Bioscara

Generated by Doxygen 1.9.8

Chapter 1

Documentation

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

- The joint firmware in the `/Arduino` directory
- The interfacing library used for communicating with the joints in the `/ROS2` directory.

1.1 Usage

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

Chapter 2

README

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

Chapter 3

README

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

Chapter 4

README

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

Chapter 5

README

The packages are structured according to this guide: RTW Package Structure

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

```
install(
  DIRECTORY hardware/include/
  DESTINATION include/ros2_control_demo_example_1
)
install(
  DIRECTORY description/launch description/ros2_control description/urdf
  DESTINATION share/ros2_control_demo_example_1
)
install(
  DIRECTORY bringup/launch bringup/config
  DESTINATION share/ros2_control_demo_example_1
)
install(TARGETS ros2_control_demo_example_1
  EXPORT export_ros2_control_demo_example_1
  ARCHIVE DESTINATION lib
  LIBRARY DESTINATION lib
  RUNTIME DESTINATION bin
)
```

TODO:

- [ ] Format and rework this content

Chapter 6

Todo List

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

Implement a return code for read only functions

- Implement clearStall function

Member bioscara_hardware_interface::BioscaraHardwareInterface::on_init (const hardware_interface::↩ HardwareComponentInterfaceParams &params) override

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

Chapter 7

Namespace Index

7.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 8

Hierarchical Index

8.1 Class Hierarchy

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

Chapter 9

Class Index

9.1 Class List

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

Chapter 10

File Index

10.1 File List

Here is a list of all files with brief descriptions:

Chapter 11

Namespace Documentation

11.1 bioscara Namespace Reference

Functions

- generate_launch_description ()

11.1.1 Function Documentation

11.1.1.1 generate_launch_description()

```
bioscara.generate_launch_description ( )
```

11.2 bioscara_hardware_driver Namespace Reference

Generic BaseGripper object to interact with the robot gripper.

Classes

- class BaseGripper
- class BaseJoint

  TODO.
- class Gripper
- class Joint

  Representing a single joint on the I2C bus.
- class MockGripper
- class MockJoint

Enumerations

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

  Enum defining common error types.

Functions

- std::string error_to_string (err_type_t err)

  Converts an error code to a string and returns it.

11.2.1 Detailed Description

Generic BaseGripper object to interact with the robot gripper.

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

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

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

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

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

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

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

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

11.2.2 Enumeration Type Documentation

11.2.2.1 err_type_t

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

Enum defining common error types.

Enumerator

OK	
ERROR	
NOT_HOMED	
NOT_ENABLED	
STALLED	
NOT_INIT	
COMM_ERROR	
INVALID_ARGUMENT	
INCORRECT_STATE	

11.2.3 Function Documentation

11.2.3.1 error_to_string()

```
std::string bioscara_hardware_driver::error_to_string (
            err_type_t err )
```

Converts an error code to a string and returns it.

Parameters

err	

Returns

std::string

11.3 bioscara_hardware_interface Namespace Reference

Classes

- class BioscaraHardwareInterface

  The bioscara hardware interface class.

Variables

- constexpr char HW_IF_HOME [ ] = "home"

11.3.1 Variable Documentation

11.3.1.1 HW_IF_HOME

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

11.4 display Namespace Reference

Functions

- generate_launch_description ()

11.4.1 Function Documentation

11.4.1.1 generate_launch_description()

```
display.generate_launch_description ( )
```

11.5 gazebo Namespace Reference

Functions

- generate_launch_description ()

11.5.1 Function Documentation

11.5.1.1 generate_launch_description()

```
gazebo.generate_launch_description ( )
```

11.6 setup Namespace Reference

Variables

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

11.6.1 Variable Documentation

11.6.1.1 data_files

```
setup.data_files
```

11.6.1.2 description

```
setup.description
```

11.6.1.3 entry_points

```
setup.entry_points
```

11.6.1.4 install_requires

```
setup.install_requires
```

11.6.1.5 license

```
setup.license
```

11.6.1.6 maintainer

```
setup.maintainer
```

11.6.1.7 maintainer_email

```
setup.maintainer_email
```

11.6.1.8 name

```
setup.name
```

11.6.1.9 package_name

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

11.6.1.10 packages

```
setup.packages
```

11.6.1.11 tests_require

```
setup.tests_require
```

11.6.1.12 version

```
setup.version
```

11.6.1.13 zip_safe

```
setup.zip_safe
```

11.7 test_joint_trajectory_controller Namespace Reference

Functions

- generate_launch_description ()

11.7.1 Function Documentation

11.7.1.1 generate_launch_description()

```
test_joint_trajectory_controller.generate_launch_description ( )
```

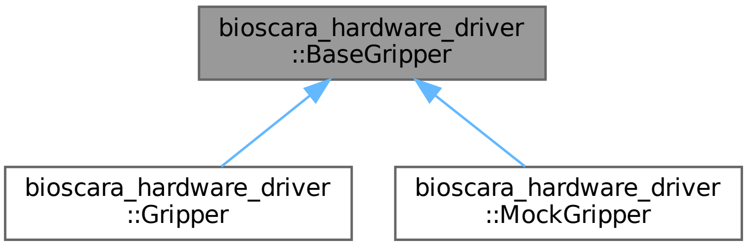
Chapter 12

Class Documentation

12.1 bioscara_hardware_driver::BaseGripper Class Reference

`#include <mBaseGripper.h>`

Inheritance diagram for bioscara_hardware_driver::BaseGripper:


bioscara_hardware_driver ::BaseGripper
bioscara_hardware_driver ::Gripper
bioscara_hardware_driver ::MockGripper

Public Member Functions

- BaseGripper (void)
- virtual err_type_t init (void)

  Placeholder, does nothing.
- virtual err_type_t deinit (void)

  Placeholder, does nothing.
- virtual err_type_t enable (void)

  Prepares the servo for use.
- virtual err_type_t disable (void)

  Disables the servo.
- virtual err_type_t setPosition (float width)

  Sets the gripper width in m from the closed position.
- virtual err_type_t setServoPosition (float angle)

  Sets the servo position of the gripper actuator in degrees.
- virtual void setReduction (float reduction)

  Manually set reduction.
- virtual void setOffset (float offset)

  Manually set offset.

12.1.1 Constructor & Destructor Documentation

12.1.1.1 BaseGripper()

```
bioscara_hardware_driver::BaseGripper::BaseGripper (
             void  )
```

12.1.2 Member Function Documentation

12.1.2.1 deinit()

```
err_type_t bioscara_hardware_driver::BaseGripper::deinit (
             void  )  [virtual]
```

Placeholder, does nothing.

Returns

0

12.1.2.2 disable()

```
err_type_t bioscara_hardware_driver::BaseGripper::disable (
             void  )  [virtual]
```

Disables the servo.

Returns

non-zero error code.

Reimplemented in bioscara_hardware_driver::Gripper.

12.1.2.3 enable()

```
err_type_t bioscara_hardware_driver::BaseGripper::enable (
             void  )  [virtual]
```

Prepares the servo for use.

Returns

non-zero error code.

Reimplemented in bioscara_hardware_driver::Gripper.

12.1.2.4 init()

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

Placeholder, does nothing.

Returns

0

12.1.2.5 setOffset()

```
void bioscara_hardware_driver::BaseGripper::setOffset (
             float offset )  [virtual]
```

Manually set offset.

Reimplemented in bioscara_hardware_driver::Gripper.

12.1.2.6 setPosition()

```
err_type_t bioscara_hardware_driver::BaseGripper::setPosition (
             float width )  [virtual]
```

Sets the gripper width in m from the closed position.

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

Parameters

width	width in m.

Reimplemented in bioscara_hardware_driver::Gripper.

12.1.2.7 setReduction()

```
void bioscara_hardware_driver::BaseGripper::setReduction (
             float reduction )  [virtual]
```

Manually set reduction.

Parameters

reduction	

Reimplemented in bioscara_hardware_driver::Gripper.

12.1.2.8 setServoPosition()

```
err_type_t bioscara_hardware_driver::BaseGripper::setServoPosition (
            float angle )  [virtual]
```

Sets the servo position of the gripper actuator in degrees.

Parameters

angle	in degrees.

Reimplemented in bioscara_hardware_driver::Gripper.

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

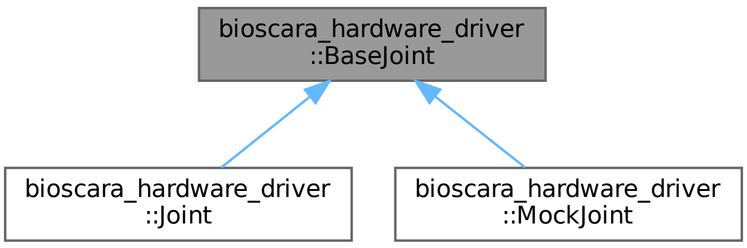
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mBaseGripper.h
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mBaseGripper.cpp

12.2 bioscara_hardware_driver::BaseJoint Class Reference

TODO.

`#include <mBaseJoint.h>`

Inheritance diagram for bioscara_hardware_driver::BaseJoint:


bioscara_hardware_driver ::BaseJoint
bioscara_hardware_driver ::Joint
bioscara_hardware_driver ::MockJoint

Public Types

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

  register and command definitions

Public Member Functions

- BaseJoint (const std::string name)

  Create a Joint object.
- ∼BaseJoint (void)
- virtual err_type_t init (void)

  Initialization, derived classes may override this.
- virtual err_type_t deinit (void)

  Deinitialization, derived classes may override this.
- virtual err_type_t enable (u_int8_t driveCurrent, u_int8_t holdCurrent)

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

  disenganges the joint motors, derived classes may override this.
- virtual err_type_t home (float velocity, u_int8_t sensitivity, u_int8_t current)

  Blocking implementation to home the joint, derived classes may override this.
- virtual err_type_t startHoming (float velocity, u_int8_t sensitivity, u_int8_t current)

  non-blocking implementation to home the joint, derived classes may override this.
- virtual err_type_t postHoming (void)

  perform tasks after a non-blocking homing, derived classes may override this.
- virtual err_type_t getPosition (float &pos)=0

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

  get the current joint position in radians or m for cylindrical and prismatic joints respectively. Derived class mayy override this.
- virtual err_type_t moveSteps (int32_t steps)

  Move full steps. Derived class may override this.
- virtual err_type_t getVelocity (float &vel)=0

  get the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class must override this.
- virtual err_type_t setVelocity (float vel)

  Set the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.
- virtual err_type_t checkOrientation (float angle=10.0)

  Calls the checkOrientation method of the motor. Checks in which direction the motor is turning. Derived class may override this.
- virtual err_type_t stop (void)

  Stops the motor. Derived class may override this.
- virtual err_type_t disableCL (void)

  Disables the Closed-Loop PID Controller Derived class may override this.
- virtual err_type_t setDriveCurrent (u_int8_t current)

  Set the Drive Current. Derived class may override this.
- virtual err_type_t setHoldCurrent (u_int8_t current)

  Set the Hold Current. Derived class may override this.
- virtual err_type_t setBrakeMode (u_int8_t mode)

  Set Brake Mode. Derived class may override this.
- virtual err_type_t setMaxAcceleration (float maxAccel)

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

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

- virtual err_type_t enableStallguard (u_int8_t sensitivity)

    Enable encoder stall detection of the joint. Derived class may override this.
- virtual bool isHomed (void)

    Checks the state if the motor is homed.
- virtual bool isEnabled (void)

    Checks the state if the motor is enabled.
- virtual bool isStalled (void)

    Checks if the motor is stalled.
- virtual bool isBusy (void)

    Checks if the joint controller is busy processing a blocking command.
- virtual err_type_t getFlags (u_int8_t &flags)
- virtual err_type_t getFlags (void)
- virtual stp_reg_t getCurrentBCmd (void)

    get the currently active blocking command

Public Attributes

- std::string name

Protected Member Functions

- virtual void wait_while_busy (const float period_ms)

    Blocking loop waiting for BUSY flag to reset.
- virtual err_type_t _home (float velocity, u_int8_t sensitivity, u_int8_t current)=0

    Call to start the homing sequence of a joint.

Protected Attributes

- u_int8_t flags = 0b00001100

    State flags transmitted with every I2C transaction.
- stp_reg_t current_b_cmd = NONE

    Keeps track if a blocking command is being executed.

12.2.1 Detailed Description

TODO.

12.2.2 Member Enumeration Documentation

12.2.2.1 stp_reg_t

```
enum bioscara_hardware_driver::BaseJoint::stp_reg_t
```

register and command definitions

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

Enumerator

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

12.2.3 Constructor & Destructor Documentation

12.2.3.1 BaseJoint()

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

Create a Joint object.

The Joint object represents a single joint.

Parameters

name	string device name for identification

12.2.3.2 ∼BaseJoint()

```
bioscara_hardware_driver::BaseJoint::∼BaseJoint (
            void )
```

12.2.4 Member Function Documentation

12.2.4.1 _home()

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

Call to start the homing sequence of a joint.

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

Parameters

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.

Implemented in bioscara_hardware_driver::Joint, and bioscara_hardware_driver::MockJoint.

12.2.4.2 checkOrientation()

```
err_type_t bioscara_hardware_driver::BaseJoint::checkOrientation (
            float angle = 10.0 ) [virtual]
```

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

As the orientation check is blocking on the motor, this 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.

Reimplemented in bioscara_hardware_driver::Joint, and bioscara_hardware_driver::MockJoint.

12.2.4.3 deinit()

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

Deinitialization, derived classes may override this.

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.4 disable()

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

disenganges the joint motors, derived classes may override this.

Returns

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

Reimplemented in bioscara_hardware_driver::MockJoint.

12.2.4.5 disableCL()

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

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

Returns

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

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.6 enable()

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

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

Reimplemented in bioscara_hardware_driver::Joint, and bioscara_hardware_driver::MockJoint.

12.2.4.7 enableStallguard()

```
err_type_t bioscara_hardware_driver::BaseJoint::enableStallguard (
             u_int8_t sensitivity )  [virtual]
```

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

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

Note

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

Parameters

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

Returns

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

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.8 getCurrentBCmd()

```
BaseJoint::stp_reg_t bioscara_hardware_driver::BaseJoint::getCurrentBCmd (
             void  )  [virtual]
```

get the currently active blocking command

Returns

The the command of type stp_reg_t

12.2.4.9 getFlags() [1/2]

```
err_type_t bioscara_hardware_driver::BaseJoint::getFlags (
             u_int8_t & flags )  [virtual]
```

ping the joint to get the latest driver state flags

Parameters

flags	if succesfull, populated with the latest flags

Returns

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

12.2.4.10 getFlags() [2/2]

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

Overload of BaseJoint::getFlags(u_int8_t &flags)

Returns

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

Reimplemented in bioscara_hardware_driver::Joint, and bioscara_hardware_driver::MockJoint.

12.2.4.11 getPosition()

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

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

Warning

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

Parameters

pos	

Returns

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

Implemented in bioscara_hardware_driver::Joint, and bioscara_hardware_driver::MockJoint.

12.2.4.12 getVelocity()

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

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

Parameters

vel	

Returns

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

Implemented in bioscara_hardware_driver::Joint, and bioscara_hardware_driver::MockJoint.

12.2.4.13 home()

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

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

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

Additionally this method returns:

Returns

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

12.2.4.14 init()

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

Initialization, derived classes may override this.

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.15 isBusy()

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

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

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

Returns

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

12.2.4.16 isEnabled()

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

Checks the state if the motor is enabled.

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

Returns

true if the motor is enabled, false if not.

12.2.4.17 isHomed()

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

Checks the state if the motor is homed.

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

Returns

true if the motor is homed, false if not.

Reimplemented in bioscara_hardware_driver::MockJoint.

12.2.4.18 isStalled()

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

Checks if the motor is stalled.

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

Returns

true if the motor is stalled, false if not.

12.2.4.19 moveSteps()

```
err_type_t bioscara_hardware_driver::BaseJoint::moveSteps (
             int32_t steps ) [virtual]
```

Move full steps. Derived class may override this.

This function can be called even when not homed.

Parameters

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.

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.20 postHoming()

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

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

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

Returns

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

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.21 setBrakeMode()

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

Set Brake Mode. Derived class may override this.

Parameters

mode	Freewheel: 0, Coolbrake: 1, Hardbrake: 2

Returns

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

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.22 setDriveCurrent()

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

Set the Drive Current. Derived class may override this.

Warning

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

Parameters

current	0% - 100% of driver current

Returns

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

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.23 setHoldCurrent()

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

Set the Hold Current. Derived class may override this.

Warning

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

Parameters

current	0% - 100% of driver current

Returns

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

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.24 setMaxAcceleration()

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

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

Parameters

maxAccel	maximum joint acceleration.

Returns

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

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.25 setMaxVelocity()

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

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

Parameters

maxVel	maximum joint velocity.

Returns

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

Reimplemented in bioscara_hardware_driver::Joint.

12.2.4.26 setPosition()

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

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

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.

Reimplemented in bioscara_hardware_driver::Joint, and bioscara_hardware_driver::MockJoint.

12.2.4.27 setVelocity()

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

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

Parameters

vel	

Returns

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

Reimplemented in bioscara_hardware_driver::Joint, and bioscara_hardware_driver::MockJoint.

12.2.4.28 startHoming()

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

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

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

12.2.4.29 stop()

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

Stops the motor. Derived class may override this.

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

Returns

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

Reimplemented in bioscara_hardware_driver::Joint, and bioscara_hardware_driver::MockJoint.

12.2.4.30 wait_while_busy()

```
void bioscara_hardware_driver::BaseJoint::wait_while_busy (
             const float period_ms )  [protected], [virtual]
```

Blocking loop waiting for BUSY flag to reset.

Parameters

period_ms	time in ms between polls.

12.2.5 Member Data Documentation

12.2.5.1 current_b_cmd

```
stp_reg_t bioscara_hardware_driver::BaseJoint::current_b_cmd = NONE  [protected]
```

Keeps track if a blocking command is being executed.

12.2.5.2 flags

```
u_int8_t bioscara_hardware_driver::BaseJoint::flags = 0b00001100  [protected]
```

State flags transmitted with every I2C transaction.

