

Bioscara

Generated by Doxygen 1.9.8

1 Documentation	1
1.1 Usage	1
2 README	3
3 README	5
4 README	7
5 README	9
6 Todo List	11
7 Namespace Index	13
7.1 Namespace List	13
8 Hierarchical Index	15
8.1 Class Hierarchy	15
9 Class Index	17
9.1 Class List	17
10 File Index	19
10.1 File List	19
11 Namespace Documentation	21
11.1 bioscara Namespace Reference	21
11.1.1 Function Documentation	21
11.1.1.1 generate_launch_description()	21
11.2 bioscara.hardware_driver Namespace Reference	21
11.2.1 Detailed Description	22
11.2.2 Enumeration Type Documentation	22
11.2.2.1 err_type_t	22
11.2.3 Function Documentation	23
11.2.3.1 error_to_string()	23
11.3 bioscara.hardware_interface Namespace Reference	23
11.3.1 Variable Documentation	23
11.3.1.1 HW_IF_HOME	23
11.4 display Namespace Reference	23
11.4.1 Function Documentation	23
11.4.1.1 generate_launch_description()	23
11.5 gazebo Namespace Reference	24
11.5.1 Function Documentation	24
11.5.1.1 generate_launch_description()	24
11.6 setup Namespace Reference	24
11.6.1 Variable Documentation	24

11.6.1.1 data_files	24
11.6.1.2 description	24
11.6.1.3 entry_points	24
11.6.1.4 install_requires	25
11.6.1.5 license	25
11.6.1.6 maintainer	25
11.6.1.7 maintainer_email	25
11.6.1.8 name	25
11.6.1.9 package_name	25
11.6.1.10 packages	25
11.6.1.11 tests_require	25
11.6.1.12 version	25
11.6.1.13 zip_safe	25
11.7 test_joint_trajectory_controller Namespace Reference	25
11.7.1 Function Documentation	25
11.7.1.1 generate_launch_description()	25
12 Class Documentation	27
12.1 bioscara.hardware_driver::BaseGripper Class Reference	27
12.1.1 Constructor & Destructor Documentation	28
12.1.1.1 BaseGripper()	28
12.1.2 Member Function Documentation	28
12.1.2.1 deinit()	28
12.1.2.2 disable()	28
12.1.2.3 enable()	28
12.1.2.4 init()	29
12.1.2.5 setOffset()	29
12.1.2.6 setPosition()	29
12.1.2.7 setReduction()	29
12.1.2.8 setServoPosition()	30
12.2 bioscara.hardware_driver::BaseJoint Class Reference	30
12.2.1 Detailed Description	32
12.2.2 Member Enumeration Documentation	32
12.2.2.1 stp_reg_t	32
12.2.3 Constructor & Destructor Documentation	33
12.2.3.1 BaseJoint()	33
12.2.3.2 ~BaseJoint()	34
12.2.4 Member Function Documentation	34
12.2.4.1 _home()	34
12.2.4.2 checkOrientation()	34
12.2.4.3 deinit()	35
12.2.4.4 disable()	35

12.2.4.5 disableCL()	35
12.2.4.6 enable()	35
12.2.4.7 enableStallguard()	36
12.2.4.8 getCurrentBCmd()	36
12.2.4.9 getFlags() [1/2]	36
12.2.4.10 getFlags() [2/2]	37
12.2.4.11 getPosition()	37
12.2.4.12 getVelocity()	37
12.2.4.13 home()	38
12.2.4.14 init()	38
12.2.4.15 isBusy()	38
12.2.4.16 isEnabled()	39
12.2.4.17 isHomed()	39
12.2.4.18 isStalled()	39
12.2.4.19 moveSteps()	39
12.2.4.20 postHoming()	40
12.2.4.21 setBrakeMode()	40
12.2.4.22 setDriveCurrent()	40
12.2.4.23 setHoldCurrent()	41
12.2.4.24 setMaxAcceleration()	41
12.2.4.25 setMaxVelocity()	42
12.2.4.26 setPosition()	42
12.2.4.27 setVelocity()	42
12.2.4.28 startHoming()	43
12.2.4.29 stop()	43
12.2.4.30 wait_while_busy()	43
12.2.5 Member Data Documentation	44
12.2.5.1 current_b_cmd	44
12.2.5.2 flags	44
12.2.5.3 name	44
12.3 bioscara_hw_interface::BioscaraHardwareInterface Class Reference	44
12.3.1 Detailed Description	47
12.3.2 Member Function Documentation	47
12.3.2.1 activate_joint()	47
12.3.2.2 deactivate_joint()	48
12.3.2.3 on_activate()	48
12.3.2.4 on_cleanup()	48
12.3.2.5 on_configure()	49
12.3.2.6 on_deactivate()	49
12.3.2.7 on_error()	50
12.3.2.8 on_init()	50
12.3.2.9 on_shutdown()	52

12.3.2.10 perform_command_mode_switch()	52
12.3.2.11 prepare_command_mode_switch()	53
12.3.2.12 read()	53
12.3.2.13 split_interface_string_to_joint_and_name()	54
12.3.2.14 start_homing()	54
12.3.2.15 stop_homing()	55
12.3.2.16 write()	55
12.3.3 Member Data Documentation	56
12.3.3.1 _joint_cfg	56
12.3.3.2 _joint_command_modes	56
12.3.3.3 _joints	57
12.4 bioscara.hardware_driver::Gripper Class Reference	57
12.4.1 Constructor & Destructor Documentation	59
12.4.1.1 Gripper()	59
12.4.2 Member Function Documentation	59
12.4.2.1 disable()	59
12.4.2.2 enable()	59
12.4.2.3 setOffset()	60
12.4.2.4 setPosition()	60
12.4.2.5 setReduction()	60
12.4.2.6 setServoPosition()	60
12.4.3 Member Data Documentation	61
12.4.3.1 freq	61
12.4.3.2 max	61
12.4.3.3 min	61
12.4.3.4 offset	61
12.4.3.5 pwm	61
12.4.3.6 reduction	61
12.5 bioscara.hardware_driver::Joint Class Reference	62
12.5.1 Detailed Description	65
12.5.2 Constructor & Destructor Documentation	65
12.5.2.1 Joint()	65
12.5.2.2 ~Joint()	66
12.5.3 Member Function Documentation	66
12.5.3.1 _home()	66
12.5.3.2 checkCom()	67
12.5.3.3 checkOrientation()	67
12.5.3.4 deinit()	67
12.5.3.5 disableCL()	68
12.5.3.6 enable()	68
12.5.3.7 enableStallguard()	68
12.5.3.8 getFlags()	69

12.5.3.9 getHomingOffset()	69
12.5.3.10 getPosition()	69
12.5.3.11 getVelocity()	70
12.5.3.12 init()	70
12.5.3.13 moveSteps()	70
12.5.3.14 postHoming()	71
12.5.3.15 read()	71
12.5.3.16 setBrakeMode()	72
12.5.3.17 setDriveCurrent()	72
12.5.3.18 setHoldCurrent()	73
12.5.3.19 setHomingOffset()	73
12.5.3.20 setMaxAcceleration()	73
12.5.3.21 setMaxVelocity()	74
12.5.3.22 setPosition()	74
12.5.3.23 setVelocity()	74
12.5.3.24 stop()	75
12.5.3.25 write()	75
12.5.4 Member Data Documentation	76
12.5.4.1 address	76
12.5.4.2 handle	76
12.5.4.3 max	76
12.5.4.4 min	76
12.5.4.5 offset	76
12.5.4.6 reduction	76
12.6 bioscara_hw_interface::BioscaraHardwareInterface::joint_config_t Struct Reference	77
12.6.1 Detailed Description	77
12.6.2 Member Data Documentation	77
12.6.2.1 drive_current	77
12.6.2.2 hold_current	78
12.6.2.3 homing	78
12.6.2.4 i2c_address	78
12.6.2.5 max	78
12.6.2.6 max_acceleration	78
12.6.2.7 max_velocity	78
12.6.2.8 min	78
12.6.2.9 reduction	78
12.6.2.10 stall_threshold	78
12.7 bioscara_hw_interface::BioscaraHardwareInterface::joint_homing_config_t Struct Reference	79
12.7.1 Detailed Description	79
12.7.2 Member Data Documentation	79
12.7.2.1 acceleration	79
12.7.2.2 current	79

12.7.2.3 speed	79
12.7.2.4 threshold	79
12.8 Lowpass Class Reference	80
12.8.1 Constructor & Destructor Documentation	80
12.8.1.1 Lowpass()	80
12.8.2 Member Function Documentation	80
12.8.2.1 resetState()	80
12.8.2.2 updateState()	80
12.8.3 Member Data Documentation	81
12.8.3.1 K	81
12.8.3.2 tau	81
12.8.3.3 Ts	81
12.8.3.4 x	81
12.9 bioscara.hardware_driver::MockGripper Class Reference	81
12.9.1 Constructor & Destructor Documentation	82
12.9.1.1 MockGripper()	82
12.10 bioscara.hardware_driver::MockJoint Class Reference	83
12.10.1 Constructor & Destructor Documentation	86
12.10.1.1 MockJoint()	86
12.10.2 Member Function Documentation	86
12.10.2.1 _home()	86
12.10.2.2 checkOrientation()	87
12.10.2.3 disable()	87
12.10.2.4 enable()	87
12.10.2.5 getDeltaT()	87
12.10.2.6 getFlags()	88
12.10.2.7 getPosition()	88
12.10.2.8 getVelocity()	88
12.10.2.9 isHomed()	89
12.10.2.10 setPosition()	89
12.10.2.11 setVelocity()	89
12.10.2.12 stop()	90
12.10.3 Member Data Documentation	90
12.10.3.1 async_start_time	90
12.10.3.2 last_set_position	90
12.10.3.3 last_set_velocity	90
12.10.3.4 op_mode	90
12.10.3.5 q	91
12.10.3.6 qd	91
12.11 MovMax Class Reference	91
12.11.1 Constructor & Destructor Documentation	91
12.11.1.1 MovMax()	91

12.11.2 Member Function Documentation	91
12.11.2.1 updateState()	91
12.11.3 Member Data Documentation	92
12.11.3.1 cb_data	92
12.11.3.2 cb_index	92
12.11.3.3 M	92
12.12 RPI_PWM Class Reference	92
12.12.1 Detailed Description	93
12.12.2 Constructor & Destructor Documentation	93
12.12.2.1 ~RPI_PWM()	93
12.12.3 Member Function Documentation	93
12.12.3.1 disable()	93
12.12.3.2 enable()	93
12.12.3.3 setDutyCycle()	93
12.12.3.4 setDutyCycleNS()	93
12.12.3.5 setPeriod()	93
12.12.3.6 start()	94
12.12.3.7 stop()	94
12.12.3.8 writeSYS()	94
12.12.4 Member Data Documentation	94
12.12.4.1 chippath	94
12.12.4.2 per	94
12.12.4.3 pwmpath	94
13 File Documentation	95
13.1 Arduino/joint/configuration.h File Reference	95
13.1.1 Detailed Description	96
13.1.2 Macro Definition Documentation	96
13.1.2.1 ADR	96
13.1.2.2 MAXACCEL	96
13.1.2.3 MAXVEL	96
13.2 configuration.h	97
13.3 Arduino/joint/filters.h File Reference	97
13.3.1 Detailed Description	98
13.4 filters.h	99
13.5 Arduino/joint/joint.h File Reference	99
13.5.1 Detailed Description	101
13.5.2 Macro Definition Documentation	101
13.5.2.1 ACK	101
13.5.2.2 DUMP_BUFFER	101
13.5.2.3 MAX_BUFFER	102
13.5.2.4 NACK	102

13.5.2.5 RFLAGS_SIZE	102
13.5.3 Enumeration Type Documentation	102
13.5.3.1 stp_reg_t	102
13.5.4 Function Documentation	103
13.5.4.1 readValue()	103
13.5.4.2 writeValue()	103
13.6 joint.h	104
13.7 Arduino/joint/joint.ino File Reference	105
13.7.1 Detailed Description	106
13.7.2 Macro Definition Documentation	106
13.7.2.1 J3	106
13.7.3 Function Documentation	107
13.7.3.1 blocking_handler()	107
13.7.3.2 loop()	107
13.7.3.3 non_blocking_handler()	107
13.7.3.4 receiveEvent()	108
13.7.3.5 requestEvent()	108
13.7.3.6 setup()	108
13.7.4 Variable Documentation	109
13.7.4.1 reg	109
13.7.4.2 rx_buf	109
13.7.4.3 rx_data_ready	109
13.7.4.4 rx_length	109
13.7.4.5 stepper	109
13.7.4.6 tx_buf	109
13.7.4.7 tx_length	109
13.8 Arduino/joint/stall.h File Reference	110
13.8.1 Detailed Description	110
13.8.2 Function Documentation	111
13.8.2.1 stall_threshold()	111
13.9 stall.h	111
13.10 docs/DOCS_README.md File Reference	111
13.11 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscaraBringup/launch/bioscara.launch.py File Reference	111
13.12 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscaraBringup/launch/test_joint_trajectory_controller.launch.py File Reference	112
13.13 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscaraBringup/README.md File Reference	112
13.14 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscaraControllers/README.md File Reference	112
13.15 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscaraDescription/README.md File Reference	112
13.16 ROS2/ros2_scara_ws/src/dalsa_bioscara/README.md File Reference	112
13.17 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscaraDescription/bioscaraDescription/__init__.py File Reference	112

13.18 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/launch/display.launch.py File Reference	112
13.19 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/launch/gazebo.launch.py File Reference	113
13.20 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/setup.py File Reference	113
13.21 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/common.h File Reference	113
13.21.1 Detailed Description	114
13.21.2 Macro Definition Documentation	115
13.21.2.1 DUMP_BUFFER	115
13.22 common.h	115
13.23 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mBaseGripper.h File Reference	115
13.23.1 Detailed Description	117
13.24 mBaseGripper.h	117
13.25 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mBaseJoint.h File Reference	118
13.25.1 Detailed Description	119
13.26 mBaseJoint.h	119
13.27 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mGripper.h File Reference	121
13.27.1 Detailed Description	122
13.28 mGripper.h	122
13.29 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mJoint.h File Reference	123
13.29.1 Detailed Description	124
13.30 mJoint.h	124
13.31 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mJoint.hpp File Reference	126
13.31.1 Detailed Description	127
13.32 mJoint.hpp	128
13.33 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mMockGripper.h File Reference	129
13.33.1 Detailed Description	130
13.34 mMockGripper.h	131
13.35 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mMockJoint.h File Reference	131
13.35.1 Detailed Description	132
13.36 mMockJoint.h	133
13.37 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/uErr.h File Reference	134
13.37.1 Macro Definition Documentation	135
13.37.1.1 RETURN_ON_ERROR	135
13.37.1.2 RETURN_ON_FALSE	135
13.37.1.3 RETURN_ON_NEGATIVE	136

13.38 uErr.h	136
13.39 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/uI2C.h File Reference	137
13.39.1 Detailed Description	139
13.39.2 Macro Definition Documentation	139
13.39.2.1 ACK	139
13.39.2.2 MAX_BUFFER	139
13.39.2.3 NACK	139
13.39.2.4 RFLAGS_SIZE	140
13.39.3 Function Documentation	140
13.39.3.1 closeI2CDevHandle()	140
13.39.3.2 openI2CDevHandle()	140
13.39.3.3 readFromI2CDev()	140
13.39.3.4 writeToI2CDev()	141
13.40 uI2C.h	141
13.41 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/uPWM.h File Reference	142
13.41.1 Detailed Description	143
13.42 uPWM.h	144
13.43 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/uTransmission.h File Reference	145
13.43.1 Macro Definition Documentation	146
13.43.1.1 ACTUATOR2JOINT	146
13.43.1.2 DEG2RAD	146
13.43.1.3 JOINT2ACTUATOR	146
13.43.1.4 M_PI	147
13.43.1.5 RAD2DEG	147
13.44 uTransmission.h	147
13.45 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/joint_comm_node.cpp File Reference	147
13.45.1 Function Documentation	148
13.45.1.1 INT_handler()	148
13.45.1.2 main()	148
13.45.2 Variable Documentation	148
13.45.2.1 J1	148
13.45.2.2 J2	148
13.45.2.3 J3	149
13.45.2.4 J4	149
13.46 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mBaseGripper.cpp File Reference	149
13.47 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mBaseJoint.cpp File Reference	150
13.48 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mGripper.cpp File Reference	150
13.49 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mJoint.cpp File Reference	151

13.50 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockGripper.cpp File Reference	152
13.51 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockJoint.cpp File Reference	152
13.52 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/uErr.cpp File Reference	153
13.53 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/i2C.cpp File Reference	154
13.53.1 Function Documentation	155
13.53.1.1 closeI2CDevHandle()	155
13.53.1.2 openI2CDevHandle()	155
13.53.1.3 readFromI2CDev()	156
13.53.1.4 writeToI2CDev()	156
13.54 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include/bioscara_hardware_interface/bioscara_hardware.hpp File Reference	157
13.55 bioscara_hardware.hpp	158
13.56 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/src/bioscara_hardware.cpp File Reference	160
Index	161

Chapter 1

Documentation

This documentation currently documents how the robot controller communicates with the joint controllers, this includes:

- The joint firmware in the [/Arduino](#) directory
- The interfacing library used for communicating with the joints in the [/ROS2](#) directory.

1.1 Usage

the joint_communication library is structured as a ROS2 package but can also be used in another build toolchain. If that is the case ensure the include paths are still correct.

Chapter 2

README

This package contains all launch and config files for the robot to work.

Chapter 3

README

This package contains all custom controllers used for the bioscara robot.

Chapter 4

README

All configuration parameters are stored in the config/bioscara_parameters file.

Chapter 5

README

The packages are structured according to this guide: [RTW Package Structure](#)

When compiling the package is installed in the share/ directory. Also the URDF is stored there. The [bioscara.launch.py](#) file expects to find the urdf there. This is done in the packages cmake file

```
install(
  DIRECTORY hardware/include/
  DESTINATION include/ros2_control_demo_example_1
)
install(
  DIRECTORY description/launch_description/ros2_control_description/urdf
  DESTINATION share/ros2_control_demo_example_1
)
install(
  DIRECTORY bringup/launch bringup/config
  DESTINATION share/ros2_control_demo_example_1
)
install(TARGETS ros2_control_demo_example_1
  EXPORT export_ros2_control_demo_example_1
  ARCHIVE DESTINATION lib
  LIBRARY DESTINATION lib
  RUNTIME DESTINATION bin
)
```

TODO:

- [] Format and rework this content

Chapter 6

Todo List

Member `bioscara.hardware_driver::Joint::read` (const stp_reg_t reg, T &data, u_int8_t &flags) .

Implement a return code for read only functions

- Implement clearStall function

Member `bioscara.hardware_interface::BioscaraHardwareInterface::on_init` (const hardware_interface::HardwareComponentInterfaceParams ¶ms) override

threshold and current are uint8_t, if a number larger outside $0 < n < 255$ is passed as a parameters it will overflow.

Chapter 7

Namespace Index

7.1 Namespace List

Here is a list of all namespaces with brief descriptions:

bioscara	21
bioscara_hw	
Generic BaseGripper object to interact with the robot gripper	21
bioscara_hw_interface	23
display	23
gazebo	24
setup	24
test_joint_trajectory_controller	25

Chapter 8

Hierarchical Index

8.1 Class Hierarchy

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

bioscara.hardware_driver::BaseGripper	27
bioscara.hardware_driver::Gripper	57
bioscara.hardware_driver::MockGripper	81
bioscara.hardware_driver::BaseJoint	30
bioscara.hardware_driver::Joint	62
bioscara.hardware_driver::MockJoint	83
bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t	77
bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing_config_t	79
Lowpass	80
MovMax	91
RPI_PWM	92
hardware_interface::SystemInterface	
bioscara.hardware_interface::BioscaraHardwareInterface	44

Chapter 9

Class Index

9.1 Class List

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

<code>bioscara_hardware_driver::BaseGripper</code>	27
<code>bioscara_hardware_driver::BaseJoint</code>	
TODO	30
<code>bioscara_hardware_interface::BioscaraHardwareInterface</code>	
The bioscara hardware interface class	44
<code>bioscara_hardware_driver::Gripper</code>	57
<code>bioscara_hardware_driver::Joint</code>	
Representing a single joint on the I2C bus	62
<code>bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t</code>	
Configuration structure holding the passed paramters from the ros2_control urdf	77
<code>bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing_config_t</code>	
Configuration structure holding the passed homing paramters from the ros2_control urdf	79
<code>Lowpass</code>	80
<code>bioscara_hardware_driver::MockGripper</code>	81
<code>bioscara_hardware_driver::MockJoint</code>	83
<code>MovMax</code>	91
<code>RPI_PWM</code>	
PWM class for the Raspberry PI 4 and 5	92

Chapter 10

File Index

10.1 File List

Here is a list of all files with brief descriptions:

Arduino/joint/ configuration.h	Configuration definitions for Joint 1 to Joint 4	95
Arduino/joint/ filters.h	Helper classes for FIR and IIR filters	97
Arduino/joint/ joint.h	Joint firmware header	99
Arduino/joint/ joint.ino	Joint firmware	105
Arduino/joint/ stall.h	Helper functions for improved stall detection	110
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Bringup/launch/ bioscara.launch.py		111
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Bringup/launch/ test_joint_trajectory_controller.launch.py		112
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Description/ setup.py		113
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Description/bioscara Description/ __init__.py		112
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Description/launch/ display.launch.py		112
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Description/launch/ gazebo.launch.py		113
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Hardware Driver/include/bioscara Hardware Driver/ common.h	A file containing utility macros and functions	113
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Hardware Driver/include/bioscara Hardware Driver/ mBaseGripper.h	File containing the BaseGripper class	115
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Hardware Driver/include/bioscara Hardware Driver/ mBaseJoint.h	File including the BaseJoint class	118
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Hardware Driver/include/bioscara Hardware Driver/ mGripper.h	File containing the Gripper class	121
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Hardware Driver/include/bioscara Hardware Driver/ mJoint.h	File including the Joint class	123
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara Hardware Driver/include/bioscara Hardware Driver/ mJoint.hpp	Templated functions for the Joint class	126

ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_↔ driver/mMockGripper.h	129
File containing the MockGripper class	
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_↔ driver/mMockJoint.h	131
File including the MockJoint class	
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_↔ driver/uErr.h	134
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_↔ driver/uI2C.h	137
Low level utility for I2C communication on Raspberry Pi using Igpio library	
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_↔ driver/uPWM.h	142
Includes source code for Hardware PWM generation on Raspberry Pi 4	
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_↔ driver/uTransmission.h	145
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/joint_comm_node.cpp	147
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mBaseGripper.cpp	149
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mBaseJoint.cpp	150
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mGripper.cpp	150
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mJoint.cpp	151
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockGripper.cpp	152
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockJoint.cpp	152
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/uErr.cpp	153
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/uI2C.cpp	154
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include/bioscara_hardware_↔ interface/bioscara_hardware.hpp	157
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/src/bioscara_hardware.cpp	160

Chapter 11

Namespace Documentation

11.1 bioscara Namespace Reference

Functions

- [generate_launch_description \(\)](#)

11.1.1 Function Documentation

11.1.1.1 [generate_launch_description\(\)](#)

```
bioscara.generate_launch_description ( )
```

11.2 bioscara.hardware_driver Namespace Reference

Generic [BaseGripper](#) object to interact with the robot gripper.

Classes

- class [BaseGripper](#)
- class [BaseJoint](#)

TODO.
- class [Gripper](#)
- class [Joint](#)

Representing a single joint on the I2C bus.
- class [MockGripper](#)
- class [MockJoint](#)

Enumerations

- enum class [err_type_t](#) {
 OK = 0 , [ERROR](#) = -1 , [NOT_HOMED](#) = -2 , [NOT_ENABLED](#) = -3 ,
 [STALLED](#) = -4 , [NOT_INIT](#) = -5 , [COMM_ERROR](#) = -6 , [INVALID_ARGUMENT](#) = -101 ,
 [INCORRECT_STATE](#) = -109 }
- Enum defining common error types.*

Functions

- std::string `error_to_string (err_type_t err)`
Converts an error code to a string and returns it.

11.2.1 Detailed Description

Generic `BaseGripper` object to interact with the robot gripper.

Derviced class from the `BaseGripper` class to interact with the hardware gripper.

This class is a wrapper function to interact with the robot gripper either through a `MockGripper` or the hardware `Gripper`.

An example application is shown below. Note that depending on the build toolchain the include path can differ. This example assumes the `bioscara_hardware_driver` package is built with ROS2.

```
// #include "bioscara_hardware_driver/mGripper.h"
#include "bioscara_hardware_driver/mMockGripper.h"
int main(int argc, char **argv)
{
    MockGripper gripper;
    gripper.init();
    if(gripper.enable() != 0){
        cerr << "Failed to engage gripper" << endl;
        return -1;
    }

    if (gripper.setPosition(40) != 0)
    {
        cerr << "setting position failed" << endl;
        return -1;
    }

    if(gripper.disable() != 0){
        cerr << "Failed to disengage gripper" << endl;
        return -1;
    }

    gripper.deinit();
    return 0;
}
```

This class is a wrapper function to interact with a PWM servo gripper.

11.2.2 Enumeration Type Documentation

11.2.2.1 `err_type_t`

```
enum class bioscara_hardware_driver::err_type_t [strong]
```

Enum defining common error types.

Enumerator

OK	
ERROR	
NOT_HOMED	
NOT_ENABLED	
STALLED	
NOT_INIT	
COMM_ERROR	
INVALID_ARGUMENT	
INCORRECT_STATE	

11.2.3 Function Documentation

11.2.3.1 error_to_string()

```
std::string bioscara.hardware_driver::error_to_string (
```

<code>err_type_t</code>	<code>err</code>
-------------------------	------------------

```
)
```

Converts an error code to a string and returns it.

Parameters

<code>err</code>	
------------------	--

Returns

`std::string`

11.3 bioscara.hardware_interface Namespace Reference

Classes

- class [BioscaraHardwareInterface](#)

The bioscara hardware interface class.

Variables

- `constexpr char HW_IF_HOME [] = "home"`

11.3.1 Variable Documentation

11.3.1.1 HW_IF_HOME

```
constexpr char bioscara.hardware_interface::HW_IF_HOME [ ] = "home" [constexpr]
```

11.4 display Namespace Reference

Functions

- [generate_launch_description \(\)](#)

11.4.1 Function Documentation

11.4.1.1 generate_launch_description()

```
display.generate_launch_description ( )
```

11.5 gazebo Namespace Reference

Functions

- [generate_launch_description \(\)](#)

11.5.1 Function Documentation

11.5.1.1 [generate_launch_description\(\)](#)

```
gazebo.generate_launch_description ( )
```

11.6 setup Namespace Reference

Variables

- str [package_name](#) = 'bioscara_description'
- [name](#)
- [version](#)
- [packages](#)
- [data_files](#)
- [install_requires](#)
- [zip_safe](#)
- [maintainer](#)
- [maintainer_email](#)
- [description](#)
- [license](#)
- [tests_require](#)
- [entry_points](#)

11.6.1 Variable Documentation

11.6.1.1 [data_files](#)

```
setup.data_files
```

11.6.1.2 [description](#)

```
setup.description
```

11.6.1.3 [entry_points](#)

```
setup.entry_points
```

11.6.1.4 install_requires

```
setup.install_requires
```

11.6.1.5 license

```
setup.license
```

11.6.1.6 maintainer

```
setup.maintainer
```

11.6.1.7 maintainer_email

```
setup.maintainer_email
```

11.6.1.8 name

```
setup.name
```

11.6.1.9 package_name

```
str setup.package_name = 'bioscara_description'
```

11.6.1.10 packages

```
setup.packages
```

11.6.1.11 tests_require

```
setup.tests_require
```

11.6.1.12 version

```
setup.version
```

11.6.1.13 zip_safe

```
setup.zip_safe
```

11.7 test_joint_trajectory_controller Namespace Reference

Functions

- [generate_launch_description \(\)](#)

11.7.1 Function Documentation

11.7.1.1 generate_launch_description()

```
test_joint_trajectory_controller.generate_launch_description ( )
```

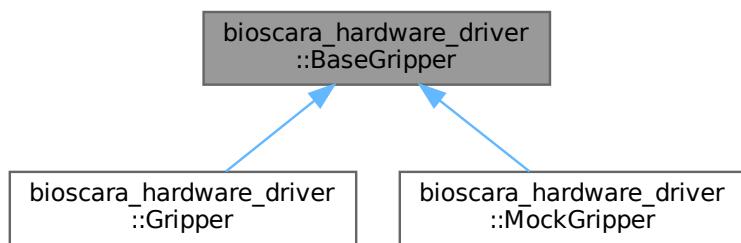

Chapter 12

Class Documentation

12.1 bioscara.hardware_driver::BaseGripper Class Reference

```
#include <mBaseGripper.h>
```

Inheritance diagram for bioscara.hardware_driver::BaseGripper:



Public Member Functions

- **BaseGripper** (void)
- virtual **err_type_t init** (void)
Placeholder, does nothing.
- virtual **err_type_t deinit** (void)
Placeholder, does nothing.
- virtual **err_type_t enable** (void)
Prepares the servo for use.
- virtual **err_type_t disable** (void)
Disables the servo.
- virtual **err_type_t setPosition** (float width)
Sets the gripper width in m from the closed position.
- virtual **err_type_t setServoPosition** (float angle)
Sets the servo position of the gripper actuator in degrees.
- virtual void **setReduction** (float reduction)
Manually set reduction.
- virtual void **setOffset** (float offset)
Manually set offset.

12.1.1 Constructor & Destructor Documentation

12.1.1.1 BaseGripper()

```
bioscara.hardware_driver::BaseGripper::BaseGripper (
    void )
```

12.1.2 Member Function Documentation

12.1.2.1 deinit()

```
err_type_t bioscara.hardware_driver::BaseGripper::deinit (
    void ) [virtual]
```

Placeholder, does nothing.

Returns

0

12.1.2.2 disable()

```
err_type_t bioscara.hardware_driver::BaseGripper::disable (
    void ) [virtual]
```

Disables the servo.

Returns

non-zero error code.

Reimplemented in [bioscara.hardware_driver::Gripper](#).

12.1.2.3 enable()

```
err_type_t bioscara.hardware_driver::BaseGripper::enable (
    void ) [virtual]
```

Prepares the servo for use.

Returns

non-zero error code.

Reimplemented in [bioscara.hardware_driver::Gripper](#).

12.1.2.4 init()

```
err_type_t bioscara.hardware_driver::BaseGripper::init (
    void ) [virtual]
```

Placeholder, does nothing.

Returns

0

12.1.2.5 setOffset()

```
void bioscara.hardware_driver::BaseGripper::setOffset (
    float offset ) [virtual]
```

Manually set offset.

Reimplemented in [bioscara.hardware_driver::Gripper](#).

12.1.2.6 setPosition()

```
err_type_t bioscara.hardware_driver::BaseGripper::setPosition (
    float width ) [virtual]
```

Sets the gripper width in m from the closed position.

Arguments outside the allowed range are bounded to limit min and max.

Parameters

<i>width</i>	width in m.
--------------	-------------

Reimplemented in [bioscara.hardware_driver::Gripper](#).

12.1.2.7 setReduction()

```
void bioscara.hardware_driver::BaseGripper::setReduction (
    float reduction ) [virtual]
```

Manually set reduction.

Parameters

<i>reduction</i>	
------------------	--

Reimplemented in [bioscara.hardware_driver::Gripper](#).

12.1.2.8 setServoPosition()

```
err_type_t bioscara.hardware_driver::BaseGripper::setServoPosition (
    float angle ) [virtual]
```

Sets the servo position of the gripper actuator in degrees.

Parameters

<i>angle</i>	in degrees.
--------------	-------------

Reimplemented in [bioscara.hardware_driver::Gripper](#).

