Bioscara

Generated by Doxygen 1.9.8

Chapter 1

Todo List

Member Joint_comms::addJoint (const int address, const std::string name, const float gearRatio, const int offset)

Measure joint ranges

- Investigate if possible to make independent of homing

Member Joint_comms::checkOrientations (std::vector< float > angle_v)

Only execute if not performed before

- save in private flag and inhibit movement if this has not been executed.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Gripper Class Reference

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

Public Member Functions

- Gripper (void)
- int init (void)
- int deinit (void)
- int enable (void)
- int disable (void)
- int setPosition (float width)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Gripper()

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

4.1.2 Member Function Documentation

4.1.2.1 deinit()

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

4.1.2.2 disable()

```
int Gripper::disable (
            void  )
```

4.1.2.3 enable()

```
int Gripper::enable (
            void  )
```

4.1.2.4 init()

```
int Gripper::init (
            void  )
```

4.1.2.5 setPosition()

```
int Gripper::setPosition (
            float width )
```

Sets the gripper position/width in mm from the closed position

Parameters

width	width in mm

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

- /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/mGripper.h
- /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/src/mGripper.cpp

4.2 Joint Class Reference

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

Public Member Functions

- Joint (const int address, const std::string name, const float gearRatio, const int offset)
- int init (void)
- int deinit (void)
- int printInfo (void)
- int getPosition (float &angle)
- int setPosition (float angle)
- int getVelocity (float &degps)
- int setVelocity (float degps)
- int checkOrientation (float angle=10.0)
- int enable (u_int8_t driveCurrent, u_int8_t holdCurrent)
- int disable (void)
- int home (u_int8_t direction, u_int8_t rpm, int8_t sensitivity, u_int8_t current)
- int stop (bool mode)
- int disableCL (void)
- int setDriveCurrent (u_int8_t current)

- int setHoldCurrent (u_int8_t current)
- int setBrakeMode (u_int8_t mode)
- int getStall (u_int8_t &stall)
- int enableStallguard (u_int8_t sensitivity)

  Enable encoder stall detection. A detected stall can be reset by homeing.
- int getIsHomed (u_int8_t &homed)
- int getIsHomed (void)
- bool isHomed (void)
- int getIsSetup (u_int8_t &setup)
- int getIsSetup (void)
- bool isSetup (void)
- int moveSteps (int32_t steps)
- int checkCom (void)
- u_int8_t getFlags (void)

Public Attributes

- std::string name

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Joint()

```
Joint::Joint (
            const int address,
            const std::string name,
            const float gearRatio,
            const int offset )
```

4.2.2 Member Function Documentation

4.2.2.1 checkCom()

```
int Joint::checkCom (
            void  )
```

4.2.2.2 checkOrientation()

```
int Joint::checkOrientation (
            float angle = 10.0 )
```

4.2.2.3 deinit()

```
int Joint::deinit (
            void  )
```

4.2.2.4 disable()

```
int Joint::disable (
            void  )
```

disenganges the joint motor without closing i2c handle

Returns

error code.

4.2.2.5 disableCL()

```
int Joint::disableCL (
            void  )
```

Disables the Closed-Loop PID Controller

Returns

error code.

4.2.2.6 enable()

```
int Joint::enable (
            u_int8_t driveCurrent,
            u_int8_t holdCurrent )
```

Initialize the driver

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

error code.

4.2.2.7 enableStallguard()

```
int Joint::enableStallguard (
            u_int8_t sensitivity )
```

Enable encoder stall detection. A detected stall can be reset by homeing.

Parameters

sensitivity	Encoder stalldetect sensitivity - From -100 to 10 where lower number is less sensitive and higher is more sensitive

4.2.2.8 getFlags()

```
u_int8_t Joint::getFlags (
            void  )
```

get driver state flags

Returns

flags.

4.2.2.9 getIsHomed() [1/2]

```
int Joint::getIsHomed (
            u_int8_t & homed )
```

checks if the joint is homed from the joint

Parameters

homed	not homed: 0, homed: 1

Returns

error code.

4.2.2.10 getIsHomed() [2/2]

```
int Joint::getIsHomed (
            void  )
```

checks if the joint is homed from the joint

Returns

error code.

4.2.2.11 getIsSetup() [1/2]

```
int Joint::getIsSetup (
            u_int8_t & setup )
```

checks if the joint is setup from the joint

Parameters

setup	not setup: 0, setup: 1

Returns

error code.

4.2.2.12 getIsSetup() [2/2]

```
int Joint::getIsSetup (
            void  )
```

checks if the joint is setup from the joint

Returns

error code.

4.2.2.13 getPosition()

```
int Joint::getPosition (
            float & angle )
```

4.2.2.14 getStall()

```
int Joint::getStall (
            u_int8_t & stall )
```

checks if the motor is stalled

Parameters

stall	not stalled: 0, stalled: 1

Returns

error code.

4.2.2.15 getVelocity()

```
int Joint::getVelocity (
            float & degps )
```

4.2.2.16 home()

```
int Joint::home (
            u_int8_t direction,
            u_int8_t rpm,
            int8_t sensitivity,
            u_int8_t current )
```

make motor move to end stop

Parameters

direction	CCW: 0, CW: 1.
rpm	speed of motor in rpm $>$ 10
sensitivity	Encoder stalldetect sensitivity - From -100 to 10 where lower number is less sensitive and higher is more sensitive
current	homeing current, determines how easy it is to stop the motor and thereby provoke a stall

Returns

error code.

4.2.2.17 init()

```
int Joint::init (
            void  )
```

4.2.2.18 isHomed()

```
bool Joint::isHomed (
            void  )
```

Returns

the isHomed state variable.

4.2.2.19 isSetup()

```
bool Joint::isSetup (
            void  )
```

Returns

the isSetup state variable.

4.2.2.20 moveSteps()

```
int Joint::moveSteps (
            int32_t steps )
```

4.2.2.21 printInfo()

```
int Joint::printInfo (
            void  )
```

4.2.2.22 setBrakeMode()

```
int Joint::setBrakeMode (
            u_int8_t mode )
```

Set Brake Mode

Parameters

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

Returns

error code.

4.2.2.23 setDriveCurrent()

```
int Joint::setDriveCurrent (
            u_int8_t current )
```

Set the Drive Current

Parameters

current	0% - 100% of driver current

Returns

error code.

4.2.2.24 setHoldCurrent()

```
int Joint::setHoldCurrent (
            u_int8_t current )
```

Set the Hold Current

Parameters

current	0% - 100% of driver current

Returns

error code.

4.2.2.25 setPosition()

```
int Joint::setPosition (
            float angle )
```

4.2.2.26 setVelocity()

```
int Joint::setVelocity (
            float degps )
```

4.2.2.27 stop()

```
int Joint::stop (
            bool mode )
```

Stops the motor

Note

When stopping the motor in soft mode, wait sufficiently long until the motor has stopped. Since the stop() function in the motor controller is blocking. Continously checking the busy flag also might interfere with the stop() function on the controller side.