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

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

STALL is set if a stall from the stall detection is sensed and the joint is stopped. The flag is cleared when the joint is homed or the Stallguard enabled.
BUSY is set if the slave is busy processing a previous command.
NOTHOMED is cleared if the joint is homed. Movement is only allowed if this flag is clear
NOTENABLED is cleared if the joint is enabled after calling Joint::enable()

12.2.5.3 name

```
std::string bioscara_hardware_driver::BaseJoint::name
```

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

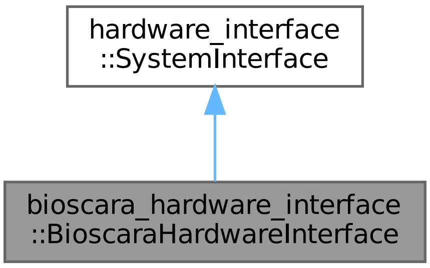
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mBaseJoint.h
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mBaseJoint.cpp

12.3 bioscara_hardware_interface::BioscaraHardwareInterface Class Reference

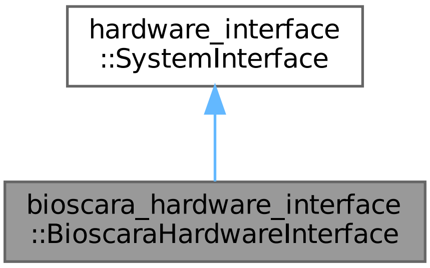
The bioscara hardware interface class.

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

Inheritance diagram for bioscara_hardware_interface::BioscaraHardwareInterface:



Collaboration diagram for bioscara_hardware_interface::BioscaraHardwareInterface:



Classes

- struct joint_config_t

  configuration structure holding the passed paramters from the ros2_control urdf
- struct joint_homing_config_t

  configuration structure holding the passed homing paramters from the ros2_control urdf

Public Member Functions

- hardware_interface::CallbackReturn on_init (const hardware_interface::HardwareComponentInterface↩ Params &params) override

  Called on initialization to the `unconfigured` state.
- hardware_interface::CallbackReturn on_shutdown (const rclcpp_lifecycle::State &previous_state) override

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

- hardware_interface::CallbackReturn on_configure (const rclcpp_lifecycle::State &previous_state) override

  Called on the 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::return_type perform_command_mode_switch (const std::vector< std::string > &start←_interfaces, const std::vector< std::string > &stop_interfaces) override

  Perform the mode-switching for the new command interface combination.
- hardware_interface::CallbackReturn on_error (const rclcpp_lifecycle::State &previous_state) override

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

Private Member Functions

- bioscara_hardware_driver::err_type_t start_homing (const std::string name, float velocity)

  wrapper method to start homing.
- bioscara_hardware_driver::err_type_t stop_homing (const std::string name)

  wrapper method to stop homing.
- void split_interface_string_to_joint_and_name (std::string interface, std::string &joint_name, std::string &interface_name)

  Split a interface string like "<joint_name>/<interface_name>" to "<joint_name>" and "<interface_name>".
- bioscara_hardware_driver::err_type_t activate_joint (const std::string name)

  Enables each joint, enables the stall detection and sets the maximmum acceleration.
- bioscara_hardware_driver::err_type_t deactivate_joint (const std::string name)

  Disables each joint.

Private Attributes

- std::unordered_map< std::string, std::unique_ptr< bioscara_hardware_driver::BaseJoint > > _joints

  unordered map storing the pointers to BaseJoint objects. This will either be a MockJoint or Joint.
- std::unordered_map< std::string, joint_config_t > _joint_cfg

  unordered map storing the configuration struct of the joints.
- std::unordered_map< std::string, std::set< std::string > > _joint_command_modes

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

12.3.1 Detailed Description

The bioscara hardware interface class.

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

Hardware Lifecycle
The hardware follows the ros2_control hardware interface lifecyle which intern is following the `ROS2 managed node lifecycle`.

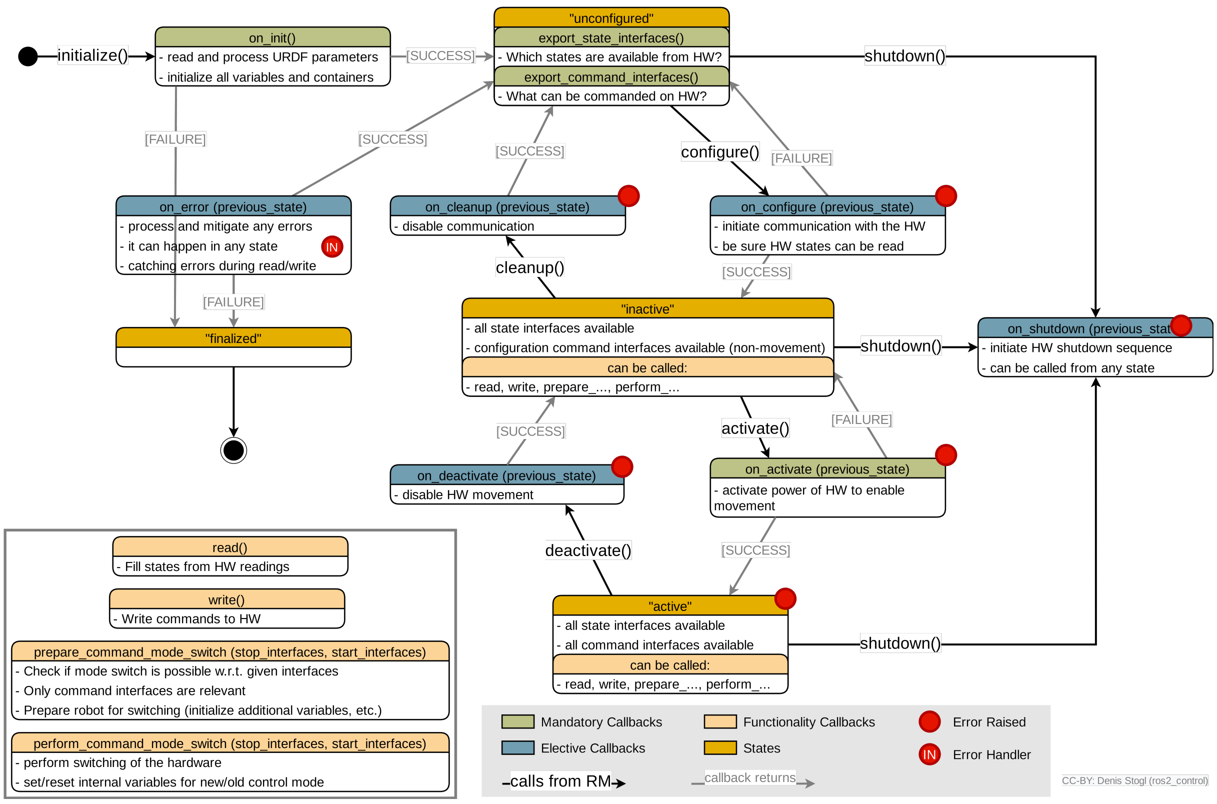

initialize()
on_init()
- read and process URDF parameters
- initialize all variables and containers
[SUCCESS]
"unconfigured"
export_state_interfaces()
- Which states are available from HW?
export_command_interfaces()
- What can be commanded on HW?
shutdown()
[FAILURE]
[SUCCESS]
[SUCCESS]
configure()
[FAILURE]
on_error (previous_state)
- process and mitigate any errors
- it can happen in any state
- catching errors during read/write
IN
on_cleanup (previous_state)
- disable communication
on_configure (previous_state)
- initiate communication with the HW
- be sure HW states can be read
cleanup()
[SUCCESS]
[FAILURE]
"finalized"
"inactive"
- all state interfaces available
- configuration command interfaces available (non-movement)
can be called:
- read, write, prepare_..., perform_...
shutdown()
on_shutdown (previous_stat
- initiate HW shutdown sequence
- can be called from any state
[SUCCESS]
activate()
[FAILURE]
on_deactivate (previous_state)
- disable HW movement
on_activate (previous_state)
- activate power of HW to enable movement
deactivate()
[SUCCESS]
"active"
- all state interfaces available
- all command interfaces available
can be called:
- read, write, prepare_..., perform_...
shutdown()
read()
- Fill states from HW readings
write()
- Write commands to HW
prepare_command_mode_switch (stop_interfaces, start_interfaces)
- Check if mode switch is possible w.r.t. given interfaces
- Only command interfaces are relevant
- Prepare robot for switching (initialize additional variables, etc.)
perform_command_mode_switch (stop_interfaces, start_interfaces)
- perform switching of the hardware
- set/reset internal variables for new/old control mode
Mandatory Callbacks
Elective Callbacks
Functionality Callbacks
States
Error Raised
IN
Error Handler
calls from RM
callback returns
CC-BY: Denis Stogl (ros2_control)

Figure 12.1 Hardware interface lifecycle

12.3.2 Member Function Documentation

12.3.2.1 activate_joint()

```
bioscara_hardware_driver::err_type_t bioscara_hardware_interface::BioscaraHardwareInterface↩
::activate_joint (
            const std::string name ) [private]
```

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

Parameters

name	joint name to enable

Returns

bioscara_hardware_driver::err_type_t

12.3.2.2 deactivate_joint()

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

Disables each joint.

Parameters

name	joint name to disable

Returns

bioscara_hardware_driver::err_type_t

12.3.2.3 on_activate()

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

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

Calls activate_joint() to enable the joints.
It is allowed to activate the hardware even if it is not homed. To home the joint the homing_controller must be activated, but generally a hardware component must be active in order for controllers to become active.
To prohibit movement on activation the set point for each position command interface is set equal to the current measured position, and the velocity command is set to 0.0 for each command interface. The current values are obtained by calling the read() method once which populates the state interfaces with values.

Parameters

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.

12.3.2.4 on_cleanup()

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

Called on the 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

12.3.2.5 on_configure()

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

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

Establish and test connection to each joint.

Parameters

previous_state	

Returns

hardware_interface::CallbackReturn

12.3.2.6 on_deactivate()

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

Called on the 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

12.3.2.7 on_error()

```
hardware_interface::CallbackReturn bioscara_hardware_interface::BioscaraHardwareInterface↩
::on_error (
             const rclcpp_lifecycle::State & previous_state )  [override]
```

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

According to the `ros2_control documentation`:

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

Since the hardware will 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

12.3.2.8 on_init()

```
hardware_interface::CallbackReturn bioscara_hardware_interface::BioscaraHardwareInterface↩
::on_init (
```

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

Called on initialization to the `unconfigured` state.

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

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

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

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

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

Parameters

params	

Returns

hardware_interface::CallbackReturn

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

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

12.3.2.9 on_shutdown()

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

Called on the 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

12.3.2.10 perform_command_mode_switch()

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

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

Performs the following actions:

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

Note

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

Parameters

`in`	*start_interfaces*	vector of string identifiers for the command interfaces starting.
`in`	*stop_interfaces*	vector of string identifiers for the command interfaces stopping.

Returns

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

12.3.2.11 prepare_command_mode_switch()

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

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

For safe operation only one controller may interact with the hardware at the time. For example if the velocity JTC is active and has claimed the velocity command interfaces it is technically possible to activate the position JTC (or a homing controller, or others) that claim a different command interface (position in this case). However if both controllers are active they start writing to the hardware simultaneously which is to be avoided. For this reason a book keeping mechanism has been implemented which stores the currently active command interfaces for each joint in the _joint_command_modes member. Each joint has a set of active command interfaces. When a controller switch is performed the interfaces that should be stopped are removed from each joint set, then the one that should be started are added, if they are already present an error is thrown. Lastly a validation is performed. Currently the validation is simple since each joint may only have one command interface. The validation can be expanded for 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.
  - [ERROR] Activating 'position' or 'velocity' command interface if the joint is not homed (Joint::isHomed() == false).

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

12.3.2.12 read()

```
hardware_interface::return_type bioscara_hardware_interface::BioscaraHardwareInterface::read (
            const rclcpp::Time & time,
            const rclcpp::Duration & period )  [override]
```

Reads from the hardware and populates the state interfaces.

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

- State interface "position" -> Joint::getPosition()
- State interface "velocity" -> Joint::getVelocity()
- State interface "home" -> Joint::isHomed()
  - This does not actually trigger a communication, instead it relies on the return flags of the previous transmissions. Since position and velocity have been called 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

12.3.2.13 split_interface_string_to_joint_and_name()

```
void bioscara_hardware_interface::BioscaraHardwareInterface::split_interface_string_to_joint↩
_and_name (
            std::string interface,
            std::string & joint_name,
            std::string & interface_name )  [private]
```

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

Parameters

interface	
joint_name	
interface_name	

12.3.2.14 start_homing()

```
bioscara_hardware_driver::err_type_t bioscara_hardware_interface::BioscaraHardwareInterface↩
::start_homing (
             const std::string name,
             float velocity )  [private]
```

wrapper method to start homing.

Activate the joint, set homing acceleration and start homing.

Parameters

name	
velocity	

Returns

bioscara_hardware_driver::err_type_t

12.3.2.15 stop_homing()

```
bioscara_hardware_driver::err_type_t bioscara_hardware_interface::BioscaraHardwareInterface↩
::stop_homing (
            const std::string name )  [private]
```

wrapper method to stop homing.

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

Parameters

name	

Returns

bioscara_hardware_driver::err_type_t

12.3.2.16 write()

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

Writes commands to the hardware from the command interfaces.

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

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

- Command interface "home" -> Joint::startHoming()
  - If the commanded value in "home" is != 0.0 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

12.3.3 Member Data Documentation

12.3.3.1 _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.

12.3.3.2 _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.

12.3.3.3 _joints

```
std::unordered_map<std::string, std::unique_ptr<bioscara_hardware_driver::BaseJoint> > bioscara↩
_hardware_interface::BioscaraHardwareInterface::_joints [private]
```

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

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

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

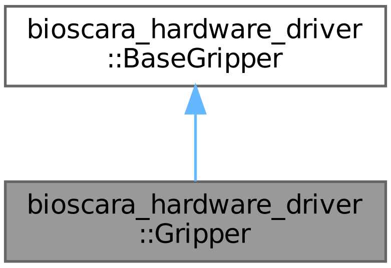
The documentation for this class was generated from the following files:

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

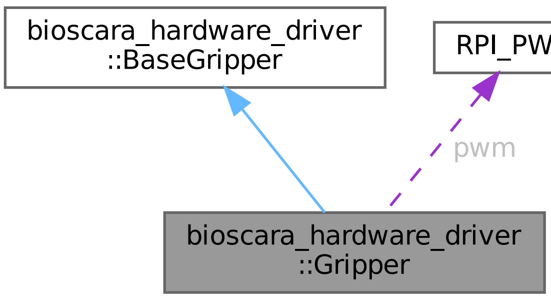
12.4 bioscara_hardware_driver::Gripper Class Reference

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

Inheritance diagram for bioscara_hardware_driver::Gripper:


bioscara_hardware_driver ::BaseGripper
bioscara_hardware_driver ::Gripper

Collaboration diagram for bioscara_hardware_driver::Gripper:



Public Member Functions

- Gripper (float reduction, float offset, float min, float max)

  Constructor of the hardware Gripper object.
- err_type_t enable (void) override

  Prepares the servo for use.
- err_type_t disable (void) override

  Disables the servo.
- err_type_t setPosition (float width) override

  Sets the gripper width in m from the closed position.
- err_type_t setServoPosition (float angle) override

  Sets the servo position of the gripper actuator in degrees.
- void setReduction (float reduction)

  Manually set reduction.
- void setOffset (float offset)

  Manually set offset.

Public Member Functions inherited from bioscara_hardware_driver::BaseGripper

- BaseGripper (void)
- virtual err_type_t init (void)

  Placeholder, does nothing.
- virtual err_type_t deinit (void)

  Placeholder, does nothing.

Protected Attributes

- float reduction = 1

  Joint to actuator reduction ratio.
- float offset = 0

  Joint position offset.
- float min = 0

  Joint lower limit.
- float max = 0

  Joint upper limit.

Private Attributes

- RPI_PWM pwm
- int freq = 50

12.4.1 Constructor & Destructor Documentation

12.4.1.1 Gripper()

```
bioscara_hardware_driver::Gripper::Gripper (
              float reduction,
              float offset,
              float min,
              float max )
```

Constructor of the hardware Gripper object.

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

Parameters

reduction	
offset	
min	minimum width in m.
max	maxmimum width in m.

12.4.2 Member Function Documentation

12.4.2.1 disable()

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

Disables the servo.

Stops the servo and disables the PWM generation.

Returns

return code of bioscara_hardware_driver::esp_err_t type.

Reimplemented from bioscara_hardware_driver::BaseGripper.

12.4.2.2 enable()

```
err_type_t bioscara_hardware_driver::Gripper::enable (
              void  )  [override], [virtual]
```

Prepares the servo for use.

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

Returns

return code of bioscara_hardware_driver::esp_err_t type

Reimplemented from bioscara_hardware_driver::BaseGripper.

12.4.2.3 setOffset()

```
void bioscara_hardware_driver::Gripper::setOffset (
             float offset )  [virtual]
```

Manually set offset.

Reimplemented from bioscara_hardware_driver::BaseGripper.

12.4.2.4 setPosition()

```
err_type_t bioscara_hardware_driver::Gripper::setPosition (
             float width )  [override], [virtual]
```

Sets the gripper width in m from the closed position.

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

Parameters

width	width in m.

Reimplemented from bioscara_hardware_driver::BaseGripper.

12.4.2.5 setReduction()

```
void bioscara_hardware_driver::Gripper::setReduction (
             float reduction )  [virtual]
```

Manually set reduction.

Parameters

reduction	

Reimplemented from bioscara_hardware_driver::BaseGripper.

12.4.2.6 setServoPosition()

```
err_type_t bioscara_hardware_driver::Gripper::setServoPosition (
             float angle )  [override], [virtual]
```

Sets the servo position of the gripper actuator in degrees.

Parameters

angle	in degrees.

Reimplemented from bioscara_hardware_driver::BaseGripper.

12.4.3 Member Data Documentation

12.4.3.1 freq

```
int bioscara_hardware_driver::Gripper::freq = 50  [private]
```

12.4.3.2 max

```
float bioscara_hardware_driver::Gripper::max = 0  [protected]
```

Joint upper limit.