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

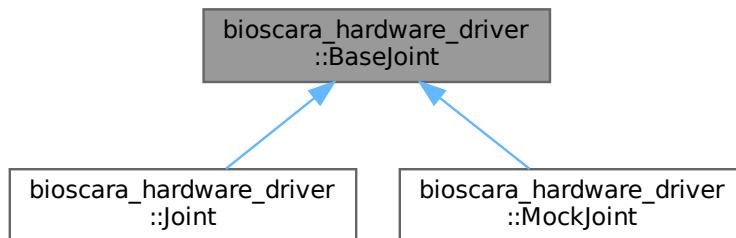
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/include/bioscara.hardware_driver/mBaseGripper.h
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/src/mBaseGripper.cpp

12.2 bioscara.hardware_driver::BaseJoint Class Reference

TODO.

```
#include <mBaseJoint.h>
```

Inheritance diagram for bioscara.hardware_driver::BaseJoint:



Public Types

- enum `stp_reg_t` {
 `NONE` = 0x00, `PING` = 0x0f, `SETUP` = 0x10, `SETRPM` = 0x11,
 `GETDRIVERRPM` = 0x12, `MOVESTEPS` = 0x13, `MOVEANGLE` = 0x14, `MOVETOANGLE` = 0x15,
 `GETMOTORSTATE` = 0x16, `RUNCOTINOUS` = 0x17, `ANGLEMOVED` = 0x18, `SETCURRENT` = 0x19,
 `SETHOLDCURRENT` = 0x1A, `SETMAXACCELERATION` = 0x1B, `SETMAXDECELERATION` = 0x1C,
 `SETMAXVELOCITY` = 0x1D,
 `ENABLESTALLGUARD` = 0x1E, `DISABLESTALLGUARD` = 0x1F, `CLEARSTALL` = 0x20, `SETBRAKEMODE` = 0x22,
 `ENABLEPID` = 0x23, `DISABLEPID` = 0x24, `ENABLECLOSEDLOOP` = 0x25, `DISABLECLOSEDLOOP` = 0x26,
 `SETCONTROLTHRESHOLD` = 0x27, `MOVETOEND` = 0x28, `STOP` = 0x29, `GETPIDERROR` = 0x2A,
 `CHECKORIENTATION` = 0x2B, `GETENCODERRPM` = 0x2C, `HOME` = 0x2D, `HOMEOFFSET` = 0x2E
 }

register and command definitions

Public Member Functions

- **BaseJoint** (const std::string name)
Create a Joint object.
- **~BaseJoint** (void)
- **virtual err_type_t init** (void)
Initialization, derived classes may override this.
- **virtual err_type_t deinit** (void)
Deinitialization, derived classes may override this.
- **virtual err_type_t enable** (u_int8_t driveCurrent, u_int8_t holdCurrent)
Setup the joint and engages motor, derived classes may override this.
- **virtual err_type_t disable** (void)
disengages the joint motors, derived classes may override this.
- **virtual err_type_t home** (float velocity, u_int8_t sensitivity, u_int8_t current)
Blocking implementation to home the joint, derived classes may override this.
- **virtual err_type_t startHoming** (float velocity, u_int8_t sensitivity, u_int8_t current)
non-blocking implementation to home the joint, derived classes may override this.
- **virtual err_type_t postHoming** (void)
perform tasks after a non-blocking homing, derived classes may override this.
- **virtual err_type_t getPosition** (float &pos)=0
get the current joint position in radians or m for cylindrical and prismatic joints respectively. Derived class must override this.
- **virtual err_type_t setPosition** (float pos)
get the current joint position in radians or m for cylindrical and prismatic joints respectively. Derived class may override this.
- **virtual err_type_t moveSteps** (int32_t steps)
Move full steps. Derived class may override this.
- **virtual err_type_t getVelocity** (float &vel)=0
get the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class must override this.
- **virtual err_type_t setVelocity** (float vel)
Set the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.
- **virtual err_type_t checkOrientation** (float angle=10.0)
Calls the checkOrientation method of the motor. Checks in which direction the motor is turning. Derived class may override this.
- **virtual err_type_t stop** (void)
Stops the motor. Derived class may override this.
- **virtual err_type_t disableCL** (void)
Disables the Closed-Loop PID Controller Derived class may override this.
- **virtual err_type_t setDriveCurrent** (u_int8_t current)
Set the Drive Current. Derived class may override this.
- **virtual err_type_t setHoldCurrent** (u_int8_t current)
Set the Hold Current. Derived class may override this.
- **virtual err_type_t setBrakeMode** (u_int8_t mode)
Set Brake Mode. Derived class may override this.
- **virtual err_type_t setMaxAcceleration** (float maxAccel)
Set the maximum permitted joint acceleration (and deceleration) in rad/s^2 or m/s^2 for cylindrical and prismatic joints respectively. Derived class may override this.
- **virtual err_type_t setMaxVelocity** (float maxVel)
Set the maximum permitted joint velocity in rad/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.

- virtual `err_type_t enableStallguard (u_int8_t sensitivity)`
Enable encoder stall detection of the joint. Derived class may override this.
- virtual bool `isHomed ()`
Checks the state if the motor is homed.
- virtual bool `isEnabled ()`
Checks the state if the motor is enabled.
- virtual bool `isStalled ()`
Checks if the motor is stalled.
- virtual bool `isBusy ()`
Checks if the joint controller is busy processing a blocking command.
- virtual `err_type_t getFlags (u_int8_t &flags)`
- virtual `err_type_t getFlags ()`
- virtual `stp_reg_t getCurrentBCmd ()`
get the currently active blocking command

Public Attributes

- std::string `name`

Protected Member Functions

- virtual void `wait_while_busy (const float period_ms)`
Blocking loop waiting for BUSY flag to reset.
- virtual `err_type_t _home (float velocity, u_int8_t sensitivity, u_int8_t current)=0`
Call to start the homing sequence of a joint.

Protected Attributes

- u_int8_t `flags` = 0b00001100
State flags transmitted with every I2C transaction.
- `stp_reg_t current_b_cmd` = NONE
Keeps track if a blocking command is being executed.

12.2.1 Detailed Description

TODO.

12.2.2 Member Enumeration Documentation

12.2.2.1 `stp_reg_t`

```
enum bioscara.hardware_driver::BaseJoint::stp_reg_t
```

register and command definitions

a register can be read (R) or written (W), each register has a size in bytes. The payload can be split into multiple values or just be a single value. Note that not all functions are implemented.

Enumerator

NONE	Used for signalling purposes.
PING	R; Size: 1; [(char) ACK].
SETUP	W; Size: 2; [(uint8) holdCurrent, (uint8) driveCurrent].
SETRPM	W; Size: 4; [(float) RPM].
GETDRIVERRPM	
MOVESTEPS	W; Size: 4; [(int32) steps].
MOVEANGLE	
MOVETOANGLE	W; Size: 4; [(float) degrees].
GETMOTORSTATE	
RUNCOTINOUS	
ANGLEMOVED	R; Size: 4; [(float) degrees].
SETCURRENT	W; Size: 1; [(uint8) driveCurrent].
SETHOLDCURRENT	W; Size: 1; [(uint8) holdCurrent].
SETMAXACCELERATION	
SETMAXDECELERATION	
SETMAXVELOCITY	
ENABLESTALLGUARD	W; Size: 1; [(uint8) threshold].
DISABLESTALLGUARD	
CLEARSTALL	
SETBRAKEMODE	W; Size: 1; [(uint8) mode].
ENABLEPID	
DISABLEPID	
ENABLECLOSEDLOOP	
DISABLECLOSEDLOOP	W; Size: 1; [(uint8) 0].
SETCONTROLTHRESHOLD	
MOVETOEND	
STOP	W; Size: 1; [(uint8) mode].
GETPIDERROR	
CHECKORIENTATION	W; Size: 4; [(float) degrees].
GETENCODERRPM	R; Size: 4; [(float) RPM].
HOME	W; Size: 4; [(uint8) current, (int8) sensitivity, (uint8) speed, (uint8) direction].
HOMEOFFSET	R/W; Size: 4; [(float) -].

12.2.3 Constructor & Destructor Documentation

12.2.3.1 BaseJoint()

```
bioscara_hardware_driver::BaseJoint::BaseJoint (
    const std::string name )
```

Create a [Joint](#) object.

The [Joint](#) object represents a single joint.

Parameters

<i>name</i>	string device name for identification
-------------	---------------------------------------

12.2.3.2 ~BaseJoint()

```
bioscara.hardware_driver::BaseJoint::~BaseJoint (
    void )
```

12.2.4 Member Function Documentation

12.2.4.1 _home()

```
virtual err_type_t bioscara.hardware_driver::BaseJoint::_home (
    float velocity,
    u_int8_t sensitivity,
    u_int8_t current ) [protected], [pure virtual]
```

Call to start the homing sequence of a joint.

First the joint will check the motor wiring by executing the checkOrientation internally. Then it will set the specified speed until a resistance which drives the PID error above the specified threshold is encountered. At this point the stepper stops and zeros the encoder.

Parameters

<i>velocity</i>	signed velocity in rad/s or m/s. Must be between $1.0 < \text{RAD2DEG}(\text{JOINT2ACTUATOR}(\text{velocity}, \text{reduction}, 0)) / 6 < 250.0$
<i>sensitivity</i>	Encoder pid error threshold 0 to 255.
<i>current</i>	homing current, determines how easy it is to stop the motor and thereby provoke a stall

Returns

0 on success, -1 on communication error, -3 when the motor is not enabled, -5 if the joint is not initialized, -101 if the velocity is zero, -102 if absolute value of the velocity is outside the specified limits.

Implemented in [bioscara.hardware_driver::Joint](#), and [bioscara.hardware_driver::MockJoint](#).

12.2.4.2 checkOrientation()

```
err_type_t bioscara.hardware_driver::BaseJoint::checkOrientation (
    float angle = 10.0 ) [virtual]
```

Calls the checkOrientation method of the motor. Checks in which direction the motor is turning. Derived class may override this.

As the orientation check is blocking on the motor, this function returns when the isBusy flag is clear again.

Parameters

<i>angle</i>	degrees how much the motor should turn. A few degrees is sufficient.
--------------	--

Returns

0 on success, -1 on communication error, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented in [bioscara.hardware_driver::Joint](#), and [bioscara.hardware_driver::MockJoint](#).

12.2.4.3 deinit()

```
err_type_t bioscara.hardware_driver::BaseJoint::deinit (
    void ) [virtual]
```

Deinitialization, derived classes may override this.

Reimplemented in [bioscara.hardware_driver::Joint](#).

12.2.4.4 disable()

```
err_type_t bioscara.hardware_driver::BaseJoint::disable (
    void ) [virtual]
```

disengages the joint motors, derived classes may override this.

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented in [bioscara.hardware_driver::MockJoint](#).

12.2.4.5 disableCL()

```
err_type_t bioscara.hardware_driver::BaseJoint::disableCL (
    void ) [virtual]
```

Disables the Closed-Loop PID Controller Derived class may override this.

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented in [bioscara.hardware_driver::Joint](#).

12.2.4.6 enable()

```
err_type_t bioscara.hardware_driver::BaseJoint::enable (
    u_int8_t driveCurrent,
    u_int8_t holdCurrent ) [virtual]
```

Setup the joint and engages motor, derived classes may override this.

Reimplemented in [bioscara.hardware_driver::Joint](#), and [bioscara.hardware_driver::MockJoint](#).

12.2.4.7 enableStallguard()

```
err_type_t bioscara.hardware_driver::BaseJoint::enableStallguard (
    u_int8_t sensitivity) [virtual]
```

Enable encoder stall detection of the joint. Derived class may override this.

If the PID error exceeds the set threshold a stall is triggered and the motor disabled. A detected stall can be reset by homing or reenabling the joint using [enable\(\)](#).

Note

If stall detection shall be enabled, invoke this method AFTER enabling the joint with [enable\(\)](#).

Parameters

<i>sensitivity</i>	value of threshold. 0 - 255 where lower is more sensitive.
--------------------	--

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented in [bioscara.hardware_driver::Joint](#).

12.2.4.8 getCurrentBCmd()

```
BaseJoint::stp_reg_t bioscara.hardware_driver::BaseJoint::getCurrentBCmd (
    void) [virtual]
```

get the currently active blocking command

Returns

The the command of type `stp_reg_t`

12.2.4.9 getFlags() [1/2]

```
err_type_t bioscara.hardware_driver::BaseJoint::getFlags (
    u_int8_t & flags) [virtual]
```

ping the joint to get the latest driver state flags

Parameters

<i>flags</i>	if succesfull, populated with the latest flags
--------------	--

Returns

0 on success, -5 if the joint is not initialized.

12.2.4.10 getFlags() [2/2]

```
err_type_t bioscara_hardware_driver::BaseJoint::getFlags (
    void ) [virtual]
```

Overload of [BaseJoint::getFlags\(u_int8_t &flags\)](#)

Returns

0 on success, -5 if the joint is not initialized.

Reimplemented in [bioscara_hardware_driver::Joint](#), and [bioscara_hardware_driver::MockJoint](#).

12.2.4.11 getPosition()

```
virtual err_type_t bioscara_hardware_driver::BaseJoint::getPosition (
    float & pos ) [pure virtual]
```

get the current joint position in radians or m for cylindrical and prismatic joints respectively. Derived class must override this.

Warning

If the joint is not homed this method does not return an error. Instead `pos` will be 0.0.

Parameters

<i>pos</i>	
------------	--

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Implemented in [bioscara_hardware_driver::Joint](#), and [bioscara_hardware_driver::MockJoint](#).

12.2.4.12 getVelocity()

```
virtual err_type_t bioscara_hardware_driver::BaseJoint::getVelocity (
    float & vel ) [pure virtual]
```

get the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class must override this.

Parameters

<i>vel</i>	
------------	--

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Implemented in [bioscara_hardware_driver::Joint](#), and [bioscara_hardware_driver::MockJoint](#).

12.2.4.13 home()

```
err_type_t bioscara_hardware_driver::BaseJoint::home (
    float velocity,
    u_int8_t sensitivity,
    u_int8_t current ) [virtual]
```

Blocking implementation to home the joint, derived classes may override this.

A blocking implementation which only returns after the the joint is no longer BUSY. See [Joint::_home\(\)](#) for documentation.

Additionally this method returns:

Returns

-2 when not homed succesfull (isHomed flag still not set), -109 if the joint is already currently homing (for example from a call to [Joint::startHoming\(\)](#)).

12.2.4.14 init()

```
err_type_t bioscara_hardware_driver::BaseJoint::init (
    void ) [virtual]
```

Initialization, derived classes may override this.

Reimplemented in [bioscara_hardware_driver::Joint](#).

12.2.4.15 isBusy()

```
bool bioscara_hardware_driver::BaseJoint::isBusy (
    void ) [virtual]
```

Checks if the joint controller is busy processing a blocking command.

Reads the internal state flags from the last transmission. If an update is neccessary call [Joint::getFlags\(\)](#) before invoking this function.

Returns

true if a blocking command is currently executing, false if not.

12.2.4.16 isEnabled()

```
bool bioscara.hardware_driver::BaseJoint::isEnabled (
    void ) [virtual]
```

Checks the state if the motor is enabled.

Reads the internal state flags from the last transmission. If an update is neccessary call [Joint::getFlags\(\)](#) before invoking this function. If the motor actually can move depends on the state of the STALLED flag which can be checked using [Joint::isStalled\(\)](#).

Returns

true if the motor is enabled, false if not.

12.2.4.17 isHomed()

```
bool bioscara.hardware_driver::BaseJoint::isHomed (
    void ) [virtual]
```

Checks the state if the motor is homed.

Reads the internal state flags from the last transmission. If an update is neccessary call [Joint::getFlags\(\)](#) before invoking this function.

Returns

true if the motor is homed, false if not.

Reimplemented in [bioscara.hardware_driver::MockJoint](#).

12.2.4.18 isStalled()

```
bool bioscara.hardware_driver::BaseJoint::isStalled (
    void ) [virtual]
```

Checks if the motor is stalled.

Reads the internal state flags from the last transmission. If an update is neccessary call [Joint::getFlags\(\)](#) before invoking this function.

Returns

true if the motor is stalled, false if not.

12.2.4.19 moveSteps()

```
err_type_t bioscara.hardware_driver::BaseJoint::moveSteps (
    int32_t steps ) [virtual]
```

Move full steps. Derived class may override this.

This function can be called even when not homed.

Parameters

<code>steps</code>	number of full steps
--------------------	----------------------

Returns

0 on success, -1 on communication error, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented in [bioscara_hw_driver::Joint](#).

12.2.4.20 postHoming()

```
err_type_t bioscara_hw_driver::BaseJoint::postHoming (
    void ) [virtual]
```

perform tasks after a non-blocking homing, derived classes may override this.

This method resets the current_b_cmd to NONE, checks if the joint is homed, and saves the homing offset to the joint.

Returns

0 on success, -109 if the current_b_cmd is not HOME, -1 on communication error, -2 when not homed, -5 if the joint is not initialized.

Reimplemented in [bioscara_hw_driver::Joint](#).

12.2.4.21 setBrakeMode()

```
err_type_t bioscara_hw_driver::BaseJoint::setBrakeMode (
    u_int8_t mode ) [virtual]
```

Set Brake Mode. Derived class may override this.

Parameters

<code>mode</code>	Freewheel: 0, Coolbrake: 1, Hardbrake: 2
-------------------	--

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented in [bioscara_hw_driver::Joint](#).

12.2.4.22 setDriveCurrent()

```
err_type_t bioscara_hw_driver::BaseJoint::setDriveCurrent (
    u_int8_t current ) [virtual]
```

Set the Drive Current. Derived class may override this.

Warning

This function is unreliable and not well tested. Use [Joint::enable\(\)](#) instead!

Parameters

<i>current</i>	0% - 100% of driver current
----------------	-----------------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented in [bioscara_hardware_driver::Joint](#).

12.2.4.23 setHoldCurrent()

```
err_type_t bioscara_hardware_driver::BaseJoint::setHoldCurrent (
    u_int8_t current ) [virtual]
```

Set the Hold Current. Derived class may override this.

Warning

This function is unreliable and not well tested. Use [Joint::enable\(\)](#) instead!

Parameters

<i>current</i>	0% - 100% of driver current
----------------	-----------------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented in [bioscara_hardware_driver::Joint](#).

12.2.4.24 setMaxAcceleration()

```
err_type_t bioscara_hardware_driver::BaseJoint::setMaxAcceleration (
    float maxAccel ) [virtual]
```

Set the maximum permitted joint acceleration (and deceleration) in rad/s^2 or m/s^2 for cylindrical and prismatic joints respectively. Derived class may override this.

Parameters

<i>maxAccel</i>	maximum joint acceleration.
-----------------	-----------------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented in [bioscara_hw_driver::Joint](#).

12.2.4.25 setMaxVelocity()

```
err_type_t bioscara_hw_driver::BaseJoint::setMaxVelocity (
    float maxVel ) [virtual]
```

Set the maximum permitted joint velocity in rad/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.

Parameters

<i>maxVel</i>	maximum joint velocity.
---------------	-------------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented in [bioscara_hw_driver::Joint](#).

12.2.4.26 setPosition()

```
err_type_t bioscara_hw_driver::BaseJoint::setPosition (
    float pos ) [virtual]
```

get the current joint position in radians or m for cylindrical and prismatic joints respectively. Derived class may override this.

Parameters

<i>pos</i>	in rad or m
------------	-------------

Returns

0 on success, -1 on communication error, -2 when not homed, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented in [bioscara_hw_driver::Joint](#), and [bioscara_hw_driver::MockJoint](#).

12.2.4.27 setVelocity()

```
err_type_t bioscara_hw_driver::BaseJoint::setVelocity (
    float vel ) [virtual]
```

Set the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.

Parameters

<i>vel</i>	
------------	--

Returns

0 on success, -1 on communication error, -2 when not homed, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented in [bioscara.hardware_driver::Joint](#), and [bioscara.hardware_driver::MockJoint](#).

12.2.4.28 startHoming()

```
err_type_t bioscara.hardware_driver::BaseJoint::startHoming (
    float velocity,
    u_int8_t sensitivity,
    u_int8_t current ) [virtual]
```

non-blocking implementation to home the joint, derived classes may override this.

See [Joint::_home\(\)](#) for documentation. The current_b_cmd flag is set to HOME This method returns immediately after starting the homing sequence. This should be used when the blocking implementation is not acceptable. For example in the update loop of the [bioscara.hardware_interface::BioscaraHardwareInterface::write\(\)](#).

Additionally this method returns:

Returns

-109 if the joint is already currently homing (for example from a call to [Joint::startHoming\(\)](#)).

12.2.4.29 stop()

```
err_type_t bioscara.hardware_driver::BaseJoint::stop (
    void ) [virtual]
```

Stops the motor. Derived class may override this.

Stops the motor by setting the maximum velocity to zero and the position setpoint to the current position

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented in [bioscara.hardware_driver::Joint](#), and [bioscara.hardware_driver::MockJoint](#).

12.2.4.30 wait_while_busy()

```
void bioscara.hardware_driver::BaseJoint::wait_while_busy (
    const float period_ms ) [protected], [virtual]
```

Blocking loop waiting for BUSY flag to reset.

Parameters

<i>period_ms</i>	time in ms between polls.
------------------	---------------------------

12.2.5 Member Data Documentation**12.2.5.1 current_b_cmd**

```
stp_reg_t bioscara.hardware_driver::BaseJoint::current_b_cmd = NONE [protected]
```

Keeps track if a blocking command is being executed.

12.2.5.2 flags

```
u_int8_t bioscara.hardware_driver::BaseJoint::flags = 0b00001100 [protected]
```

State flags transmitted with every I2C transaction.

The transmission flags purpose are to transmit the joints current state. Note: They can not be used as error indication of the execution of a transmitted write command, since commands are executed after the I2C transaction is completed. The status flags are one byte with following structure:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
reserved	reserved	reserved	reserved	NOTENABLED	NOTHOMED	BUSY	STALL

STALL is set if a stall from the stall detection is sensed and the joint is stopped. The flag is cleared when the joint is homed or the Stallguard enabled.

BUSY is set if the slave is busy processing a previous command.

NOTHOMED is cleared if the joint is homed. Movement is only allowed if this flag is clear

NOTENABLED is cleared if the joint is enabled after calling [Joint::enable\(\)](#)

12.2.5.3 name

```
std::string bioscara.hardware_driver::BaseJoint::name
```

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

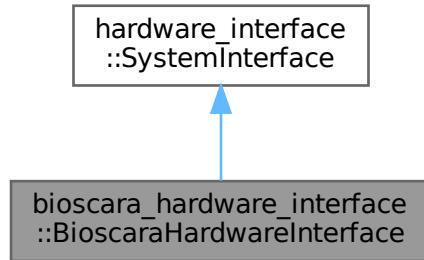
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/include/bioscara.hardware_driver/[mBaseJoint.h](#)
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/src/[mBaseJoint.cpp](#)

12.3 bioscara.hardware_interface::BioscaraHardwareInterface Class Reference

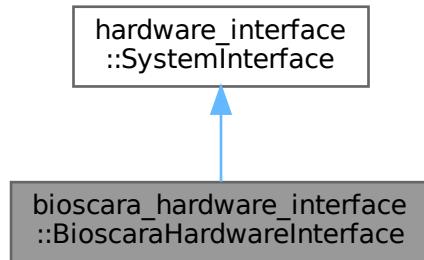
The bioscara hardware interface class.

```
#include <bioscara_hardware.hpp>
```

Inheritance diagram for bioscara_hardware_interface::BioscaraHardwareInterface:



Collaboration diagram for bioscara_hardware_interface::BioscaraHardwareInterface:



Classes

- struct [joint_config_t](#)
configuration structure holding the passed paramters from the ros2_control urdf
- struct [joint_homing_config_t](#)
configuration structure holding the passed homing paramters from the ros2_control urdf

Public Member Functions

- hardware_interface::CallbackReturn [on_init](#) (const hardware_interface::HardwareComponentInterface<
Params ¶ms) override
Called on initialization to the unconfigured state.
- hardware_interface::CallbackReturn [on_shutdown](#) (const rclcpp_lifecycle::State &previous_state) override
Called on the transistion from the inactive, unconfigured and active to the finalized state.

- hardware_interface::CallbackReturn [on_configure](#) (const rclcpp_lifecycle::State &previous_state) override
Called on the transition from the unconfigured to the inactive state.
- hardware_interface::CallbackReturn [on_cleanup](#) (const rclcpp_lifecycle::State &previous_state) override
Called on the transition from the inactive to the unconfigured state.
- hardware_interface::CallbackReturn [on_activate](#) (const rclcpp_lifecycle::State &previous_state) override
Called on the transition from the inactive to the active state.
- hardware_interface::CallbackReturn [on_deactivate](#) (const rclcpp_lifecycle::State &previous_state) override
Called on the transition from the active to the inactive state.
- hardware_interface::return_type [read](#) (const rclcpp::Time &time, const rclcpp::Duration &period) override
Reads from the hardware and populates the state interfaces.
- hardware_interface::return_type [write](#) (const rclcpp::Time &time, const rclcpp::Duration &period) override
Writes commands to the hardware from the command interfaces.
- hardware_interface::return_type [prepare_command_mode_switch](#) (const std::vector< std::string > &start_interfaces, const std::vector< std::string > &stop_interfaces) override
Performs checks and book keeping of the active control mode when changing controllers.
- hardware_interface::return_type [perform_command_mode_switch](#) (const std::vector< std::string > &start_interfaces, const std::vector< std::string > &stop_interfaces) override
Perform the mode-switching for the new command interface combination.
- hardware_interface::CallbackReturn [on_error](#) (const rclcpp_lifecycle::State &previous_state) override
Called when an error in any state or state transition is thrown.

Private Member Functions

- [bioscara.hardware_driver::err_type_t start_homing](#) (const std::string name, float velocity)
wrapper method to start homing.
- [bioscara.hardware_driver::err_type_t stop_homing](#) (const std::string name)
wrapper method to stop homing.
- void [split_interface_string_to_joint_and_name](#) (std::string interface, std::string &joint_name, std::string &interface_name)
Split a interface string like "<joint_name>/<interface_name>" to "<joint_name>" and "<interface_name>".
- [bioscara.hardware_driver::err_type_t activate_joint](#) (const std::string name)
Enables each joint, enables the stall detection and sets the maximum acceleration.
- [bioscara.hardware_driver::err_type_t deactivate_joint](#) (const std::string name)
Disables each joint.

Private Attributes

- std::unordered_map< std::string, std::unique_ptr< bioscara.hardware_driver::BaseJoint > > [_joints](#)
unordered map storing the pointers to BaseJoint objects. This will either be a MockJoint or Joint.
- std::unordered_map< std::string, [joint_config_t](#) > [_joint_cfg](#)
unordered map storing the configuration struct of the joints.
- std::unordered_map< std::string, std::set< std::string > > [_joint_command_modes](#)
unordered map of sets storing the active command interfaces for each joint.

12.3.1 Detailed Description

The bioscara hardware interface class.

The hardware interface serves to wrap custom hardware interaction in the standardized ros2_control architecture.

Hardware Lifecycle

The hardware follows the ros2_control hardware interface lifecycle which intern is following the [ROS2 managed node lifecycle](#).

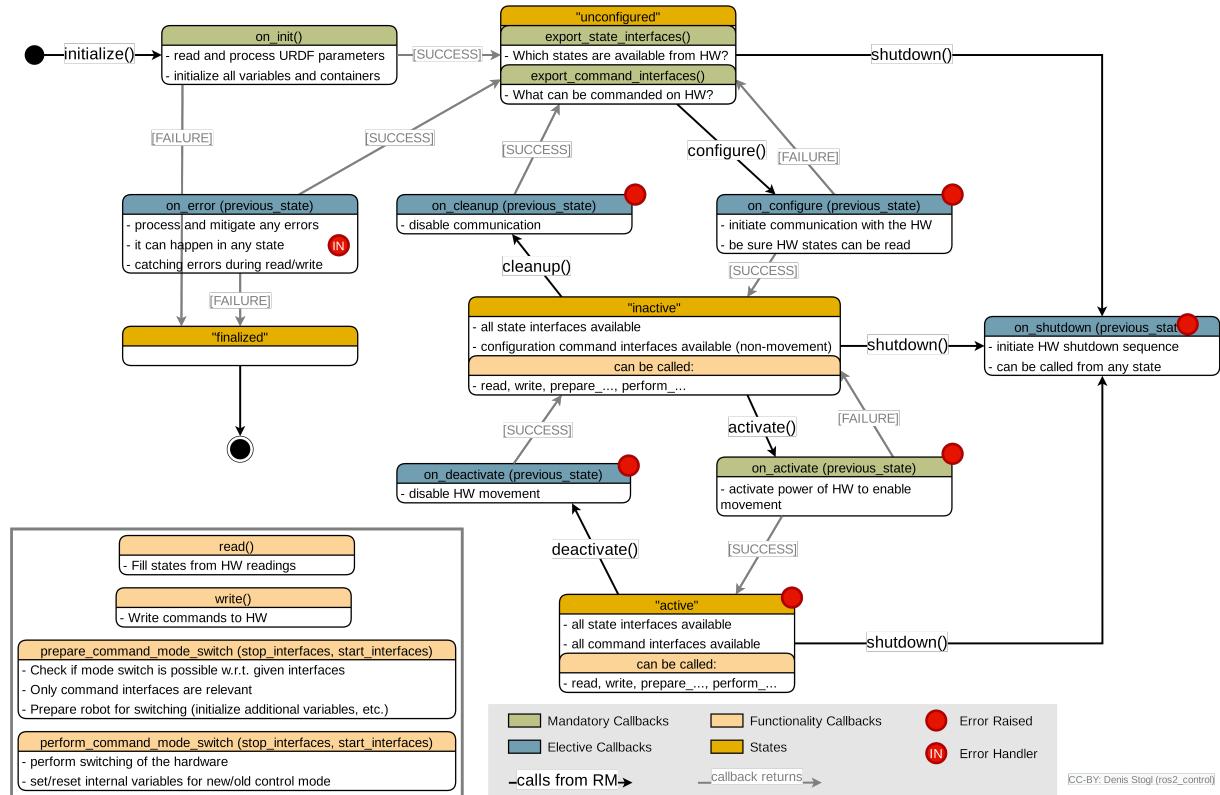


Figure 12.1 Hardware interface lifecycle

12.3.2 Member Function Documentation

12.3.2.1 activate_joint()

```
bioscara.hardware_driver::err_type_t bioscara.hardware_interface::BioscaraHardwareInterface::activate_joint (
    const std::string name ) [private]
```

Enables each joint, enables the stall detection and sets the maximum acceleration.

Parameters

<code>name</code>	joint name to enable
-------------------	----------------------

Returns

<code>bioscara_hardware_driver::err_type_t</code>	
---	--

12.3.2.2 deactivate_joint()

```
bioscara_hardware_driver::err_type_t bioscara_hardware_interface::BioscaraHardwareInterface<-->
::deactivate_joint (
    const std::string name ) [private]
```

Disables each joint.

Parameters

<code>name</code>	joint name to disable
-------------------	-----------------------

Returns

<code>bioscara_hardware_driver::err_type_t</code>	
---	--

12.3.2.3 on_activate()

```
hardware_interface::CallbackReturn bioscara_hardware_interface::BioscaraHardwareInterface<-->
::on_activate (
    const rclcpp_lifecycle::State & previous_state ) [override]
```

Called on the transition from the inactive to the active state.

Calls [activate_joint\(\)](#) to enable the joints.

It is allowed to activate the hardware even if it is not homed. To home the joint the homing_controller must be activated, but generally a hardware component must be active in order for controllers to become active.

To prohibit movement on activation the set point for each position command interface is set equal to the current measured position, and the velocity command is set to 0.0 for each command interface. The current values are obtained by calling the [read\(\)](#) method once which populates the state interfaces with values.

Parameters

<code>previous_state</code>	
-----------------------------	--

Returns

<code>hardware_interface::CallbackReturn</code>	
---	--

Below a workaround to force a read cycle of all joints to get initial values for the state interfaces. These will be copied to the command interface to prevent movement at startup.

12.3.2.4 on_cleanup()

```
hardware_interface::CallbackReturn bioscara_hardware_interface::BioscaraHardwareInterface<-->
::on_cleanup (
    const rclcpp_lifecycle::State & previous_state ) [override]
```

Called on the transition from the `inactive` to the `unconfigured` state.

Disconnect from the joints.

Parameters

<code>previous_state</code>	<input type="button" value=""/>
-----------------------------	---------------------------------

Returns

`hardware_interface::CallbackReturn`

Disconnect from the joints and throw error if it fails

12.3.2.5 `on_configure()`

```
hardware_interface::CallbackReturn bioscara.hardware_interface::BioscaraHardwareInterface::on_configure (
    const rclcpp_lifecycle::State & previous_state ) [override]
```

Called on the transition from the `unconfigured` to the `inactive` state.

Establish and test connection to each joint.

