Bioscara

Generated by Doxygen 1.9.8

1 Documentation	1
1.1 Usage	1
2 README	3
3 README	5
4 README	7
5 Todo List	9
6 Namespace Index	11
6.1 Namespace List	11
7 Hierarchical Index	13
7.1 Class Hierarchy	13
8 Class Index	15
8.1 Class List	15
9 File Index	17
9.1 File List	17
10 Namespace Documentation	19
10.1 bioscara Namespace Reference	19
10.1.1 Function Documentation	
10.1.1.1 generate_launch_description()	
10.2 bioscara_hardware_interface Namespace Reference	
10.2.1 Variable Documentation	
10.2.1.1 HW_IF_HOME	
10.3 display Namespace Reference	20
10.3.1 Function Documentation	20
10.3.1.1 generate_launch_description()	
10.4 gazebo Namespace Reference	
10.4.1 Function Documentation	
10.4.1.1 generate_launch_description()	20
10.5 setup Namespace Reference	
10.5.1 Variable Documentation	21
10.5.1.1 data_files	21
10.5.1.2 description	21
10.5.1.3 entry_points	21
10.5.1.4 install_requires	21
10.5.1.5 license	21
10.5.1.6 maintainer	21
10.5.1.7 maintainer_email	
10.5.1.8 name	21

10.5.1.9 package_name	 21
10.5.1.10 packages	 21
10.5.1.11 tests_require	 22
10.5.1.12 version	 22
10.5.1.13 zip_safe	 22
10.6 test_copyright Namespace Reference	 22
10.6.1 Function Documentation	 22
10.6.1.1 test_copyright()	 22
10.7 test_flake8 Namespace Reference	 22
10.7.1 Function Documentation	 22
10.7.1.1 test_flake8()	 22
10.8 test_joint_trajectory_controller Namespace Reference	 22
10.8.1 Function Documentation	 23
10.8.1.1 generate_launch_description()	 23
10.9 test_pep257 Namespace Reference	 23
10.9.1 Function Documentation	 23
10.9.1.1 test_pep257()	 23
11 Class Documentation	25
11.1 bioscara_hardware_interface::BioscaraHardwareInterface Class Reference	25
11.1.1 Detailed Description	27
11.1.2 Member Function Documentation	27
11.1.2.1 on_activate()	27
11.1.2.2 on_cleanup()	28
11.1.2.3 on_configure()	28
11.1.2.4 on_deactivate()	29
11.1.2.5 on_error()	 29
11.1.2.6 on_init()	 30
11.1.2.7 on_shutdown()	 30
11.1.2.8 prepare_command_mode_switch()	 31
11.1.2.9 read()	 32
11.1.2.10 write()	 32
11.1.3 Member Data Documentation	 33
11.1.3.1 _joint_cfg	 33
11.1.3.2 _joint_command_modes	 33
11.1.3.3 _joints	 34
11.2 Gripper Class Reference	 34
11.2.1 Detailed Description	 35
11.2.2 Constructor & Destructor Documentation	 35
11.2.2.1 Gripper()	 35
11.2.3 Member Function Documentation	 35
11.2.3.1 deinit()	 35

11.2.3.2 disable()		36
11.2.3.3 enable()		36
11.2.3.4 init()		36
11.2.3.5 setPosition()		36
11.2.4 Member Data Documentation		37
11.2.4.1 pwm		37
11.3 Joint Class Reference		37
11.3.1 Detailed Description		39
11.3.2 Member Enumeration Documentation		40
11.3.2.1 stp_reg_t		40
11.3.3 Constructor & Destructor Documentation		41
11.3.3.1 Joint()		41
11.3.3.2 ~ Joint()		42
11.3.4 Member Function Documentation		42
11.3.4.1 _home()		42
11.3.4.2 checkCom()		43
11.3.4.3 checkOrientation()		43
11.3.4.4 deinit()		43
11.3.4.5 disable()		44
11.3.4.6 disableCL()		44
11.3.4.7 enable()		44
11.3.4.8 enableStallguard()		44
11.3.4.9 getCurrentBCmd()		45
11.3.4.10 getFlags()		45
11.3.4.11 getHomingOffset()		45
11.3.4.12 getPosition()		45
11.3.4.13 getVelocity()		46
11.3.4.14 home()		46
11.3.4.15 init()		46
11.3.4.16 isBusy()		47
11.3.4.17 isEnabled()		47
11.3.4.18 isHomed()		47
11.3.4.19 isStalled()		47
11.3.4.20 moveSteps()		48
11.3.4.21 postHoming()		49
11.3.4.22 printlnfo()		49
11.3.4.23 read()		49
11.3.4.24 setBrakeMode()		50
11.3.4.25 setDriveCurrent()		50
11.3.4.26 setHoldCurrent()		50
11.3.4.27 setHomingOffset()		51
11.3.4.28 setMayAcceleration()		51

11.3.4.29 setMaxVelocity()	51
11.3.4.30 setPosition()	52
11.3.4.31 setVelocity()	52
11.3.4.32 startHoming()	52
11.3.4.33 stop()	53
11.3.4.34 wait_while_busy()	53
11.3.4.35 write()	53
11.3.5 Member Data Documentation	54
11.3.5.1 address	54
11.3.5.2 current_b_cmd	54
11.3.5.3 flags	54
11.3.5.4 handle	55
11.3.5.5 max	55
11.3.5.6 min	55
11.3.5.7 name	55
11.3.5.8 offset	55
11.3.5.9 reduction	55
11.4 bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t Struct Reference	56
11.4.1 Detailed Description	56
11.4.2 Member Data Documentation	56
11.4.2.1 drive_current	56
11.4.2.2 hold_current	57
11.4.2.3 homing	57
11.4.2.4 i2c_address	57
11.4.2.5 max	57
11.4.2.6 max_acceleration	57
11.4.2.7 max_velocity	57
11.4.2.8 min	57
11.4.2.9 reduction	57
11.4.2.10 stall_threshold	57
$11.5\ bioscara_hardware_interface:: BioscaraHardwareInterface:: joint_homing_config_t\ Struct\ Reference\ .$	58
11.5.1 Detailed Description	58
11.5.2 Member Data Documentation	58
11.5.2.1 acceleration	58
11.5.2.2 current	58
11.5.2.3 speed	58
11.5.2.4 threshold	58
11.6 RPI_PWM Class Reference	59
11.6.1 Detailed Description	59
11.6.2 Constructor & Destructor Documentation	59
11.6.2.1 ∼RPI_PWM()	59
11.6.3 Member Function Documentation	59

11.6.3.1 disable()	. 59
11.6.3.2 enable()	. 59
11.6.3.3 setDutyCycle()	. 59
11.6.3.4 setDutyCycleNS()	. 60
11.6.3.5 setPeriod()	. 60
11.6.3.6 start()	. 60
11.6.3.7 stop()	. 60
11.6.3.8 writeSYS()	. 60
11.6.4 Member Data Documentation	. 61
11.6.4.1 chippath	. 61
11.6.4.2 per	. 61
11.6.4.3 pwmpath	. 61
12 File Documentation	63
12.1 Arduino/joint/configuration.h File Reference	
12.1.1 Detailed Description	
12.1.2 Macro Definition Documentation	
12.1.2.1 ADR	
12.1.2.2 MAXACCEL	
12.1.2.3 MAXVEL	
12.2 configuration.h	
12.3 Arduino/joint/joint.h File Reference	
12.3.1 Detailed Description	
12.3.2 Macro Definition Documentation	
12.3.2.1 ACK	
12.3.2.2 DUMP BUFFER	
12.3.2.3 MAX_BUFFER	
12.3.2.4 NACK	. 68
12.3.2.5 RFLAGS_SIZE	
12.3.3 Enumeration Type Documentation	. 68
12.3.3.1 stp_reg_t	. 68
12.3.4 Function Documentation	. 69
12.3.4.1 readValue()	. 69
12.3.4.2 writeValue()	. 69
12.4 joint.h	. 70
12.5 Arduino/joint/joint.ino File Reference	. 71
12.5.1 Detailed Description	. 72
12.5.2 Macro Definition Documentation	. 72
12.5.2.1 J4	. 72
12.5.3 Function Documentation	. 72
12.5.3.1 blocking_handler()	. 72
12 5 3 2 loop()	73

12.5.3.3 non_blocking_handler()	73
12.5.3.4 receiveEvent()	73
12.5.3.5 requestEvent()	75
12.5.3.6 setup()	75
12.5.4 Variable Documentation	75
12.5.4.1 reg	75
12.5.4.2 rx_buf	75
12.5.4.3 rx_data_ready	75
12.5.4.4 rx_length	75
12.5.4.5 stepper	76
12.5.4.6 tx_buf	76
12.5.4.7 tx_data_ready	76
12.5.4.8 tx_length	76
12.6 docs/DOCS_README.md File Reference	76
12.7 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup/launch/bioscara.launch.py File Reference	76
12.8 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup/launch/test_joint_trajectory_controller.launchy File Reference	<mark>ch.</mark> ∉ 76
12.9 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup/README.md File Reference	77
12.10 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/README.md File Reference	77
12.11 ROS2/ros2_scara_ws/src/dalsa_bioscara/README.md File Reference	77
12.12 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/bioscara_description/initpy File Reference	77
12.13 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/launch/display.launch.py File Reference	77
12.14 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/launch/gazebo.launch.py File Reference	77
12.15 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/setup.py File Reference	77
12.16 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/test/test_copyright.py File Reference	78
12.17 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/test/test_flake8.py File Reference .	78
12.18 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/test/test_pep257.py File Reference	78
12.19 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware ↔	70
_driver/common.h File Reference	79 70
12.19.2 Macro Definition Documentation	79 80
12.19.2 Macro Delinition Documentation	80
12.20 common.h	80
12.21 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware ←driver/mGripper.h File Reference	
12.21.1 Detailed Description	81 82
12.22 mGripper.h	
	82
12.23 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware ←driver/mJoint.h File Reference	83
12.23.1 Detailed Description	84
12.23.2 Macro Definition Documentation	84

12.23.2.1 ACTUATOR2JOINT	84
12.23.2.2 DEG2RAD	85
12.23.2.3 JOINT2ACTUATOR	85
12.23.2.4 M_PI	85
12.23.2.5 RAD2DEG	85
12.24 mJoint.h	85
12.25 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware ←driver/mJoint.hpp File Reference	87
12.25.1 Detailed Description	88
12.26 mJoint.hpp	89
12.27 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware ←driver/ul2C.h File Reference	
12.27.1 Detailed Description	91
12.27.2 Macro Definition Documentation	91
12.27.2.1 ACK	91
12.27.2.2 MAX_BUFFER	92
12.27.2.3 NACK	92
12.27.2.4 RFLAGS_SIZE	92
12.27.3 Function Documentation	92
12.27.3.1 closel2CDevHandle()	92
12.27.3.2 openI2CDevHandle()	92
12.27.3.3 readFromI2CDev()	93
12.27.3.4 writeToI2CDev()	93
12.28 ul2C.h	94
12.29 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware ←driver/uPWM.h File Reference	94
12.29.1 Detailed Description	95
12.30 uPWM.h	96
12.31 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/joint_comm_node.cpp File Reference	97
12.31.1 Function Documentation	98
12.31.1.1 INT_handler()	98
12.31.1.2 main()	98
12.31.2 Variable Documentation	99
12.31.2.1 J1	99
12.31.2.2 J2	99
12.31.2.3 J3	99
12.31.2.4 J4	99
12.32 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mGripper.cpp File Reference	e 99
12.33 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mJoint.cpp File Reference	100
12.34 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/ul2C.cpp File Reference .	101
12.34.1 Function Documentation	101
12 34 1 1 closel2CDevHandle()	101

	12.34.1.2 openI2CDevHandle()	102
	12.34.1.3 readFromI2CDev()	102
	12.34.1.4 writeToI2CDev()	102
12.35 hardwa	ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include/bioscara_care_interface/bioscara_hardware.hpp File Reference	103
12.36 biosca	ra_hardware.hpp	104
	ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/src/bioscara_hardware.cpp	106
Index		107

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.