12.4.3.3 min

```
float bioscara_hardware_driver::Gripper::min = 0  [protected]
```

Joint lower limit.

12.4.3.4 offset

```
float bioscara_hardware_driver::Gripper::offset = 0  [protected]
```

Joint position offset.

12.4.3.5 pwm

```
RPI_PWM bioscara_hardware_driver::Gripper::pwm  [private]
```

12.4.3.6 reduction

```
float bioscara_hardware_driver::Gripper::reduction = 1  [protected]
```

Joint to actuator reduction ratio.

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

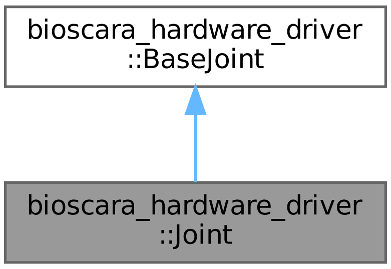
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mGripper.h
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mGripper.cpp

12.5 bioscara_hardware_driver::Joint Class Reference

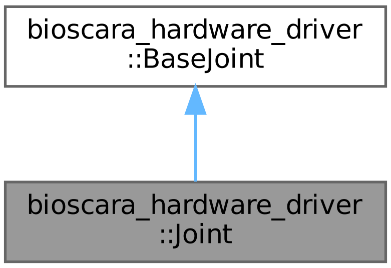
Representing a single joint on the I2C bus.

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

Inheritance diagram for bioscara_hardware_driver::Joint:



Collaboration diagram for bioscara_hardware_driver::Joint:



Public Member Functions

- Joint (const std::string name, const int address, const float reduction, const float min, const float max)

    Create a Joint object.
- ∼Joint (void)
- err_type_t init (void) override

    Established connection to a joint via I2C.
- err_type_t deinit (void) override

    Disconnects from a joint.
- err_type_t enable (u_int8_t driveCurrent, u_int8_t holdCurrent) override

    Setup the joint and engages motor.

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

Public Member Functions inherited from bioscara_hardware_driver::BaseJoint

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

- virtual bool isEnabled (void)

    Checks the state if the motor is enabled.
- virtual bool isStalled (void)

    Checks if the motor is stalled.
- virtual bool isBusy (void)

    Checks if the joint controller is busy processing a blocking command.
- virtual err_type_t getFlags (u_int8_t &flags)
- virtual stp_reg_t getCurrentBCmd (void)

    get the currently active blocking command

Protected Member Functions

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

    Call to start the homing sequence of a joint.
- err_type_t checkCom (void)

    Check if communication to the joint is established.

Protected Member Functions inherited from bioscara_hardware_driver::BaseJoint

- virtual void wait_while_busy (const float period_ms)

    Blocking loop waiting for BUSY flag to reset.

Protected Attributes

- float reduction = 1

    Joint to actuator reduction ratio.
- float offset = 0

    Joint position offset.
- float min = 0

    Joint lower limit.
- float max = 0

    Joint upper limit.

Protected Attributes inherited from bioscara_hardware_driver::BaseJoint

- u_int8_t flags = 0b00001100

    State flags transmitted with every I2C transaction.
- stp_reg_t current_b_cmd = NONE

    Keeps track if a blocking command is being executed.

Private Member Functions

- template<typename T >
  int read (const stp_reg_t reg, T &data, u_int8_t &flags)

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

    Wrapper function to send command to the I2C slave.

Private Attributes

- int address

  I2C adress.
- int handle = -1

  I2C bus handle.

Additional Inherited Members

Public Types inherited from bioscara_hardware_driver::BaseJoint

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

  register and command definitions

Public Attributes inherited from bioscara_hardware_driver::BaseJoint

- std::string name

12.5.1 Detailed Description

Representing a single joint on the I2C bus.

12.5.2 Constructor & Destructor Documentation

12.5.2.1 Joint()

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

Create a Joint object.

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

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

12.5.2.2 ∼Joint()

```
bioscara_hardware_driver::Joint::∼Joint (
            void )
```

12.5.3 Member Function Documentation

12.5.3.1 _home()

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

Call to start the homing sequence of a joint.

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

Parameters

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.

Implements bioscara_hardware_driver::BaseJoint.

12.5.3.2 checkCom()

```
err_type_t bioscara_hardware_driver::Joint::checkCom (
             void )  [protected]
```

Check if communication to the joint is established.

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

Returns

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

12.5.3.3 checkOrientation()

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

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

As the orientation check is blocking on the motor, this 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.

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.4 deinit()

```
err_type_t bioscara_hardware_driver::Joint::deinit (
             void )  [override], [virtual]
```

Disconnects from a joint.

Removes the joint from the I2C bus.

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.5 disableCL()

```
err_type_t bioscara_hardware_driver::Joint::disableCL (
             void  )  [override], [virtual]
```

Disables the Closed-Loop PID Controller.

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.6 enable()

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

Setup the joint and engages motor.

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.7 enableStallguard()

```
err_type_t bioscara_hardware_driver::Joint::enableStallguard (
             u_int8_t sensitivity )  [override], [virtual]
```

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

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

Note

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

Parameters

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

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.8 getFlags()

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

get driver state flags

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.9 getHomingOffset()

```
err_type_t bioscara_hardware_driver::Joint::getHomingOffset (
            float & offset )
```

Retrieves the homing position from the last homing.

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

Returns

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

12.5.3.10 getPosition()

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

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

Warning

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

Parameters

pos	

Returns

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

Implements bioscara_hardware_driver::BaseJoint.

12.5.3.11 getVelocity()

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

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

Parameters

vel	

Returns

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

Implements bioscara_hardware_driver::BaseJoint.

12.5.3.12 init()

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

Established connection to a joint via I2C.

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

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.13 moveSteps()

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

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.

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.14 postHoming()

```
err_type_t bioscara_hardware_driver::Joint::postHoming (
            void  )  [override], [virtual]
```

perform tasks after a non-blocking homing.

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

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.15 read()

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

Wrapper function to request data from the I2C slave.

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

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

Template Parameters

T	Datatype of value to be transmitted

Parameters

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

Returns

0 on OK, negative on error

12.5.3.16 setBrakeMode()

```
err_type_t bioscara_hardware_driver::Joint::setBrakeMode (
              u_int8_t mode )  [override], [virtual]
```

Set Brake Mode.

Parameters

mode	Freewheel: 0, Coolbrake: 1, Hardbrake: 2

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.17 setDriveCurrent()

```
err_type_t bioscara_hardware_driver::Joint::setDriveCurrent (
              u_int8_t current )  [override], [virtual]
```

Set the Drive Current.

Warning

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

Parameters

current	0% - 100% of driver current

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.18 setHoldCurrent()

```
err_type_t bioscara_hardware_driver::Joint::setHoldCurrent (
             u_int8_t current )  [override], [virtual]
```

Set the Hold Current.

Warning

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

Parameters

current	0% - 100% of driver current

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.19 setHomingOffset()

```
err_type_t bioscara_hardware_driver::Joint::setHomingOffset (
             const float offset )
```

Stores the homing position on the joint.

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

Returns

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

12.5.3.20 setMaxAcceleration()

```
err_type_t bioscara_hardware_driver::Joint::setMaxAcceleration (
             float maxAccel )  [override], [virtual]
```

Set the maximum permitted joint acceleration (and deceleration) in rad/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.

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.21 setMaxVelocity()

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

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

Parameters

maxVel	maximum joint velocity.

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.22 setPosition()

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

set 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.

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.23 setVelocity()

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

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

Parameters

vel	

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.24 stop()

```
err_type_t bioscara_hardware_driver::Joint::stop (
             void  )  [override], [virtual]
```

Stops the motor.

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

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.5.3.25 write()

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

Wrapper function to send command to the I2C slave.

Allocates a buffer of size sizeof(T) + RFLAGS_SIZE. Copyies *data* to the buffer and invokes writeToI2CDev(). The flags received from the transaction are copied to *flags*. The flags are described in Joint::read().

Template Parameters

T	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

12.5.4 Member Data Documentation

12.5.4.1 address

```
int bioscara_hardware_driver::Joint::address  [private]
```

I2C adress.

12.5.4.2 handle

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

I2C bus handle.

12.5.4.3 max

```
float bioscara_hardware_driver::Joint::max = 0  [protected]
```

Joint upper limit.

12.5.4.4 min

```
float bioscara_hardware_driver::Joint::min = 0  [protected]
```

Joint lower limit.

12.5.4.5 offset

```
float bioscara_hardware_driver::Joint::offset = 0  [protected]
```

Joint position offset.

12.5.4.6 reduction

```
float bioscara_hardware_driver::Joint::reduction = 1  [protected]
```

Joint to actuator reduction ratio.

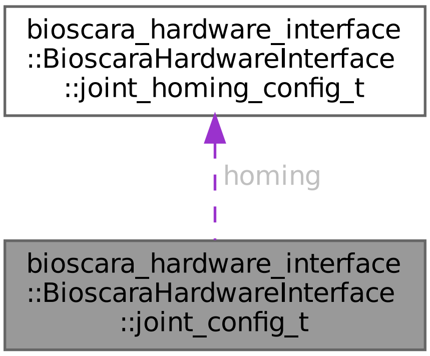
The documentation for this class was generated from the following files:

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mJoint.h
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mJoint.hpp
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mJoint.cpp

12.6 bioscara_hardware_interface::BioscaraHardwareInterface::joint_↩ config_t Struct Reference

configuration structure holding the passed paramters from the ros2_control urdf

Collaboration diagram for bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t:


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

Public Attributes

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

12.6.1 Detailed Description

configuration structure holding the passed paramters from the ros2_control urdf

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

12.6.2 Member Data Documentation

12.6.2.1 drive_current

```
u_int8_t bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::drive_current
```

12.6.2.2 hold_current

```
u_int8_t bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::hold_current
```

12.6.2.3 homing

```
joint_homing_config_t bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t↩
::homing
```

12.6.2.4 i2c_address

```
int bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::i2c_address
```

12.6.2.5 max

```
float bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::max
```

12.6.2.6 max_acceleration

```
float bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::max_acceleration
```

12.6.2.7 max_velocity

```
float bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::max_velocity
```

12.6.2.8 min

```
float bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::min
```

12.6.2.9 reduction

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

12.6.2.10 stall_threshold

```
u_int8_t bioscara_hardware_interface::BioscaraHardwareInterface::joint_config_t::stall_↩
threshold
```

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

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include/bioscara_hardware_↩ interface/bioscara_hardware.hpp

12.7 bioscara_hardware_interface::BioscaraHardwareInterface::joint_↩ homing_config_t Struct Reference

configuration structure holding the passed homing paramters from the ros2_control urdf

Public Attributes

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

12.7.1 Detailed Description

configuration structure holding the passed homing paramters from the ros2_control urdf

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

12.7.2 Member Data Documentation

12.7.2.1 acceleration

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

12.7.2.2 current

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

12.7.2.3 speed

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

12.7.2.4 threshold

```
u_int8_t bioscara_hardware_interface::BioscaraHardwareInterface::joint_homing_config_t::threshold
= 10
```

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

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/include/bioscara_hardware_↩ interface/bioscara_hardware.hpp

12.8 Lowpass Class Reference

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

Public Member Functions

- Lowpass (float gain=1, float sampleTime=0.1, float timeconstant=1.0)

  A simple lowpass filter class.
- float updateState (float u)
- void resetState (void)

Protected Attributes

- float K
- float Ts
- float tau
- float x

12.8.1 Constructor & Destructor Documentation

12.8.1.1 Lowpass()

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

A simple lowpass filter class.

Parameters

gain	
sampleTime	
timeconstant	

12.8.2 Member Function Documentation

12.8.2.1 resetState()

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

12.8.2.2 updateState()

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

12.8.3 Member Data Documentation

12.8.3.1 K

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

12.8.3.2 tau

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

12.8.3.3 Ts

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

12.8.3.4 x

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

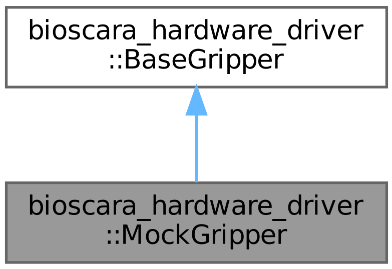
The documentation for this class was generated from the following file:

- Arduino/joint/filters.h

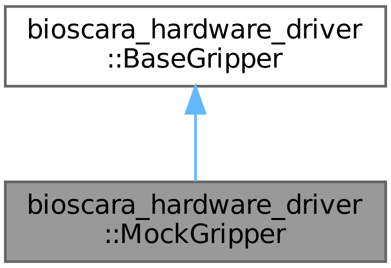
12.9 bioscara_hardware_driver::MockGripper Class Reference

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

Inheritance diagram for bioscara_hardware_driver::MockGripper:


bioscara_hardware_driver ::BaseGripper
bioscara_hardware_driver ::MockGripper

Collaboration diagram for bioscara_hardware_driver::MockGripper:


bioscara_hardware_driver ::BaseGripper
bioscara_hardware_driver ::MockGripper

Public Member Functions

- MockGripper (void)

Public Member Functions inherited from bioscara_hardware_driver::BaseGripper

- BaseGripper (void)
- virtual err_type_t init (void)

  Placeholder, does nothing.
- virtual err_type_t deinit (void)

  Placeholder, does nothing.
- virtual err_type_t enable (void)

  Prepares the servo for use.
- virtual err_type_t disable (void)

  Disables the servo.
- virtual err_type_t setPosition (float width)

  Sets the gripper width in m from the closed position.
- virtual err_type_t setServoPosition (float angle)

  Sets the servo position of the gripper actuator in degrees.
- virtual void setReduction (float reduction)

  Manually set reduction.
- virtual void setOffset (float offset)

  Manually set offset.

12.9.1 Constructor & Destructor Documentation

12.9.1.1 MockGripper()

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

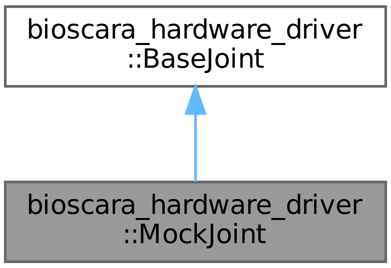
The documentation for this class was generated from the following files:

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mMockGripper.h
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockGripper.cpp

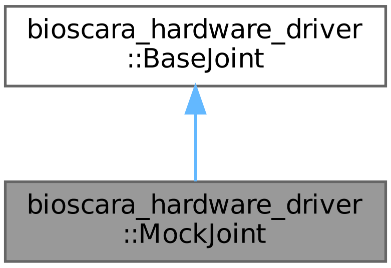
12.10 bioscara_hardware_driver::MockJoint Class Reference

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

Inheritance diagram for bioscara_hardware_driver::MockJoint:


bioscara_hardware_driver ::BaseJoint
bioscara_hardware_driver ::MockJoint

Collaboration diagram for bioscara_hardware_driver::MockJoint:


bioscara_hardware_driver ::BaseJoint
bioscara_hardware_driver ::MockJoint

Public Member Functions

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

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

    disenganges the joint motors, derived classes may override this.
- err_type_t getPosition (float &pos) override

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

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

- err_type_t getVelocity (float &vel) override

  get the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class must override this.
- err_type_t setVelocity (float vel) override

  Set the current joint velocity in radians/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.
- err_type_t checkOrientation (float angle=10.0) override

  Calls the checkOrientation method of the motor. Checks in which direction the motor is turning. Derived class may override this.
- err_type_t stop (void) override

  Stops the motor. Derived class may override this.
- err_type_t getFlags (void) override
- bool isHomed (void) override

  Checks the state if the motor is homed.

Public Member Functions inherited from bioscara_hardware_driver::BaseJoint

- BaseJoint (const std::string name)

  Create a Joint object.
- ∼BaseJoint (void)
- virtual err_type_t init (void)

  Initialization, derived classes may override this.
- virtual err_type_t deinit (void)

  Deinitialization, derived classes may override this.
- virtual err_type_t home (float velocity, u_int8_t sensitivity, u_int8_t current)

  Blocking implementation to home the joint, derived classes may override this.
- virtual err_type_t startHoming (float velocity, u_int8_t sensitivity, u_int8_t current)

  non-blocking implementation to home the joint, derived classes may override this.
- virtual err_type_t postHoming (void)

  perform tasks after a non-blocking homing, derived classes may override this.
- virtual err_type_t moveSteps (int32_t steps)

  Move full steps. Derived class may override this.
- virtual err_type_t disableCL (void)

  Disables the Closed-Loop PID Controller Derived class may override this.
- virtual err_type_t setDriveCurrent (u_int8_t current)

  Set the Drive Current. Derived class may override this.
- virtual err_type_t setHoldCurrent (u_int8_t current)

  Set the Hold Current. Derived class may override this.
- virtual err_type_t setBrakeMode (u_int8_t mode)

  Set Brake Mode. Derived class may override this.
- virtual err_type_t setMaxAcceleration (float maxAccel)

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

  Set the maximum permitted joint velocity in rad/s or m/s for cylindrical and prismatic joints respectively. Derived class may override this.
- virtual err_type_t enableStallguard (u_int8_t sensitivity)

  Enable encoder stall detection of the joint. Derived class may override this.
- virtual bool isEnabled (void)

  Checks the state if the motor is enabled.
- virtual bool isStalled (void)

Checks if the motor is stalled.

- virtual bool isBusy (void)

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

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

  get the currently active blocking command

Protected Member Functions

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

  Call to start the homing sequence of a joint.

Protected Member Functions inherited from bioscara_hardware_driver::BaseJoint

- virtual void wait_while_busy (const float period_ms)

  Blocking loop waiting for BUSY flag to reset.

Private Member Functions

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

Private Attributes

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

Additional Inherited Members

Public Types inherited from bioscara_hardware_driver::BaseJoint

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

  register and command definitions

Public Attributes inherited from bioscara_hardware_driver::BaseJoint

- std::string name

Protected Attributes inherited from bioscara_hardware_driver::BaseJoint

- u_int8_t flags = 0b00001100

  State flags transmitted with every I2C transaction.
- stp_reg_t current_b_cmd = NONE

  Keeps track if a blocking command is being executed.