Parameters

mode	Hard: 0, Soft: 1

Returns

error code.

4.2.3 Member Data Documentation

4.2.3.1 name

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

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

- /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/mJoint.h
- /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/mJoint.hpp
- /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/src/mJoint.cpp

4.3 Joint_comms Class Reference

Communication object for all joints.

`#include <mJointCom.h>`

Public Member Functions

- Joint_comms (void)
- ~Joint_comms ()
- int init (void)

  Initializes all joints.
- int deinit (void)

  Frees all joints from the I2C bus.
- void addJoint (const int address, const std::string name, const float gearRatio, const int offset)

  add Joints.
- int enables (std::vector< u_int8_t > driveCurrent_v, std::vector< u_int8_t > holdCurrent_v)

  Engages the joints.
- int enables (u_int8_t driveCurrent, u_int8_t holdCurrent)

  Engages the joints with the same current settings for all joints.
- int disables (void)

  Disenganges the joint without closing i2c handle.
- int home (std::string name, u_int8_t direction, u_int8_t rpm, int8_t sensitivity, u_int8_t current)

  Executes the homing sequence of a joint.
- int getPositions (std::vector< float > &angle_v)

  Get the positions of all joints.
- int setPositions (std::vector< float > angle_v)

  Set the positions of all joints.
- int getVelocities (std::vector< float > &degps_v)

  Get the velocities of all joints.
- int setVelocities (std::vector< float > degps_v)

  Set the velocities of all joints.
- int checkOrientations (std::vector< float > angle_v)

  Sequentially checks the orientations of each joint.
- int checkOrientations (float angle=10.0)

  Overload to use standard angle of 10 degrees.
- int stops (bool mode)

  Stops the motors.
- int disableCLs (void)

  Disables the Closed-Loop PID Controllers.
- int setDriveCurrents (std::vector< u_int8_t > current)

  Set the drive Currents.
- int setDriveCurrents (u_int8_t current)

  Overload to set all drive currents to the same value.
- int setHoldCurrents (std::vector< u_int8_t > current)

  Set the Hold Currents.
- int setHoldCurrents (u_int8_t current)

  Overload to set all hold currents to the same value.
- int setBrakeModes (u_int8_t mode)

  Set Brake Modes.
- int enableStallguards (std::vector< u_int8_t > thresholds)

  Enable encoder stall detection.

Public Attributes

- std::vector< Joint > joints

  Internal vector storing the Joint objects.

4.3.1 Detailed Description

Communication object for all joints.

CLass handling interfacing with the joints.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Joint_comms()

```
Joint_comms::Joint_comms (
            void  )
```

4.3.2.2 ~Joint_comms()

```
Joint_comms::~Joint_comms ( )
```

4.3.3 Member Function Documentation

4.3.3.1 addJoint()

```
void Joint_comms::addJoint (
            const int address,
            const std::string name,
            const float gearRatio,
            const int offset )
```

add Joints.

Appends a joint to internal vector.

Parameters

addresses	1-byte I2C device adress (0x11 ... 0x14) for J1 ... J4
names	string device name for output logs
gearRatio	gear ratio of joint. This is used to transform position and velocity commands in joint units to the stepper units. Signed: sign depends if homed CW or CCW. J1: 35; J2: -360/4 (4 mm per revolution); J3: 24; J4: 12;
offset	offset between encoder zero and joint zero (in joint units). J1: TBD; J2: -TBD (negative because homed at top); J3: TBD; J4: TBD;

Todo
- Measure joint ranges
- Investigate if possible to make independent of homing

4.3.3.2 checkOrientations() [1/2]

```
int Joint_comms::checkOrientations (
            float angle = 10.0 )
```

Overload to use standard angle of 10 degrees.

Returns

error code.

4.3.3.3 checkOrientations() [2/2]

```
int Joint_comms::checkOrientations (
            std::vector< float > angle_v )
```

Sequentially checks the orientations of each joint.

This function must only be called after the joint has been powered down. This function must be called after the joint has been enabled with enables() and before any movement.

Parameters

angle_v	vector of degrees to rotate to check the orientation. Should be small values of a few degrees.

Returns

error code.

Todo
- Only execute if not performed before
- save in private flag and inhibit movement if this has not been executed.

4.3.3.4 deinit()

```
int Joint_comms::deinit (
            void  )
```

Frees all joints from the I2C bus.

Deinitializes all joints by removing them from the I2C bus.

Returns

0 on success, non-zero otherwise

4.3.3.5 disableCLs()

```
int Joint_comms::disableCLs (
            void  )
```

Disables the Closed-Loop PID Controllers.

Returns

error code.

4.3.3.6 disables()

```
int Joint_comms::disables (
            void  )
```

Disenganges the joint without closing i2c handle.

Call this function when the joint should be in freedrive mode.

Returns

error code.

4.3.3.7 enables() [1/2]

```
int Joint_comms::enables (
            std::vector< u_int8_t > driveCurrent_v,
            std::vector< u_int8_t > holdCurrent_v )
```

Engages the joints.

Sets the drive and hold currents for each joint and engages the motor. Currents are in percent of driver max. output (2.5A, check with datasheet)

Parameters

driveCurrent_v	vector of drive currents 0-100. the i'th vector entry sets the current for the i'th added joint.
holdCurrent_v	vector of hold currents 0-100. the i'th vector entry sets the current for the i'th added joint.

Returns

error code.

4.3.3.8 enables() [2/2]

```
int Joint_comms::enables (
```

```
            u_int8_t driveCurrent,
            u_int8_t holdCurrent )
```

Engages the joints with the same current settings for all joints.

In this overload the same drive and hold currents are written to every joint.

Parameters

driveCurrent	drive current 0-100.
holdCurrent	hold current 0-100.

Returns

error code.

4.3.3.9 enableStallguards()

```
int Joint_comms::enableStallguards (
            std::vector< u_int8_t > thresholds )
```

Enable encoder stall detection.

If the PID error exceeds the set threshold a stall is triggered and the motor disabled. A detected stall can be reset by homeing.

Parameters

thresholds	Vector of thresholds. 0 - 255 where lower is more sensitive.

4.3.3.10 getPositions()

```
int Joint_comms::getPositions (
            std::vector< float > & angle_v )
```

Get the positions of all joints.

The current positions of all joints are returned. The units are degrees and mm for revolute and prismatic joints respectively.

Parameters

angle_v	Reference to allocated vector of appropriate size to hold all joint positions.

Returns

error code.

4.3.3.11 getVelocities()

```
int Joint_comms::getVelocities (
            std::vector< float > & degps_v )
```

Get the velocities of all joints.

The current velocities of all joints are returned. The units are degrees/s and mm/s for revolute and prismatic joints respectively.

