) [inline], [static]
```

Definition at line 83 of file BoundingBox.hpp.

Here is the call graph for this function:



4.3.3.11 y()

```
int BoundingBox::y ( ) const [inline]
```

Get y (top) coordinate.

Definition at line 64 of file BoundingBox.hpp.

Here is the caller graph for this function:



4.3.3.12 yLt()

```
static bool BoundingBox::yLt (const BoundingBox & first, const BoundingBox & second) [inline], [static]
```

Definition at line 85 of file BoundingBox.hpp.

Here is the call graph for this function:



4.3.4 Member Data Documentation

4.3.4.1 m_Height

```
int BoundingBox::m_Height [protected]
```

Definition at line 93 of file BoundingBox.hpp.

4.3.4.2 m_IsEmpty

```
bool BoundingBox::m_IsEmpty [protected]
```

Definition at line 98 of file BoundingBox.hpp.

4.3.4.3 m_PenColor

```
Color BoundingBox::m_PenColor [protected]
```

Definition at line 96 of file BoundingBox.hpp.

4.3.4.4 m_RegionID

```
int BoundingBox::m_RegionID [protected]
```

Definition at line 94 of file BoundingBox.hpp.

4.3.4.5 m_Width

```
int BoundingBox::m_Width [protected]
```

Definition at line 93 of file BoundingBox.hpp.

4.3.4.6 m_X

```
int BoundingBox::m_X [protected]
```

Definition at line 92 of file BoundingBox.hpp.

4.3.4.7 m_Y

```
int BoundingBox::m_Y [protected]
```

Definition at line 92 of file BoundingBox.hpp.

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

- PSE_2018_Gruppe1/src/test/mocks/[BoundingBox.hpp](#)

4.4 Color Class Reference

RGB [Color](#).

```
#include <BoundingBox.hpp>
```

Public Member Functions

- [Color \(\)](#)
Default constructor (black).
- [Color \(unsigned char r, unsigned char g, unsigned char b\)](#)
Parametrized constructor.
- [unsigned char r \(\) const](#)
Get red component.
- [unsigned char g \(\) const](#)
Get green component.
- [unsigned char b \(\) const](#)
Get blue component.

Private Attributes

- [unsigned char m_R](#)
- [unsigned char m_G](#)
- [unsigned char m_B](#)

4.4.1 Detailed Description

RGB [Color](#).

Definition at line 6 of file `BoundingBox.hpp`.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 [Color\(\)](#) [1/2]

```
Color::Color ( ) [inline]
```

Default constructor (black).

Definition at line 9 of file `BoundingBox.hpp`.

4.4.2.2 Color() [2/2]

```
Color::Color (
    unsigned char r,
    unsigned char g,
    unsigned char b ) [inline]
```

Parametrized constructor.

Definition at line 13 of file BoundingBox.hpp.

4.4.3 Member Function Documentation

4.4.3.1 b()

```
unsigned char Color::b () const [inline]
```

Get blue component.

Definition at line 28 of file BoundingBox.hpp.

4.4.3.2 g()

```
unsigned char Color::g () const [inline]
```

Get green component.

Definition at line 23 of file BoundingBox.hpp.

4.4.3.3 r()

```
unsigned char Color::r () const [inline]
```

Get red component.

Definition at line 18 of file BoundingBox.hpp.

4.4.4 Member Data Documentation

4.4.4.1 m_B

```
unsigned char Color::m_B [private]
```

Definition at line 33 of file BoundingBox.hpp.

4.4.4.2 m_G

```
unsigned char Color::m_G [private]
```

Definition at line 33 of file BoundingBox.hpp.

4.4.4.3 m_R

```
unsigned char Color::m_R [private]
```

Definition at line 33 of file BoundingBox.hpp.

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

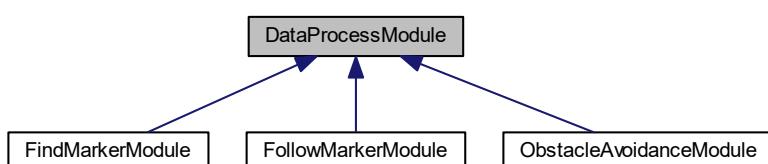
- PSE_2018_Gruppe1/src/test/mock/BoundingBox.hpp

4.5 DataProcessModule Class Reference

All automatic mode modules that have to work with data from the [SensorManager](#) use this base class as their interface.

```
#include <DataProcessModule.hpp>
```

Inheritance diagram for DataProcessModule:



Public Member Functions

- virtual void [processData \(\)=0](#)

Pure virtual interface method that has to be implemented by every module that uses the [SensorManager](#).

4.5.1 Detailed Description

All automatic mode modules that have to work with data from the [SensorManager](#) use this base class as their interface.

Definition at line 16 of file [DataProcessModule.hpp](#).

4.5.2 Member Function Documentation

4.5.2.1 `processData()`

```
virtual void DataProcessModule::processData ( ) [pure virtual]
```

Pure virtual interface method that has to be implemented by every module that uses the [SensorManager](#).

Caller must ensure that [SensorManager::runOnce\(\)](#) has been called before.

Implemented in [FindMarkerModule](#), [FollowMarkerModule](#), and [ObstacleAvoidanceModule](#).

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

- PSE_2018_Gruppe1/src/[DataProcessModule.hpp](#)

4.6 DriveCommand Struct Reference

This struct is used during communication between DriveCommandPublishers and DriveCommandSubscribers.

```
#include <DriveCommand.hpp>
```

Public Member Functions

- [DriveCommand](#) ([DriveCommandPublisherType](#) source, int speed, int steering)
Constructs a new [DriveCommand](#) for publishing to a [DriveCommandSubscriber](#).

Public Attributes

- [DriveCommandPublisherType](#) source
- int speed
the [DriveCommandSubscriber](#) can use this to assign priorities to the modules
- int steering

4.6.1 Detailed Description

This struct is used during communication between DriveCommandPublishers and DriveCommandSubscribers.

Definition at line 15 of file DriveCommand.hpp.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 DriveCommand()

```
DriveCommand::DriveCommand (
    DriveCommandPublisherType source,
    int speed,
    int steering ) [inline]
```

Constructs a new [DriveCommand](#) for publishing to a [DriveCommandSubscriber](#).

Parameters

<code>source</code>	The DriveCommandPublisherType of the publishing object
<code>speed</code>	The desired speed (range: -100...100)
<code>steering</code>	The desired ratio between left motor speed and right motor speed (range: -100...100)

Definition at line 24 of file DriveCommand.hpp.

4.6.3 Member Data Documentation

4.6.3.1 source

```
DriveCommandPublisherType DriveCommand::source
```

Definition at line 31 of file DriveCommand.hpp.

4.6.3.2 speed

```
int DriveCommand::speed
```

the [DriveCommandSubscriber](#) can use this to assign priorities to the modules

Definition at line 32 of file DriveCommand.hpp.

4.6.3.3 steering

```
int DriveCommand::steering
```

Definition at line 33 of file [DriveCommand.hpp](#).

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

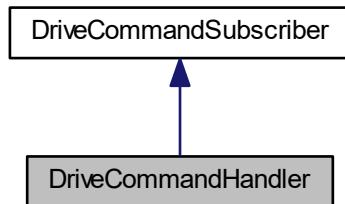
- [PSE_2018_Gruppe1/src/DriveCommand.hpp](#)

4.7 DriveCommandHandler Class Reference

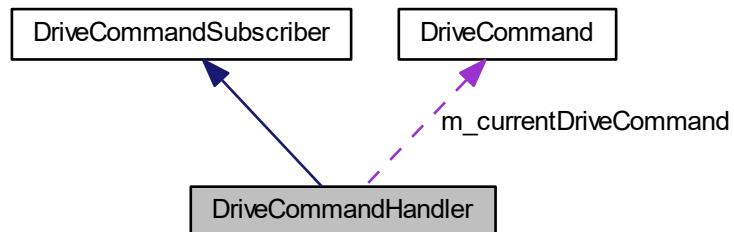
This class can receive [DriveCommand](#) structs, prioritizes them by their source and forwards the values to the [DriveController](#) once [updateMotor\(\)](#) gets called.

```
#include <DriveCommandHandler.hpp>
```

Inheritance diagram for [DriveCommandHandler](#):



Collaboration diagram for [DriveCommandHandler](#):



Public Member Functions

- `DriveCommandHandler ()`
Initialises the currentDriveCommand with MANUAL_COMMAND, (speed)=0, (steering)=0.
- `~DriveCommandHandler () override`
- `void trySetDriveCommand (DriveCommand command) override`
Accepts a DriveCommand from a DriveCommandPublisher.
- `void updateMotor ()`
Flushes the currently held DriveCommand to the DriveController.
- `void forceDriveCommand (DriveCommand command) override`
forces a specific drive command to be executed instead of suggesting it like in trySetDriveCommand()

Private Attributes

- `DriveCommand m_currentDriveCommand`

4.7.1 Detailed Description

This class can receive `DriveCommand` structs, prioritizes them by their source and forwards the values to the `DriveController` once `updateMotor()` gets called.

Definition at line 18 of file `DriveCommandHandler.hpp`.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 `DriveCommandHandler()`

```
DriveCommandHandler::DriveCommandHandler ( )
```

Initialises the currentDriveCommand with MANUAL_COMMAND, (speed)=0, (steering)=0.

Definition at line 11 of file `DriveCommandHandler.cpp`.

4.7.2.2 `~DriveCommandHandler()`

```
DriveCommandHandler::~DriveCommandHandler ( ) [override], [default]
```

4.7.3 Member Function Documentation

4.7.3.1 `forceDriveCommand()`

```
void DriveCommandHandler::forceDriveCommand (
    DriveCommand command ) [override], [virtual]
```

forces a specific drive command to be executed instead of suggesting it like in `trySetDriveCommand()`

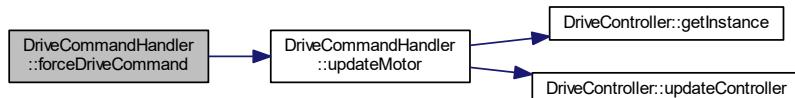
Parameters

<code>command</code>	The DriveCommand to be executed
----------------------	---

Implements [DriveCommandSubscriber](#).

Definition at line 23 of file [DriveCommandHandler.cpp](#).

Here is the call graph for this function:



4.7.3.2 trySetDriveCommand()

```
void DriveCommandHandler::trySetDriveCommand (
    DriveCommand command ) [override], [virtual]
```

Accepts a [DriveCommand](#) from a [DriveCommandPublisher](#).

If its priority is higher than the last [DriveCommand](#) received after the last call to [updateMotor\(\)](#), the old [DriveCommand](#) is discarded and the new one is saved instead.

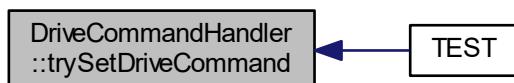
Parameters

<code>command</code>	The DriveCommand as suggested by one of the DriveCommandPublisher s
----------------------	---

Implements [DriveCommandSubscriber](#).

Definition at line 15 of file [DriveCommandHandler.cpp](#).

Here is the caller graph for this function:



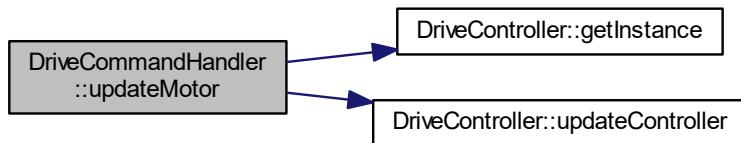
4.7.3.3 updateMotor()

```
void DriveCommandHandler::updateMotor ( )
```

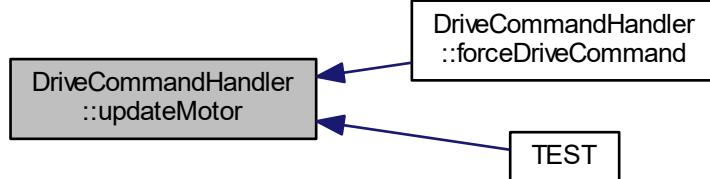
Flushes the currently held [DriveCommand](#) to the [DriveController](#).

Definition at line 29 of file [DriveCommandHandler.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.4 Member Data Documentation

4.7.4.1 m_currentDriveCommand

```
DriveCommand DriveCommandHandler::m_currentDriveCommand [private]
```

Definition at line 48 of file [DriveCommandHandler.hpp](#).

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

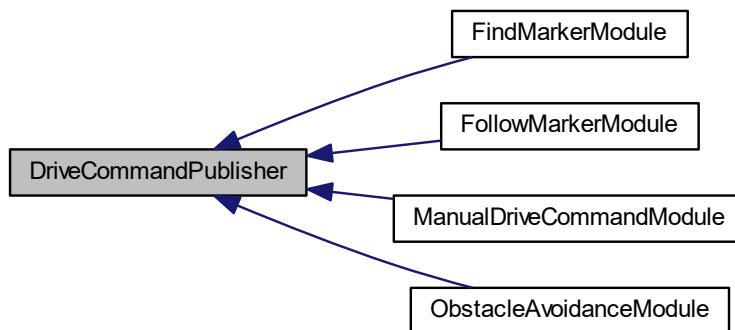
- PSE_2018_Gruppe1/src/[DriveCommandHandler.hpp](#)
- PSE_2018_Gruppe1/src/[DriveCommandHandler.cpp](#)

4.8 DriveCommandPublisher Class Reference

Base class for all classes that are able to publish DriveCommands to a [DriveCommandSubscriber](#).

```
#include <DriveCommandPublisher.hpp>
```

Inheritance diagram for DriveCommandPublisher:



Public Member Functions

- `virtual ~DriveCommandPublisher ()`
- `DriveCommandPublisher (DriveCommandPublisherType type)`
Constructs a new `DriveCommandPublisher` with a given Type.
- `void addSubscriber (std::shared_ptr< DriveCommandSubscriber > subscriber)`
adds a new subscriber to the list of `DriveCommandSubscribers`

Protected Member Functions

- `void updateSubscribers (int speed, int steering)`
Pushes a new `DriveCommand` with this classes' `DriveCommandPublisherType`, speed and steering to all registered subscribers.
- `void forceSubscribers (int speed, int steering)`
Forces a new `DriveCommand` with this classes' `DriveCommandPublisherType`, speed and steering to all registered subscribers.

Private Attributes

- `const DriveCommandPublisherType m_driveCommandPublisherType`
- `std::vector< std::shared_ptr< DriveCommandSubscriber > > m_subscribers`
used for the source field in `DriveCommand` messages.

4.8.1 Detailed Description

Base class for all classes that are able to publish DriveCommands to a [DriveCommandSubscriber](#).

Definition at line 17 of file DriveCommandPublisher.hpp.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ~DriveCommandPublisher()

```
DriveCommandPublisher::~DriveCommandPublisher ( ) [virtual], [default]
```

4.8.2.2 DriveCommandPublisher()

```
DriveCommandPublisher::DriveCommandPublisher ( DriveCommandPublisherType type ) [explicit]
```

Constructs a new [DriveCommandPublisher](#) with a given Type.

Parameters

<code>type</code>	will be used for the "source" field in the DriveCommand struct. Subscribers may prioritize by source type.
-------------------	--

Definition at line 12 of file DriveCommandPublisher.cpp.

4.8.3 Member Function Documentation

4.8.3.1 addSubscriber()

```
void DriveCommandPublisher::addSubscriber ( std::shared_ptr< DriveCommandSubscriber > subscriber )
```

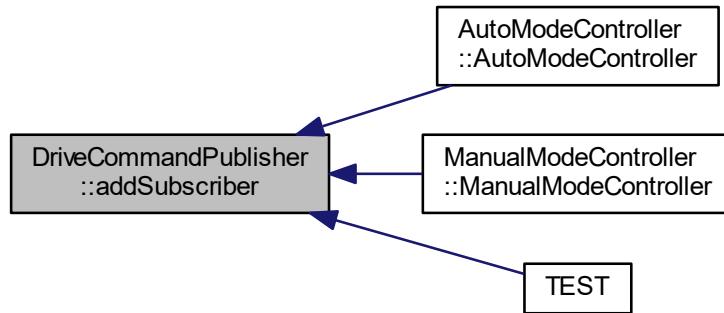
adds a new subscriber to the list of DriveCommandSubscribers

Parameters

<code>subscriber</code>	which class to add to the list
-------------------------	--------------------------------

Definition at line 18 of file DriveCommandPublisher.cpp.

Here is the caller graph for this function:



4.8.3.2 forceSubscribers()

```
void DriveCommandPublisher::forceSubscribers (
    int speed,
    int steering ) [protected]
```

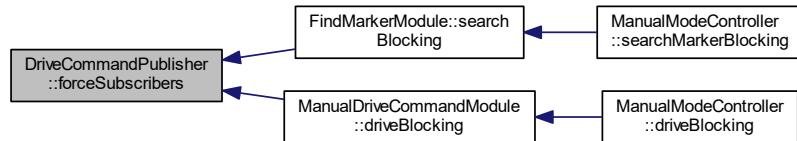
Forces a new [DriveCommand](#) with this classes' `DriveCommandPublisherType`, speed and steering to all registered subscribers.

Parameters

<code>speed</code>	The desired speed that this object wants to set.
<code>steering</code>	The desired steering that this object wants to set.

Definition at line 33 of file `DriveCommandPublisher.cpp`.

Here is the caller graph for this function:



4.8.3.3 updateSubscribers()

```
void DriveCommandPublisher::updateSubscribers (
    int speed,
    int steering ) [protected]
```

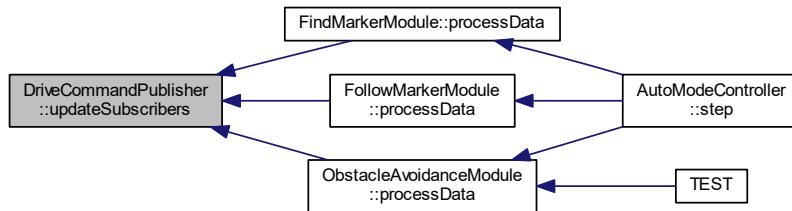
Pushes a new [DriveCommand](#) with this classes' DriveCommandPublisherType, speed and steering to all registered subscribers.

Parameters

<i>speed</i>	The desired speed that this object wants to set.
<i>steering</i>	The desired steering that this object wants to set.

Definition at line 23 of file DriveCommandPublisher.cpp.

Here is the caller graph for this function:



4.8.4 Member Data Documentation

4.8.4.1 m_driveCommandPublisherType

```
const DriveCommandPublisherType DriveCommandPublisher::m_driveCommandPublisherType [private]
```

Definition at line 55 of file DriveCommandPublisher.hpp.

4.8.4.2 m_subscribers

```
std::vector<std::shared_ptr<DriveCommandSubscriber> > DriveCommandPublisher::m_subscribers [private]
```

used for the source field in [DriveCommand](#) messages.

Definition at line 56 of file DriveCommandPublisher.hpp.

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

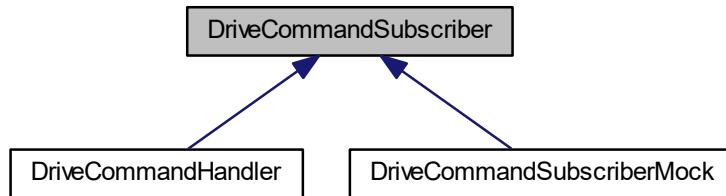
- PSE_2018_Gruppe1/src/[DriveCommandPublisher.hpp](#)
- PSE_2018_Gruppe1/src/[DriveCommandPublisher.cpp](#)

4.9 DriveCommandSubscriber Class Reference

DriveCommandSubscribers can receive [DriveCommand](#) structs from objects that are derived from [DriveCommandPublisher](#).

```
#include <DriveCommandSubscriber.hpp>
```

Inheritance diagram for DriveCommandSubscriber:



Public Member Functions

- [DriveCommandSubscriber \(\)=default](#)
Creates a new [DriveCommandSubscriber](#).
- [virtual ~DriveCommandSubscriber \(\)=default](#)
- [virtual void trySetDriveCommand \(DriveCommand command\)=0](#)
"Receive handler" that gets called from all [DriveCommandPublishers](#) that this [DriveCommandSubscriber](#) is subscribed to.
- [virtual void forceDriveCommand \(DriveCommand command\)=0](#)
"Receive handler" that gets called from a [DriveCommandPublisher](#).

4.9.1 Detailed Description

DriveCommandSubscribers can receive [DriveCommand](#) structs from objects that are derived from [DriveCommandPublisher](#).

Definition at line 16 of file [DriveCommandSubscriber.hpp](#).

4.9.2 Constructor & Destructor Documentation

4.9.2.1 DriveCommandSubscriber()

```
DriveCommandSubscriber::DriveCommandSubscriber ( ) [default]
```

Creates a new [DriveCommandSubscriber](#).

4.9.2.2 ~DriveCommandSubscriber()

```
virtual DriveCommandSubscriber::~DriveCommandSubscriber ( ) [virtual], [default]
```

4.9.3 Member Function Documentation

4.9.3.1 forceDriveCommand()

```
virtual void DriveCommandSubscriber::forceDriveCommand ( DriveCommand command ) [pure virtual]
```

"Receive handler" that gets called from a [DriveCommandPublisher](#).

Forces the MotorController to execute the

Parameters

<i>command.</i>	<input type="button" value=""/>
-----------------	---------------------------------

Implemented in [DriveCommandHandler](#), and [DriveCommandSubscriberMock](#).

4.9.3.2 trySetDriveCommand()

```
virtual void DriveCommandSubscriber::trySetDriveCommand ( DriveCommand command ) [pure virtual]
```

"Receive handler" that gets called from all DriveCommandPublishers that this [DriveCommandSubscriber](#) is subscribed to.

Parameters

<i>command</i>	The DriveCommand that has to be handled
----------------	---

Implemented in [DriveCommandHandler](#), and [DriveCommandSubscriberMock](#).

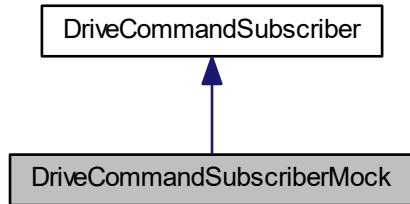
The documentation for this class was generated from the following file:

- PSE_2018_Gruppe1/src/[DriveCommandSubscriber.hpp](#)

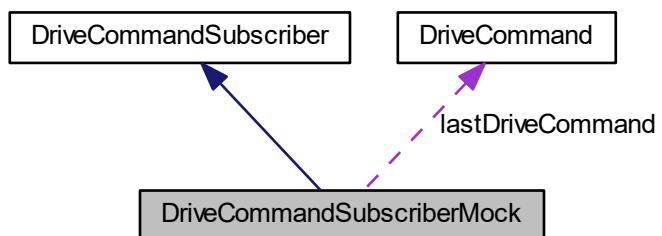
4.10 DriveCommandSubscriberMock Class Reference

```
#include <DriveCommandSubscriberMock.hpp>
```

Inheritance diagram for DriveCommandSubscriberMock:



Collaboration diagram for DriveCommandSubscriberMock:



Public Member Functions

- void [trySetDriveCommand \(DriveCommand command\) override](#)
"Receive handler" that gets called from all DriveCommandPublishers that this DriveCommandSubscriber is subscribed to.
- void [forceDriveCommand \(DriveCommand command\) override](#)
"Receive handler" that gets called from a DriveCommandPublisher.
- [DriveCommand getLastDriveCommand \(\)](#)
- [DriveCommandSubscriberMock \(\)](#)

Private Attributes

- [DriveCommand lastDriveCommand](#)

4.10.1 Detailed Description

Definition at line 6 of file DriveCommandSubscriberMock.hpp.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 DriveCommandSubscriberMock()

```
DriveCommandSubscriberMock::DriveCommandSubscriberMock ( ) [inline]
```

Definition at line 20 of file DriveCommandSubscriberMock.hpp.

4.10.3 Member Function Documentation

4.10.3.1 forceDriveCommand()

```
void DriveCommandSubscriberMock::forceDriveCommand (DriveCommand command) [inline], [override], [virtual]
```

"Receive handler" that gets called from a [DriveCommandPublisher](#).

Forces the MotorController to execute the

Parameters

command.

Implements [DriveCommandSubscriber](#).

Definition at line 12 of file DriveCommandSubscriberMock.hpp.

4.10.3.2 getLastDriveCommand()

```
DriveCommand DriveCommandSubscriberMock::getLastDriveCommand ( ) [inline]
```

Definition at line 16 of file DriveCommandSubscriberMock.hpp.

4.10.3.3 trySetDriveCommand()

```
void DriveCommandSubscriberMock::trySetDriveCommand (DriveCommand command) [inline], [override], [virtual]
```

"Receive handler" that gets called from all DriveCommandPublishers that this [DriveCommandSubscriber](#) is subscribed to.

Parameters

<i>command</i>	The DriveCommand that has to be handled
----------------	---

Implements [DriveCommandSubscriber](#).

Definition at line 8 of file [DriveCommandSubscriberMock.hpp](#).

4.10.4 Member Data Documentation

4.10.4.1 lastDriveCommand

`DriveCommand DriveCommandSubscriberMock::lastDriveCommand [private]`

Definition at line 20 of file [DriveCommandSubscriberMock.hpp](#).

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

- PSE_2018_Gruppe1/src/test/mock/[DriveCommandSubscriberMock.hpp](#)

4.11 DriveController Class Reference

```
#include <DriveController.hpp>
```

Public Member Functions

- virtual [~DriveController](#) ()=default
- bool [updateDirect](#) (int speed, int steering)
- bool [updateDirectMotor](#) (int speedLeft, int speedRight)
- bool [updateController](#) (int speed, int steering)
- int [getSpeed](#) () const
- int [getSteering](#) () const
- void [setTestMode](#) (bool) const

Static Public Member Functions

- static [DriveController](#) * [getInstance](#) ()
Get the singleton instance of the drive controller.

Private Attributes

- int [motor_speed](#)
- int [motor_steering](#)

4.11.1 Detailed Description

Definition at line 10 of file DriveController.hpp.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 ~DriveController()

```
virtual DriveController::~DriveController ( ) [virtual], [default]
```

4.11.3 Member Function Documentation

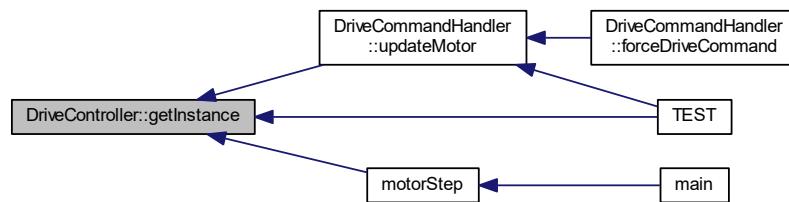
4.11.3.1 getInstance()

```
static DriveController* DriveController::getInstance ( ) [inline], [static]
```

Get the singleton instance of the drive controller.

Definition at line 14 of file DriveController.hpp.

Here is the caller graph for this function:



4.11.3.2 `getSpeed()`

```
int DriveController::getSpeed ( ) const [inline]
```

Definition at line 34 of file DriveController.hpp.

Here is the caller graph for this function:



4.11.3.3 `getSteering()`

```
int DriveController::getSteering ( ) const [inline]
```

Definition at line 38 of file DriveController.hpp.

Here is the caller graph for this function:



4.11.3.4 `setTestMode()`

```
void DriveController::setTestMode ( bool ) const [inline]
```

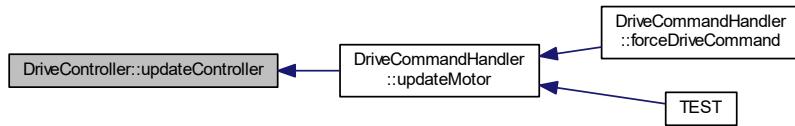
Definition at line 42 of file DriveController.hpp.

4.11.3.5 updateController()

```
bool DriveController::updateController (
    int speed,
    int steering ) [inline]
```

Definition at line 28 of file DriveController.hpp.

Here is the caller graph for this function:



4.11.3.6 updateDirect()

```
bool DriveController::updateDirect (
    int speed,
    int steering ) [inline]
```

Definition at line 22 of file DriveController.hpp.

4.11.3.7 updateDirectMotor()

```
bool DriveController::updateDirectMotor (
    int speedLeft,
    int speedRight ) [inline]
```

Definition at line 25 of file DriveController.hpp.

4.11.4 Member Data Documentation

4.11.4.1 motor_speed

```
int DriveController::motor_speed [private]
```

Definition at line 47 of file DriveController.hpp.

4.11.4.2 motor_steering

```
int DriveController::motor_steering [private]
```

Definition at line 48 of file DriveController.hpp.

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

- PSE_2018_Gruppe1/src/test/mocks/DriveController.hpp

4.12 FileLogger Class Reference

Class to write log messages to a file called "events.log".