12.10.1 Constructor & Destructor Documentation

12.10.1.1 MockJoint()

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

12.10.2 Member Function Documentation

12.10.2.1 _home()

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

Call to start the homing sequence of a joint.

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

Parameters

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.

Implements bioscara_hardware_driver::BaseJoint.

12.10.2.2 checkOrientation()

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

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

As the orientation check is blocking on the motor, this 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.

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.10.2.3 disable()

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

disenganges the joint motors, derived classes may override this.

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.10.2.4 enable()

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

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.10.2.5 getDeltaT()

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

12.10.2.6 getFlags()

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

Overload of BaseJoint::getFlags(u_int8_t &flags)

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.10.2.7 getPosition()

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

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

Warning

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

Parameters

pos	

Returns

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

Implements bioscara_hardware_driver::BaseJoint.

12.10.2.8 getVelocity()

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

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

Parameters

vel	

Returns

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

Implements bioscara_hardware_driver::BaseJoint.

12.10.2.9 isHomed()

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

Checks the state if the motor is homed.

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

Returns

true if the motor is homed, false if not.

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.10.2.10 setPosition()

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

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

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.

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.10.2.11 setVelocity()

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

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

Parameters

vel	

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.10.2.12 stop()

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

Stops the motor. Derived class may override this.

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

Returns

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

Reimplemented from bioscara_hardware_driver::BaseJoint.

12.10.3 Member Data Documentation

12.10.3.1 async_start_time

```
std::chrono::_V2::system_clock::time_point bioscara_hardware_driver::MockJoint::async_start_←
time = last_set_position  [private]
```

12.10.3.2 last_set_position

```
std::chrono::_V2::system_clock::time_point bioscara_hardware_driver::MockJoint::last_set_←
position = std::chrono::high_resolution_clock::now()  [private]
```

12.10.3.3 last_set_velocity

```
std::chrono::_V2::system_clock::time_point bioscara_hardware_driver::MockJoint::last_set_←
velocity = last_set_position  [private]
```

12.10.3.4 op_mode

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

12.10.3.5 q

```
float bioscara_hardware_driver::MockJoint::q = 0.0  [private]
```

12.10.3.6 qd

```
float bioscara_hardware_driver::MockJoint::qd = 0.0  [private]
```

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

- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mMockJoint.h
- ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockJoint.cpp

12.11 MovMax Class Reference

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

Public Member Functions

- MovMax (float windowSize)
- float updateState (float u)

Protected Attributes

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

12.11.1 Constructor & Destructor Documentation

12.11.1.1 MovMax()

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

12.11.2 Member Function Documentation

12.11.2.1 updateState()

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

12.11.3 Member Data Documentation

12.11.3.1 cb_data

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

12.11.3.2 cb_index

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

12.11.3.3 M

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

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

- Arduino/joint/filters.h

12.12 RPI_PWM Class Reference

PWM class for the Raspberry PI 4 and 5.

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

Public Member Functions

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

Private Member Functions

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

Private Attributes

- int per = 0
- std::string chippath
- std::string pwmpath

12.12.1 Detailed Description

PWM class for the Raspberry PI 4 and 5.

12.12.2 Constructor & Destructor Documentation

12.12.2.1 ∼RPI_PWM()

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

12.12.3 Member Function Documentation

12.12.3.1 disable()

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

12.12.3.2 enable()

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

12.12.3.3 setDutyCycle()

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

Sets the duty cycle in percent 0 - 100.

Parameters

v	The duty cycle in percent.

Returns

>0 on success and -1 after an error.

12.12.3.4 setDutyCycleNS()

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

12.12.3.5 setPeriod()

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

12.12.3.6 start()

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

Starts the PWM

Parameters

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)

Returns

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

12.12.3.7 stop()

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

Stops the PWM

12.12.3.8 writeSYS()

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

12.12.4 Member Data Documentation

12.12.4.1 chippath

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

12.12.4.2 per

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

12.12.4.3 pwmpath

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

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

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

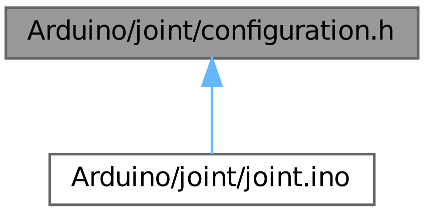
Chapter 13

File Documentation

13.1 Arduino/joint/configuration.h File Reference

Configuration definitions for Joint 1 to Joint 4.

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



Macros

- #define ADR 0x11

  I2C adress of joint n is 0x1n.
- #define MAXACCEL 10000

  Maximum acceleration in steps/s^2. Can be set for each joint depending on inertia. If set to high stalls might trigger since PID error grows too large.
- #define MAXVEL 800

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

13.1.1 Detailed Description

Configuration definitions for Joint 1 to Joint 4.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

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

13.1.2 Macro Definition Documentation

13.1.2.1 ADR

```
#define ADR 0x11
```

I2C adress of joint n is 0x1n.

13.1.2.2 MAXACCEL

```
#define MAXACCEL 10000
```

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

13.1.2.3 MAXVEL

```
#define MAXVEL 800
```

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

13.2 configuration.h

Go to the documentation of this file.

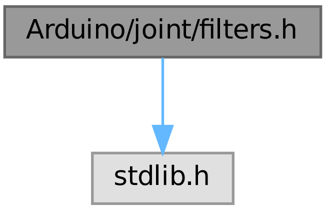
```

#ifndef CONFIGURATION_H
#define CONFIGURATION_H

#if defined(J1)
#define ADR 0x11
#define MAXACCEL 0
#define MAXVEL 0
#define STALL_WINDOW_B1 12
#define STALL_WINDOW_B2 450
#define STALL_WINDOW_OFFSET 90.0
#define STALL_SLOPE 0.0

#elif defined(J2)
#define ADR 0x12
#define MAXACCEL 0
#define MAXVEL 0
#define STALL_WINDOW_B1 12
#define STALL_WINDOW_B2 450
#define STALL_WINDOW_OFFSET 90.0
#define STALL_SLOPE 0.0

#elif defined(J3)
#define ADR 0x13
#define MAXACCEL 0
#define MAXVEL 0
#define STALL_WINDOW_B1 12
#define STALL_WINDOW_B2 450
#define STALL_WINDOW_OFFSET 90.0
#define STALL_SLOPE 0.0

#elif defined(J4)
#define ADR 0x14
#define MAXACCEL 0
#define MAXVEL 0
#define STALL_WINDOW_B1 12
#define STALL_WINDOW_B2 450
#define STALL_WINDOW_OFFSET 90.0
#define STALL_SLOPE 0.0
#else

/* Below only defined for documentation */
#define ADR 0x11

#define MAXACCEL 10000

#define MAXVEL 800
#error "No Joint has been defined. Define one of 'JX' where X 1,2,3,4"
#endif

#endif
```

13.3 Arduino/joint/filters.h File Reference

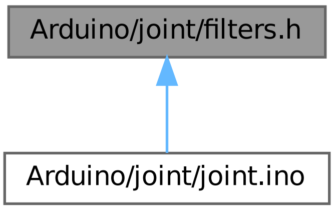
Helper classes for FIR and IIR filters.

```
#include <stdlib.h>
```

Include dependency graph for filters.h:



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



Classes

- class Lowpass
- class MovMax

13.3.1 Detailed Description

Helper classes for FIR and IIR filters.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

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

13.4 filters.h

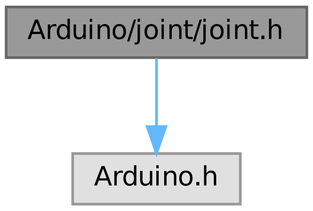
Go to the documentation of this file.

```
#include <stdlib.h>

class Lowpass {

protected:
  float K, Ts, tau, x;

public:

  Lowpass(float gain = 1, float sampleTime = 0.1, float timeconstant = 1.0) {
    this->K = gain;
    this->Ts = sampleTime;
    this->tau = timeconstant;
    x = 0.0;
  }

  float updateState(float u) {
    x = (1 - (Ts / tau)) * x + K * (Ts / tau) * u;
    return x;
  }

  void resetState(void){
    x = 0.0;
  }
};

class MovMax {

protected:
  unsigned int M = 200;  // Window Size

  float *cb_data;
  unsigned int cb_index;


public:

  MovMax(float windowSize)
    : M(windowSize), cb_index(0), cb_data(0) {

    cb_data = (float *)malloc(windowSize * sizeof(float));  // allocate memory for buffer
  }

  float updateState(float u) {

    cb_data[cb_index] = u;
    cb_index = (cb_index + 1) % M;


    float max = 0;
    for (size_t i = 0; i < M; i++) {
      if (cb_data[i] > max) {
        max = cb_data[i];
      }
    }

    return max;
  }
};
```

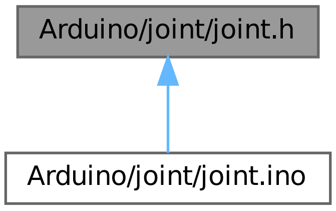
13.5 Arduino/joint/joint.h File Reference

joint firmware header

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



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



Macros

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

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

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

  Macro to dump a buffer to the serial console.

Enumerations

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

DISABLESTALLGUARD = 0x1F , CLEARSTALL = 0x20 , SETBRAKEMODE = 0x22 , ENABLEPID = 0x23 , DISABLEPID = 0x24 , 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.

13.5.1 Detailed Description

joint firmware header

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

This file contains definitions and macros for the joint firmware.

13.5.2 Macro Definition Documentation

13.5.2.1 ACK

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

13.5.2.2 DUMP_BUFFER

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

Value:

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

Macro to dump a buffer to the serial console.

Parameters

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

13.5.2.3 MAX_BUFFER

```
#define MAX_BUFFER 4
```

Maximum size of I2C Payload in bytes.

4 bytes used to transmit floats and int32_t

13.5.2.4 NACK

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

13.5.2.5 RFLAGS_SIZE

```
#define RFLAGS_SIZE 1
```

Size of the return flags in bytes.

Only one byte used and hence set to 1.

13.5.3 Enumeration Type Documentation

13.5.3.1 stp_reg_t

```
enum stp_reg_t
```

register and command definitions

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

Enumerator

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

Enumerator

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

13.5.4 Function Documentation

13.5.4.1 readValue()

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

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

Parameters

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

13.5.4.2 writeValue()

```
template<typename T >
int writeValue (
            const T val,
            uint8_t * txBuf,
            size_t & tx_length )
```

Writes a value of the specified type to a buffer.

Parameters

val	Reference to input variable
txBuf	pointer to tx buffer
tx_length	Length of the buffer returne

Returns

0 On success

13.6 joint.h

Go to the documentation of this file.

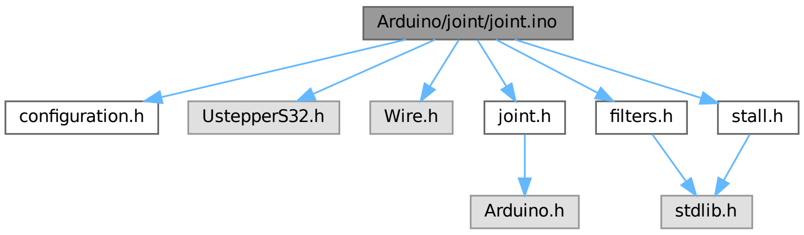
```

#ifndef JOINT_H
#define JOINT_H
#include <Arduino.h>

#define ACK 'O'
#define NACK 'N'

#define MAX_BUFFER 4 // Bytes

#define RFLAGS_SIZE 1

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

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,
};

template <typename T>
void readValue(T &val, uint8_t *rxBuf, size_t rx_length)
{
  memcpy(&val, rxBuf, rx_length);
}

template <typename T>
int writeValue(const T val, uint8_t *txBuf, size_t &tx_length)
{
  tx_length = sizeof(T);
  memcpy(txBuf, &val, tx_length);
  return 0;
}

#endif
```

13.7 Arduino/joint/joint.ino File Reference

joint firmware

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

Include dependency graph for joint.ino:



Macros

- #define J3
  Define either joint that is to be flashed.

Functions

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

- void requestEvent ()

  I2C request event Handler.
- void setup (void)

  Setup Peripherals.
- void loop (void)

  Main loop.

Variables

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

13.7.1 Detailed Description

joint firmware

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

This file contains the joint firmware.

13.7.2 Macro Definition Documentation

13.7.2.1 J3

```
#define J3
```

Define either joint that is to be flashed.

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

13.7.3 Function Documentation

13.7.3.1 blocking_handler()

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

Handles commands received via I2C.

Warning

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

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

Parameters

reg	command that should be executed.

13.7.3.2 loop()

```
void loop (
            void )
```

Main loop.

Executes the following:

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

13.7.3.3 non_blocking_handler()

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

Handles read request received via I2C.

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

Parameters

reg	command to execute/register to read.

13.7.3.4 receiveEvent()

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

I2C receive event Handler.

Reads the content of the received message. Saves the register so it can be used in the main loop. If the master invokes the read() function the message contains only the register byte and no payload. If the master invokes the write() the message has a payload of appropriate size for the command. Every I2C transaction starts with a receive event when the command is sent and is 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

13.7.3.5 requestEvent()

```
void requestEvent ( )
```

I2C request event Handler.

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

13.7.3.6 setup()

```
void setup (
            void )
```

Setup Peripherals.

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

13.7.4 Variable Documentation

13.7.4.1 reg

```
uint8_t reg = 0
```

13.7.4.2 rx_buf

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

13.7.4.3 rx_data_ready

```
bool rx_data_ready = 0
```

13.7.4.4 rx_length

```
size_t rx_length = 0
```

13.7.4.5 stepper

```
UstepperS32 stepper
```

13.7.4.6 tx_buf

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

13.7.4.7 tx_length

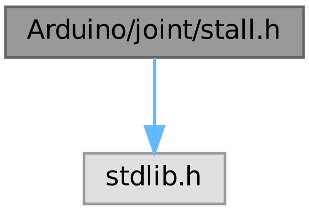
```
size_t tx_length = 0
```

13.8 Arduino/joint/stall.h File Reference

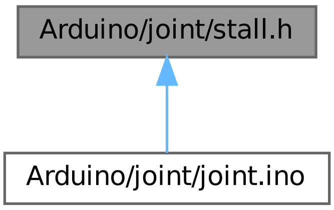
Helper functions for improved stall detection.

```
#include <stdlib.h>
```

Include dependency graph for stall.h:



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



Functions

- float stall_threshold (float qd_rad, float offset)

    computes the speed adaptive threshold.

13.8.1 Detailed Description

Helper functions for improved stall detection.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

13.8.2 Function Documentation

13.8.2.1 stall_threshold()

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

computes the speed adaptive threshold.

Parameters

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

13.9 stall.h

Go to the documentation of this file.

```

#include <stdlib.h>

float stall_threshold(float qd_rad, float offset){
  /* y = ax + b */
  float a = STALL_SLOPE;
  float b = offset;
  if((abs(qd_rad) >= STALL_WINDOW_B1) && (abs(qd_rad) <= STALL_WINDOW_B2)){
    b += STALL_WINDOW_OFFSET;
  }
  return a*qd_rad+b;
}
```

13.10 docs/DOCS_README.md File Reference

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

Namespaces

- namespace bioscara

Functions

- bioscara.generate_launch_description ()

13.12 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_↩ bringup/launch/test_joint_trajectory_controller.launch.py File Reference

Namespaces

- namespace test_joint_trajectory_controller

Functions

- test_joint_trajectory_controller.generate_launch_description ()

13.13 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_bringup/↩ README.md File Reference

13.14 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_controllers/↩ README.md File Reference

13.15 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_description/↩ README.md File Reference

13.16 ROS2/ros2_scara_ws/src/dalsa_bioscara/README.md File Reference

13.17 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_↩ description/bioscara_description/__init__.py File Reference

13.18 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_↩ description/launch/display.launch.py File Reference

Namespaces

- namespace display

Functions

- display.generate_launch_description ()

13.19 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_↩ description/launch/gazebo.launch.py File Reference

Namespaces

- namespace gazebo

Functions

- gazebo.generate_launch_description ()

13.20 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_↩ description/setup.py File Reference

Namespaces

- namespace setup

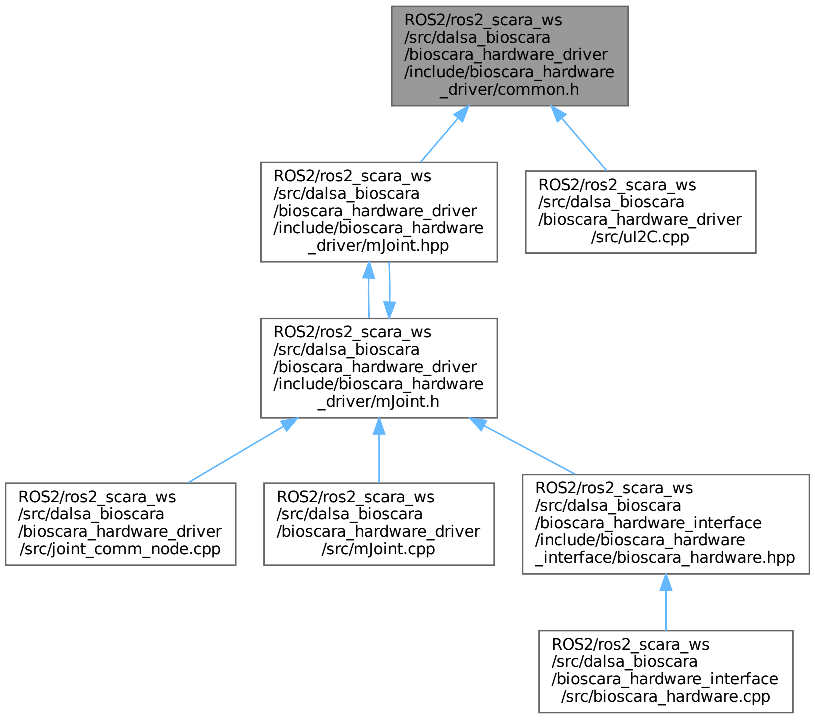
Variables

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

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

A file containing utility macros and functions.

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


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/common.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/uI2C.cpp
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 /src/joint_comm_node.cpp
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/mJoint.cpp
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

Macros

- #define DUMP_BUFFER(buffer, size)

    Macro to dump a buffer to cout.