2 Documentation

README

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

4 README

README

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

6 README

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

```
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
)
install(TARGETS ros2_control_demo_example_1
ARCHIVE DESTINATION lib
LIBRARY DESTINATION lib
RUNTIME DESTINATION bin
```

TODO:

• [] Format and rework this content

8 README

Todo List

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.

Member Joint::read (const stp_reg_t reg, T &data, u_int8_t &flags)

Implement a return code for read only functions

· Implement clearStall function

10 Todo List

Namespace Index

6.1 Namespace List

Here is a list of all namespaces with brief descriptions:

oscara
oscara_hardware_interface
splay
zebo
tup
st_copyright
st_flake8
st_joint_trajectory_controller
st pep257

12 Namespace Index

Hierarchical Index

7.1 Class Hierarchy

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

Gripper	34
Joint	37
bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t	56
bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing_config_t	58
RPI_PWM	59
hardware_interface::SystemInterface	
bioscara hardware interface::BioscaraHardwareInterface	25

14 Hierarchical Index

Class Index

8.1 Class List

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

bioscara_	_hardware_interface::BioscaraHardwareInterface	
	The bioscara hardware interface class	25
Gripper		
	Gripper object to interact with the robot gripper	34
Joint		
	Representing a single joint on the I2C bus	37
bioscara_	_hardware_interface::BioscaraHardwareInterface::joint_config_t	
	Configuration structure holding the passed paramters from the ros2_control urdf	56
bioscara_	_hardware_interface::BioscaraHardwareInterface::joint_homing_config_t	
	Configuration structure holding the passed homing paramters from the ros2_control urdf	58
RPI_PWI	M	
	PWM class for the Raspherry PL4 and 5	59

16 Class Index

File Index

9.1 File List

Here is a list of all files with brief descriptions:

Arduino/joint/configuration.h	
Configuration definitions for Joint 1 to Joint 4	63
Arduino/joint/joint.h	
Joint firmware header	65
Arduino/joint/joint.ino	
Joint firmware	71
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup/launch/bioscara.launch.py	76
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup/launch/test_joint_trajectory_controller.launch.py	•
76	
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/setup.py	77
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/bioscara_description/initpy	77
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/launch/display.launch.py	77
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/launch/gazebo.launch.py	77
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/test/test_copyright.py	78
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/test/test_flake8.py	78
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/test/test_pep257.py	78
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_←	
driver/common.h	
A file containing utility macros and functions	79
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_←	
driver/mGripper.h	
File containing the Gripper class	81
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_←	
driver/mJoint.h	
File including the Joint class	83
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_←	
driver/mJoint.hpp	
Templated functions for the Joint class	87
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_←	
driver/ul2C.h	
Low level utility for I2C communication on Raspberry Pi using Igpio library	89
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_←	
driver/uPWM.h	
Includes source code for Hardware PWM generation on Raspberry Pi 4	94
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/joint_comm_node.cpp	97

18 File Index

ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mGripper.cpp	99
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mJoint.cpp	100
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/ul2C.cpp	101
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include/bioscara_hardware_	
interface/bioscara_hardware.hpp	103
ROS2/ros2 scara ws/src/dalsa bioscara/bioscara hardware interface/src/bioscara hardware.cpp	106

Namespace Documentation

10.1 bioscara Namespace Reference

Functions

• generate_launch_description ()

10.1.1 Function Documentation

10.1.1.1 generate_launch_description()

bioscara.generate_launch_description ()

10.2 bioscara_hardware_interface Namespace Reference

Classes

· class BioscaraHardwareInterface

The bioscara hardware interface class.

Variables

• constexpr char HW_IF_HOME [] = "home"

10.2.1 Variable Documentation

10.2.1.1 HW_IF_HOME

constexpr char bioscara_hardware_interface::HW_IF_HOME[] = "home" [constexpr]

10.3 display Namespace Reference

Functions

• generate_launch_description ()

10.3.1 Function Documentation

10.3.1.1 generate_launch_description()

```
display.generate_launch_description ( )
```

10.4 gazebo Namespace Reference

Functions

• generate_launch_description ()

10.4.1 Function Documentation

10.4.1.1 generate_launch_description()

```
gazebo.generate_launch_description ( )
```

10.5 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

10.5.1 Variable Documentation

10.5.1.1 data_files

setup.data_files

10.5.1.2 description

setup.description

10.5.1.3 entry_points

setup.entry_points

10.5.1.4 install_requires

setup.install_requires

10.5.1.5 license

setup.license

10.5.1.6 maintainer

setup.maintainer

10.5.1.7 maintainer_email

setup.maintainer_email

10.5.1.8 name

setup.name

10.5.1.9 package_name

str setup.package_name = 'bioscara_description'

10.5.1.10 packages

setup.packages

10.5.1.11 tests_require

```
setup.tests_require
```

10.5.1.12 version

setup.version

10.5.1.13 zip_safe

setup.zip_safe

10.6 test_copyright Namespace Reference

Functions

• test_copyright ()

10.6.1 Function Documentation

10.6.1.1 test_copyright()

```
test_copyright.test_copyright ( )
```

10.7 test_flake8 Namespace Reference

Functions

• test_flake8 ()

10.7.1 Function Documentation

10.7.1.1 test_flake8()

```
test_flake8.test_flake8 ( )
```

10.8 test_joint_trajectory_controller Namespace Reference

Functions

• generate_launch_description ()

10.8.1 Function Documentation

10.8.1.1 generate_launch_description()

```
test_joint_trajectory_controller.generate_launch_description ( )
```

10.9 test_pep257 Namespace Reference

Functions

test_pep257 ()

10.9.1 Function Documentation

10.9.1.1 test_pep257()

```
test_pep257.test_pep257 ( )
```

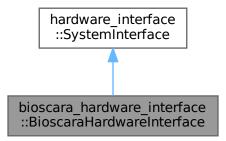
Class Documentation

11.1 bioscara_hardware_interface::BioscaraHardwareInterface Class Reference

The bioscara hardware interface class.

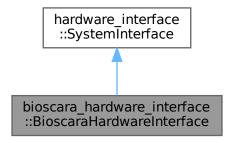
#include <bioscara_hardware.hpp>

Inheritance diagram for bioscara_hardware_interface::BioscaraHardwareInterface:



26 Class Documentation

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_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 transistion from the unconfigured to the inactive state.
- hardware_interface::CallbackReturn on_cleanup (const rclcpp_lifecycle::State &previous_state) override Called on the transistion from the inactive to the unconfigured state.
- hardware_interface::CallbackReturn on_activate (const rclcpp_lifecycle::State &previous_state) override Called on the transistion from the inactive to the active state.
- hardware_interface::CallbackReturn on_deactivate (const rclcpp_lifecycle::State &previous_state) override Called on the transistion 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::CallbackReturn on_error (const rclcpp_lifecycle::State &previous_state) override Called when an error in any state or state transition is thrown.

Private Attributes

- std::unordered_map < std::string, Joint > _joints unordered map storing the Joint objects.
- 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.

11.1.1 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 lifecyle which intern is following the ROS2 managed node lifecycle.

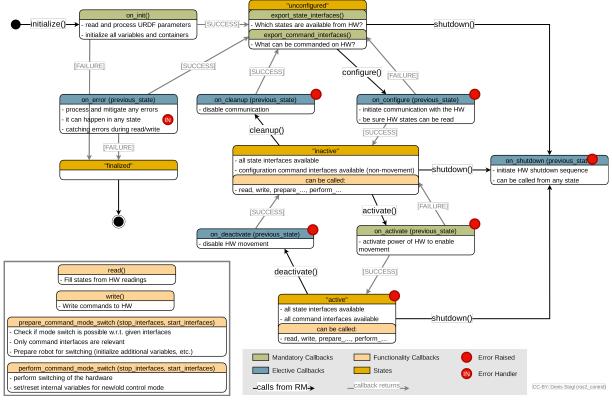


Figure 11.1 Hardware interface lifecycle

11.1.2 Member Function Documentation

11.1.2.1 on_activate()

hardware_interface::CallbackReturn bioscara_hardware_interface::BioscaraHardwareInterface← ::on_activate (

const rclcpp_lifecycle::State & previous_state) [override]

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

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

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

```
previous_state
```

Returns

hardware interface::CallbackReturn

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

11.1.2.2 on_cleanup()

Called on the transistion from the inactive to the unconfigured state.

Disconnect from the joints.

Parameters

```
previous state
```

Returns

hardware interface::CallbackReturn

Disconnect from the joints and throw error if it fails

11.1.2.3 on_configure()

Called on the transistion from the unconfigured to the inactive state.

Establish and test connection to each joint.

Parameters

previous_state

Returns

hardware_interface::CallbackReturn

11.1.2.4 on deactivate()

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

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

Parameters

previous_state

Returns

hardware_interface::CallbackReturn

disable the joints and throw error if it fails

11.1.2.5 on_error()

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 <code>UNCONFIGURED</code> state, if any ERROR or FAILURE happens the hardware ends in <code>FINALIZED</code> 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 immediatly 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, disableing the joints to allow backdriving.

Parameters

```
previous state
```

Returns

hardware_interface::CallbackReturn

11.1.2.6 on init()

Loop over all joints decribed 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

Loop over all GPIOs decribed in the hardware description file, check if they have the home command and state interface defined.

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

11.1.2.7 on_shutdown()

Called on the transistion 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

previous_state

Returns

hardware interface::CallbackReturn

11.1.2.8 prepare command mode switch()

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 furture 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.

Parameters

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

Returns

hardware interface::return type

11.1.2.9 read()

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 immediatly before the return flags are assumed to be valid.
 - If the 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

time	
period	

Returns

hardware_interface::return_type

11.1.2.10 write()

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 the 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 polulated 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

time	
period	

Returns

hardware_interface::return_type

11.1.3 Member Data Documentation

11.1.3.1 _joint_cfg

std::unordered_map<std::string, joint_config_t> bioscara_hardware_interface::BioscaraHardware← Interface::_joint_cfg [private]

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.