```
#include <FileLogger.hpp>
```

Public Member Functions

- void [flushLogs \(\)](#)
Takes the content of m_messageQueue and writes them to the log file.
- [~FileLogger \(\)](#)
- void [log \(std::string message\)](#)
pushes the message to the message queue

Static Public Member Functions

- static [FileLogger & getInstance \(\)](#)
Singleton getInstance()

Private Member Functions

- [FileLogger \(\)](#)
- [FileLogger \(const FileLogger &\) = default](#)
- [FileLogger & operator= \(const FileLogger &\)](#)

Private Attributes

- FILE * [mLogFile](#)
- std::queue< std::string > [m_messageQueue](#)

4.12.1 Detailed Description

Class to write log messages to a file called "events.log".

Messages are buffered in a queue and are written to the file when [flushLogs\(\)](#) is called or when the singleton instance gets destroyed

Definition at line 11 of file FileLogger.hpp.

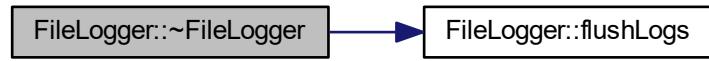
4.12.2 Constructor & Destructor Documentation

4.12.2.1 ~FileLogger()

```
FileLogger::~FileLogger ( ) [inline]
```

Definition at line 42 of file FileLogger.hpp.

Here is the call graph for this function:



4.12.2.2 FileLogger() [1/2]

```
FileLogger::FileLogger ( ) [inline], [private]
```

Definition at line 62 of file FileLogger.hpp.

4.12.2.3 FileLogger() [2/2]

```
FileLogger::FileLogger ( 
    const FileLogger & ) [private], [default]
```

4.12.3 Member Function Documentation

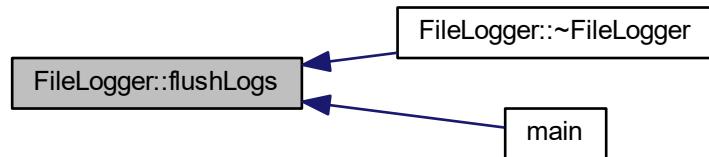
4.12.3.1 flushLogs()

```
void FileLogger::flushLogs ( ) [inline]
```

Takes the content of m_messageQueue and writes them to the log file.

Definition at line 17 of file FileLogger.hpp.

Here is the caller graph for this function:



4.12.3.2 getInstance()

```
static FileLogger& FileLogger::getInstance ( ) [inline], [static]
```

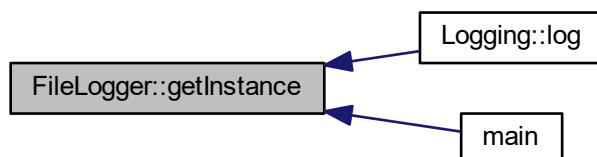
Singleton [getInstance\(\)](#)

Returns

an instance of this [FileLogger](#)

Definition at line 36 of file FileLogger.hpp.

Here is the caller graph for this function:



4.12.3.3 log()

```
void FileLogger::log (
    std::string message ) [inline]
```

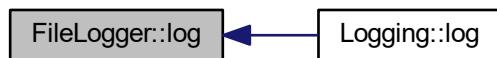
pushes the message to the message queue

Parameters

<i>message</i>	the string that is to be pushed to the queue
----------------	--

Definition at line 58 of file FileLogger.hpp.

Here is the caller graph for this function:



4.12.3.4 operator=()

```
FileLogger& FileLogger::operator= (
    const FileLogger & ) [private]
```

4.12.4 Member Data Documentation

4.12.4.1 mLogFile

```
FILE* FileLogger::mLogFile [private]
```

Definition at line 75 of file FileLogger.hpp.

4.12.4.2 m_messageQueue

```
std::queue<std::string> FileLogger::m_messageQueue [private]
```

Definition at line 76 of file FileLogger.hpp.

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

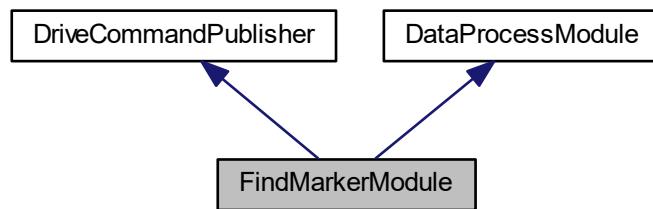
- PSE_2018_Gruppe1/src/[FileLogger.hpp](#)

4.13 FindMarkerModule Class Reference

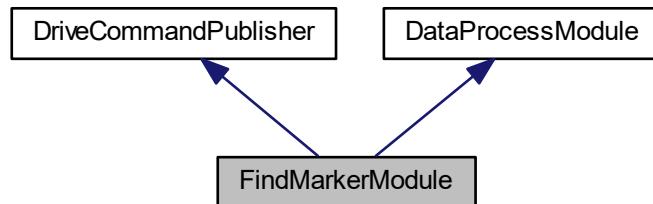
This module can be active in manual and in auto mode.

```
#include <FindMarkerModule.hpp>
```

Inheritance diagram for FindMarkerModule:



Collaboration diagram for FindMarkerModule:



Public Member Functions

- **FindMarkerModule ()**
Constructs a new `FindMarkerModule`.
- **~FindMarkerModule () override**
- **void `abort` ()**
aborts execution of the currently running blocking marker search
- **void `processData` () override**
Main update function that gets called from mainloop.
- **void `searchBlocking` ()**
Triggers a blocking marker search.
- **void `setMaxDriveDuration` (std::chrono::milliseconds duration)**
sets the maximum duration.

Static Private Member Functions

- static bool [foundMarker \(\)](#)

Retrieves the sensorList from [SensorManager](#) and checks whether it is not empty.

Private Attributes

- std::chrono::milliseconds [m_maxDuration](#)
- std::atomic< bool > [m_stop](#)

maximum time this module can work in blocking mode until it aborts

Static Private Attributes

- static const int [SEARCH_SPEED](#) = 10
used in blocking mode to abort operation
- static const int [MINIMUM_CONFIDENT](#) = 50

Additional Inherited Members

4.13.1 Detailed Description

This module can be active in manual and in auto mode.

It rotates the TivSeg until it detects at least one marker.

Definition at line 20 of file [FindMarkerModule.hpp](#).

4.13.2 Constructor & Destructor Documentation

4.13.2.1 [FindMarkerModule\(\)](#)

```
FindMarkerModule::FindMarkerModule ( )
```

Constructs a new [FindMarkerModule](#).

Definition at line 12 of file [FindMarkerModule.cpp](#).

4.13.2.2 [~FindMarkerModule\(\)](#)

```
FindMarkerModule::~FindMarkerModule ( ) [override], [default]
```

4.13.3 Member Function Documentation

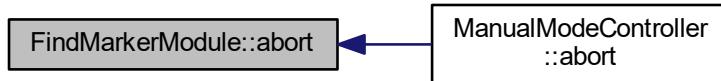
4.13.3.1 abort()

```
void FindMarkerModule::abort ( )
```

aborts execution of the currently running blocking marker search

Definition at line 16 of file FindMarkerModule.cpp.

Here is the caller graph for this function:



4.13.3.2 foundMarker()

```
bool FindMarkerModule::foundMarker ( ) [static], [private]
```

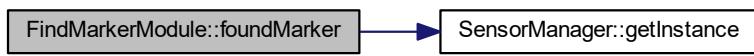
Retrieves the sensorList from [SensorManager](#) and checks whether it is not empty.

Returns

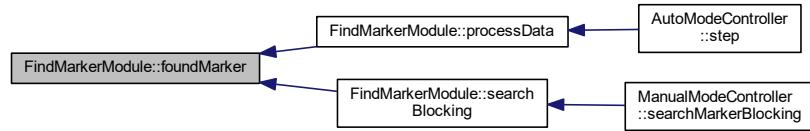
Marker list not empty? -> At least one marker was found -> true. Marker list empty? -> No marker -> false

Definition at line 58 of file FindMarkerModule.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.13.3.3 `processData()`

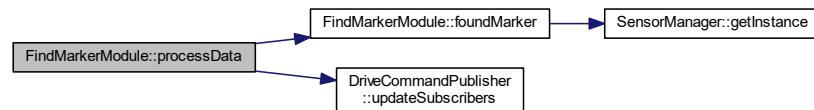
```
void FindMarkerModule::processData ( ) [override], [virtual]
```

Main update function that gets called from mainloop.

Implements [DataProcessModule](#).

Definition at line 18 of file `FindMarkerModule.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



4.13.3.4 searchBlocking()

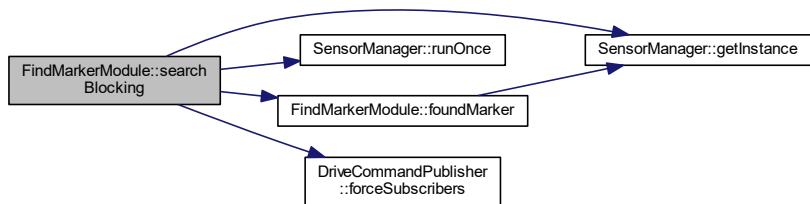
```
void FindMarkerModule::searchBlocking ( )
```

Triggers a blocking marker search.

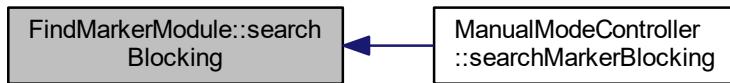
Returns if a Marker has been found or maximum drive duration has exceeded.

Definition at line 26 of file FindMarkerModule.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.13.3.5 setMaxDriveDuration()

```
void FindMarkerModule::setMaxDriveDuration (
    std::chrono::milliseconds duration )
```

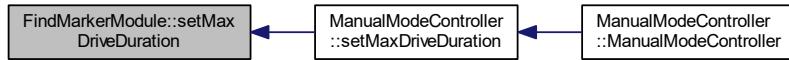
sets the maximum duration.

Parameters

<i>duration</i>	the allowed duration in milliseconds
-----------------	--------------------------------------

Definition at line 56 of file FindMarkerModule.cpp.

Here is the caller graph for this function:



4.13.4 Member Data Documentation

4.13.4.1 m_maxDuration

```
std::chrono::milliseconds FindMarkerModule::m_maxDuration [private]
```

Definition at line 59 of file FindMarkerModule.hpp.

4.13.4.2 m_stop

```
std::atomic<bool> FindMarkerModule::m_stop [private]
```

maximum time this module can work in blocking mode until it aborts

Definition at line 60 of file FindMarkerModule.hpp.

4.13.4.3 MINIMUM_CONFIDENT

```
const int FindMarkerModule::MINIMUM_CONFIDENT = 50 [static], [private]
```

Definition at line 63 of file FindMarkerModule.hpp.

4.13.4.4 SEARCH_SPEED

```
const int FindMarkerModule::SEARCH_SPEED = 10 [static], [private]
```

used in blocking mode to abort operation

Definition at line 62 of file FindMarkerModule.hpp.

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

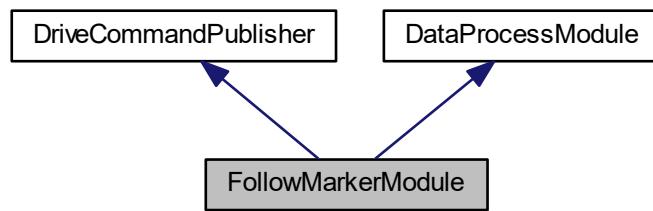
- PSE_2018_Gruppe1/src/[FindMarkerModule.hpp](#)
- PSE_2018_Gruppe1/src/[FindMarkerModule.cpp](#)

4.14 FollowMarkerModule Class Reference

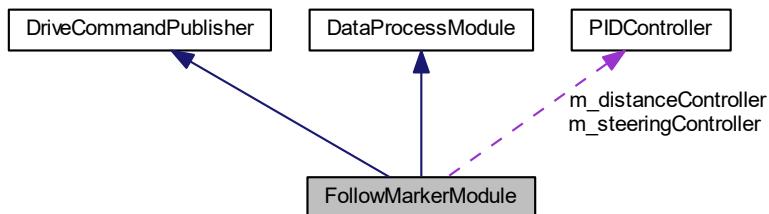
This module retrieves the markerList from the [SensorManager](#), checks for markers and tries to follow the one with the best confidence until it loses track of it.

```
#include <FollowMarkerModule.hpp>
```

Inheritance diagram for FollowMarkerModule:



Collaboration diagram for FollowMarkerModule:



Public Member Functions

- [`FollowMarkerModule \(\)`](#)
Constructs a new `FollowMarkerModule`.
- [`~FollowMarkerModule \(\) override`](#)
- [`void processData \(\) override`](#)
Main update function that gets called periodically.
- [`void setDistance \(double distance\)`](#)
Sets the desired distance that will be used as for the distance `PIDController`.
- [`void setMarkerId \(int markerId\)`](#)
NOT USED RIGHT NOW! Sets an ID for a marker that will be followed.

Private Attributes

- double `m_setDistance`
- bool `m_followsMarker`
- int `m_currentMarkerId`
- `PIDController m_distanceController`
- `PIDController m_steeringController`

Additional Inherited Members

4.14.1 Detailed Description

This module retrieves the markerList from the [SensorManager](#), checks for markers and tries to follow the one with the best confidence until it loses track of it.

After that, it follows the next marker with the highest confidence.

Definition at line 19 of file `FollowMarkerModule.hpp`.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 `FollowMarkerModule()`

```
FollowMarkerModule::FollowMarkerModule ( )
```

Constructs a new [FollowMarkerModule](#).

Definition at line 15 of file `FollowMarkerModule.cpp`.

4.14.2.2 `~FollowMarkerModule()`

```
FollowMarkerModule::~FollowMarkerModule ( ) [override], [default]
```

4.14.3 Member Function Documentation

4.14.3.1 processData()

```
void FollowMarkerModule::processData ( ) [override], [virtual]
```

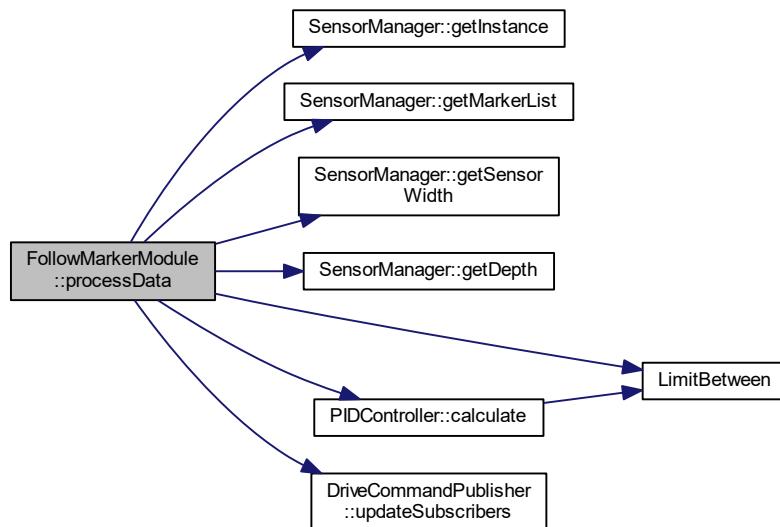
Main update function that gets called periodically.

gets a list of markers and steers towards the marker with the highest confidence value

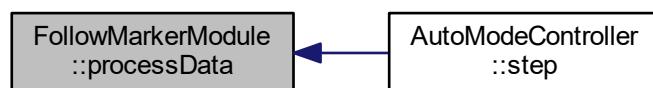
Implements [DataProcessModule](#).

Definition at line 30 of file [FollowMarkerModule.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.3.2 setDistance()

```
void FollowMarkerModule::setDistance (
    double distance )
```

Sets the desired distance that will be used as for the distance [PIDController](#).

Parameters

<i>distance</i>	desired distance in m
-----------------	-----------------------

Definition at line 93 of file FollowMarkerModule.cpp.

Here is the caller graph for this function:



4.14.3.3 setMarkerId()

```
void FollowMarkerModule::setMarkerId (
    int markerId )
```

NOT USED RIGHT NOW! Sets an ID for a marker that will be followed.

Definition at line 95 of file FollowMarkerModule.cpp.

4.14.4 Member Data Documentation

4.14.4.1 m_currentMarkerId

```
int FollowMarkerModule::m_currentMarkerId [private]
```

Definition at line 48 of file FollowMarkerModule.hpp.

4.14.4.2 m_distanceController

```
PIDController FollowMarkerModule::m_distanceController [private]
```

Definition at line 50 of file FollowMarkerModule.hpp.

4.14.4.3 m_followsMarker

```
bool FollowMarkerModule::m_followsMarker [private]
```

Definition at line 47 of file FollowMarkerModule.hpp.

4.14.4.4 m_setDistance

```
double FollowMarkerModule::m_setDistance [private]
```

Definition at line 46 of file FollowMarkerModule.hpp.

4.14.4.5 m_steeringController

```
PIDController FollowMarkerModule::m_steeringController [private]
```

Definition at line 51 of file FollowMarkerModule.hpp.

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

- PSE_2018_Gruppe1/src/FollowMarkerModule.hpp
- PSE_2018_Gruppe1/src/FollowMarkerModule.cpp

4.15 Logging Class Reference

This is a wrapper class for Diagnostics.

```
#include <Logging.hpp>
```

Static Public Member Functions

- static void [setMaxLevel](#) (DiagnosticLevel level)
Wrapper for Diagnostics::setMaxLevel() that also sets the max level for the current [FileLogger](#) instance.
- static void [log](#) (DiagnosticLevel level, const char *file, int line, const char *message,...)
Wrapper for Diagnostics::log() that also logs the message to the current [FileLogger](#) instance.

4.15.1 Detailed Description

This is a wrapper class for Diagnostics.

Apart from forwarding everything to Diagnostics, it also logs every received message to an instance of [FileLogger](#)

Definition at line 31 of file Logging.hpp.

4.15.2 Member Function Documentation

4.15.2.1 log()

```
void Logging::log (
    DiagnosticLevel level,
    const char * file,
    int line,
    const char * message,
    ... ) [static]
```

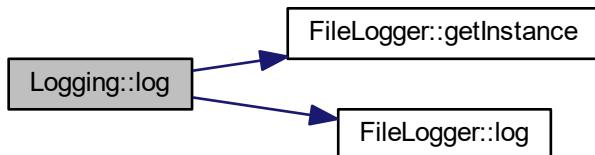
Wrapper for `Diagnostics::log()` that also logs the message to the current [FileLogger](#) instance.

Parameters

<i>level</i>	Log level of the message
<i>file</i>	File where the message was generated - filled by macro
<i>line</i>	Line where the message was generated - filled by macro
<i>message</i>	The actual message - can use printf style placeholders
...	printf style variadic replacements for placeholders in the message

Definition at line 10 of file `Logging.cpp`.

Here is the call graph for this function:



4.15.2.2 setMaxLevel()

```
void Logging::setMaxLevel (
    DiagnosticLevel level ) [static]
```

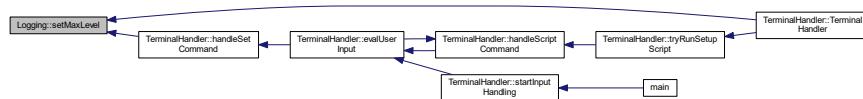
Wrapper for `Diagnostics::setMaxLevel()` that also sets the max level for the current [FileLogger](#) instance.

Parameters

<i>level</i>	max level of log messages
--------------	---------------------------

Definition at line 49 of file Logging.cpp.

Here is the caller graph for this function:



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

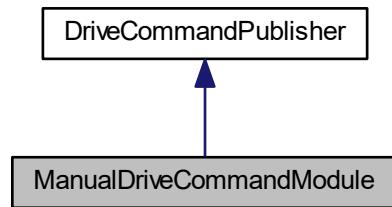
- PSE_2018_Gruppe1/src/[Logging.hpp](#)
- PSE_2018_Gruppe1/src/[Logging.cpp](#)

4.16 ManualDriveCommandModule Class Reference

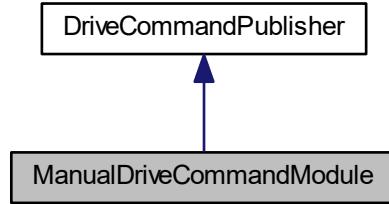
This module is used for blocking drive operations via terminal command or script and can only be invoked via terminal.

```
#include <ManualDriveCommandModule.hpp>
```

Inheritance diagram for ManualDriveCommandModule:



Collaboration diagram for ManualDriveCommandModule:



Public Member Functions

- `ManualDriveCommandModule ()`
Constructs a new `ManualDriveCommandModule`.
- `~ManualDriveCommandModule ()` override
- `void abort ()`
Aborts all blocking operations that might still be running.
- `void driveBlocking (std::chrono::milliseconds time, int velocity, int steering)`
Sends `DriveCommand` structs for the given time and blocks the calling thread.
- `void setMaxDriveDuration (std::chrono::milliseconds duration)`
Sets the maximum duration this module is allowed to drive without user interaction.

Private Attributes

- `std::chrono::milliseconds m_maxDuration`
- `std::atomic< bool > m_Stop`

Additional Inherited Members

4.16.1 Detailed Description

This module is used for blocking drive operations via terminal command or script and can only be invoked via terminal.

Definition at line 21 of file `ManualDriveCommandModule.hpp`.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 ManualDriveCommandModule()

```
ManualDriveCommandModule::ManualDriveCommandModule ( )
```

Constructs a new [ManualDriveCommandModule](#).

Definition at line 13 of file [ManualDriveCommandModule.cpp](#).

4.16.2.2 ~ManualDriveCommandModule()

```
ManualDriveCommandModule::~ManualDriveCommandModule ( ) [override], [default]
```

4.16.3 Member Function Documentation

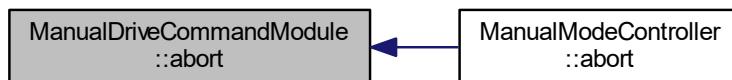
4.16.3.1 abort()

```
void ManualDriveCommandModule::abort ( )
```

Aborts all blocking operations that might still be running.

Definition at line 22 of file [ManualDriveCommandModule.cpp](#).

Here is the caller graph for this function:



4.16.3.2 driveBlocking()

```
void ManualDriveCommandModule::driveBlocking (
    std::chrono::milliseconds time,
    int velocity,
    int steering )
```

Sends [DriveCommand](#) structs for the given time and blocks the calling thread.

Parameters

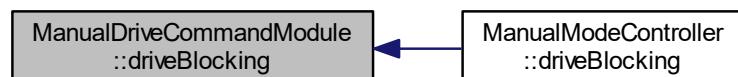
<i>time</i>	time to drive in milliseconds
<i>velocity</i>	desired speed
<i>steering</i>	desired direction (as ratio between left/right motor speed)

Definition at line 24 of file ManualDriveCommandModule.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**4.16.3.3 setMaxDriveDuration()**

```
void ManualDriveCommandModule::setMaxDriveDuration (
    std::chrono::milliseconds duration )
```

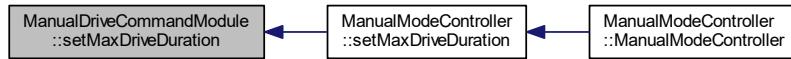
Sets the maximum duration this module is allowed to drive without user interaction.

Parameters

<i>duration</i>	maximum allowed time for command execution in milliseconds
-----------------	--

Definition at line 47 of file ManualDriveCommandModule.cpp.

Here is the caller graph for this function:



4.16.4 Member Data Documentation

4.16.4.1 m_maxDuration

```
std::chrono::milliseconds ManualDriveCommandModule::m_maxDuration [private]
```

Definition at line 51 of file `ManualDriveCommandModule.hpp`.

4.16.4.2 m_Stop

```
std::atomic<bool> ManualDriveCommandModule::m_Stop [private]
```

Definition at line 52 of file `ManualDriveCommandModule.hpp`.

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

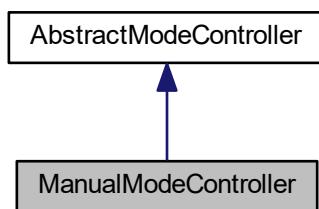
- PSE_2018_Gruppe1/src/[ManualDriveCommandModule.hpp](#)
- PSE_2018_Gruppe1/src/[ManualDriveCommandModule.cpp](#)

4.17 ManualModeController Class Reference

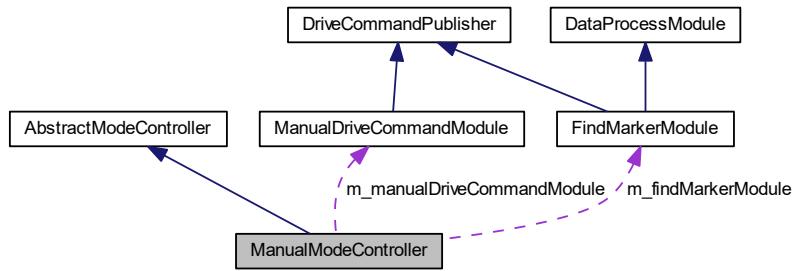
Child of [AbstractModeController](#) that is used for manual operation.