13.21.1 Detailed Description

A file containing utility macros and functions.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

13.21.2 Macro Definition Documentation

13.21.2.1 DUMP_BUFFER

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

Value:

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

Macro to dump a buffer to cout.

Parameters

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

13.22 common.h

Go to the documentation of this file.

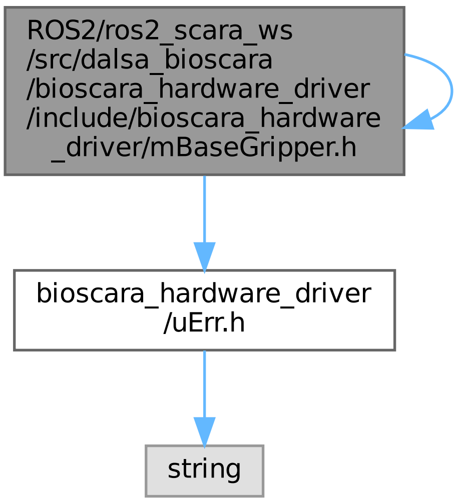
```
#ifndef COMMON_H
#define COMMON_H

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

#endif // COMMON_H
```

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

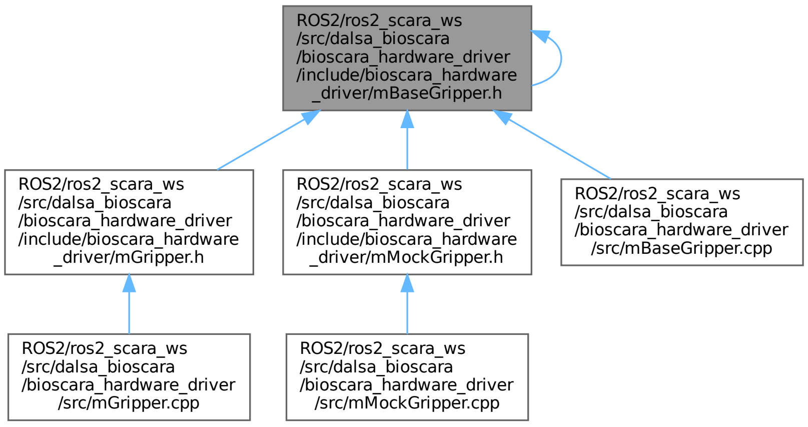
File containing the BaseGripper class.

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

Include dependency graph for mBaseGripper.h:



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



Classes

- class bioscara_hardware_driver::BaseGripper

Namespaces

- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

13.23.1 Detailed Description

File containing the BaseGripper class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

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

13.24 mBaseGripper.h

Go to the documentation of this file.

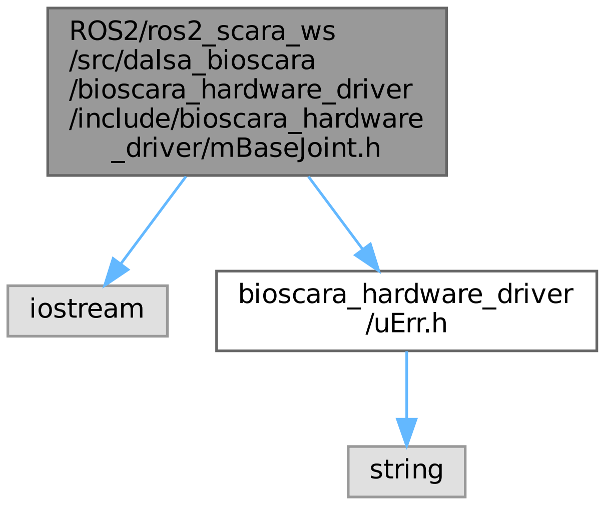
```
#ifndef MBASEGRIPPER_H
#define MBASEGRIPPER_H
#include "bioscara_hardware_driver/mBaseGripper.h"
#include "bioscara_hardware_driver/uErr.h"

namespace bioscara_hardware_driver
{
    class BaseGripper
    {
    public:
        BaseGripper(void);

        virtual err_type_t init(void);

        virtual err_type_t deinit(void);

        virtual err_type_t enable(void);

        virtual err_type_t disable(void);

        virtual err_type_t setPosition(float width);

        virtual err_type_t setServoPosition(float angle);

        virtual void setReduction(float reduction);

        virtual void setOffset(float offset);

    protected:
    private:
    };
}
#endif // MBASEGRIPPER_H
```

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

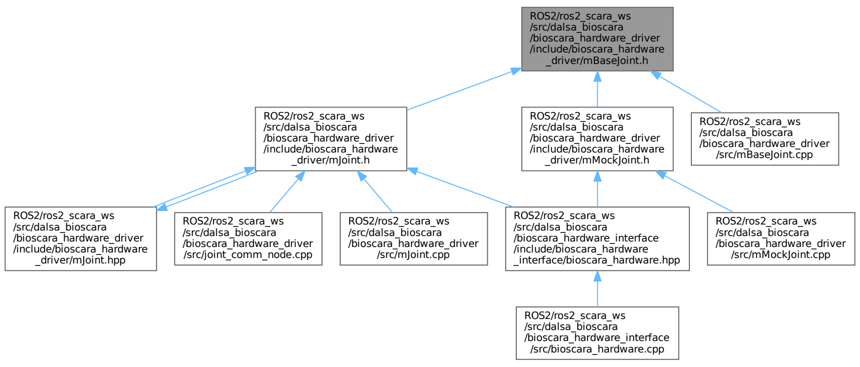
File including the BaseJoint class.

```
#include <iostream>
#include "bioscara_hardware_driver/uErr.h"
```

Include dependency graph for mBaseJoint.h:


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/mBaseJoint.h
iostream
bioscara_hardware_driver /uErr.h
string

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


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/mBaseJoint.h
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/mJoint.h
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/mMockJoint.h
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/mBaseJoint.cpp
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/joint_comm_node.cpp
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/mJoint.cpp
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_driver /src/mMockJoint.cpp
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_interface /src/bioscara_hardware.cpp

Classes

- class bioscara_hardware_driver::BaseJoint

  TODO.

Namespaces

- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

13.25.1 Detailed Description

File including the BaseJoint class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-29

Copyright

Copyright (c) 2025

13.26 mBaseJoint.h

Go to the documentation of this file.

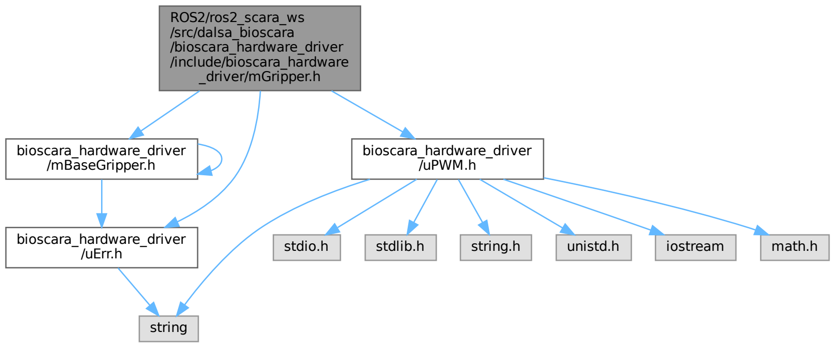
```
#ifndef MBASEJOINT_H
#define MBASEJOINT_H

#include <iostream>
#include "bioscara_hardware_driver/uErr.h"

namespace bioscara_hardware_driver
{
  class BaseJoint
  {
  public:
    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,
    };

    BaseJoint(const std::string name);
    ~BaseJoint(void);

    virtual err_type_t init(void);

    virtual err_type_t deinit(void);

    virtual err_type_t enable(u_int8_t driveCurrent, u_int8_t holdCurrent);

    virtual err_type_t disable(void);

    virtual err_type_t home(float velocity, u_int8_t sensitivity, u_int8_t current);

    virtual err_type_t startHoming(float velocity, u_int8_t sensitivity, u_int8_t current);

    virtual err_type_t postHoming(void);

    virtual err_type_t getPosition(float &pos) = 0;

    virtual err_type_t setPosition(float pos);

    virtual err_type_t moveSteps(int32_t steps);

    virtual err_type_t getVelocity(float &vel) = 0;

    virtual err_type_t setVelocity(float vel);

    virtual err_type_t checkOrientation(float angle = 10.0);

    virtual err_type_t stop(void);

    virtual err_type_t disableCL(void);

    virtual err_type_t setDriveCurrent(u_int8_t current);

    virtual err_type_t setHoldCurrent(u_int8_t current);

    virtual err_type_t setBrakeMode(u_int8_t mode);

    virtual err_type_t setMaxAcceleration(float maxAccel);

    virtual err_type_t setMaxVelocity(float maxVel);

    virtual err_type_t enableStallguard(u_int8_t sensitivity);

    virtual bool isHomed(void);

    virtual bool isEnabled(void);

    virtual bool isStalled(void);

    virtual bool isBusy(void);

    virtual err_type_t getFlags(u_int8_t &flags);

    virtual err_type_t getFlags(void);

    virtual stp_reg_t getCurrentBCmd(void);

    std::string name;

  protected:
    virtual void wait_while_busy(const float period_ms);

    virtual err_type_t _home(float velocity, u_int8_t sensitivity, u_int8_t current) = 0;

    u_int8_t flags = 0b00001100;

    stp_reg_t current_b_cmd = NONE;

  private:
  };
}
#endif
```

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

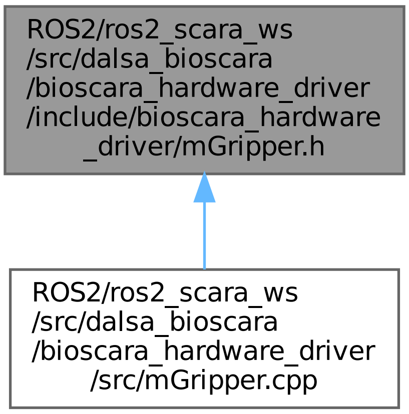
File containing the Gripper class.

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

Include dependency graph for mGripper.h:



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



Classes

- class bioscara_hardware_driver::Gripper

Namespaces

- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

13.27.1 Detailed Description

File containing the Gripper class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

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

13.28 mGripper.h

Go to the documentation of this file.

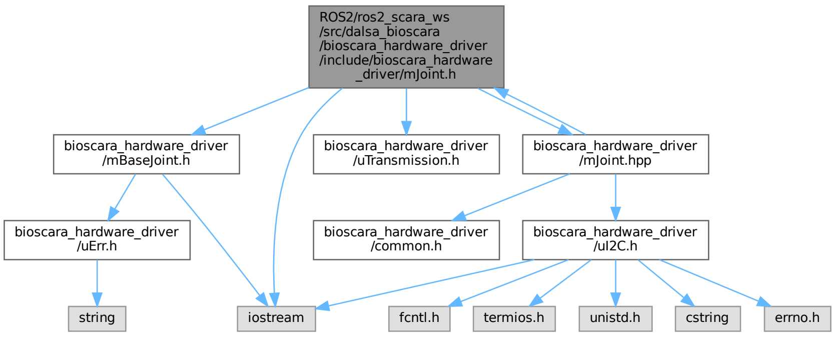
```cpp

#ifndef MGRIPPER_H
#define MGRIPPER_H
#include "bioscara_hardware_driver/mBaseGripper.h"
#include "bioscara_hardware_driver/uPWM.h"
#include "bioscara_hardware_driver/uErr.h"

namespace bioscara_hardware_driver
{
    class Gripper : public BaseGripper
    {
    public:
        Gripper(float reduction, float offset, float min, float max);

        err_type_t enable(void) override;

        err_type_t disable(void) override;

        err_type_t setPosition(float width) override;

        err_type_t setServoPosition(float angle) override;

        void setReduction(float reduction);

        void setOffset(float offset);

    protected:
        float reduction = 1;
        float offset = 0;
        float min = 0;
        float max = 0;
    private:
        RPI_PWM pwm;
        int freq = 50;
    };
}
#endif // MGRIPPER_H
```

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

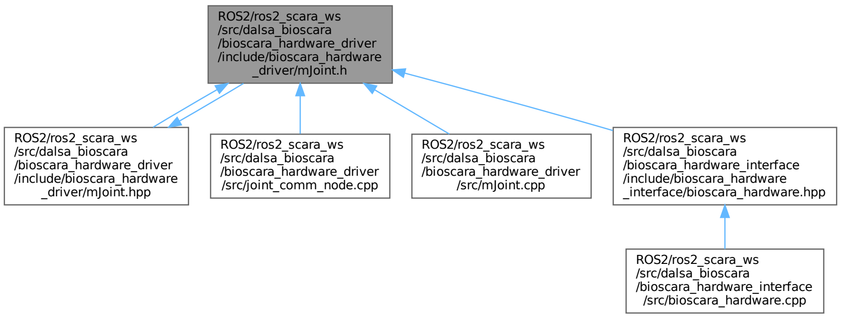
File including the Joint class.

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

Include dependency graph for mJoint.h:



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



Classes

- class bioscara_hardware_driver::Joint

    Representing a single joint on the I2C bus.

Namespaces

- namespace bioscara_hardware_driver

    Generic BaseGripper object to interact with the robot gripper.

13.29.1 Detailed Description

File including the Joint class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-29

Copyright

Copyright (c) 2025

13.30 mJoint.h

Go to the documentation of this file.

```

#ifndef MJOINT_H
#define MJOINT_H

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

namespace bioscara_hardware_driver
{
  class Joint : public BaseJoint
  {
  public:
    Joint(const std::string name, const int address, const float reduction, const float min, const
float max);
    ~Joint(void);

    err_type_t init(void) override;

    err_type_t deinit(void) override;

    err_type_t enable(u_int8_t driveCurrent, u_int8_t holdCurrent) override;

    // /**
    //  * @brief disenganges the joint motor without closing i2c handle
    //  * @return 0 on success,
    //  * -1 on communication error,
    //   -5 if the joint is not initialized.
    //  */
    // int disable(void);

    // /**
    //  * @brief 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:
    //  * @return -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()).
    //  */
    // int home(float velocity, u_int8_t sensitivity, u_int8_t current);

    // /**
    //  * @brief 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:
    //  * @return -109 if the joint is already currently homing (for example from a call to
Joint::startHoming()).
    //  */
    // int startHoming(float velocity, u_int8_t sensitivity, u_int8_t current);

    err_type_t postHoming(void) override;

    err_type_t getPosition(float &pos) override;

    err_type_t setPosition(float pos) override;

    err_type_t moveSteps(int32_t steps) override;

    err_type_t getVelocity(float &vel) override;

    err_type_t setVelocity(float vel) override;

    err_type_t checkOrientation(float angle = 10.0) override;

    err_type_t stop(void) override;
    err_type_t disableCL(void) override;

    err_type_t setDriveCurrent(u_int8_t current) override;

    err_type_t setHoldCurrent(u_int8_t current) override;

    err_type_t setBrakeMode(u_int8_t mode) override;

    err_type_t setMaxAcceleration(float maxAccel) override;

    err_type_t setMaxVelocity(float maxVel) override;

    err_type_t enableStallguard(u_int8_t sensitivity) override;

    // /**
    //  * @brief 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.
    //  *
    //  * @return true if the motor is homed,
    //  * false if not.
    //  */
    // bool isHomed(void);

    // /**
    //  * @brief 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().
    //  *
    //  * @return true if the motor is enabled,
    //  * false if not.
    //  */
    // bool isEnabled(void);

    // /**
    //  * @brief 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.
    //  * @return true if the motor is stalled,
    //  * false if not.
    //  */
    // bool isStalled(void);

    // /**
    //  * @brief 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.
    //  * @return true if a blocking command is currently executing,
    //  * false if not.
    //  */
    // bool isBusy(void);

    err_type_t getFlags(void) override;

    err_type_t getHomingOffset(float &offset);

    err_type_t setHomingOffset(const float offset);

    // /**
    //  * @brief get the currently active blocking command
    //  *
    //  * @return The the command of type stp_reg_t
    //  */
    // stp_reg_t getCurrentBCmd(void);

  protected:
    // /**
    //  * @brief Blocking loop waiting for BUSY flag to reset.
    //  *
    //  * @param period_ms time in ms between polls.
    //  */
    // void wait_while_busy(const float period_ms);

    err_type_t _home(float velocity, u_int8_t sensitivity, u_int8_t current);

    err_type_t checkCom(void);

    // /**
    //  * @brief 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: \n
    //  *
    //  * |BIT7|BIT6|BIT5|BIT4|BIT3|BIT2|BIT1|BIT0|
    //  * | ---- | ---- | ---- | ---- | ---- | ---- | ---- | ---- |
    //  * |reserved|reserved|reserved|reserved|NOTENABLED|NOTHOMED|BUSY|STALL|
    //  *
    //  * \b 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. \n
    //  * \b BUSY is set if the slave is busy processing a previous command. \n
    //  * \b NOTHOMED is cleared if the joint is homed. Movement is only allowed if this flag is clear
\n
    //  * \b NOTENABLED is cleared if the joint is enabled after calling Joint::enable()
    //  */
    // u_int8_t flags = 0x00;

    float reduction = 1;
    float offset = 0;
    float min = 0;
    float max = 0;

    // stp_reg_t current_b_cmd = NONE; ///< Keeps track if a blocking command is being executed

  private:
    template <typename T>
    int read(const stp_reg_t reg, T &data, u_int8_t &flags);

    template <typename T>
    int write(const stp_reg_t reg, T data, u_int8_t &flags);

    int address;
    int handle = -1;
  };
}
#include "bioscara_hardware_driver/mJoint.hpp"

#endif
```

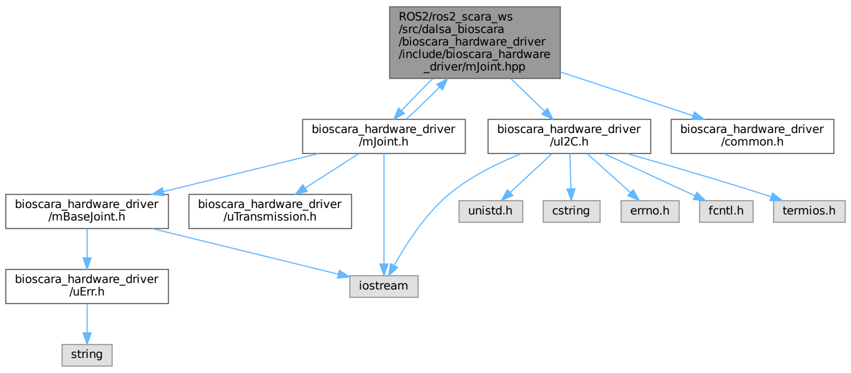
13.31 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/include/bioscara_hardware_driver/mJoint.hpp File Reference

Templated functions for the Joint class.