Parameters

<code>previous_state</code>	<input type="button" value=""/>
-----------------------------	---------------------------------

Returns

`hardware_interface::CallbackReturn`

12.3.2.6 `on_deactivate()`

```
hardware_interface::CallbackReturn bioscara.hardware_interface::BioscaraHardwareInterface::on_deactivate (
    const rclcpp_lifecycle::State & previous_state ) [override]
```

Called on the transition from the `active` to the `inactive` state.

Disables all joints and thereby allows backdriving. State interfaces continue to be updated.

Parameters

<code>previous_state</code>	<input type="button" value=""/>
-----------------------------	---------------------------------

Returns

```
hardware_interface::CallbackReturn
```

disable the joints and throw error if it fails

12.3.2.7 on_error()

```
hardware_interface::CallbackReturn bioscara.hardware_interface::BioscaraHardwareInterface<-->
::on_error (
    const rclcpp_lifecycle::State & previous_state ) [override]
```

Called when an error in any state or state transition is thrown.

According to the [ros2_control documentation](#):

Error handling follows the node lifecycle. If successful CallbackReturn::SUCCESS is returned and hardware is again in UNCONFIGURED state, if any ERROR or FAILURE happens the hardware ends in FINALIZED state and can not be recovered. The only option is to reload the complete plugin, but there is currently no service for this in the Controller Manager.

Since the hardware will immediately return to the unconfigured state ([source](#)) if the error could be handled we manually call the transition functions which would normally be called to this state. Those are:

- **Previous state:** active
 - Deactivate hardware ([on_deactivate\(\)](#)) -> inactive
 - Clean-Up hardware ([on_cleanup\(\)](#)) -> unconfigured
- **Previous state:** inactive
 - Deactivate hardware ([on_deactivate\(\)](#)) -> inactive
 - * call the deactivate function anyway regardless if state was active or inactive. For example if the [on_activate\(\)](#) function fails on Joint::enableStallguard() the joint will have been enabled, to disable it invoke [on_deactivate\(\)](#).
 - Clean-Up hardware ([on_cleanup\(\)](#)) -> unconfigured

In particular the deactivation is important. For example if a joint stalls the [read\(\)](#) or [write\(\)](#) methods throw an error, which will be handled here and allow the hardware to be deactivated, disabling the joints to allow backdriving.

Parameters

<i>previous_state</i>	<input type="button" value=""/>
-----------------------	---------------------------------

Returns

```
hardware_interface::CallbackReturn
```

12.3.2.8 on_init()

```
hardware_interface::CallbackReturn bioscara.hardware_interface::BioscaraHardwareInterface<-->
::on_init (
```

```
const hardware_interface::HardwareComponentInterfaceParams & params ) [override]
```

Called on initialization to the `unconfigured` state.

Performs the following checks on the configures joints parsed from the URDF description:

- Each joint must have the 3 command interfaces (in this order): 'position', 'velocity', 'home'
- Each joint must have the 3 state interfaces (in this order): 'position', 'velocity', 'home'

Stores the configuration parameters for each joint in the `_joint_cfg` map. Each joint must have these parameters:

- `i2c_address` (int, HEX)
- `reduction` (float)
- `min` (float)
- `max` (float)
- `stall_threshold` (int, DEC)
- `hold_current` (int, DEC)
- `drive_current` (int, DEC)
- `max_acceleration` (float)
- `max_velocity` (float)
- `homming`
 - `speed` (float)
 - `threshold` (int, DEC)
 - `current` (int, DEC)
 - `acceleration` (float)

Adds each joint to the internal `_joints` map. Creates a `MockJoint` object if the `use_mock_hardware` parameter is 'True' or 'true', or else a hardware Joint.

Parameters

<code>params</code>	<input type="button" value=""/>
---------------------	---------------------------------

Returns

`hardware_interface::CallbackReturn`

Loop over all joints described in the hardware description file, check if they have the position and velocity command and state interface defined and finally add them to the internal `_joints` list

Todo threshold and current are `uint8_t`, if a number larger outside $0 < n < 255$ is passed as a parameters it will overflow.

12.3.2.9 `on_shutdown()`

```
hardware_interface::CallbackReturn bioscara.hardware_interface::BioscaraHardwareInterface::on_shutdown (
    const rclcpp_lifecycle::State & previous_state ) [override]
```

Called on the transition from the inactive, unconfigured and active to the finalized state.

When transitioning directly from active to finalized `on_deactivate()` is automatically called before [Source Code](#). If the previous state is either inactive or active the `on_cleanup()` method is called first. Then regardless of the previous state, the `_joints` map is cleared.

Parameters

<code>previous_state</code>	
-----------------------------	--

Returns

`hardware_interface::CallbackReturn`

12.3.2.10 `perform_command_mode_switch()`

```
hardware_interface::return_type bioscara.hardware_interface::BioscaraHardwareInterface::perform_command_mode_switch (
    const std::vector< std::string > & start_interfaces,
    const std::vector< std::string > & stop_interfaces ) [override]
```

Perform the mode-switching for the new command interface combination.

Performs the following actions:

- **On activation:**

- **home** interface:

- * Reset command to 0.0. This clears any remaining commands that have been written to the command interface while the hardware was unable to act on it. For example if it was inactive or the homing command was not the active command mode.

Note

This is part of the realtime update loop, and should be fast.

Parameters

in	<code>start_interfaces</code>	vector of string identifiers for the command interfaces starting.
in	<code>stop_interfaces</code>	vector of string identifiers for the command interfaces stopping.

Returns

`return_type::OK` if the new command interface combination can be switched to (or) if the interface key is not relevant to this system. Returns `return_type::ERROR` otherwise.

12.3.2.11 prepare_command_mode_switch()

```
hardware_interface::return_type bioscara_hardware_interface::BioscaraHardwareInterface::prepare←
_command_mode_switch (
    const std::vector< std::string > & start_interfaces,
    const std::vector< std::string > & stop_interfaces ) [override]
```

Performs checks and book keeping of the active control mode when changing controllers.

For safe operation only one controller may interact with the hardware at the time. For example if the velocity JTC is active and has claimed the velocity command interfaces it is technically possible to activate the position JTC (or a homing controller, or others) that claim a different command interface (position in this case). However if both controllers are active they start writing to the hardware simultaneously which is to be avoided. For this reason a book keeping mechanism has been implemented which stores the currently active command interfaces for each joint in the `_joint_command_modes` member. Each joint has a set of active command interfaces. When a controller switch is performed the interfaces that should be stopped are removed from each joint set, then the one that should be started are added, if they are already present an error is thrown. Lastly a validation is performed. Currently the validation is simple since each joint may only have one command interface. The validation can be expanded for future use cases that require a combination of active command interfaces per joint for example.

The following basic checks are implemented:

- **On deactivation:**

- [ERROR] Homing command interfaces may only be deactivated if no current homing process is ongoing (`Joint::getCurrentBCmd() != Joint::HOME`)
- [WARN] Deactivating a velocity command interface if the velocity set point is 0.0.
- [WARN] Deactivating a command interface that has not been started. This should not happen.

- **On activation:**

- [ERROR] Activating a command interface that is already started. This should not happen.
- [ERROR] Activating a second command interface for a joint.
- [ERROR] Activating 'position' or 'velocity' command interface if the joint is not homed (`Joint::isHomed() == false`).

Parameters

<code>start_interfaces</code>	command interfaces that should be started in the form "joint/interface"
<code>stop_interfaces</code>	command interfaces that should be stopped in the form "joint/interface"

Returns

`hardware_interface::return_type`

12.3.2.12 read()

```
hardware_interface::return_type bioscara_hardware_interface::BioscaraHardwareInterface::read (
```

```
const rclcpp::Time & time,
const rclcpp::Duration & period ) [override]
```

Reads from the hardware and populates the state interfaces.

Iterates over all state interfaces and calls the corresponding Joint method.

- State interface "position" -> Joint::getPosition()
- State interface "velocity" -> Joint::getVelocity()
- State interface "home" -> Joint::isHomed()
 - This does not actually trigger a communication, instead it relies on the return flags of the previous transmissions. Since position and velocity have been called immediately before the return flags are assumed to be valid.
 - If the homing of a joint has been activated through the command interface (Joint::getCurrentBCmd() == Joint::HOME) the device signals BUSY (Joint::isBusy()) as long as it is still homing. If the BUSY flag is reset while the current command is still Joint::HOME we can assume the homing has finished. Then the "home" command interface of the joint is reset to 0.0, which will stop the homing (perform cleanup tasks) at the next write cycle.

Parameters

<i>time</i>	
<i>period</i>	

Returns

hardware_interface::return_type

12.3.2.13 split_interface_string_to_joint_and_name()

```
void bioscara.hardware_interface::BioscaraHardwareInterface::split_interface_string_to_joint_and_name (
    std::string interface,
    std::string & joint_name,
    std::string & interface_name ) [private]
```

Split a interface string like "<joint_name>/<interface_name>" to "<joint_name>" and "<interface_name>".

Parameters

<i>interface</i>	
<i>joint_name</i>	
<i>interface_name</i>	

12.3.2.14 start_homing()

```
bioscara.hardware_driver::err_type_t bioscara.hardware_interface::BioscaraHardwareInterface::start_homing (
```

```
const std::string name,
float velocity ) [private]
```

wrapper method to start homing.

Activate the joint, set homing acceleration and start homing.

Parameters

<i>name</i>	
<i>velocity</i>	

Returns

[bioscara.hardware_driver::err_type_t](#)

12.3.2.15 stop_homing()

```
bioscara.hardware_driver::err_type_t bioscara.hardware_interface::BioscaraHardwareInterface<-
::stop_homing (
    const std::string name ) [private]
```

wrapper method to stop homing.

Stop the homing. Reset acceleration and velocity and perform the postHoming cleanup, then deactivate the joint.

Parameters

<i>name</i>	
-------------	--

Returns

[bioscara.hardware_driver::err_type_t](#)

12.3.2.16 write()

```
hardware_interface::return_type bioscara.hardware_interface::BioscaraHardwareInterface::write
(
    const rclcpp::Time & time,
    const rclcpp::Duration & period ) [override]
```

Writes commands to the hardware from the command interfaces.

In contrast to the [read\(\)](#) method the [write\(\)](#) method only loops over the command interfaces that are currently active defined by the [BioscaraHardwareInterface::joint_command_modes](#) map. See [prepare_command_mode_switch\(\)](#) for a detailed reasoning why this approach has been chosen.

- Command interface "position" -> Joint::setPosition()
- Command interface "velocity" -> Joint::setVelocity()

- Command interface "home" -> Joint::startHoming()
 - If the commanded value in "home" is != 0.0 and the joint is currently executing a blocking function, for example homing (Joint::getCurrentBCmd() == Joint::NONE), the homing sequence is started with the speed, sensitivity, current and acceleration defined in the BioscaraHardwareInterface::_joint_cfg which is populated from the hardware description urdf. The direction of the homing is determined by the sign of the command interface value.
 - If the commanded value in "home" is = 0.0 and the joint is currently executing homing, the homing is stopped. This can either happen prematurely through user input or when the homing is completed which is registered in [read\(\)](#).

Parameters

<i>time</i>	
<i>period</i>	

Returns

hardware_interface::return_type

12.3.3 Member Data Documentation

12.3.3.1 _joint_cfg

```
std::unordered_map<std::string, joint\_config\_t

```

unordered map storing the configuration struct of the joints.

An unordered map is chosen to simplify acces via the joint name, as this conforms well with the ROS2_control hardware interface. The map does not need to be ordered. Search, insertion, and removal of elements have average constant-time complexity.

12.3.3.2 _joint_command_modes

```
std::unordered_map<std::string, std::set<std::string>> bioscara.hardware_interface::BioscaraHardwareInterface::_joint_command_modes [private]
```

unordered map of sets storing the active command interfaces for each joint.

Each joint can have a set of active command interfaces. This type of structure is chosen to group interfaces by joint. In the [write\(\)](#) function the interface name can simply be constructed by concatenating joint name with interface name. Although currently only one active command interface is allowed at the time, a set can be used to store multiple command interfaces that are acceptable to be combined, for example it would be acceptable to set velocity and driver current and hence that would be an allowable combination.

An unordered map is chosen to simplify acces via the joint name, as this conforms well with the ROS2_control hardware interface. The map does not need to be ordered. Search, insertion, and removal of elements have average constant-time complexity.

12.3.3.3 _joints

```
std::unordered_map<std::string, std::unique_ptr<bioscara\_hardware\_driver::BaseJoint> > bioscara_hardware_interface::BioscaraHardwareInterface::_joints [private]
```

unordered map storing the pointers to BaseJoint objects. This will either be a MockJoint or Joint.

An unordered map is chosen to simplify acces via the joint name, as this conforms well with the ROS2_control hardware interface. The map does not need to be ordered. Search, insertion, and removal of elements have average constant-time complexity.

Since the BaseJoint methods are implemented as virtual, dynamic method dispatch can be utilized to call the correct implementation of a method. So either BaseJoint::foo() or Joint::foo()/MockJoint::foo() if foo() is overwritten in Joint or MockJoint. A smart pointer is used to guarantee destruction when the pointer is destructed. A unique pointer is used to prevent copying of the object.

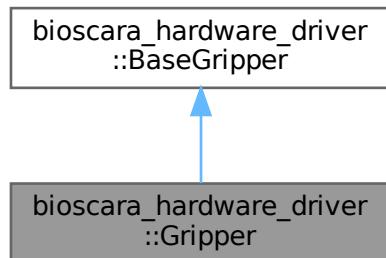
The documentation for this class was generated from the following files:

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include/bioscara_hardware_interface/[bioscara_hardware.hpp](#)
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/src/[bioscara_hardware.cpp](#)

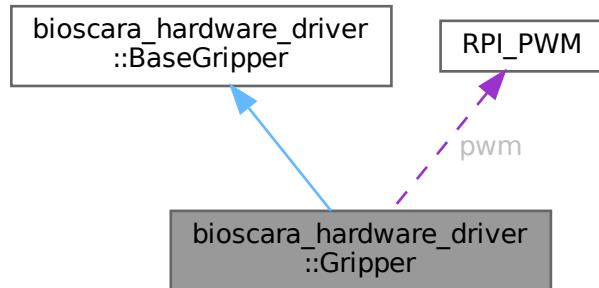
12.4 bioscara_hardware_driver::Gripper Class Reference

```
#include <mGripper.h>
```

Inheritance diagram for bioscara_hardware_driver::Gripper:



Collaboration diagram for bioscara.hardware_driver::Gripper:



Public Member Functions

- `Gripper (float reduction, float offset, float min, float max)`
Constructor of the hardware `Gripper` object.
- `err_type_t enable (void) override`
Prepares the servo for use.
- `err_type_t disable (void) override`
Disables the servo.
- `err_type_t setPosition (float width) override`
Sets the gripper width in m from the closed position.
- `err_type_t setServoPosition (float angle) override`
Sets the servo position of the gripper actuator in degrees.
- `void setReduction (float reduction)`
Manually set reduction.
- `void setOffset (float offset)`
Manually set offset.

Public Member Functions inherited from bioscara.hardware_driver::BaseGripper

- `BaseGripper (void)`
- `virtual err_type_t init (void)`
Placeholder, does nothing.
- `virtual err_type_t_deinit (void)`
Placeholder, does nothing.

Protected Attributes

- `float reduction = 1`
Joint to actuator reduction ratio.
- `float offset = 0`
Joint position offset.
- `float min = 0`
Joint lower limit.
- `float max = 0`
Joint upper limit.

Private Attributes

- RPI_PWM `pwm`
- int `freq` = 50

12.4.1 Constructor & Destructor Documentation**12.4.1.1 Gripper()**

```
bioscara.hardware_driver::Gripper::Gripper (
    float reduction,
    float offset,
    float min,
    float max )
```

Constructor of the hardware [Gripper](#) object.

The gripper width in m is converted to a PWM dutycycle via the JOINT2ACTUATOR macro.

Parameters

<code>reduction</code>	
<code>offset</code>	
<code>min</code>	minimum width in m.
<code>max</code>	maximum width in m.

12.4.2 Member Function Documentation**12.4.2.1 disable()**

```
err_type_t bioscara.hardware_driver::Gripper::disable (
    void ) [override], [virtual]
```

Disables the servo.

Stops the servo and disables the PWM generation.

Returns

return code of bioscara.hardware_driver::esp_err_t type.

Reimplemented from [bioscara.hardware_driver::BaseGripper](#).

12.4.2.2 enable()

```
err_type_t bioscara.hardware_driver::Gripper::enable (
    void ) [override], [virtual]
```

Prepares the servo for use.

Starts the PWM generation but does not set a position. Must be called before a position is set. The PWM pin is GPIO18. PWM chip is 0, channel 0.

Returns

return code of bioscara.hardware_driver::esp_err_t type

Reimplemented from [bioscara.hardware_driver::BaseGripper](#).

12.4.2.3 setOffset()

```
void bioscara.hardware_driver::Gripper::setOffset (
    float offset ) [virtual]
```

Manually set offset.

Reimplemented from [bioscara.hardware_driver::BaseGripper](#).

12.4.2.4 setPosition()

```
err_type_t bioscara.hardware_driver::Gripper::setPosition (
    float width ) [override], [virtual]
```

Sets the gripper width in m from the closed position.

Arguments outside the allowed range are bounded to limit min and max.

Parameters

<i>width</i>	width in m.
--------------	-------------

Reimplemented from [bioscara.hardware_driver::BaseGripper](#).

12.4.2.5 setReduction()

```
void bioscara.hardware_driver::Gripper::setReduction (
    float reduction ) [virtual]
```

Manually set reduction.

Parameters

<i>reduction</i>	
------------------	--

Reimplemented from [bioscara.hardware_driver::BaseGripper](#).

12.4.2.6 setServoPosition()

```
err_type_t bioscara.hardware_driver::Gripper::setServoPosition (
    float angle ) [override], [virtual]
```

Sets the servo position of the gripper actuator in degrees.

Parameters

<i>angle</i>	in degrees.
--------------	-------------

Reimplemented from [bioscara.hardware_driver::BaseGripper](#).

12.4.3 Member Data Documentation

12.4.3.1 freq

```
int bioscara.hardware_driver::Gripper::freq = 50 [private]
```

12.4.3.2 max

```
float bioscara.hardware_driver::Gripper::max = 0 [protected]
```

[Joint](#) upper limit.

12.4.3.3 min

```
float bioscara.hardware_driver::Gripper::min = 0 [protected]
```

[Joint](#) lower limit.

12.4.3.4 offset

```
float bioscara.hardware_driver::Gripper::offset = 0 [protected]
```

[Joint](#) position offset.

12.4.3.5 pwm

```
RPI_PWM bioscara.hardware_driver::Gripper::pwm [private]
```

12.4.3.6 reduction

```
float bioscara.hardware_driver::Gripper::reduction = 1 [protected]
```

[Joint](#) to actuator reduction ratio.

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

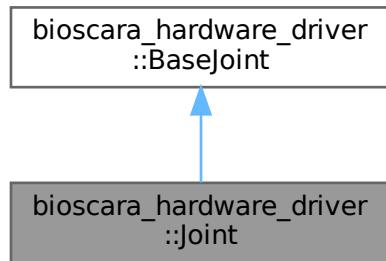
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/include/bioscara.hardware_driver/[mGripper.h](#)
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/src/[mGripper.cpp](#)

12.5 bioscara.hardware_driver::Joint Class Reference

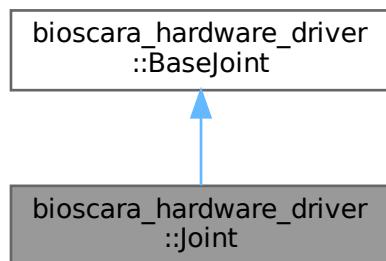
Representing a single joint on the I2C bus.

```
#include <mJoint.h>
```

Inheritance diagram for bioscara.hardware_driver::Joint:



Collaboration diagram for bioscara.hardware_driver::Joint:



Public Member Functions

- `Joint` (const std::string `name`, const int `address`, const float `reduction`, const float `min`, const float `max`)
Create a `Joint` object.
- `~Joint` (void)
- `err_type_t init` (void) override
Established connection to a joint via I2C.
- `err_type_t_deinit` (void) override
Disconnects from a joint.
- `err_type_t enable` (u_int8_t `driveCurrent`, u_int8_t `holdCurrent`) override
Setup the joint and engages motor.

- **err_type_t postHoming** (void) override
perform tasks after a non-blocking homing.
- **err_type_t getPosition** (float &pos) override
get the current joint position in radians or m for cylindrical and prismatic joints respectively.
- **err_type_t setPosition** (float pos) override
set the current joint position in radians or m for cylindrical and prismatic joints respectively.
- **err_type_t moveSteps** (int32_t steps) override
Move full steps.
- **err_type_t getVelocity** (float &vel) override
get the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively.
- **err_type_t setVelocity** (float vel) override
Set the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively.
- **err_type_t checkOrientation** (float angle=10.0) override
Calls the checkOrientation method of the motor. Checks in which direction the motor is turning.
- **err_type_t stop** (void) override
Stops the motor.
- **err_type_t disableCL** (void) override
Disables the Closed-Loop PID Controller.
- **err_type_t setDriveCurrent** (u_int8_t current) override
Set the Drive Current.
- **err_type_t setHoldCurrent** (u_int8_t current) override
Set the Hold Current.
- **err_type_t setBrakeMode** (u_int8_t mode) override
Set Brake Mode.
- **err_type_t setMaxAcceleration** (float maxAccel) override
Set the maximum permitted joint acceleration (and deceleration) in rad/s^2 or m/s^2 for cylindrical and prismatic joints respectively.
- **err_type_t setMaxVelocity** (float maxVel) override
Set the maximum permitted joint velocity in rad/s or m/s for cylindrical and prismatic joints respectively.
- **err_type_t enableStallguard** (u_int8_t sensitivity) override
Enable encoder stall detection of the joint. Derived class may override this.
- **err_type_t getFlags** (void) override
- **err_type_t getHomingOffset** (float &offset)
Retrieves the homing position from the last homing.
- **err_type_t setHomingOffset** (const float offset)
Stores the homing position on the joint.

Public Member Functions inherited from bioscara_hardware_driver::BaseJoint

- **BaseJoint** (const std::string name)
Create a Joint object.
- **~BaseJoint** (void)
- **virtual err_type_t disable** (void)
disengages the joint motors, derived classes may override this.
- **virtual err_type_t home** (float velocity, u_int8_t sensitivity, u_int8_t current)
Blocking implementation to home the joint, derived classes may override this.
- **virtual err_type_t startHoming** (float velocity, u_int8_t sensitivity, u_int8_t current)
non-blocking implementation to home the joint, derived classes may override this.
- **virtual bool isHomed** (void)
Checks the state if the motor is homed.

- virtual bool `isEnabled` (void)
Checks the state if the motor is enabled.
- virtual bool `isStalled` (void)
Checks if the motor is stalled.
- virtual bool `isBusy` (void)
Checks if the joint controller is busy processing a blocking command.
- virtual `err_type_t getFlags` (`u_int8_t &flags`)
- virtual `stp_reg_t getCurrentBCmd` (void)
get the currently active blocking command

Protected Member Functions

- `err_type_t _home` (`float velocity, u_int8_t sensitivity, u_int8_t current`)
Call to start the homing sequence of a joint.
- `err_type_t checkCom` (void)
Check if communication to the joint is established.

Protected Member Functions inherited from `bioscara.hardware_driver::BaseJoint`

- virtual void `wait_while_busy` (`const float period_ms`)
Blocking loop waiting for BUSY flag to reset.

Protected Attributes

- float `reduction` = 1
Joint to actuator reduction ratio.
- float `offset` = 0
Joint position offset.
- float `min` = 0
Joint lower limit.
- float `max` = 0
Joint upper limit.

Protected Attributes inherited from `bioscara.hardware_driver::BaseJoint`

- `u_int8_t flags` = 0b00001100
State flags transmitted with every I2C transaction.
- `stp_reg_t current_b_cmd` = `NONE`
Keeps track if a blocking command is being executed.

Private Member Functions

- template<typename T >
int `read` (`const stp_reg_t reg, T &data, u_int8_t &flags`)
Wrapper function to request data from the I2C slave.
- template<typename T >
int `write` (`const stp_reg_t reg, T data, u_int8_t &flags`)
Wrapper function to send command to the I2C slave.

Private Attributes

- int `address`
I2C address.
- int `handle` = -1
I2C bus handle.

Additional Inherited Members**Public Types inherited from bioscara.hardware_driver::BaseJoint**

- enum `stp_reg_t` {

NONE = 0x00 , PING = 0x0f , SETUP = 0x10 , SETRPM = 0x11 ,

GETDRIVERRPM = 0x12 , MOVESTEPS = 0x13 , MOVEANGLE = 0x14 , MOVETOANGLE = 0x15 ,

GETMOTORSTATE = 0x16 , RUNCOTINOUS = 0x17 , ANGLEMOVED = 0x18 , SETCURRENT = 0x19 ,

SETHOLDCURRENT = 0x1A , SETMAXACCELERATION = 0x1B , SETMAXDECELERATION = 0x1C ,

SETMAXVELOCITY = 0x1D ,

ENABLESTALLGUARD = 0x1E , DISABLESTALLGUARD = 0x1F , CLEARSTALL = 0x20 , SETBRAKEMODE

= 0x22 ,

ENABLEPID = 0x23 , DISABLEPID = 0x24 , ENABLECLOSEDLOOP = 0x25 , DISABLECLOSEDLOOP =
 0x26 ,

SETCONTROLTHRESHOLD = 0x27 , MOVETOEND = 0x28 , STOP = 0x29 , GETPIDERROR = 0x2A ,

CHECKORIENTATION = 0x2B , GETENCODERRPM = 0x2C , HOME = 0x2D , HOMEOFFSET = 0x2E }

register and command definitions

Public Attributes inherited from bioscara.hardware_driver::BaseJoint

- std::string `name`

12.5.1 Detailed Description

Representing a single joint on the I2C bus.

12.5.2 Constructor & Destructor Documentation**12.5.2.1 Joint()**

```
bioscara.hardware_driver::Joint::Joint (
    const std::string name,
    const int address,
    const float reduction,
    const float min,
    const float max )
```

Create a `Joint` object.

The `Joint` object represents a single joint and its actuator. Each `Joint` has a transmission with the following relationship:

$$\begin{aligned} \text{actuator position} &= (\text{joint position} - \text{offset}) * \text{reduction} \\ \text{joint position} &= \text{actuator position} / \text{reduction} + \text{offset} \end{aligned}$$

Parameters

<i>name</i>	string device name for identification
<i>address</i>	1-byte I2C device address (0x11 ... 0x14) for J1 ... J4
<i>reduction</i>	gear reduction of the joint. This is used to transform position and velocity values between joint units and actuator (stepper) units. The sign depends on the direction the motor is mounted and is turning. Adjust such that the joint moves in the positive direction on positive joint commands. Cable polarity has no effect since the motors automatically adjust to always run in the 'right' direction from their point of view. J1: 35 J2: -2*pi/0.004 (4 mm linear movement per stepper revolution) J3: 24 J4: 12
<i>min</i>	lower joint limit in joint units. J1: -3.04647 J2: -0.0016 J3: -2.62672 J4: -3.01069
<i>max</i>	upper joint limit in joint units. J1: 3.04647 J2: 0.3380 J3: 2.62672 J4: 3.01069

12.5.2.2 ~Joint()

```
bioscara.hardware_driver::Joint::~Joint (
    void )
```

12.5.3 Member Function Documentation**12.5.3.1 _home()**

```
err_type_t bioscara.hardware_driver::Joint::_home (
    float velocity,
    u_int8_t sensitivity,
    u_int8_t current ) [protected], [virtual]
```

Call to start the homing sequence of a joint.

First the joint will check the motor wiring by executing the checkOrientation internally. Then it will set the specified speed until a resistance which drives the PID error above the specified threshold is encountered. At this point the stepper stops and zeros the encoder.

Parameters

<i>velocity</i>	signed velocity in rad/s or m/s. Must be between $1.0 < \text{RAD2DEG}(\text{JOINT2ACTUATOR}(\text{velocity}, \text{reduction}, 0)) / 6 < 250.0$
<i>sensitivity</i>	Encoder pid error threshold 0 to 255.
<i>current</i>	homing current, determines how easy it is to stop the motor and thereby provoke a stall

Returns

0 on success, -1 on communication error, -3 when the motor is not enabled, -5 if the joint is not initialized, -101 if the velocity is zero, -102 if absolute value of the velocity is outside the specified limits.

Implements [bioscara.hardware_driver::BaseJoint](#).

12.5.3.2 checkCom()

```
err_type_t bioscara.hardware_driver::Joint::checkCom (
    void ) [protected]
```

Check if communication to the joint is established.

Sends a PING to and expects a ACK from the joint.

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

12.5.3.3 checkOrientation()

```
err_type_t bioscara.hardware_driver::Joint::checkOrientation (
    float angle = 10.0 ) [override], [virtual]
```

Calls the checkOrientation method of the motor. Checks in which direction the motor is turning.

As the orientation check is blocking on the motor, this function returns when the isBusy flag is clear again.

Parameters

<code>angle</code>	degrees how much the motor should turn. A few degrees is sufficient.
--------------------	--

Returns

0 on success, -1 on communication error, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.4 deinit()

```
err_type_t bioscara.hardware_driver::Joint::deinit (
    void ) [override], [virtual]
```

Disconnects from a joint.

Removes the joint from the I2C bus.

Returns

0 on success, -1 when the joint could not be removed due to an I2C error, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.5 disableCL()

```
err_type_t bioscara.hardware_driver::Joint::disableCL (
    void ) [override], [virtual]
```

Disables the Closed-Loop PID Controller.

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.6 enable()

```
err_type_t bioscara.hardware_driver::Joint::enable (
    u_int8_t driveCurrent,
    u_int8_t holdCurrent ) [override], [virtual]
```

Setup the joint and engages motor.

This function prepares the motor for movement. After successfull execution the joint is ready to accept [Joint::setPosition\(\)](#) and [Joint::setVelocity\(\)](#) commands.

The function sets the drive and hold current for the specified joint and engages the motor. The currents are in percent of driver max. output (2.5A, check with TMC5130 datasheet or uStepper documentation)

Parameters

<i>driveCurrent</i>	drive current in 0-100 % of 2.5A output (check uStepper doc.)
<i>holdCurrent</i>	hold current in 0-100 % of 2.5A output (check uStepper doc.)

Returns

0 on success, -1 on communication error, -3 when the motor is not enabled, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.7 enableStallguard()

```
err_type_t bioscara.hardware_driver::Joint::enableStallguard (
    u_int8_t sensitivity ) [override], [virtual]
```

Enable encoder stall detection of the joint. Derived class may override this.

If the PID error exceeds the set threshold a stall is triggered and the motor disabled. A detected stall can be reset by homing or reenabling the joint using [enable\(\)](#).

Note

If stall detection shall be enabled, invoke this method AFTER enabling the joint with [enable\(\)](#).

Parameters

<i>sensitivity</i>	value of threshold. 0 - 255 where lower is more sensitive.
--------------------	--

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.8 getFlags()

```
err_type_t bioscara.hardware_driver::Joint::getFlags (
    void ) [override], [virtual]
```

get driver state flags

Returns

0 on success, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.9 getHomingOffset()

```
err_type_t bioscara.hardware_driver::Joint::getHomingOffset (
    float & offset )
```

Retrieves the homing position from the last homing.

The homing position is stored on the joint to make it persistent as long as the joint is powered up.

Returns

0 on success, -1 on communication error, -2 when not homed, -5 if the joint is not initialized.

12.5.3.10 getPosition()

```
err_type_t bioscara.hardware_driver::Joint::getPosition (
    float & pos ) [override], [virtual]
```

get the current joint position in radians or m for cylindrical and prismatic joints respectively.

Warning

If the joint is not homed this method does not return an error. Instead *pos* will be 0.0.

Parameters

<i>pos</i>	<input type="text"/>
------------	----------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Implements [bioscara_hw_driver::BaseJoint](#).

12.5.3.11 getVelocity()

```
err_type_t bioscara_hw_driver::Joint::getVelocity (
    float & vel ) [override], [virtual]
```

get the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively.

Parameters

<i>vel</i>	<input type="text"/>
------------	----------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Implements [bioscara_hw_driver::BaseJoint](#).

12.5.3.12 init()

```
err_type_t bioscara_hw_driver::Joint::init (
    void ) [override], [virtual]
```

Established connection to a joint via I2C.

Adds the joint to the I2C bus and tests if is responsive by sending a PING.

Returns

0 on success, -1 on when no ACK is received from the joint, -2 if the I2C device could not be opened given the joint address.

Reimplemented from [bioscara_hw_driver::BaseJoint](#).