```
#include <ManualModeController.hpp>
```

Inheritance diagram for `ManualModeController`:



Collaboration diagram for ManualModeController:



Public Member Functions

- `~ManualModeController () override`
- `ManualModeController (std::shared_ptr< DriveCommandHandler > driveCommandHandler)`
Constructs a new `ManualModeController`.
- `void driveBlocking (std::chrono::milliseconds duration, double velocity, double steering)`
Drives for the given time with the given parameters.
- `void searchMarkerBlocking ()`
Trigger a searches for a Marker.
- `void setMaxDriveDuration (std::chrono::milliseconds maxDriveDuration)`
Sets the maximum duration this module is allowed to execute operations without user interaction.
- `void abort () override`
Aborts all currently running operations.
- `void step () override`
has to be called in a loop for continuous execution of the controller.

Private Attributes

- `ManualDriveCommandModule m_manualDriveCommandModule`
- `FindMarkerModule m_findMarkerModule`
- `std::atomic< bool > m_isInBlockingCall`

Static Private Attributes

- `static constexpr int DEFAULT_MAX_DRIVE_DURATION = 10`

Additional Inherited Members

4.17.1 Detailed Description

Child of `AbstractModeController` that is used for manual operation.

Definition at line 18 of file `ManualModeController.hpp`.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 ~ManualModeController()

```
ManualModeController::~ManualModeController ( ) [override], [default]
```

4.17.2.2 ManualModeController()

```
ManualModeController::ManualModeController (
    std::shared_ptr< DriveCommandHandler > driveCommandHandler ) [explicit]
```

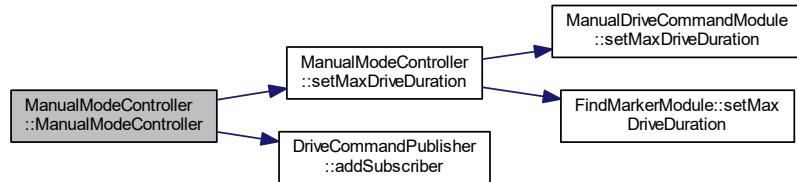
Constructs a new [ManualModeController](#).

Parameters

<code>driveCommandHandler</code>	used to wire up publishers in manual mode (publish-subscribe)
----------------------------------	---

Definition at line 12 of file `ManualModeController.cpp`.

Here is the call graph for this function:



4.17.3 Member Function Documentation

4.17.3.1 abort()

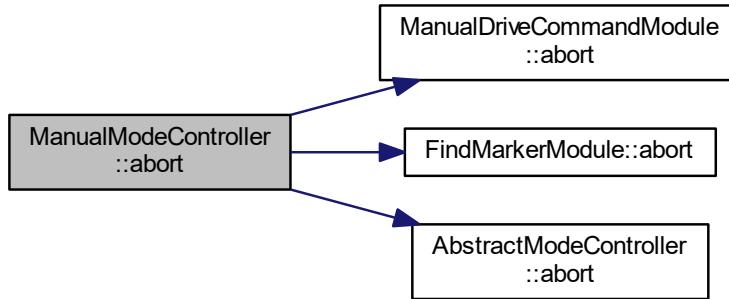
```
void ManualModeController::abort ( ) [override], [virtual]
```

Aborts all currently running operations.

Reimplemented from [AbstractModeController](#).

Definition at line 44 of file ManualModeController.cpp.

Here is the call graph for this function:



4.17.3.2 driveBlocking()

```
void ManualModeController::driveBlocking (
    std::chrono::milliseconds duration,
    double velocity,
    double steering )
```

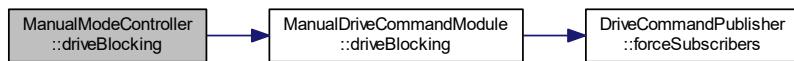
Drives for the given time with the given parameters.

Parameters

<i>duration</i>	duration in milliseconds for the drive operation
<i>velocity</i>	desired velocity for the drive operation (range: -100...100)
<i>steering</i>	desired direction for the drive operation as a ratio between left and right motor speeds (range: -100...100)

Definition at line 24 of file ManualModeController.cpp.

Here is the call graph for this function:



4.17.3.3 searchMarkerBlocking()

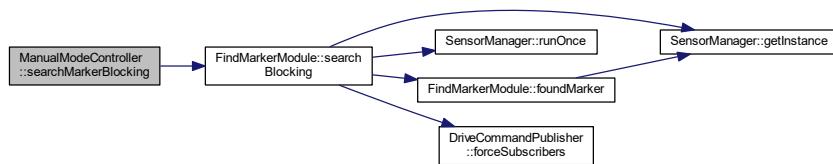
```
void ManualModeController::searchMarkerBlocking( )
```

Trigger a searches for a Marker.

Returns, when marker has found or maximim time has exceeded

Definition at line 31 of file ManualModeController.cpp.

Here is the call graph for this function:



4.17.3.4 setMaxDriveDuration()

```
void ManualModeController::setMaxDriveDuration(
    std::chrono::milliseconds maxDriveDuration )
```

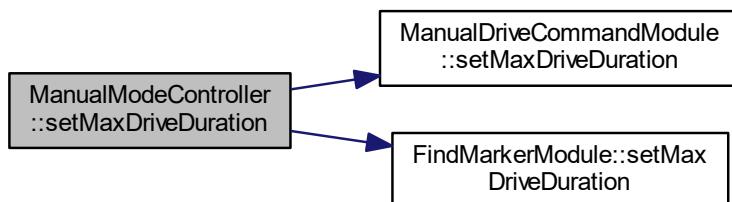
Sets the maximum duration this module is allowed to execute operations without user interaction.

Parameters

<i>maxDriveDuration</i>	duration in milliseconds
-------------------------	--------------------------

Definition at line 38 of file ManualModeController.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.3.5 step()

```
void ManualModeController::step ( ) [override], [virtual]
```

has to be called in a loop for continuous execution of the controller.

Triggers all DriveCommandModules to process the Sensor-Data.

Implements [AbstractModeController](#).

Definition at line 54 of file [ManualModeController.cpp](#).

4.17.4 Member Data Documentation

4.17.4.1 DEFAULT_MAX_DRIVE_DURATION

```
constexpr int ManualModeController::DEFAULT_MAX_DRIVE_DURATION = 10 [static], [private]
```

Definition at line 64 of file [ManualModeController.hpp](#).

4.17.4.2 m_findMarkerModule

```
FindMarkerModule ManualModeController::m_findMarkerModule [private]
```

Definition at line 61 of file [ManualModeController.hpp](#).

4.17.4.3 m_isInBlockingCall

```
std::atomic<bool> ManualModeController::m_isInBlockingCall [private]
```

Definition at line 62 of file ManualModeController.hpp.

4.17.4.4 m_manualDriveCommandModule

```
ManualDriveCommandModule ManualModeController::m_manualDriveCommandModule [private]
```

Definition at line 60 of file ManualModeController.hpp.

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

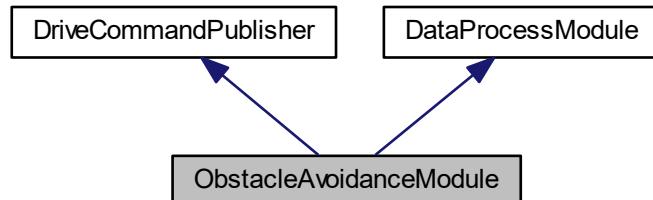
- PSE_2018_Gruppe1/src/[ManualModeController.hpp](#)
- PSE_2018_Gruppe1/src/[ManualModeController.cpp](#)

4.18 ObstacleAvoidanceModule Class Reference

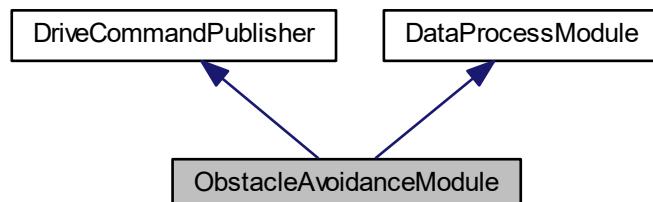
Used in automatic mode to detect and avoid obstacles during automatic driving operations.

```
#include <ObstacleAvoidanceModule.hpp>
```

Inheritance diagram for ObstacleAvoidanceModule:



Collaboration diagram for ObstacleAvoidanceModule:



Public Member Functions

- [ObstacleAvoidanceModule \(\)](#)
Constructs a new ObstacleAvoidanceModule.
- [~ObstacleAvoidanceModule \(\) override](#)
- [void processData \(\) override](#)
Triggers input data processing.

Static Private Attributes

- [static constexpr double MIN_DISTANCE = 0.6](#)

Additional Inherited Members

4.18.1 Detailed Description

Used in automatic mode to detect and avoid obstacles during automatic driving operations.

Publishes DriveCommands if an obstacle is detected and actively tries to turn away from the obstacle.

Definition at line 17 of file ObstacleAvoidanceModule.hpp.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 ObstacleAvoidanceModule()

```
ObstacleAvoidanceModule::ObstacleAvoidanceModule ( )
```

Constructs a new [ObstacleAvoidanceModule](#).

Definition at line 12 of file ObstacleAvoidanceModule.cpp.

4.18.2.2 ~ObstacleAvoidanceModule()

```
ObstacleAvoidanceModule::~ObstacleAvoidanceModule ( ) [override], [default]
```

4.18.3 Member Function Documentation

4.18.3.1 processData()

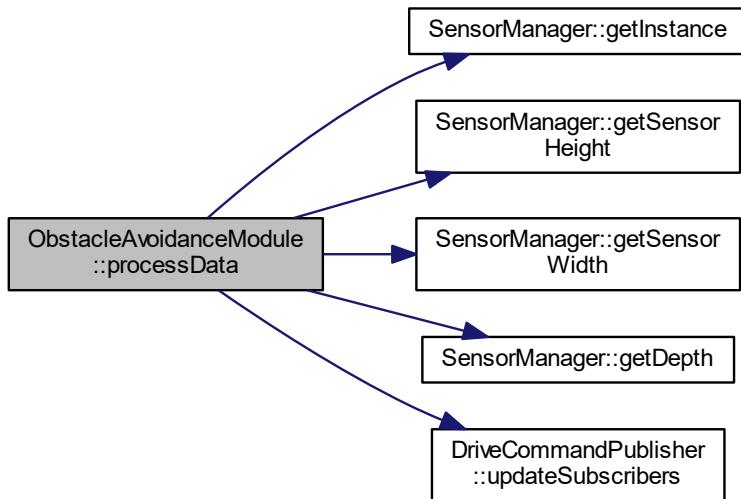
```
void ObstacleAvoidanceModule::processData( ) [override], [virtual]
```

Triggers input data processing.

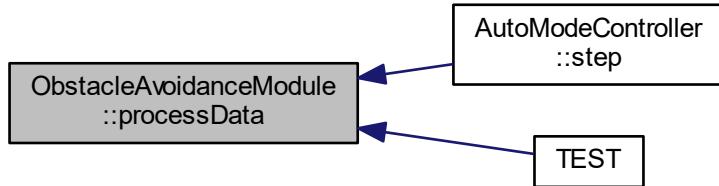
Implements [DataProcessModule](#).

Definition at line 16 of file ObstacleAvoidanceModule.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.4 Member Data Documentation

4.18.4.1 MIN_DISTANCE

```
constexpr double ObstacleAvoidanceModule::MIN_DISTANCE = 0.6 [static], [private]
```

Definition at line 33 of file ObstacleAvoidanceModule.hpp.

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

- PSE_2018_Gruppe1/src/[ObstacleAvoidanceModule.hpp](#)
- PSE_2018_Gruppe1/src/[ObstacleAvoidanceModule.cpp](#)

4.19 PIDController Class Reference

Basic universal PID controller implementation.

```
#include <PIDController.hpp>
```

Public Member Functions

- [PIDController](#) (double P, double I, double D, double dt=0.001, double min=-100.0, double max=100.0)
Constructs a new PID controller object.
- double [calculate](#) (double error)
main calc function of the PIDController
- virtual [~PIDController](#) ()

Private Attributes

- double [m_d](#)
- double [m_error](#)
- double [m_errorDot](#)
- double [m_i](#)
- double [m_integralError](#)
- double [m_p](#)
- double [m_dt](#)
- double [m_minVal](#)
- double [m_maxVal](#)

4.19.1 Detailed Description

Basic universal PID controller implementation.

Definition at line 13 of file PIDController.hpp.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 PIDController()

```
PIDController::PIDController (
    double P,
    double I,
    double D,
    double dt = 0.001,
    double min = -100.0,
    double max = 100.0 )
```

Constructs a new PID controller object.

Parameters

<i>P</i>	Proportional factor
<i>I</i>	Integral factor
<i>D</i>	Differential factor
<i>dt</i>	Time delta between calls to calculate()
<i>min</i>	Minimal output of controller
<i>max</i>	Maximal output of controller

Returns

A new [PIDController](#) object

Definition at line 13 of file PIDController.cpp.

4.19.2.2 ~PIDController()

```
PIDController::~PIDController ( ) [virtual], [default]
```

4.19.3 Member Function Documentation**4.19.3.1 calculate()**

```
double PIDController::calculate (
    double error )
```

main calc function of the [PIDController](#)

Parameters

<i>error</i>	difference between target and actual values
--------------	---

Returns

controller output

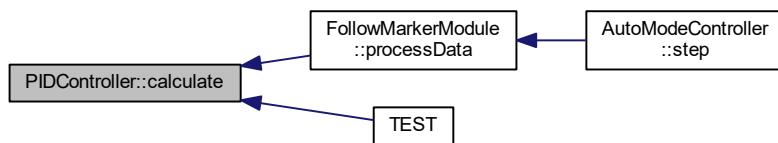
< Integrate errors as long as the result does not overflow

Definition at line 28 of file PIDController.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.19.4 Member Data Documentation

4.19.4.1 m_d

```
double PIDController::m_d [private]
```

Definition at line 38 of file PIDController.hpp.

4.19.4.2 m_dt

```
double PIDController::m_dt [private]
```

Definition at line 44 of file PIDController.hpp.

4.19.4.3 m_error

```
double PIDController::m_error [private]
```

Definition at line 39 of file PIDController.hpp.

4.19.4.4 m_errorDot

```
double PIDController::m_errorDot [private]
```

Definition at line 40 of file PIDController.hpp.

4.19.4.5 m_i

```
double PIDController::m_i [private]
```

Definition at line 41 of file PIDController.hpp.

4.19.4.6 m_integralError

```
double PIDController::m_integralError [private]
```

Definition at line 42 of file PIDController.hpp.

4.19.4.7 m_maxVal

```
double PIDController::m_maxVal [private]
```

Definition at line 46 of file PIDController.hpp.

4.19.4.8 m_minVal

```
double PIDController::m_minVal [private]
```

Definition at line 45 of file PIDController.hpp.

4.19.4.9 m_p

```
double PIDController::m_p [private]
```

Definition at line 43 of file PIDController.hpp.

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

- PSE_2018_Gruppe1/src/[PIDController.hpp](#)
- PSE_2018_Gruppe1/src/[PIDController.cpp](#)

4.20 SensorManager Class Reference

Sensor and image processing manager.

```
#include <SensorManager.hpp>
```

Public Member Functions

- `SensorManager (const SensorManager &) = delete`
- `SensorManager & operator= (const SensorManager &) = delete`
- `bool runOnce ()`
Update color and depth data, and perform marker detection.
- `std::vector< MarkerInfo > getMarkerList ()`
Get the list of currently detected markers.
- `int getSensorWidth ()`
Get the width of the sensor data.
- `int getSensorHeight ()`
Get the height of the sensor data.
- `double getDepth (int x, int y)`
Get the depth [m] at the given pixel in the sensor data.
- `float getFps ()`
Get an average of the current processing FPS.
- `void setDebugMarkerInfoList (const std::vector< MarkerInfo > &debugMarkerInfoList)`
- `void setDepth (double depth)`

Static Public Member Functions

- `static SensorManager * getInstance ()`
Get the singleton instance of the sensor manager.

Protected Member Functions

- `SensorManager () = default`
- `~SensorManager () = default`

Protected Attributes

- `std::vector< MarkerInfo > debugMarkerInfoList`
- `double depth {}`

4.20.1 Detailed Description

Sensor and image processing manager.

Definition at line 10 of file SensorManager.hpp.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 SensorManager() [1/2]

```
SensorManager::SensorManager (
    const SensorManager & ) [delete]
```

4.20.2.2 SensorManager() [2/2]

```
SensorManager::SensorManager ( ) [protected], [default]
```

4.20.2.3 ~SensorManager()

```
SensorManager::~SensorManager ( ) [protected], [default]
```

4.20.3 Member Function Documentation

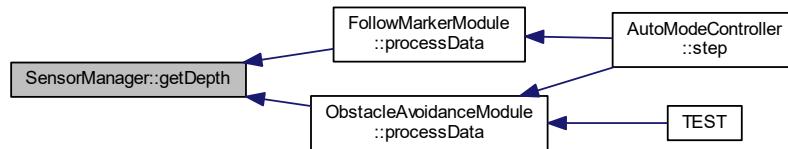
4.20.3.1 getDepth()

```
double SensorManager::getDepth (
    int x,
    int y ) [inline]
```

Get the depth [m] at the given pixel in the sensor data.

Definition at line 44 of file SensorManager.hpp.

Here is the caller graph for this function:



4.20.3.2 getFps()

```
float SensorManager::getFps ( ) [inline]
```

Get an average of the current processing FPS.

Definition at line 49 of file SensorManager.hpp.

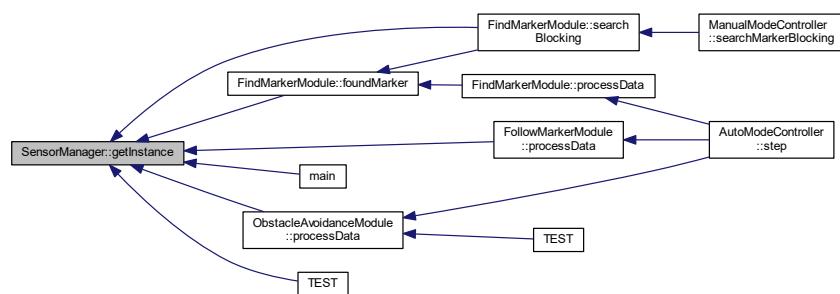
4.20.3.3 getInstance()

```
static SensorManager* SensorManager::getInstance ( ) [inline], [static]
```

Get the singleton instance of the sensor manager.

Definition at line 17 of file SensorManager.hpp.

Here is the caller graph for this function:



4.20.3.4 getMarkerList()

```
std::vector<MarkerInfo> SensorManager::getMarkerList ( ) [inline]
```

Get the list of currently detected markers.

Definition at line 29 of file SensorManager.hpp.

Here is the caller graph for this function:



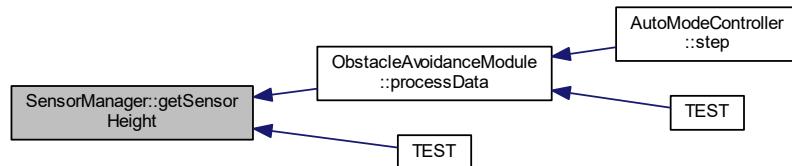
4.20.3.5 getSensorHeight()

```
int SensorManager::getSensorHeight ( ) [inline]
```

Get the height of the sensor data.

Definition at line 39 of file SensorManager.hpp.

Here is the caller graph for this function:



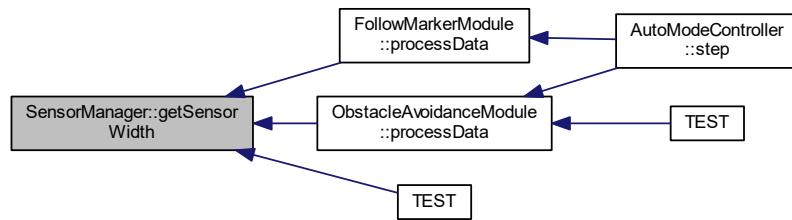
4.20.3.6 getSensorWidth()

```
int SensorManager::getSensorWidth ( ) [inline]
```

Get the width of the sensor data.

Definition at line 34 of file SensorManager.hpp.

Here is the caller graph for this function:



4.20.3.7 operator=()

```
SensorManager& SensorManager::operator= (
    const SensorManager & ) [delete]
```

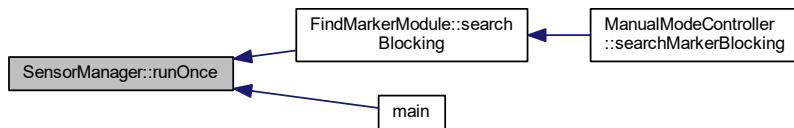
4.20.3.8 runOnce()

```
bool SensorManager::runOnce ( ) [inline]
```

Update color and depth data, and perform marker detection.

Definition at line 24 of file SensorManager.hpp.

Here is the caller graph for this function:



4.20.3.9 setDebugMarkerInfoList()

```
void SensorManager::setDebugMarkerInfoList ( const std::vector< MarkerInfo > & debugMarkerInfoList ) [inline]
```

Definition at line 53 of file SensorManager.hpp.

Here is the caller graph for this function:



4.20.3.10 setDepth()

```
void SensorManager::setDepth (
    double depth ) [inline]
```

Definition at line 57 of file SensorManager.hpp.

Here is the caller graph for this function:



4.20.4 Member Data Documentation

4.20.4.1 debugMarkerInfoList

```
std::vector<MarkerInfo> SensorManager::debugMarkerInfoList [protected]
```

Definition at line 65 of file SensorManager.hpp.

4.20.4.2 depth

```
double SensorManager::depth {} [protected]
```

Definition at line 66 of file SensorManager.hpp.

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

- PSE_2018_Gruppe1/src/test/mocks/SensorManager.hpp

4.21 TerminalHandler Class Reference

```
#include <TerminalHandler.hpp>
```

Public Member Functions

- `TerminalHandler (std::shared_ptr< ManualModeController > manualModeController, std::shared_ptr< AutoModeController > autoModeController)`

Constructs a new TerminalHandler.
- `virtual ~TerminalHandler ()`
- `void abort ()`

Aborts all running operations.
- `void startInputHandling ()`

Main entry point to the TerminalHandler.
- `bool isInAutoMode () const`

Checks whether the user chose to operate in automatic mode.
- `TerminalHandler (std::shared_ptr< ManualModeController > manualModeController, std::shared_ptr< AutoModeController > autoModeController)`

Constructs a new TerminalHandler.
- `virtual ~TerminalHandler ()`
- `void abort ()`

Aborts all running operations.
- `void startInputHandling ()`

Main entry point to the TerminalHandler.
- `bool isInAutoMode () const`

Checks whether the user chose to operate in automatic mode.
- `bool handleSetCommand (std::vector< std::string > &commandTokens)`

Handles a command if its first token is set.
- `bool handleDriveCommand (std::vector< std::string > &commandTokens)`

Handles a command if its first token is drive.
- `bool handleScriptCommand (std::vector< std::string > &commandTokens, int recursionDepth)`

Handles a command if its first token is script.
- `bool handleSearchCommand (std::vector< std::string > &commandTokens)`

Handles a command if its first token is search.
- `bool handleModeCommand (std::vector< std::string > &commandTokens)`

Handles a command if its first token is mode.
- `bool evalUserInput (const std::string &input, bool scriptMode=false, int recursionDepth=0)`

Gets called whenever the user entered a new line and tries to parse the entered command.
- `bool tryRunSetupScript ()`

checks if a script called setup.pse is placed besides the application executable.

Static Public Member Functions

- `static void splitString (const std::string &input, std::vector< std::string > &result, const char *delim)`

Splits a given input string at a 1-character delimiter.

Private Member Functions

- bool `handleSetCommand` (std::vector< std::string > &commandTokens)
Handles a command if its first token is set.
- bool `handleDriveCommand` (std::vector< std::string > &commandTokens)
Handles a command if its first token is drive.
- bool `handleScriptCommand` (std::vector< std::string > &commandTokens, int recursionDepth)
Handles a command if its first token is script.
- bool `handleSearchCommand` (std::vector< std::string > &commandTokens)
Handles a command if its first token is search.
- bool `handleModeCommand` (std::vector< std::string > &commandTokens)
Handles a command if its first token is mode.
- bool `evalUserInput` (const std::string &input, bool scriptMode=false, int recursionDepth=0)
Gets called whenever the user entered a new line and tries to parse the entered command.
- bool `tryRunSetupScript` ()
checks if a script called setup.pse is placed besides the application executable.

Static Private Member Functions

- static void `splitString` (const std::string &input, std::vector< std::string > &result, const char *delim)
Splits a given input string at a 1-character delimiter.

Private Attributes

- std::shared_ptr< ManualModeController > `m_manualModeController`
- std::shared_ptr< AutoModeController > `m_autoModeController`
- std::atomic< bool > `m_isInAutoMode` {false}
- std::atomic< bool > `m_handleInput` {true}
- std::atomic< bool > `m_abortScript` {true}

Static Private Attributes

- static constexpr int `MAX_RECURSION_DEPTH` = 5
- static constexpr DiagnosticLevel `DEFAULT_DIAGNOSTICS_MAX_LEVEL` = DiagnosticLevel::DIAG_ERROR

4.21.1 Detailed Description

Definition at line 14 of file TerminalHandler.hpp.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 TerminalHandler() [1/2]

```
TerminalHandler::TerminalHandler (
    std::shared_ptr< ManualModeController > manualModeController,
    std::shared_ptr< AutoModeController > autoModeController )
```

Constructs a new `TerminalHandler`.

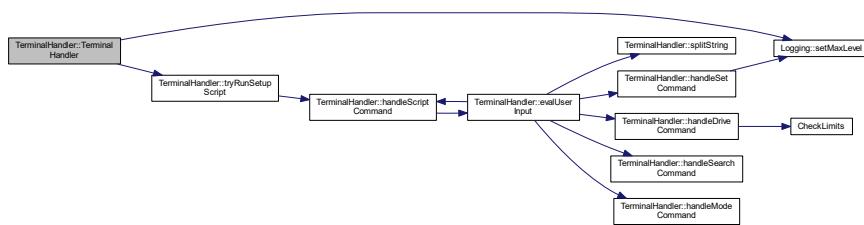
Constructor for `TerminalHandler`.

Parameters

<i>manualModeController</i>	Controller for manual operation.
<i>autoModeController</i>	Controller for automatic marker based operation.
<i>manualModeController</i>	Controller for manual mode commands
<i>autoModeController</i>	Controller for automatic mode commands

Definition at line 25 of file TerminalHandler.cpp.

Here is the call graph for this function:



4.21.2.2 ~TerminalHandler() [1/2]

TerminalHandler::~TerminalHandler () [virtual], [default]

4.21.2.3 TerminalHandler() [2/2]

```
TerminalHandler::TerminalHandler (
    std::shared_ptr< ManualModeController > manualModeController,
    std::shared_ptr< AutoModeController > autoModeController )
```

Constructs a new [TerminalHandler](#).

Parameters

<i>manualModeController</i>	Controller for manual operation.
<i>autoModeController</i>	Controller for automatic marker based operation.

4.21.2.4 ~TerminalHandler() [2/2]

virtual TerminalHandler::~TerminalHandler () [virtual]

4.21.3 Member Function Documentation

4.21.3.1 `abort()` [1/2]

```
void TerminalHandler::abort ( )
```

Aborts all running operations.

Abort the current command execution, whatever is being run right now.

Definition at line 46 of file TerminalHandler.cpp.

4.21.3.2 `abort()` [2/2]

```
void TerminalHandler::abort ( )
```

Aborts all running operations.

4.21.3.3 `evalUserInput()` [1/2]