Parameters

degps_v	Reference to allocated vector of appropriate size to hold all joint velocities.

Returns

error code.

4.3.3.12 home()

```
int Joint_comms::home (
            std::string name,
            u_int8_t direction,
            u_int8_t rpm,
            int8_t sensitivity,
            u_int8_t current )
```

Executes the homing sequence of a joint.

The joint will drive in the specified direction with the specified speed until a resistance which drives the current above the specified threshold is encountered. At this point the stepper stops and zeros the encoder.

Parameters

name	joint name.
direction	CCW: 0, CW: 1.
rpm	speed of motor in rpm > 10
sensitivity	Encoder stalldetect sensitivity - From -100 to 10 where lower number is less sensitive and higher is more sensitive
current	homeing current, determines how easy it is to stop the motor and thereby provoke a stall

Returns

error code.

4.3.3.13 init()

```
int Joint_comms::init (
            void  )
```

Initializes all joints.

Warning

Add some joints using addJoint() before calling this function. Iterates over all joints and initializes them on the I2C bus and tests if they are responsive.

Returns

0 on success, non-zero otherwise

4.3.3.14 setBrakeModes()

```
int Joint_comms::setBrakeModes (
            u_int8_t mode )
```

Set Brake Modes.

Applies the same brake modes to all joints. usefull to disengage all motors.

Parameters

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

Returns

error code.

4.3.3.15 setDriveCurrents() [1/2]

```
int Joint_comms::setDriveCurrents (
            std::vector< u_int8_t > current )
```

Set the drive Currents.

Warning

This function is unreliable and not well tested. Use enables() instead!

Parameters

current	0% - 100% of driver current

Returns

error code.

4.3.3.16 setDriveCurrents() [2/2]

```
int Joint_comms::setDriveCurrents (
            u_int8_t current )
```

Overload to set all drive currents to the same value.

Warning

This function is unreliable and not well tested. Use enables() instead!

Parameters

current	0% - 100% of driver current

Returns

error code.

4.3.3.17 setHoldCurrents() [1/2]

```
int Joint_comms::setHoldCurrents (
            std::vector< u_int8_t > current )
```

Set the Hold Currents.

Warning

This function is unreliable and not well tested. Use enables() instead!

Parameters

current	0% - 100% of driver current

Returns

error code.

4.3.3.18 setHoldCurrents() [2/2]

```
int Joint_comms::setHoldCurrents (
            u_int8_t current )
```

Overload to set all hold currents to the same value.

Warning

This function is unreliable and not well tested. Use enables() instead!

Parameters

current	0% - 100% of driver current

Returns

error code.

4.3.3.19 setPositions()

```
int Joint_comms::setPositions (
            std::vector< float > angle_v )
```

Set the positions of all joints.

Set new target positons of all joints. The units are degrees and mm for revolute and prismatic joints respectively.

Parameters

angle_v	Vector of new target positions.

Returns

error code.

4.3.3.20 setVelocities()

```
int Joint_comms::setVelocities (
            std::vector< float > degps_v )
```

Set the velocities of all joints.

Set new target positons of all joints. The units are degrees and mm for revolute and prismatic joints respectively.

Parameters

degps_v	Vector of new target velocities.

Returns

error code.

4.3.3.21 stops()

```
int Joint_comms::stops (
            bool mode )
```

Stops the motors.

Stops all motors either soft or hard.

Parameters

mode	Hard: 0, Soft: 1

Returns

error code.

4.3.4 Member Data Documentation

4.3.4.1 joints

```
std::vector<Joint> Joint_comms::joints
```

Internal vector storing the Joint objects.

A Joint can be added by invoking addJoint()∗

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

- /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/mJointCom.h
- /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/src/mJointCom.cpp

4.4 RPI_PWM Class Reference

```
#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

4.4.1 Detailed Description

PWM class for the Raspberry PI 5

4.4.2 Constructor & Destructor Documentation

4.4.2.1 ∼RPI_PWM()

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

4.4.3 Member Function Documentation

4.4.3.1 setDutyCycle()

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

Sets the duty cycle.

Parameters

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

4.4.3.2 start()

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

Starts the PWM

Parameters

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

4.4.3.3 stop()

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

Stops the PWM

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

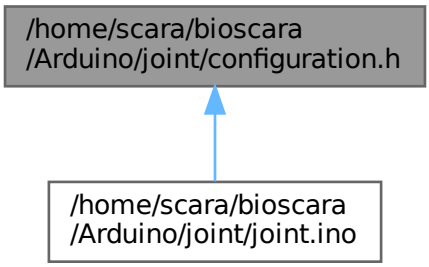
- /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/uPWM.h

Chapter 5

File Documentation

5.1 /home/scara/bioscara/Arduino/joint/configuration.h File Reference

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



5.2 configuration.h

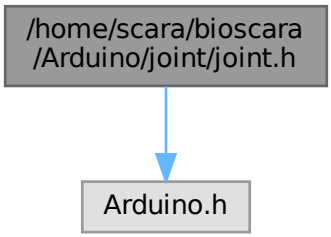
Go to the documentation of this file.

```
00001 #ifndef CONFIGURATION_H
00002 #define CONFIGURATION_H
00003
00004 #if defined(J1)
00005 #define ADR 0x11
00006 #define MAXACCEL 10000
00007 #define MAXVEL 800
00008
00009 #elif defined(J2)
00010 #define ADR 0x12
00011 #define MAXACCEL 10000
00012 #define MAXVEL 800
00013
00014 #elif defined(J3)
00015 #define ADR 0x13
00016 #define MAXACCEL 10000
00017 #define MAXVEL 800
00018
00019 #elif defined(J4)
00020 #define ADR 0x14
```

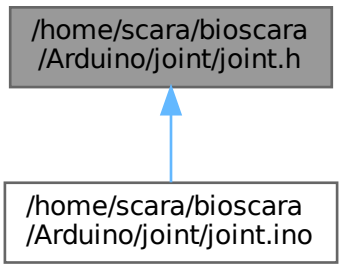
```
00021 #define MAXACCEL 10000
00022 #define MAXVEL 800
00023 #else
00024 #error "No Joint has been defined. Define one of 'JX' where X 1,2,3,4"
00025 #endif
00026
00027 #endif
```

5.3 /home/scara/bioscara/Arduino/joint/joint.h File Reference

`#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
- #define RFLAGS_SIZE 1
- #define DUMP_BUFFER(buffer, size)

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 , ISSTALLED = 0x21 , SETBRAKEMODE = 0x22 ,
  ENABLEPID = 0x23 , DISABLEPID = 0x24 , ENABLECLOSEDLOOP = 0x25 , DISABLECLOSEDLOOP = 0x26 ,
  SETCONTROLTHRESHOLD = 0x27 , MOVETOEND = 0x28 , STOP = 0x29 , GETPIDERROR = 0x2A ,
  CHECKORIENTATION = 0x2B , GETENCODERRPM = 0x2C , HOME = 0x2D , ISHOMED = 0x2E ,
  ISSETUP = 0x2F }

Functions

- uint8_t generateChecksum (const uint8_t *buffer, size_t length)
- template<typename T >
  void readValue (T &val, uint8_t *rxBuf, size_t rx_length)
- template<typename T >
  int writeValue (const T val, uint8_t *txBuf, size_t &tx_length)

5.3.1 Macro Definition Documentation

5.3.1.1 ACK

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

5.3.1.2 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(); \
}
```