12.5.3.13 moveSteps()

```
err_type_t bioscara_hw_driver::Joint::moveSteps (
    int32_t steps ) [override], [virtual]
```

Move full steps.

This function can be called even when not homed.

Parameters

<code>steps</code>	number of full steps
--------------------	----------------------

Returns

0 on success, -1 on communication error, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.14 postHoming()

```
err_type_t bioscara.hardware_driver::Joint::postHoming (
    void ) [override], [virtual]
```

perform tasks after a non-blocking homing.

This method resets the current_b_cmd to NONE, checks if the joint is homed, and saves the homing offset to the joint.

Returns

0 on success, -109 if the current_b_cmd is not HOME, -1 on communication error, -2 when not homed, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.15 read()

```
template<typename T >
int bioscara.hardware_driver::Joint::read (
    const stp_reg_t reg,
    T & data,
    u_int8_t & flags ) [private]
```

Wrapper function to request data from the I2C slave.

Allocates a buffer of size sizeof(T) + RFLAGS_SIZE. invokes [readFromI2CDev\(\)](#), and copies the received payload to *data* and the transmisison flags to *flags*. See [Joint::flags](#) for details.

- Todo**
- Implement a return code for read only functions
 - Implement clearStall function

Template Parameters

<code>T</code>	Datatype of value to be transmitted
----------------	-------------------------------------

Parameters

<i>reg</i>	stp_reg_t register to read
<i>data</i>	reference to store payload.
<i>flags</i>	reference to a byte which stores the return flags

Returns

0 on OK, negative on error

12.5.3.16 setBrakeMode()

```
err_type_t bioscara.hardware_driver::Joint::setBrakeMode (
    u_int8_t mode ) [override], [virtual]
```

Set Brake Mode.

Parameters

<i>mode</i>	Freewheel: 0, Coolbrake: 1, Hardbrake: 2
-------------	--

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.17 setDriveCurrent()

```
err_type_t bioscara.hardware_driver::Joint::setDriveCurrent (
    u_int8_t current ) [override], [virtual]
```

Set the Drive Current.

Warning

This function is unreliable and not well tested. Use [Joint::enable\(\)](#) instead!

Parameters

<i>current</i>	0% - 100% of driver current
----------------	-----------------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.18 setHoldCurrent()

```
err_type_t bioscara.hardware_driver::Joint::setHoldCurrent (
    u_int8_t current ) [override], [virtual]
```

Set the Hold Current.

Warning

This function is unreliable and not well tested. Use [Joint::enable\(\)](#) instead!

Parameters

<i>current</i>	0% - 100% of driver current
----------------	-----------------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.19 setHomingOffset()

```
err_type_t bioscara.hardware_driver::Joint::setHomingOffset (
    const float offset )
```

Stores the homing position on the joint.

The homing position is stored on the joint to make it persistent as long as the joint is powered up.

Returns

0 on success, -1 on communication error, -2 if not homed, -5 if the joint is not initialized.

12.5.3.20 setMaxAcceleration()

```
err_type_t bioscara.hardware_driver::Joint::setMaxAcceleration (
    float maxAccel ) [override], [virtual]
```

Set the maximum permitted joint acceleration (and deceleration) in rad/s² or m/s² for cylindrical and prismatic joints respectively.

Parameters

<i>maxAccel</i>	maximum joint acceleration.
-----------------	-----------------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara_hw_driver::BaseJoint](#).

12.5.3.21 setMaxVelocity()

```
err_type_t bioscara_hw_driver::Joint::setMaxVelocity (
    float maxVel ) [override], [virtual]
```

Set the maximum permitted joint velocity in rad/s or m/s for cylindrical and prismatic joints respectively.

Parameters

<i>maxVel</i>	maximum joint velocity.
---------------	-------------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara_hw_driver::BaseJoint](#).

12.5.3.22 setPosition()

```
err_type_t bioscara_hw_driver::Joint::setPosition (
    float pos ) [override], [virtual]
```

set the current joint position in radians or m for cylindrical and prismatic joints respectively.

Parameters

<i>pos</i>	in rad or m
------------	-------------

Returns

0 on success, -1 on communication error, -2 when not homed, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented from [bioscara_hw_driver::BaseJoint](#).

12.5.3.23 setVelocity()

```
err_type_t bioscara_hw_driver::Joint::setVelocity (
    float vel ) [override], [virtual]
```

Set the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively.

Parameters

<i>vel</i>	
------------	--

Returns

0 on success, -1 on communication error, -2 when not homed, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.24 stop()

```
err_type_t bioscara.hardware_driver::Joint::stop (
    void ) [override], [virtual]
```

Stops the motor.

Stops the motor by setting the maximum velocity to zero and the position setpoint to the current position

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara.hardware_driver::BaseJoint](#).

12.5.3.25 write()

```
template<typename T >
int bioscara.hardware_driver::Joint::write (
    const stp_reg_t reg,
    T data,
    u_int8_t & flags ) [private]
```

Wrapper function to send command to the I2C slave.

Allocates a buffer of size `sizeof(T) + RFLAGS_SIZE`. Copies `data` to the buffer and invokes [writeToI2CDev\(\)](#). The flags received from the transaction are copied to `flags`. The flags are described in [Joint::read\(\)](#).

Template Parameters

<i>T</i>	Datatype of value to be transmitted
----------	-------------------------------------

Parameters

<i>reg</i>	stp_reg_t command to execute
<i>data</i>	payload to transmit. It is the users responsibility to populate the right amount of data for the relevant register
<i>flags</i>	reference to a byte which stores the return flags

Returns

0 on OK, negative on error

12.5.4 Member Data Documentation

12.5.4.1 address

```
int bioscara.hardware_driver::Joint::address [private]
```

I2C adress.

12.5.4.2 handle

```
int bioscara.hardware_driver::Joint::handle = -1 [private]
```

I2C bus handle.

12.5.4.3 max

```
float bioscara.hardware_driver::Joint::max = 0 [protected]
```

[Joint](#) upper limit.

12.5.4.4 min

```
float bioscara.hardware_driver::Joint::min = 0 [protected]
```

[Joint](#) lower limit.

12.5.4.5 offset

```
float bioscara.hardware_driver::Joint::offset = 0 [protected]
```

[Joint](#) position offset.

12.5.4.6 reduction

```
float bioscara.hardware_driver::Joint::reduction = 1 [protected]
```

[Joint](#) to actuator reduction ratio.

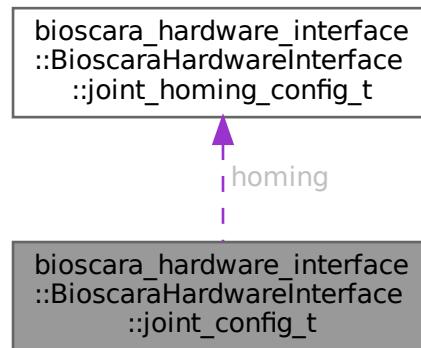
The documentation for this class was generated from the following files:

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/include/bioscara.hardware_driver/[mJoint.h](#)
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/include/bioscara.hardware_driver/[mJoint.hpp](#)
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/src/[mJoint.cpp](#)

12.6 bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t Struct Reference

configuration structure holding the passed paramters from the ros2_control urdf

Collaboration diagram for bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t:



Public Attributes

- int `i2c_address`
- float `reduction` = 1
- float `min`
- float `max`
- u_int8_t `drive_current`
- u_int8_t `hold_current`
- u_int8_t `stall_threshold`
- float `max_velocity`
- float `max_acceleration`
- `joint_homing_config_t` `homing`

12.6.1 Detailed Description

configuration structure holding the passed paramters from the ros2_control urdf

Saving all parameters on initialization in a structure allows for quick access during runtime.

12.6.2 Member Data Documentation

12.6.2.1 drive_current

```
u_int8_t bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::drive_current
```

12.6.2.2 hold_current

```
u_int8_t bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::hold_current
```

12.6.2.3 homing

```
joint_homing_config_t bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::homing
```

12.6.2.4 i2c_address

```
int bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::i2c_address
```

12.6.2.5 max

```
float bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::max
```

12.6.2.6 max_acceleration

```
float bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::max_acceleration
```

12.6.2.7 max_velocity

```
float bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::max_velocity
```

12.6.2.8 min

```
float bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::min
```

12.6.2.9 reduction

```
float bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::reduction = 1
```

12.6.2.10 stall_threshold

```
u_int8_t bioscara.hardware_interface::BioscaraHardwareInterface::joint_config_t::stall_threshold
```

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

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_interface/include/bioscara.hardware_interface/bioscara.hardware.hpp

12.7 bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing_config_t Struct Reference

configuration structure holding the passed homing paramters from the ros2_control urdf

Public Attributes

- float `speed` = 0
- u_int8_t `threshold` = 10
- u_int8_t `current` = 10
- float `acceleration` = 0.01

12.7.1 Detailed Description

configuration structure holding the passed homing paramters from the ros2_control urdf

Saving all parameters on initialization in a structure allows for quick access during runtime.

12.7.2 Member Data Documentation

12.7.2.1 acceleration

```
float bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing_config_t::acceleration  
= 0.01
```

12.7.2.2 current

```
u_int8_t bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing_config_t::current  
= 10
```

12.7.2.3 speed

```
float bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing_config_t::speed = 0
```

12.7.2.4 threshold

```
u_int8_t bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing_config_t::threshold  
= 10
```

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

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_interface/include/bioscara.hardware_interface/[bioscara.hardware.hpp](#)

12.8 Lowpass Class Reference

```
#include <filters.h>
```

Public Member Functions

- [Lowpass](#) (float gain=1, float sampleTime=0.1, float timeconstant=1.0)
A simple lowpass filter class.
- float [updateState](#) (float u)
- void [resetState](#) (void)

Protected Attributes

- float [K](#)
- float [Ts](#)
- float [tau](#)
- float [x](#)

12.8.1 Constructor & Destructor Documentation

12.8.1.1 Lowpass()

```
Lowpass::Lowpass (
    float gain = 1,
    float sampleTime = 0.1,
    float timeconstant = 1.0 ) [inline]
```

A simple lowpass filter class.

Parameters

<i>gain</i>	
<i>sampleTime</i>	
<i>timeconstant</i>	

12.8.2 Member Function Documentation

12.8.2.1 resetState()

```
void Lowpass::resetState (
    void ) [inline]
```

12.8.2.2 updateState()

```
float Lowpass::updateState (
    float u ) [inline]
```

12.8.3 Member Data Documentation

12.8.3.1 K

```
float Lowpass::K [protected]
```

12.8.3.2 tau

```
float Lowpass::tau [protected]
```

12.8.3.3 Ts

```
float Lowpass::Ts [protected]
```

12.8.3.4 x

```
float Lowpass::x [protected]
```

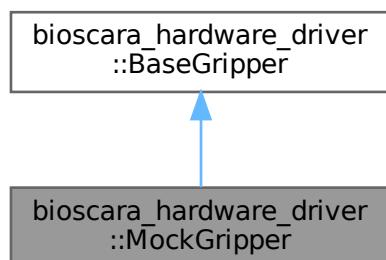
The documentation for this class was generated from the following file:

- Arduino/joint/[filters.h](#)

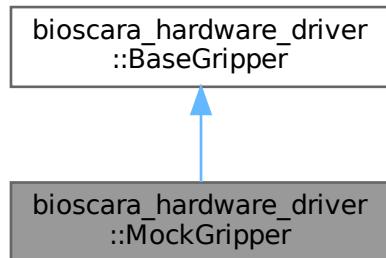
12.9 bioscara.hardware_driver::MockGripper Class Reference

```
#include <mMockGripper.h>
```

Inheritance diagram for bioscara.hardware_driver::MockGripper:



Collaboration diagram for bioscara_hardware_driver::MockGripper:



Public Member Functions

- [MockGripper](#) (void)

Public Member Functions inherited from [bioscara_hardware_driver::BaseGripper](#)

- [BaseGripper](#) (void)
- virtual [err_type_t init](#) (void)
Placeholder, does nothing.
- virtual [err_type_t_deinit](#) (void)
Placeholder, does nothing.
- virtual [err_type_t enable](#) (void)
Prepares the servo for use.
- virtual [err_type_t disable](#) (void)
Disables the servo.
- virtual [err_type_t setPosition](#) (float width)
Sets the gripper width in m from the closed position.
- virtual [err_type_t setServoPosition](#) (float angle)
Sets the servo position of the gripper actuator in degrees.
- virtual void [setReduction](#) (float reduction)
Manually set reduction.
- virtual void [setOffset](#) (float offset)
Manually set offset.

12.9.1 Constructor & Destructor Documentation

12.9.1.1 [MockGripper\(\)](#)

```
bioscara_hardware_driver::MockGripper::MockGripper (
    void
)
```

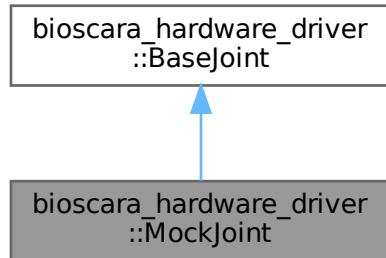
The documentation for this class was generated from the following files:

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/[mMockGripper.h](#)
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/[mMockGripper.cpp](#)

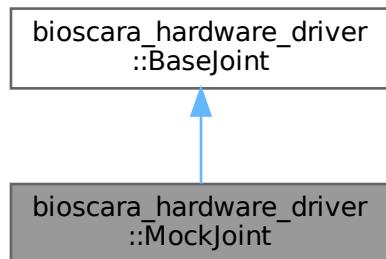
12.10 bioscara_hardware_driver::MockJoint Class Reference

```
#include <mMockJoint.h>
```

Inheritance diagram for bioscara_hardware_driver::MockJoint:



Collaboration diagram for bioscara_hardware_driver::MockJoint:



Public Member Functions

- [MockJoint](#) (const std::string *name*)
- [err_type_t enable](#) (u_int8_t *driveCurrent*, u_int8_t *holdCurrent*) override

Setup the joint and engages motor, derived classes may override this.
- [err_type_t disable](#) (void) override

disengages the joint motors, derived classes may override this.
- [err_type_t getPosition](#) (float &*pos*) override

get the current joint position in radians or m for cylindrical and prismatic joints respectively. Derived class must override this.
- [err_type_t setPosition](#) (float *pos*) override

get the current joint position in radians or m for cylindrical and prismatic joints respectively. Derived class may override this.

- `err_type_t getVelocity` (float &vel) override
get the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class must override this.
- `err_type_t setVelocity` (float vel) override
Set the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.
- `err_type_t checkOrientation` (float angle=10.0) override
Calls the checkOrientation method of the motor. Checks in which direction the motor is turning. Derived class may override this.
- `err_type_t stop` (void) override
Stops the motor. Derived class may override this.
- `err_type_t getFlags` (void) override
- `bool isHomed` (void) override
Checks the state if the motor is homed.

Public Member Functions inherited from `bioscara_hardware_driver::BaseJoint`

- `BaseJoint` (const std::string name)
Create a `Joint` object.
- `~BaseJoint` (void)
- virtual `err_type_t init` (void)
Initialization, derived classes may override this.
- virtual `err_type_t deinit` (void)
Deinitialization, derived classes may override this.
- virtual `err_type_t home` (float velocity, u_int8_t sensitivity, u_int8_t current)
Blocking implementation to home the joint, derived classes may override this.
- virtual `err_type_t startHoming` (float velocity, u_int8_t sensitivity, u_int8_t current)
non-blocking implementation to home the joint, derived classes may override this.
- virtual `err_type_t postHoming` (void)
perform tasks after a non-blocking homing, derived classes may override this.
- virtual `err_type_t moveSteps` (int32_t steps)
Move full steps. Derived class may override this.
- virtual `err_type_t disableCL` (void)
Disables the Closed-Loop PID Controller Derived class may override this.
- virtual `err_type_t setDriveCurrent` (u_int8_t current)
Set the Drive Current. Derived class may override this.
- virtual `err_type_t setHoldCurrent` (u_int8_t current)
Set the Hold Current. Derived class may override this.
- virtual `err_type_t setBrakeMode` (u_int8_t mode)
Set Brake Mode. Derived class may override this.
- virtual `err_type_t setMaxAcceleration` (float maxAccel)
Set the maximum permitted joint acceleration (and deceleration) in rad/s^2 or m/s^2 for cylindrical and prismatic joints respectively. Derived class may override this.
- virtual `err_type_t setMaxVelocity` (float maxVel)
Set the maximum permitted joint velocity in rad/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.
- virtual `err_type_t enableStallguard` (u_int8_t sensitivity)
Enable encoder stall detection of the joint. Derived class may override this.
- virtual `bool isEnabled` (void)
Checks the state if the motor is enabled.
- virtual `bool isStalled` (void)

Checks if the motor is stalled.

- virtual bool `isBusy` (void)

Checks if the joint controller is busy processing a blocking command.

- virtual `err_type_t getFlags` (`u_int8_t &flags`)
- virtual `stp_reg_t getCurrentBCmd` (void)

get the currently active blocking command

Protected Member Functions

- `err_type_t _home` (float `velocity`, `u_int8_t sensitivity`, `u_int8_t current`)

Call to start the homing sequence of a joint.

Protected Member Functions inherited from bioscara_hardware_driver::BaseJoint

- virtual void `wait_while_busy` (const float `period_ms`)

Blocking loop waiting for BUSY flag to reset.

Private Member Functions

- float `getDeltaT` (`std::chrono::_V2::system_clock::time_point &last_call`, `bool update=true`)

Private Attributes

- float `q` = 0.0
- float `qd` = 0.0
- `std::chrono::_V2::system_clock::time_point last_set_position` = `std::chrono::high_resolution_clock::now()`
- `std::chrono::_V2::system_clock::time_point last_set_velocity` = `last_set_position`
- `std::chrono::_V2::system_clock::time_point async_start_time` = `last_set_position`
- `stp_reg_t op_mode` = `NONE`

Additional Inherited Members

Public Types inherited from bioscara_hardware_driver::BaseJoint

- enum `stp_reg_t` {

`NONE` = 0x00 , `PING` = 0x0f , `SETRPM` = 0x10 , `SETRPM` = 0x11 ,
 `GETDRIVERRPM` = 0x12 , `MOVESTEPS` = 0x13 , `MOVEANGLE` = 0x14 , `MOVETOANGLE` = 0x15 ,
 `GETMOTORSTATE` = 0x16 , `RUNCOTINOUS` = 0x17 , `ANGLEMOVED` = 0x18 , `SETCURRENT` = 0x19 ,
 `SETHOLDCURRENT` = 0x1A , `SETMAXACCELERATION` = 0x1B , `SETMAXDECELERATION` = 0x1C ,
 `SETMAXVELOCITY` = 0x1D ,
 `ENABLESTALLGUARD` = 0x1E , `DISABLESTALLGUARD` = 0x1F , `CLEARSTALL` = 0x20 , `SETBRAKEMODE`
= 0x22 ,
 `ENABLEPID` = 0x23 , `DISABLEPID` = 0x24 , `ENABLECLOSEDLOOP` = 0x25 , `DISABLECLOSEDLOOP`
= 0x26 ,
 `SETCONTROLTHRESHOLD` = 0x27 , `MOVETOEND` = 0x28 , `STOP` = 0x29 , `GETPIDERROR` = 0x2A ,
 `CHECKORIENTATION` = 0x2B , `GETENCODERRPM` = 0x2C , `HOME` = 0x2D , `HOMEOFFSET` = 0x2E }

register and command definitions

Public Attributes inherited from bioscara_hardware_driver::BaseJoint

- std::string `name`

Protected Attributes inherited from bioscara_hardware_driver::BaseJoint

- u_int8_t `flags` = 0b00001100
State flags transmitted with every I2C transaction.
- stp_reg_t `current_b_cmd` = NONE
Keeps track if a blocking command is being executed.

12.10.1 Constructor & Destructor Documentation

12.10.1.1 MockJoint()

```
bioscara_hardware_driver::MockJoint::MockJoint (
    const std::string name )
```

12.10.2 Member Function Documentation

12.10.2.1 _home()

```
err_type_t bioscara_hardware_driver::MockJoint::_home (
    float velocity,
    u_int8_t sensitivity,
    u_int8_t current ) [protected], [virtual]
```

Call to start the homing sequence of a joint.

First the joint will check the motor wiring by executing the `checkOrientation` internally. Then it will set the specified speed until a resistance which drives the PID error above the specified threshold is encountered. At this point the stepper stops and zeros the encoder.

Parameters

<code>velocity</code>	signed velocity in rad/s or m/s. Must be between $1.0 < \text{RAD2DEG}(\text{JOINT2ACTUATOR}(\text{velocity}, \text{reduction}, 0)) / 6 < 250.0$
<code>sensitivity</code>	Encoder pid error threshold 0 to 255.
<code>current</code>	homing current, determines how easy it is to stop the motor and thereby provoke a stall

Returns

0 on success, -1 on communication error, -3 when the motor is not enabled, -5 if the joint is not initialized, -101 if the velocity is zero, -102 if absolute value of the velocity is outside the specified limits.

Implements [bioscara_hardware_driver::BaseJoint](#).

12.10.2.2 checkOrientation()

```
err_type_t bioscara_hardware_driver::MockJoint::checkOrientation (
    float angle = 10.0 ) [override], [virtual]
```

Calls the checkOrientation method of the motor. Checks in which direction the motor is turning. Derived class may override this.

As the orientation check is blocking on the motor, this function returns when the isBusy flag is clear again.

Parameters

<code>angle</code>	degrees how much the motor should turn. A few degrees is sufficient.
--------------------	--

Returns

0 on success, -1 on communication error, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented from [bioscara_hardware_driver::BaseJoint](#).

12.10.2.3 disable()

```
err_type_t bioscara_hardware_driver::MockJoint::disable (
    void ) [override], [virtual]
```

disengages the joint motors, derived classes may override this.

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara_hardware_driver::BaseJoint](#).

12.10.2.4 enable()

```
err_type_t bioscara_hardware_driver::MockJoint::enable (
    u_int8_t driveCurrent,
    u_int8_t holdCurrent ) [override], [virtual]
```

Setup the joint and engages motor, derived classes may override this.

Reimplemented from [bioscara_hardware_driver::BaseJoint](#).

12.10.2.5 getDeltaT()

```
float bioscara_hardware_driver::MockJoint::getDeltaT (
    std::chrono::_V2::system_clock::time_point & last_call,
    bool update = true ) [private]
```

12.10.2.6 getFlags()

```
err_type_t bioscara_hardware_driver::MockJoint::getFlags (
    void ) [override], [virtual]
```

Overload of [BaseJoint::getFlags\(u_int8_t &flags\)](#)

Returns

0 on success, -5 if the joint is not initialized.

Reimplemented from [bioscara_hardware_driver::BaseJoint](#).

12.10.2.7 getPosition()

```
err_type_t bioscara_hardware_driver::MockJoint::getPosition (
    float & pos ) [override], [virtual]
```

get the current joint position in radians or m for cylindrical and prismatic joints respectively. Derived class must override this.

Warning

If the joint is not homed this method does not return an error. Instead `pos` will be 0.0.

Parameters

<code>pos</code>	<input type="text"/>
------------------	----------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Implements [bioscara_hardware_driver::BaseJoint](#).

12.10.2.8 getVelocity()

```
err_type_t bioscara_hardware_driver::MockJoint::getVelocity (
    float & vel ) [override], [virtual]
```

get the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class must override this.

Parameters

<code>vel</code>	<input type="text"/>
------------------	----------------------

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Implements [bioscara_hardware_driver::BaseJoint](#).

12.10.2.9 isHomed()

```
bool bioscara_hardware_driver::MockJoint::isHomed (
    void ) [override], [virtual]
```

Checks the state if the motor is homed.

Reads the internal state flags from the last transmission. If an update is neccessary call [Joint::getFlags\(\)](#) before invoking this function.

Returns

true if the motor is homed, false if not.

Reimplemented from [bioscara_hardware_driver::BaseJoint](#).

12.10.2.10 setPosition()

```
err_type_t bioscara_hardware_driver::MockJoint::setPosition (
    float pos ) [override], [virtual]
```

get the current joint position in radians or m for cylindrical and prismatic joints respectively. Derived class may override this.

Parameters

<i>pos</i>	in rad or m
------------	-------------

Returns

0 on success, -1 on communication error, -2 when not homed, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented from [bioscara_hardware_driver::BaseJoint](#).

12.10.2.11 setVelocity()

```
err_type_t bioscara_hardware_driver::MockJoint::setVelocity (
    float vel ) [override], [virtual]
```

Set the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.

Parameters

<code>vel</code>	
------------------	--

Returns

0 on success, -1 on communication error, -2 when not homed, -3 when the motor is not enabled, -4 when the motor is stalled, -5 if the joint is not initialized.

Reimplemented from [bioscara_hw_driver::BaseJoint](#).

12.10.2.12 stop()

```
err_type_t bioscara_hw_driver::MockJoint::stop (
    void ) [override], [virtual]
```

Stops the motor. Derived class may override this.

Stops the motor by setting the maximum velocity to zero and the position setpoint to the current position

Returns

0 on success, -1 on communication error, -5 if the joint is not initialized.

Reimplemented from [bioscara_hw_driver::BaseJoint](#).

12.10.3 Member Data Documentation**12.10.3.1 async_start_time**

```
std::chrono::_V2::system_clock::time_point bioscara_hw_driver::MockJoint::async_start_<-
time = last_set_position [private]
```

12.10.3.2 last_set_position

```
std::chrono::_V2::system_clock::time_point bioscara_hw_driver::MockJoint::last_set_<-
position = std::chrono::high_resolution_clock::now() [private]
```

12.10.3.3 last_set_velocity

```
std::chrono::_V2::system_clock::time_point bioscara_hw_driver::MockJoint::last_set_<-
velocity = last_set_position [private]
```

12.10.3.4 op_mode

```
stp_reg_t bioscara_hw_driver::MockJoint::op_mode = NONE [private]
```

12.10.3.5 q

```
float bioscara.hardware_driver::MockJoint::q = 0.0 [private]
```

12.10.3.6 qd

```
float bioscara.hardware_driver::MockJoint::qd = 0.0 [private]
```

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

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/[mMockJoint.h](#)
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/[mMockJoint.cpp](#)

12.11 MovMax Class Reference

```
#include <filters.h>
```

Public Member Functions

- [MovMax](#) (float *windowSize*)
- float [updateState](#) (float *u*)

Protected Attributes

- unsigned int *M* = 200
- float * *cb_data*
- unsigned int *cb_index*

12.11.1 Constructor & Destructor Documentation

12.11.1.1 [MovMax\(\)](#)

```
MovMax::MovMax (
    float windowSize ) [inline]
```

12.11.2 Member Function Documentation

12.11.2.1 [updateState\(\)](#)

```
float MovMax::updateState (
    float u ) [inline]
```

12.11.3 Member Data Documentation

12.11.3.1 cb_data

```
float* MovMax::cb_data [protected]
```

12.11.3.2 cb_index

```
unsigned int MovMax::cb_index [protected]
```

12.11.3.3 M

```
unsigned int MovMax::M = 200 [protected]
```

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

- Arduino/joint/[filters.h](#)

12.12 RPI_PWM Class Reference

PWM class for the Raspberry PI 4 and 5.

```
#include <uPWM.h>
```

Public Member Functions

- int [start](#) (int channel, int frequency, float duty_cycle=0, int chip=2)
- void [stop](#) ()
- [~RPI_PWM](#) ()
- int [setDutyCycle](#) (float v) const

Private Member Functions

- void [setPeriod](#) (int ns) const
- int [setDutyCycleNS](#) (int ns) const
- void [enable](#) () const
- void [disable](#) () const
- int [writeSYS](#) (std::string filename, int value) const

Private Attributes

- int [per](#) = 0
- std::string [chippath](#)
- std::string [pwmpath](#)

12.12.1 Detailed Description

PWM class for the Raspberry PI 4 and 5.

12.12.2 Constructor & Destructor Documentation

12.12.2.1 ~RPI_PWM()

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

12.12.3 Member Function Documentation

12.12.3.1 disable()

```
void RPI_PWM::disable ( ) const [inline], [private]
```

12.12.3.2 enable()

```
void RPI_PWM::enable ( ) const [inline], [private]
```

12.12.3.3 setDutyCycle()

```
int RPI_PWM::setDutyCycle (
    float v ) const [inline]
```

Sets the duty cycle in percent 0 - 100.

Parameters

v	The duty cycle in percent.
---	----------------------------

Returns

>0 on success and -1 after an error.

12.12.3.4 setDutyCycleNS()

```
int RPI_PWM::setDutyCycleNS (
    int ns ) const [inline], [private]
```

12.12.3.5 setPeriod()

```
void RPI_PWM::setPeriod (
    int ns ) const [inline], [private]
```

12.12.3.6 start()

```
int RPI_PWM::start (
    int channel,
    int frequency,
    float duty_cycle = 0,
    int chip = 2 ) [inline]
```

Starts the PWM

Parameters

<i>channel</i>	The GPIO channel which is 2 or 3 for the RPI5
<i>frequency</i>	The PWM frequency
<i>duty_cycle</i>	The initial duty cycle of the PWM (default 0)
<i>chip</i>	The chip number (for RPI5 it's 2)

Returns

>0 on success and -1 if an error has happened.

12.12.3.7 stop()

```
void RPI_PWM::stop () [inline]
```

Stops the PWM

12.12.3.8 writeSYS()

```
int RPI_PWM::writeSYS (
    std::string filename,
    int value ) const [inline], [private]
```

12.12.4 Member Data Documentation

12.12.4.1 chippath

```
std::string RPI_PWM::chippath [private]
```

12.12.4.2 per

```
int RPI_PWM::per = 0 [private]
```

12.12.4.3 pwmpath

```
std::string RPI_PWM::pwmpath [private]
```

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

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/uPWM.h

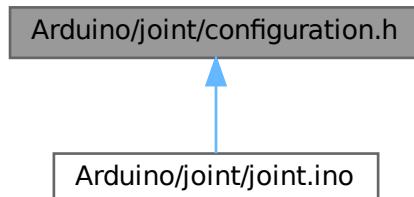
Chapter 13

File Documentation

13.1 Arduino/joint/configuration.h File Reference

Configuration definitions for Joint 1 to Joint 4.

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



Macros

- `#define ADR 0x11`
I2C adress of joint n is 0x1n.
- `#define MAXACCEL 10000`
Maximum acceleration in steps/s^2. Can be set for each joint depending on inertia. If set to high stalls might trigger since PID error grows too large.
- `#define MAXVEL 800`
Maximum velocity in steps/s. Can be set for each joint. If set to high stalls might trigger since PID error grows too large.

13.1.1 Detailed Description

Configuration definitions for Joint 1 to Joint 4.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

This file shall be included AFTER one of J1, J2, J3 or J4 have been defined.

13.1.2 Macro Definition Documentation

13.1.2.1 ADR

```
#define ADR 0x11
```

I2C adress of joint n is 0x1n.

13.1.2.2 MAXACCEL

```
#define MAXACCEL 10000
```

Maximum acceleration in steps/s². Can be set for each joint depending on inertia. If set to high stalls might trigger since PID error grows too large.

13.1.2.3 MAXVEL

```
#define MAXVEL 800
```

Maximum velocity in steps/s. Can be set for each joint. If set to high stalls might trigger since PID error grows too large.