```
bool TerminalHandler::evalUserInput (
    const std::string & input,
    bool scriptMode = false,
    int recursionDepth = 0 )
```

Gets called whenever the user entered a new line and tries to parse the entered command.

Parameters

<code>input</code>	Command string as entered by the user.
<code>scriptMode</code>	Scripts also use this function. In order to block them from using certain commands, handleScriptCommand() sets this to true.
<code>recursionDepth</code>	A script using the script command to call itself might cause infinite recursion. This parameter counts the depth of the current call so we can abort if it gets too deep.

Returns

true if input contained a valid command and the command execution succeeded

4.21.3.4 evalUserInput() [2/2]

```
bool TerminalHandler::evalUserInput (
    const std::string & input,
    bool scriptMode = false,
    int recursionDepth = 0 ) [private]
```

Gets called whenever the user entered a new line and tries to parse the entered command.

Tries to execute a given command string.

Parameters

<i>input</i>	Command string as entered by the user.
<i>scriptMode</i>	Scripts also use this function. In order to block them from using certain commands, <code>handleScriptCommand()</code> sets this to true.
<i>recursionDepth</i>	A script using the script command to call itself might cause infinite recursion. This parameter counts the depth of the current call so we can abort if it gets too deep.

Returns

true if input contained a valid command and the command execution succeeded

Parameters

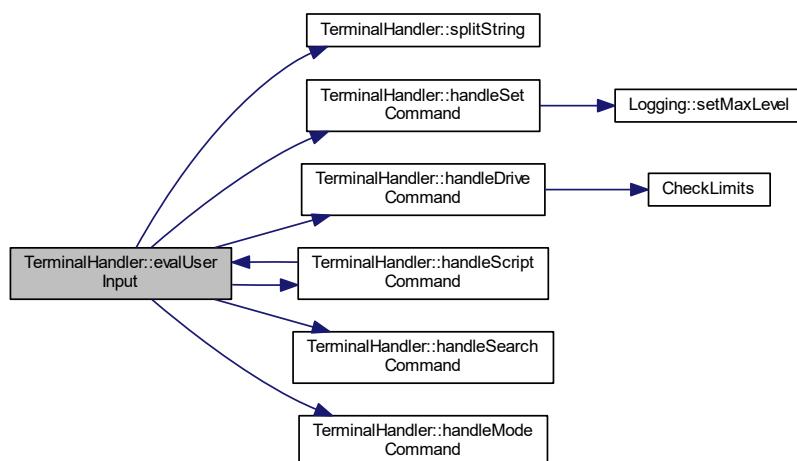
<i>input</i>	the command string
--------------	--------------------

Returns

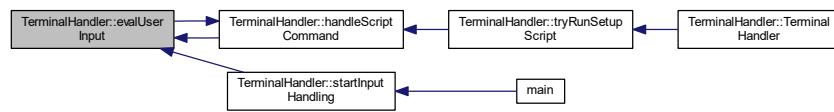
True on success, False on Fail

Definition at line 340 of file TerminalHandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.21.3.5 handleDriveCommand() [1/2]

```
bool TerminalHandler::handleDriveCommand (
    std::vector< std::string > & commandTokens )
```

Handles a command if its first token is `drive`.

Drives with the given speed in a given direction for a given time if all three parameters. are within the boundaries that are given in the specification.

Parameters

<i>commandTokens</i>	a token representation of the entered user command, split at every space character.
----------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

4.21.3.6 handleDriveCommand() [2/2]

```
bool TerminalHandler::handleDriveCommand (
    std::vector< std::string > & commandTokens ) [private]
```

Handles a command if its first token is `drive`.

Handles the "drive" command (see spec)

Drives with the given speed in a given direction for a given time if all three parameters. are within the boundaries that are given in the specification.

Parameters

<i>commandTokens</i>	a token representation of the entered user command, split at every space character.
----------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

Parameters

<i>commandTokens</i>	all tokens that were extracted from the input
----------------------	---

Returns

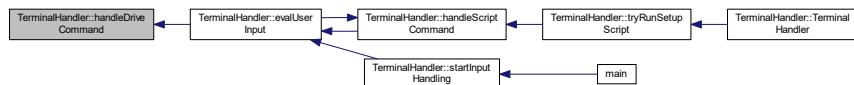
True on success, False on Fail

Definition at line 173 of file TerminalHandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**4.21.3.7 handleModeCommand() [1/2]**

```
bool TerminalHandler::handleModeCommand (
    std::vector< std::string > & commandTokens )
```

Handles a command if its first token is mode.

If the second token is a valid mode of operation according to the specification (1 or 2), the mode is switched accordingly.

Parameters

<i>commandTokens</i>	a token representation of the entered user command, split at every space character.
----------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

4.21.3.8 handleModeCommand() [2/2]

```
bool TerminalHandler::handleModeCommand (
    std::vector< std::string > & commandTokens ) [private]
```

Handles a command if its first token is mode.

Handles the "mode" command (see spec)

If the second token is a valid mode of operation according to the specification (1 or 2), the mode is switched accordingly.

Parameters

<i>commandTokens</i>	a token representation of the entered user command, split at every space character.
----------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

Parameters

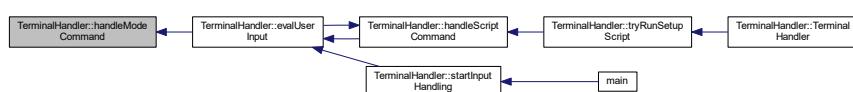
<i>commandTokens</i>	all tokens that were extracted from the input
----------------------	---

Returns

True on success, False on Fail

Definition at line 301 of file TerminalHandler.cpp.

Here is the caller graph for this function:



4.21.3.9 handleScriptCommand() [1/2]

```
bool TerminalHandler::handleScriptCommand (
    std::vector< std::string > & commandTokens,
    int recursionDepth )
```

Handles a command if its first token is `script`.

If a script is found at the given path, this command handler executes the script line by line.

Parameters

<code>commandTokens</code>	a token representation of the entered user command, split at every space character.
----------------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

4.21.3.10 handleScriptCommand() [2/2]

```
bool TerminalHandler::handleScriptCommand (
    std::vector< std::string > & commandTokens,
    int recursionDepth ) [private]
```

Handles a command if its first token is `script`.

Handles the "script" command (see spec)

If a script is found at the given path, this command handler executes the script line by line.

Parameters

<code>commandTokens</code>	a token representation of the entered user command, split at every space character.
----------------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

Parameters

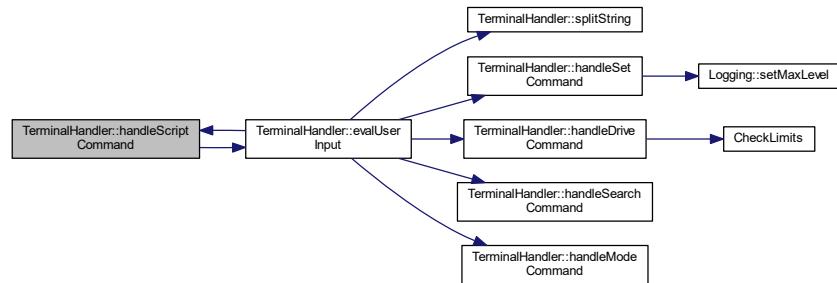
<code>commandTokens</code>	all tokens that were extracted from the input
----------------------------	---

Returns

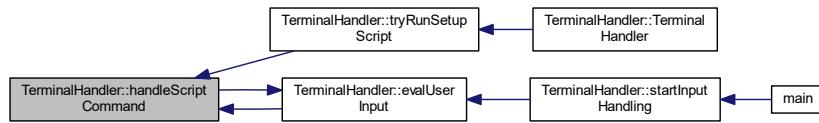
True on success, False on Fail

Definition at line 214 of file TerminalHandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.21.3.11 handleSearchCommand() [1/2]

```
bool TerminalHandler::handleSearchCommand (
    std::vector< std::string > & commandTokens )
```

Handles a command if its first token is search.

Starts a new blocking marker search.

Parameters

<i>commandTokens</i>	a token representation of the entered user command, split at every space character.
----------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

4.21.3.12 handleSearchCommand() [2/2]

```
bool TerminalHandler::handleSearchCommand (
    std::vector< std::string > & commandTokens ) [private]
```

Handles a command if its first token is `search`.

Handles the "search" command (see spec)

Starts a new blocking marker search.

Parameters

<code>commandTokens</code>	a token representation of the entered user command, split at every space character.
----------------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

Parameters

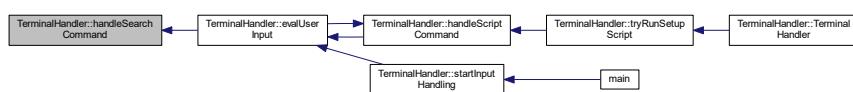
<code>commandTokens</code>	all tokens that were extracted from the input
----------------------------	---

Returns

True on success, False on Fail

Definition at line 288 of file TerminalHandler.cpp.

Here is the caller graph for this function:



4.21.3.13 handleSetCommand() [1/2]

```
bool TerminalHandler::handleSetCommand (
    std::vector< std::string > & commandTokens )
```

Handles a command if its first token is `set`.

Sets the specified values, as described in the specification.

Parameters

<code>commandTokens</code>	a token representation of the entered user command, split at every space character.
----------------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

4.21.3.14 handleSetCommand() [2/2]

```
bool TerminalHandler::handleSetCommand (
    std::vector< std::string > & commandTokens ) [private]
```

Handles a command if its first token is set.

Executes the "set" command (see spec)

Sets the specified values, as described in the specification.

Parameters

<i>commandTokens</i>	a token representation of the entered user command, split at every space character.
----------------------	---

Returns

true if the user input did not contain errors and the command succeeded. false otherwise.

Parameters

<i>commandTokens</i>	all tokens that were extracted from the input
----------------------	---

Returns

True on success, False on Fail

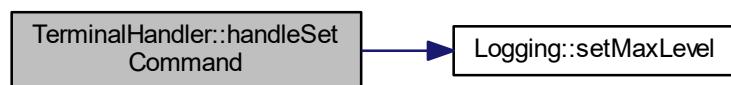
< set timeout switch

< set distance switch

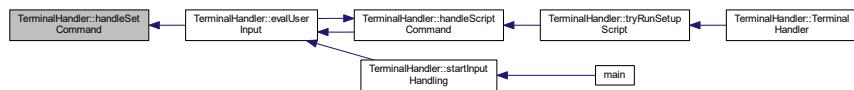
< set logging level switch

Definition at line 91 of file TerminalHandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.21.3.15 `isInAutoMode()` [1/2]

```
bool TerminalHandler::isInAutoMode() const
```

Checks whether the user chose to operate in automatic mode.

Returns

value of m_IsInAutoMode.

Definition at line 423 of file TerminalHandler.cpp.

4.21.3.16 `isInAutoMode()` [2/2]

```
bool TerminalHandler::isInAutoMode() const
```

Checks whether the user chose to operate in automatic mode.

Returns

value of m_IsInAutoMode.

4.21.3.17 `splitString()` [1/2]

```
static void TerminalHandler::splitString(
    const std::string & input,
    std::vector< std::string > & result,
    const char * delim ) [static]
```

Splits a given input string at a 1-character delimiter.

Parameters

<i>input</i>	String that has to be split
<i>result</i>	Reference to a existing vector that wil lbe used to store the resulting tokens.
<i>delim</i>	1-character delimiter that is used to split the input string.

4.21.3.18 splitString() [2/2]

```
void TerminalHandler::splitString (
    const std::string & input,
    std::vector< std::string > & result,
    const char * delim ) [static], [private]
```

Splits a given input string at a 1-character delimiter.

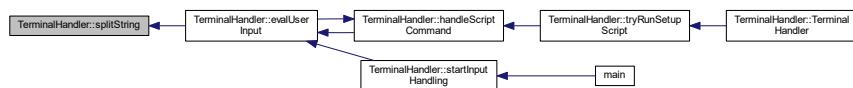
Splits a given string at a given delimiter and saves the resulting tokens in the vector result.

Parameters

<i>input</i>	String that has to be split
<i>result</i>	Reference to a existing vector that wil lbe used to store the resulting tokens.
<i>delim</i>	1-character delimiter that is used to split the input string.
<i>input</i>	The string
<i>result</i>	A ref to a vector that will be filled with the tokens
<i>delim</i>	The delimiter for the split operation

Definition at line 69 of file TerminalHandler.cpp.

Here is the caller graph for this function:



4.21.3.19 startInputHandling() [1/2]

```
void TerminalHandler::startInputHandling ( )
```

Main entry point to the [TerminalHandler](#).

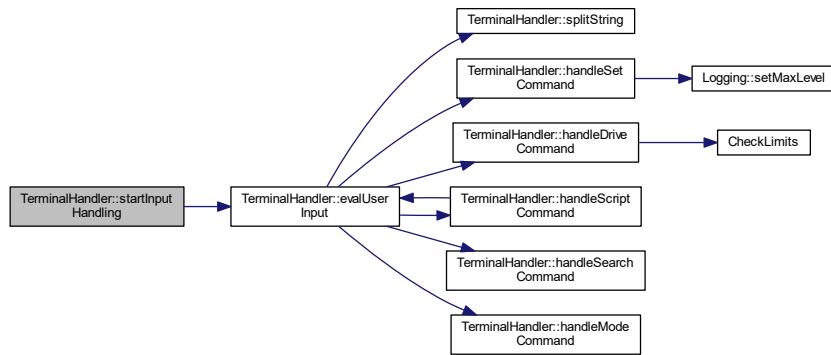
Main Terminal Handler function.

Displays a banner and opens a user prompt. ONLY CALL THIS FROM A SEPARATE THREAD!

Will run in its own thread.

Definition at line 403 of file TerminalHandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.21.3.20 `startInputHandling()` [2/2]

```
void TerminalHandler::startInputHandling ( )
```

Main entry point to the [TerminalHandler](#).

Displays a banner and opens a user prompt. ONLY CALL THIS FROM A SEPARATE THREAD!

4.21.3.21 `tryRunSetupScript()` [1/2]

```
bool TerminalHandler::tryRunSetupScript ( )
```

checks if a script called setup.pse is placed besides the application executable.

If such a script is found, it is executed.

Returns

true if the script was successfully executed

4.21.3.22 tryRunSetupScript() [2/2]

```
bool TerminalHandler::tryRunSetupScript ( ) [private]
```

checks if a script called setup.pse is placed besides the application executable.

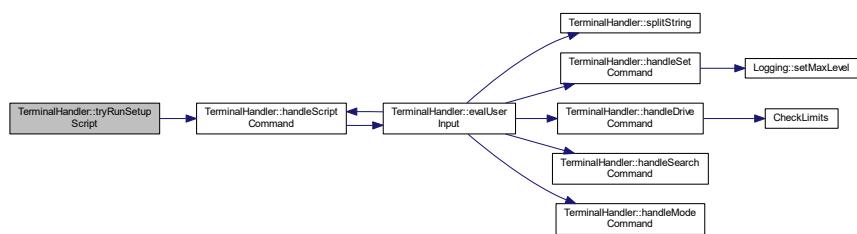
If such a script is found, it is executed.

Returns

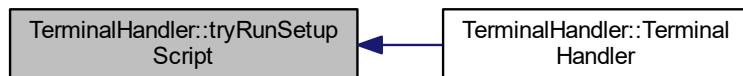
true if the script was successfully executed

Definition at line 34 of file TerminalHandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.21.4 Member Data Documentation

4.21.4.1 DEFAULT_DIAGNOSTICS_MAX_LEVEL

```
static constexpr DiagnosticLevel TerminalHandler::DEFAULT_DIAGNOSTICS_MAX_LEVEL = DiagnosticLevel::DIAG_ERROR
[static], [private]
```

Definition at line 123 of file TerminalHandler.hpp.

4.21.4.2 m_abortScript

```
std::atomic< bool > TerminalHandler::m_abortScript {true} [private]
```

Definition at line 50 of file TerminalHandler.hpp.

4.21.4.3 m_autoModeController

```
std::shared_ptr< AutoModeController > TerminalHandler::m_autoModeController [private]
```

Definition at line 47 of file TerminalHandler.hpp.

4.21.4.4 m_handleInput

```
std::atomic< bool > TerminalHandler::m_handleInput {true} [private]
```

Definition at line 49 of file TerminalHandler.hpp.

4.21.4.5 m_isInAutoMode

```
std::atomic< bool > TerminalHandler::m_isInAutoMode {false} [private]
```

Definition at line 48 of file TerminalHandler.hpp.

4.21.4.6 m_manualModeController

```
std::shared_ptr< ManualModeController > TerminalHandler::m_manualModeController [private]
```

Definition at line 46 of file TerminalHandler.hpp.

4.21.4.7 MAX_RECUSION_DEPTH

```
static constexpr int TerminalHandler::MAX_RECUSION_DEPTH = 5 [static], [private]
```

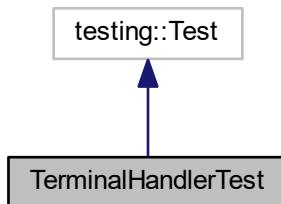
Definition at line 120 of file TerminalHandler.hpp.

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

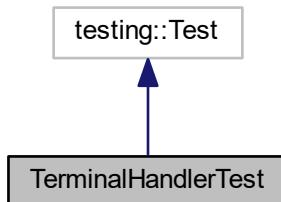
- PSE_2018_Gruppe1/src/TerminalHandler.hpp
- PSE_2018_Gruppe1/src/TerminalHandler.cpp

4.22 TerminalHandlerTest Class Reference

Inheritance diagram for TerminalHandlerTest:



Collaboration diagram for TerminalHandlerTest:



Protected Member Functions

- void [SetUp \(\)](#) override
- void [TearDown \(\)](#) override

Protected Attributes

- std::shared_ptr< [DriveCommandHandler](#) > driveCommandHandler
- std::shared_ptr< [AutoModeController](#) > autoModeController
- std::shared_ptr< [ManualModeController](#) > manualModeController
- std::shared_ptr< [TerminalHandler](#) > terminalHandler

4.22.1 Detailed Description

Definition at line 11 of file TerminalHandlerTest.cpp.

4.22.2 Member Function Documentation

4.22.2.1 SetUp()

```
void TerminalHandlerTest::SetUp ( ) [inline], [override], [protected]
```

Definition at line 19 of file TerminalHandlerTest.cpp.

4.22.2.2 TearDown()

```
void TerminalHandlerTest::TearDown ( ) [inline], [override], [protected]
```

Definition at line 27 of file TerminalHandlerTest.cpp.

4.22.3 Member Data Documentation

4.22.3.1 autoModeController

```
std::shared_ptr<AutoModeController> TerminalHandlerTest::autoModeController [protected]
```

Definition at line 15 of file TerminalHandlerTest.cpp.

4.22.3.2 driveCommandHandler

```
std::shared_ptr<DriveCommandHandler> TerminalHandlerTest::driveCommandHandler [protected]
```

Definition at line 14 of file TerminalHandlerTest.cpp.

4.22.3.3 manualModeController

```
std::shared_ptr<ManualModeController> TerminalHandlerTest::manualModeController [protected]
```

Definition at line 16 of file TerminalHandlerTest.cpp.

4.22.3.4 terminalHandler

```
std::shared_ptr<TerminalHandler> TerminalHandlerTest::terminalHandler [protected]
```

Definition at line 17 of file TerminalHandlerTest.cpp.

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

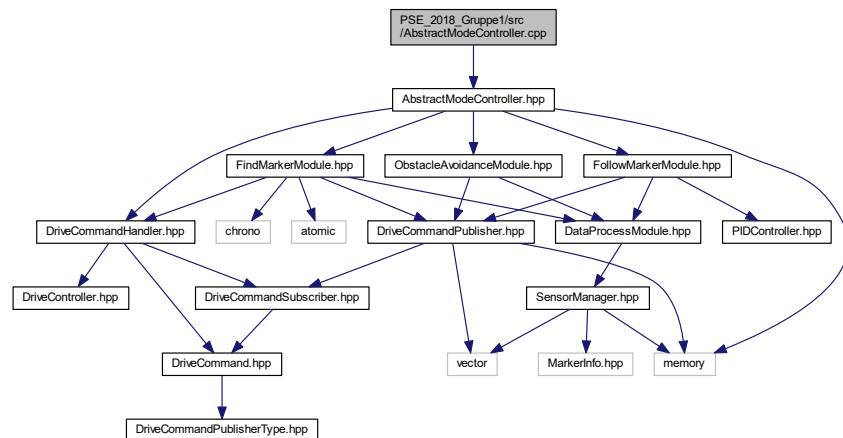
- PSE_2018_Gruppe1/src/test/TerminalHandlerTest.cpp

Chapter 5

File Documentation

5.1 PSE_2018_Gruppe1/src/AbstractModeController.cpp File Reference

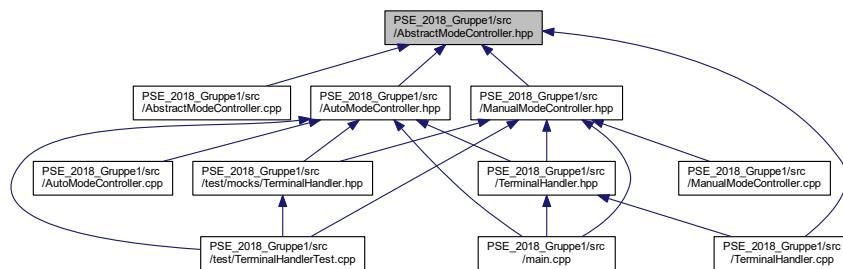
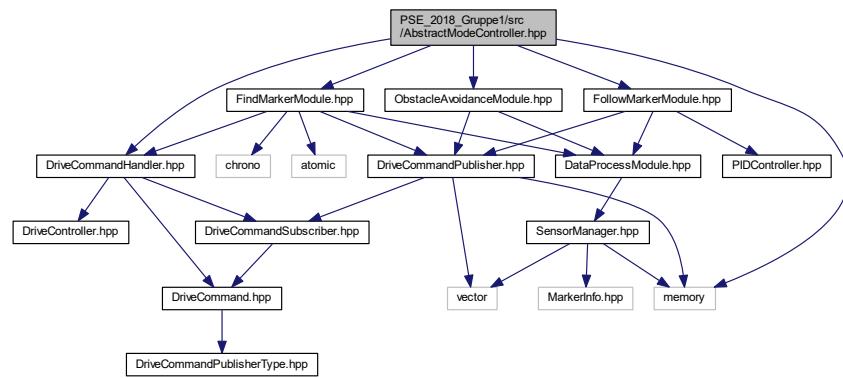
```
#include "AbstractModeController.hpp"  
Include dependency graph for AbstractModeController.cpp:
```



5.2 PSE_2018_Gruppe1/src/AbstractModeController.hpp File Reference

```
#include "DriveCommandHandler.hpp"  
#include "FindMarkerModule.hpp"  
#include "FollowMarkerModule.hpp"  
#include "ObstacleAvoidanceModule.hpp"
```

```
#include <memory>
Include dependency graph for AbstractModeController.hpp:
```



Classes

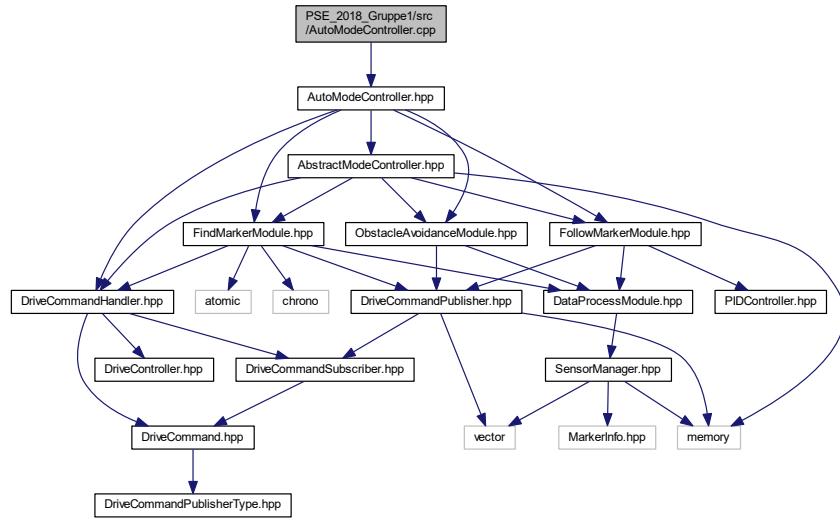
- class [AbstractModeController](#)

Abstract base class for [AutoModeController](#) and [ManualModeController](#).

5.3 PSE_2018_Gruppe1/src/AutoModeController.cpp File Reference

```
#include "AutoModeController.hpp"
```

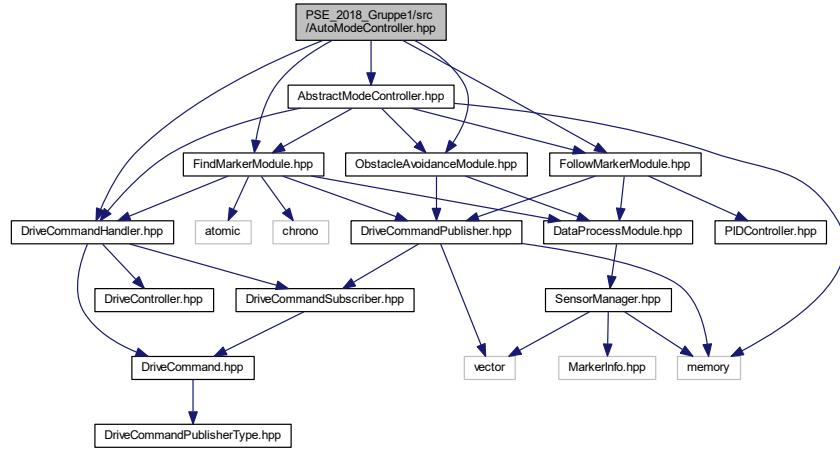
Include dependency graph for AutoModeController.cpp:



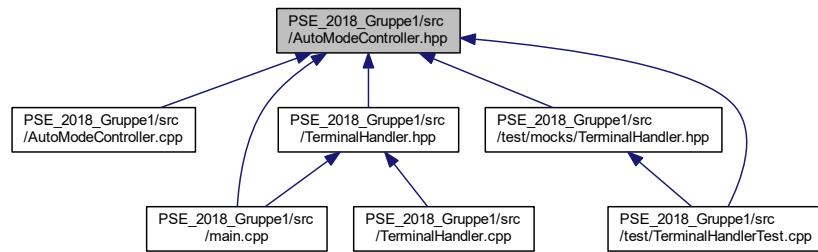
5.4 PSE_2018_Gruppe1/src/AutoModeController.hpp File Reference