11.1.3.2 _joint_command_modes

std::unordered_map<std::string, std::set<std::string> > bioscara_hardware_interface::Bioscara←
HardwareInterface::_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.

11.1.3.3 _joints

std::unordered_map<std::string, Joint> bioscara_hardware_interface::BioscaraHardwareInterface← ::_joints [private]

unordered map storing the Joint objects.

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.

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

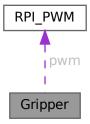
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include/bioscara_hardware_
 interface/bioscara hardware.hpp
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/src/bioscara_hardware.cpp

11.2 Gripper Class Reference

Gripper object to interact with the robot gripper.

#include <mGripper.h>

Collaboration diagram for Gripper:



Public Member Functions

- · Gripper (void)
- int init (void)

Placeholder, does nothing.

• int deinit (void)

Placeholder, does nothing.

• int enable (void)

Prepares the servo for use.

• int disable (void)

Disables the servo.

• int setPosition (float width)

Sets the gripper width in mm from the closed position.

Private Attributes

RPI_PWM pwm

11.2.1 Detailed Description

Gripper object to interact with the robot gripper.

This class is a wrapper function to interact with a PWM servo 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"
int main(int argc, char **argv)
{
    Gripper 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;
```

11.2.2 Constructor & Destructor Documentation

11.2.2.1 Gripper()

```
Gripper::Gripper (
     void )
```

11.2.3 Member Function Documentation

11.2.3.1 deinit()

```
int Gripper::deinit (
     void )
```

Placeholder, does nothing.

Returns

0

11.2.3.2 disable()

Disables the servo.

Stops the servo and disables the PWM generation.

Returns

non-zero error code.

11.2.3.3 enable()

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. \ast

Returns

non-zero error code.

11.2.3.4 init()

Placeholder, does nothing.

Returns

0

11.2.3.5 setPosition()

Sets the gripper width in mm from the closed position.

Arguments outside the allowed range are bounded to limit.

Parameters

width width in mm. 30 - 85 mm are currently allowed. With a new gripper this should be changed.

11.2.4 Member Data Documentation

11.2.4.1 pwm

```
RPI_PWM Gripper::pwm [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/mGripper.h
- ROS2/ros2 scara ws/src/dalsa bioscara/bioscara hardware driver/src/mGripper.cpp

11.3 Joint Class Reference

Representing a single joint on the I2C bus.

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

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

- Joint (const std::string name, const int address, const float reduction, const float min, const float max)
 Create a Joint object.
- ∼Joint (void)
- int init (void)

Established connection to a joint via I2C.

· int deinit (void)

Disconnects from a joint.

int enable (u_int8_t driveCurrent, u_int8_t holdCurrent)

Setup the joint and engages motor.

· int disable (void)

disenganges the joint motor without closing i2c handle

int home (float velocity, u_int8_t sensitivity, u_int8_t current)

Blocking implementation to home the joint.

• int startHoming (float velocity, u_int8_t sensitivity, u_int8_t current)

non-blocking implementation to home the joint.

int postHoming (void)

perform tasks after a non-blocking homing.

- int printInfo (void)
- int getPosition (float &pos)

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

• int setPosition (float pos)

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

int moveSteps (int32_t steps)

Move full steps.

int getVelocity (float &vel)

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

int setVelocity (float vel)

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

• int checkOrientation (float angle=10.0)

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

• int stop (void)

Stops the motor.

· int disableCL (void)

Disables the Closed-Loop PID Controller.

• int setDriveCurrent (u_int8_t current)

Set the Drive Current.

• int setHoldCurrent (u_int8_t current)

Set the Hold Current.

• int setBrakeMode (u int8 t mode)

Set Brake Mode.

• int setMaxAcceleration (float maxAccel)

Set the maximum permitted joint acceleration (and deceleration) in rad/s^2 or rad/s^2 for cylindrical and prismatic joints respectively.

int setMaxVelocity (float maxVel)

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

int enableStallguard (u_int8_t sensitivity)

Enable encoder stall detection of the joint.

bool isHomed (void)

Checks the state if the motor is homed.

bool isEnabled (void)

Checks the state if the motor is enabled.

bool isStalled (void)

Checks if the motor is stalled.

• bool isBusy (void)

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

int checkCom (void)

Check if communication to the joint is established.

- u int8 t getFlags (void)
- int getHomingOffset (float &offset)

Retrieves the homing position from the last homing.

int setHomingOffset (const float offset)

Stores the homing position on the joint.

stp_reg_t getCurrentBCmd (void)

get the currently active blocking command

Public Attributes

• std::string name

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.

void wait_while_busy (const float period_ms)

Blocking loop waiting for BUSY flag to reset.

• int _home (float velocity, u_int8_t sensitivity, u_int8_t current)

Call to start the homing sequence of a joint.

Private Attributes

```
• u_int8_t flags = 0x00
```

State flags transmitted with every I2C transaction.

· int address

I2C adress.

• 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.

• stp_reg_t current_b_cmd = NONE

Keeps track if a blocking command is being executed.

• int handle = -1

I2C bus handle.

11.3.1 Detailed Description

Representing a single joint on the I2C bus.

11.3.2 Member Enumeration Documentation

11.3.2.1 stp_reg_t

enum Joint::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) -].

11.3.3 Constructor & Destructor Documentation

11.3.3.1 Joint()

Create a Joint object.

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

actuator position = (joint position - offset) * reduction joint position = actuator position / reduction + offset

Parameters

name	string device name for identification
address	1-byte I2C device adress (0x11 0x14) for J1 J4
reduction	gear reduction of the joint. This is used to transform position and velocity values between in 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 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
min	lower joint limit in joint units. J1: -3.04647 J2: -0.0016 J3: -2.62672 J4: -3.01069
max	upper joint limit in joint units. J1: 3.04647 J2: 0.3380 J3: 2.62672 J4: 3.01069

11.3.3.2 \sim Joint()

```
Joint::∼Joint (
void )
```

11.3.4 Member Function Documentation

11.3.4.1 _home()

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

velocity	signed velocity in rad/s or m/s. Must be between 1.0 < RAD2DEG(JOINT2ACTUATOR(velocity, reduction, 0)) / 6 < 250.0
sensitivity	Encoder pid error threshold 0 to 255.
current	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.

11.3.4.2 checkCom()

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.

11.3.4.3 checkOrientation()

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 this function returns when the isBusy flag is clear again.

Parameters

angle 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.

11.3.4.4 deinit()

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.

11.3.4.5 disable()

disenganges the joint motor without closing i2c handle

Returns

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

11.3.4.6 disableCL()

Disables the Closed-Loop PID Controller.

Returns

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

11.3.4.7 enable()

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 ets 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

driveCurrent	drive current in 0-100 % of 2.5A output (check uStepper doc.)
holdCurrent	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.

11.3.4.8 enableStallguard()

Enable encoder stall detection of the joint.

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 by reenabling the stall guard.

Parameters

	thresholds	value of threshold. 0 - 255 where lower is more sensitive.	1
--	------------	------------------------------------------------------------	---

Returns

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

11.3.4.9 getCurrentBCmd()

get the currently active blocking command

Returns

The the command of type stp_reg_t

11.3.4.10 getFlags()

get driver state flags

Returns

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

11.3.4.11 getHomingOffset()

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.

11.3.4.12 getPosition()

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

```
pos
```

Returns

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

11.3.4.13 getVelocity()

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

Parameters



Returns

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

11.3.4.14 home()

Blocking implementation to home the joint.

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()).

11.3.4.15 init()

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.

11.3.4.16 isBusy()

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.

11.3.4.17 isEnabled()

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.

11.3.4.18 isHomed()

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.

11.3.4.19 isStalled()

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.

11.3.4.20 moveSteps()

Move full steps.

This function can be called even when not homed.

Parameters

steps	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.

11.3.4.21 postHoming()

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.

11.3.4.22 printlnfo()

11.3.4.23 read()

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

T Datatype of value to be transmitted

Parameters

ſ	reg	stp_reg_t register to read
	data	reference to store payload.
Ī	flags	reference to a byte which stores the return flags

Returns

0 on OK, negative on error

11.3.4.24 setBrakeMode()

Set Brake Mode.

Parameters

mode Freewheel: 0, Coolbrake: 1, Hard	lbrake: 2
---------------------------------------	-----------

Returns

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

11.3.4.25 setDriveCurrent()

Set the Drive Current.

Warning

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

Parameters

```
current 0% - 100% of driver current
```

Returns

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

11.3.4.26 setHoldCurrent()

Set the Hold Current.

Warning

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

Parameters

```
current 0% - 100% of driver current
```

Returns

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

11.3.4.27 setHomingOffset()

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.

11.3.4.28 setMaxAcceleration()

Set the maximum permitted joint acceleration (and deceleration) in rad/s 2 or m/s 2 for cylindrical and prismatic joints respectively.

Parameters

```
maxAccel maximum joint acceleration.
```

Returns

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

11.3.4.29 setMaxVelocity()

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

Parameters

Returns

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

11.3.4.30 setPosition()

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

Parameters

```
pos 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.

11.3.4.31 setVelocity()

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

Parameters



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.

11.3.4.32 startHoming()

non-blocking implementation to home the joint.

See Joint::_home() for documentation. The current_b_cmd flag is set to HOME This method returns immediatly 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()).

11.3.4.33 stop()

```
int Joint::stop (
     void )
```

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.

11.3.4.34 wait_while_busy()

Blocking loop waiting for BUSY flag to reset.

Parameters

```
period_ms time in ms between polls.
```

11.3.4.35 write()

Wrapper function to send command to the I2C slave.

Allocates a buffer of size sizeof(T) + RFLAGS_SIZE. Copyies *data* to the buffer and invokes writeTol2CDev(). The flags received from the transaction are copied to *flags*. The flags are described in Joint::read().

Template Parameters

Т	Datatype of value to be transmitted
---	-------------------------------------

Parameters

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

Returns

0 on OK, negative on error

11.3.5 Member Data Documentation

11.3.5.1 address

int Joint::address [private]

I2C adress.

11.3.5.2 current_b_cmd

```
stp_reg_t Joint::current_b_cmd = NONE [private]
```

Keeps track if a blocking command is being executed.

11.3.5.3 flags

```
u_int8_t Joint::flags = 0x00 [private]
```

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()

11.3.5.4 handle

```
int Joint::handle = -1 [private]
```

I2C bus handle.

11.3.5.5 max

```
float Joint::max = 0 [private]
```

Joint upper limit.

11.3.5.6 min

```
float Joint::min = 0 [private]
```

Joint lower limit.

11.3.5.7 name

```
std::string Joint::name
```

11.3.5.8 offset

```
float Joint::offset = 0 [private]
```

Joint position offset.

11.3.5.9 reduction