5.3.1.3 MAX_BUFFER

```
#define MAX_BUFFER 4
```

5.3.1.4 NACK

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

5.3.1.5 RFLAGS_SIZE

```
#define RFLAGS_SIZE 1
```

5.3.2 Enumeration Type Documentation

5.3.2.1 stp_reg_t

```
enum stp_reg_t
```

Enumerator

PING	
SETUP	
SETRPM	
GETDRIVERRPM	
MOVESTEPS	
MOVEANGLE	
MOVETOANGLE	
GETMOTORSTATE	
RUNCOTINOUS	
ANGLEMOVED	
SETCURRENT	
SETHOLDCURRENT	
SETMAXACCELERATION	
SETMAXDECELERATION	
SETMAXVELOCITY	
ENABLESTALLGUARD	
DISABLESTALLGUARD	
CLEARSTALL	
ISSTALLED	
SETBRAKEMODE	
ENABLEPID	
DISABLEPID	
ENABLECLOSEDLOOP	
DISABLECLOSEDLOOP	
SETCONTROLTHRESHOLD	
MOVETOEND	
STOP	
GETPIDERROR	
CHECKORIENTATION	
GETENCODERRPM	
HOME	
ISHOMED	
ISSETUP	

5.3.3 Function Documentation

5.3.3.1 generateChecksum()

```
uint8_t generateChecksum (
            const uint8_t * buffer,
            size_t length )
```

Compute the two' complement checksum of the `buffer` according to SAE J1708

Parameters

buffer	Pointer to buffer to compute checksum off
length	Length of the buffer

Returns

Two's complement checksum.

5.3.3.2 readValue()

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

Reads a value from Serial Buffer of the specified type `T` into `val`

Parameters

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

5.3.3.3 writeValue()

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

Writes a value to the Serial output buffer using a intermediate buffer.

Parameters

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

Returns

0 On success

5.4 joint.h

Go to the documentation of this file.

```
00001 #ifndef JOINT_H
00002 #define JOINT_H
00003 #include <Arduino.h>
00004
00005 #define ACK 'O'
00006 #define NACK 'N'
00007
00008 #define MAX_BUFFER 4  // Bytes
00009 #define RFLAGS_SIZE 1
00010
00011 #define DUMP_BUFFER(buffer, size) \
00012   { \
00013     Serial.print("Buffer dump: "); \
00014     for (size_t i = 0; i < size; i++) { \
00015       Serial.print(buffer[i], HEX); \
00016       Serial.print(" "); \
00017     } \
00018     Serial.println(); \
00019   }
00020
00021 enum stp_reg_t {
00022   PING = 0x0f,
00023   SETUP = 0x10,
00024   SETRPM = 0x11,
00025   GETDRIVERRPM = 0x12,
00026   MOVESTEPS = 0x13,
00027   MOVEANGLE = 0x14,
00028   MOVETOANGLE = 0x15,
00029   GETMOTORSTATE = 0x16,
00030   RUNCOTINOUS = 0x17,
00031   ANGLEMOVED = 0x18,
00032   SETCURRENT = 0x19,
00033   SETHOLDCURRENT = 0x1A,
00034   SETMAXACCELERATION = 0x1B,
00035   SETMAXDECELERATION = 0x1C,
00036   SETMAXVELOCITY = 0x1D,
00037   ENABLESTALLGUARD = 0x1E,
00038   DISABLESTALLGUARD = 0x1F,
00039   CLEARSTALL = 0x20,
00040   ISSTALLED = 0x21,
00041   SETBRAKEMODE = 0x22,
00042   ENABLEPID = 0x23,
00043   DISABLEPID = 0x24,
00044   ENABLECLOSEDLOOP = 0x25,
00045   DISABLECLOSEDLOOP = 0x26,
00046   SETCONTROLTHRESHOLD = 0x27,
00047   MOVETOEND = 0x28,
00048   STOP = 0x29,
00049   GETPIDERROR = 0x2A,
00050   CHECKORIENTATION = 0x2B,
00051   GETENCODERRPM = 0x2C,
00052   HOME = 0x2D,
00053   ISHOMED = 0x2E,
00054   ISSETUP = 0x2F
00055 };
00056
00063 uint8_t generateChecksum(const uint8_t *buffer, size_t length);
00064
00065 // /**
00066 //  * Reads a value from Serial Buffer of the specified type `T` into `val`
00067 //  * @param val Reference to output variable
00068 //  * @param rxBuf Buffer to read value from
00069 //  * @param rx_length Length of the buffer
00070 //  * @return 0 On success, -1 on Timeout, -2 on CHK fail
00071 //  */
00072 // template<typename T>
00073 // int readValue(T &val, uint8_t *rxBuf, int rx_length);
00074
00075 // /**
00076 //  * Reads a value from Serial Buffer of the specified type `T` into `val`
00077 //  * @param val Reference to output variable
00078 //  * @param rxBuf Buffer to read value from
00079 //  * @param rx_length Length of the buffer
00080 //  * @return 0 On success, -1 on Timeout, -2 on CHK fail
```

```
00081 //  */
00082 // template<typename T>
00083 // int writeValue(const T val, uint8_t *txBuf, int tx_length);
00084
00085
00086
00087
00094 template<typename T>
00095 void readValue(T &val, uint8_t *rxBuf, size_t rx_length) {
00096   memcpy(&val, rxBuf, rx_length);
00097 }
00098
00106 template<typename T>
00107 int writeValue(const T val, uint8_t *txBuf, size_t &tx_length) {
00108   tx_length = sizeof(T);
00109   memcpy(txBuf, &val, tx_length);
00110   return 0;
00111 }
00112
00113 #endif
```

5.5 /home/scara/bioscara/Arduino/joint/joint.ino File Reference

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

Macros

- #define J1

Functions

- void stepper_receive_handler (uint8_t reg)
- void stepper_request_handler (uint8_t reg)
- void receiveEvent (int n)
- void requestEvent ()
- void setup (void)
- void loop (void)

Variables

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

5.5.1 Macro Definition Documentation

5.5.1.1 J1

```
#define J1
```

5.5.2 Function Documentation

5.5.2.1 loop()

```
void loop (
             void  )
```

5.5.2.2 receiveEvent()

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

5.5.2.3 requestEvent()

```
void requestEvent ( )
```

5.5.2.4 setup()

```
void setup (
             void  )
```

5.5.2.5 stepper_receive_handler()

```
void stepper_receive_handler (
             uint8_t reg )
```

Handles commands, is called from the main loop since it contains blocking function calls which can not be called from the I2C ISR.