```
#include "DriveCommandHandler.hpp"
#include "AbstractModeController.hpp"
#include "ObstacleAvoidanceModule.hpp"
#include "FollowMarkerModule.hpp"
#include "FindMarkerModule.hpp"
Include dependency graph for AutoModeController.hpp:
```

Include dependency graph for AutoModeController.hpp:



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



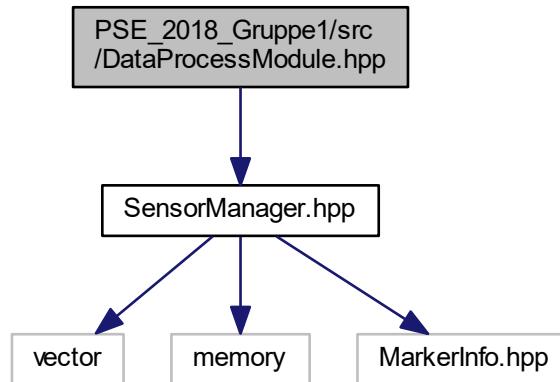
Classes

- class [AutoModeController](#)

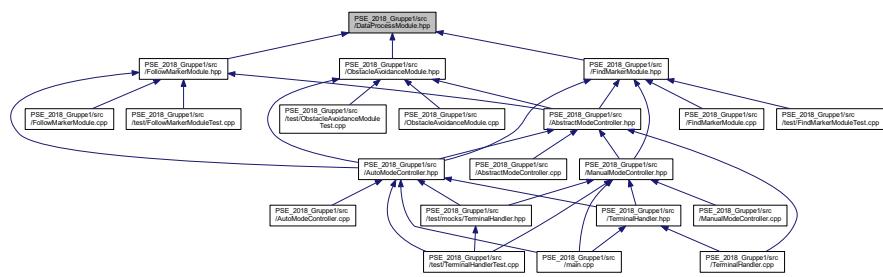
Controls the execution of all three submodules in automatic driving mode.

5.5 PSE_2018_Gruppe1/src/DataProcessModule.hpp File Reference

```
#include "SensorManager.hpp"
Include dependency graph for DataProcessModule.hpp:
```



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



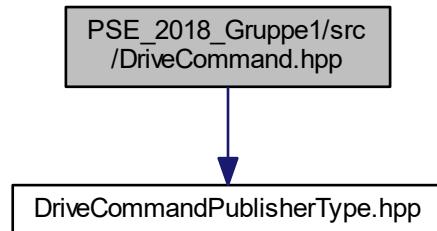
Classes

- class DataProcessModule

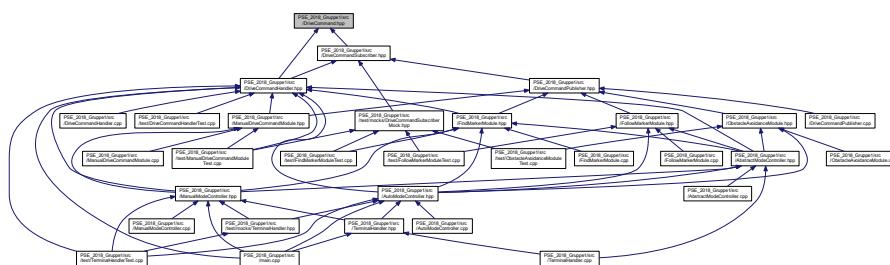
All automatic mode modules that have to work with data from the [SensorManager](#) use this base class as their interface.

5.6 PSE_2018_Gruppe1/src/DriveCommand.hpp File Reference

```
#include "DriveCommandPublisherType.hpp"
Include dependency graph for DriveCommand.hpp:
```



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



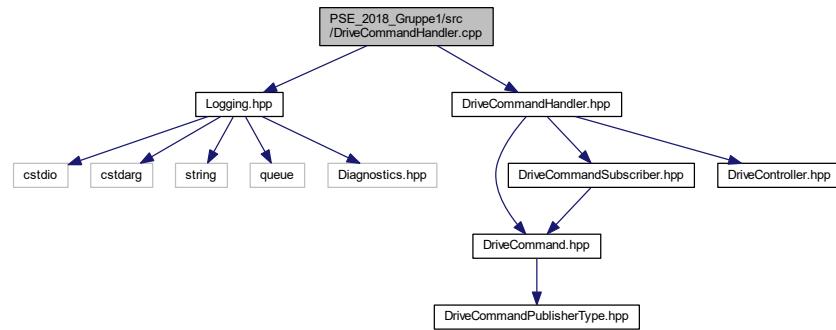
Classes

- struct [DriveCommand](#)

This struct is used during communication between DriveCommandPublishers and DriveCommandSubscribers.

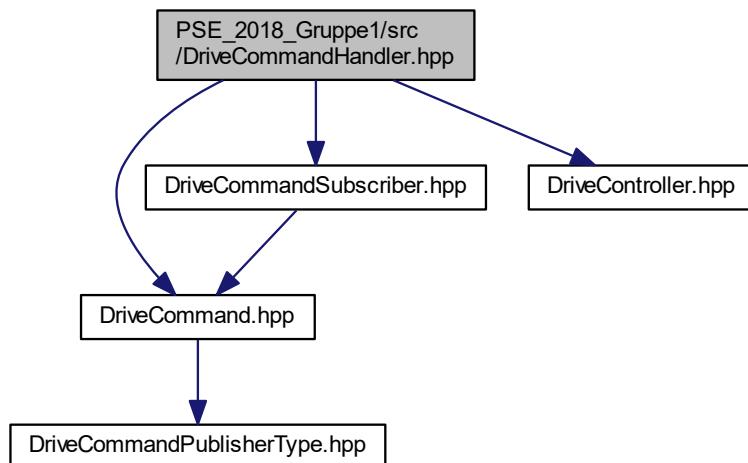
5.7 PSE_2018_Gruppe1/src/DriveCommandHandler.cpp File Reference

```
#include "Logging.hpp"
#include "DriveCommandHandler.hpp"
Include dependency graph for DriveCommandHandler.cpp:
```

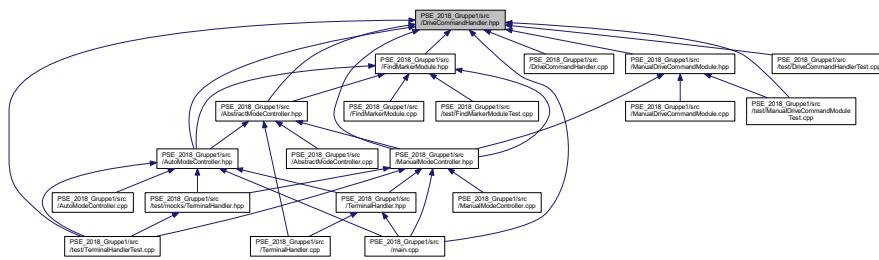


5.8 PSE_2018_Gruppe1/src/DriveCommandHandler.hpp File Reference

```
#include "DriveCommand.hpp"
#include "DriveCommandSubscriber.hpp"
#include "DriveController.hpp"
Include dependency graph for DriveCommandHandler.hpp:
```



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



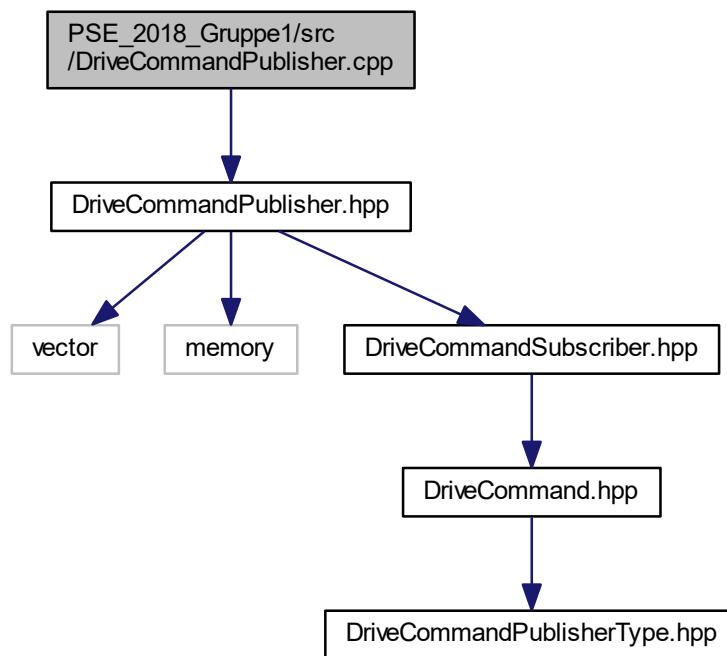
Classes

- class [DriveCommandHandler](#)

This class can receive `DriveCommand` structs, prioritizes them by their source and forwards the values to the `DriveController` once `updateMotor()` gets called.

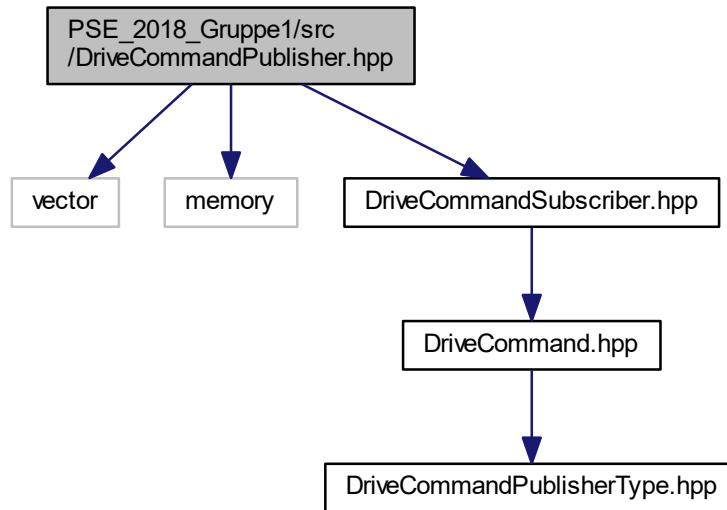
5.9 PSE_2018_Gruppe1/src/DriveCommandPublisher.cpp File Reference

```
#include "DriveCommandPublisher.hpp"
Include dependency graph for DriveCommandPublisher.cpp:
```

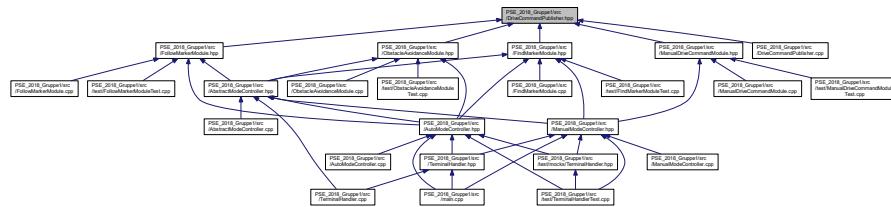


5.10 PSE_2018_Gruppe1/src/DriveCommandPublisher.hpp File Reference

```
#include <vector>
#include <memory>
#include "DriveCommandSubscriber.hpp"
Include dependency graph for DriveCommandPublisher.hpp:
```



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



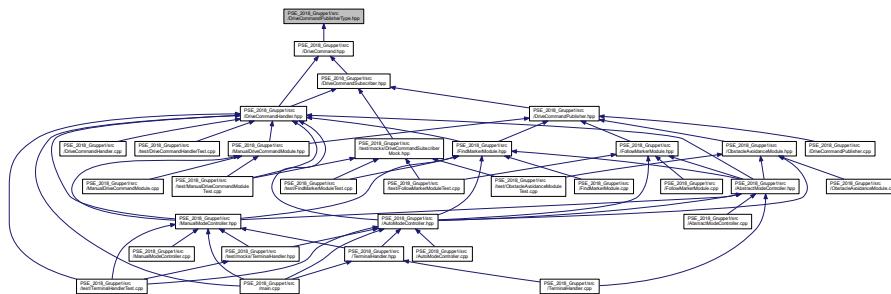
Classes

- class [DriveCommandPublisher](#)

Base class for all classes that are able to publish DriveCommands to a [DriveCommandSubscriber](#).

5.11 PSE_2018_Gruppe1/src/DriveCommandPublisherType.hpp File Reference

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



Enumerations

- enum `DriveCommandPublisherType` {
 `NO_COMMAND` = 0x0, `FIND_MARKER` = 0x1, `FOLLOW_MARKER` = 0x2, `OBSTACLE_AVOIDANCE` = 0x3, `MANUAL_COMMAND` = 0x4
 }

The `DriveCommand` types, with following priorities low: `FIND_MARKER FOLLOW_MARKER OBSTACLE_AVOIDANCE` high `MANUAL_COMMAND`.

5.11.1 Enumeration Type Documentation

5.11.1.1 DriveCommandPublisherType

```
enum DriveCommandPublisherType
```

The `DriveCommand` types, with following priorities low: `FIND_MARKER FOLLOW_MARKER OBSTACLE_AVOIDANCE` high `MANUAL_COMMAND`.

Enumerator

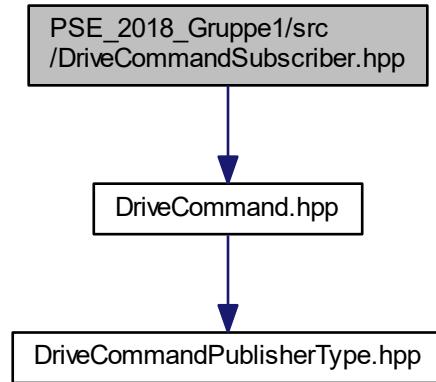
<code>NO_COMMAND</code>	
<code>FIND_MARKER</code>	
<code>FOLLOW_MARKER</code>	
<code>OBSTACLE_AVOIDANCE</code>	
<code>MANUAL_COMMAND</code>	

Definition at line 17 of file `DriveCommandPublisherType.hpp`.

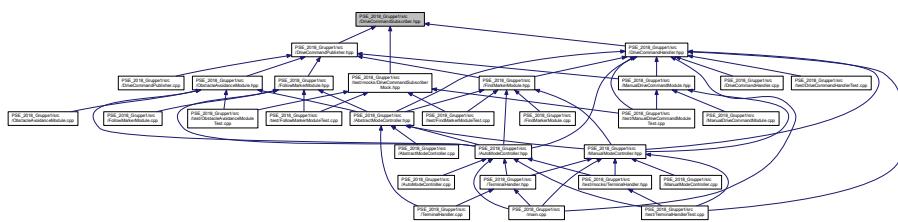
5.12 PSE_2018_Gruppe1/src/DriveCommandSubscriber.hpp File Reference

```
#include "DriveCommand.hpp"
```

Include dependency graph for DriveCommandSubscriber.hpp:



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



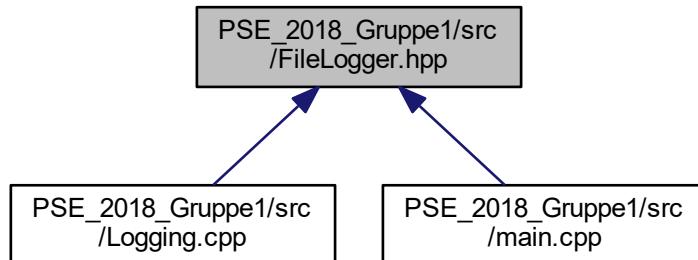
Classes

- class [DriveCommandSubscriber](#)

DriveCommandSubscribers can receive [DriveCommand](#) structs from objects that are derived from [DriveCommandPublisher](#).

5.13 PSE_2018_Gruppe1/src/FileLogger.hpp File Reference

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



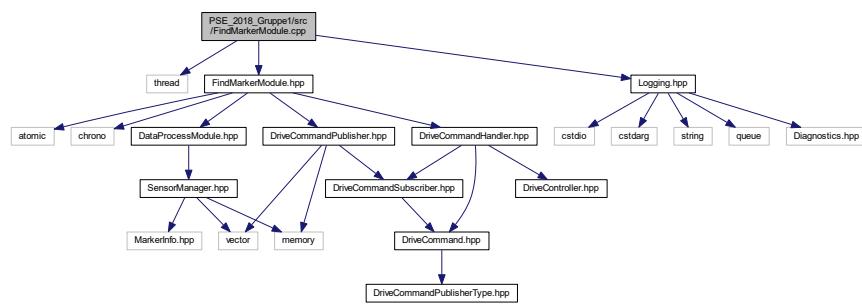
Classes

- class [FileLogger](#)
Class to write log messages to a file called "events.log".

5.14 PSE_2018_Gruppe1/src/FindMarkerModule.cpp File Reference

```
#include <thread>
#include "FindMarkerModule.hpp"
#include "Logging.hpp"
```

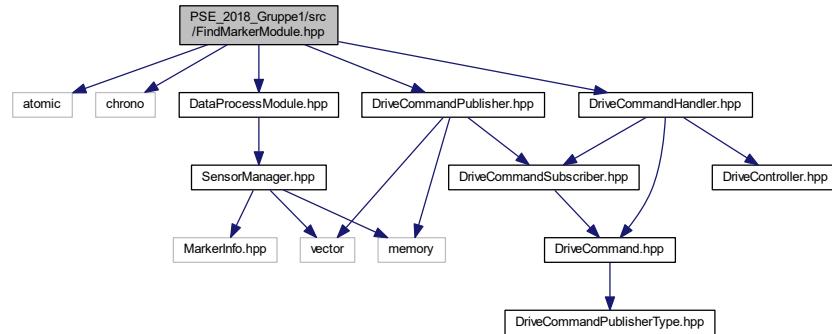
Include dependency graph for `FindMarkerModule.cpp`:



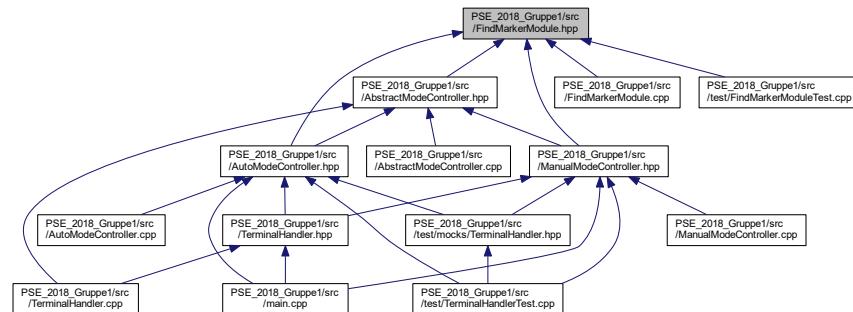
5.15 PSE_2018_Gruppe1/src/FindMarkerModule.hpp File Reference

```
#include <atomic>
#include <chrono>
```

```
#include "DriveCommandPublisher.hpp"
#include "DataProcessModule.hpp"
#include "DriveCommandHandler.hpp"
Include dependency graph for FindMarkerModule.hpp:
```



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



Classes

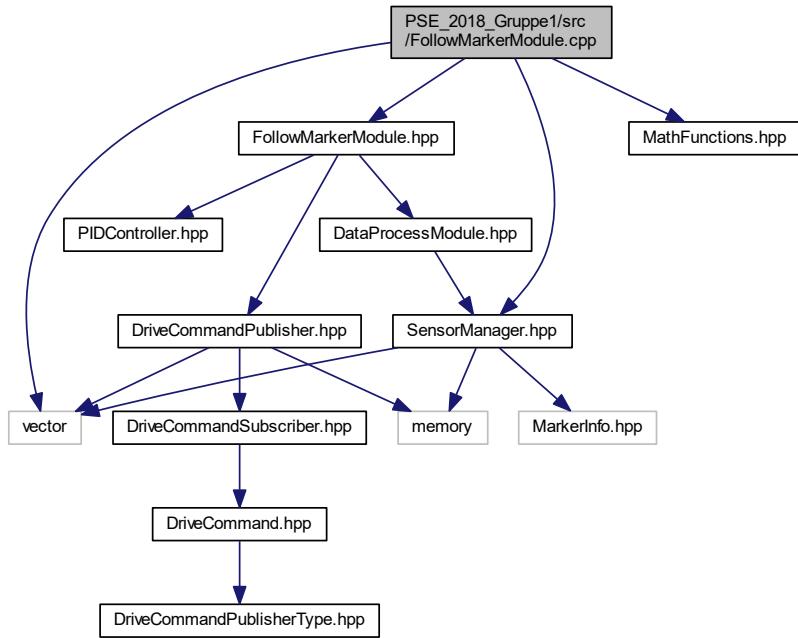
- class [FindMarkerModule](#)

This module can be active in manual and in auto mode.

5.16 PSE_2018_Gruppe1/src/FollowMarkerModule.cpp File Reference

```
#include <vector>
#include "SensorManager.hpp"
#include "FollowMarkerModule.hpp"
```

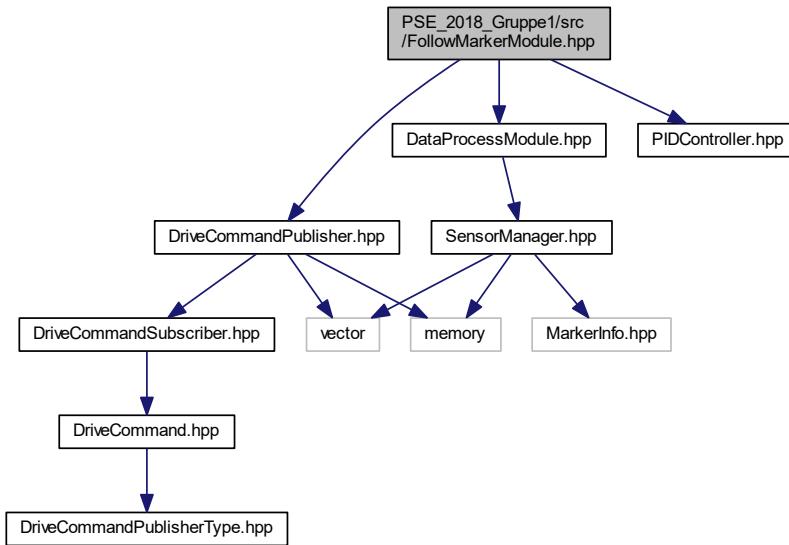
```
#include "MathFunctions.hpp"
Include dependency graph for FollowMarkerModule.cpp:
```



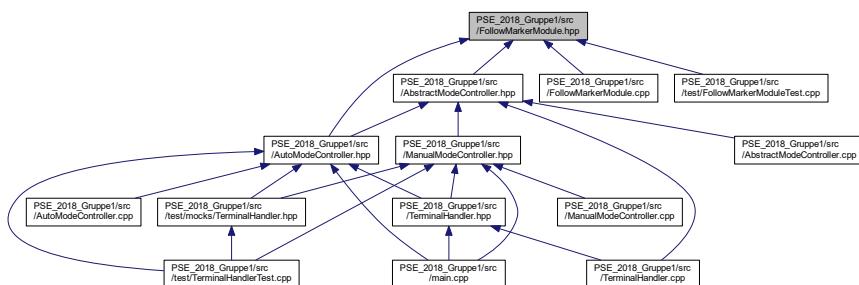
5.17 PSE_2018_Gruppe1/src/FollowMarkerModule.hpp File Reference

```
#include "DriveCommandPublisher.hpp"
#include "DataProcessModule.hpp"
#include "PIDController.hpp"
```

Include dependency graph for FollowMarkerModule.hpp:



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



Classes

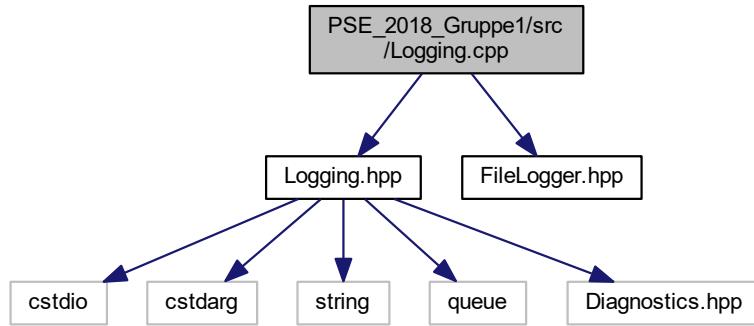
- class [FollowMarkerModule](#)

This module retrieves the markerList from the [SensorManager](#), checks for markers and tries to follow the one with the best confidence until it loses track of it.

5.18 PSE_2018_Gruppe1/src/Logging.cpp File Reference

```
#include "Logging.hpp"
#include "FileLogger.hpp"
```

Include dependency graph for Logging.cpp:



Variables

- static DiagnosticLevel sLogLevel = DiagnosticLevel::DIAG_VERBOSE

5.18.1 Variable Documentation

5.18.1.1 sLogLevel

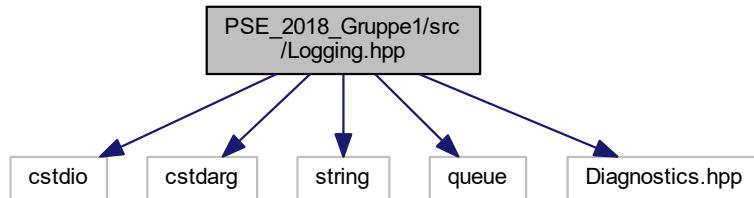
DiagnosticLevel sLogLevel = DiagnosticLevel::DIAG_VERBOSE [static]

Definition at line 8 of file Logging.cpp.

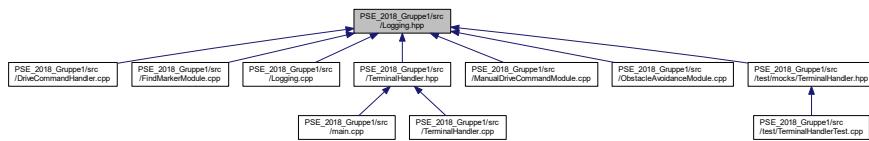
5.19 PSE_2018_Gruppe1/src/Logging.hpp File Reference

```
#include <cstdio>
#include <cstdarg>
#include <string>
#include <queue>
#include <Diagnostics.hpp>
```

Include dependency graph for Logging.hpp:



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



Classes

- class [Logging](#)

This is a wrapper class for Diagnostics.

Macros

- `#define DIAG_VERBOSE(msg, ...) Logging::log(DiagnosticLevel::DIAG_VERBOSE, __FILE__, __LINE__←, msg, ##__VA_ARGS__)`
- `#define DIAG_DEBUG(msg, ...) Logging::log(DiagnosticLevel::DIAG_DEBUG, __FILE__, __LINE__, msg, ##__VA_ARGS__)`
- `#define DIAG_INFO(msg, ...) Logging::log(DiagnosticLevel::DIAG_INFO, __FILE__, __LINE__, msg, ##__VA_ARGS__)`
- `#define DIAG_WARNING(msg, ...) Logging::log(DiagnosticLevel::DIAG_WARNING, __FILE__, __LINE__←, msg, ##__VA_ARGS__)`
- `#define DIAG_ERROR(msg, ...) Logging::log(DiagnosticLevel::DIAG_ERROR, __FILE__, __LINE__, msg, ##__VA_ARGS__)`

5.19.1 Macro Definition Documentation

5.19.1.1 DIAG_DEBUG

```
#define DIAG_DEBUG (
    msg,
    ... ) Logging::log(DiagnosticLevel::DIAG_DEBUG, __FILE__, __LINE__, msg, ##__VA_ARGS__)
```

Definition at line 22 of file Logging.hpp.

5.19.1.2 DIAG_ERROR

```
#define DIAG_ERROR (
    msg,
    ... ) Logging::log(DiagnosticLevel::DIAG_ERROR, __FILE__, __LINE__, msg, ##__VA_ARGS__)
```

Definition at line 25 of file Logging.hpp.

5.19.1.3 DIAG_INFO

```
#define DIAG_INFO(
    msg,
    ... ) Logging::log(DiagnosticLevel::DIAG_INFO, __FILE__, __LINE__, msg, ##__VA_ARGS__)
```

Definition at line 23 of file Logging.hpp.

5.19.1.4 DIAG_VERBOSE