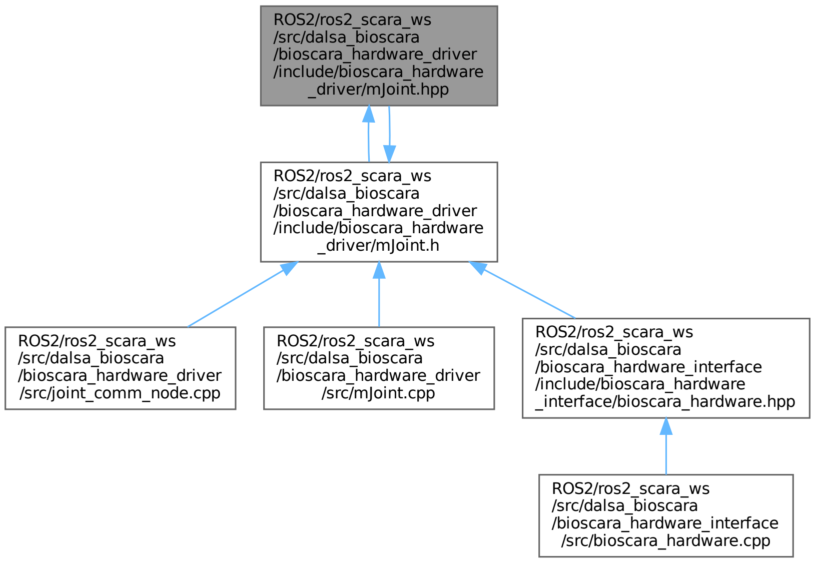
```
#include "bioscara_hardware_driver/mJoint.h"
#include "bioscara_hardware_driver/uI2C.h"
```

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

Include dependency graph for mJoint.hpp:


ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mJoint.hpp
bioscara_hardware_driver/mJoint.h
bioscara_hardware_driver/uI2C.h
bioscara_hardware_driver/common.h
bioscara_hardware_driver/mBaseJoint.h
bioscara_hardware_driver/uTransmission.h
unistd.h
cstring
errno.h
fcntl.h
termios.h
bioscara_hardware_driver/uErr.h
iostream
string

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


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/include/bioscara_hardware_driver/mJoint.h
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/joint_comm_node.cpp
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mJoint.cpp
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

Namespaces

- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

13.31.1 Detailed Description

Templated functions for the Joint class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-29

Copyright

Copyright (c) 2025

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

13.32 mJoint.hpp

Go to the documentation of this file.

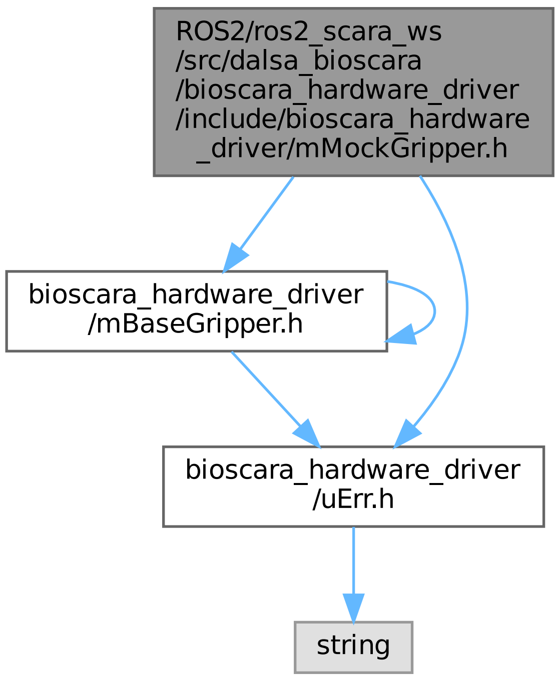
```

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

namespace bioscara_hardware_driver
{
    template <typename T>
    int Joint::read(const stp_reg_t reg, T &data, u_int8_t &flags)
    {
        size_t size = sizeof(T) + RFLAGS_SIZE;
        char buf[MAX_BUFFER + RFLAGS_SIZE];
        int n = readFromI2CDev(this->handle, reg, buf, size);
        if (n != static_cast<int>(size))
        {
            return -1;
        }
        memcpy(&data, buf, size - RFLAGS_SIZE);
        memcpy(&flags, buf + size - RFLAGS_SIZE, RFLAGS_SIZE);
        return 0;
    }

    template <typename T>
    int Joint::write(const stp_reg_t reg, T data, u_int8_t &flags)
    {
        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);
        rc = rc > 0 ? 0 : rc;
        memcpy(&flags, buf + size - RFLAGS_SIZE, RFLAGS_SIZE);
        return rc;
    }
}
```

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

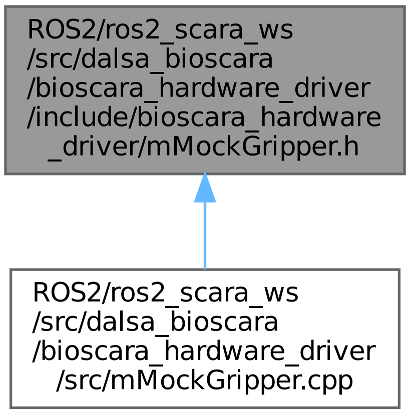
File containing the MockGripper class.

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

Include dependency graph for mMockGripper.h:



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



Classes

- class bioscara_hardware_driver::MockGripper

Namespaces

- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

13.33.1 Detailed Description

File containing the MockGripper class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Copyright (c) 2025

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

13.34 mMockGripper.h

Go to the documentation of this file.

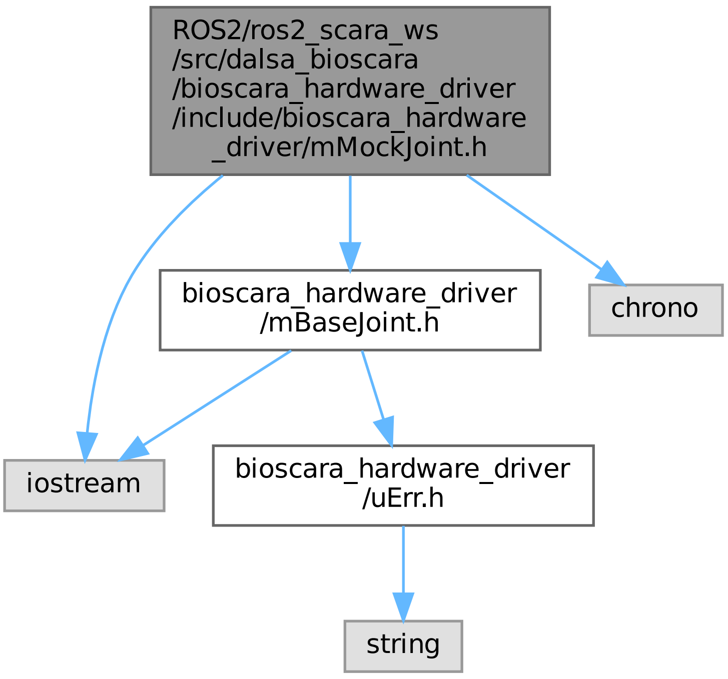
```
#ifndef MMOCKGRIPPER_H
#define MMOCKGRIPPER_H
#include "bioscara_hardware_driver/mBaseGripper.h"
#include "bioscara_hardware_driver/uErr.h"
namespace bioscara_hardware_driver
{
    class MockGripper : public BaseGripper
    {
    public:
        MockGripper(void);

    protected:
    private:
    };
}
#endif // MMOCKGRIPPER_H
```

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

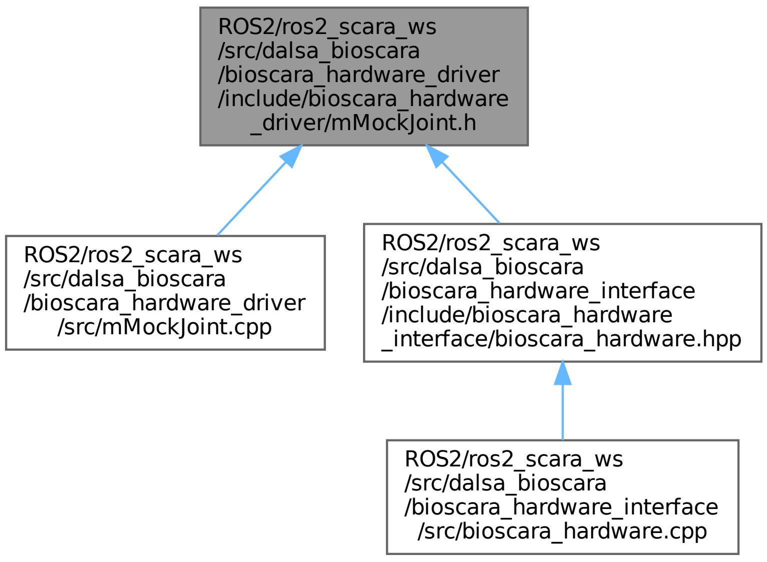
File including the MockJoint class.

```
#include <iostream>
#include "bioscara_hardware_driver/mBaseJoint.h"
#include <chrono>
```

Include dependency graph for mMockJoint.h:



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


ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/mMockJoint.h
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockJoint.cpp
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

Classes

- class bioscara_hardware_driver::MockJoint

Namespaces

- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

13.35.1 Detailed Description

File including the MockJoint class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-29

Copyright

Copyright (c) 2025

13.36 mMockJoint.h

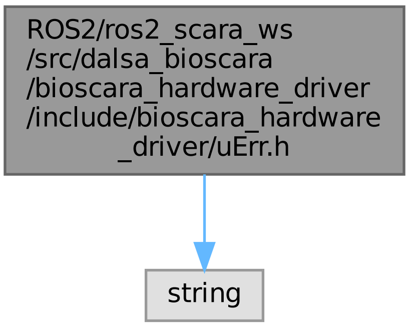
Go to the documentation of this file.

```
#ifndef MMOCKJOINT_H
#define MMOCKJOINT_H

#include <iostream>
#include "bioscara_hardware_driver/mBaseJoint.h"
#include <chrono>

namespace bioscara_hardware_driver
{
  class MockJoint : public BaseJoint
  {
  public:
    MockJoint(const std::string name);

    err_type_t enable(u_int8_t driveCurrent, u_int8_t holdCurrent) override;

    err_type_t disable(void) override;

    err_type_t getPosition(float &pos) override;

    err_type_t setPosition(float pos) override;

    err_type_t getVelocity(float &vel) override;

    err_type_t setVelocity(float vel) override;

    err_type_t checkOrientation(float angle = 10.0) override;

    err_type_t stop(void) override;

    err_type_t getFlags(void) override;

    bool isHomed(void) override;

  protected:
    err_type_t _home(float velocity, u_int8_t sensitivity, u_int8_t current);

  private:
    float q = 0.0;
    float qd = 0.0;

    std::chrono::_V2::system_clock::time_point last_set_position =
std::chrono::high_resolution_clock::now();
    std::chrono::_V2::system_clock::time_point last_set_velocity = last_set_position;
    std::chrono::_V2::system_clock::time_point async_start_time = last_set_position;
    float getDeltaT(std::chrono::_V2::system_clock::time_point &last_call, bool update = true);

    stp_reg_t op_mode = NONE;
  };
}
#endif
```

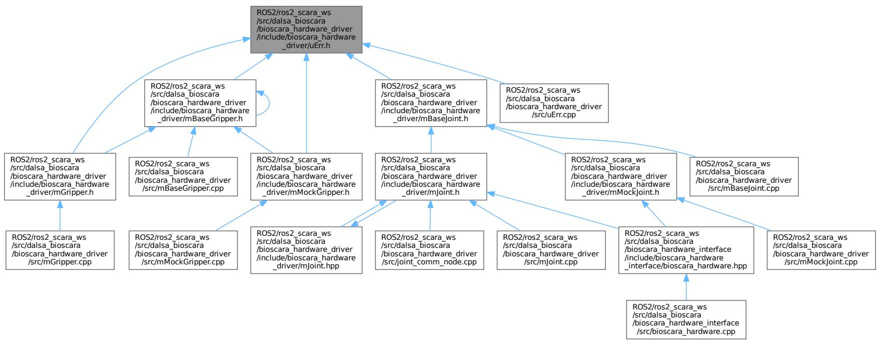
13.37 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/include/bioscara_hardware_driver/uErr.h File Reference

`#include <string>`

Include dependency graph for uErr.h:


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/uErr.h
string

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


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/uErr.h
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/mBaseGripper.h
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/mBaseJoint.h
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/uErr.cpp
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/mBaseGripper.cpp
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/mMockGripper.h
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/mJoint.h
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/mMockJoint.h
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/mBaseJoint.cpp
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/mGripper.cpp
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/mMockGripper.cpp
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/joint_comm_node.cpp
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/mJoint.cpp
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_driver /src/mMockJoint.cpp
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_interface /src/bioscara_hardware.cpp

Namespaces

- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

Macros

- #define RETURN_ON_ERROR(x)

  Macro which executes a function and returns from the calling function with the error code if the called function fails.
- #define RETURN_ON_FALSE(a, err_code)

  Macro which returns the calling function with specified error_code if the given condition is false.
- #define RETURN_ON_NEGATIVE(a, err_code)

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

Enumerations

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

  Enum defining common error types.

Functions

- std::string bioscara_hardware_driver::error_to_string (err_type_t err)

  Converts an error code to a string and returns it.

13.37.1 Macro Definition Documentation

13.37.1.1 RETURN_ON_ERROR

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

Value:
```
    do                                                              \
    {                                                               \
        bioscara_hardware_driver::err_type_t err_rc_ = (x);         \
        if (err_rc_ != bioscara_hardware_driver::err_type_t::OK)   \
        {                                                           \
            return err_rc_;                                         \
        }                                                           \
    } while (0);
```

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

Adapted from the `ESP-IDF`

Parameters

x	function to call

13.37.1.2 RETURN_ON_FALSE

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

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

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

Adapted from the `ESP-IDF`

Parameters

a	expression that evaluates to true or false
err_code	return code to return on false

13.37.1.3 RETURN_ON_NEGATIVE

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

Value:

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

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

Adapted from the ESP-IDF

Parameters

a	expression that evaluates to a signed number
err_code	return code to return on false

13.38 uErr.h

Go to the documentation of this file.

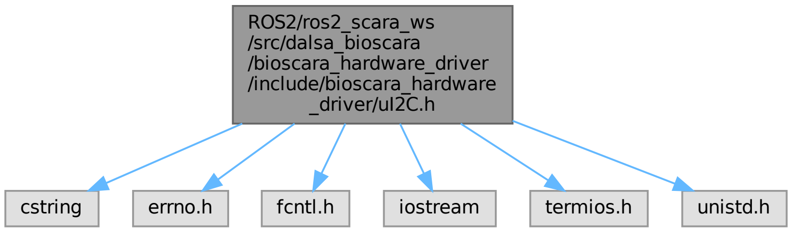
```
#ifndef UERR_H
#define UERR_H

#include <string>

namespace bioscara_hardware_driver
{
    enum class err_type_t
    {
        OK = 0,
        ERROR = -1,
        NOT_HOMED = -2,
        NOT_ENABLED = -3,
        STALLED = -4,
        NOT_INIT = -5,
        COMM_ERROR = -6,
        INVALID_ARGUMENT = -101,
        INCORRECT_STATE = -109,

    };

    std::string error_to_string(err_type_t err);
}

#define RETURN_ON_ERROR(x)                                          \
    do                                                              \
    {                                                               \
        bioscara_hardware_driver::err_type_t err_rc_ = (x);         \
        if (err_rc_ != bioscara_hardware_driver::err_type_t::OK)    \
        {                                                           \
            return err_rc_;                               \
        }                                                 \
    } while (0);

#define RETURN_ON_FALSE(a, err_code) \
    do                               \
    {                                \
        if (!(a))                    \
        {                            \
            return err_code;         \
        }                            \
    } while (0);

#define RETURN_ON_NEGATIVE(a, err_code) \
    do                                  \
    {                                   \
        if ((a) < 0)                    \
        {                               \
            return err_code;            \
        }                               \
    } while (0);

#endif // UERR_H
```

13.39 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/include/bioscara_hardware_driver/uI2C.h File Reference