Parameters

reg	command code

5.5.2.6 stepper_request_handler()

```
void stepper_request_handler (
             uint8_t reg )
```

Handles read request, is called 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 code

5.5.3 Variable Documentation

5.5.3.1 reg

```
uint8_t reg = 0
```

5.5.3.2 rx_buf

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

5.5.3.3 rx_data_ready

```
bool rx_data_ready = 0
```

5.5.3.4 rx_length

```
size_t rx_length = 0
```

5.5.3.5 stepper

```
UstepperS32 stepper
```

5.5.3.6 tx_buf

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

5.5.3.7 tx_data_ready

```
bool tx_data_ready = 0
```

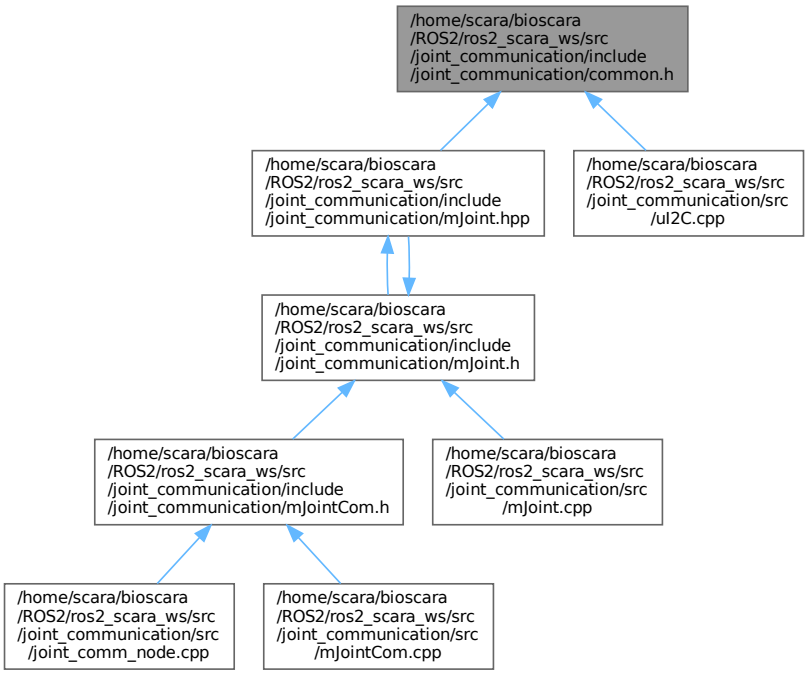
5.5.3.8 tx_length

```
size_t tx_length = 0
```

5.6 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/common.h File Reference

A file containing utility macros and functions.

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



Macros

- #define DUMP_BUFFER(buffer, size)

  Macro to dump a buffer to cout.

5.6.1 Detailed Description

A file containing utility macros and functions.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

5.6.2 Macro Definition Documentation

5.6.2.1 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

5.7 common.h

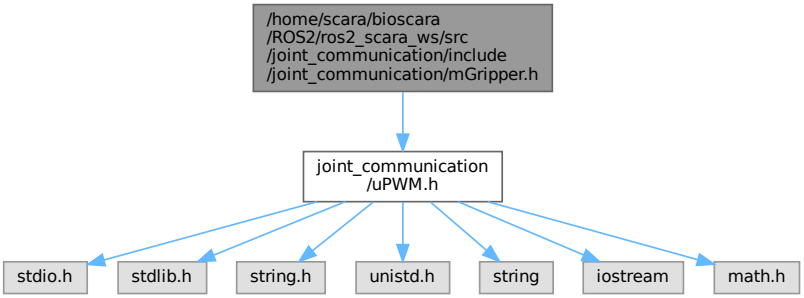
Go to the documentation of this file.

```
00001
00011 #ifndef COMMON_H
00012 #define COMMON_H
00013
00014
00021 #define DUMP_BUFFER(buffer, size)      \
00022   {                                     \
00023     std::cout « "Buffer dump: ";      \
00024     for (size_t i = 0; i < size; i++) \
00025     {                                   \
00026       printf("%#x ", buffer[i]);        \
00027     }                                   \
00028     std::cout « std::endl;            \
00029   }
00030
00031 #endif // COMMON_H
```

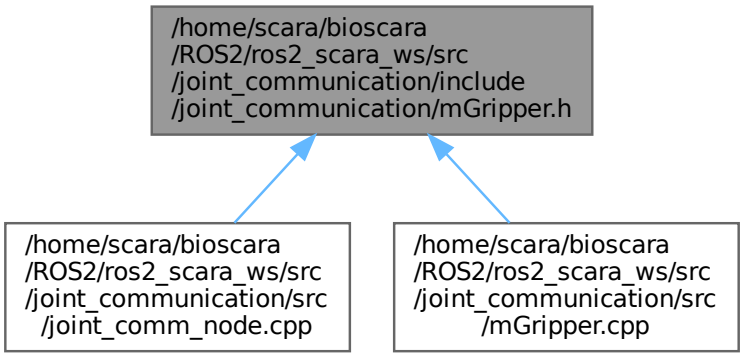
5.8 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/mGripper.h File Reference

```
#include "joint_communication/uPWM.h"
```

Include dependency graph for mGripper.h:



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



Classes

- class Gripper

5.9 mGripper.h

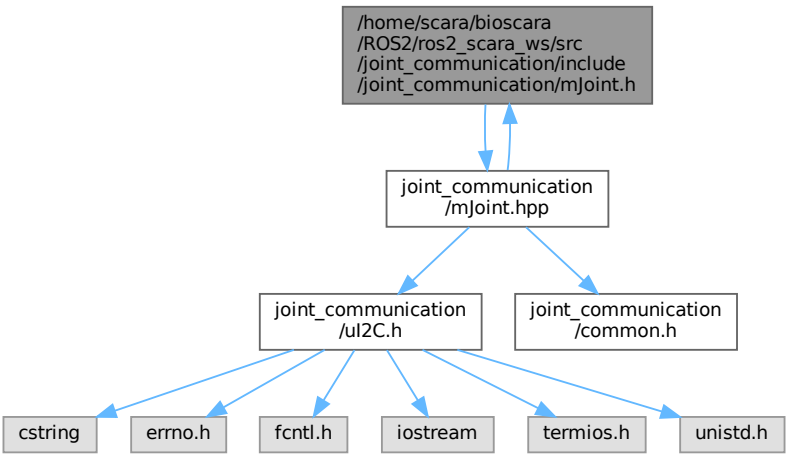
Go to the documentation of this file.

```
00001 #ifndef MGRIPPER_H
00002 #define MGRIPPER_H
```