```
#define DIAG_VERBOSE(
    msg,
    ... ) Logging::log(DiagnosticLevel::DIAG_VERBOSE, __FILE__, __LINE__, msg, ##__VA_ARGS__)
```

Definition at line 21 of file Logging.hpp.

5.19.1.5 DIAG_WARNING

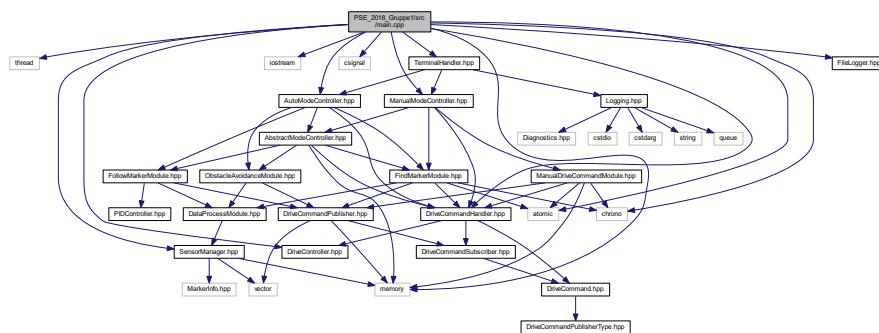
```
#define DIAG_WARNING(
    msg,
    ... ) Logging::log(DiagnosticLevel::DIAG_WARNING, __FILE__, __LINE__, msg, ##__VA_ARGS__)
```

Definition at line 24 of file Logging.hpp.

5.20 PSE_2018_Gruppe1/src/main.cpp File Reference

```
#include <thread>
#include <atomic>
#include <chrono>
#include <iostream>
#include <memory>
#include <csignal>
#include <SensorManager.hpp>
#include <DriveController.hpp>
#include "AutoModeController.hpp"
#include "ManualModeController.hpp"
#include "DriveCommandHandler.hpp"
#include "TerminalHandler.hpp"
```

```
#include "FileLogger.hpp"
Include dependency graph for main.cpp:
```



Functions

- void `motorStep ()`
this functions has to run the whole time the program is alive.
- int `main (int argc, char **argv)`
Main function of the project.

Variables

- `std::atomic< bool > g_doStep`
- `std::atomic< bool > g_appRunning`
- static `bool s_flushLogInstant`

5.20.1 Function Documentation

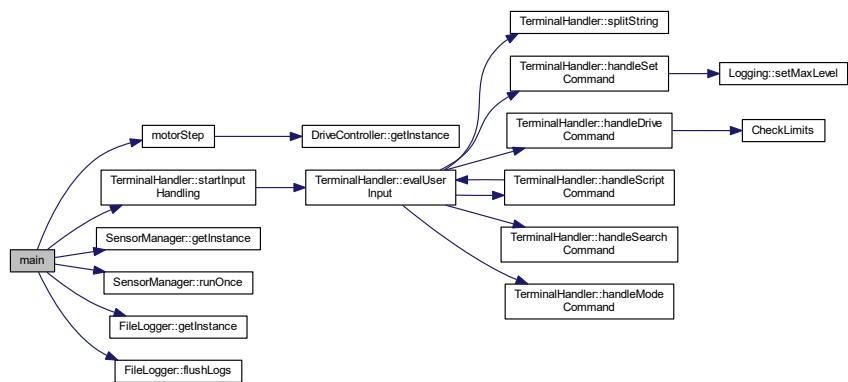
5.20.1.1 main()

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

Main function of the project.

Definition at line 45 of file `main.cpp`.

Here is the call graph for this function:



5.20.1.2 `motorStep()`

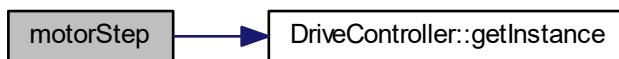
```
void motorStep ( )
```

this functions has to run the whole time the program is alive.

It calls `step()` on the `DriveController` every 10 ms.

Definition at line 26 of file `main.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.2 Variable Documentation

5.20.2.1 g_appRunning

```
std::atomic<bool> g_appRunning
```

Definition at line 18 of file main.cpp.

5.20.2.2 g_doStep

```
std::atomic<bool> g_doStep
```

Definition at line 17 of file main.cpp.

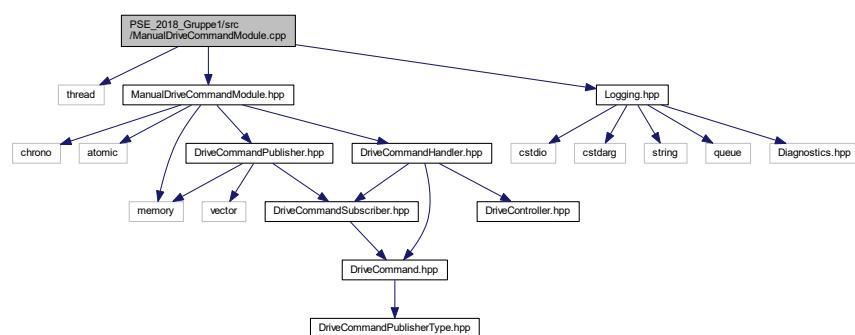
5.20.2.3 s_flushLogInstant

```
bool s_flushLogInstant [static]
```

Definition at line 20 of file main.cpp.

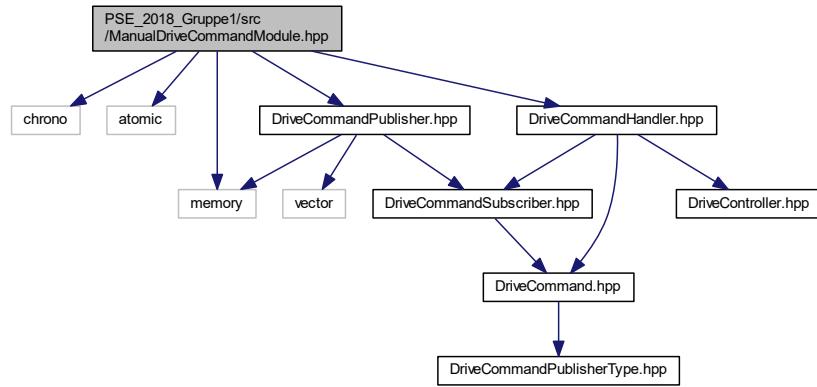
5.21 PSE_2018_Gruppe1/src/ManualDriveCommandModule.cpp File Reference

```
#include <thread>
#include "ManualDriveCommandModule.hpp"
#include "Logging.hpp"
Include dependency graph for ManualDriveCommandModule.cpp:
```

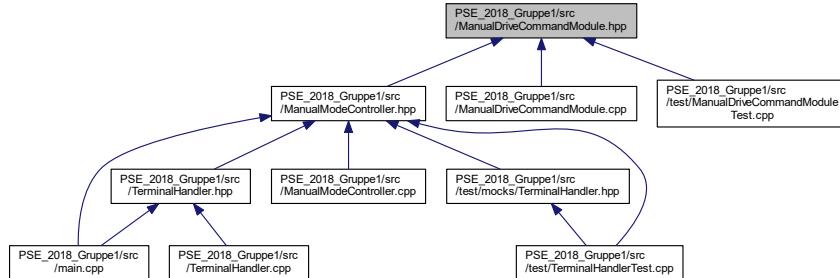


5.22 PSE_2018_Gruppe1/src/ManualDriveCommandModule.hpp File Reference

```
#include <chrono>
#include <atomic>
#include <memory>
#include "DriveCommandPublisher.hpp"
#include "DriveCommandHandler.hpp"
Include dependency graph for ManualDriveCommandModule.hpp:
```



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



Classes

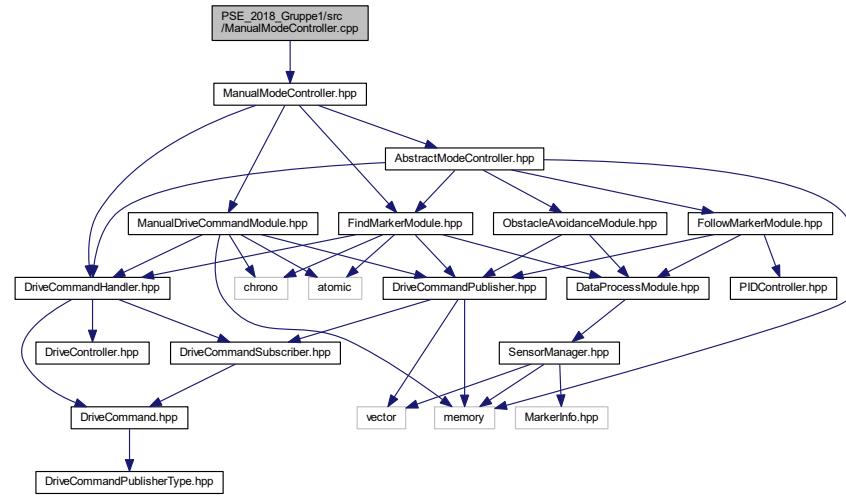
- class [ManualDriveCommandModule](#)

This module is used for blocking drive operations via terminal command or script and can only be invoked via terminal.

5.23 PSE_2018_Gruppe1/src/ManualModeController.cpp File Reference

```
#include "ManualModeController.hpp"
```

Include dependency graph for ManualModeController.cpp:

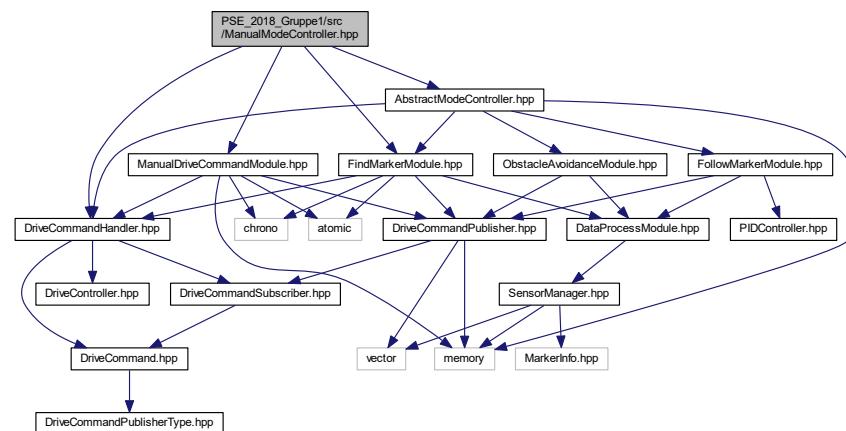


5.24 PSE_2018_Gruppe1/src/ManualModeController.hpp File Reference

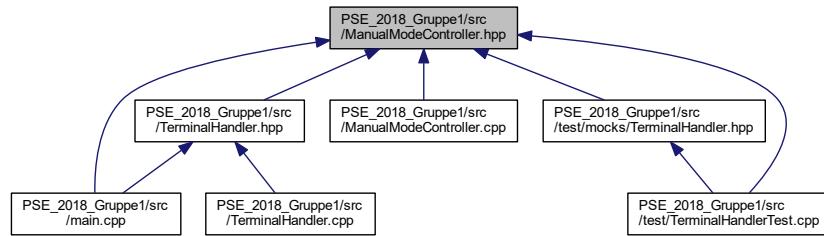
```

#include "DriveCommandHandler.hpp"
#include "ManualDriveCommandModule.hpp"
#include "AbstractModeController.hpp"
#include "FindMarkerModule.hpp"
  
```

Include dependency graph for ManualModeController.hpp:



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

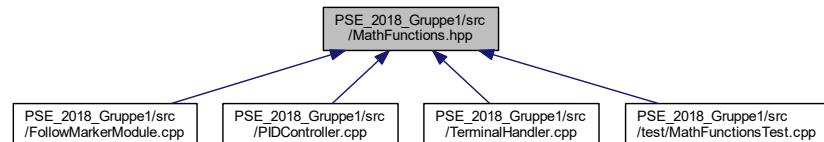


Classes

- class [ManualModeController](#)
Child of [AbstractModeController](#) that is used for manual operation.

5.25 PSE_2018_Gruppe1/src/MathFunctions.hpp File Reference

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



Functions

- template<class t >
`static t LimitBetween (t val, t lowerBound, t upperBound)`
Limits two arbitrary values between lowerBound and upperBound (inclusive the borders)
- template<class t >
`static bool CheckLimits (t val, t lowerBound, t upperBound)`
Checks whether a specified value lies between two specified bounds.

5.25.1 Function Documentation

5.25.1.1 CheckLimits()

```
template<class t >
static bool CheckLimits (
    t val,
    t lowerBound,
    t upperBound ) [static]
```

Checks whether a specified value lies between two specified bounds.

Template Parameters

<i>t</i>	Value types (most of the time implicitly inferred by the compiler)
----------	--

Parameters

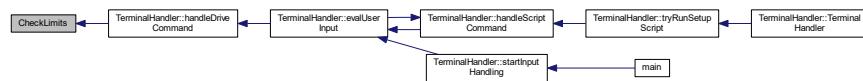
<i>val</i>	Value to check
<i>lowerBound</i>	Lower bound for the value
<i>upperBound</i>	Upper bound for the value

Returns

true if *lowerBound* <= *val* <= *upperBound*

Definition at line 40 of file MathFunctions.hpp.

Here is the caller graph for this function:

**5.25.1.2 LimitBetween()**

```
template<class t >
static t LimitBetween (
    t val,
    t lowerBound,
    t upperBound ) [static]
```

Limits two arbitrary values between *lowerBound* and *upperBound* (inclusive the borders)

Template Parameters

<i>t</i>	Value types (most of the time implicitly inferred by the compiler)
----------	--

Parameters

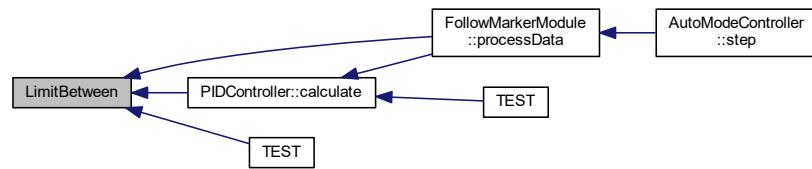
<i>val</i>	Value that has to be limited.
<i>lowerBound</i>	Lower boundary for the return value.
<i>upperBound</i>	Upper boundary for the return value.

Returns

returns val if it is within the bounds, lowerBound resp. upperBound otherwise.

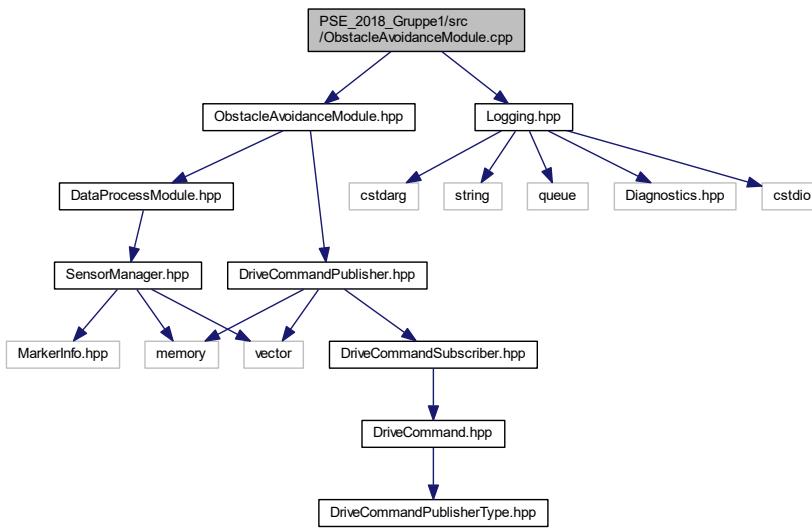
Definition at line 19 of file MathFunctions.hpp.

Here is the caller graph for this function:



5.26 PSE_2018_Gruppe1/src/ObstacleAvoidanceModule.cpp File Reference

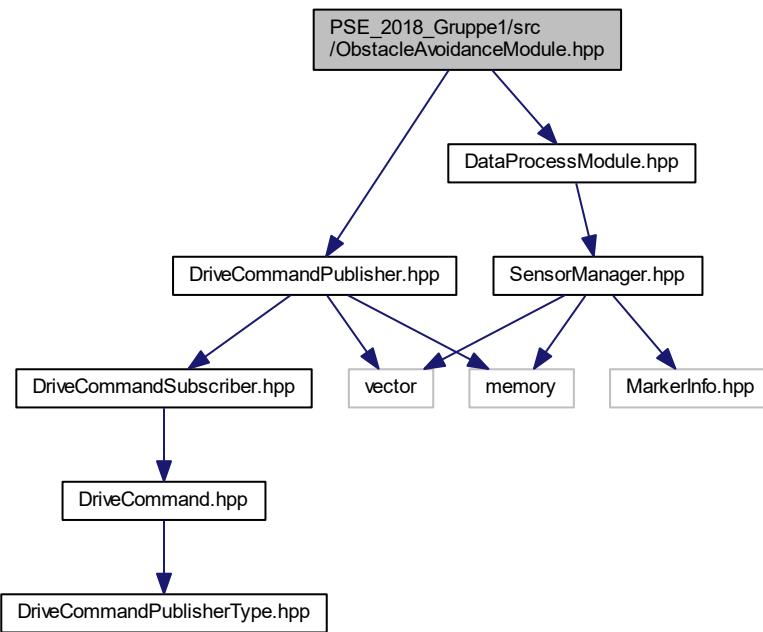
```
#include "ObstacleAvoidanceModule.hpp"
#include "Logging.hpp"
Include dependency graph for ObstacleAvoidanceModule.cpp:
```



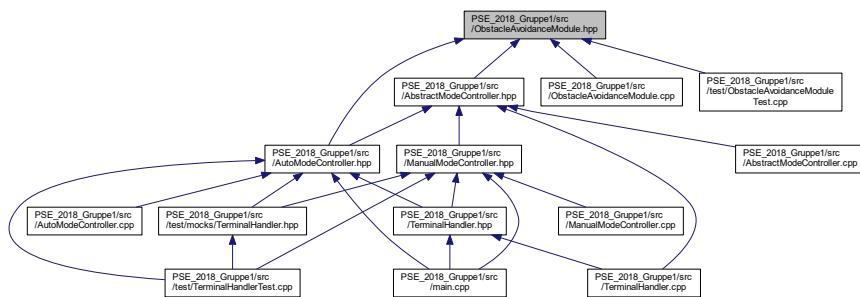
5.27 PSE_2018_Gruppe1/src/ObstacleAvoidanceModule.hpp File Reference

```
#include "DriveCommandPublisher.hpp"
#include "DataProcessModule.hpp"
```

Include dependency graph for ObstacleAvoidanceModule.hpp:



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



Classes

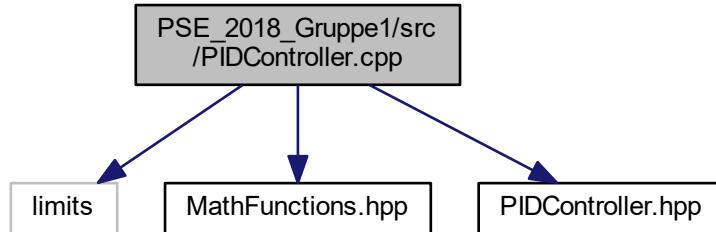
- class `ObstacleAvoidanceModule`

Used in automatic mode to detect and avoid obstacles during automatic driving operations.

5.28 PSE_2018_Gruppe1/src/PIDController.cpp File Reference

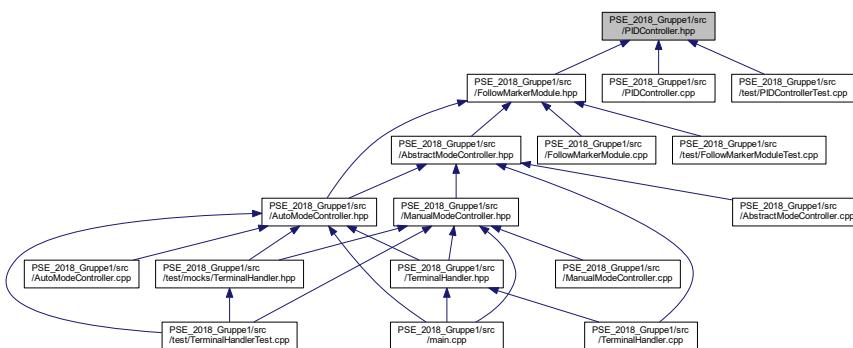
```
#include <limits>
#include "MathFunctions.hpp"
```

```
#include "PIDController.hpp"
Include dependency graph for PIDController.cpp:
```



5.29 PSE_2018_Gruppe1/src/PIDController.hpp File Reference

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



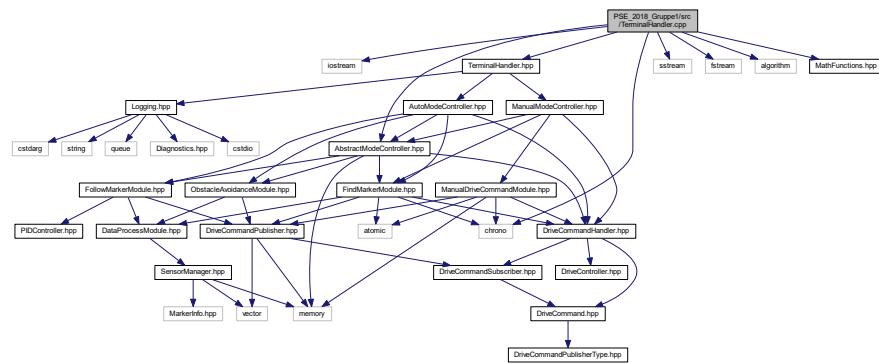
Classes

- class [PIDController](#)
Basic universal PID controller implementation.

5.30 PSE_2018_Gruppe1/src/TerminalHandler.cpp File Reference

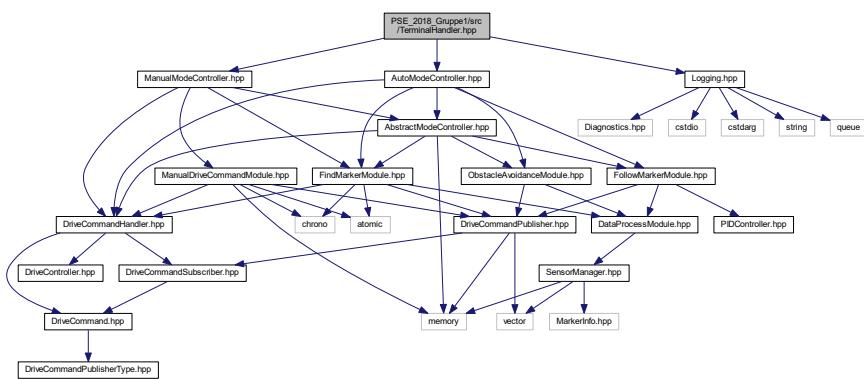
```
#include <iostream>
#include <chrono>
#include <sstream>
#include <fstream>
#include <algorithm>
#include "AbstractModeController.hpp"
```

```
#include "TerminalHandler.hpp"
#include "MathFunctions.hpp"
Include dependency graph for TerminalHandler.cpp:
```

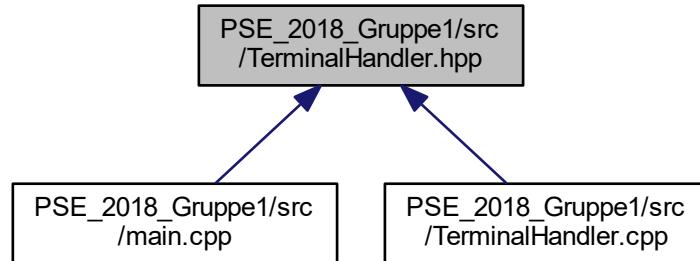


5.31 PSE_2018_Gruppe1/src/TerminalHandler.hpp File Reference

```
#include "AutoModeController.hpp"
#include "ManualModeController.hpp"
#include "Logging.hpp"
Include dependency graph for TerminalHandler.hpp:
```



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



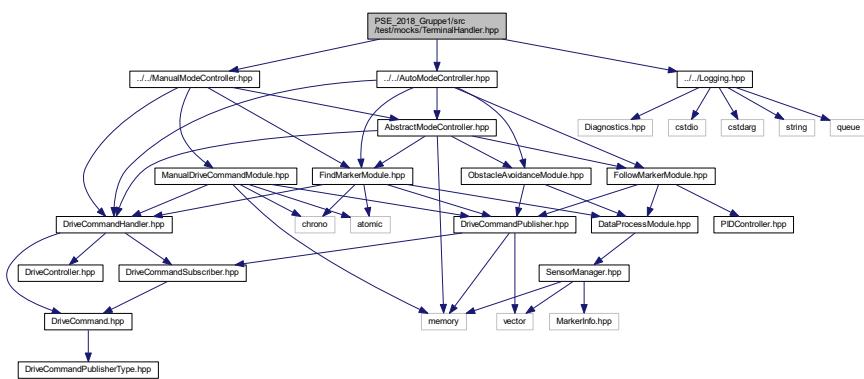
Classes

- class [TerminalHandler](#)

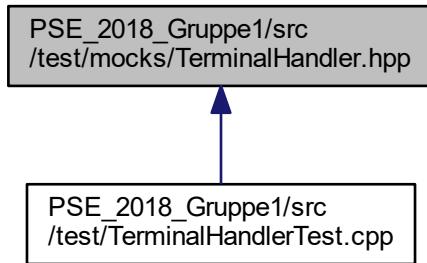
5.32 PSE_2018_Gruppe1/src/test/mocks/TerminalHandler.hpp File Reference

```
#include "../../AutoModeController.hpp"
#include "../../ManualModeController.hpp"
#include "../../Logging.hpp"
```

Include dependency graph for TerminalHandler.hpp:



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

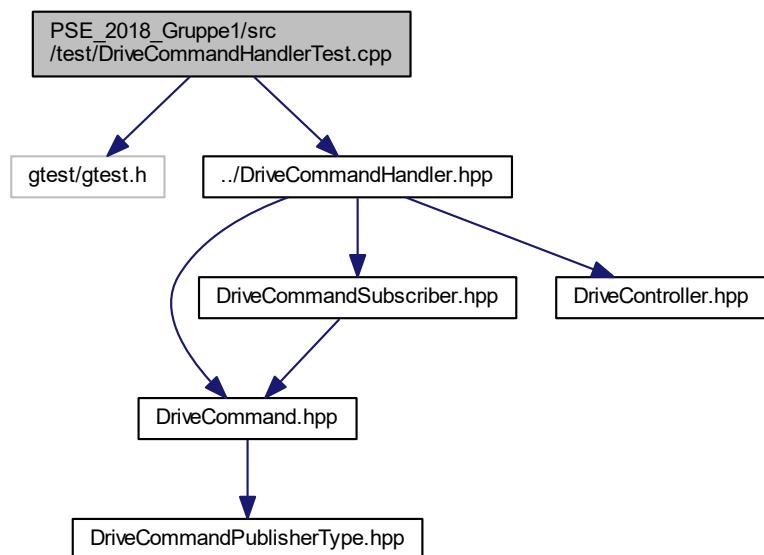


Classes

- class [TerminalHandler](#)

5.33 PSE_2018_Gruppe1/src/test/DriveCommandHandlerTest.cpp File Reference