```
float Joint::reduction = 1 [private]
```

Joint to actuator reduction ratio.

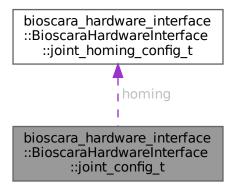
The documentation for this class was generated from the following files:

- 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

11.4 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

11.4.1 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.

11.4.2 Member Data Documentation

11.4.2.1 drive_current

u_int8_t bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::drive_current

11.4.2.2 hold_current

u_int8_t bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::hold_current

11.4.2.3 homing

joint_homing_config_t bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t↔
::homing

11.4.2.4 i2c_address

int bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::i2c_address

11.4.2.5 max

float bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::max

11.4.2.6 max acceleration

 $\verb|float| bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::max_acceleration| and the property of the p$

11.4.2.7 max_velocity

 $\verb|float| bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::max_velocity| and the property of the prope$

11.4.2.8 min

float bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::min

11.4.2.9 reduction

float bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::reduction = 1

11.4.2.10 stall threshold

 $\verb|u_int8_t| bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::stall_ \leftrightarrow threshold \\$

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

11.5 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

11.5.1 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.

11.5.2 Member Data Documentation

11.5.2.1 acceleration

float bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing_config_t::acceleration
= 0.01

11.5.2.2 current

u_int8_t bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing_config_t::current
= 10

11.5.2.3 speed

float bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing_config_t::speed = 0

11.5.2.4 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

11.6 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

11.6.1 Detailed Description

PWM class for the Raspberry PI 4 and 5.

11.6.2 Constructor & Destructor Documentation

```
11.6.2.1 ∼RPI_PWM()
```

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

11.6.3 Member Function Documentation

11.6.3.1 disable()

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

11.6.3.2 enable()

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

11.6.3.3 setDutyCycle()

Sets the duty cycle.

Parameters

V	The duty cycle in percent.		
return	>0 on success and -1 after an error.		

11.6.3.4 setDutyCycleNS()

11.6.3.5 setPeriod()

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

11.6.3.6 start()

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

Starts the PWM

Parameters

channel	The GPIO channel which is 2 or 3 for the RPI5
frequency	The PWM frequency
duty_cycle	The initial duty cycle of the PWM (default 0)
chip	The chip number (for RPI5 it's 2)
return	>0 on success and -1 if an error has happened.

11.6.3.7 stop()

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

Stops the PWM

11.6.3.8 writeSYS()

11.6.4 Member Data Documentation

11.6.4.1 chippath

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

11.6.4.2 per

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

11.6.4.3 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

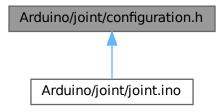
Chapter 12

File Documentation

12.1 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^{\wedge} 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.

12.1.1 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.

12.1.2 Macro Definition Documentation

12.1.2.1 ADR

#define ADR 0x11

I2C adress of joint n is 0x1n.

12.1.2.2 MAXACCEL

#define MAXACCEL 10000

Maximum acceleration in steps/s $^{\wedge}$ 2. Can be set for each joint depending on inertia. If set to high stalls might trigger since PID error grows too large.

12.1.2.3 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.

12.2 configuration.h 65

12.2 configuration.h

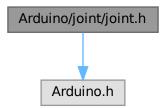
Go to the documentation of this file.

```
00001
00014 #ifndef CONFIGURATION_H
00015 #define CONFIGURATION_H
00016
00017 #if defined(J1)
00019 #define ADR 0x11
00020 #define MAXACCEL 10000
00021 #define MAXVEL 800
00022
00023 #elif defined(J2)
00024 #define ADR 0x12
00025 #define MAXACCEL 10000
00026 #define MAXVEL 800
00027
00028 #elif defined(J3)
00029 #define ADR 0x13
00030 #define MAXACCEL 10000
00031 #define MAXVEL 800
00032
00033 #elif defined(J4)
00033 #edfine ADR 0x14
00035 #define MAXACCEL 10000
00036 #define MAXVEL 800
00037 #else
00038
00039 /\star Below only defined for documentation \star/
00043 #define ADR 0x11
00044
00049 #define MAXACCEL 10000
00055 #define MAXVEL 800 00056 #error "No Joint has been defined. Define one of 'JX' where X 1,2,3,4" 00057 #endif
00058
00059 #endif
```

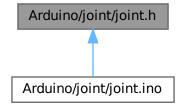
12.3 Arduino/joint/joint.h File Reference

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 , ENABLECLOSEDLOOP = 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.

12.3.1 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.

12.3.2 Macro Definition Documentation

12.3.2.1 ACK

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

12.3.2.2 DUMP_BUFFER

Value:

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

Macro to dump a buffer to the serial console.

Parameters

ı	1	
	buπer	pointer to a buffer to dump to the console
	size	number of bytes to dump

12.3.2.3 MAX_BUFFER

```
#define MAX_BUFFER 4
```

Maximum size of I2C Payload in bytes.

4 bytes used to transmit floats and int32_t

12.3.2.4 NACK

#define NACK 'N'

12.3.2.5 RFLAGS_SIZE

```
#define RFLAGS_SIZE 1
```

Size of the return flags in bytes.

Only one byte used and hence set to 1.

12.3.3 Enumeration Type Documentation

12.3.3.1 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].
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].

Enumerator

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) -].

12.3.4 Function Documentation

12.3.4.1 readValue()

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

Parameters

val	Reference to output variable
rxBuf	Buffer to read value from
rx_length	Length of the buffer

12.3.4.2 writeValue()

Writes a value of the specified type to a buffer.

Parameters

val	Reference to input variable
-----	-----------------------------

Parameters

txBuf	pointer to tx buffer
tx_length	Length of the buffer returne

Returns

0 On success

12.4 joint.h

Go to the documentation of this file.

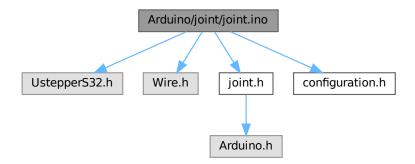
```
00014 #ifndef JOINT_H
00015 #define JOINT_H
00016 #include <Arduino.h>
00017
00018 #define ACK 'O'
00019 #define NACK 'N'
00020
00026 #define MAX_BUFFER 4 // Bytes
00027
00033 #define RFLAGS_SIZE 1
00034
00041 #define DUMP_BUFFER(buffer, size)
00042
00043
           Serial.print("Buffer dump: ");
00044
           for (size_t i = 0; i < size; i++)
00045
             Serial.print(buffer[i], HEX);
00046
             Serial.print(" ");
00047
00048
00049
           Serial.println();
00050
00051
00060 enum stp_reg_t
00061 {
00062
         PING = 0x0f,
         SETUP = 0x10,
SETRPM = 0x11,
00063
00064
         GETDRIVERRPM = 0x12,
MOVESTEPS = 0x13,
00065
00066
00067
         MOVEANGLE = 0x14,
         MOVETOANGLE = 0x15,
00068
00069
         GETMOTORSTATE = 0x16,
         RUNCOTINOUS = 0x17,
ANGLEMOVED = 0x18,
00070
00071
         SETCURRENT = 0x19,
SETHOLDCURRENT = 0x1A,
00072
00073
00074
         SETMAXACCELERATION = 0x1B,
00075
         SETMAXDECELERATION = 0x1C,
         SETMAXVELOCITY = 0x1D,
ENABLESTALLGUARD = 0x1E,
DISABLESTALLGUARD = 0x1F,
00076
00077
00078
00079
         CLEARSTALL = 0x20,
         SETBRAKEMODE = 0x22,
08000
         ENABLEPID = 0x23,
DISABLEPID = 0x24,
00081
00082
         ENABLECLOSEDLOOP = 0x25,
00083
         DISABLECLOSEDLOOP = 0x26,
SETCONTROLTHRESHOLD = 0x27,
00084
00085
00086
         MOVETOEND = 0x28,
         STOP = 0x29,
GETPIDERROR = 0x2A,
00087
00088
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 {
    tx_length = sizeof(T);
00118    memcpy(txBuf, &val, tx_length);
00119    return 0;
00120 }
00121
00122    #endif
```

12.5 Arduino/joint/joint.ino File Reference

joint firmware

```
#include <UstepperS32.h>
#include <Wire.h>
#include "joint.h"
#include "configuration.h"
Include dependency graph for joint.ino:
```



Macros

• #define J4

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 tx_data_ready = 0
bool rx_data_ready = 0
size_t tx_length = 0
size_t rx_length = 0
```

12.5.1 Detailed Description

```
joint firmware
```

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

This file contains the joint firmware.

12.5.2 Macro Definition Documentation

12.5.2.1 J4

```
#define J4
```

Define either joint that is to be flashed.

Define either J1, J2, J3 or J4 and subsequently include configuration.h

12.5.3 Function Documentation

12.5.3.1 blocking_handler()

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

reg command that should be executed.

Homing has been cancled from ISR (f.x. STOP)

12.5.3.2 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

12.5.3.3 non_blocking_handler()

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

reg command to execute/register to read.

12.5.3.4 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 immediatly 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

n the number of bytes read from the controller device: MAX_BUFFER

12.5.3.5 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.

12.5.3.6 setup()

```
void setup (
     void )
```

Setup Peripherals.

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

12.5.4 Variable Documentation

12.5.4.1 reg

```
uint8\_t reg = 0
```

12.5.4.2 rx_buf

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

12.5.4.3 rx_data_ready

```
bool rx_data_ready = 0
```

12.5.4.4 rx_length

```
size_t rx_length = 0
```

12.5.4.5 stepper

```
UstepperS32 stepper
```

12.5.4.6 tx_buf

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

12.5.4.7 tx_data_ready

```
bool tx_data_ready = 0
```

12.5.4.8 tx_length

```
size_t tx_length = 0
```

12.6 docs/DOCS README.md File Reference

12.7 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_← bringup/launch/bioscara.launch.py File Reference

Namespaces

• namespace bioscara

Functions

- bioscara.generate_launch_description ()
- 12.8 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 ()

- 12.9 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup/

 README.md File Reference
- 12.10 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/

 README.md File Reference
- 12.11 ROS2/ros2_scara_ws/src/dalsa_bioscara/README.md File Reference
- 12.12 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_
 description/bioscara_description/__init__.py File Reference
- 12.13 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_ description/launch/display.launch.py File Reference

Namespaces

· namespace display

Functions

- display.generate launch description ()
- 12.14 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_← description/launch/gazebo.launch.py File Reference

Namespaces

· namespace gazebo

Functions

- gazebo.generate_launch_description ()
- 12.15 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

12.16 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_ description/test/test_copyright.py File Reference

Namespaces

· namespace test_copyright

Functions

• test_copyright.test_copyright ()

12.17 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_← description/test/test_flake8.py File Reference

Namespaces

• namespace test_flake8

Functions

• test_flake8.test_flake8 ()

12.18 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_ description/test/test_pep257.py File Reference

Namespaces

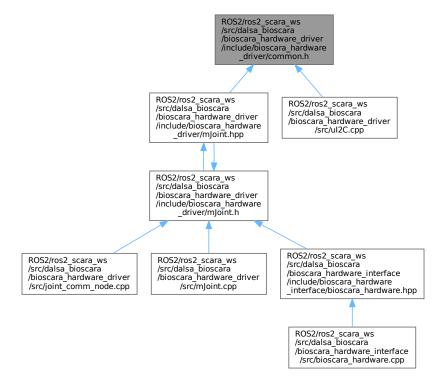
namespace test_pep257

test_pep257.test_pep257 ()

12.19 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.