13.2 configuration.h

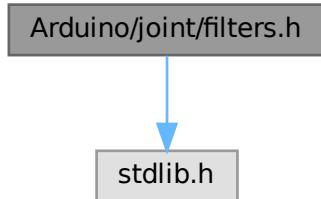
[Go to the documentation of this file.](#)

```
00001  
00014 #ifndef CONFIGURATION_H  
00015 #define CONFIGURATION_H  
00016  
00017 #if defined(J1)  
00019 #define ADR 0x11  
00020 #define MAXACCEL 0  
00021 #define MAXVEL 0  
00022 #define STALL_WINDOW_B1 12  
00023 #define STALL_WINDOW_B2 450  
00024 #define STALL_WINDOW_OFFSET 90.0  
00025 #define STALL_SLOPE 0.0  
00026  
00027 #elif defined(J2)  
00028 #define ADR 0x12  
00029 #define MAXACCEL 0  
00030 #define MAXVEL 0  
00031 #define STALL_WINDOW_B1 12  
00032 #define STALL_WINDOW_B2 450  
00033 #define STALL_WINDOW_OFFSET 90.0  
00034 #define STALL_SLOPE 0.0  
00035  
00036 #elif defined(J3)  
00037 #define ADR 0x13  
00038 #define MAXACCEL 0  
00039 #define MAXVEL 0  
00040 #define STALL_WINDOW_B1 12  
00041 #define STALL_WINDOW_B2 450  
00042 #define STALL_WINDOW_OFFSET 90.0  
00043 #define STALL_SLOPE 0.0  
00044  
00045 #elif defined(J4)  
00046 #define ADR 0x14  
00047 #define MAXACCEL 0  
00048 #define MAXVEL 0  
00049 #define STALL_WINDOW_B1 12  
00050 #define STALL_WINDOW_B2 450  
00051 #define STALL_WINDOW_OFFSET 90.0  
00052 #define STALL_SLOPE 0.0  
00053 #else  
00054  
00055 /* Below only defined for documentation */  
00059 #define ADR 0x11  
00060  
00065 #define MAXACCEL 10000  
00066  
00071 #define MAXVEL 800  
00072 #error "No Joint has been defined. Define one of 'JX' where X 1,2,3,4"  
00073 #endif  
00074  
00075 #endif
```

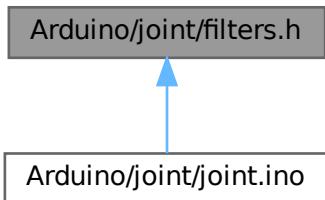
13.3 Arduino/joint/filters.h File Reference

Helper classes for FIR and IIR filters.

```
#include <stdlib.h>
Include dependency graph for filters.h:
```



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



Classes

- class [Lowpass](#)
- class [MovMax](#)

13.3.1 Detailed Description

Helper classes for FIR and IIR filters.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

This file contains time series filters that can be used to filter data.

13.4 filters.h

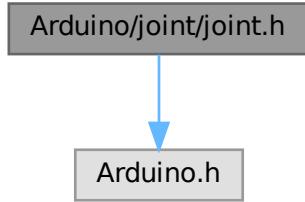
[Go to the documentation of this file.](#)

```
00001
00013 #include <stdlib.h>
00014
00015 class Lowpass {
00016
00017 protected:
00018     float K, Ts, tau, x;
00019
00020 public:
00021
00029     Lowpass(float gain = 1, float sampleTime = 0.1, float timeconstant = 1.0) {
00030         this->K = gain;
00031         this->Ts = sampleTime;
00032         this->tau = timeconstant;
00033         x = 0.0;
00034     }
00035
00036     float updateState(float u) {
00037         x = (1 - (Ts / tau)) * x + K * (Ts / tau) * u;
00038         return x;
00039     }
00040
00041     void resetState(void) {
00042         x = 0.0;
00043     }
00044 };
00045
00046 class MovMax {
00047
00048 protected:
00049     unsigned int M = 200; // Window Size
00050
00051     float *cb_data;
00052     unsigned int cb_index;
00053
00054
00055 public:
00056
00057     MovMax(float windowSize)
00058         : M(windowSize), cb_index(0), cb_data(0) {
00059
00060         cb_data = (float *)malloc(windowSize * sizeof(float)); // allocate memory for buffer
00061     }
00062
00063     float updateState(float u) {
00064
00065         cb_data[cb_index] = u;
00066         cb_index = (cb_index + 1) % M;
00067
00068         float max = 0;
00069         for (size_t i = 0; i < M; i++) {
00070             if (cb_data[i] > max) {
00071                 max = cb_data[i];
00072             }
00073         }
00074     }
00075
00076     return max;
00077 }
00078 };
```

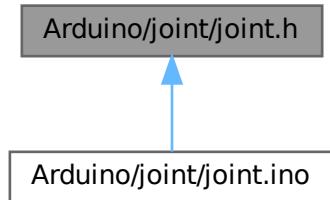
13.5 Arduino/joint/joint.h File Reference

joint firmware header

```
#include <Arduino.h>
Include dependency graph for joint.h:
```



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



Macros

- `#define ACK 'O'`
- `#define NACK 'N'`
- `#define MAX_BUFFER 4`

Maximum size of I2C Payload in bytes.
- `#define RFLAGS_SIZE 1`

Size of the return flags in bytes.
- `#define DUMP_BUFFER(buffer, size)`

Macro to dump a buffer to the serial console.

Enumerations

- enum `stp_reg_t` {
 `PING = 0x0f` , `SETUP = 0x10` , `SETRPM = 0x11` , `GETDRIVERRPM = 0x12` ,
 `MOVESTEPS = 0x13` , `MOVEANGLE = 0x14` , `MOVETOANGLE = 0x15` , `GETMOTORSTATE = 0x16` ,
 `RUNCOTINOUS = 0x17` , `ANGLEMOVED = 0x18` , `SETCURRENT = 0x19` , `SETHOLDCURRENT = 0x1A` ,
 `SETMAXACCELERATION = 0x1B` , `SETMAXDECELERATION = 0x1C` , `SETMAXVELOCITY = 0x1D` ,
 `ENABLESTALLGUARD = 0x1E` ,
 }

```

DISABLESTALLGUARD = 0x1F , CLEARSTALL = 0x20 , SETBRAKEMODE = 0x22 , ENABLEPID = 0x23 ,
DISABLEPID = 0x24 , ENABLCLOSEDLOOP = 0x25 , DISABLECLOSEDLOOP = 0x26 , SETCONTROLTHRESHOLD
= 0x27 ,
MOVETOEND = 0x28 , STOP = 0x29 , GETPIDERROR = 0x2A , CHECKORIENTATION = 0x2B ,
GETENCODERRPM = 0x2C , HOME = 0x2D , HOMEOFFSET = 0x2E }

register and command definitions

```

Functions

- template<typename T >
void **readValue** (T &val, uint8_t *rxBuf, size_t **rx_length**)
Reads a value from a buffer to a value of the specified type.
- template<typename T >
int **writeValue** (const T val, uint8_t *txBuf, size_t &**tx_length**)
Writes a value of the specified type to a buffer.

13.5.1 Detailed Description

joint firmware header

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

This file contains definitions and macros for the joint firmware.

13.5.2 Macro Definition Documentation

13.5.2.1 ACK

```
#define ACK 'O'
```

13.5.2.2 DUMP_BUFFER

```
#define DUMP_BUFFER(
    buffer,
    size )
```

Value:

```
{
    Serial.print("Buffer dump: ");
    \\
    for (size_t i = 0; i < size; i++) \\
    {
        Serial.print(buffer[i], HEX);
        Serial.print(" ");
    }
    Serial.println();
}
```

Macro to dump a buffer to the serial console.

Parameters

<i>buffer</i>	pointer to a buffer to dump to the console
<i>size</i>	number of bytes to dump

13.5.2.3 MAX_BUFFER

```
#define MAX_BUFFER 4
```

Maximum size of I2C Payload in bytes.

4 bytes used to transmit floats and int32_t

13.5.2.4 NACK

```
#define NACK 'N'
```

13.5.2.5 RFLAGS_SIZE

```
#define RFLAGS_SIZE 1
```

Size of the return flags in bytes.

Only one byte used and hence set to 1.

13.5.3 Enumeration Type Documentation**13.5.3.1 stp_reg_t**

```
enum stp_reg_t
```

register and command definitions

a register can be read (R) or written (W), each register has a size in bytes. The payload can be split into multiple values or just be a single value. Note that not all functions are implemented.

Enumerator

PING	R; Size: 1; [(char) ACK].
SETUP	W; Size: 2; [(uint8) holdCurrent, (uint8) driveCurrent].
SETRPM	W; Size: 4; [(float) RPM].
GETDRIVERRPM	
MOVESTEPS	W; Size: 4; [(int32) steps].
MOVEANGLE	
MOVETOANGLE	W; Size: 4; [(float) degrees].
GETMOTORSTATE	
RUNCOTINOUS	
ANGLEMOVED	R; Size: 4; [(float) degrees].

Enumerator

SETCURRENT	W; Size: 1; [(uint8) driveCurrent].
SETHOLDCURRENT	W; Size: 1; [(uint8) holdCurrent].
SETMAXACCELERATION	W; Size: 4; [(float) deg/s^2].
SETMAXDECELERATION	
SETMAXVELOCITY	W; Size: 4; [(float) deg/s].
ENABLESTALLGUARD	W; Size: 1; [(uint8) threshold].
DISABLESTALLGUARD	
CLEARSTALL	
SETBRAKEMODE	W; Size: 1; [(uint8) mode].
ENABLEPID	
DISABLEPID	
ENABLECLOSEDLOOP	
DISABLECLOSEDLOOP	W; Size: 1; [(uint8) 0].
SETCONTROLTHRESHOLD	
MOVETOEND	
STOP	W; Size: 1; [(uint8) mode].
GETPIDERROR	
CHECKORIENTATION	W; Size: 4; [(float) degrees].
GETENCODERRPM	R; Size: 4; [(float) RPM].
HOME	W; Size: 4; [(uint8) current, (uint8) sensitivity, (uint8) speed, (uint8) direction].
HOMEOFFSET	R/W; Size: 4; [(float) -].

13.5.4 Function Documentation**13.5.4.1 `readValue()`**

```
template<typename T >
void readValue (
    T & val,
    uint8_t * rxBuf,
    size_t rx_length )
```

Reads a value from a buffer to a value of the specified type.

Parameters

<i>val</i>	Reference to output variable
<i>rxBuf</i>	Buffer to read value from
<i>rx_length</i>	Length of the buffer

13.5.4.2 `writeValue()`

```
template<typename T >
int writeValue (
    const T val,
```

```
    uint8_t * txBuf,
    size_t & tx_length )
```

Writes a value of the specified type to a buffer.

Parameters

<i>val</i>	Reference to input variable
<i>txBuf</i>	pointer to tx buffer
<i>tx_length</i>	Length of the buffer returne

Returns

0 On success

13.6 joint.h

[Go to the documentation of this file.](#)

```
00001
00014 #ifndef JOINT_H
00015 #define JOINT_H
00016 #include <Arduino.h>
00017
00018 #define ACK 'O'
00019 #define NACK 'N'
00020
00026 #define MAX_BUFFER 4 // Bytes
00027
00033 #define RFLAGS_SIZE 1
00034
00041 #define DUMP_BUFFER(buffer, size) \
00042 { \
00043     Serial.print("Buffer dump: "); \
00044     for (size_t i = 0; i < size; i++) \
00045     { \
00046         Serial.print(buffer[i], HEX); \
00047         Serial.print(" "); \
00048     } \
00049     Serial.println(); \
00050 }
00051
00060 enum stp_reg_t
00061 {
00062     PING = 0x0f,
00063     SETUP = 0x10,
00064     SETRPM = 0x11,
00065     GETDRIVERRPM = 0x12,
00066     MOVESTEPS = 0x13,
00067     MOVEANGLE = 0x14,
00068     MOVETOANGLE = 0x15,
00069     GETMOTORSTATE = 0x16,
00070     RUNCOTINOUS = 0x17,
00071     ANGLEMOVED = 0x18,
00072     SETCURRENT = 0x19,
00073     SETHOLDCURRENT = 0x1A,
00074     SETMAXACCELERATION = 0x1B,
00075     SETMAXDECELERATION = 0x1C,
00076     SETMAXVELOCITY = 0x1D,
00077     ENABLESTALLGUARD = 0x1E,
00078     DISABLESTALLGUARD = 0x1F,
00079     CLEARSTALL = 0x20,
00080     SETBRAKEMODE = 0x22,
00081     ENABLEPID = 0x23,
00082     DISABLEPID = 0x24,
00083     ENABLECLOSEDLOOP = 0x25,
00084     DISABLECLOSEDLOOP = 0x26,
00085     SETCONTROLTHRESHOLD = 0x27,
00086     MOVETOEND = 0x28,
00087     STOP = 0x29,
00088     GETPIDERROR = 0x2A,
00089     CHECKORIENTATION = 0x2B,
00090     GETENCODERRPM = 0x2C,
```

```

00091     HOME = 0x2D,
00092     HOMEOFFSET = 0x2E,
00093 };
00094
00101 template <typename T>
00102 void readValue(T &val, uint8_t *rxBuf, size_t rx_length)
00103 {
00104     memcpy(&val, rxBuf, rx_length);
00105 }
00106
00114 template <typename T>
00115 int writeValue(const T val, uint8_t *txBuf, size_t &tx_length)
00116 {
00117     tx_length = sizeof(T);
00118     memcpy(txBuf, &val, tx_length);
00119     return 0;
00120 }
00121
00122 #endif

```

13.7 Arduino/joint/joint.ino File Reference

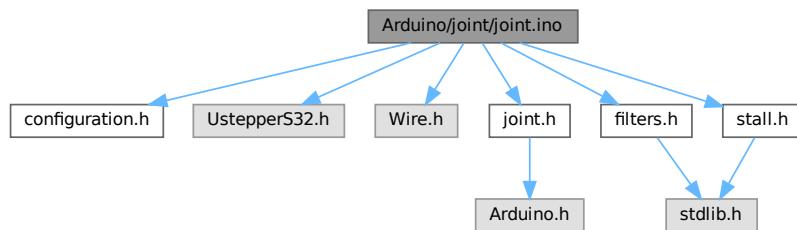
joint firmware

```

#include "configuration.h"
#include <UstepperS32.h>
#include <Wire.h>
#include "joint.h"
#include "filters.h"
#include "stall.h"

```

Include dependency graph for joint.ino:



Macros

- **#define J3**

Define either joint that is to be flashed.

Functions

- void **blocking_handler** (uint8_t reg)
Handles commands received via I2C.
- void **non_blocking_handler** (uint8_t reg)
Handles read request received via I2C.
- void **receiveEvent** (int n)
I2C receive event Handler.

- void [requestEvent \(\)](#)
I2C request event Handler.
- void [setup \(void\)](#)
Setup Peripherals.
- void [loop \(void\)](#)
Main loop.

Variables

- UstepperS32 [stepper](#)
- uint8_t [reg](#) = 0
- uint8_t [rx_buf](#) [MAX_BUFFER] = { 0 }
- uint8_t [tx_buf](#) [MAX_BUFFER+RFLAGS_SIZE] = { 0 }
- bool [rx_data_ready](#) = 0
- size_t [tx_length](#) = 0
- size_t [rx_length](#) = 0

13.7.1 Detailed Description

joint firmware

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

This file contains the joint firmware.

13.7.2 Macro Definition Documentation

13.7.2.1 J3

```
#define J3
```

Define either joint that is to be flashed.

Define either J1, J2, J3 or J4 and subsequently include [configuration.h](#)

13.7.3 Function Documentation

13.7.3.1 blocking_handler()

```
void blocking_handler (
    uint8_t reg )
```

Handles commands received via I2C.

Warning

This is a blocking function which may take some time to execute. This function must not be called from an ISR or callback! Call from main loop instead.

The registers handled in this handler are those whose implementation can take time and can thereby not be called directly from the request handler.

Parameters

<i>reg</i>	command that should be executed.
------------	----------------------------------

13.7.3.2 loop()

```
void loop (
    void )
```

Main loop.

Executes the following:

1. if isStallguardEnabled: compares stepper.getPidError() with stallguardThreshold and sets isStalled flag.
2. if rx_data_ready: set isBusy flag to indicate device is busy. Invoke blocking_handler. Clear isBusy flag to indicate device is no longer busy

13.7.3.3 non_blocking_handler()

```
void non_blocking_handler (
    uint8_t reg )
```

Handles read request received via I2C.

Can be invoked from the I2C ISR since reads from the stepper are non-blocking. Also Handling reads and the subsequent wire.write(), did not work from the main loop.

Parameters

<i>reg</i>	command to execute/register to read.
------------	--------------------------------------

13.7.3.4 receiveEvent()

```
void receiveEvent (
    int n )
```

I2C receive event Handler.

Reads the content of the received message. Saves the register so it can be used in the main loop. If the master invokes the read() function the message contains only the register byte and no payload. If the master invokes the write() the message has a payload of appropriate size for the command. Every I2C transaction starts with a receive event when the command is sent and is immediately followed by a request since at minimum the flags need to be transmitted back. This means that the receive handler and request handler are always executed sequentially. The main loop is not executed since both handlers are ISRs. For a read request the message looks like this:

< [REG]
> [TXBUFn]...[TXBUF2][TXBUF1][TXBUF0][FLAGS]

For a command the message looks like this:

< [REG][RXBUFn]...[RXBUF2][RXBUF1][RXBUF0]
> [FLAGS]

The payload is read into the rx_buf, rx_length is set to the payload length.

Parameters

<i>n</i>	the number of bytes read from the controller device: MAX_BUFFER
----------	---

13.7.3.5 requestEvent()

```
void requestEvent ( )
```

I2C request event Handler.

Sends the response data to the master. Every transaction begins with a receive event. The request event is always triggered since at a minimum the status flags are returned to the master. Hence this function is only invoked after the [receiveEvent\(\)](#) handler has been called. The function calls the [non_blocking_handler\(\)](#) which is non-blocking. Since most Ustepper functions are non-blocking as they just read/write registers to the stepper driver/encoder they can be handled directly in the ISR. The [non_blocking_handler\(\)](#) populates the tx_buf with relevant data, the current state flags are appended to the tx_buf and then it is send to the master.

13.7.3.6 setup()

```
void setup (
    void )
```

Setup Peripherals.

Setup I2C with the address ADR, and begin Serial for debugging with baudrate 9600.

13.7.4 Variable Documentation

13.7.4.1 reg

```
uint8_t reg = 0
```

13.7.4.2 rx_buf

```
uint8_t rx_buf[MAX_BUFFER] = { 0 }
```

13.7.4.3 rx_data_ready

```
bool rx_data_ready = 0
```

13.7.4.4 rx_length

```
size_t rx_length = 0
```

13.7.4.5 stepper

```
UstepperS32 stepper
```

13.7.4.6 tx_buf

```
uint8_t tx_buf[MAX_BUFFER+RFLAGS_SIZE] = { 0 }
```

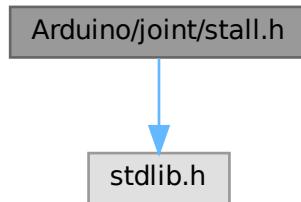
13.7.4.7 tx_length

```
size_t tx_length = 0
```

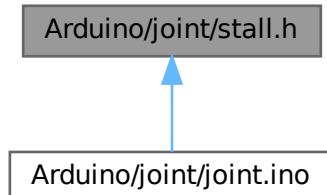
13.8 Arduino/joint/stall.h File Reference

Helper functions for improved stall detection.

```
#include <stdlib.h>
Include dependency graph for stall.h:
```



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



Functions

- float [stall_threshold](#) (float qd_rad, float offset)
computes the speed adaptive threshold.

13.8.1 Detailed Description

Helper functions for improved stall detection.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

13.8.2 Function Documentation

13.8.2.1 stall_threshold()

```
float stall_threshold (
    float qd_rad,
    float offset )
```

computes the speed adaptive threshold.

Parameters

<code>qd_rad</code>	speed in rad/s. Should be measured speed because set speed is only available in velocity mode.
---------------------	--

13.9 stall.h

[Go to the documentation of this file.](#)

```
00001
00012 #include <stdlib.h>
00013
00020 float stall_threshold(float qd_rad, float offset){
00021     /* y = ax + b */
00022     float a = STALL_SLOPE;
00023     float b = offset;
00024     if((abs(qd_rad) >= STALL_WINDOW_B1) && (abs(qd_rad) <= STALL_WINDOW_B2)) {
00025         b += STALL_WINDOW_OFFSET;
00026     }
00027     return a*qd_rad+b;
00028 }
```

13.10 docs/DOCS_README.md File Reference

13.11 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_↔bringup/launch/bioscara.launch.py File Reference

Namespaces

- namespace [bioscara](#)

Functions

- bioscara.generate_launch_description ()

**13.12 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_←
bringup/launch/test_joint_trajectory_controller.launch.py File
Reference****Namespaces**

- namespace test_joint_trajectory_controller

Functions

- test_joint_trajectory_controller.generate_launch_description ()

**13.13 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup/←
README.md File Reference****13.14 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_controllers/←
README.md File Reference****13.15 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/←
README.md File Reference****13.16 ROS2/ros2_scara_ws/src/dalsa_bioscara/README.md File
Reference****13.17 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_←
description/bioscara_description/__init__.py File Reference****13.18 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_←
description/launch/display.launch.py File Reference****Namespaces**

- namespace display

Functions

- display.generate_launch_description ()

13.19 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/launch/gazebo.launch.py File Reference

Namespaces

- namespace `gazebo`

Functions

- `gazebo.generate_launch_description()`

13.20 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/setup.py File Reference

Namespaces

- namespace `setup`

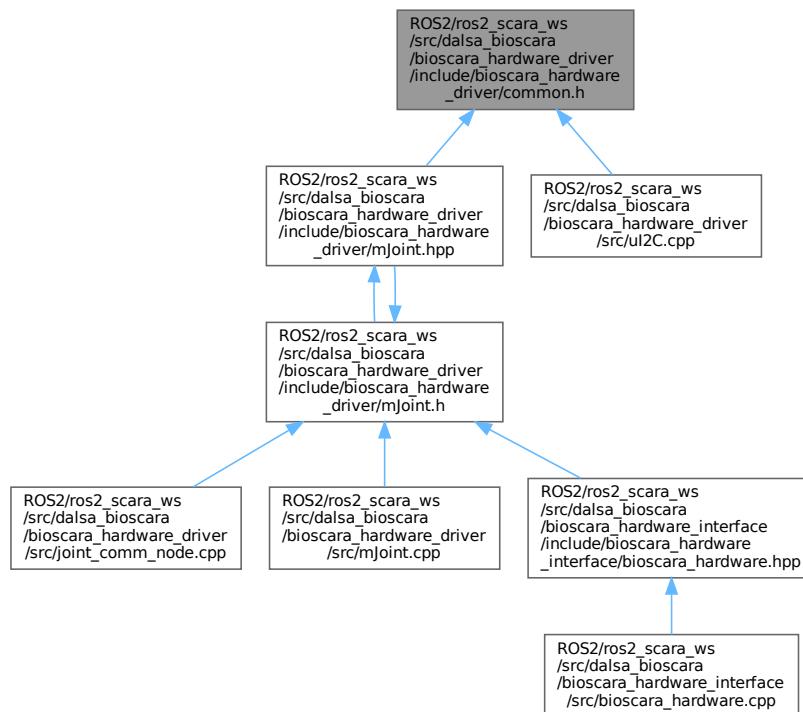
Variables

- str `setup.package_name` = 'bioscara_description'
- `setup.name`
- `setup.version`
- `setup.packages`
- `setup.data_files`
- `setup.install_requires`
- `setup.zip_safe`
- `setup.maintainer`
- `setup.maintainer_email`
- `setup.description`
- `setup.license`
- `setup.tests_require`
- `setup.entry_points`

13.21 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/common.h File Reference

A file containing utility macros and functions.

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



Macros

- `#define DUMP_BUFFER(buffer, size)`
Macro to dump a buffer to cout.

13.21.1 Detailed Description

A file containing utility macros and functions.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

13.21.2 Macro Definition Documentation

13.21.2.1 DUMP_BUFFER

```
#define DUMP_BUFFER(
    buffer,
    size )
```

Value:

```
{
    std::cout << "Buffer dump: ";
    for (size_t i = 0; i < size; i++) \
    {
        printf("%#x ", buffer[i]);
    }
    std::cout << std::endl;
}
```

Macro to dump a buffer to cout.

Parameters

<i>buffer</i>	pointer to a buffer to dump to the console
<i>size</i>	number of bytes to dump

13.22 common.h

[Go to the documentation of this file.](#)

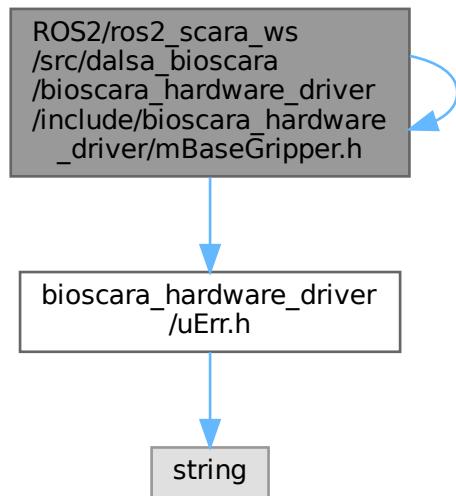
```
00001
00011 #ifndef COMMON_H
00012 #define COMMON_H
00013
00020 #define DUMP_BUFFER(buffer, size)
00021 {
00022     std::cout << "Buffer dump: ";
00023     for (size_t i = 0; i < size; i++) \
00024     {
00025         printf("%#x ", buffer[i]);
00026     }
00027     std::cout << std::endl;
00028 }
00029 #endif // COMMON_H
```

13.23 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mBaseGripper.h File Reference

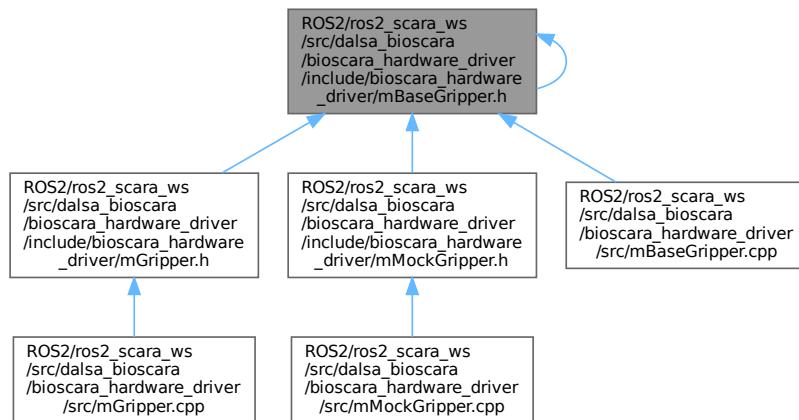
File containing the BaseGripper class.

```
#include "bioscara_hardware_driver/mBaseGripper.h"
#include "bioscara_hardware_driver/uErr.h"
```

Include dependency graph for mBaseGripper.h:



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



Classes

- class [bioscara_hardware_driver::BaseGripper](#)

Namespaces

- namespace [bioscara_hardware_driver](#)
Generic BaseGripper object to interact with the robot gripper.

13.23.1 Detailed Description

File containing the BaseGripper class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

Dont include this file directly, instead use one of the derived classes.

13.24 mBaseGripper.h

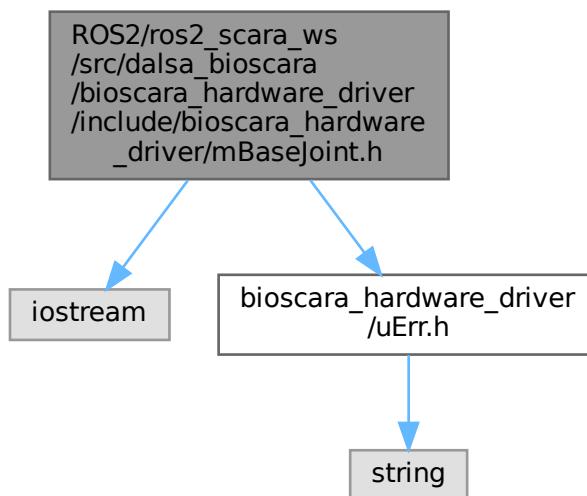
[Go to the documentation of this file.](#)

```
00001
00013 #ifndef MBASEGRIPPER_H
00014 #define MBASEGRIPPER_H
00015 #include "bioscara_hardware_driver/mBaseGripper.h"
00016 #include "bioscara_hardware_driver/uErr.h"
00017
00058 namespace bioscara_hardware_driver
00059 {
00060     class BaseGripper
00061     {
00062         public:
00063             BaseGripper(void);
00064
00070             virtual err_type_t init(void);
00071
00077             virtual err_type_t_deinit(void);
00078
00084             virtual err_type_t enable(void);
00085
00091             virtual err_type_t disable(void);
00092
00099             virtual err_type_t setPosition(float width);
00100
00106             virtual err_type_t setServoPosition(float angle);
00107
00113             virtual void setReduction(float reduction);
00114
00118             virtual void setOffset(float offset);
00119
00120         protected:
00121         private:
00122     };
00123 }
00124 #endif // MBASEGRIPPER_H
```

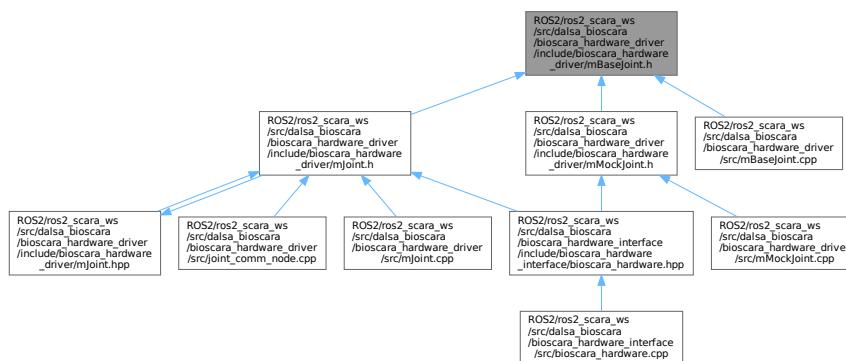
13.25 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mBaseJoint.h File Reference

File including the BaseJoint class.

```
#include <iostream>
#include "bioscara_hardware_driver/uErr.h"
Include dependency graph for mBaseJoint.h:
```



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



Classes

- class [bioscara_hardware_driver::BaseJoint](#)
TODO.

Namespaces

- namespace bioscara.hardware_driver

Generic *BaseGripper* object to interact with the robot gripper.