```
#include <gtest/gtest.h>
#include "../DriveCommandHandler.hpp"
Include dependency graph for DriveCommandHandlerTest.cpp:
```



Functions

- TEST (DriveCommandHandler, commandPriorities)

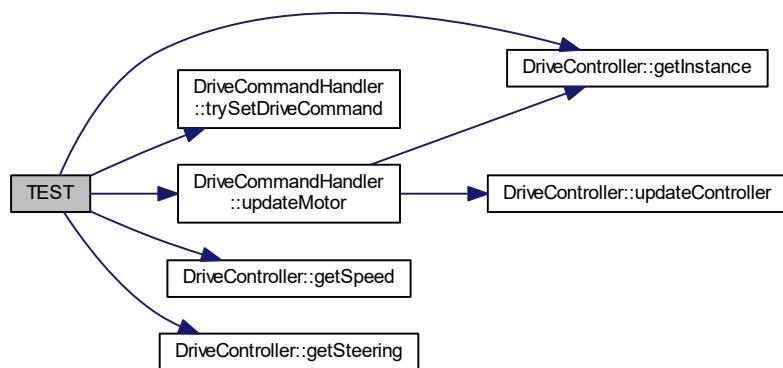
5.33.1 Function Documentation

5.33.1.1 TEST()

```
TEST (
    DriveCommandHandler ,
    commandPriorities )
```

Definition at line 4 of file DriveCommandHandlerTest.cpp.

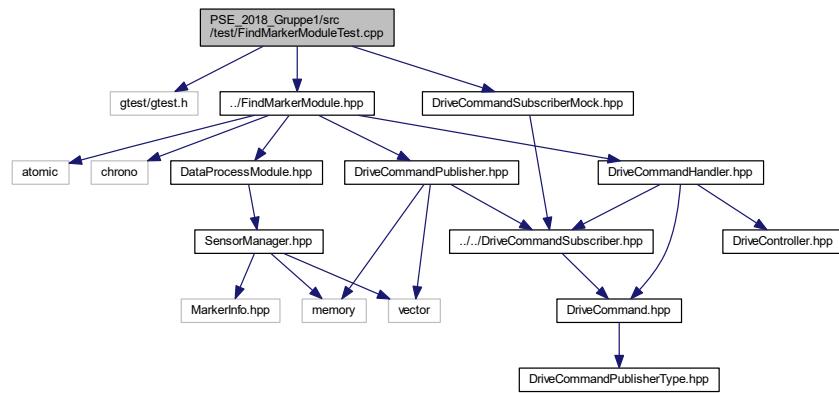
Here is the call graph for this function:



5.34 PSE_2018_Gruppe1/src/test/FindMarkerModuleTest.cpp File Reference

```
#include <gtest/gtest.h>
#include <DriveCommandSubscriberMock.hpp>
```

```
#include "../FindMarkerModule.hpp"
Include dependency graph for FindMarkerModuleTest.cpp:
```



Functions

- `TEST` (`FindMarkerModuleTest`, `findMarkerTest`)
- `TEST` (`FindMarkerModuleTest`, `findMarkerBlockingTest`)

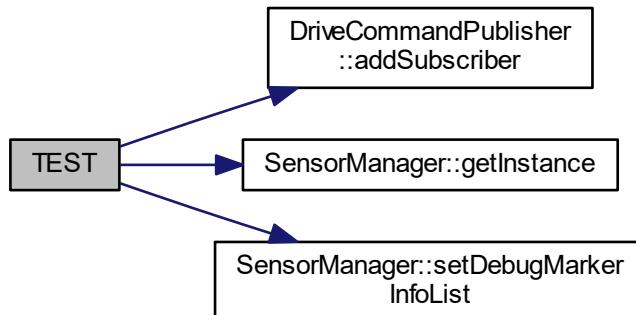
5.34.1 Function Documentation

5.34.1.1 TEST() [1/2]

```
TEST (
    FindMarkerModuleTest ,
    findMarkerTest )
```

Definition at line 5 of file `FindMarkerModuleTest.cpp`.

Here is the call graph for this function:

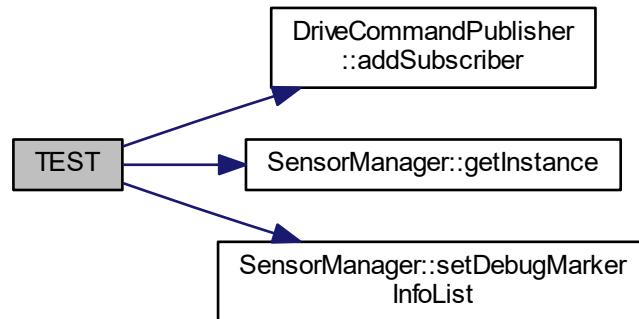


5.34.1.2 TEST() [2/2]

```
TEST (
    FindMarkerModuleTest ,
    findMarkerBlockingTest )
```

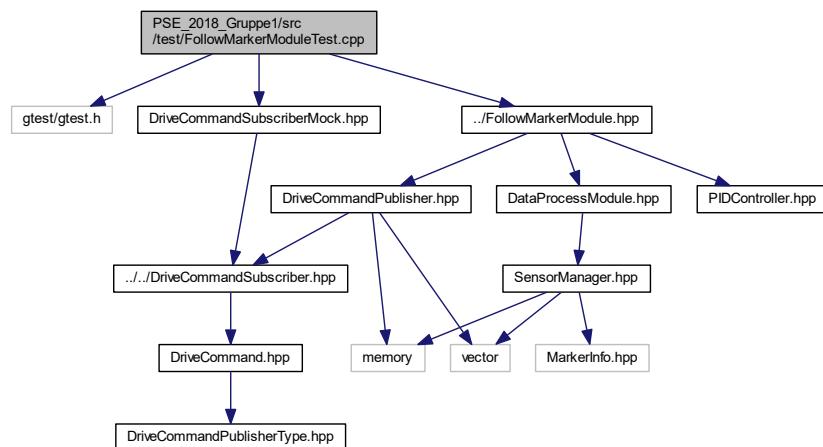
Definition at line 46 of file FindMarkerModuleTest.cpp.

Here is the call graph for this function:



5.35 PSE_2018_Gruppe1/src/test/FollowMarkerModuleTest.cpp File Reference

```
#include <gtest/gtest.h>
#include <DriveCommandSubscriberMock.hpp>
#include "../FollowMarkerModule.hpp"
Include dependency graph for FollowMarkerModuleTest.cpp:
```



Functions

- **TEST** (FollowMarkerModuleTest, no_marker)

If there is no marker for the `FollowMarkerModule` to find, it should not send any commands.
- **TEST** (FollowMarkerModuleTest, marker_right)

If the marker is to the right of the sensor's center, `FindMarkerModule` should start commanding values that make the robot steer right.
- **TEST** (FollowMarkerModuleTest, marker_left)

Check whether the `FollowMarkerModule` correctly commands negative steering values if we have a marker to the left.
- **TEST** (FollowMarkerModuleTest, marker_near)

If we are closer to the marker than we should be, `FollowMarkerModeul` should emit commands that make the robot drive back.
- **TEST** (FollowMarkerModuleTest, marker_far)

Check whether the `FollowMarkerModule` correctly drives towards a marker if the desired distance has not yet been reached.

5.35.1 Function Documentation

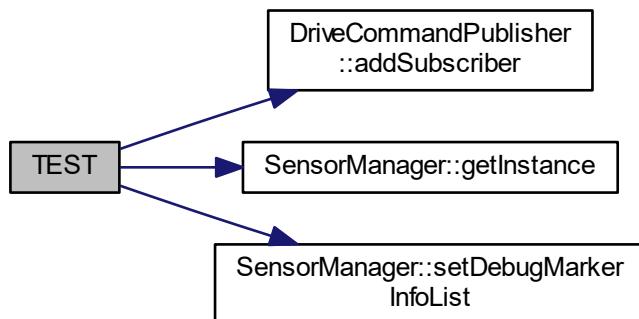
5.35.1.1 TEST() [1/5]

```
TEST (
    FollowMarkerModuleTest ,
    no_marker )
```

If there is no marker for the `FollowMarkerModule` to find, it should not send any commands.

Definition at line 9 of file FollowMarkerModuleTest.cpp.

Here is the call graph for this function:



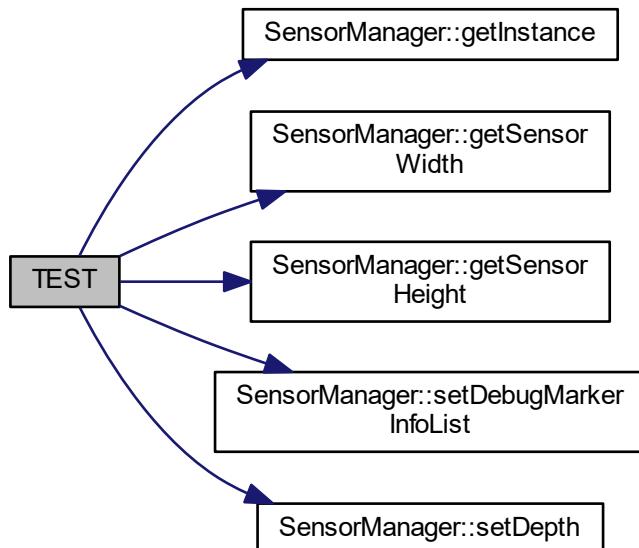
5.35.1.2 TEST() [2/5]

```
TEST (
    FollowMarkerModuleTest ,
    marker_right )
```

If the marker is to the right of the sensor's center, [FindMarkerModule](#) should start commanding values that make the robot steer right.

Definition at line 32 of file [FollowMarkerModuleTest.cpp](#).

Here is the call graph for this function:



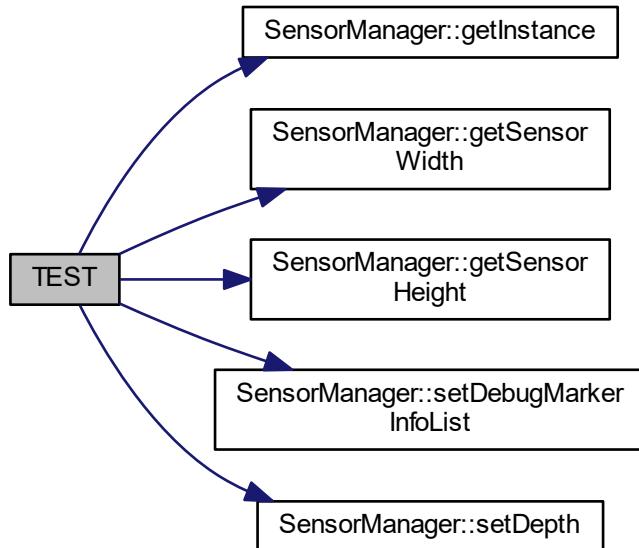
5.35.1.3 TEST() [3/5]

```
TEST (
    FollowMarkerModuleTest ,
    marker_left )
```

Check whether the [FollowMarkerModule](#) correctly commands negative steering values if we have a marker to the left.

Definition at line 65 of file [FollowMarkerModuleTest.cpp](#).

Here is the call graph for this function:



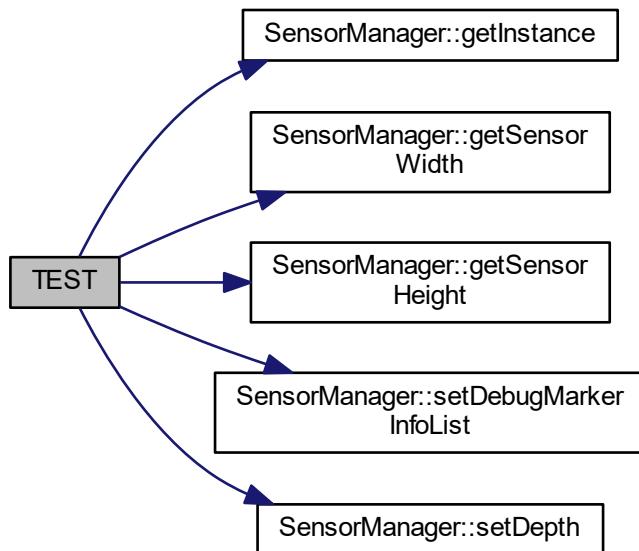
5.35.1.4 TEST() [4/5]

```
TEST (
    FollowMarkerModuleTest ,
    marker_near )
```

If we are closer to the marker than we should be, FollowMarkerModeul should emit commands that make the robot drive back.

Definition at line 98 of file FollowMarkerModuleTest.cpp.

Here is the call graph for this function:



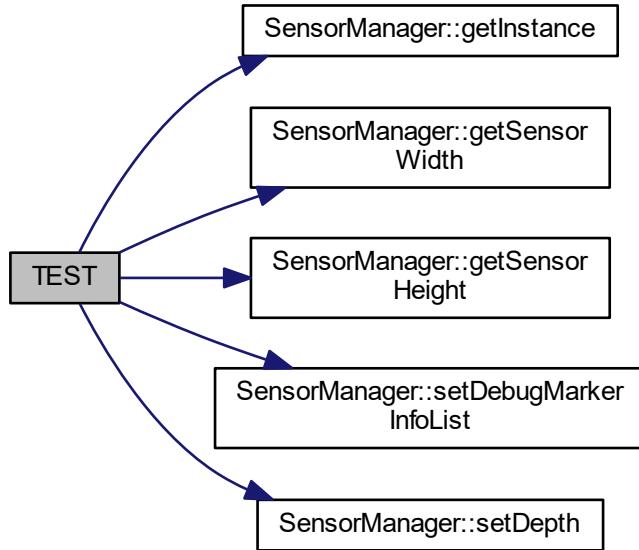
5.35.1.5 TEST() [5/5]

```
TEST (
    FollowMarkerModuleTest ,
    marker_far )
```

Check whether the [FollowMarkerModule](#) correctly drives towards a marker if the desired distance has not yet been reached.

Definition at line 131 of file [FollowMarkerModuleTest.cpp](#).

Here is the call graph for this function:

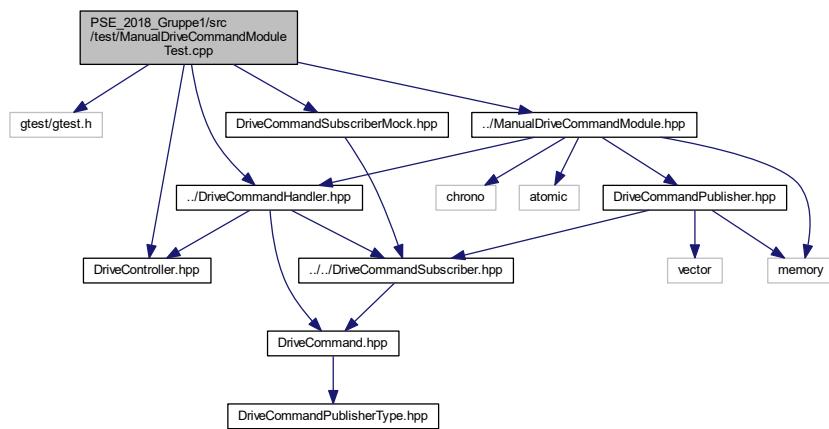


5.36 PSE_2018_Gruppe1/src/test/ManualDriveCommandModuleTest.cpp File Reference

```

#include <gtest/gtest.h>
#include <DriveCommandSubscriberMock.hpp>
#include <DriveController.hpp>
#include "../DriveCommandHandler.hpp"
#include "../ManualDriveCommandModule.hpp"
  
```

Include dependency graph for ManualDriveCommandModuleTest.cpp:



Functions

- TEST (ManualDriveCommandModuleTest, driveBlockingTest)

5.36.1 Function Documentation

5.36.1.1 TEST()

```
TEST (
    ManualDriveCommandModuleTest ,
    driveBlockingTest )
```

Definition at line 7 of file ManualDriveCommandModuleTest.cpp.

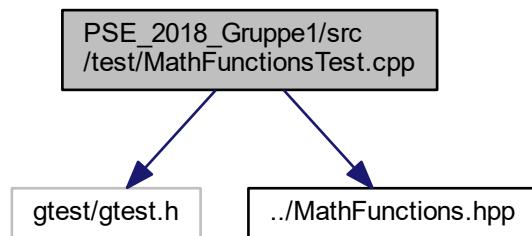
Here is the call graph for this function:



5.37 PSE_2018_Gruppe1/src/test/MathFunctionsTest.cpp File Reference

```
#include <gtest/gtest.h>
#include "../MathFunctions.hpp"
```

Include dependency graph for MathFunctionsTest.cpp:



Functions

- [TEST](#) (MathFunctionsTest, LIMITRANGE)

5.37.1 Function Documentation

5.37.1.1 TEST()

```
TEST (
    MathFunctionsTest ,
    LIMITRANGE )
```

Definition at line 5 of file MathFunctionsTest.cpp.

Here is the call graph for this function:



5.38 PSE_2018_Gruppe1/src/test/mocks/BoundingBox.hpp File Reference

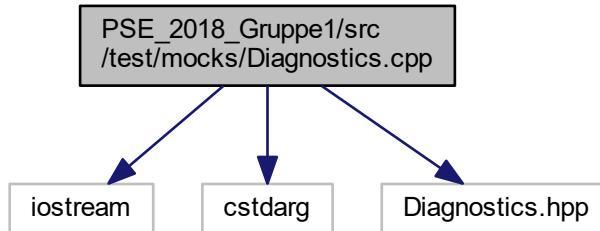
Classes

- class [Color](#)
RGB Color.
- class [BoundingBox](#)
Helper class to track and draw bounding boxes on screen.

5.39 PSE_2018_Gruppe1/src/test/mocks/Diagnostics.cpp File Reference

```
#include <iostream>
#include <cstdarg>
```

```
#include "Diagnostics.hpp"
Include dependency graph for Diagnostics.cpp:
```



Variables

- static DiagnosticLevel `maxLevel` = `DiagnosticLevel::DIAG_VERBOSE`

5.39.1 Variable Documentation

5.39.1.1 `maxLevel`

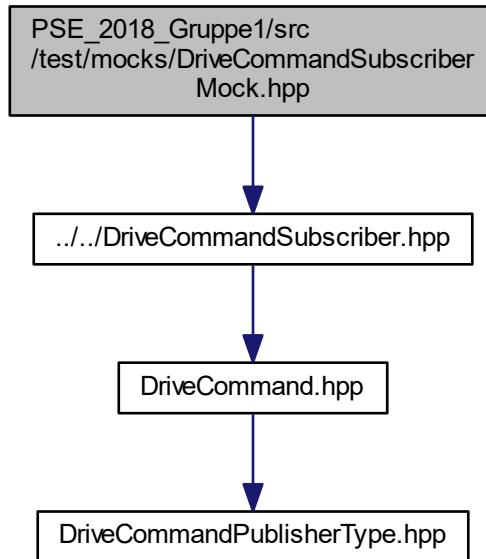
```
DiagnosticLevel maxLevel = DiagnosticLevel::DIAG_VERBOSE [static]
```

Definition at line 9 of file `Diagnostics.cpp`.

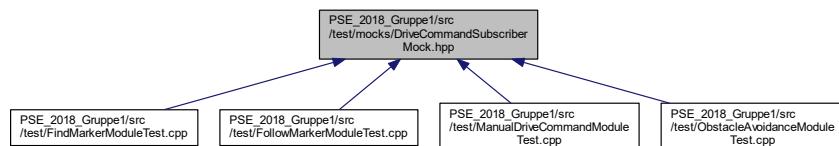
5.40 PSE_2018_Gruppe1/src/test/mock/DriveCommandSubscriberMock.hpp File Reference

```
#include "../../DriveCommandSubscriber.hpp"
```

Include dependency graph for DriveCommandSubscriberMock.hpp:



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

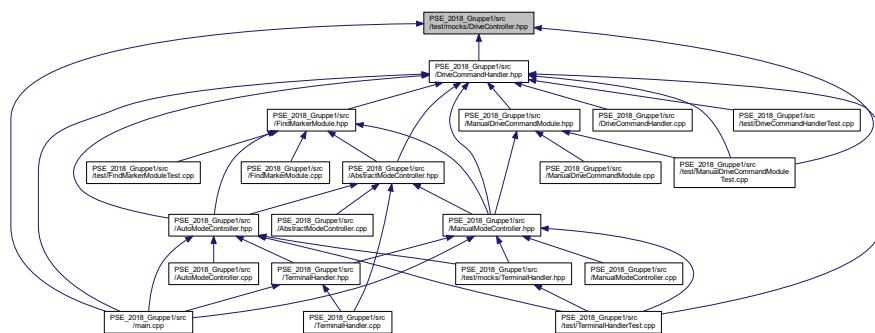


Classes

- class [DriveCommandSubscriberMock](#)

5.41 PSE_2018_Gruppe1/src/test/mocks/DriveController.hpp File Reference

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

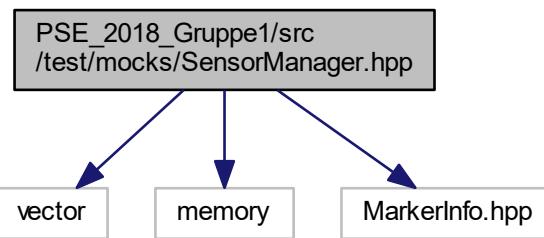


Classes

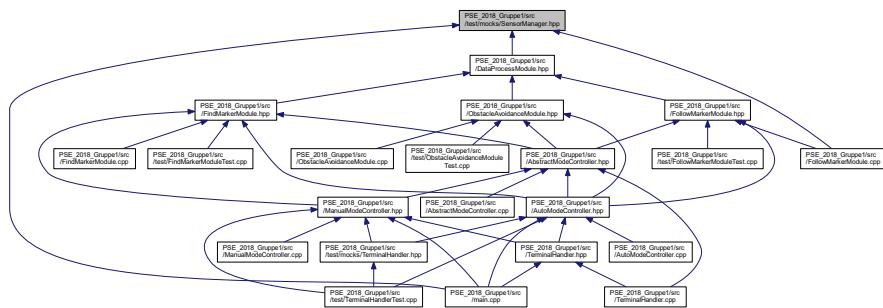
- class [DriveController](#)

5.42 PSE_2018_Gruppe1/src/test/mocks/SensorManager.hpp File Reference

```
#include <vector>
#include <memory>
#include <MarkerInfo.hpp>
Include dependency graph for SensorManager.hpp:
```



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



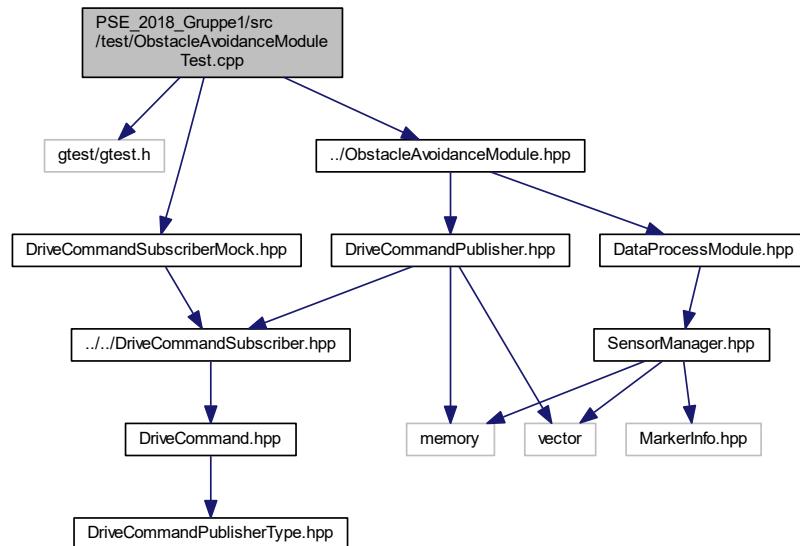
Classes

- class [SensorManager](#)

Sensor and image processing manager.

5.43 PSE_2018_Gruppe1/src/test/ObstacleAvoidanceModuleTest.cpp File Reference

```
#include <gtest/gtest.h>
#include <DriveCommandSubscriberMock.hpp>
#include "../ObstacleAvoidanceModule.hpp"
Include dependency graph for ObstacleAvoidanceModuleTest.cpp:
```



Functions

- [TEST](#) (ObstacleAvoidanceModuleTest, obstacleTest)

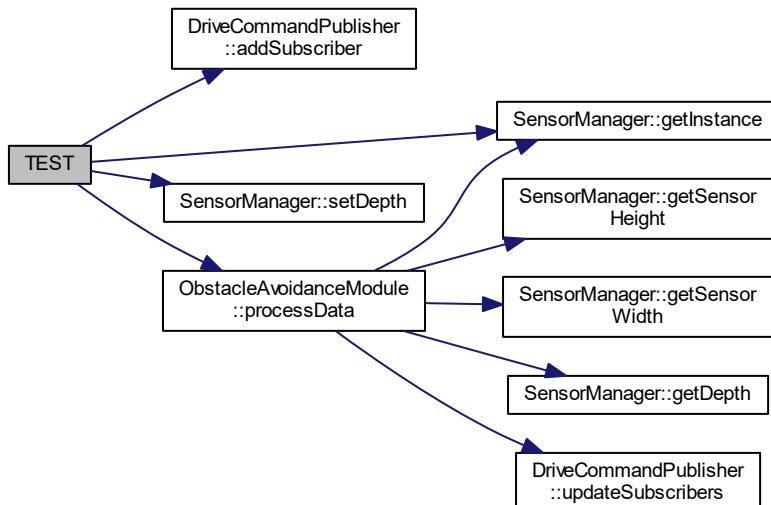
5.43.1 Function Documentation

5.43.1.1 TEST()

```
TEST (
    ObstacleAvoidanceModuleTest ,
    obstacleTest )
```

Definition at line 5 of file ObstacleAvoidanceModuleTest.cpp.

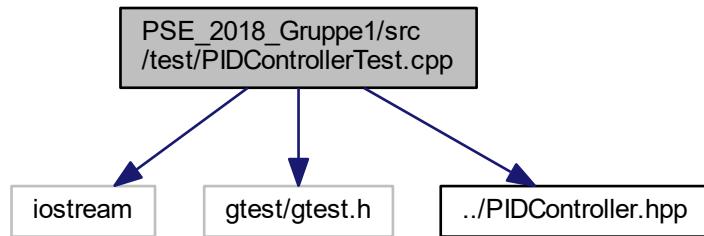
Here is the call graph for this function:



5.44 PSE_2018_Gruppe1/src/test/PIDControllerTest.cpp File Reference

```
#include <iostream>
#include <gtest/gtest.h>
#include "../PIDController.hpp"
```

Include dependency graph for PIDControllerTest.cpp:



Functions

- **TEST** (PIDControllerTest, PID)

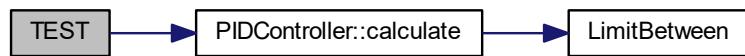
5.44.1 Function Documentation

5.44.1.1 TEST()