12.19.1 Detailed Description

A file containing utility macros and functions.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

12.19.2 Macro Definition Documentation

12.19.2.1 **DUMP_BUFFER**

Macro to dump a buffer to cout.

Parameters

buffer	pointer to a buffer to dump to the console
size	number of bytes to dump

12.20 common.h

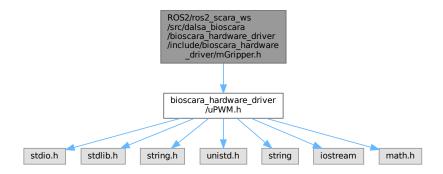
Go to the documentation of this file.

```
00001
00011 #ifndef COMMON_H
00012 #define COMMON_H
00014
00021 #define DUMP_BUFFER(buffer, size)
00022 {
         std::cout « "Buffer dump: ";
for (size_t i = 0; i < size; i++)
00023
00024
00025
00026
             printf("%#x ", buffer[i]);
00027
00028
          std::cout « std::endl;
00023
00030
00031 #endif // COMMON_H
```

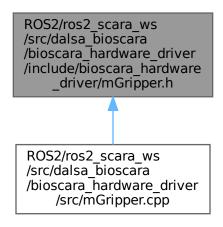
ROS2/ros2 scara ws/src/dalsa bioscara/bioscara hardware \leftarrow 12.21 driver/include/bioscara hardware driver/mGripper.h File Reference

File containing the Gripper class.

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



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



Classes

· class Gripper

Gripper object to interact with the robot gripper.

12.21.1 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.

12.22 mGripper.h

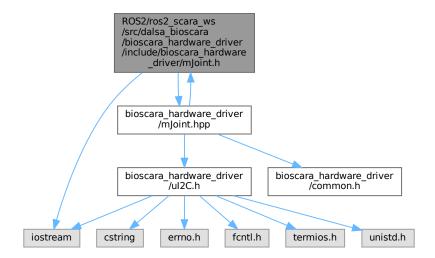
Go to the documentation of this file.

```
00013 #ifndef MGRIPPER_H
00014 #define MGRIPPER_H
00015 #include "bioscara_hardware_driver/uPWM.h"
00016
00054 class Gripper
00055 {
00056 public:
00057
          Gripper(void);
00058
00064
          int init(void);
00065
00071
          int deinit (void);
00072
00081
           int enable(void);
00082
00090
00091
           int disable(void);
00098
           int setPosition(float width);
00099
00100 private:
00101 RPI_PWM pwm;
00102 };
00103 #endif // MGRIPPER_H
```

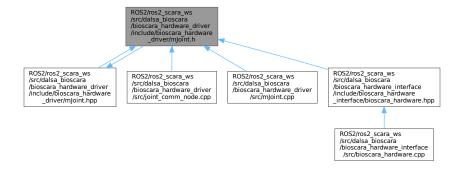
ROS2/ros2 scara ws/src/dalsa bioscara/bioscara hardware \leftarrow 12.23 driver/include/bioscara hardware driver/mJoint.h File Reference

File including the Joint class.

#include <iostream> #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 Joint

Representing a single joint on the I2C bus.

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.

12.23.1 Detailed Description

File including the Joint class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-29

Copyright

Copyright (c) 2025

12.23.2 Macro Definition Documentation

12.23.2.1 ACTUATOR2JOINT

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

12.24 mJoint.h 85

12.23.2.2 DEG2RAD

```
#define DEG2RAD( deg \ ) \ (deg \ * \ \underline{M_PI} \ / \ 180)
```

Macro to convert degree to radians.

12.23.2.3 JOINT2ACTUATOR

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

12.23.2.4 M PI

```
#define M_PI 3.14159265358979323846
```

12.23.2.5 RAD2DEG

```
#define RAD2DEG( rad ) (rad / M_PI * 180)
```

Macro to convert radians to degree.

12.24 mJoint.h

Go to the documentation of this file.

```
00001
00012 #ifndef MJOINT_H
00013 #define MJOINT_H
00014
00015 #include <iostream>
00016
00025 \#define JOINT2ACTUATOR(in, reduction, offset) (reduction * (in - offset))
00026
00035 #define ACTUATOR2JOINT(in, reduction, offset) (in / reduction + offset)
00036
00041 #define M_PI 3.14159265358979323846
00042
00047 \#define RAD2DEG(rad) (rad / M_PI * 180)
00048
00053 #define DEG2RAD(deg) (deg * M_PI / 180)
00054
00059 class Joint
00060 {
```

```
00061 public:
00071
        enum stp_reg_t
00072
          NONE = 0x00,
00073
          PING = 0x0f,
00074
          SETUP = 0x10,
00075
00076
          SETRPM = 0x11,
00077
          GETDRIVERRPM = 0x12,
          MOVESTEPS = 0x13,
MOVEANGLE = 0x14,
00078
00079
          MOVETOANGLE = 0x15,
08000
          GETMOTORSTATE = 0x16,
00081
          RUNCOTINOUS = 0x17,
00082
00083
          ANGLEMOVED = 0x18,
00084
          SETCURRENT = 0x19,
          SETHOLDCURRENT = 0x1A,
SETMAXACCELERATION = 0x1B,
SETMAXDECELERATION = 0x1C,
00085
00086
00087
          SETMAXVELOCITY = 0x1D,
00088
00089
          ENABLESTALLGUARD = 0x1E,
00090
          DISABLESTALLGUARD = 0x1F,
00091
          CLEARSTALL = 0x20,
          SETBRAKEMODE = 0 \times 22.
00092
00093
          ENABLEPID = 0x23,
00094
          DISABLEPID = 0x24,
00095
          ENABLECLOSEDLOOP = 0x25,
00096
          DISABLECLOSEDLOOP = 0x26,
00097
          SETCONTROLTHRESHOLD = 0x27.
00098
          MOVETOEND = 0x28,
00099
          STOP = 0x29.
          GETPIDERROR = 0x2A,
CHECKORIENTATION = 0x2B,
00100
00101
00102
          GETENCODERRPM = 0x2C,
00103
          HOME = 0x2D,
          HOMEOFFSET = 0x2E,
00104
00105
00106
00139
        Joint (const std::string name, const int address, const float reduction, const float min, const float
      max);
00140
        ~Joint (void);
00141
00151
        int init (void);
00152
00162
        int deinit (void);
00163
00178
        int enable(u_int8_t driveCurrent, u_int8_t holdCurrent);
00179
00186
        int disable (void);
00187
00197
        int home(float velocity, u_int8_t sensitivity, u_int8_t current);
00198
00209
        int startHoming(float velocity, u_int8_t sensitivity, u_int8_t current);
00210
00224
        int postHoming(void);
00225
00226
        int printInfo(void);
00227
00240
        int getPosition(float &pos);
00241
00254
        int setPosition(float pos);
00255
00268
        int moveSteps(int32_t steps);
00269
00279
        int getVelocity(float &vel);
00280
00293
        int setVelocity(float vel);
00294
00307
        int checkOrientation(float angle = 10.0);
00308
00319
        int stop(void);
00325
        int disableCL(void);
00326
00334
        int setDriveCurrent(u_int8_t current);
00335
00344
        int setHoldCurrent(u int8 t current);
00345
00352
        int setBrakeMode(u_int8_t mode);
00353
00362
        int setMaxAcceleration(float maxAccel);
00363
00372
        int setMaxVelocity(float maxVel);
00373
00383
        int enableStallguard(u_int8_t sensitivity);
00384
00393
        bool isHomed(void);
00394
00404
        bool isEnabled(void);
```

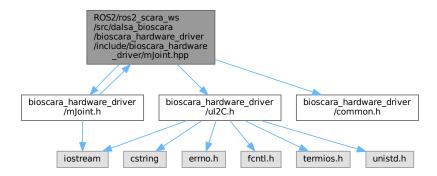
```
00405
00413
       bool isStalled(void);
00414
00422
       bool isBusy (void);
00423
00432
       int checkCom(void);
00433
00439
       u_int8_t getFlags(void);
00440
00451
       int getHomingOffset(float &offset);
00452
       int setHomingOffset(const float offset);
00463
00464
00470
       stp_reg_t getCurrentBCmd(void);
00471
00472
       std::string name;
00473
00474 protected:
00475 private:
00476
       template <typename T>
00477
       int read(const stp_reg_t reg, T &data, u_int8_t &flags);
00478
00479
       template <typename T>
       int write(const stp_reg_t reg, T data, u_int8_t &flags);
00480
00481
       void wait_while_busy(const float period_ms);
00488
00506
       int _home(float velocity, u_int8_t sensitivity, u_int8_t current);
00507
00526
       u int8 t flags = 0x00;
00527
00528
       int address;
00529
       float reduction = 1;
00530
       float offset = 0;
00531
       float min = 0;
float max = 0;
00532
00533
       stp_reg_t current_b_cmd = NONE;
00535
00536
       int handle = -1;
00537 };
00538
00539 #include "bioscara_hardware_driver/mJoint.hpp"
00540
00541 #endif
```

ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_ driver/include/bioscara hardware driver/mJoint.hpp File Reference

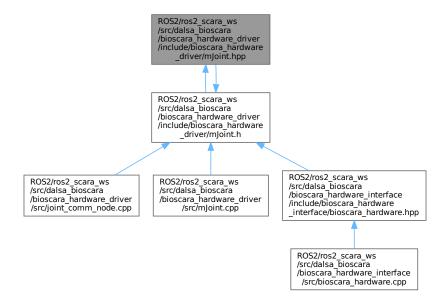
Templated functions for the Joint class.

```
#include "bioscara_hardware_driver/mJoint.h"
#include "bioscara_hardware_driver/uI2C.h"
#include "bioscara_hardware_driver/common.h"
```

Include dependency graph for mJoint.hpp:



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



12.25.1 Detailed Description

Templated functions for the Joint class.

Author

Sebastian Storz

Version

0.1

12.26 mJoint.hpp 89

Date

2025-05-29

Copyright

Copyright (c) 2025

This header must be included at the END of the mJoint.h file.