13.25.1 Detailed Description

File including the BaseJoint class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-29

Copyright

Copyright (c) 2025

13.26 mBaseJoint.h

[Go to the documentation of this file.](#)

```
00001  
00012 #ifndef MBASEJOINT_H  
00013 #define MBASEJOINT_H  
00014  
00015 #include <iostream>  
00016 #include "bioscara.hardware_driver/uErr.h"  
00017  
00018 namespace bioscara.hardware_driver  
00019 {  
00020     class BaseJoint  
00021     {  
00022         public:  
00023             enum stp_reg_t  
00024             {  
00025                 NONE = 0x00,  
00026                 PING = 0x0f,  
00027                 SETUP = 0x10,  
00028                 SETRPM = 0x11,  
00029                 GETDRIVERRPM = 0x12,  
00030                 MOVESTEPS = 0x13,  
00031                 MOVEANGLE = 0x14,  
00032                 MOVETOANGLE = 0x15,  
00033                 GETMOTORSTATE = 0x16,  
00034                 RUNCOTINOUS = 0x17,  
00035                 ANGLEMOVED = 0x18,  
00036                 SETCURRENT = 0x19,  
00037                 SETHOLDCURRENT = 0x1A,  
00038                 SETMAXACCELERATION = 0x1B,  
00039                 SETMAXDECELERATION = 0x1C,  
00040                 SETMAXVELOCITY = 0x1D,  
00041                 ENABLESTALLGUARD = 0x1E,  
00042                 DISABLESTALLGUARD = 0x1F,  
00043                 CLEARSTALL = 0x20,  
00044                 SETBRAKEMODE = 0x22,  
00045                 ENABLEPID = 0x23,
```

```

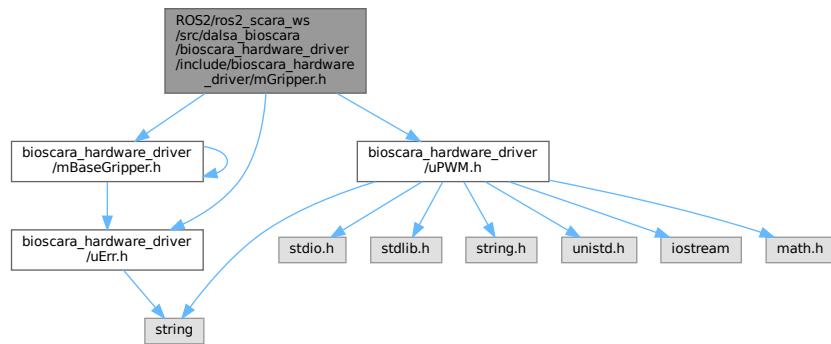
00059     DISABLEPID = 0x24,
00060     ENABLECLOSEDLOOP = 0x25,
00061     DISABLECLOSEDLOOP = 0x26,
00062     SETCONTROLTRESHOLD = 0x27,
00063     MOVETOEND = 0x28,
00064     STOP = 0x29,
00065     GETPIDERROR = 0x2A,
00066     CHECKORIENTATION = 0x2B,
00067     GETENCODERRPM = 0x2C,
00068     HOME = 0x2D,
00069     HOMEOFFSET = 0x2E,
00070 };
00071
00072     BaseJoint(const std::string name);
00073     ~BaseJoint(void);
00074
00075     virtual err_type_t init(void);
00076
00077     virtual err_type_t_deinit(void);
00078
00079     virtual err_type_t enable(u_int8_t driveCurrent, u_int8_t holdCurrent);
00080
00081     virtual err_type_t disable(void);
00082
00083     virtual err_type_t home(float velocity, u_int8_t sensitivity, u_int8_t current);
00084
00085     virtual err_type_t startHoming(float velocity, u_int8_t sensitivity, u_int8_t current);
00086
00087     virtual err_type_t postHoming(void);
00088
00089     virtual err_type_t getPosition(float &pos) = 0;
00090
00091     virtual err_type_t setPosition(float pos);
00092
00093     virtual err_type_t moveSteps(int32_t steps);
00094
00095     virtual err_type_t getVelocity(float &vel) = 0;
00096
00097     virtual err_type_t setVelocity(float vel);
00098
00099     virtual err_type_t checkOrientation(float angle = 10.0);
00100
00101     virtual err_type_t stop(void);
00102
00103     virtual err_type_t disableCL(void);
00104
00105     virtual err_type_t setDriveCurrent(u_int8_t current);
00106
00107     virtual err_type_t setHoldCurrent(u_int8_t current);
00108
00109     virtual err_type_t setBrakeMode(u_int8_t mode);
00110
00111     virtual err_type_t setMaxAcceleration(float maxAccel);
00112
00113     virtual err_type_t setMaxVelocity(float maxVel);
00114
00115     virtual err_type_t enableStallguard(u_int8_t sensitivity);
00116
00117     virtual bool isHomed(void);
00118
00119     virtual bool isEnabled(void);
00120
00121     virtual bool isStalled(void);
00122
00123     virtual bool isBusy(void);
00124
00125     virtual err_type_t getFlags(u_int8_t &flags);
00126
00127     virtual err_type_t getFlags(void);
00128
00129     virtual stp_reg_t getCurrentBCmd(void);
00130
00131     std::string name;
00132
00133 protected:
00134     virtual void wait_while_busy(const float period_ms);
00135
00136     virtual err_type_t _home(float velocity, u_int8_t sensitivity, u_int8_t current) = 0;
00137
00138     u_int8_t flags = 0b00001100;
00139
00140     stp_reg_t current_b_cmd = NONE;
00141
00142 private:
00143 };
00144
00145 #endif

```

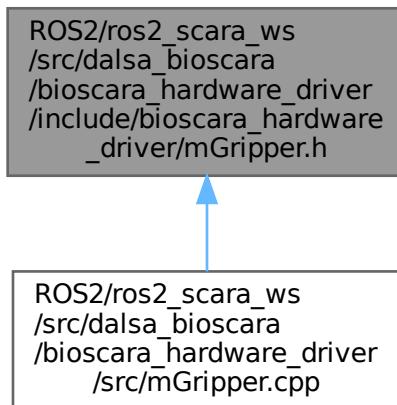
13.27 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mGripper.h File Reference

File containing the Gripper class.

```
#include "bioscara_hardware_driver/mBaseGripper.h"
#include "bioscara_hardware_driver/uPWM.h"
#include "bioscara_hardware_driver/uErr.h"
Include dependency graph for mGripper.h:
```



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



Classes

- class [bioscara_hardware_driver::Gripper](#)

Namespaces

- namespace bioscara.hardware_driver
Generic BaseGripper object to interact with the robot gripper.

13.27.1 Detailed Description

File containing the Gripper class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

Include this file for API functions to interact with the gripper.

13.28 mGripper.h

[Go to the documentation of this file.](#)

```

00001
00013 #ifndef MGRIPPER_H
00014 #define MGRIPPER_H
00015 #include "bioscara.hardware_driver/mBaseGripper.h"
00016 #include "bioscara.hardware_driver/uPWM.h"
00017 #include "bioscara.hardware_driver/uErr.h"
00018
00025 namespace bioscara.hardware_driver
00026 {
00027     class Gripper : public BaseGripper
00028     {
00029         public:
00040             Gripper(float reduction, float offset, float min, float max);
00041
00050             err_type_t enable(void) override;
00051
00059             err_type_t disable(void) override;
00060
00061             err_type_t setPosition(float width) override;
00062
00063             err_type_t setServoPosition(float angle) override;
00064
00070             void setReduction(float reduction);
00071
00075             void setOffset(float offset);
00076
00077         protected:
00078             float reduction = 1;
00079             float offset = 0;
00080             float min = 0;
00081             float max = 0;
00082         private:
00083             RPI_PWM pwm;
00084             int freq = 50;
00085     };
00086 }
00087 #endif // MGRIPPER_H

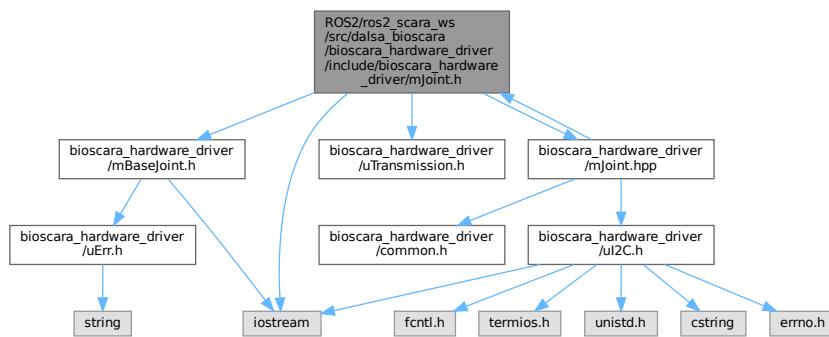
```

13.29 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mJoint.h File Reference

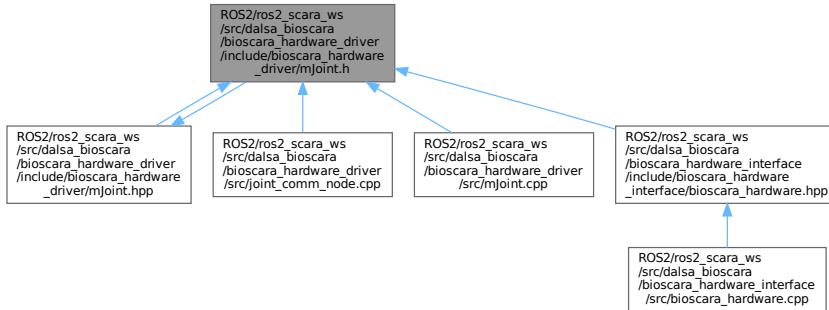
File including the Joint class.

```
#include <iostream>
#include "bioscara_hardware_driver/mBaseJoint.h"
#include "bioscara_hardware_driver/uTransmission.h"
#include "bioscara_hardware_driver/mJoint.hpp"
```

Include dependency graph for mJoint.h:



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



Classes

- class `bioscara_hardware_driver::Joint`
Representing a single joint on the I2C bus.

Namespaces

- namespace `bioscara_hardware_driver`
Generic `BaseGripper` object to interact with the robot gripper.

13.29.1 Detailed Description

File including the Joint class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-29

Copyright

Copyright (c) 2025

13.30 mJoint.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef MJOINT_H
00013 #define MJOINT_H
00014
00015 #include <iostream>
00016 #include "bioscara_hardware_driver/mBaseJoint.h"
00017 #include "bioscara_hardware_driver/uTransmission.h"
00018
00019 namespace bioscara_hardware_driver
00020 {
00025   class Joint : public BaseJoint
00026   {
00027     public:
00060       Joint(const std::string name, const int address, const float reduction, const float min, const
00061         float max);
00061       ~Joint(void);
00062
00072     err_type_t init(void) override;
00073
00083     err_type_t_deinit(void) override;
00084
00099     err_type_t_enable(u_int8_t driveCurrent, u_int8_t holdCurrent) override;
00100
00101     // /**
00102     // * @brief disengages the joint motor without closing i2c handle
00103     // * @return 0 on success,
00104     // * -1 on communication error,
00105     // * -5 if the joint is not initialized.
00106     // */
00107     // int disable(void);
00108
00109     // /**
00110     // * @brief Blocking implementation to home the joint.
00111     // *
00112     // * A blocking implementation which only returns after the the joint is no longer BUSY. See
00113     // * Joint::_home() for documentation.
00114     // * Additionally this method returns:
00115     // * @return -2 when not homed succesfull (isHomed flag still not set),
00116     // * -109 if the joint is already currently homing (for example from a call to
00117     // * Joint::startHoming().
00118     // * int home(float velocity, u_int8_t sensitivity, u_int8_t current);
00119
00120     // /**

```

```

00121     // * @brief non-blocking implementation to home the joint.
00122     // *
00123     // * See Joint::home() for documentation. The current_b_cmd flag is set to HOME
00124     // * This method returns immediatly after starting the homing sequence. This should be used when
00125     // * the blocking implementation is not acceptable.
00126     // * For example in the update loop of the
00127     bioscara_hardware_interface::BioscaraHardwareInterface::write().
00128
00129     // * Additionally this method returns:
00130     // * @return -109 if the joint is already currently homing (for example from a call to
00131     Joint::startHoming().
00132     // */
00133     // int startHoming(float velocity, u_int8_t sensitivity, u_int8_t current);
00134
00135     err_type_t postHoming(void) override;
00136
00137     err_type_t getPosition(float &pos) override;
00138
00139     err_type_t setPosition(float pos) override;
00140
00141     err_type_t moveSteps(int32_t steps) override;
00142
00143     err_type_t getVelocity(float &vel) override;
00144
00145     err_type_t setVelocity(float vel) override;
00146
00147     err_type_t checkOrientation(float angle = 10.0) override;
00148
00149     err_type_t stop(void) override;
00150     err_type_t disableCL(void) override;
00151
00152     err_type_t setDriveCurrent(u_int8_t current) override;
00153
00154     err_type_t setHoldCurrent(u_int8_t current) override;
00155
00156     err_type_t setBrakeMode(u_int8_t mode) override;
00157
00158     err_type_t setMaxAcceleration(float maxAccel) override;
00159
00160     err_type_t setMaxVelocity(float maxVel) override;
00161
00162     err_type_t enableStallguard(u_int8_t sensitivity) override;
00163
00164
00165     // /**
00166     // * @brief Checks the state if the motor is homed.
00167     // *
00168     // * Reads the internal state flags from the last transmission. If an update is neccessary call
00169     Joint::getFlags() before invoking this function.
00170     // *
00171     // * @return true if the motor is homed,
00172     // * false if not.
00173     // */
00174     // bool isHomed(void);
00175
00176     // /**
00177     // * @brief Checks the state if the motor is enabled.
00178     // *
00179     // * Reads the internal state flags from the last transmission. If an update is neccessary call
00180     Joint::getFlags() before invoking this function.
00181     // * If the motor actually can move depends on the state of the STALLED flag which can be checked
00182     using Joint::isStalled().
00183     // *
00184     // * @return true if the motor is enabled,
00185     // * false if not.
00186     // */
00187     // bool isEnabled(void);
00188
00189     // /**
00190     // * @brief Checks if the motor is stalled.
00191     // *
00192     // * Reads the internal state flags from the last transmission. If an update is neccessary call
00193     Joint::getFlags() before invoking this function.
00194     // * @return true if the motor is stalled,
00195     // * false if not.
00196     // */
00197     // bool isStalled(void);
00198
00199     // /**
00200     // * @brief Checks if the joint controller is busy processing a blocking command.
00201     // *
00202     // * Reads the internal state flags from the last transmission. If an update is neccessary call
00203     Joint::getFlags() before invoking this function.
00204     // * @return true if a blocking command is currently executing,
00205     // * false if not.
00206     // */
00207     // bool isBusy(void);
00208

```

```

00349     err_type_t getFlags(void) override;
00350
00361     err_type_t getHomingOffset(float &offset);
00362
00373     err_type_t setHomingOffset(const float offset);
00374
00375     // /**
00376     // * @brief get the currently active blocking command
00377     // *
00378     // * @return The the command of type stp_reg_t
00379     // */
00380     // stp_reg_t getCurrentBCmd(void);
00381
00382     protected:
00383     // /**
00384     // * @brief Blocking loop waiting for BUSY flag to reset.
00385     // *
00386     // * @param period_ms time in ms between polls.
00387     // */
00388     // void wait_while_busy(const float period_ms);
00389
00407     err_type_t _home(float velocity, u_int8_t sensitivity, u_int8_t current);
00408
00417     err_type_t checkCom(void);
00418
00419     // /**
00420     // * @brief State flags transmitted with every I2C transaction.
00421     // *
00422     // * The transmission flags purpose are to transmit the joints current state.
00423     // * Note: They can not be used as error indication of the execution of a transmitted write
00424     // * command,
00425     // * since commands are executed after the I2C transaction is completed. The status flags are one
00426     // * byte with following structure: \n
00427     // * |BIT7|BIT6|BIT5|BIT4|BIT3|BIT2|BIT1|BIT0|
00428     // * |---|---|---|---|---|---|---|---|
00429     // * |reserved|reserved|reserved|reserved|NOTENABLED|NOTHOMED|BUSY|STALL|
00430     // *
00431     // * \b STALL is set if a stall from the stall detection is sensed and the joint is stopped.
00432     // * The flag is cleared when the joint is homed or the Stallguard enabled. \n
00433     // * \b BUSY is set if the slave is busy processing a previous command. \n
00434     // * \b NOTHOMED is cleared if the joint is homed. Movement is only allowed if this flag is clear
00435     // * \b NOTENABLED is cleared if the joint is enabled after calling Joint::enable()
00436     // */
00437     // u_int8_t flags = 0x00;
00438
00439     float reduction = 1;
00440     float offset = 0;
00441     float min = 0;
00442     float max = 0;
00443
00444     // stp_reg_t current_b_cmd = NONE; //< Keeps track if a blocking command is being executed
00445
00446     private:
00447     template <typename T>
00448     int read(const stp_reg_t reg, T &data, u_int8_t &flags);
00449
00450     template <typename T>
00451     int write(const stp_reg_t reg, T data, u_int8_t &flags);
00452
00453     int address;
00454     int handle = -1;
00455 };
00456 }
00457 #include "bioscara.hardware_driver/mJoint.hpp"
00458
00459 #endif

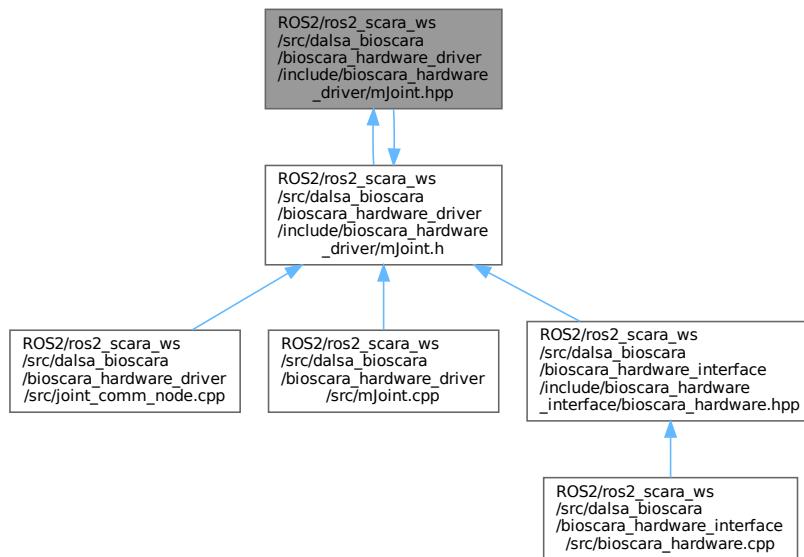
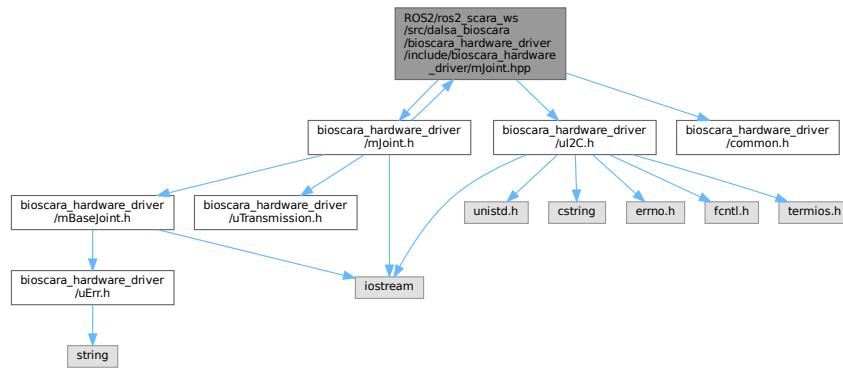
```

13.31 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara.hardware_driver/include/bioscara.hardware_driver/mJoint.hpp File Reference

Templated functions for the Joint class.

```
#include "bioscara.hardware_driver/mJoint.h"
#include "bioscara.hardware_driver/uI2C.h"
```

```
#include "bioscara_hardware_driver/common.h"
Include dependency graph for mJoint.hpp:
```



Namespaces

- namespace [bioscara_hardware_driver](#)
Generic [BaseGripper](#) object to interact with the robot gripper.

13.31.1 Detailed Description

Templated functions for the Joint class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-29

Copyright

Copyright (c) 2025

This header must be included at the END of the [mJoint.h](#) file.

13.32 mJoint.hpp

[Go to the documentation of this file.](#)

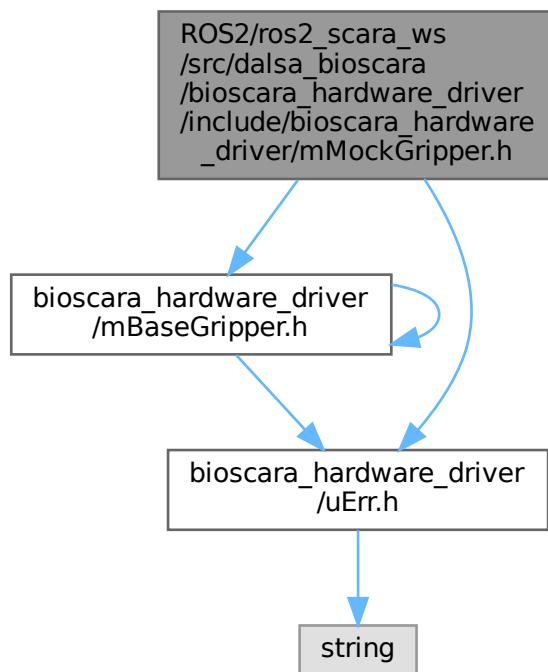
```
00001
00012 #include "bioscara_hardware_driver/mJoint.h"
00013 #include "bioscara_hardware_driver/uI2C.h"
00014 #include "bioscara_hardware_driver/common.h"
00015
00016 namespace bioscara_hardware_driver
00017 {
00033     template <typename T>
00034     int Joint::read(const stp_reg_t reg, T &data, u_int8_t &flags)
00035     {
00036         size_t size = sizeof(T) + RFLAGS_SIZE;
00037         char buf[MAX_BUFFER + RFLAGS_SIZE];
00038         int n = readFromI2CDev(this->handle, reg, buf, size);
00039         if (n != static_cast<int>(size))
00040         {
00041             return -1;
00042         }
00043         memcpy(&data, buf, size - RFLAGS_SIZE);
00044         memcpy(&flags, buf + size - RFLAGS_SIZE, RFLAGS_SIZE);
00045         return 0;
00046     }
00047
00063     template <typename T>
00064     int Joint::write(const stp_reg_t reg, T data, u_int8_t &flags)
00065     {
00066         size_t size = sizeof(T) + RFLAGS_SIZE;
00067         char buf[MAX_BUFFER + RFLAGS_SIZE];
00068         memcpy(buf, &data, size - RFLAGS_SIZE);
00069         int rc = writeToI2CDev(this->handle, reg, buf, size - RFLAGS_SIZE, buf + size - RFLAGS_SIZE);
00070         rc = rc > 0 ? 0 : rc;
00071         memcpy(&flags, buf + size - RFLAGS_SIZE, RFLAGS_SIZE);
00072         return rc;
00073     }
00074 }
```

13.33 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mMockGripper.h File Reference

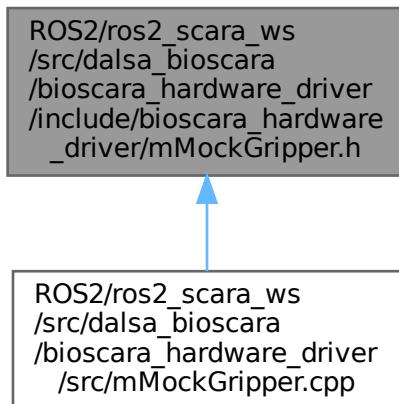
File containing the MockGripper class.

```
#include "bioscara_hardware_driver/mBaseGripper.h"
#include "bioscara_hardware_driver/uErr.h"
```

Include dependency graph for mMockGripper.h:



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



Classes

- class [bioscara.hardware_driver::MockGripper](#)

Namespaces

- namespace [bioscara.hardware_driver](#)
Generic BaseGripper object to interact with the robot gripper.

13.33.1 Detailed Description

File containing the MockGripper class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

Include this file for API functions to interact with the MockGripper.

13.34 mMockGripper.h

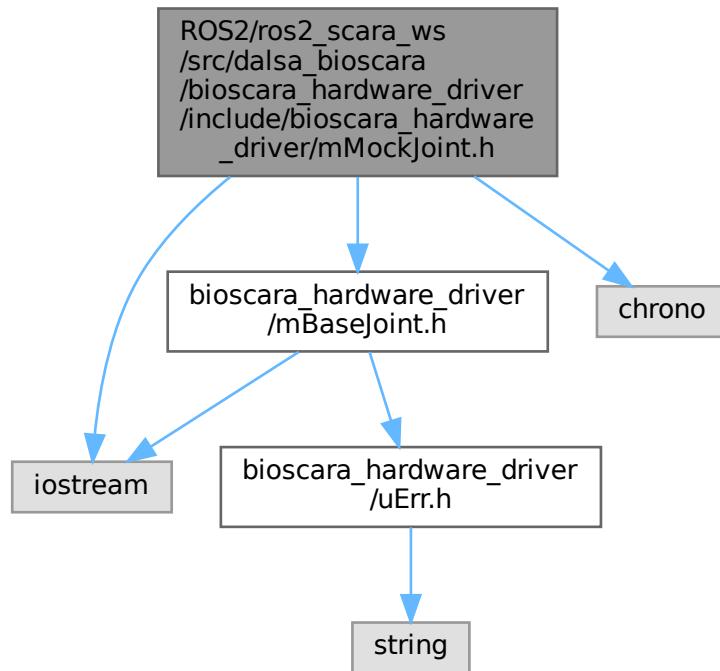
[Go to the documentation of this file.](#)

```
00001
00013 #ifndef MMOCKGRIPPER_H
00014 #define MMOCKGRIPPER_H
00015 #include "bioscara_hardware_driver/mBaseGripper.h"
00016 #include "bioscara_hardware_driver/uErr.h"
00021 namespace bioscara_hardware_driver
00022 {
00023     class MockGripper : public BaseGripper
00024     {
00025         public:
00026             MockGripper(void);
00027
00028         protected:
00029         private:
00030     };
00031 }
00032 #endif // MMOCKGRIPPER_H
```

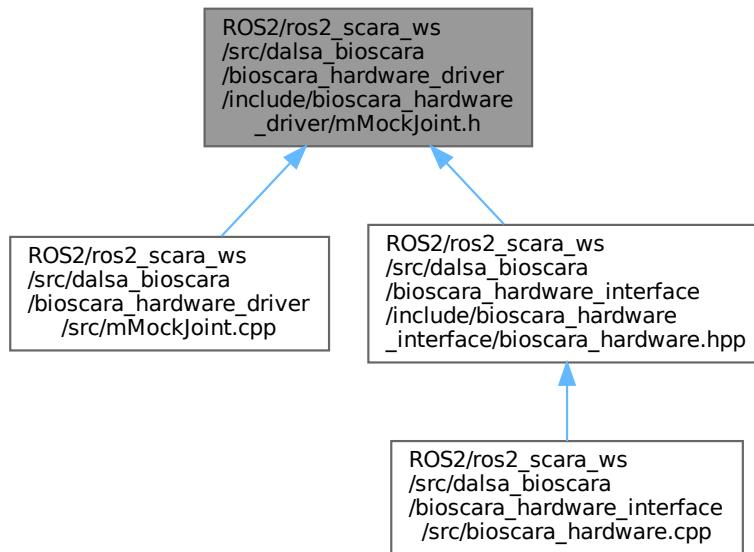
13.35 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mMockJoint.h File Reference

File including the MockJoint class.

```
#include <iostream>
#include "bioscara_hardware_driver/mBaseJoint.h"
#include <chrono>
Include dependency graph for mMockJoint.h:
```



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



Classes

- class [bioscara.hardware_driver::MockJoint](#)

Namespaces

- namespace [bioscara.hardware_driver](#)
Generic `BaseGripper` object to interact with the robot gripper.

13.35.1 Detailed Description

File including the MockJoint class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-29

Copyright

Copyright (c) 2025

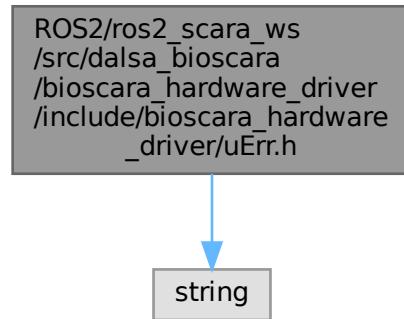
13.36 mMockJoint.h

[Go to the documentation of this file.](#)

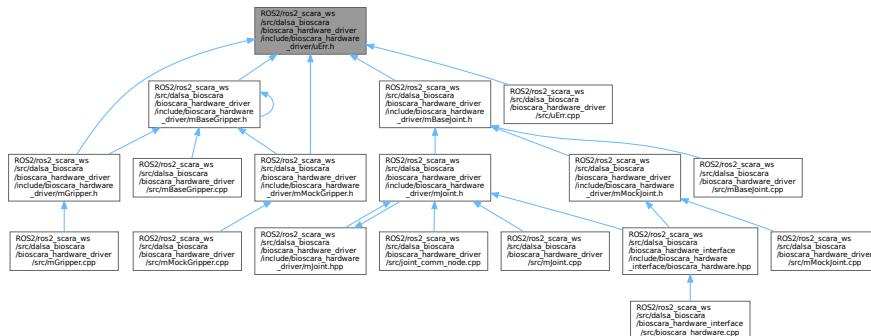
```
00001
00012 #ifndef MMOCKJOINT_H
00013 #define MMOCKJOINT_H
00014
00015 #include <iostream>
00016 #include "bioscara_hardware_driver/mBaseJoint.h"
00017 #include <chrono>
00018
00019 namespace bioscara_hardware_driver
00020 {
00021     class MockJoint : public BaseJoint
00022     {
00023     public:
00024         MockJoint(const std::string name);
00025
00026         err_type_t enable(u_int8_t driveCurrent, u_int8_t holdCurrent) override;
00027
00028         err_type_t disable(void) override;
00029
00030         err_type_t getPosition(float &pos) override;
00031
00032         err_type_t setPosition(float pos) override;
00033
00034         err_type_t getVelocity(float &vel) override;
00035
00036         err_type_t setVelocity(float vel) override;
00037
00038         err_type_t checkOrientation(float angle = 10.0) override;
00039
00040         err_type_t stop(void) override;
00041
00042         err_type_t getFlags(void) override;
00043
00044         bool isHomed(void) override;
00045
00046     protected:
00047         err_type_t _home(float velocity, u_int8_t sensitivity, u_int8_t current);
00048
00049     private:
00050         float q = 0.0;
00051         float qd = 0.0;
00052
00053         std::chrono::_V2::system_clock::time_point last_set_position =
00054             std::chrono::high_resolution_clock::now();
00055         std::chrono::_V2::system_clock::time_point last_set_velocity = last_set_position;
00056         std::chrono::_V2::system_clock::time_point async_start_time = last_set_position;
00057         float getDeltaT(std::chrono::_V2::system_clock::time_point &last_call, bool update = true);
00058         stp_reg_t op_mode = NONE;
00059     };
00060 }
00061 #endif
```

13.37 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/uErr.h File Reference

```
#include <string>
Include dependency graph for uErr.h:
```



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



Namespaces

- namespace `bioscara_hardware_driver`
Generic `BaseGripper` object to interact with the robot gripper.

Macros

- #define `RETURN_ON_ERROR(x)`
Macro which executes a function and returns from the calling function with the error code if the called function fails.
- #define `RETURN_ON_FALSE(a, err_code)`
Macro which returns the calling function with specified error_code if the given condition is false.
- #define `RETURN_ON_NEGATIVE(a, err_code)`
Macro which returns the calling function with specified error_code if the given condition is negative.

Enumerations

- enum class bioscara.hardware_driver::err_type_t {
 bioscara.hardware_driver::OK = 0 , bioscara.hardware_driver::ERROR = -1 , bioscara.hardware_driver::NOT_HOMED = -2 , bioscara.hardware_driver::NOT_ENABLED = -3 ,
 bioscara.hardware_driver::STALLED = -4 , bioscara.hardware_driver::NOT_INIT = -5 , bioscara.hardware_driver::COMM_ERROR = -6 , bioscara.hardware_driver::INVALID_ARGUMENT = -101 ,
 bioscara.hardware_driver::INCORRECT_STATE = -109 }

Enum defining common error types.

Functions

- std::string bioscara.hardware_driver::error_to_string (err_type_t err)
Converts an error code to a string and returns it.

13.37.1 Macro Definition Documentation

13.37.1.1 RETURN_ON_ERROR

```
#define RETURN_ON_ERROR(
    x )
```

Value:

```
do
{
    bioscara.hardware_driver::err_type_t err_rc_ = (x);
    if (err_rc_ != bioscara.hardware_driver::err_type_t::OK)
    {
        return err_rc_;
    }
} while (0);
```

Macro which executes a function and returns from the calling function with the error code if the called function fails.

Adapted from the [ESP-IDF](#)

Parameters

x	function to call
---	------------------

13.37.1.2 RETURN_ON_FALSE

```
#define RETURN_ON_FALSE(
    a,
    err_code )
```

Value:

```
do
{
    if (!(a))
    {
        return err_code;
    }
} while (0);
```

Macro which returns the calling function with specified error_code if the given condition is false.

Adapted from the [ESP-IDF](#)

Parameters

<i>a</i>	expression that evaluates to true or false
<i>err_code</i>	return code to return on false

13.37.1.3 RETURN_ON_NEGATIVE

```
#define RETURN_ON_NEGATIVE(
    a,
    err_code )
```

Value:

```
do
{
    if ((a) < 0)
    {
        return err_code;
    }
} while (0);
```

Macro which returns the calling function with specified error_code if the given condition is negative.

Adapted from the [ESP-IDF](#)

Parameters

<i>a</i>	expression that evaluates to a signed number
<i>err_code</i>	return code to return on false

13.38 uErr.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef UERR_H
00012 #define UERR_H
00013
00014 #include <string>
00015
00016 namespace bioscara_hardware_driver
00017 {
00022     enum class err_type_t
00023     {
00024         OK = 0,
00025         ERROR = -1,
00026         NOT_HOMED = -2,
00027         NOT_ENABLED = -3,
00028         STALLED = -4,
00029         NOT_INIT = -5,
00030         COMM_ERROR = -6,
00031         INVALID_ARGUMENT = -101,
00032         INCORRECT_STATE = -109,
00033     };
00035
00042     std::string error_to_string(err_type_t err);
00043 }
```

```
00044
00052 #define RETURN_ON_ERROR(x)
00053     do
00054     {
00055         bioscara_hardware_driver::err_type_t err_rc_ = (x);
00056         if (err_rc_ != bioscara_hardware_driver::err_type_t::OK)
```

```

00058         return err_rc_;
00059     }
00060 } while (0);
00061
00069 #define RETURN_ON_FALSE(a, err_code) \
00070     do \
00071     { \
00072         if (!(a)) \
00073         { \
00074             return err_code; \
00075         } \
00076     } while (0);
00077
00085 #define RETURN_ON_NEGATIVE(a, err_code) \
00086     do \
00087     { \
00088         if ((a) < 0) \
00089         { \
00090             return err_code; \
00091         } \
00092     } while (0);
00093
00094 #endif // UERR_H

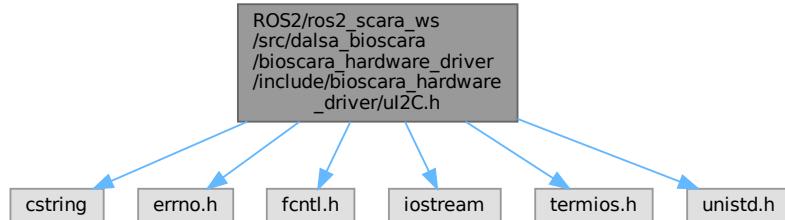
```

13.39 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/uI2C.h File Reference

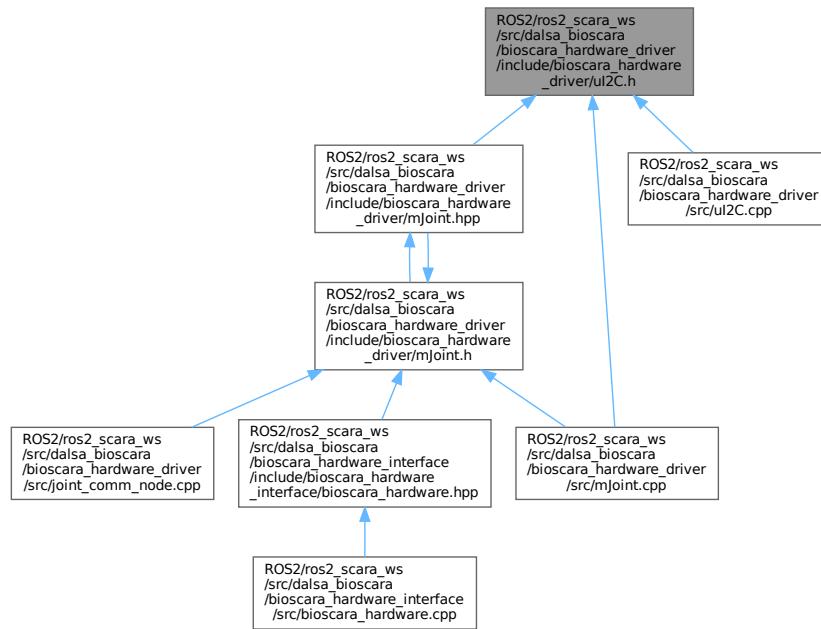
Low level utility for I2C communication on Raspberry Pi using Igpio library.

```
#include <cstring>
#include <errno.h>
#include <fcntl.h>
#include <iostream>
#include <termios.h>
#include <unistd.h>
```

Include dependency graph for uI2C.h:



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



Macros

- `#define ACK 'O'`
- `#define NACK 'N'`
- `#define RFLAGS_SIZE 1`
Size of the return flags in bytes.
- `#define MAX_BUFFER 4`
Maximum size of I2C Payload in bytes.