```
TEST (
    PIDControllerTest ,
    PID )
```

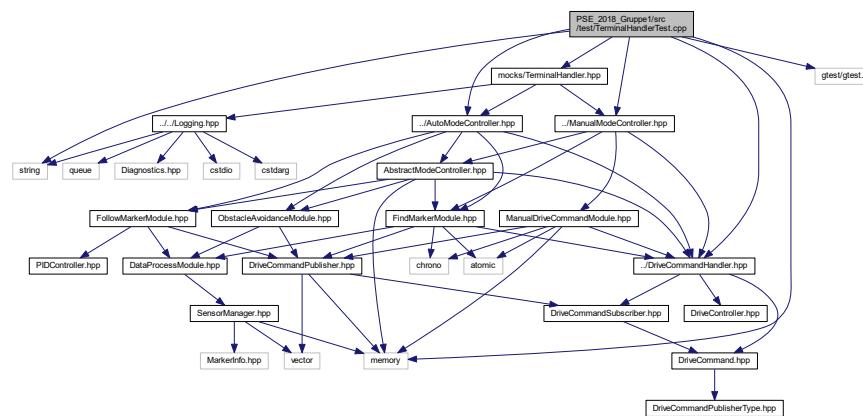
Definition at line 7 of file PIDControllerTest.cpp.

Here is the call graph for this function:



5.45 PSE_2018_Gruppe1/src/test/TerminalHandlerTest.cpp File Reference

```
#include <string>
#include <memory>
#include <gtest/gtest.h>
#include "../DriveCommandHandler.hpp"
#include "../AutoModeController.hpp"
#include "../ManualModeController.hpp"
#include "mocks/TerminalHandler.hpp"
Include dependency graph for TerminalHandlerTest.cpp:
```



Classes

- class [TerminalHandlerTest](#)

Functions

- [TEST_F \(TerminalHandlerTest, SPLITSTRING\)](#)
tests the `TerminalHandler::splitString()` function
- [TEST_F \(TerminalHandlerTest, MODECOMMAND\)](#)
test parsing of the mode-command - handled by `TerminalHandler::handleModeCommand()`
- [TEST_F \(TerminalHandlerTest, SET_L_COMMAND\)](#)
test parsing of the set command if the first parameter is "-l" - handled by `TerminalHandler::HandleSetCommand()` sets log level so only values between 0 and 4 are allowed
- [TEST_F \(TerminalHandlerTest, SET_T_COMMAND\)](#)
test parsing of the set command if the first parameter is "-t" - handled by `TerminalHandler::HandleSetCommand()` sets timeout, so only positive values are allowed
- [TEST_F \(TerminalHandlerTest, SET_D_COMMAND\)](#)
test parsing of the set command if the first parameter is "-d" - handled by `TerminalHandler::HandleSetCommand()` sets distance to vehicle in front so only positive values are allowed
- [TEST_F \(TerminalHandlerTest, DRIVE_COMMAND_NUM_PARAMS\)](#)
tests whether `TerminalHandler::handleDriveCommand()` correctly checks for the number of supplied arguments
- [TEST_F \(TerminalHandlerTest, DRIVE_COMMAND_PARAM_SANITY\)](#)
Tests whether `TerminalHandler::handleDriveCommand()` accepts non-numerical input for its three arguments.
- [TEST_F \(TerminalHandlerTest, DRIVE_COMMAND_TIME\)](#)
Test whether drive command times are handled correctly in `TerminalHandler::handleDriveCommand()`

- [TEST_F](#) ([TerminalHandlerTest](#), [DRIVE_COMMAND_VELO](#))
Tests whether [TerminalHandler::handleDriveCommand\(\)](#) handles various velocity values correctly.
- [TEST_F](#) ([TerminalHandlerTest](#), [DRIVE_COMMAND_STEER](#))
Tests whether [TerminalHandler::handleDriveCommand\(\)](#) handles various steering values correctly.
- [TEST_F](#) ([TerminalHandlerTest](#), [DISABLED_BLOCKING_DONT_ENABLE](#))

5.45.1 Function Documentation

5.45.1.1 TEST_F() [1/11]

```
TEST_F (
```

```
    TerminalHandlerTest ,
```

```
    SPLITSTRING )
```

tests the [TerminalHandler::splitString\(\)](#) function

Definition at line 33 of file TerminalHandlerTest.cpp.

5.45.1.2 TEST_F() [2/11]

```
TEST_F (
```

```
    TerminalHandlerTest ,
```

```
    MODECOMMAND )
```

test parsing of the mode-command - handled by [TerminalHandler::handleModeCommand\(\)](#)

Definition at line 67 of file TerminalHandlerTest.cpp.

5.45.1.3 TEST_F() [3/11]

```
TEST_F (
```

```
    TerminalHandlerTest ,
```

```
    SET_L_COMMAND )
```

test parsing of the set command if the first parameter is "-l" - handled by [TerminalHandler::HandleSetCommand\(\)](#)
sets log level so only values between 0 and 4 are allowed

Definition at line 92 of file TerminalHandlerTest.cpp.

5.45.1.4 TEST_F() [4/11]

```
TEST_F (   
    TerminalHandlerTest ,  
    SET_T_COMMAND )
```

test parsing of the set command if the first parameter is "-t" - handled by TerminalHandler::HandleSetCommand()
sets timeout, so only positive values are allowed

Definition at line 123 of file TerminalHandlerTest.cpp.

5.45.1.5 TEST_F() [5/11]

```
TEST_F (   
    TerminalHandlerTest ,  
    SET_D_COMMAND )
```

test parsing of the set command if the first parameter is "-d" - handled by TerminalHandler::HandleSetCommand()
sets distance to vehicle in front so only positive values are allowed

Definition at line 154 of file TerminalHandlerTest.cpp.

5.45.1.6 TEST_F() [6/11]

```
TEST_F (   
    TerminalHandlerTest ,  
    DRIVE_COMMAND_NUM_PARAMS )
```

tests whether TerminalHandler::handleDriveCommand() correctly checks for the number of supplied arguments

Definition at line 184 of file TerminalHandlerTest.cpp.

5.45.1.7 TEST_F() [7/11]

```
TEST_F (   
    TerminalHandlerTest ,  
    DRIVE_COMMAND_PARAM_SANITY )
```

Tests whether TerminalHandler::handleDriveCommand() accepts non-numerical input for its three arguments.

Definition at line 204 of file TerminalHandlerTest.cpp.

5.45.1.8 TEST_F() [8/11]

```
TEST_F (
    TerminalHandlerTest ,
    DRIVE_COMMAND_TIME )
```

Test whether drive command times are handled correctly in [TerminalHandler::handleDriveCommand\(\)](#)

Definition at line 217 of file TerminalHandlerTest.cpp.

5.45.1.9 TEST_F() [9/11]

```
TEST_F (
    TerminalHandlerTest ,
    DRIVE_COMMAND_VELO )
```

Tests whether [TerminalHandler::handleDriveCommand\(\)](#) handles various velocity values correctly.

Definition at line 255 of file TerminalHandlerTest.cpp.

5.45.1.10 TEST_F() [10/11]

```
TEST_F (
    TerminalHandlerTest ,
    DRIVE_COMMAND_STEER )
```

Tests whether [TerminalHandler::handleDriveCommand\(\)](#) handles various steering values correctly.

Definition at line 291 of file TerminalHandlerTest.cpp.

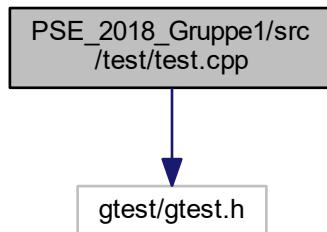
5.45.1.11 TEST_F() [11/11]

```
TEST_F (
    TerminalHandlerTest ,
    DISABLED_BLOCKING_DONT_ENABLE )
```

Definition at line 329 of file TerminalHandlerTest.cpp.

5.46 PSE_2018_Gruppe1/src/test/test.cpp File Reference

```
#include <gtest/gtest.h>
Include dependency graph for test.cpp:
```



Functions

- [TEST](#) (EmptyTest, TrueAndFalse)
- int [main](#) (int argc, char **argv)

5.46.1 Function Documentation

5.46.1.1 main()

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

Definition at line 10 of file test.cpp.

5.46.1.2 TEST()

```
TEST (
    EmptyTest ,
    TrueAndFalse )
```

Definition at line 3 of file test.cpp.

Index

~AbstractModeController
 AbstractModeController, 8

~AutoModeController
 AutoModeController, 11

~DriveCommandHandler
 DriveCommandHandler, 27

~DriveCommandPublisher
 DriveCommandPublisher, 31

~DriveCommandSubscriber
 DriveCommandSubscriber, 34

~DriveController
 DriveController, 39

~FileLogger
 FileLogger, 43

~FindMarkerModule
 FindMarkerModule, 47

~FollowMarkerModule
 FollowMarkerModule, 53

~ManualDriveCommandModule
 ManualDriveCommandModule, 60

~ManualModeController
 ManualModeController, 64

~ObstacleAvoidanceModule
 ObstacleAvoidanceModule, 69

~PIDController
 PIDController, 73

~SensorManager
 SensorManager, 77

~TerminalHandler
 TerminalHandler, 84

abort
 AbstractModeController, 8
 FindMarkerModule, 48
 ManualDriveCommandModule, 60
 ManualModeController, 64
 TerminalHandler, 85

AbstractModeController, 7
 ~AbstractModeController, 8
 abort, 8
 AbstractModeController, 8
 m_driveCommandHandler, 9
 step, 9

addSubscriber
 DriveCommandPublisher, 31

areaLt
 BoundingBox, 16

AutoModeController, 10
 ~AutoModeController, 11
 AutoModeController, 11

 DEFAULT_DISTANCE, 13
 m_findMarkerModule, 13
 m_followMarkerModule, 13
 m_obstacleAvoidanceModule, 13
 setDistance, 11
 step, 12

autoModeController
 TerminalHandlerTest, 100

b
 Color, 22

BoundingBox, 14
 areaLt, 16
 BoundingBox, 15
 getArea, 16
 getHeight, 16
 getRegionID, 17
 getWidth, 17
 isOverlapping, 17
 join, 17
 m_Height, 19
 m_IsEmpty, 19
 m_PenColor, 20
 m_RegionID, 20
 m_Width, 20
 m_X, 20
 m_Y, 20
 setPen, 17
 x, 18
 xLt, 18
 y, 18
 yLt, 19

calculate
 PIDController, 73

CheckLimits
 MathFunctions.hpp, 123

Color, 21
 b, 22
 Color, 21
 g, 22
 m_B, 22
 m_G, 23
 m_R, 23
 r, 22

DEFAULT_DIAGNOSTICS_MAX_LEVEL
 TerminalHandler, 97

DEFAULT_DISTANCE
 AutoModeController, 13

DEFAULT_MAX_DRIVE_DURATION
 ManualModeController, 67

DIAG_DEBUG
 Logging.hpp, 116

DIAG_ERROR
 Logging.hpp, 116

DIAG_INFO
 Logging.hpp, 116

DIAG_VERBOSE
 Logging.hpp, 117

DIAG_WARNING
 Logging.hpp, 117

DataProcessModule, 23
 processData, 24

debugMarkerInfoList
 SensorManager, 81

depth
 SensorManager, 81

Diagnostics.cpp
 maxLevel, 142

driveBlocking
 ManualDriveCommandModule, 60
 ManualModeController, 65

DriveCommand, 24
 DriveCommand, 25
 source, 25
 speed, 25
 steering, 25

DriveCommandHandler, 26
 ~DriveCommandHandler, 27
 DriveCommandHandler, 27
 forceDriveCommand, 27
 m_currentDriveCommand, 29
 trySetDriveCommand, 28
 updateMotor, 28

driveCommandHandler
 TerminalHandlerTest, 100

DriveCommandHandlerTest.cpp
 TEST, 132

DriveCommandPublisher, 30
 ~DriveCommandPublisher, 31
 addSubscriber, 31
 DriveCommandPublisher, 31
 forceSubscribers, 32
 m_driveCommandPublisherType, 33
 m_subscribers, 33
 updateSubscribers, 32

DriveCommandPublisherType
 DriveCommandPublisherType.hpp, 109

DriveCommandPublisherType.hpp
 DriveCommandPublisherType, 109

DriveCommandSubscriber, 34
 ~DriveCommandSubscriber, 34
 DriveCommandSubscriber, 34
 forceDriveCommand, 35
 trySetDriveCommand, 35

DriveCommandSubscriberMock, 35
 DriveCommandSubscriberMock, 37

forceDriveCommand, 37

getLastDriveCommand, 37

lastDriveCommand, 38

trySetDriveCommand, 37

DriveController, 38
 ~DriveController, 39
 getInstance, 39
 getSpeed, 39
 getSteering, 40
 motor_speed, 41
 motor_steering, 41
 setTestMode, 40
 updateController, 40
 updateDirect, 41
 updateDirectMotor, 41

evalUserInput
 TerminalHandler, 85

FileLogger, 42
 ~FileLogger, 43
 FileLogger, 43
 flushLogs, 43
 getInstance, 44
 log, 44
 mLogFile, 45
 m_messageQueue, 45
 operator=, 45

FindMarkerModule, 46
 ~FindMarkerModule, 47
 abort, 48
 FindMarkerModule, 47
 foundMarker, 48
 m_maxDuration, 51
 m_stop, 51
 MINIMUM_CONFIDENT, 51
 processData, 49
 SEARCH_SPEED, 51
 searchBlocking, 49
 setMaxDriveDuration, 50

FindMarkerModuleTest.cpp
 TEST, 133

flushLogs
 FileLogger, 43

FollowMarkerModule, 52
 ~FollowMarkerModule, 53
 FollowMarkerModule, 53
 m_currentMarkerId, 55
 m_distanceController, 55
 m_followsMarker, 55
 m_setDistance, 56
 m_steeringController, 56
 processData, 53
 setDistance, 54
 setMarkerId, 55

FollowMarkerModuleTest.cpp
 TEST, 135–138

forceDriveCommand
 DriveCommandHandler, 27

DriveCommandSubscriber, 35
DriveCommandSubscriberMock, 37
forceSubscribers
 DriveCommandPublisher, 32
foundMarker
 FindMarkerModule, 48

g
 Color, 22
g_appRunning
 main.cpp, 120
g_doStep
 main.cpp, 120
getArea
 BoundingBox, 16
getDepth
 SensorManager, 77
getFps
 SensorManager, 77
getHeight
 BoundingBox, 16
getInstance
 DriveController, 39
 FileLogger, 44
 SensorManager, 78
getLastDriveCommand
 DriveCommandSubscriberMock, 37
getMarkerList
 SensorManager, 78
getRegionID
 BoundingBox, 17
getSensorHeight
 SensorManager, 78
getSensorWidth
 SensorManager, 79
getSpeed
 DriveController, 39
getSteering
 DriveController, 40
getWidth
 BoundingBox, 17

handleDriveCommand
 TerminalHandler, 87
handleModeCommand
 TerminalHandler, 88, 89
handleScriptCommand
 TerminalHandler, 89, 90
handleSearchCommand
 TerminalHandler, 91
handleSetCommand
 TerminalHandler, 92, 93

isInAutoMode
 TerminalHandler, 94
isOverlapping
 BoundingBox, 17
join

BoundingBox, 17
lastDriveCommand
 DriveCommandSubscriberMock, 38
LimitBetween
 MathFunctions.hpp, 125
log
 FileLogger, 44
 Logging, 57
 Logging, 56
 log, 57
 setMaxLevel, 57
 Logging.cpp
 sLogLevel, 115
 Logging.hpp
 DIAG_DEBUG, 116
 DIAG_ERROR, 116
 DIAG_INFO, 116
 DIAG_VERBOSE, 117
 DIAG_WARNING, 117

m_Height
 BoundingBox, 19
m_IsEmpty
 BoundingBox, 19
m_PenColor
 BoundingBox, 20
m_RegionID
 BoundingBox, 20
m_Stop
 ManualDriveCommandModule, 62
m_Width
 BoundingBox, 20
m_abortScript
 TerminalHandler, 97
m_autoModeController
 TerminalHandler, 98
m_B
 Color, 22
m_currentDriveCommand
 DriveCommandHandler, 29
m_currentMarkerId
 FollowMarkerModule, 55
m_d
 PIDController, 74
m_distanceController
 FollowMarkerModule, 55
m_driveCommandHandler
 AbstractModeController, 9
m_driveCommandPublisherType
 DriveCommandPublisher, 33
m_dt
 PIDController, 74
m_error
 PIDController, 74
m_errorDot
 PIDController, 74
m_findMarkerModule
 AutoModeController, 13

ManualModeController, 67
 m_followMarkerModule
 AutoModeController, 13
 m_followsMarker
 FollowMarkerModule, 55
 m_G
 Color, 23
 m_handleInput
 TerminalHandler, 98
 m_i
 PIDController, 75
 m_integralError
 PIDController, 75
 m_isInAutoMode
 TerminalHandler, 98
 m_isInBlockingCall
 ManualModeController, 67
 m_logFile
 FileLogger, 45
 m_manualDriveCommandModule
 ManualModeController, 68
 m_manualModeController
 TerminalHandler, 98
 m_maxDuration
 FindMarkerModule, 51
 ManualDriveCommandModule, 62
 m_maxVal
 PIDController, 75
 m_messageQueue
 FileLogger, 45
 m_minVal
 PIDController, 75
 m_obstacleAvoidanceModule
 AutoModeController, 13
 m_p
 PIDController, 75
 m_R
 Color, 23
 m_setDistance
 FollowMarkerModule, 56
 m_steeringController
 FollowMarkerModule, 56
 m_stop
 FindMarkerModule, 51
 m_subscribers
 DriveCommandPublisher, 33
 m_X
 BoundingBox, 20
 m_Y
 BoundingBox, 20
 MAX_RECUSION_DEPTH
 TerminalHandler, 98
 MIN_DISTANCE
 ObstacleAvoidanceModule, 70
 MINIMUM_CONFIDENT
 FindMarkerModule, 51
 main
 main.cpp, 118
 test.cpp, 152
 main.cpp
 g_appRunning, 120
 g_doStep, 120
 main, 118
 motorStep, 119
 s_flushLogInstant, 120
 ManualDriveCommandModule, 58
 ~ManualDriveCommandModule, 60
 abort, 60
 driveBlocking, 60
 m_Stop, 62
 m_maxDuration, 62
 ManualDriveCommandModule, 59
 setMaxDriveDuration, 61
 ManualDriveCommandModuleTest.cpp
 TEST, 140
 ManualModeController, 62
 ~ManualModeController, 64
 abort, 64
 DEFAULT_MAX_DRIVE_DURATION, 67
 driveBlocking, 65
 m_findMarkerModule, 67
 m_isInBlockingCall, 67
 m_manualDriveCommandModule, 68
 ManualModeController, 64
 searchMarkerBlocking, 65
 setMaxDriveDuration, 66
 step, 67
 manualModeController
 TerminalHandlerTest, 100
 MathFunctions.hpp
 CheckLimits, 123
 LimitBetween, 125
 MathFunctionsTest.cpp
 TEST, 141
 maxLevel
 Diagnostics.cpp, 142
 motor_speed
 DriveController, 41
 motor_steering
 DriveController, 41
 motorStep
 main.cpp, 119
 ObstacleAvoidanceModule, 68
 ~ObstacleAvoidanceModule, 69
 MIN_DISTANCE, 70
 ObstacleAvoidanceModule, 69
 processData, 69
 ObstacleAvoidanceModuleTest.cpp
 TEST, 146
 operator=
 FileLogger, 45
 SensorManager, 79
 PIDController, 71
 ~PIDController, 73
 calculate, 73

m_d, 74
m_dt, 74
m_error, 74
m_errorDot, 74
m_i, 75
m_integralError, 75
m_maxVal, 75
m_minVal, 75
m_p, 75
PIDController, 71
PIDControllerTest.cpp
TEST, 147
PSE_2018_Gruppe1/src/AbstractModeController.cpp,
101
PSE_2018_Gruppe1/src/AbstractModeController.hpp,
101
PSE_2018_Gruppe1/src/AutoModeController.cpp, 102
PSE_2018_Gruppe1/src/AutoModeController.hpp, 103
PSE_2018_Gruppe1/src/DataProcessModule.hpp, 104
PSE_2018_Gruppe1/src/DriveCommand.hpp, 105
PSE_2018_Gruppe1/src/DriveCommandHandler.cpp,
106
PSE_2018_Gruppe1/src/DriveCommandHandler.hpp,
106
PSE_2018_Gruppe1/src/DriveCommandPublisher.cpp,
107
PSE_2018_Gruppe1/src/DriveCommandPublisher.hpp,
108
PSE_2018_Gruppe1/src/DriveCommandPublisher.hpp
Type.hpp, 109
PSE_2018_Gruppe1/src/DriveCommandSubscriber.hpp,
109
PSE_2018_Gruppe1/src/FileLogger.hpp, 111
PSE_2018_Gruppe1/src/FindMarkerModule.cpp, 111
PSE_2018_Gruppe1/src/FindMarkerModule.hpp, 111
PSE_2018_Gruppe1/src/FollowMarkerModule.cpp, 112
PSE_2018_Gruppe1/src/FollowMarkerModule.hpp, 113
PSE_2018_Gruppe1/src/Logging.cpp, 114
PSE_2018_Gruppe1/src/Logging.hpp, 115
PSE_2018_Gruppe1/src/ManualDriveCommand
Module.cpp, 120
PSE_2018_Gruppe1/src/ManualDriveCommand
Module.hpp, 121
PSE_2018_Gruppe1/src/ManualModeController.cpp,
121
PSE_2018_Gruppe1/src/ManualModeController.hpp,
122
PSE_2018_Gruppe1/src/MathFunctions.hpp, 123
PSE_2018_Gruppe1/src/ObstacleAvoidanceModule.hpp,
126
PSE_2018_Gruppe1/src/ObstacleAvoidanceModule.hpp,
126
PSE_2018_Gruppe1/src/PIDController.cpp, 127
PSE_2018_Gruppe1/src/PIDController.hpp, 128
PSE_2018_Gruppe1/src/TerminalHandler.cpp, 128
PSE_2018_Gruppe1/src/TerminalHandler.hpp, 129
PSE_2018_Gruppe1/src/main.cpp, 117
PSE_2018_Gruppe1/src/test/DriveCommandHandler
Test.cpp, 131
PSE_2018_Gruppe1/src/test/FindMarkerModuleTest.hpp,
132
PSE_2018_Gruppe1/src/test/FollowMarkerModule
Test.cpp, 134
PSE_2018_Gruppe1/src/test/ManualDriveCommand
ModuleTest.cpp, 139
PSE_2018_Gruppe1/src/test/MathFunctionsTest.cpp,
140
PSE_2018_Gruppe1/src/test/ObstacleAvoidance
ModuleTest.cpp, 145
PSE_2018_Gruppe1/src/test/PIDControllerTest.cpp,
146
PSE_2018_Gruppe1/src/test/TerminalHandlerTest.cpp,
148
PSE_2018_Gruppe1/src/test/mocks/BoundingBox.hpp,
141
PSE_2018_Gruppe1/src/test/mocks/Diagnostics.hpp,
141
PSE_2018_Gruppe1/src/test/mocks/DriveCommand
SubscriberMock.hpp, 142
PSE_2018_Gruppe1/src/test/mocks/DriveController.hpp,
144
PSE_2018_Gruppe1/src/test/mocks/SensorManager.hpp,
144
PSE_2018_Gruppe1/src/test/mocks/TerminalHandler.hpp,
130
PSE_2018_Gruppe1/src/test/test.cpp, 152
processData
DataProcessModule, 24
FindMarkerModule, 49
FollowMarkerModule, 53
ObstacleAvoidanceModule, 69

r
Color, 22
runOnce
SensorManager, 79

s_flushLogInstant
main.cpp, 120
s_logLevel
Logging.cpp, 115
SEARCH_SPEED
FindMarkerModule, 51
searchBlocking
FindMarkerModule, 49
searchMarkerBlocking
ManualModeController, 65
SensorManager, 76
~SensorManager, 77
debugMarkerInfoList, 81
depth, 81
getDepth, 77
getFps, 77
getInstance, 78
getMarkerList, 78
getSensorHeight, 78
getSensorWidth, 79

operator=, 79
 runOnce, 79
 SensorManager, 77
 setDebugMarkerInfoList, 80
 setDepth, 80
 setDebugMarkerInfoList
 SensorManager, 80
 setDepth
 SensorManager, 80
 setDistance
 AutoModeController, 11
 FollowMarkerModule, 54
 setMarkerId
 FollowMarkerModule, 55
 setMaxDriveDuration
 FindMarkerModule, 50
 ManualDriveCommandModule, 61
 ManualModeController, 66
 setMaxLevel
 Logging, 57
 setPen
 BoundingBox, 17
 setTestMode
 DriveController, 40
 SetUp
 TerminalHandlerTest, 100
 source
 DriveCommand, 25
 speed
 DriveCommand, 25
 splitString
 TerminalHandler, 94, 95
 startInputHandling
 TerminalHandler, 95, 96
 steering
 DriveCommand, 25
 step
 AbstractModeController, 9
 AutoModeController, 12
 ManualModeController, 67
TEST_F
 TerminalHandlerTest.cpp, 149–151
TEST
 DriveCommandHandlerTest.cpp, 132
 FindMarkerModuleTest.cpp, 133
 FollowMarkerModuleTest.cpp, 135–138
 ManualDriveCommandModuleTest.cpp, 140
 MathFunctionsTest.cpp, 141
 ObstacleAvoidanceModuleTest.cpp, 146
 PIDControllerTest.cpp, 147
 test.cpp, 152
 TearDown
 TerminalHandlerTest, 100
 TerminalHandler, 81
 ~TerminalHandler, 84
 abort, 85
 DEFAULT_DIAGNOSTICS_MAX_LEVEL, 97
 evalUserInput, 85
 handleDriveCommand, 87
 handleModeCommand, 88, 89
 handleScriptCommand, 89, 90
 handleSearchCommand, 91
 handleSetCommand, 92, 93
 isInAutoMode, 94
 m_abortScript, 97
 m_autoModeController, 98
 m_handleInput, 98
 m_isInAutoMode, 98
 m_manualModeController, 98
 MAX_RECURSION_DEPTH, 98
 splitString, 94, 95
 startInputHandling, 95, 96
 TerminalHandler, 83, 84
 tryRunSetupScript, 96
 terminalHandler
 TerminalHandlerTest, 100
 TerminalHandlerTest, 99
 autoModeController, 100
 driveCommandHandler, 100
 manualModeController, 100
 SetUp, 100
 TearDown, 100
 terminalHandler, 100
 TerminalHandlerTest.cpp
 TEST_F, 149–151
 test.cpp
 main, 152
 TEST, 152
 tryRunSetupScript
 TerminalHandler, 96
 trySetDriveCommand
 DriveCommandHandler, 28
 DriveCommandSubscriber, 35
 DriveCommandSubscriberMock, 37
 updateController
 DriveController, 40
 updateDirect
 DriveController, 41
 updateDirectMotor
 DriveController, 41
 updateMotor
 DriveCommandHandler, 28
 updateSubscribers
 DriveCommandPublisher, 32
 x
 BoundingBox, 18
 xLt
 BoundingBox, 18
 y
 BoundingBox, 18
 yLt
 BoundingBox, 19