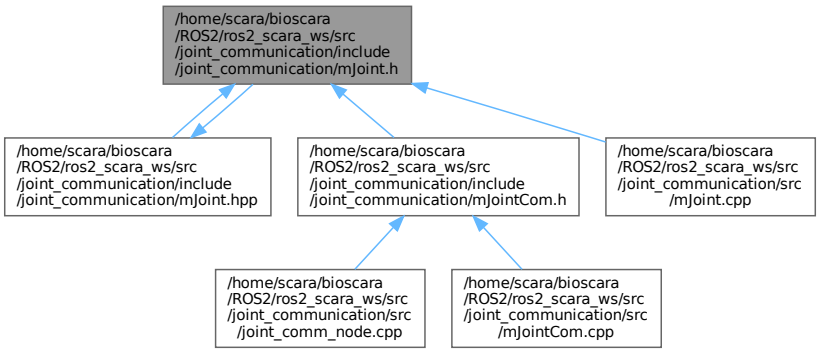
```
00003 #include "joint_communication/uPWM.h"
00004
00005 class Gripper
00006 {
00007 public:
00008     Gripper(void);
00009
00010     int init(void);
00011     int deinit(void);
00012     int enable(void);
00013     int disable(void);
00018     int setPosition(float width);
00019
00020 private:
00021     RPI_PWM pwm;
00022 };
00023 #endif // MGRIPPER_H
```

5.10 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_↩ communication/include/joint_communication/mJoint.h File Reference

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



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



Classes

- class Joint

Macros

- #define JOINT2ENCODERANGLE(jointAngle, gearRatio, offset) (gearRatio ∗ (jointAngle + offset))
- #define ENCODER2JOINTANGLE(encoderAngle, gearRatio, offset) (encoderAngle / gearRatio - offset)

5.10.1 Macro Definition Documentation

5.10.1.1 ENCODER2JOINTANGLE

```
#define ENCODER2JOINTANGLE(
              encoderAngle,
              gearRatio,
              offset ) (encoderAngle / gearRatio - offset)
```

5.10.1.2 JOINT2ENCODERANGLE

```
#define JOINT2ENCODERANGLE(
              jointAngle,
              gearRatio,
              offset ) (gearRatio * (jointAngle + offset))
```

5.11 mJoint.h

Go to the documentation of this file.

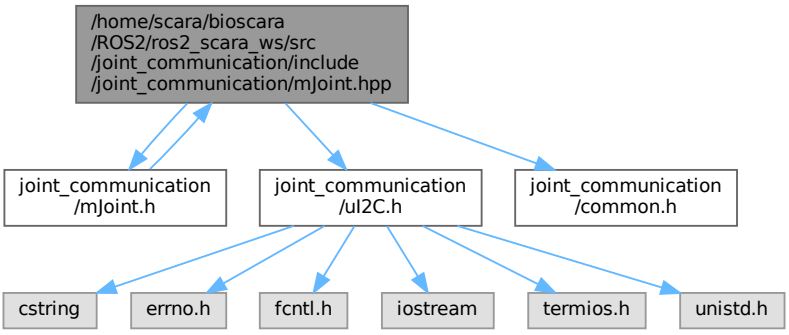
```
00001 #ifndef MJOINT_H
00002 #define MJOINT_H
00003
00004 #define JOINT2ENCODERANGLE(jointAngle, gearRatio, offset) (gearRatio * (jointAngle + offset))
00005 #define ENCODER2JOINTANGLE(encoderAngle, gearRatio, offset) (encoderAngle / gearRatio - offset)
00006
00007 class Joint
00008 {
00009 public:
00010   Joint(const int address, const std::string name, const float gearRatio, const int offset);
00011   // ~Joint();
00012
00013   int init(void);
00014   int deinit(void);
00015   int printInfo(void);
00016   int getPosition(float &angle);
00017   int setPosition(float angle);
00018   int getVelocity(float &degps);
00019   int setVelocity(float degps);
00020   int checkOrientation(float angle = 10.0);
00021
00028   int enable(u_int8_t driveCurrent, u_int8_t holdCurrent);
00029
00034   int disable(void);
00035
00045   int home(u_int8_t direction, u_int8_t rpm, int8_t sensitivity, u_int8_t current);
00046
00055   int stop(bool mode);
00060   int disableCL(void);
00061
00067   int setDriveCurrent(u_int8_t current);
00068
00074   int setHoldCurrent(u_int8_t current);
00075
00081   int setBrakeMode(u_int8_t mode);
00082
00088   int getStall(u_int8_t &stall);
00089
00094   int enableStallguard(u_int8_t sensitivity);
00095
00101   int getIsHomed(u_int8_t &homed);
00102
00107   int getIsHomed(void);
00108
00112   bool isHomed(void);
00113
00119   int getIsSetup(u_int8_t &setup);
00120
00125   int getIsSetup(void);
00126
00130   bool isSetup(void);
00131
00132   int moveSteps(int32_t steps);
00133   int checkCom(void);
00134
00139   u_int8_t getFlags(void);
00140
00141   std::string name;
00142
00143 protected:
00144 private:
00145   enum stp_reg_t
00146   {
00147     PING = 0x0f,
00148     SETUP = 0x10,
00149     SETRPM = 0x11,
00150     GETDRIVERRPM = 0x12,
00151     MOVESTEPS = 0x13,
00152     MOVEANGLE = 0x14,
00153     MOVETOANGLE = 0x15,
00154     GETMOTORSTATE = 0x16,
00155     RUNCOTINOUS = 0x17,
00156     ANGLEMOVED = 0x18,
00157     SETCURRENT = 0x19,
00158     SETHOLDCURRENT = 0x1A,
00159     SETMAXACCELERATION = 0x1B,
00160     SETMAXDECELERATION = 0x1C,
00161     SETMAXVELOCITY = 0x1D,
00162     ENABLESTALLGUARD = 0x1E,
00163     DISABLESTALLGUARD = 0x1F,
00164     CLEARSTALL = 0x20,
00165     ISSTALLED = 0x21,
```

```
00166     SETBRAKEMODE = 0x22,
00167     ENABLEPID = 0x23,
00168     DISABLEPID = 0x24,
00169     ENABLECLOSEDLOOP = 0x25,
00170     DISABLECLOSEDLOOP = 0x26,
00171     SETCONTROLTHRESHOLD = 0x27,
00172     MOVETOEND = 0x28,
00173     STOP = 0x29,
00174     GETPIDERROR = 0x2A,
00175     CHECKORIENTATION = 0x2B,
00176     GETENCODERRPM = 0x2C,
00177     HOME = 0x2D,
00178     ISHOMED = 0x2E,
00179     ISSETUP = 0x2F
00180   };
00181
00182   template <typename T>
00183   int read(const stp_reg_t reg, T &data, u_int8_t &flags);
00184
00185   template <typename T>
00186   int write(const stp_reg_t reg, T data, u_int8_t &flags);
00187
00188   u_int8_t flags = 0x00;
00189
00190   u_int8_t ishomed = 0;
00191   u_int8_t issetup = 0;
00192
00193   int address;
00194   float gearRatio = 1;
00195   int offset = 0;
00196
00197   int handle = -1;
00198 };
00199
00200 #include "joint_communication/mJoint.hpp"
00201
00202 #endif
```