12.26 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
00031 template <typename T>
00032 int Joint::read(const stp_reg_t reg, T &data, u_int8_t &flags)
00033 {
          size_t size = sizeof(T) + RFLAGS_SIZE;
          char buf[MAX_BUFFER+RFLAGS_SIZE];
00036
          int n = readFromI2CDev(this->handle, reg, buf, size);
00037
          if (n != static_cast<int>(size))
00038
00039
              return -1;
00040
00041
          memcpy(&data, buf, size - RFLAGS_SIZE);
00042
          memcpy(&flags, buf + size - RFLAGS_SIZE, RFLAGS_SIZE);
00043
           return 0;
00044 }
00045
00061 template <typename T>
00062 int Joint::write(const stp_reg_t reg, T data, u_int8_t &flags)
00064
          size_t size = sizeof(T) + RFLAGS_SIZE;
          char buf[MAX_BUFFER+RFLAGS_SIZE];
memcpy(buf, &data, size - RFLAGS_SIZE);
int rc = writeToI2CDev(this->handle, reg, buf, size - RFLAGS_SIZE, buf + size - RFLAGS_SIZE);
00065
00066
00067
00068
          rc = rc > 0 ? 0 : rc;
00069
          memcpy(&flags, buf + size - RFLAGS_SIZE, RFLAGS_SIZE);
00070
00071 }
```

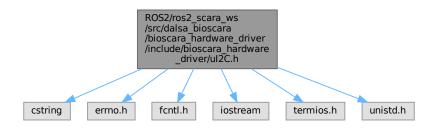
12.27 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_ driver/include/bioscara_hardware_driver/ul2C.h File Reference

Low level utility for I2C communication on Raspberry Pi using Igpio library.

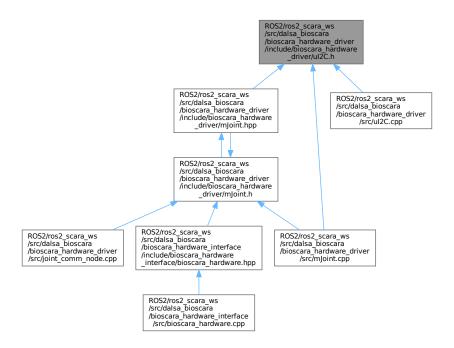
```
#include <cstring>
#include <errno.h>
#include <fcntl.h>
#include <iostream>
#include <termios.h>
```

#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 writeTol2CDev (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

12.27.1 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

12.27.2 Macro Definition Documentation

12.27.2.1 ACK

#define ACK 'O'

12.27.2.2 MAX_BUFFER

```
#define MAX_BUFFER 4
```

Maximum size of I2C Payload in bytes.

4 bytes used to transmit floats and int32_t

12.27.2.3 NACK

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

12.27.2.4 RFLAGS_SIZE

```
#define RFLAGS_SIZE 1
```

Size of the return flags in bytes.

Only one byte used and hence set to 1.

12.27.3 Function Documentation

12.27.3.1 closel2CDevHandle()

close an I2C device on the bus

Parameters

dev_handle | device handle obtained from openI2CDevHandle

Returns

0 on OK, negative on error.

12.27.3.2 openI2CDevHandle()

```
int openI2CDevHandle ( {\tt const\ int\ } dev\_addr\ )
```

Initiates an I2C device on the bus.

Parameters

dev addr	7-bit device adress [0 - 0x7F]
----------	--------------------------------

Returns

the device handle, negative on error.

12.27.3.3 readFromI2CDev()

reads block of bytes from device to buffer

Parameters

dev_handle	device handle obtained from openI2CDevHandle
reg	the command/data register
buffer	pointer to data buffer to hold received values
data_length	number of bytes to read

Returns

number of bytes read, negative on error.

12.27.3.4 writeToI2CDev()

writes block of bytes from buffer to device

Parameters

dev_handle	device handle obtained from openI2CDevHandle
reg	the command/data register
tx_buffer	pointer to data buffer holding the data to send
data_length	number of bytes to send
RFLAGS_buffer	buffer to hold returned flags

Returns

0 on OK, negative on error.

12.28 ul2C.h

Go to the documentation of this file.

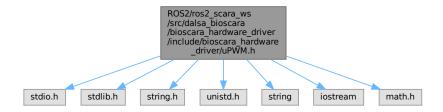
```
00001
00028 #ifndef USERIAL H
00029 #define USERIAL_H
00030 #include <cstring>
00031 #include <errno.h>
00032 #include <fcntl.h>
00033 #include <iostream>
00034 #include <termios.h>
00035 #include <unistd.h>
00036
00037 #define ACK 'O'
00038 #define NACK 'N'
00039
00043 #define RFLAGS_SIZE 1
00044
00048 #define MAX_BUFFER 4 // Bytes
00055 int openI2CDevHandle(const int dev_addr);
00056
00065 int readFromI2CDev(const int dev_handle, const int req, 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
```

12.29 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_ driver/include/bioscara_hardware_driver/uPWM.h File Reference

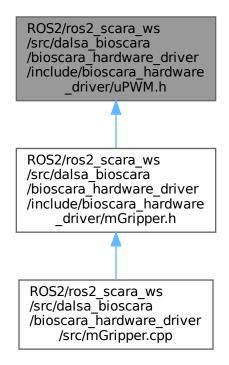
Includes source code for Hardware PWM generation on Raspberry Pi 4.

```
#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.

12.29.1 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

12.30 uPWM.h

Go to the documentation of this file.

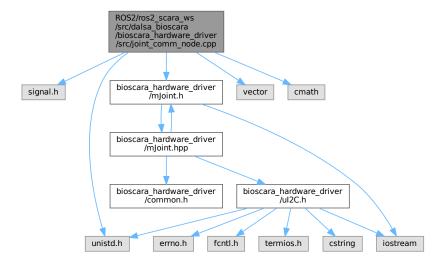
```
00019 #ifndef ___RPIPWM
00020 #define ___RPIPWM
00021
00022 #include <stdio.h>
00023 #include <stdlib.h>
00024 #include <string.h>
00025 #include <unistd.h>
00026 #include <string>
00027 #include <iostream>
00028 #include <math.h>
00029
00033 class RPI_PWM
00034 {
00035 public:
00044
         int start (int channel, int frequency, float duty_cycle = 0, int chip = 2)
00045
              chippath = "/sys/class/pwm/pwmchip" + std::to_string(chip);
00046
              pwmpath = chippath + "/pwm" + std::to_string(channel);
std::string p = chippath + "/export";
00047
              FILE *const fp = fopen(p.c_str(), "w");
00049
00050
              if (NULL == fp)
00051
              {
                  std::cerr « "PWM device does not exist. Make sure to add 'dtoverlay=pwm-2chan' to
00052
     /boot/firmware/config.txt.\n";
                  return -1;
00054
00055
              const int r = fprintf(fp, "%d", channel);
00056
              fclose(fp);
00057
              if (r < 0)
    return r;</pre>
00058
00059
              usleep(100000); // it takes a while till the PWM subdir is created
00060
              per = (int)1E9 / frequency;
00061
              setPeriod(per);
00062
              setDutyCycle(duty_cycle);
00063
              enable();
00064
              return r:
00065
          }
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
             const int r = writeSYS(pwmpath + "/" + "duty_cycle", ns);
00101
00102
         }
00103
00104
         void enable() const
00105
             writeSYS(pwmpath + "/" + "enable", 1);
00106
00107
00108
00109
         void disable() const
00110
             writeSYS(pwmpath + "/" + "enable", 0);
00111
00112
00113
00114
         int per = 0;
00115
         std::string chippath;
00116
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
```

12.31 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.cpp:



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)

12.31.1 Function Documentation

12.31.1.1 INT_handler()

```
void INT_handler ( \quad \text{int } s \ )
```

12.31.1.2 main()

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

12.31.2 Variable Documentation

12.31.2.1 J1

```
Joint J1("j1", 0x11, 35, -3.04647, 3.04647) (
    "j1" ,
    0x11 ,
    35 ,
    -3. 04647,
    3. 04647 )
```

12.31.2.2 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)
```

12.31.2.3 J3

```
Joint J3("j3", 0x13, 24, -2.62672, 2.62672) (
    "j3",
    0x13,
    24,
    -2. 62672,
    2. 62672)
```

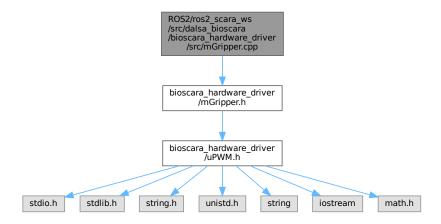
12.31.2.4 J4

```
Joint J4("j4", 0x14, 12, -3.01069, 3.01069) (
    "j4",
    0x14,
    12,
    -3. 01069,
    3. 01069)
```

12.32 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_ driver/src/mGripper.cpp File Reference

```
#include "bioscara_hardware_driver/mGripper.h"
```

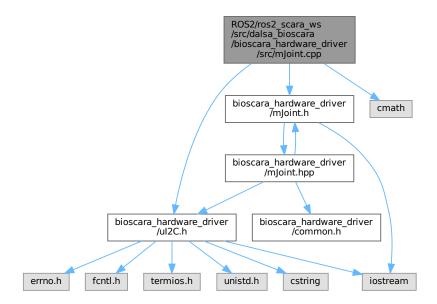
Include dependency graph for mGripper.cpp:



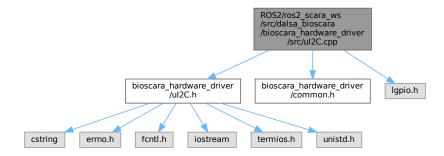
12.33 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_ driver/src/mJoint.cpp File Reference

```
#include "bioscara_hardware_driver/uI2C.h"
#include "bioscara_hardware_driver/mJoint.h"
#include <cmath>
```

Include dependency graph for mJoint.cpp:



```
#include "bioscara_hardware_driver/uI2C.h"
#include "bioscara_hardware_driver/common.h"
#include <lgpio.h>
Include dependency graph for uI2C.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 writeTol2CDev (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 closel2CDevHandle (int &dev_handle)

close an I2C device on the bus

12.34.1 Function Documentation

12.34.1.1 closel2CDevHandle()

close an I2C device on the bus

Parameters

dev_handle	device handle obtained from openI2CDevHandle

Returns

0 on OK, negative on error.

12.34.1.2 openI2CDevHandle()

```
int openI2CDevHandle ( {\tt const\ int\ } dev\_addr\ )
```

Initiates an I2C device on the bus.

Parameters

dev_addr	7-bit device adress [0 - 0x7F]
----------	--------------------------------

Returns

the device handle, negative on error.

12.34.1.3 readFromI2CDev()

reads block of bytes from device to buffer

Parameters

dev_handle	device handle obtained from openI2CDevHandle	
reg	the command/data register	
buffer	pointer to data buffer to hold received values	
data_length	number of bytes to read	

Returns

number of bytes read, negative on error.