Functions

- `int openI2CDevHandle (const int dev_addr)`
Initiates an I2C device on the bus.
- `int readFromI2CDev (const int dev_handle, const int reg, char *buffer, const int data_length)`
reads block of bytes from device to buffer
- `int writeToI2CDev (const int dev_handle, const int reg, char *tx_buffer, const int data_length, char *RFLAGS_buffer)`
writes block of bytes from buffer to device
- `int closeI2CDevHandle (int &dev_handle)`
close an I2C device on the bus

13.39.1 Detailed Description

Low level utility for I2C communication on Raspberry Pi using Igpio library.

Author

Sebastian Storz

Version

0.1

Date

2025-05-28

Copyright

Copyright (c) 2025

Igpio needs to be installed and linked! Installation:

```
cd ~
sudo apt update
sudo apt install -y swig
wget https://github.com/joan2937/lg/archive/master.zip
unzip master.zip
cd lg-master
make
sudo make install
cd ..
sudo rm -rf lg-master
rm master.zip
```

bash

13.39.2 Macro Definition Documentation

13.39.2.1 ACK

```
#define ACK 'O'
```

13.39.2.2 MAX_BUFFER

```
#define MAX_BUFFER 4
```

Maximum size of I2C Payload in bytes.

4 bytes used to transmit floats and int32_t

13.39.2.3 NACK

```
#define NACK 'N'
```

13.39.2.4 RFLAGS_SIZE

```
#define RFLAGS_SIZE 1
```

Size of the return flags in bytes.

Only one byte used and hence set to 1.

13.39.3 Function Documentation

13.39.3.1 closeI2CDevHandle()

```
int closeI2CDevHandle (
    int & dev_handle )
```

close an I2C device on the bus

Parameters

<i>dev_handle</i>	device handle obtained from openI2CDevHandle
-------------------	--

Returns

0 on OK, negative on error.

13.39.3.2 openI2CDevHandle()

```
int openI2CDevHandle (
    const int dev_addr )
```

Initiates an I2C device on the bus.

Parameters

<i>dev_addr</i>	7-bit device address [0 - 0x7F]
-----------------	---------------------------------

Returns

the device handle, negative on error.

13.39.3.3 readFromI2CDev()

```
int readFromI2CDev (
    const int dev_handle,
    const int reg,
    char * buffer,
    const int data_length )
```

reads block of bytes from device to buffer

Parameters

<i>dev_handle</i>	device handle obtained from <code>openI2CDevHandle</code>
<i>reg</i>	the command/data register
<i>buffer</i>	pointer to data buffer to hold received values
<i>data_length</i>	number of bytes to read

Returns

number of bytes read, negative on error.

13.39.3.4 writeToI2CDev()

```
int writeToI2CDev (
    const int dev_handle,
    const int reg,
    char * tx_buffer,
    const int data_length,
    char * RFLAGS_buffer )
```

writes block of bytes from buffer to device

Parameters

<i>dev_handle</i>	device handle obtained from <code>openI2CDevHandle</code>
<i>reg</i>	the command/data register
<i>tx_buffer</i>	pointer to data buffer holding the data to send
<i>data_length</i>	number of bytes to send
<i>RFLAGS_buffer</i>	buffer to hold returned flags

Returns

0 on OK, negative on error.

13.40 ul2C.h

[Go to the documentation of this file.](#)

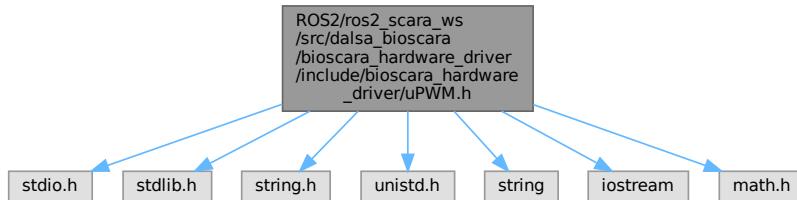
```
00001
00028 #ifndef USERIAL_H
00029 #define USERIAL_H
00030 #include <cstring>
00031 #include <errno.h>
00032 #include <fcntl.h>
00033 #include <iostream>
00034 #include <termios.h>
00035 #include <unistd.h>
00036
00037 #define ACK 'O'
00038 #define NACK 'N'
00039
00043 #define RFLAGS_SIZE 1
00044
00048 #define MAX_BUFFER 4 // Bytes
00049
00055 int openI2CDevHandle(const int dev_addr);
```

```
00056 int readFromI2CDev(const int dev_handle, const int reg, char *buffer, const int data_length);
00066
00076 int writeToI2CDev(const int dev_handle, const int reg, char *tx_buffer, const int data_length, char
  *RFLAGS_buffer);
00077
00083 int closeI2CDevHandle(int &dev_handle);
00084
00085
00086 #endif
```

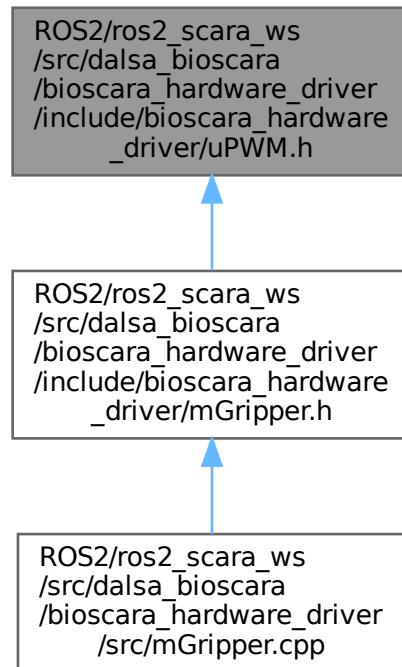
13.41 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↓ driver/include/bioscara_hardware_driver/uPWM.h File Reference

Includes source code for Hardware PWM generation on Raspberry Pi 4.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <string>
#include <iostream>
#include <math.h>
Include dependency graph for uPWM.h:
```



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



Classes

- class [RPI_PWM](#)
PWM class for the Raspberry Pi 4 and 5.

13.41.1 Detailed Description

Includes source code for Hardware PWM generation on Raspberry Pi 4.

Author

Sebastian Storz and Bernd Porr, bernd.porr@glasgow.ac.uk

Version

0.1

Date

2025-05-27

I copied this from: https://github.com/berndporr/rpi_pwm/blob/main/rpi_pwm.h and slightly modified it.

Igpio, the library used for I2C access can only generate soft PWM. The timing jitter will cause the servo to fidget. This may cause it to overheat and wear out prematurely.

Copyright

Copyright (c) 2025

13.42 uPWM.h

[Go to the documentation of this file.](#)

```

00001
00019 #ifndef __RPI_PWM
00020 #define __RPI_PWM
00021
00022 #include <stdio.h>
00023 #include <stdlib.h>
00024 #include <string.h>
00025 #include <unistd.h>
00026 #include <string>
00027 #include <iostream>
00028 #include <math.h>
00029
00033 class RPI_PWM
00034 {
00035 public:
00044     int start(int channel, int frequency, float duty_cycle = 0, int chip = 2)
00045     {
00046         chippath = "/sys/class/pwm/pwmchip" + std::to_string(chip);
00047         pwmpath = chippath + "/pwm" + std::to_string(channel);
00048         std::string p = chippath + "/export";
00049         FILE *const fp = fopen(p.c_str(), "w");
00050         if (NULL == fp)
00051         {
00052             std::cerr << "PWM device does not exist. Make sure to add 'dtoverlay=pwm-2chan' to
00053 /boot/firmware/config.txt.\n";
00054             return -1;
00055         }
00056         const int r = fprintf(fp, "%d", channel);
00057         fclose(fp);
00058         if (r < 0)
00059             return r;
00060         usleep(100000); // it takes a while till the PWM subdir is created
00061         per = (int)1E9 / frequency;
00062         setPeriod(per);
00063         setDutyCycle(duty_cycle);
00064         enable();
00065         return r;
00066     }
00070     void stop()
00071     {
00072         disable();
00073     }
00074
00075 ~RPI_PWM()
00076 {
00077     disable();
00078 }
00079
00085 inline int setDutyCycle(float v) const
00086 {
00087     const int dc = (int)round((float)per * (v / 100.0));
00088     const int r = setDutyCycleNS(dc);
00089     return r;
00090 }
00091
00092 private:
00093     void setPeriod(int ns) const

```

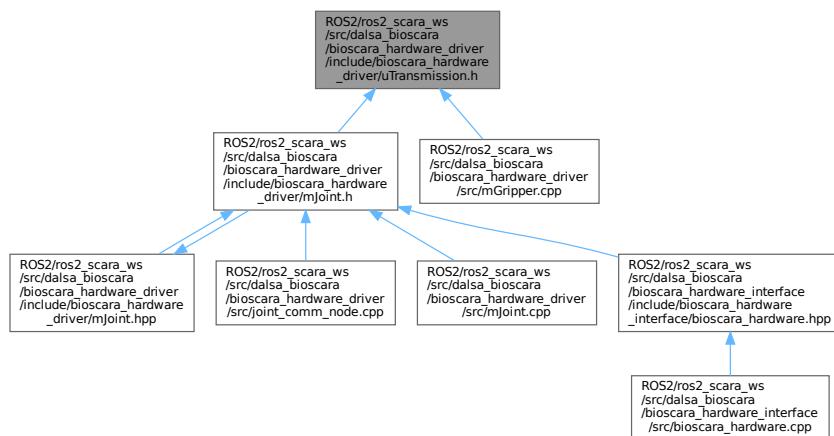
```

00094     {
00095         writeSYS(pwmpath + "/" + "period", ns);
00096     }
00097
00098     inline int setDutyCycleNS(int ns) const
00099     {
00100         const int r = writeSYS(pwmpath + "/" + "duty_cycle", ns);
00101         return r;
00102     }
00103
00104     void enable() const
00105     {
00106         writeSYS(pwmpath + "/" + "enable", 1);
00107     }
00108
00109     void disable() const
00110     {
00111         writeSYS(pwmpath + "/" + "enable", 0);
00112     }
00113
00114     int per = 0;
00115
00116     std::string chippath;
00117     std::string pwmpath;
00118
00119     inline int writeSYS(std::string filename, int value) const
00120     {
00121         FILE *const fp = fopen(filename.c_str(), "w");
00122         if (NULL == fp)
00123         {
00124             return -1;
00125         }
00126         const int r = fprintf(fp, "%d", value);
00127         fclose(fp);
00128         return r;
00129     }
00130 };
00131
00132 #endif

```

13.43 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/uTransmission.h File Reference

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



Macros

- `#define JOINT2ACTUATOR(in, reduction, offset) (reduction * (in - offset))`

- Macro for a simple transmission from joint units to actuator units.
- #define ACTUATOR2JOINT(in, reduction, offset) (in / reduction + offset)
- Macro for a simple transmission from actuator units to joint units.
- #define M_PI 3.14159265358979323846
- pi
- #define RAD2DEG(rad) (rad / M_PI * 180)
- Macro to convert radians to degree.
- #define DEG2RAD(deg) (deg * M_PI / 180)
- Macro to convert degree to radians.

13.43.1 Macro Definition Documentation

13.43.1.1 ACTUATOR2JOINT

```
#define ACTUATOR2JOINT(
    in,
    reduction,
    offset ) (in / reduction + offset)
```

Macro for a simple transmission from actuator units to joint units.

The translation is based on the ros2_control transmission interface, simple transmission. For position reduction and offset need to be used.

For velocity and acceleration only use reduction and NO offset

For effort/torque use 1/reduction and NO offset

13.43.1.2 DEG2RAD

```
#define DEG2RAD(
    deg ) (deg * M_PI / 180)
```

Macro to convert degree to radians.

13.43.1.3 JOINT2ACTUATOR

```
#define JOINT2ACTUATOR(
    in,
    reduction,
    offset ) (reduction * (in - offset))
```

Macro for a simple transmission from joint units to actuator units.

The translation is based on the ros2_control transmission interface, simple transmission. For position reduction and offset need to be used.

For velocity and acceleration only use reduction and NO offset

For effort/torque use 1/reduction and NO offset

13.43.1.4 M_PI

```
#define M_PI 3.14159265358979323846
pi
```

13.43.1.5 RAD2DEG

```
#define RAD2DEG(
    rad ) (rad / M_PI * 180)
```

Macro to convert radians to degree.

13.44 uTransmission.h

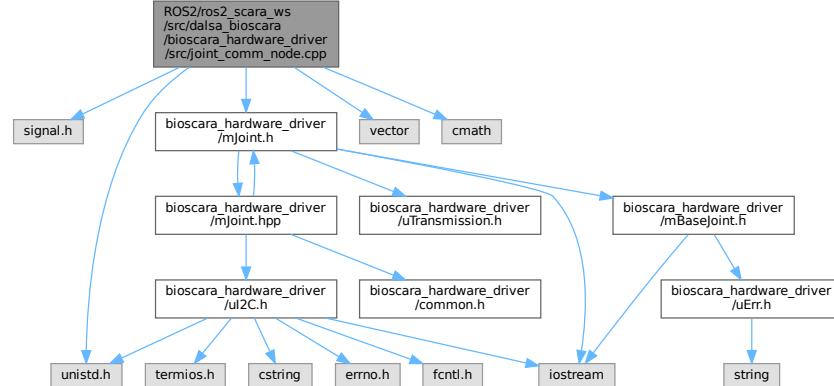
[Go to the documentation of this file.](#)

```
00001 #ifndef UTRANSMISSION_H
00002 #define UTRANSMISSION_H
00003
00012 #define JOINT2ACTUATOR(in, reduction, offset) (reduction * (in - offset))
00013
00022 #define ACTUATOR2JOINT(in, reduction, offset) (in / reduction + offset)
00023
00028 #define M_PI 3.14159265358979323846
00029
00034 #define RAD2DEG(rad) (rad / M_PI * 180)
00035
00040 #define DEG2RAD(deg) (deg * M_PI / 180)
00041 #endif //UTRANSMISSION_H
```

13.45 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/joint_comm_node.cpp File Reference

```
#include <signal.h>
#include <unistd.h>
#include "bioscara_hardware_driver/mJoint.h"
#include <vector>
#include <cmath>
```

Include dependency graph for joint_comm_node:



Functions

- void `INT_handler` (int s)
- int `main` (int argc, char **argv)

Variables

- `Joint J1` ("j1", 0x11, 35, -3.04647, 3.04647)
- `Joint J2` ("j2", 0x12, -2 *`M_PI`/0.004, 0.338, 0.0)
- `Joint J3` ("j3", 0x13, 24, -2.62672, 2.62672)
- `Joint J4` ("j4", 0x14, 12, -3.01069, 3.01069)

13.45.1 Function Documentation

13.45.1.1 INT_handler()

```
void INT_handler (
    int s )
```

13.45.1.2 main()

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

13.45.2 Variable Documentation

13.45.2.1 J1

```
Joint J1("j1", 0x11, 35, -3.04647, 3.04647) (
    "j1" ,
    0x11 ,
    35 ,
    -3. 04647,
    3. 04647 )
```

13.45.2.2 J2

```
Joint J2("j2", 0x12, -2 *M_PI/0.004, 0.338, 0.0) (
    "j2" ,
    0x12 ,
    -2 *M_PI/0. 004,
    0. 338,
    0. 0 )
```

13.45.2.3 J3

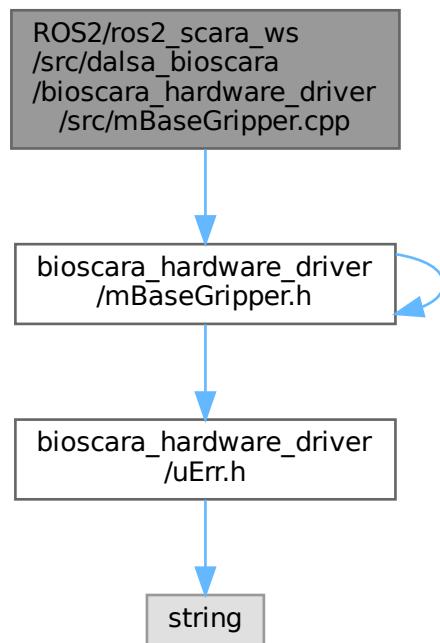
```
Joint J3("j3", 0x13, 24, -2.62672, 2.62672) (
    "j3" ,
    0x13 ,
    24 ,
    -2. 62672,
    2. 62672 )
```

13.45.2.4 J4

```
Joint J4("j4", 0x14, 12, -3.01069, 3.01069) (
    "j4" ,
    0x14 ,
    12 ,
    -3. 01069,
    3. 01069 )
```

13.46 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mBaseGripper.cpp File Reference

#include "bioscara_hardware_driver/mBaseGripper.h"
Include dependency graph for mBaseGripper.cpp:

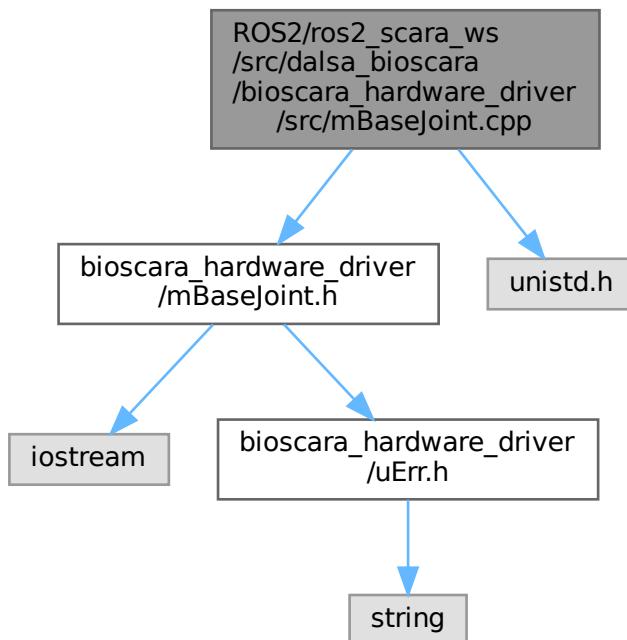


Namespaces

- namespace `bioscara_hardware_driver`
Generic `BaseGripper` object to interact with the robot gripper.

13.47 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mBaseJoint.cpp File Reference

```
#include "bioscara_hardware_driver/mBaseJoint.h"
#include <unistd.h>
Include dependency graph for mBaseJoint.cpp:
```



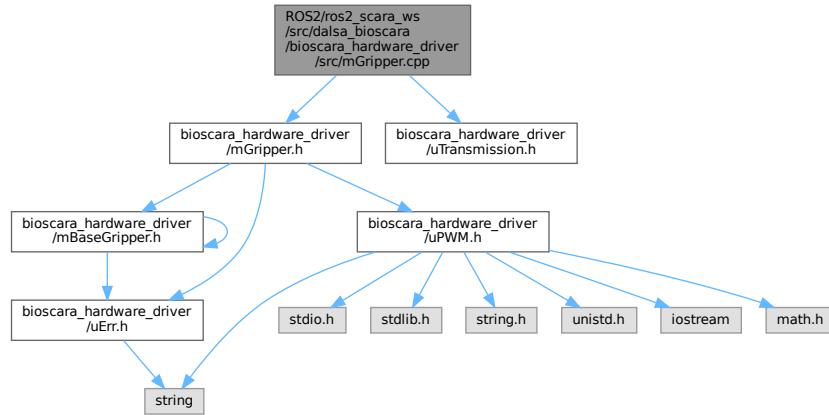
Namespaces

- namespace `bioscara_hardware_driver`
Generic `BaseGripper` object to interact with the robot gripper.

13.48 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mGripper.cpp File Reference

```
#include "bioscara_hardware_driver/mGripper.h"
#include "bioscara_hardware_driver/uTransmission.h"
```

Include dependency graph for mGripper.cpp:



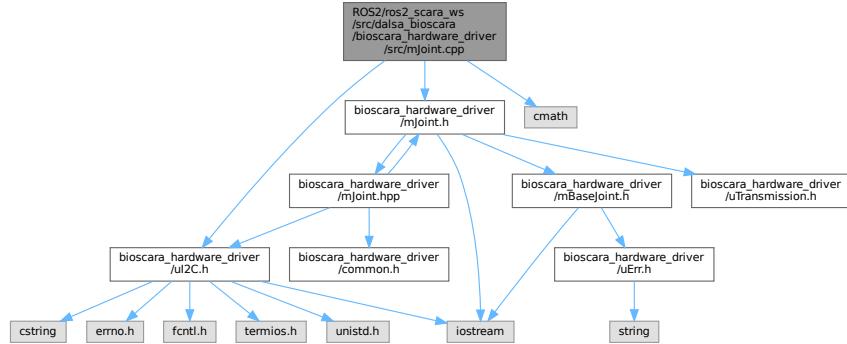
Namespaces

- namespace [bioscara_hardware_driver](#)
Generic [BaseGripper](#) object to interact with the robot gripper.

13.49 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mJoint.cpp File Reference

```
#include "bioscara_hardware_driver/uI2C.h"
#include "bioscara_hardware_driver/mJoint.h"
#include <cmath>
```

Include dependency graph for mJoint.cpp:

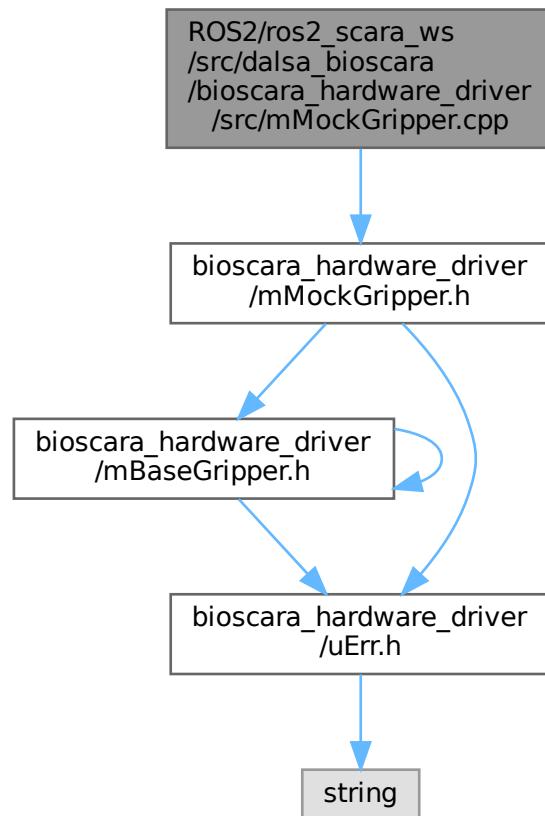


Namespaces

- namespace [bioscara_hardware_driver](#)
Generic [BaseGripper](#) object to interact with the robot gripper.

13.50 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockGripper.cpp File Reference

```
#include "bioscara_hardware_driver/mMockGripper.h"
Include dependency graph for mMockGripper.cpp:
```



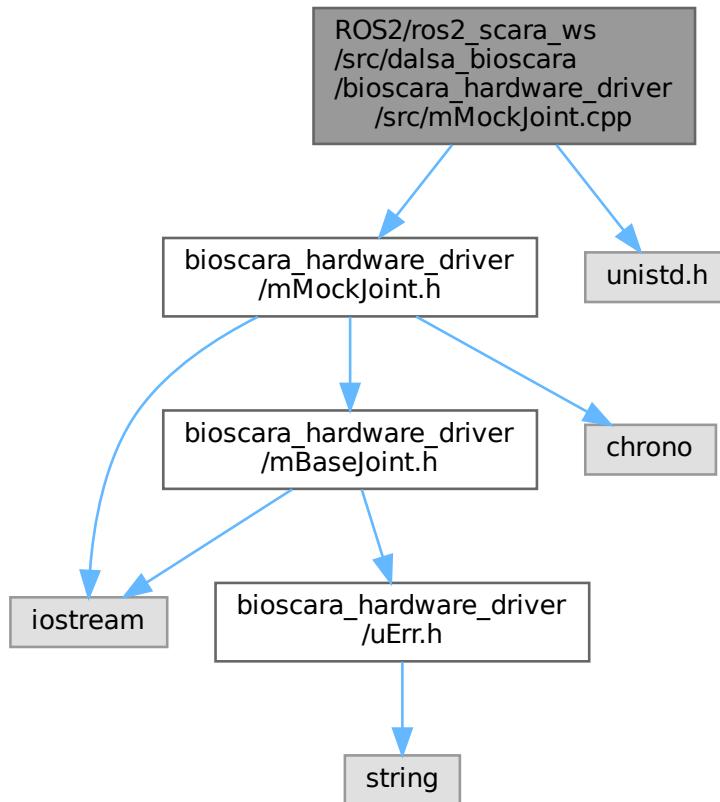
Namespaces

- namespace `bioscara_hardware_driver`
Generic `BaseGripper` object to interact with the robot gripper.

13.51 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockJoint.cpp File Reference

```
#include "bioscara_hardware_driver/mMockJoint.h"
#include <unistd.h>
```

Include dependency graph for mMockJoint.cpp:



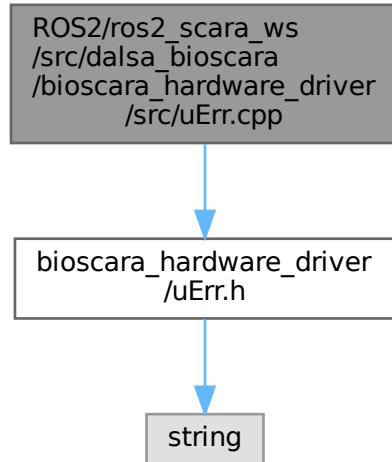
Namespaces

- namespace `bioscara_hardware_driver`
Generic `BaseGripper` object to interact with the robot gripper.

13.52 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/uErr.cpp File Reference

```
#include "bioscara_hardware_driver/uErr.h"
```

Include dependency graph for uErr.cpp:



Namespaces

- namespace `bioscara_hardware_driver`
Generic `BaseGripper` object to interact with the robot gripper.

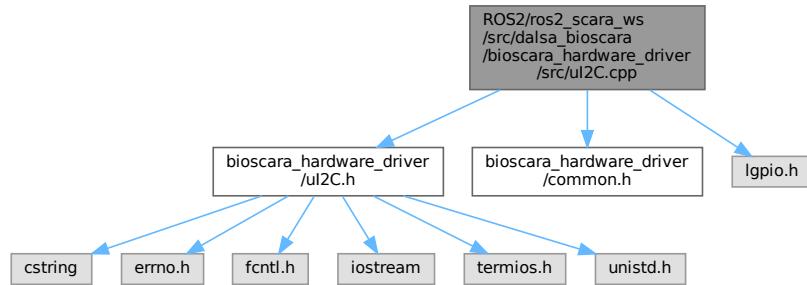
Functions

- std::string `bioscara_hardware_driver::error_to_string (err_type_t err)`
Converts an error code to a string and returns it.

13.53 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/uI2C.cpp File Reference

```
#include "bioscara_hardware_driver/uI2C.h"
#include "bioscara_hardware_driver/common.h"
#include <lpio.h>
```

Include dependency graph for I2C.cpp:



Functions

- int [openI2CDevHandle](#) (const int dev_addr)
Initiates an I2C device on the bus.
- int [readFromI2CDev](#) (const int dev_handle, const int reg, char *buffer, const int data_length)
reads block of bytes from device to buffer
- int [writeToI2CDev](#) (const int dev_handle, const int reg, char *tx_buffer, const int data_length, char *RFLAGS_buffer)
writes block of bytes from buffer to device
- int [closeI2CDevHandle](#) (int &dev_handle)
close an I2C device on the bus

13.53.1 Function Documentation

13.53.1.1 closeI2CDevHandle()

```
int closeI2CDevHandle (
    int & dev_handle )
```

close an I2C device on the bus

Parameters

<i>dev_handle</i>	device handle obtained from openI2CDevHandle
-------------------	--

Returns

0 on OK, negative on error.

13.53.1.2 openI2CDevHandle()

```
int openI2CDevHandle (
    const int dev_addr )
```

Initiates an I2C device on the bus.

Parameters

<i>dev_addr</i>	7-bit device address [0 - 0x7F]
-----------------	---------------------------------

Returns

the device handle, negative on error.

13.53.1.3 `readFromI2CDev()`

```
int readFromI2CDev (
    const int dev_handle,
    const int reg,
    char * buffer,
    const int data_length )
```

reads block of bytes from device to buffer

Parameters

<i>dev_handle</i>	device handle obtained from <code>openI2CDevHandle</code>
<i>reg</i>	the command/data register
<i>buffer</i>	pointer to data buffer to hold received values
<i>data_length</i>	number of bytes to read

Returns

number of bytes read, negative on error.

13.53.1.4 `writeToI2CDev()`

```
int writeToI2CDev (
    const int dev_handle,
    const int reg,
    char * tx_buffer,
    const int data_length,
    char * RFLAGS_buffer )
```

writes block of bytes from buffer to device

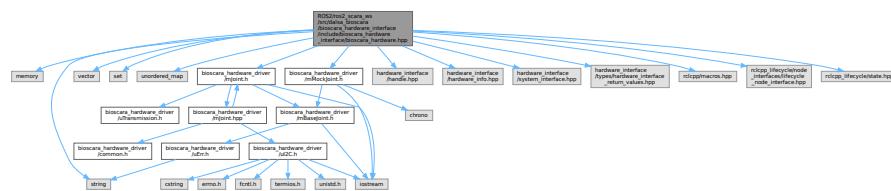
Parameters

<i>dev_handle</i>	device handle obtained from <code>openI2CDevHandle</code>
<i>reg</i>	the command/data register
<i>tx_buffer</i>	pointer to data buffer holding the data to send
<i>data_length</i>	number of bytes to send
<i>RFLAGS_buffer</i>	buffer to hold returned flags

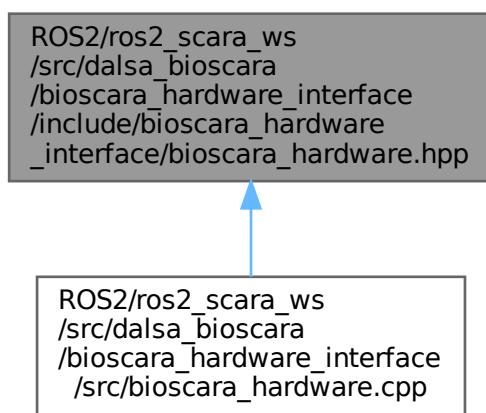
0 on OK, negative on error.

13.54 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include/bioscara_hardware_interface/bioscara_hardware.hpp File Reference