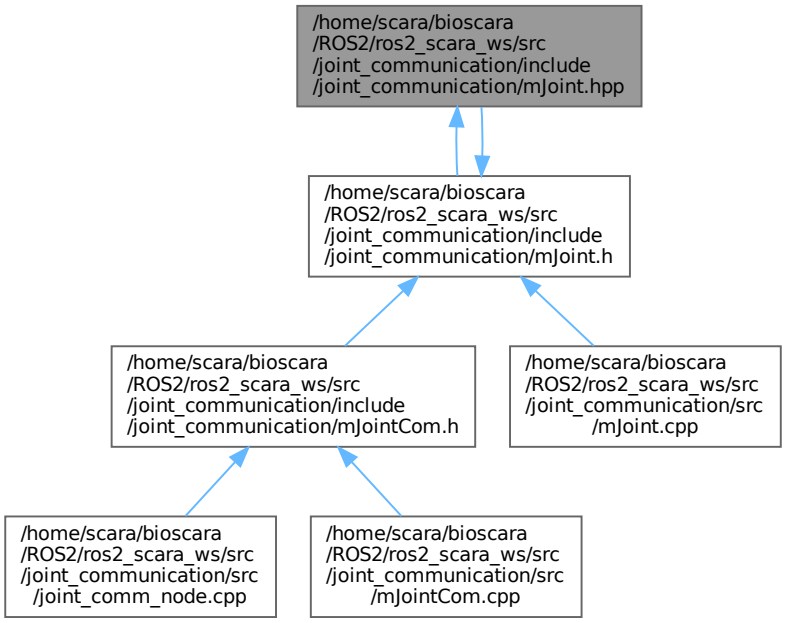
5.12 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/mJoint.hpp File Reference

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

Include dependency graph for mJoint.hpp:



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



5.13 mJoint.hpp

Go to the documentation of this file.

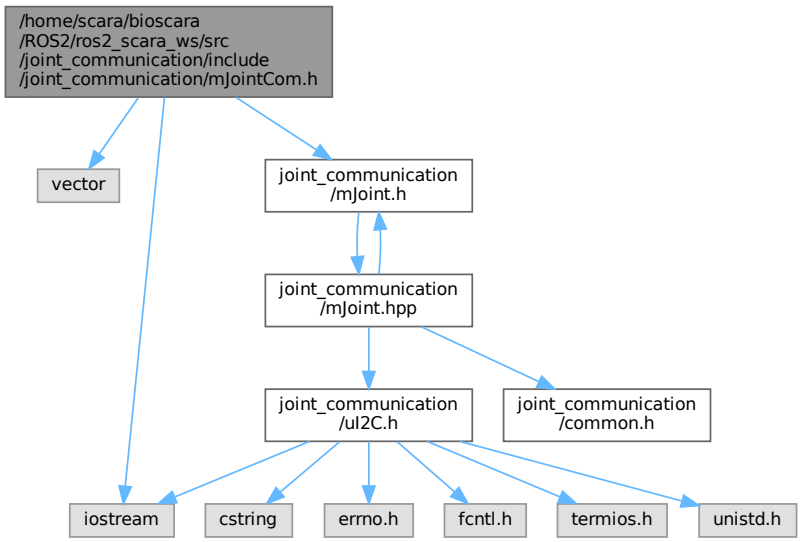
```
00001 #include "joint_communication/mJoint.h"
00002 #include "joint_communication/uI2C.h"
00003 #include "joint_communication/common.h"
00004
00005
00006 template <typename T>
00007 int Joint::read(const stp_reg_t reg, T &data, u_int8_t &flags)
00008 {
00009     size_t size = sizeof(T) + RFLAGS_SIZE;
00010     char *buf = new char[size];
00011     int n = readFromI2CDev(this->handle, reg, buf, size);
00012     if (n != static_cast<int>(size))
00013     {
00014         delete[] buf;
00015         return -1;
00016     }
00017     memcpy(&data, buf, size - RFLAGS_SIZE);
00018     memcpy(&flags, buf + size - RFLAGS_SIZE, RFLAGS_SIZE);
00019     delete[] buf;
00020     return 0;
00021 }
00022
00023 template <typename T>
00024 int Joint::write(const stp_reg_t reg, T data, u_int8_t &flags)
00025 {
00026     size_t size = sizeof(T) + RFLAGS_SIZE;
00027     char *buf = new char[size];
00028     memcpy(buf, &data, size - RFLAGS_SIZE);
00029     int rc = writeToI2CDev(this->handle, reg, buf, size - RFLAGS_SIZE, buf + size - RFLAGS_SIZE);
00030     rc = rc > 0 ? 0 : rc;
00031
00032     memcpy(&flags, buf + size - RFLAGS_SIZE, RFLAGS_SIZE);
00033     delete[] buf;
00034     return rc;
00035 }
00036
```

5.14 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/mJointCom.h File Reference

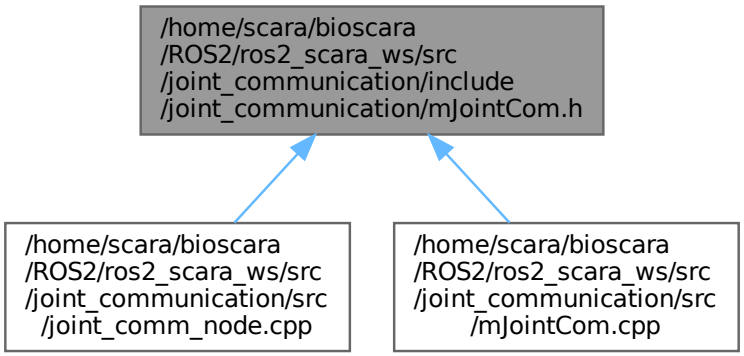
File containing the Joint_comms class.

```
#include <vector>
#include <iostream>
#include "joint_communication/mJoint.h"
```

Include dependency graph for mJointCom.h:



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



Classes

- class Joint_comms

  Communication object for all joints.

5.14.1 Detailed Description

File containing the Joint_comms class.

Author

Sebastian Storz

Version

0.1

Date

2025-05-27

Copyright

Include this file for API functions to interact with the stepper motors.

5.15 mJointCom.h

Go to the documentation of this file.