12.34.1.4 writeToI2CDev()

writes block of bytes from buffer to device

Parameters

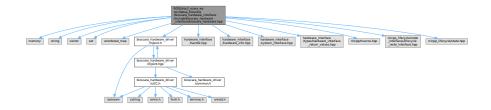
dev_handle	device handle obtained from openI2CDevHandle	
reg	the command/data register	
tx_buffer	pointer to data buffer holding the data to send	
data_length	number of bytes to send	
RFLAGS_buffer	buffer to hold returned flags	

Returns

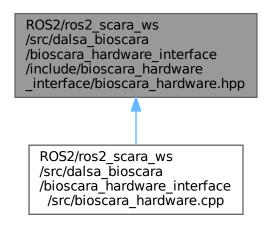
0 on OK, negative on error.

12.35 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_ interface/include/bioscara_hardware_interface/bioscara_ hardware.hpp File Reference

```
#include <memory>
#include <vector>
#include <vector>
#include <set>
#include <unordered_map>
#include "bioscara_hardware_driver/mJoint.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_hardware_interface::BioscaraHardwareInterface
 The bioscara hardware interface class.
- struct bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing_config_t
 configuration structure holding the passed homing paramters from the ros2_control urdf
- struct bioscara_hardware_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"

12.36 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 <memorv>
00019 #include <string>
00020 #include <vector>
00021 #include <set>
00022 #include <unordered_map>
00023
00024 #include "bioscara hardware driver/mJoint.h"
00025
00026 #include "hardware_interface/handle.hpp"
00027 #include "hardware_interface/hardware_info.hpp"
00028 #include "hardware_interface/system_interface.hpp"
00029 #include "hardware_interface/types/hardware_interface_return_values.hpp"
00030 #include "rclcpp/macros.hpp"
00031 #include "rclcpp_lifecycle/node_interfaces/lifecycle_node_interface.hpp"
00032 #include "rclcpp_lifecycle/state.hpp"
00033
00034 namespace bioscara_hardware_interface
00035 {
00036
          constexpr char HW IF HOME[] = "home";
00037
00051
          class BioscaraHardwareInterface : public hardware_interface::SystemInterface
00052
         public:
00053
00054
             RCLCPP_SHARED_PTR_DEFINITIONS (BioscaraHardwareInterface)
00055
00056
              hardware interface::CallbackReturn on init(
00057
                  const hardware_interface::HardwareComponentInterfaceParams &params) override;
00058
00068
              hardware_interface::CallbackReturn on_shutdown(
00069
                  const rclcpp_lifecycle::State &previous_state) override;
00070
00078
              hardware_interface::CallbackReturn on_configure(
00079
                  const rclcpp_lifecycle::State &previous_state) override;
00080
00089
              hardware_interface::CallbackReturn on_cleanup(
00090
                  const rclcpp_lifecycle::State &previous_state) override;
00091
00106
              hardware interface::CallbackReturn on activate(
00107
                  const rclcpp_lifecycle::State &previous_state) override;
00108
00117
              hardware_interface::CallbackReturn on_deactivate(
00118
                  const rclcpp_lifecycle::State &previous_state) override;
00119
00140
             hardware_interface::return_type read(
00141
                 const rclcpp::Time &time,
00142
                  const rclcpp::Duration &period) override;
00143
              hardware_interface::return_type write(
00165
00166
                  const rclcpp::Time &time,
                  const rclcpp::Duration &period) override;
00167
00168
00196
              hardware_interface::return_type prepare_command_mode_switch(
00197
                 const std::vector<std::string> &start_interfaces,
00198
                  const std::vector<std::string> &stop_interfaces) override;
00199
00227
             hardware interface::CallbackReturn on error(
00228
                 const rclcpp_lifecycle::State &previous_state) override;
00229
00230
         private:
00237
              struct joint_homing_config_t
00238
              {
00239
                  float speed = 0;
00240
                  u_int8_t threshold = 10;
                  u_int8_t current = 10;
00241
00242
                  float acceleration = 0.01;
00243
00244
00251
              struct joint_config_t
00252
00253
                  int i2c_address;
00254
                  float reduction = 1;
00255
                  float min;
00256
                  float max;
                  u_int8_t drive_current;
00257
00258
                  u int8 t hold current;
00259
                  u int8 t stall threshold;
00260
                  float max_velocity;
00261
                  float max_acceleration;
00262
                  joint_homing_config_t homing;
00263
              };
00264
00272
              std::unordered map<std::string, Joint> joints;
```

12.37 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 <iomanip>
#include <limits>
#include <memory>
#include <sstream>
#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:
```

The state of the s

Namespaces

namespace bioscara_hardware_interface

Index

```
home
                                                             on_shutdown, 30
                                                             prepare_command_mode_switch, 31
     Joint, 42
_joint_cfg
                                                             read, 31
    bioscara hardware interface::BioscaraHardwareInterface, write, 32
                                                        bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t,
_joint_command_modes
                                                                  56
     bioscara_hardware_interface::BioscaraHardwareInterface, drive_current, 56
          33
                                                             hold current, 56
                                                             homing, 57
_joints
     bioscara hardware interface::BioscaraHardwareInterface,i2c address, 57
                                                             max, 57
\simJoint
                                                             max acceleration, 57
     Joint, 42
                                                             max_velocity, 57
\simRPI PWM
                                                             min, 57
     RPI PWM, 59
                                                             reduction, 57
                                                             stall threshold, 57
acceleration
                                                        bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing_c
     bioscara_hardware_interface::BioscaraHardwareInterface::joint_fgoming_config_t,
                                                             acceleration, 58
ACK
                                                             current, 58
     joint.h, 67
                                                             speed, 58
     ul2C.h, 91
                                                             threshold, 58
ACTUATOR2JOINT
                                                        blocking handler
     mJoint.h, 84
                                                             joint.ino, 72
address
                                                        checkCom
     Joint, 54
ADR
                                                             Joint, 43
     configuration.h, 64
                                                        CHECKORIENTATION
ANGLEMOVED
                                                             Joint, 41
     Joint, 41
                                                             joint.h, 69
    joint.h, 68
                                                        checkOrientation
Arduino/joint/configuration.h, 63, 65
                                                             Joint, 43
Arduino/joint/joint.h, 65, 70
                                                        chippath
Arduino/joint/joint.ino, 71
                                                             RPI PWM, 61
                                                        CLEARSTALL
bioscara, 19
                                                             Joint, 41
     generate_launch_description, 19
                                                             joint.h, 69
bioscara hardware interface, 19
                                                        closeI2CDevHandle
     HW IF HOME, 19
                                                             ul2C.cpp, 101
bioscara_hardware_interface::BioscaraHardwareInterface,
                                                             ul2C.h, 92
          25
                                                        common.h
     _joint_cfg, 33
                                                             DUMP_BUFFER, 80
     _joint_command_modes, 33
                                                        configuration.h
     joints, 33
                                                             ADR, 64
     on_activate, 27
                                                             MAXACCEL, 64
     on cleanup, 28
                                                             MAXVEL, 64
     on configure, 28
                                                        current
     on deactivate, 29
                                                             bioscara hardware interface::BioscaraHardwareInterface::joint hom
     on_error, 29
     on_init, 30
                                                        current_b_cmd
```

1	
Joint, 54	generate_launch_description, 20 generate_launch_description
data_files	bioscara, 19
setup, 21	display, ²⁰
DEG2RAD	gazebo, 20
mJoint.h, 84	test_joint_trajectory_controller, 23
deinit	getCurrentBCmd
Gripper, 35	Joint, 45
Joint, 43	GETDRIVERRPM
description	Joint, 41
setup, 21	joint.h, 68
disable	GETENCODERRPM
Gripper, 35	Joint, 41
Joint, 43	joint.h, 69
RPI_PWM, 59	getFlags
disableCL	Joint, 45
Joint, 44	getHomingOffset
DISABLECLOSEDLOOP	Joint, 45
Joint, 41	GETMOTORSTATE
joint.h, 69	Joint, 41
DISABLEPID	joint.h, 68
Joint, 41	GETPIDERROR
joint.h, 69	Joint, 41
DISABLESTALLGUARD	joint.h, 69
Joint, 41	getPosition
joint.h, 69	Joint, 45
display, 20	getVelocity
generate_launch_description, 20	Joint, 46
docs/DOCS_README.md, 76	Gripper, 34
Documentation, 1	deinit, 35
drive_current	disable, 35
bioscara_hardware_interface::BioscaraHardw	
56	Gripper, 35
DUMP_BUFFER	init, 36
common.h, 80	pwm, 37
joint.h, 67	setPosition, 36
enable	handle
Gripper, 36	
Joint, 44	Joint, 55
RPI_PWM, 59	hold_current
ENABLECLOSEDLOOP	bioscara_hardware_interface::BioscaraHardwareInterface::joint_confi
Joint, 41	56
joint.h, 69	HOME
ENABLEPID	Joint, 41
Joint, 41	joint.h, 69
joint.h, 69	home
ENABLESTALLGUARD	Joint, 46
Joint, 41	HOMEOFFSET
joint.h, 68	Joint, 41
enableStallguard	joint.h, 69
Joint, 44	homing
entry_points	bioscara_hardware_interface::BioscaraHardwareInterface::joint_confi
setup, 21	57
octup, Z1	HW_IF_HOME bioscara_hardware_interface, 19
flags	bioscara_nardware_interrace, 13
Joint, 54	i2c_address
gozobo 20	bioscara_hardware_interface::BioscaraHardwareInterface::joint_confi
gazebo, 20	57

init	HOMEOFFSET, 41
Gripper, 36	init, 46
Joint, 46	isBusy, 46
install_requires	isEnabled, 47
setup, 21	isHomed, 47
INT handler	isStalled, 47
joint_comm_node.cpp, 98	Joint, 41
isBusy	max, 55
Joint, 46	min, 55
isEnabled	MOVEANGLE, 41
	MOVESTEPS, 41
Joint, 47 isHomed	
	moveSteps, 47
Joint, 47	MOVETOANGLE, 41
isStalled	MOVETOEND, 41
Joint, 47	name, 55
J1	NONE, 41
	offset, 55
joint_comm_node.cpp, 99	PING, 41
J2	postHoming, 49
joint_comm_node.cpp, 99	printlnfo, 49
J3	read, 49
joint_comm_node.cpp, 99	reduction, 55
J4	RUNCOTINOUS, 41
joint.ino, 72	SETBRAKEMODE, 41
joint_comm_node.cpp, 99	setBrakeMode, 50
Joint, 37	SETCONTROLTHRESHOLD, 41
_home, 42	SETCURRENT, 41
∼Joint, 42	setDriveCurrent, 50
address, 54	SETHOLDCURRENT, 41
ANGLEMOVED, 41	setHoldCurrent, 50
checkCom, 43	
CHECKORIENTATION, 41	setHomingOffset, 51
checkOrientation, 43	SETMAXACCELERATION, 41
CLEARSTALL, 41	setMaxAcceleration, 51
current b cmd, 54	SETMAXDECELERATION, 41
deinit, 43	SETMAXVELOCITY, 41
•	setMaxVelocity, 51
disable, 43	setPosition, 52
disableCL, 44	SETRPM, 41
DISABLECLOSEDLOOP, 41	SETUP, 41
DISABLEPID, 41	setVelocity, 52
DISABLESTALLGUARD, 41	startHoming, 52
enable, 44	STOP, 41
ENABLECLOSEDLOOP, 41	stop, 53
ENABLEPID, 41	stp_reg_t, 40
ENABLESTALLGUARD, 41	wait_while_busy, 53
enableStallguard, 44	write, 53
flags, 54	joint.h
getCurrentBCmd, 45	ACK, 67
GETDRIVERRPM, 41	ANGLEMOVED, 68
GETENCODERRPM, 41	CHECKORIENTATION, 69
getFlags, 45	CLEARSTALL, 69
getHomingOffset, 45	DISABLECLOSEDLOOP, 69
GETMOTORSTATE, 41	
GETPIDERROR, 41	DISABLEPID, 69
getPosition, 45	DISABLESTALLGUARD, 69
getVelocity, 46	DUMP_BUFFER, 67
handle, 55	ENABLECLOSEDLOOP, 69
	ENABLEPID, 69
HOME, 41	ENABLESTALLGUARD, 68
home, 46	

0.5555000000000000000000000000000000000	
GETDRIVERRPM, 68	M_PI
GETENCODERRPM, 69	mJoint.h, 85
GETMOTORSTATE, 68	main
GETPIDERROR, 69	joint_comm_node.cpp, 98
HOME, 69	maintainer
HOMEOFFSET, 69	setup, 21
MAX_BUFFER, 68	maintainer_email
MOVEANGLE, 68	setup, 21
MOVESTEPS, 68	max
MOVETOANGLE, 68	bioscara_hardware_interface::BioscaraHardwareInterface::joint_conf
MOVETOEND, 69	57
	Joint, 55
NACK, 68	
PING, 68	max_acceleration
readValue, 69	bioscara_hardware_interface::BioscaraHardwareInterface::joint_conf
RFLAGS_SIZE, 68	57
RUNCOTINOUS, 68	MAX_BUFFER
SETBRAKEMODE, 69	joint.h, 68
SETCONTROLTHRESHOLD, 69	ul2C.h, 91
SETCURRENT, 68	max_velocity
SETHOLDCURRENT, 68	bioscara hardware interface::BioscaraHardwareInterface::joint conf
SETMAXACCELERATION, 68	57
SETMAXDECELERATION, 68	MAXACCEL
SETMAXVELOCITY, 68	configuration.h, 64
SETRPM, 68	MAXVEL
•	
SETUP, 68	configuration.h, 64
STOP, 69	min
stp_reg_t, 68	bioscara_hardware_interface::BioscaraHardwareInterface::joint_conf
writeValue, 69	57
joint.ino	Joint, 55
blocking_handler, 72	mJoint.h
J4, 72	ACTUATOR2JOINT, 84
loop, 73	DEG2RAD, 84
non blocking handler, 73	JOINT2ACTUATOR, 85
receiveEvent, 73	M PI, 85
reg, 75	RAD2DEG, 85
requestEvent, 75	MOVEANGLE
•	
rx_buf, 75	Joint, 41
rx_data_ready, 75	joint.h, 68
rx_length, 75	MOVESTEPS
setup, 75	Joint, 41
stepper, 75	joint.h, 68
tx_buf, 76	moveSteps
tx_data_ready, 76	Joint, 47
tx length, 76	MOVETOANGLE
JOINT2ACTUATOR	Joint, 41
mJoint.h, 85	joint.h, 68
joint_comm_node.cpp	MOVETOEND
INT_handler, 98	Joint, 41
J1, 99	joint.h, 69
J2, 99	NACK
J3, 99	
J4, 99	joint.h, 68
main, 98	ul2C.h, 92
	name
license	Joint, 55
setup, 21	setup, 21
loop	non_blocking_handler
joint.ino, 73	joint.ino, 73
•	NONE