```
#include <memory>
#include <string>
#include <vector>
#include <set>
#include <unordered_map>
#include "bioscara_hardware_driver/mJoint.h"
#include "bioscara_hardware_driver/mMockJoint.h"
#include "hardware_interface/handle.hpp"
#include "hardware_interface/hardware_info.hpp"
#include "hardware_interface/system_interface.hpp"
#include "hardware_interface/types/hardware_interface_return_values.hpp"
#include "rclcpp/macros.hpp"
#include "rclcpp_lifecycle/node_interfaces/lifecycle_node_interface.hpp"
#include "rclcpp_lifecycle/state.hpp"
Include dependency graph for bioscara_hardware.hpp:
```



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



Classes

- class [bioscara_hw_interface::BioscaraHardwareInterface](#)
The bioscara hardware interface class.
 - struct [bioscara_hw_interface::BioscaraHardwareInterface::joint_homing_config_t](#)
configuration structure holding the passed homing paramters from the ros2_control urdf
 - struct [bioscara_hw_interface::BioscaraHardwareInterface::joint_config_t](#)
configuration structure holding the passed paramters from the ros2_control urdf

Namespaces

- namespace bioscara.hardware_interface

Variables

- `constexpr char bioscara.hardware_interface::HW_IF_HOME [] = "home"`

13.55 bioscara_hardware.hpp

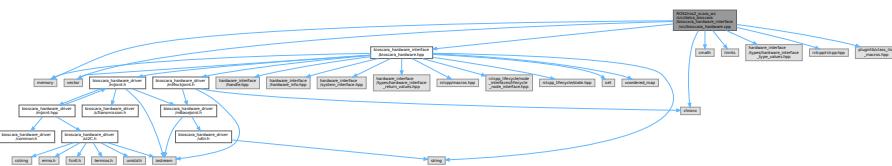
[Go to the documentation of this file.](#)

```
00001 // Copyright 2023 ros2_control Development Team
00002 //
00003 // Licensed under the Apache License, Version 2.0 (the "License");
00004 // you may not use this file except in compliance with the License.
00005 // You may obtain a copy of the License at
00006 //
00007 //     http://www.apache.org/licenses/LICENSE-2.0
00008 //
00009 // Unless required by applicable law or agreed to in writing, software
00010 // distributed under the License is distributed on an "AS IS" BASIS,
00011 // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00012 // See the License for the specific language governing permissions and
00013 // limitations under the License.
00014
00015 #ifndef BIOSCARA_HARDWARE_INTERFACE_HPP_
00016 #define BIOSCARA_HARDWARE_INTERFACE_HPP_
00017
00018 #include <memory>
00019 #include <string>
00020 #include <vector>
00021 #include <set>
00022 #include <unordered_map>
00023 #include <memory>
00024
00025 #include "bioscara_hardware_driver/mJoint.h"
00026 #include "bioscara_hardware_driver/mMockJoint.h"
00027
00028 #include "hardware_interface/handle.hpp"
00029 #include "hardware_interface/hardware_info.hpp"
00030 #include "hardware_interface/system_interface.hpp"
00031 #include "hardware_interface/types/hardware_interface_return_values.hpp"
00032 #include "rclcpp/macros.hpp"
00033 #include "rclcpp_lifecycle/node_interfaces/lifecycle_node_interface.hpp"
00034 #include "rclcpp_lifecycle/state.hpp"
00035
00036 namespace bioscara_hardware_interface
00037 {
00038     constexpr char HW_IF_HOME[] = "home";
00039
00040     class BioscaraHardwareInterface : public hardware_interface::SystemInterface
00041     {
00042     public:
00043         RCLCPP_SHARED_PTR_DEFINITIONS(BioscaraHardwareInterface)
00044
00045         hardware_interface::CallbackReturn on_init(
00046             const hardware_interface::HardwareComponentInterfaceParams &params) override;
00047
00048         hardware_interface::CallbackReturn on_shutdown(
00049             const hardware_interface::HardwareComponentInterfaceParams &params) override;
```

```
00100         const rclcpp_lifecycle::State &previous_state) override;
00101
00109     hardware_interface::CallbackReturn on_configure(
00110         const rclcpp_lifecycle::State &previous_state) override;
00111
00120     hardware_interface::CallbackReturn on_cleanup(
00121         const rclcpp_lifecycle::State &previous_state) override;
00122
00137     hardware_interface::CallbackReturn on_activate(
00138         const rclcpp_lifecycle::State &previous_state) override;
00139
00148     hardware_interface::CallbackReturn on_deactivate(
00149         const rclcpp_lifecycle::State &previous_state) override;
00150
00171     hardware_interface::return_type read(
00172         const rclcpp::Time &time,
00173         const rclcpp::Duration &period) override;
00174
00196     hardware_interface::return_type write(
00197         const rclcpp::Time &time,
00198         const rclcpp::Duration &period) override;
00199
00228     hardware_interface::return_type prepare_command_mode_switch(
00229         const std::vector<std::string> &start_interfaces,
00230         const std::vector<std::string> &stop_interfaces) override;
00231
00248     hardware_interface::return_type perform_command_mode_switch(
00249         const std::vector<std::string> &start_interfaces,
00250         const std::vector<std::string> &stop_interfaces) override;
00251
00279     hardware_interface::CallbackReturn on_error(
00280         const rclcpp_lifecycle::State &previous_state) override;
00281
00282 private:
00283     struct joint_homing_config_t
00284     {
00291         float speed = 0;
00292         u_int8_t threshold = 10;
00293         u_int8_t current = 10;
00294         float acceleration = 0.01;
00295     };
00296
00303     struct joint_config_t
00304     {
00305         int i2c_address;
00306         float reduction = 1;
00307         float min;
00308         float max;
00309         u_int8_t drive_current;
00310         u_int8_t hold_current;
00311         u_int8_t stall_threshold;
00312         float max_velocity;
00313         float max_acceleration;
00314         joint_homing_config_t homing;
00315     };
00316
00327     std::unordered_map<std::string, std::unique_ptr<bioscara.hardware_driver::BaseJoint>> _joints;
00328
00336     std::unordered_map<std::string, joint_config_t> _joint_cfg;
00337
00351     std::unordered_map<std::string, std::set<std::string>> _joint_command_modes;
00352
00361     bioscara.hardware_driver::err_type_t start_homing(const std::string name, float velocity);
00362
00370     bioscara.hardware_driver::err_type_t stop_homing(const std::string name);
00371
00379     void split_interface_string_to_joint_and_name(std::string interface, std::string &joint_name,
00380                                                 std::string &interface_name);
00380
00387     bioscara.hardware_driver::err_type_t activate_joint(const std::string name);
00388
00395     bioscara.hardware_driver::err_type_t deactivate_joint(const std::string name);
00396 };
00397
00398 } // namespace bioscara.hardware_interface
00399
00400 #endif // BIOSCARA_HARDWARE_INTERFACE_HPP_
```

13.56 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/src/bioscara_hardware.cpp File Reference

```
#include "bioscara_hardware_interface/bioscara_hardware.hpp"
#include <chrono>
#include <cmath>
#include <limits>
#include <memory>
#include <vector>
#include "hardware_interface/types/hardware_interface_type_values.hpp"
#include "rclcpp/rclcpp.hpp"
#include "pluginlib/class_list_macros.hpp"
Include dependency graph for bioscara_hardware.cpp:
```



Namespaces

- namespace bioscara.hardware_interface

Index

_home
 bioscara_hardware_driver::BaseJoint, 34
 bioscara_hardware_driver::Joint, 66
 bioscara_hardware_driver::MockJoint, 86

_joint_cfg
 bioscara_hardware_interface::BioscaraHardwareInterface, 56

_joint_command_modes
 bioscara_hardware_interface::BioscaraHardwareInterface, 56

_joints
 bioscara_hardware_interface::BioscaraHardwareInterface, 56

~BaseJoint
 bioscara_hardware_driver::BaseJoint, 34

~Joint
 bioscara_hardware_driver::Joint, 66

~RPI_PWM
 RPI_PWM, 93

acceleration
 bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing, 79

ACK
 joint.h, 101
 ul2C.h, 139

activate_joint
 bioscara_hardware_interface::BioscaraHardwareInterface, 47

ACTUATOR2JOINT
 uTransmission.h, 146

address
 bioscara_hardware_driver::Joint, 76

ADR
 configuration.h, 96

ANGLEMOVED
 bioscara_hardware_driver::BaseJoint, 33
 joint.h, 102

Arduino/joint/configuration.h, 95, 97

Arduino/joint/filters.h, 97, 99

Arduino/joint/joint.h, 99, 104

Arduino/joint/joint.ino, 105

Arduino/joint/stall.h, 110, 111

async_start_time
 bioscara_hardware_driver::MockJoint, 90

BaseGripper
 bioscara_hardware_driver::BaseGripper, 28

BaseJoint
 bioscara_hardware_driver::BaseJoint, 33

 bioscara, 21
 generate_launch_description, 21

 bioscara_hardware_driver, 21
 COMM_ERROR, 22
 err_type_t, 22
 error_to_string, 23
 INCORRECT_STATE, 22
 INVALID_ARGUMENT, 22
 NOT_ENABLED, 22
 NOT_HOMED, 22
 OK, 22
 STALLED, 22

 bioscara_hardware_driver::BaseGripper, 27
 BaseGripper, 28
 deinit, 28
 disable, 28
 enable, 28
 init, 28
 setOffset, 29
 setPosition, 29
 setReduction, 29
 setServoPosition, 29

 bioscara_hardware_driver::BaseJoint, 30
 _home, 34
 ~BaseJoint, 34

 bioscara_hardware_driver::ANGLEMOVED, 33
 BaseJoint, 33
 CHECKORIENTATION, 33
 checkOrientation, 34
 CLEARSTALL, 33
 current_b_cmd, 44
 deinit, 35
 disable, 35
 disableCL, 35
 DISABLECLOSEDLOOP, 33
 DISABLEPID, 33
 DISABLESTALLGUARD, 33
 enable, 35
 ENABLECLOSEDLOOP, 33
 ENABLEPID, 33
 ENABLESTALLGUARD, 33
 enableStallguard, 35
 flags, 44
 getCurrentBCmd, 36
 GETDRIVERERRPM, 33
 GETENCODERRPM, 33
 getFlags, 36, 37

GETMOTORSTATE, 33
 GETPIDERROR, 33
 getPosition, 37
 getVelocity, 37
 HOME, 33
 home, 38
 HOMEOFFSET, 33
 init, 38
 isBusy, 38
 isEnabled, 38
 isHomed, 39
 isStalled, 39
 MOVEANGLE, 33
 MOVESTEPS, 33
 moveSteps, 39
 MOVETOANGLE, 33
 MOVETOEND, 33
 name, 44
 NONE, 33
 PING, 33
 postHoming, 40
 RUNCOTINOUS, 33
 SETBRAKEMODE, 33
 setBrakeMode, 40
 SETCONTROLTHRESHOLD, 33
 SETCURRENT, 33
 setDriveCurrent, 40
 SETHOLDCURRENT, 33
 setHoldCurrent, 41
 SETMAXACCELERATION, 33
 setMaxAcceleration, 41
 SETMAXDECELERATION, 33
 SETMAXVELOCITY, 33
 setMaxVelocity, 42
 setPosition, 42
 SETRPM, 33
 SETUP, 33
 setVelocity, 42
 startHoming, 43
 STOP, 33
 stop, 43
 stp_reg_t, 32
 wait_while_busy, 43
bioscara.hardware_driver::Gripper, 57
 disable, 59
 enable, 59
 freq, 61
 Gripper, 59
 max, 61
 min, 61
 offset, 61
 pwm, 61
 reduction, 61
 setOffset, 59
 setPosition, 60
 setReduction, 60
 setServoPosition, 60
bioscara.hardware_driver::Joint, 62
 _home, 66
 ~Joint, 66
 address, 76
 checkCom, 67
 checkOrientation, 67
 deinit, 67
 disableCL, 67
 enable, 68
 enableStallguard, 68
 getFlags, 69
 getHomingOffset, 69
 getPosition, 69
 getVelocity, 70
 handle, 76
 init, 70
 Joint, 65
 max, 76
 min, 76
 moveSteps, 70
 offset, 76
 postHoming, 71
 read, 71
 reduction, 76
 setBrakeMode, 72
 setDriveCurrent, 72
 setHoldCurrent, 72
 setHomingOffset, 73
 setMaxAcceleration, 73
 setMaxVelocity, 74
 setPosition, 74
 setVelocity, 74
 stop, 75
 write, 75
bioscara.hardware_driver::MockGripper, 81
 MockGripper, 82
bioscara.hardware_driver::MockJoint, 83
 _home, 86
 async_start_time, 90
 checkOrientation, 86
 disable, 87
 enable, 87
 getDeltaT, 87
 getFlags, 87
 getPosition, 88
 getVelocity, 88
 isHomed, 89
 last_set_position, 90
 last_set_velocity, 90
 MockJoint, 86
 op_mode, 90
 q, 90
 qd, 91
 setPosition, 89
 setVelocity, 89
 stop, 90
bioscara.hardware_interface, 23
 HW_IF_HOME, 23

bioscara.hardware_interface::BioscaraHardwareInterface, joint.h, 103
 44
 _joint_cfg, 56
 _joint_command_modes, 56
 _joints, 56
 activate_joint, 47
 deactivate_joint, 48
 on_activate, 48
 on_cleanup, 48
 on_configure, 49
 on_deactivate, 49
 on_error, 50
 on_init, 50
 on_shutdown, 51
 perform_command_mode_switch, 52
 prepare_command_mode_switch, 53
 read, 53
 split_interface_string_to_joint_and_name, 54
 start_homing, 54
 stop_homing, 55
 write, 55
bioscara.hardware_interface::BioscaraHardwareInterface::joint_config, bioscara.hardware_interface::BioscaraHardwareInterface, 48
 77
 drive_current, 77
 hold_current, 77
 homing, 78
 i2c_address, 78
 max, 78
 max_acceleration, 78
 max_velocity, 78
 min, 78
 reduction, 78
 stall_threshold, 78
bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing, bioscara.hardware_interface::BioscaraHardwareInterface, 79
 acceleration, 79
 current, 79
 speed, 79
 threshold, 79
blocking_handler
 joint.ino, 107
cb_data
 MovMax, 92
cb_index
 MovMax, 92
checkCom
 bioscara.hardware_driver::Joint, 67
CHECKORIENTATION
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103
checkOrientation
 bioscara.hardware_driver::BaseJoint, 34
 bioscara.hardware_driver::Joint, 67
 bioscara.hardware_driver::MockJoint, 86
chippath
 RPI_PWM, 94
CLEARSTALL
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103
closeI2CDevHandle
 ul2C.cpp, 155
 ul2C.h, 140
COMM_ERROR
 bioscara.hardware_driver, 22
common.h
 DUMP_BUFFER, 115
configuration.h
 ADR, 96
 MAXACCEL, 96
 MAXVEL, 96
current
 bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing, 79
 current_b_cmd
 bioscara.hardware_driver::BaseJoint, 44
data_files
 setup, 24
deactivate_joint
 bioscara.hardware_interface::BioscaraHardwareInterface, 48
DEG2RAD
 uTransmission.h, 146
deinit
 bioscara.hardware_driver::BaseGripper, 28
 bioscara.hardware_driver::BaseJoint, 35
 bioscara.hardware_driver::Joint, 67
description
 setup, 24
disable
 bioscara.hardware_driver::BaseGripper, 28
 bioscara.hardware_driver::BaseJoint, 35
 bioscara.hardware_driver::Gripper, 59
 bioscara.hardware_driver::MockJoint, 87
 RPI_PWM, 93
disableCL
 bioscara.hardware_driver::BaseJoint, 35
 bioscara.hardware_driver::Joint, 67
DISABLECLOSEDLOOP
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103
DISABLEPID
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103
DISABLESTALLGUARD
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103
display, 23
 generate_launch_description, 23
docs/DOCS_README.md, 111
Documentation, 1
drive_current
 bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing, 77
DUMP_BUFFER
 common.h, 115
 joint.h, 101

enable
bioscara_hardware_driver::BaseGripper, 28
bioscara_hardware_driver::BaseJoint, 35
bioscara_hardware_driver::Gripper, 59
bioscara_hardware_driver::Joint, 68
bioscara_hardware_driver::MockJoint, 87
RPI_PWM, 93

ENABLECLOSEDLOOP
bioscara_hardware_driver::BaseJoint, 33
joint.h, 103

ENABLEPID
bioscara_hardware_driver::BaseJoint, 33
joint.h, 103

ENABLESTALLGUARD
bioscara_hardware_driver::BaseJoint, 33
joint.h, 103

enableStallguard
bioscara_hardware_driver::BaseJoint, 35
bioscara_hardware_driver::Joint, 68

entry_points
setup, 24

err_type_t
bioscara_hardware_driver, 22

ERROR
bioscara_hardware_driver, 22

error_to_string
bioscara_hardware_driver, 23

flags
bioscara_hardware_driver::BaseJoint, 44

freq
bioscara_hardware_driver::Gripper, 61

gazebo, 24
generate_launch_description, 24

generate_launch_description
bioscara, 21
display, 23
gazebo, 24
test_joint_trajectory_controller, 25

getCurrentBCmd
bioscara_hardware_driver::BaseJoint, 36

getDeltaT
bioscara_hardware_driver::MockJoint, 87

GETDRIVERERRPM
bioscara_hardware_driver::BaseJoint, 33
joint.h, 102

GETENCODERRPM
bioscara_hardware_driver::BaseJoint, 33
joint.h, 103

getFlags
bioscara_hardware_driver::BaseJoint, 36, 37
bioscara_hardware_driver::Joint, 69
bioscara_hardware_driver::MockJoint, 87

getHomingOffset
bioscara_hardware_driver::Joint, 69

GETMOTORSTATE
bioscara_hardware_driver::BaseJoint, 33
joint.h, 102

GETPIDERROR
bioscara_hardware_driver::BaseJoint, 33
joint.h, 103

getPosition
bioscara_hardware_driver::BaseJoint, 37
bioscara_hardware_driver::Joint, 69
bioscara_hardware_driver::MockJoint, 88

getVelocity
bioscara_hardware_driver::BaseJoint, 37
bioscara_hardware_driver::Joint, 70
bioscara_hardware_driver::MockJoint, 88

Gripper
bioscara_hardware_driver::Gripper, 59

handle
bioscara_hardware_driver::Joint, 76

hold_current
bioscara_hardware_interface::BioscaraHardwareInterface::joint_config,
77

HOME
bioscara_hardware_driver::BaseJoint, 33
joint.h, 103

home
bioscara_hardware_driver::BaseJoint, 38

HOMEOFFSET
bioscara_hardware_driver::BaseJoint, 33
joint.h, 103

homing
bioscara_hardware_interface::BioscaraHardwareInterface::joint_config,
78

HW_IF_HOME
bioscara_hardware_interface, 23

i2c_address
bioscara_hardware_interface::BioscaraHardwareInterface::joint_config,
78

INCORRECT_STATE
bioscara_hardware_driver, 22

init
bioscara_hardware_driver::BaseGripper, 28
bioscara_hardware_driver::BaseJoint, 38
bioscara_hardware_driver::Joint, 70

install_requires
setup, 24

INT_handler
joint_comm_node.cpp, 148

INVALID_ARGUMENT
bioscara_hardware_driver, 22

isBusy
bioscara_hardware_driver::BaseJoint, 38

isEnabled
bioscara_hardware_driver::BaseJoint, 38

isHomed
bioscara_hardware_driver::BaseJoint, 39
bioscara_hardware_driver::MockJoint, 89

isStalled
bioscara_hardware_driver::BaseJoint, 39

J1

joint_comm_node.cpp, 148
J2 joint_comm_node.cpp, 148
J3 joint.ino, 106
joint_comm_node.cpp, 148
J4 joint_comm_node.cpp, 149
Joint bioscara.hardware_driver::Joint, 65
joint.h ACK, 101
ANGLEMOVED, 102
CHECKORIENTATION, 103
CLEARSTALL, 103
DISABLECLOSEDLOOP, 103
DISABLEPID, 103
DISABLESTALLGUARD, 103
DUMP_BUFFER, 101
ENABLECLOSEDLOOP, 103
ENABLEPID, 103
ENABLESTALLGUARD, 103
GETDRIVERERRPM, 102
GETENCODERRPM, 103
GETMOTORSTATE, 102
GETPIDERROR, 103
HOME, 103
HOMEOFFSET, 103
MAX_BUFFER, 102
MOVEANGLE, 102
MOVESTEPS, 102
MOVETOANGLE, 102
MOVETOEND, 103
NACK, 102
PING, 102
readValue, 103
RFLAGS_SIZE, 102
RUNCOTINOUS, 102
SETBRAKEMODE, 103
SETCONTROLTHRESHOLD, 103
SETCURRENT, 103
SETHOOKCURRENT, 103
SETMAXACCELERATION, 103
SETMAXDECELERATION, 103
SETMAXVELOCITY, 103
SETRPM, 102
SETUP, 102
STOP, 103
stp_reg_t, 102
writeValue, 103
joint.ino blocking_handler, 107
J3, 106
loop, 107
non_blocking_handler, 107
receiveEvent, 108
reg, 109
requestEvent, 108
rx_buf, 109
rx_data_ready, 109
rx_length, 109
setup, 108
stepper, 109
tx_buf, 109
tx_length, 109
JOINT2ACTUATOR uTransmission.h, 146
joint_comm_node.cpp INT_handler, 148
J1, 148
J2, 148
J3, 148
J4, 149
main, 148

K Lowpass, 81

last_set_position bioscara.hardware_driver::MockJoint, 90
last_set_velocity bioscara.hardware_driver::MockJoint, 90
license setup, 25
loop joint.ino, 107
Lowpass, 80 K, 81
Lowpass, 80
resetState, 80
tau, 81
Ts, 81
updateState, 80
x, 81

M MovMax, 92
M_PI uTransmission.h, 146
main joint_comm_node.cpp, 148
maintainer setup, 25
maintainer_email setup, 25
max bioscara.hardware_driver::Gripper, 61
bioscara.hardware_driver::Joint, 76
bioscara.hardware_interface::BioscaraHardwareInterface::joint_config
78
max_acceleration bioscara.hardware_interface::BioscaraHardwareInterface::joint_config
78
MAX_BUFFER joint.h, 102
ul2C.h, 139
max_velocity

bioscara_hardware_interface::BioscaraHardwareInterfacePrivate
 bioscara_hardware_interface::BioscaraHardwareInterfacePrivateConfig_t,
 bioscara_hardware_interface::BioscaraHardwareInterface, 48
MAXACCEL
 configuration.h, 96
MAXVEL
 configuration.h, 96
min
 bioscara_hardware_driver::Gripper, 61
 bioscara_hardware_driver::Joint, 76
 bioscara_hardware_interface::BioscaraHardwareInterface, 78
MockGripper
 bioscara_hardware_driver::MockGripper, 82
MockJoint
 bioscara_hardware_driver::MockJoint, 86
MOVEANGLE
 bioscara_hardware_driver::BaseJoint, 33
 joint.h, 102
MOVESTEPS
 bioscara_hardware_driver::BaseJoint, 33
 joint.h, 102
moveSteps
 bioscara_hardware_driver::BaseJoint, 39
 bioscara_hardware_driver::Joint, 70
MOVETOANGLE
 bioscara_hardware_driver::BaseJoint, 33
 joint.h, 102
MOVETOEND
 bioscara_hardware_driver::BaseJoint, 33
 joint.h, 103
MovMax, 91
 cb_data, 92
 cb_index, 92
 M, 92
 MovMax, 91
 updateState, 91
NACK
 joint.h, 102
 ul2C.h, 139
name
 bioscara_hardware_driver::BaseJoint, 44
 setup, 25
non_blocking_handler
 joint.ino, 107
NONE
 bioscara_hardware_driver::BaseJoint, 33
NOT_ENABLED
 bioscara_hardware_driver, 22
NOT_HOMED
 bioscara_hardware_driver, 22
NOT_INIT
 bioscara_hardware_driver, 22
offset
 bioscara_hardware_driver::Gripper, 61
 bioscara_hardware_driver::Joint, 76
OK
 bioscara_hardware_driver, 22
on_cleanup
 bioscara_hardware_interface::BioscaraHardwareInterface, 48
on_configure
 bioscara_hardware_interface::BioscaraHardwareInterface, 49
on_error
 bioscara_hardware_interface::BioscaraHardwareInterface, 50
on_init
 bioscara_hardware_interface::BioscaraHardwareInterface, 50
on_shutdown
 bioscara_hardware_interface::BioscaraHardwareInterface, 51
op_mode
 bioscara_hardware_driver::MockJoint, 90
openI2CDevHandle
 ul2C.cpp, 155
 ul2C.h, 140
package_name
 setup, 25
packages
 setup, 25
per
 RPI_PWM, 94
perform_command_mode_switch
 bioscara_hardware_interface::BioscaraHardwareInterface, 52
PING
 bioscara_hardware_driver::BaseJoint, 33
 joint.h, 102
postHoming
 bioscara_hardware_driver::BaseJoint, 40
 bioscara_hardware_driver::Joint, 71
prepare_command_mode_switch
 bioscara_hardware_interface::BioscaraHardwareInterface, 53
pwm
 bioscara_hardware_driver::Gripper, 61
pwmpath
 RPI_PWM, 94
q
 bioscara_hardware_driver::MockJoint, 90
qd
 bioscara_hardware_driver::MockJoint, 91
RAD2DEG
 uTransmission.h, 147
read
 bioscara_hardware_driver::Joint, 71

bioscara_hardware_interface::BioscaraHardwareInterface
53
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include
126, 128

readFromI2CDev
ul2C.cpp, 156
ul2C.h, 140
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include
129, 131

README, 3, 5, 7, 9
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include
131, 133

readValue
joint.h, 103
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include
134, 136

receiveEvent
joint.ino, 108
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include
137, 141

reduction
bioscara_hardware_driver::Gripper, 61
bioscara_hardware_driver::Joint, 76
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include
145fig, 147
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/joint
78
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/joint
142, 144

reg
joint.ino, 109
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/motor
149

requestEvent
joint.ino, 108
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/motor
150

resetState
Lowpass, 80
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/motor
150

RETURN_ON_ERROR
uErr.h, 135
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/motor
151

RETURN_ON_FALSE
uErr.h, 135
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/motor
152

RETURN_ON_NEGATIVE
uErr.h, 136
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/motor
152

RFLAGS_SIZE
joint.h, 102
ul2C.h, 139
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/motor
153

ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/motor
111
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include
112
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include
112
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_controller
ROS2/ros2_scara_ws/src/dalsa_bioscara/README.md,
112
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description
RPI_PWM, 92
description/ __init__.py,
~RPI_PWM, 93
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description
hardware/disabled.launch.py,
112
enable, 93
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description
hardware/gazebo.launch.py,
113
per, 94
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description
README.md,
112
setDutyCycle, 93
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description
setDutyCycleNS, 93
113
setPeriod, 93
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver
driver/include/bioscara_hardware_driver/common.h,
113, 115
stop, 94
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver
driver/include/bioscara_hardware_driver/mBaseGripper.h,
115, 117
RUNCOTINOUS
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver
driver/include/bioscara_hardware_driver/BaseJoint.h,
118, 119
joint.h, 102
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver
driver/include/bioscara_hardware_driver/mGripper.h,
121, 122
joint.ino, 109
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver
driver/include/bioscara_hardware_driver/mJoint.h,
123, 124
joint.ino, 109

rx_length
 joint.ino, 109

SETBRAKEMODE
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103

setBrakeMode
 bioscara.hardware_driver::BaseJoint, 40
 bioscara.hardware_driver::Joint, 72

SETCONTROLTRESHOLD
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103

SETCURRENT
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103

setDriveCurrent
 bioscara.hardware_driver::BaseJoint, 40
 bioscara.hardware_driver::Joint, 72

setDutyCycle
 RPI_PWM, 93

setDutyCycleNS
 RPI_PWM, 93

SETHOLDCURRENT
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103

setHoldCurrent
 bioscara.hardware_driver::BaseJoint, 41
 bioscara.hardware_driver::Joint, 72

setHomingOffset
 bioscara.hardware_driver::Joint, 73

SETMAXACCELERATION
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103

setMaxAcceleration
 bioscara.hardware_driver::BaseJoint, 41
 bioscara.hardware_driver::Joint, 73

SETMAXDECELERATION
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103

SETMAXVELOCITY
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103

setMaxVelocity
 bioscara.hardware_driver::BaseJoint, 42
 bioscara.hardware_driver::Joint, 74

setOffset
 bioscara.hardware_driver::BaseGripper, 29
 bioscara.hardware_driver::Gripper, 59

setPeriod
 RPI_PWM, 93

setPosition
 bioscara.hardware_driver::BaseGripper, 29
 bioscara.hardware_driver::BaseJoint, 42
 bioscara.hardware_driver::Gripper, 60
 bioscara.hardware_driver::Joint, 74
 bioscara.hardware_driver::MockJoint, 89

setReduction
 bioscara.hardware_driver::BaseGripper, 29
 bioscara.hardware_driver::Gripper, 60

SETRPM
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 102

setServoPosition
 bioscara.hardware_driver::BaseGripper, 29
 bioscara.hardware_driver::Gripper, 60

SETUP
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 102

setup, 24
 data_files, 24
 description, 24
 entry_points, 24
 install_requires, 24
 joint.ino, 108
 license, 25
 maintainer, 25
 maintainer_email, 25
 name, 25
 package_name, 25
 packages, 25
 tests_require, 25
 version, 25
 zip_safe, 25

setVelocity
 bioscara.hardware_driver::BaseJoint, 42
 bioscara.hardware_driver::Joint, 74
 bioscara.hardware_driver::MockJoint, 89

speed
 bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing, 79

split_interface_string_to_joint_and_name
 bioscara.hardware_interface::BioscaraHardwareInterface, 54

stall.h
 stall_threshold, 111

stall_threshold
 bioscara.hardware_interface::BioscaraHardwareInterface::joint_configuration, 78
 stall.h, 111

STALLED
 bioscara.hardware_driver, 22

start
 RPI_PWM, 93

start_homing
 bioscara.hardware_interface::BioscaraHardwareInterface, 54

startHoming
 bioscara.hardware_driver::BaseJoint, 43

stepper
 joint.ino, 109

STOP
 bioscara.hardware_driver::BaseJoint, 33
 joint.h, 103

stop
 bioscara.hardware_driver::BaseJoint, 43
 bioscara.hardware_driver::Joint, 75
 bioscara.hardware_driver::MockJoint, 90

RPI_PWM, 94
stop_homing
 bioscara.hardware_interface::BioscaraHardwareInterface, 55
stp_reg_t
 bioscara.hardware_driver::BaseJoint, 32
 joint.h, 102
tau
 Lowpass, 81
test_joint_trajectory_controller, 25
 generate_launch_description, 25
tests_require
 setup, 25
threshold
 bioscara.hardware_interface::BioscaraHardwareInterface::joint_homing_config_t, 79
Todo List, 11
Ts
 Lowpass, 81
tx_buf
 joint.ino, 109
tx_length
 joint.ino, 109

uErr.h
 RETURN_ON_ERROR, 135
 RETURN_ON_FALSE, 135
 RETURN_ON_NEGATIVE, 136
ul2C.cpp
 closeI2CDevHandle, 155
 openI2CDevHandle, 155
 readFromI2CDev, 156
 writeToI2CDev, 156
ul2C.h
 ACK, 139
 closeI2CDevHandle, 140
 MAX_BUFFER, 139
 NACK, 139
 openI2CDevHandle, 140
 readFromI2CDev, 140
 RFLAGS_SIZE, 139
 writeToI2CDev, 141
updateState
 Lowpass, 80
 MovMax, 91
uTransmission.h
 ACTUATOR2JOINT, 146
 DEG2RAD, 146
 JOINT2ACTUATOR, 146
 M_PI, 146
 RAD2DEG, 147

version
 setup, 25

wait_while_busy
 bioscara.hardware_driver::BaseJoint, 43
write