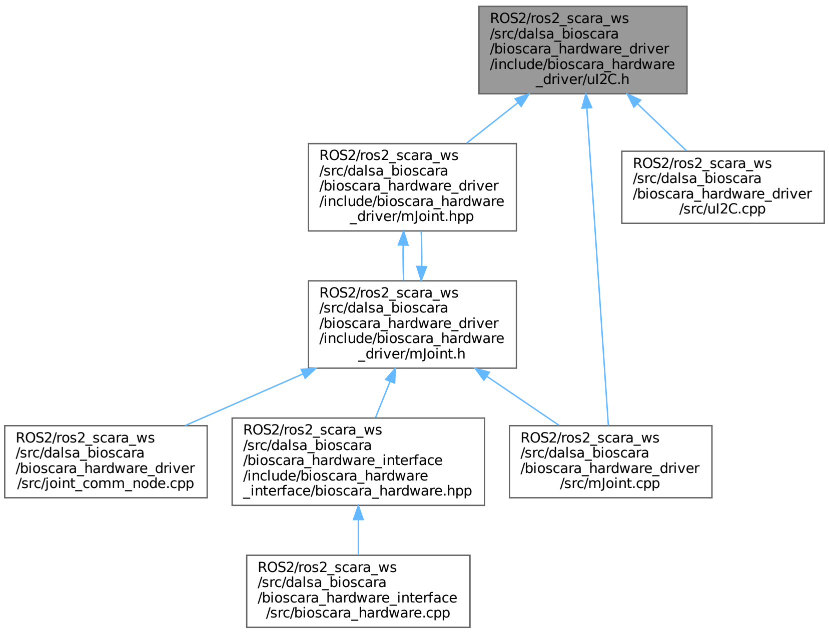
Low level utility for I2C communication on Raspberry Pi using lgpio library.

```
#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:


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/uI2C.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/uI2C.cpp
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 /src/joint_comm_node.cpp
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_driver /src/mJoint.cpp
ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_interface /src/bioscara_hardware.cpp

Macros

- #define ACK 'O'
- #define NACK 'N'
- #define RFLAGS_SIZE 1

  Size of the return flags in bytes.
- #define MAX_BUFFER 4

  Maximum size of I2C Payload in bytes.

Functions

- int openI2CDevHandle (const int dev_addr)

  Initiates an I2C device on the bus.
- int readFromI2CDev (const int dev_handle, const int reg, char ∗buffer, const int data_length)

  reads block of bytes from device to buffer
- int writeToI2CDev (const int dev_handle, const int reg, char ∗tx_buffer, const int data_length, char ∗RFLAGS_buffer)

  writes block of bytes from buffer to device
- int closeI2CDevHandle (int &dev_handle)

  close an I2C device on the bus

13.39.1 Detailed Description

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

Author

Sebastian Storz

Version

0.1

Date

2025-05-28

Copyright

Copyright (c) 2025

lgpio needs to be installed and linked! Installation:
```
cd ~
sudo apt update
sudo apt install -y swig
wget https://github.com/joan2937/lg/archive/master.zip
unzip master.zip
cd lg-master
make
sudo make install
cd ..
sudo rm -rf lg-master
rm master.zip
```

bash

13.39.2 Macro Definition Documentation

13.39.2.1 ACK

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

13.39.2.2 MAX_BUFFER

```
#define MAX_BUFFER 4
```

Maximum size of I2C Payload in bytes.

4 bytes used to transmit floats and int32_t

13.39.2.3 NACK

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

13.39.2.4 RFLAGS_SIZE

```
#define RFLAGS_SIZE 1
```

Size of the return flags in bytes.

Only one byte used and hence set to 1.

13.39.3 Function Documentation

13.39.3.1 closeI2CDevHandle()

```
int closeI2CDevHandle (
            int & dev_handle )
```

close an I2C device on the bus

Parameters

dev_handle	device handle obtained from `openI2CDevHandle`

Returns

0 on OK, negative on error.

13.39.3.2 openI2CDevHandle()

```
int openI2CDevHandle (
            const int dev_addr )
```

Initiates an I2C device on the bus.

Parameters

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

Returns

the device handle, negative on error.

13.39.3.3 readFromI2CDev()

```
int readFromI2CDev (
            const int dev_handle,
            const int reg,
            char * buffer,
            const int data_length )
```

reads block of bytes from device to buffer

Parameters

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.

13.39.3.4 writeToI2CDev()

```
int writeToI2CDev (
            const int dev_handle,
            const int reg,
            char * tx_buffer,
            const int data_length,
            char * RFLAGS_buffer )
```

writes block of bytes from buffer to device

Parameters

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.

13.40 uI2C.h

Go to the documentation of this file.

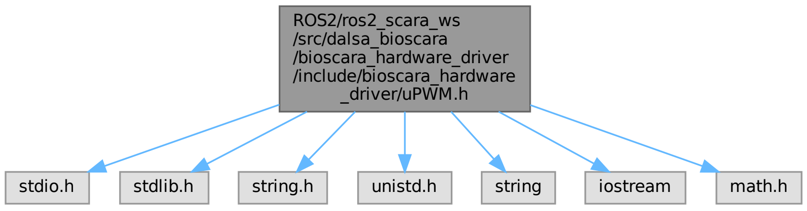
```

#ifndef USERIAL_H
#define USERIAL_H
#include <cstring>
#include <errno.h>
#include <fcntl.h>
#include <iostream>
#include <termios.h>
#include <unistd.h>

#define ACK 'O'
#define NACK 'N'

#define RFLAGS_SIZE 1

#define MAX_BUFFER 4 // Bytes

int openI2CDevHandle(const int dev_addr);
int readFromI2CDev(const int dev_handle, const int reg, char *buffer, const int data_length);

int writeToI2CDev(const int dev_handle, const int reg, char *tx_buffer, const int data_length, char
    *RFLAGS_buffer);

int closeI2CDevHandle(int &dev_handle);


#endif
```

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

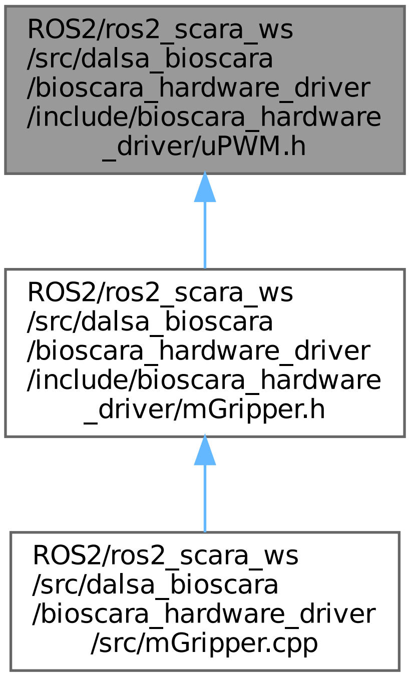
Includes source code for Hardware PWM generation on Raspberry Pi 4.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <string>
#include <iostream>
#include <math.h>
```

Include dependency graph for uPWM.h:


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /include/bioscara_hardware _driver/uPWM.h
stdio.h
stdlib.h
string.h
unistd.h
string
iostream
math.h

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


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

Classes

- class RPI_PWM

    PWM class for the Raspberry PI 4 and 5.

13.41.1 Detailed Description

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

Author

Sebastian Storz and Bernd Porr, `bernd.porr@glasgow.ac.uk`

Version

0.1

Date

2025-05-27

I copied this from: https://github.com/berndporr/rpi_pwm/blob/main/rpi_pwm.h and slightly modified it.

lgpio, the library used for I2C access can only generate soft PWM, The timing jitter will cause the servo to fidget. This may cause it to overheat and wear out prematurely.

Copyright

Copyright (c) 2025

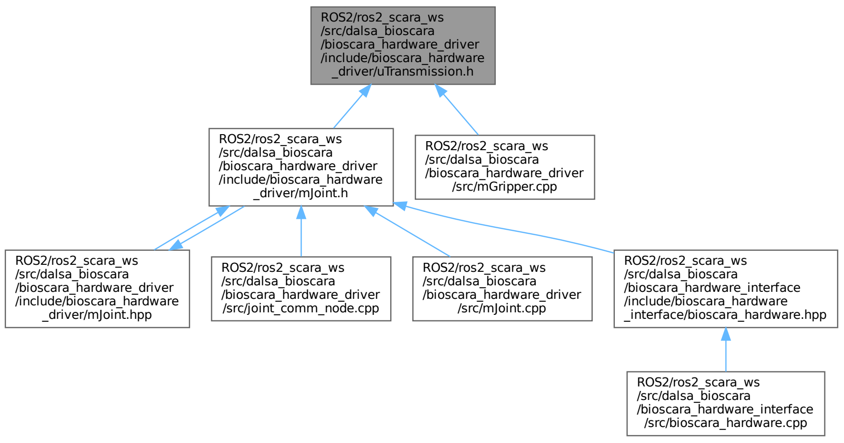
13.42 uPWM.h

Go to the documentation of this file.

```

#ifndef __RPIPWM
#define __RPIPWM

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <string>
#include <iostream>
#include <math.h>

class RPI_PWM
{
public:
    int start(int channel, int frequency, float duty_cycle = 0, int chip = 2)
    {
        chippath = "/sys/class/pwm/pwmchip" + std::to_string(chip);
        pwmpath = chippath + "/pwm" + std::to_string(channel);
        std::string p = chippath + "/export";
        FILE *const fp = fopen(p.c_str(), "w");
        if (NULL == fp)
        {
            std::cerr « "PWM device does not exist. Make sure to add 'dtoverlay=pwm-2chan' to
/boot/firmware/config.txt.\n";
            return -1;
        }
        const int r = fprintf(fp, "%d", channel);
        fclose(fp);
        if (r < 0)
            return r;
        usleep(100000); // it takes a while till the PWM subdir is created
        per = (int)1E9 / frequency;
        setPeriod(per);
        setDutyCycle(duty_cycle);
        enable();
        return r;
    }

    void stop()
    {
        disable();
    }

    ~RPI_PWM()
    {
        disable();
    }

    inline int setDutyCycle(float v) const
    {
        const int dc = (int)round((float)per * (v / 100.0));
        const int r = setDutyCycleNS(dc);
        return r;
    }

private:
    void setPeriod(int ns) const
    {
        writeSYS(pwmpath + "/" + "period", ns);
    }

    inline int setDutyCycleNS(int ns) const
    {
        const int r = writeSYS(pwmpath + "/" + "duty_cycle", ns);
        return r;
    }

    void enable() const
    {
        writeSYS(pwmpath + "/" + "enable", 1);
    }

    void disable() const
    {
        writeSYS(pwmpath + "/" + "enable", 0);
    }

    int per = 0;

    std::string chippath;
    std::string pwmpath;

    inline int writeSYS(std::string filename, int value) const
    {
        FILE *const fp = fopen(filename.c_str(), "w");
        if (NULL == fp)
        {
            return -1;
        }
        const int r = fprintf(fp, "%d", value);
        fclose(fp);
        return r;
    }
};

#endif
```

13.43 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/include/bioscara_hardware_driver/uTransmission.h File Reference

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


ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/include/bioscara_hardware_driver/uTransmission.h
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/src/mGripper.cpp
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/joint_comm_node.cpp
ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mJoint.cpp
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

Macros

- #define JOINT2ACTUATOR(in, reduction, offset) (reduction ∗ (in - offset))

Macro for a simple transmission from joint units to actuator units.
- #define ACTUATOR2JOINT(in, reduction, offset) (in / reduction + offset)

  Macro for a simple transmission from actuator units to joint units.
- #define M_PI 3.14159265358979323846

  pi
- #define RAD2DEG(rad) (rad / M_PI ∗ 180)

  Macro to convert radians to degree.
- #define DEG2RAD(deg) (deg ∗ M_PI / 180)

  Macro to convert degree to radians.

13.43.1 Macro Definition Documentation

13.43.1.1 ACTUATOR2JOINT

```
#define ACTUATOR2JOINT(
              in,
              reduction,
              offset ) (in / reduction + offset)
```

Macro for a simple transmission from actuator units to joint units.

The translation is based on the ros2_control transmission interface, simple transmission. For position reduction and offset need to be used.
For velocity and acceleration only use reduction and NO offset
For effort/torque use 1/reduction and NO offset

13.43.1.2 DEG2RAD

```
#define DEG2RAD(
              deg ) (deg ∗ M_PI / 180)
```

Macro to convert degree to radians.

13.43.1.3 JOINT2ACTUATOR

```
#define JOINT2ACTUATOR(
              in,
              reduction,
              offset ) (reduction ∗ (in - offset))
```

Macro for a simple transmission from joint units to actuator units.

The translation is based on the ros2_control transmission interface, simple transmission. For position reduction and offset need to be used.
For velocity and acceleration only use reduction and NO offset
For effort/torque use 1/reduction and NO offset

13.43.1.4 M_PI

```
#define M_PI 3.14159265358979323846
```

pi

13.43.1.5 RAD2DEG

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

Macro to convert radians to degree.

13.44 uTransmission.h

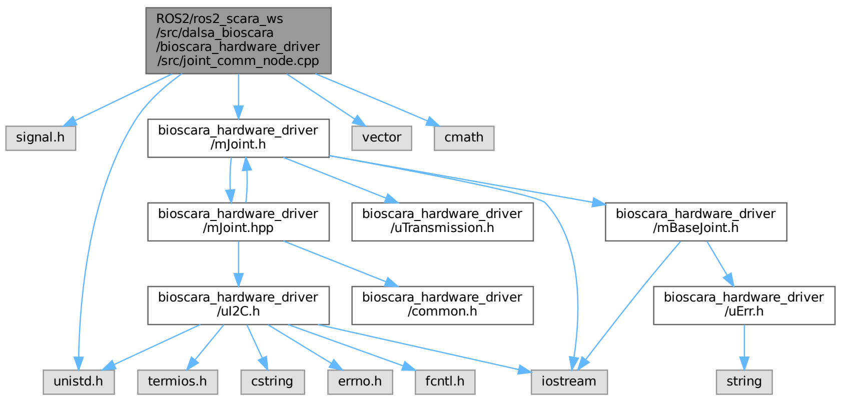
Go to the documentation of this file.

```
#ifndef UTRANSMISSION_H
#define UTRANSMISSION_H

#define JOINT2ACTUATOR(in, reduction, offset) (reduction * (in - offset))

#define ACTUATOR2JOINT(in, reduction, offset) (in / reduction + offset)

#define M_PI 3.14159265358979323846

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

#define DEG2RAD(deg) (deg * M_PI / 180)
#endif //UTRANSMISSION_H
```

13.45 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/src/joint_comm_node.cpp File Reference

```
#include <signal.h>
#include <unistd.h>
#include "bioscara_hardware_driver/mJoint.h"
#include <vector>
#include <cmath>
```

Include dependency graph for joint_comm_node.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)

13.45.1 Function Documentation

13.45.1.1 INT_handler()

```
void INT_handler (
             int s )
```

13.45.1.2 main()

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

13.45.2 Variable Documentation

13.45.2.1 J1

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

13.45.2.2 J2

```
Joint J2("j2", 0x12, -2 *M_PI/0.004, 0.338, 0.0) (
             "j2" ,
             0x12 ,
             -2 *M_PI/0.  004,
             0.  338,
             0.  0 )
```

13.45.2.3 J3

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

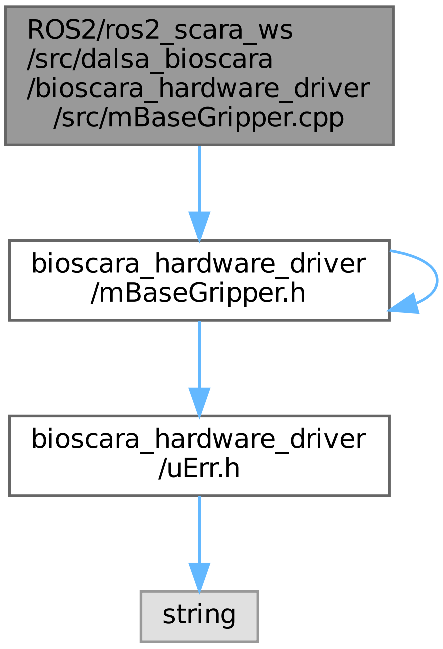
13.45.2.4 J4

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

13.46 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/src/mBaseGripper.cpp File Reference

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

Include dependency graph for mBaseGripper.cpp:


ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mBaseGripper.cpp
bioscara_hardware_driver/mBaseGripper.h
bioscara_hardware_driver/uErr.h
string

Namespaces

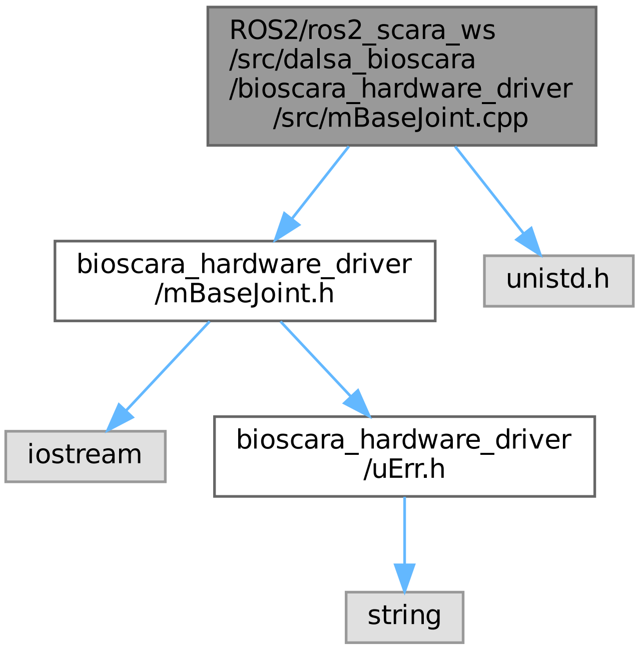
- namespace bioscara_hardware_driver

    Generic BaseGripper object to interact with the robot gripper.

13.47 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/src/mBaseJoint.cpp File Reference

```
#include "bioscara_hardware_driver/mBaseJoint.h"
#include <unistd.h>
```

Include dependency graph for mBaseJoint.cpp:



Namespaces

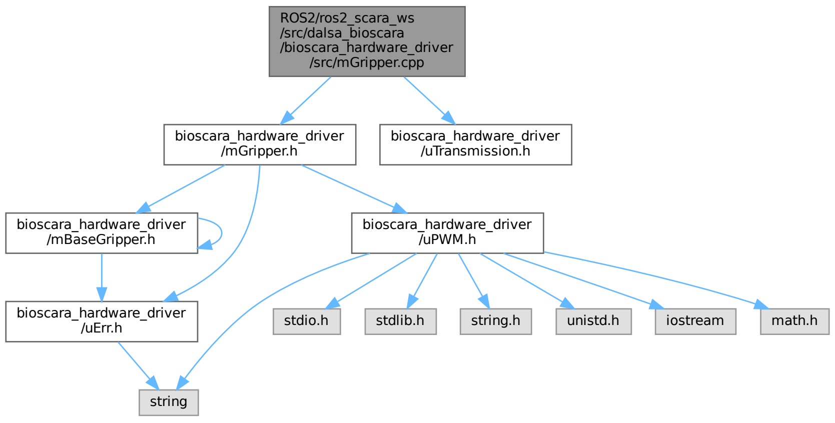
- namespace bioscara_hardware_driver

    Generic BaseGripper object to interact with the robot gripper.