```
Joint, 41
                                                                                                 README, 3, 5, 7
                                                                                                 readValue
offset
                                                                                                         joint.h, 69
        Joint, 55
                                                                                                 receiveEvent
on activate
                                                                                                         joint.ino, 73
        bioscara_hardware_interface::BioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInterface::bioscaraHardwareInt
                                                                                                         bioscara_hardware_interface::BioscaraHardwareInterface::joint_conf
on cleanup
        bioscara hardware_interface::BioscaraHardwareInterface, Joint, 55
                 28
                                                                                                 rea
on configure
                                                                                                         joint.ino, 75
        bioscara_hardware_interface::BioscaraHardwareInterfacetestEvent
                                                                                                         joint.ino, 75
on_deactivate
                                                                                                 RFLAGS SIZE
        bioscara_hardware_interface::BioscaraHardwareInterface, joint.h, 68
                                                                                                          ul2C.h, 92
                                                                                                 ROS2/ros2 scara ws/src/dalsa bioscara/bioscara bringup/launch/biosca
        bioscara_hardware_interface::BioscaraHardwareInterface,
                                                                                                                  76
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup/launch/test_jc
on init
        bioscara\_hardware\_interface:: BioscaraHardwareInterfaces: S2/ros2-scara\_ws/src/dalsa\_bioscara\_bringup/README.md
on_shutdown
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/bioscara_c
        bioscara hardware interface::BioscaraHardwareInterface,
                                                                                                 ROS2/ros2 scara ws/src/dalsa bioscara/bioscara description/launch/dis
openI2CDevHandle
                                                                                                                  77
         ul2C.cpp, 102
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/launch/gaz
        ul2C.h, 92
                                                                                                 ROS2/ros2 scara ws/src/dalsa bioscara/bioscara description/README.
package_name
        setup, 21
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/setup.py,
packages
        setup, 21
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/test/test_c
        RPI_PWM, 61
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/test/test_fl
PING
         Joint, 41
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/test/test_p
        joint.h, 68
postHoming
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/includes
        Joint, 49
prepare_command_mode_switch
                                                                                                  ROS2/ros2 scara ws/src/dalsa bioscara/bioscara hardware driver/include
        bioscara_hardware_interface::BioscaraHardwareInterface,
                                                                                                                  81.82
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/includes
printlnfo
        Joint, 49
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/includes
pwm
        Gripper, 37
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/includes
pwmpath
         RPI_PWM, 61
                                                                                                 ROS2/ros2 scara ws/src/dalsa bioscara/bioscara hardware driver/include
                                                                                                                  94.96
RAD2DEG
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/jc
        mJoint.h, 85
        bioscara_hardware_interface::BioscaraHardwareInterface;
                 31
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/m
        Joint, 49
readFromI2CDev
                                                                                                 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/u
```

101

ul2C.cpp, 102

ul2C.h, 93

```
ROS2/ros2_scara_ws/src/dalsa_bioscara_bioscara_hardware_interface/bioscara_hardware_interface/bioscara_hardware.hpp
                                                             joint.h, 68
         103, 104
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardwa8E_TiMAXA/EE/sDC/bTo/scara_hardware.cpp,
                                                             Joint, 41
         106
ROS2/ros2_scara_ws/src/dalsa_bioscara/README.md,
                                                             joint.h, 68
                                                        setMaxVelocity
         77
RPI PWM, 59
                                                             Joint, 51
     ∼RPI PWM, 59
                                                        setPeriod
     chippath, 61
                                                             RPI PWM, 60
     disable, 59
                                                        setPosition
     enable, 59
                                                             Gripper, 36
                                                            Joint, 52
    per, 61
                                                        SETRPM
    pwmpath, 61
                                                             Joint, 41
     setDutyCycle, 59
     setDutyCycleNS, 60
                                                             joint.h, 68
                                                        SETUP
     setPeriod, 60
     start, 60
                                                             Joint, 41
                                                            joint.h, 68
     stop. 60
     writeSYS, 60
                                                        setup, 20
RUNCOTINOUS
                                                             data_files, 21
    Joint, 41
                                                             description, 21
    joint.h, 68
                                                             entry points, 21
                                                             install_requires, 21
rx_buf
     joint.ino, 75
                                                             joint.ino, 75
rx_data_ready
                                                             license, 21
    joint.ino, 75
                                                             maintainer, 21
rx_length
                                                             maintainer_email, 21
    joint.ino, 75
                                                             name, 21
                                                             package name, 21
SETBRAKEMODE
                                                             packages, 21
     Joint, 41
                                                             tests_require, 21
    joint.h, 69
                                                             version, 22
setBrakeMode
                                                            zip_safe, 22
     Joint. 50
                                                        setVelocity
SETCONTROLTHRESHOLD
                                                             Joint, 52
     Joint, 41
    joint.h, 69
                                                             bioscara_hardware_interface::BioscaraHardwareInterface::joint_hom
SETCURRENT
                                                                 58
    Joint, 41
                                                        stall threshold
     joint.h, 68
                                                             bioscara hardware interface::BioscaraHardwareInterface::joint conf
setDriveCurrent
                                                                 57
     Joint, 50
                                                        start
setDutyCycle
                                                             RPI PWM, 60
     RPI PWM, 59
                                                        startHoming
setDutyCycleNS
                                                             Joint, 52
     RPI_PWM, 60
                                                        stepper
SETHOLDCURRENT
                                                             joint.ino, 75
    Joint, 41
                                                        STOP
    joint.h, 68
                                                             Joint, 41
setHoldCurrent
                                                             joint.h, 69
     Joint, 50
                                                        stop
setHomingOffset
                                                             Joint, 53
     Joint, 51
                                                             RPI PWM, 60
SETMAXACCELERATION
                                                        stp_reg_t
    Joint, 41
                                                             Joint, 40
    joint.h, 68
                                                             joint.h, 68
setMaxAcceleration
     Joint, 51
                                                        test_copyright, 22
SETMAXDECELERATION
                                                             test_copyright, 22
```

```
test_flake8, 22
    test_flake8, 22
test_joint_trajectory_controller, 22
    generate_launch_description, 23
test_pep257, 23
    test pep257, 23
tests_require
    setup, 21
threshold
    bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing_config_t,
Todo List, 9
tx_buf
    joint.ino, 76
tx_data_ready
    joint.ino, 76
tx_length
    joint.ino, 76
ul2C.cpp
    closeI2CDevHandle, 101
    openI2CDevHandle, 102
    readFromI2CDev, 102
    writeToI2CDev, 102
ul2C.h
     ACK, 91
    closeI2CDevHandle, 92
    MAX_BUFFER, 91
    NACK, 92
    openI2CDevHandle, 92
    readFromI2CDev, 93
     RFLAGS SIZE, 92
    writeToI2CDev, 93
version
    setup, 22
wait_while_busy
     Joint, 53
write
    bioscara_hardware_interface::BioscaraHardwareInterface,
         32
    Joint, 53
writeSYS
     RPI PWM, 60
writeToI2CDev
    ul2C.cpp, 102
     ul2C.h, 93
writeValue
    joint.h, 69
zip_safe
    setup, 22
```