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

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

Include dependency graph for mGripper.cpp:


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/mGripper.cpp
bioscara_hardware_driver /mGripper.h
bioscara_hardware_driver /uTransmission.h
bioscara_hardware_driver /mBaseGripper.h
bioscara_hardware_driver /uPWM.h
bioscara_hardware_driver /uErr.h
stdio.h
stdlib.h
string.h
unistd.h
iostream
math.h
string

Namespaces

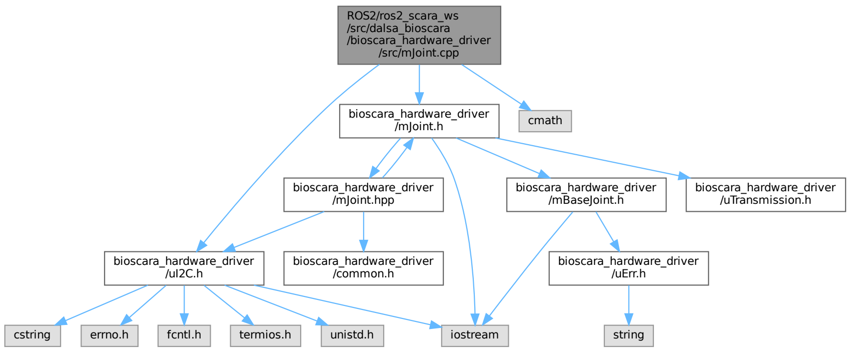
- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

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

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

Include dependency graph for mJoint.cpp:


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/mJoint.cpp
bioscara_hardware_driver /mJoint.h
cmath
bioscara_hardware_driver /mJoint.hpp
bioscara_hardware_driver /mBaseJoint.h
bioscara_hardware_driver /uTransmission.h
bioscara_hardware_driver /uI2C.h
bioscara_hardware_driver /common.h
bioscara_hardware_driver /uErr.h
cstring
errno.h
fcntl.h
termios.h
unistd.h
iostream
string

Namespaces

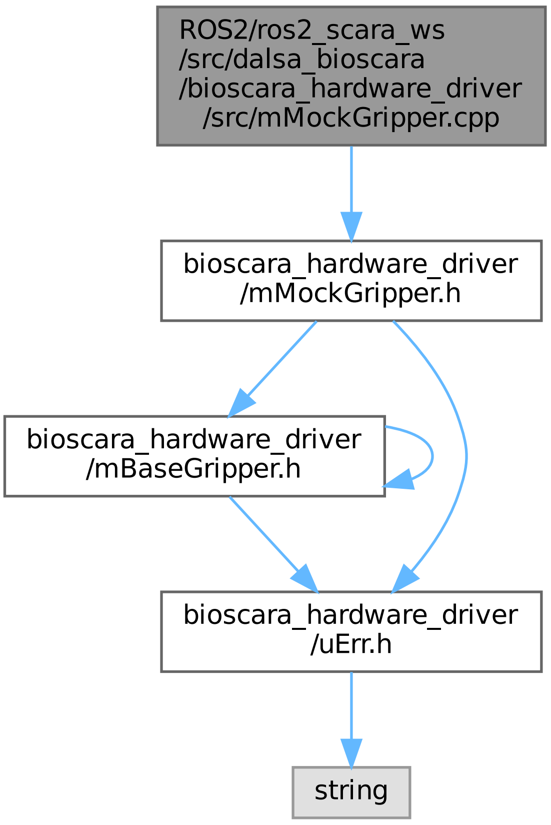
- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

13.50 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/src/mMockGripper.cpp File Reference

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

Include dependency graph for mMockGripper.cpp:


ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_driver/src/mMockGripper.cpp
bioscara_hardware_driver/mMockGripper.h
bioscara_hardware_driver/mBaseGripper.h
bioscara_hardware_driver/uErr.h
string

Namespaces

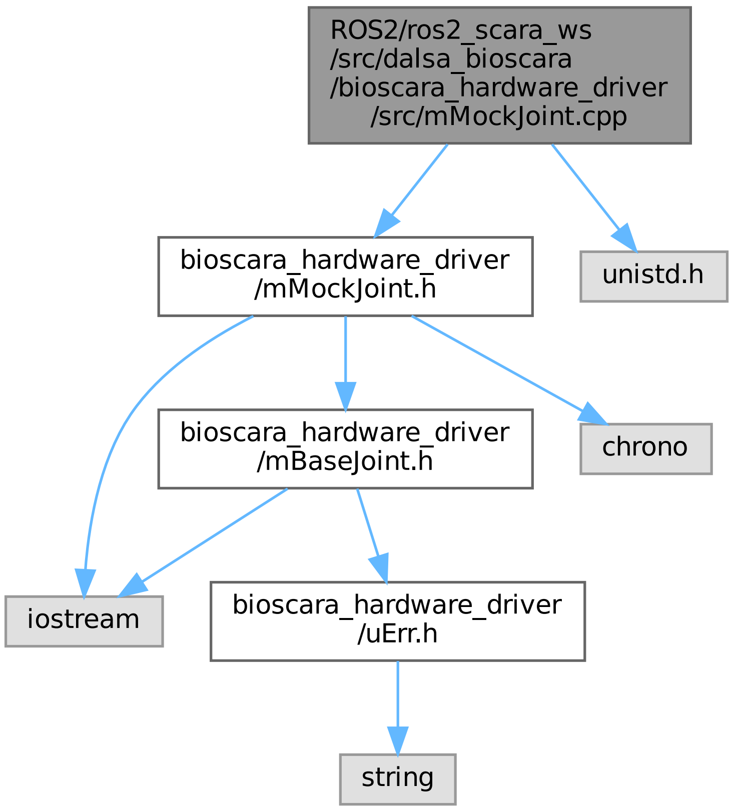
- namespace bioscara_hardware_driver

    Generic BaseGripper object to interact with the robot gripper.

13.51 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/src/mMockJoint.cpp File Reference

```
#include "bioscara_hardware_driver/mMockJoint.h"
#include <unistd.h>
```

Include dependency graph for mMockJoint.cpp:



Namespaces

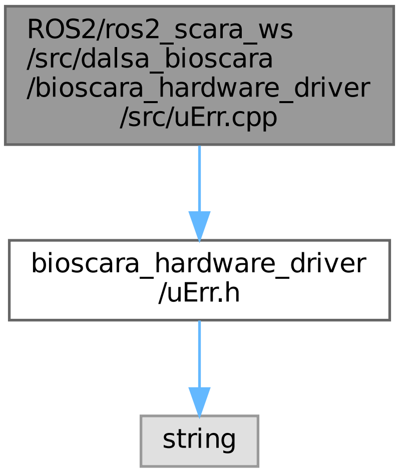
- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

13.52 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/src/uErr.cpp File Reference

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

Include dependency graph for uErr.cpp:



Namespaces

- namespace bioscara_hardware_driver

  Generic BaseGripper object to interact with the robot gripper.

Functions

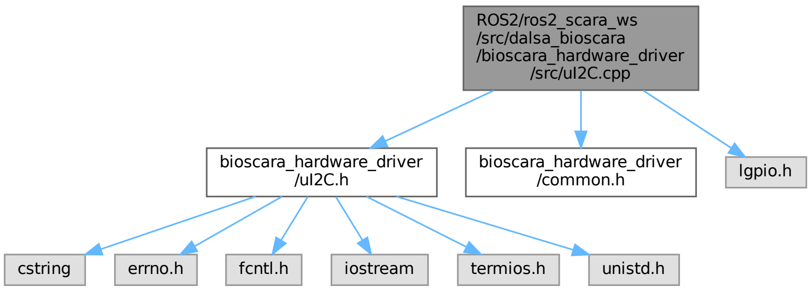
- std::string bioscara_hardware_driver::error_to_string (err_type_t err)

  Converts an error code to a string and returns it.

13.53 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ driver/src/uI2C.cpp File Reference

```
#include "bioscara_hardware_driver/uI2C.h"
#include "bioscara_hardware_driver/common.h"
#include <lgpio.h>
```

Include dependency graph for uI2C.cpp:


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_driver /src/uI2C.cpp
bioscara_hardware_driver /uI2C.h
bioscara_hardware_driver /common.h
lgpio.h
cstring
errno.h
fcntl.h
iostream
termios.h
unistd.h

Functions

- int openI2CDevHandle (const int dev_addr)

  Initiates an I2C device on the bus.
- int readFromI2CDev (const int dev_handle, const int reg, char ∗buffer, const int data_length)

  reads block of bytes from device to buffer
- int writeToI2CDev (const int dev_handle, const int reg, char ∗tx_buffer, const int data_length, char ∗RFLAGS_buffer)

  writes block of bytes from buffer to device
- int closeI2CDevHandle (int &dev_handle)

  close an I2C device on the bus

13.53.1 Function Documentation

13.53.1.1 closeI2CDevHandle()

```
int closeI2CDevHandle (
             int & dev_handle )
```

close an I2C device on the bus

Parameters

dev_handle	device handle obtained from `openI2CDevHandle`

Returns

0 on OK, negative on error.

13.53.1.2 openI2CDevHandle()

```
int openI2CDevHandle (
             const int dev_addr )
```

Initiates an I2C device on the bus.

Parameters

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

Returns

the device handle, negative on error.

13.53.1.3 readFromI2CDev()

```
int readFromI2CDev (
            const int dev_handle,
            const int reg,
            char * buffer,
            const int data_length )
```

reads block of bytes from device to buffer

Parameters

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.

13.53.1.4 writeToI2CDev()

```
int writeToI2CDev (
            const int dev_handle,
            const int reg,
            char * tx_buffer,
            const int data_length,
            char * RFLAGS_buffer )
```

writes block of bytes from buffer to device

Parameters

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

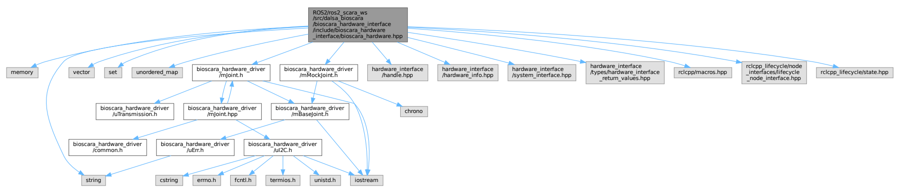
Returns

0 on OK, negative on error.

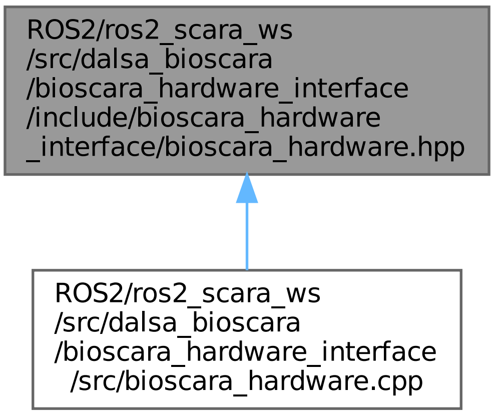
13.54 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ interface/include/bioscara_hardware_interface/bioscara_↩ hardware.hpp File Reference

```
#include <memory>
#include <string>
#include <vector>
#include <set>
#include <unordered_map>
#include "bioscara_hardware_driver/mJoint.h"
#include "bioscara_hardware_driver/mMockJoint.h"
#include "hardware_interface/handle.hpp"
#include "hardware_interface/hardware_info.hpp"
#include "hardware_interface/system_interface.hpp"
#include "hardware_interface/types/hardware_interface_return_values.hpp"
#include "rclcpp/macros.hpp"
#include "rclcpp_lifecycle/node_interfaces/lifecycle_node_interface.hpp"
#include "rclcpp_lifecycle/state.hpp"
```

Include dependency graph for bioscara_hardware.hpp:


ROS2/ros2_scara_ws /src/dalsa_bioscara /bioscara_hardware_interface /include/bioscara_hardware _interface/bioscara_hardware.hpp
memory
vector
set
unordered_map
bioscara_hardware_driver /mJoint.h
bioscara_hardware_driver /mMockJoint.h
hardware_interface /handle.hpp
hardware_interface /hardware_info.hpp
hardware_interface /system_interface.hpp
hardware_interface /types/hardware_interface _return_values.hpp
rclcpp/macros.hpp
rclcpp_lifecycle/node _interfaces/lifecycle _node_interface.hpp
rclcpp_lifecycle/state.hpp
bioscara_hardware_driver /uTransmission.h
bioscara_hardware_driver /mJoint.hpp
bioscara_hardware_driver /mBaseJoint.h
chrono
bioscara_hardware_driver /common.h
bioscara_hardware_driver /uErr.h
bioscara_hardware_driver /uI2C.h
string
cstring
errno.h
fcntl.h
termios.h
unistd.h
iostream

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


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

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"

13.55 bioscara_hardware.hpp

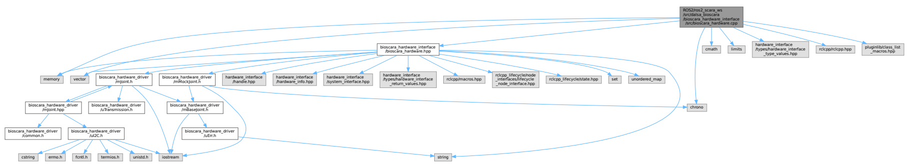
Go to the documentation of this file.

```
// Copyright 2023 ros2_control Development Team
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.

#ifndef BIOSCARA_HARDWARE_INTERFACE_HPP_
#define BIOSCARA_HARDWARE_INTERFACE_HPP_

#include <memory>
#include <string>
#include <vector>
#include <set>
#include <unordered_map>
#include <memory>

#include "bioscara_hardware_driver/mJoint.h"
#include "bioscara_hardware_driver/mMockJoint.h"

#include "hardware_interface/handle.hpp"
#include "hardware_interface/hardware_info.hpp"
#include "hardware_interface/system_interface.hpp"
#include "hardware_interface/types/hardware_interface_return_values.hpp"
#include "rclcpp/macros.hpp"
#include "rclcpp_lifecycle/node_interfaces/lifecycle_node_interface.hpp"
#include "rclcpp_lifecycle/state.hpp"

namespace bioscara_hardware_interface
{
    constexpr char HW_IF_HOME[] = "home";

    class BioscaraHardwareInterface : public hardware_interface::SystemInterface
    {
    public:
        RCLCPP_SHARED_PTR_DEFINITIONS(BioscaraHardwareInterface)


        hardware_interface::CallbackReturn on_init(
            const hardware_interface::HardwareComponentInterfaceParams &params) override;

        hardware_interface::CallbackReturn on_shutdown(
            const rclcpp_lifecycle::State &previous_state) override;

        hardware_interface::CallbackReturn on_configure(
            const rclcpp_lifecycle::State &previous_state) override;

        hardware_interface::CallbackReturn on_cleanup(
            const rclcpp_lifecycle::State &previous_state) override;

        hardware_interface::CallbackReturn on_activate(
            const rclcpp_lifecycle::State &previous_state) override;

        hardware_interface::CallbackReturn on_deactivate(
            const rclcpp_lifecycle::State &previous_state) override;

        hardware_interface::return_type read(
            const rclcpp::Time &time,
            const rclcpp::Duration &period) override;

        hardware_interface::return_type write(
            const rclcpp::Time &time,
            const rclcpp::Duration &period) override;

        hardware_interface::return_type prepare_command_mode_switch(
            const std::vector<std::string> &start_interfaces,
            const std::vector<std::string> &stop_interfaces) override;

         hardware_interface::return_type perform_command_mode_switch(
            const std::vector<std::string> & start_interfaces,
            const std::vector<std::string> & stop_interfaces) override;

        hardware_interface::CallbackReturn on_error(
            const rclcpp_lifecycle::State &previous_state) override;

    private:
        struct joint_homing_config_t
        {
            float speed = 0;
            u_int8_t threshold = 10;
            u_int8_t current = 10;
            float acceleration = 0.01;
        };

        struct joint_config_t
        {
            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;
        };

        std::unordered_map<std::string, std::unique_ptr<bioscara_hardware_driver::BaseJoint» _joints;

        std::unordered_map<std::string, joint_config_t> _joint_cfg;

        std::unordered_map<std::string, std::set<std::string» _joint_command_modes;

        bioscara_hardware_driver::err_type_t start_homing(const std::string name, float velocity);

        bioscara_hardware_driver::err_type_t stop_homing(const std::string name);

        void split_interface_string_to_joint_and_name(std::string interface, std::string &joint_name, std::string &interface_name);

        bioscara_hardware_driver::err_type_t activate_joint(const std::string name);

        bioscara_hardware_driver::err_type_t deactivate_joint(const std::string name);
    };

} // namespace bioscara_hardware_interface

#endif // BIOSCARA_HARDWARE_INTERFACE_HPP_
```

13.56 ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_↩ interface/src/bioscara_hardware.cpp File Reference

```
#include "bioscara_hardware_interface/bioscara_hardware.hpp"
#include <chrono>
#include <cmath>
#include <limits>
#include <memory>
#include <vector>
#include "hardware_interface/types/hardware_interface_type_values.hpp"
#include "rclcpp/rclcpp.hpp"
#include "pluginlib/class_list_macros.hpp"
```

Include dependency graph for bioscara_hardware.cpp:


ROS2/ros2_scara_ws/src/dalsa_bioscara/bioscara_hardware_interface/src/bioscara_hardware.cpp
bioscara_hardware_interface/bioscara_hardware.hpp
cmath
limits
hardware_interface/types/hardware_interface_type_values.hpp
rclcpp/rclcpp.hpp
pluginlib/class_list_macros.hpp
memory
vector
bioscara_hardware_driver/mJoint.h
bioscara_hardware_driver/mMockJoint.h
hardware_interface/handle.hpp
hardware_interface/hardware_info.hpp
hardware_interface/system_interface.hpp
hardware_interface/types/hardware_interface_return_values.hpp
rclcpp/macros.hpp
rclcpp_lifecycle/node_interfaces/lifecycle_node_interface.hpp
rclcpp_lifecycle/state.hpp
set
unordered_map
bioscara_hardware_driver/mJoint.hpp
bioscara_hardware_driver/uTransmission.h
bioscara_hardware_driver/mBaseJoint.h
chrono
bioscara_hardware_driver/common.h
bioscara_hardware_driver/uI2C.h
bioscara_hardware_driver/uErr.h
cstring
errno.h
fcntl.h
termios.h
unistd.h
iostream
string

Namespaces

- namespace bioscara_hardware_interface

Index