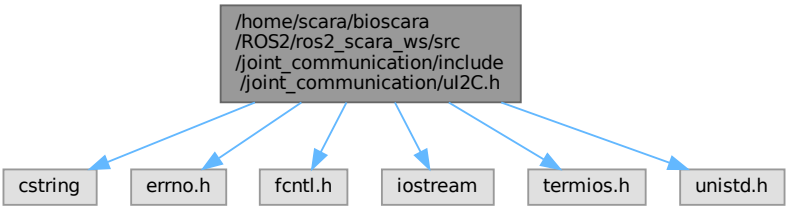
```
00001
00014 #ifndef MJOINTCOM_H
00015 #define MJOINTCOM_H
00016
00017 #include <vector>
00018 #include <iostream>
00019 #include "joint_communication/mJoint.h"
00020
00027 class Joint_comms
00028 {
00029 public:
00030   Joint_comms(void);
00031   ~Joint_comms();
00032
00041   int init(void);
00042
00050   int deinit(void);
00051
00068   void addJoint(const int address, const std::string name, const float gearRatio, const int offset);
00069
00083   int enables(std::vector<u_int8_t> driveCurrent_v, std::vector<u_int8_t> holdCurrent_v);
00084
00093   int enables(u_int8_t driveCurrent, u_int8_t holdCurrent);
00094
00101   int disables(void);
00102
00117   int home(std::string name, u_int8_t direction, u_int8_t rpm, int8_t sensitivity, u_int8_t current);
00118
00128   int getPositions(std::vector<float> &angle_v);
00129
00139   int setPositions(std::vector<float> angle_v);
```

```
00140
00150   int getVelocities(std::vector<float> &degps_v);
00151
00161   int setVelocities(std::vector<float> degps_v);
00162
00175   int checkOrientations(std::vector<float> angle_v);
00176
00182   int checkOrientations(float angle = 10.0);
00183
00191   int stops(bool mode);
00192
00197   int disableCLs(void);
00198
00207   int setDriveCurrents(std::vector<u_int8_t> current);
00208
00216   int setDriveCurrents(u_int8_t current);
00217
00225   int setHoldCurrents(std::vector<u_int8_t> current);
00226
00234   int setHoldCurrents(u_int8_t current);
00235
00243   int setBrakeModes(u_int8_t mode);
00244
00252   int enableStallguards(std::vector<u_int8_t> thresholds);
00253
00259   std::vector<Joint> joints;
00260
00261 protected:
00262 private:
00263 };
00264
00265 #endif
```

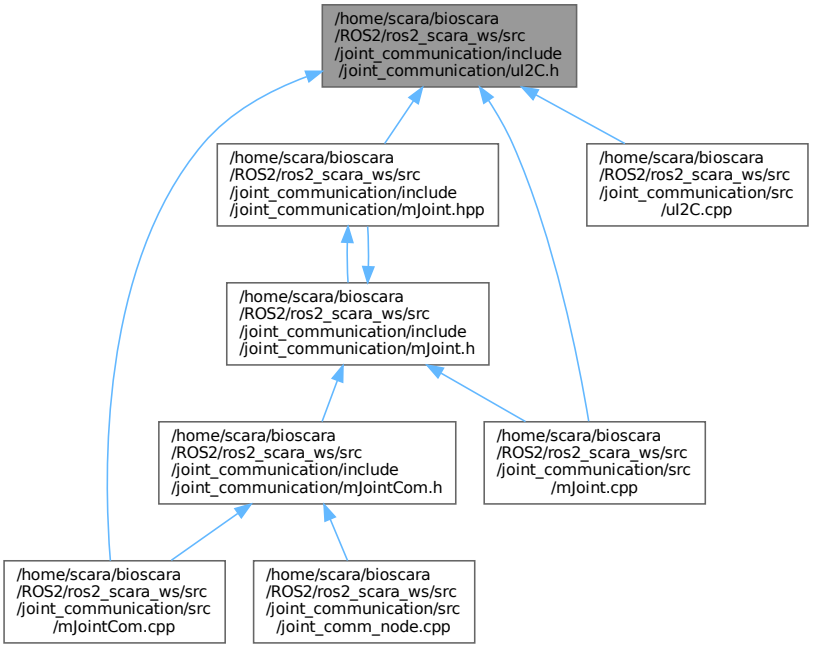
5.16 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/ul2C.h File Reference

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

Include dependency graph for ul2C.h:



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



Macros

- #define ACK 'O'
- #define NACK 'N'
- #define RFLAGS_SIZE 1
- #define MAX_BUFFER 4

Functions

- 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 (const int dev_handle)
- u_int8_t generateChecksum (const u_int8_t ∗buffer, size_t length)

5.16.1 Macro Definition Documentation

5.16.1.1 ACK

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

5.16.1.2 MAX_BUFFER

```
#define MAX_BUFFER 4
```

5.16.1.3 NACK

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

5.16.1.4 RFLAGS_SIZE

```
#define RFLAGS_SIZE 1
```

5.16.2 Function Documentation

5.16.2.1 closeI2CDevHandle()

```
int closeI2CDevHandle (
            const 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.

5.16.2.2 generateChecksum()

```
u_int8_t generateChecksum (
            const u_int8_t * buffer,
            size_t length )
```

Compute the two' complement checksum of the `buffer` according to SAE J1708

Parameters

buffer	Pointer to buffer to compute checksum off
length	Length of the buffer

Returns

Two's complement checksum.

5.16.2.3 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.

5.16.2.4 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.

5.16.2.5 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.

5.17 uI2C.h

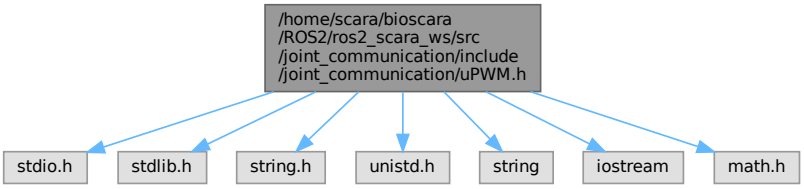
Go to the documentation of this file.

```
00001 #ifndef USERIAL_H
00002 #define USERIAL_H
00003 #include <cstring>
00004 #include <errno.h>
00005 #include <fcntl.h>
00006 #include <iostream>
00007 #include <termios.h>
00008 #include <unistd.h>
00009
00010 #define ACK 'O'
00011 #define NACK 'N'
00012 #define RFLAGS_SIZE 1
00013 #define MAX_BUFFER 4  // Bytes
00014
00015
00021 int openI2CDevHandle(const int dev_addr);
00022
00031 int readFromI2CDev(const int dev_handle, const int reg, char *buffer, const int data_length);
00032
00042 int writeToI2CDev(const int dev_handle, const int reg, char *tx_buffer, const int data_length, char
      *RFLAGS_buffer);
00043
00049 int closeI2CDevHandle(const int dev_handle);
00050
00057 u_int8_t generateChecksum(const u_int8_t *buffer, size_t length);
00058
00059 #endif
```

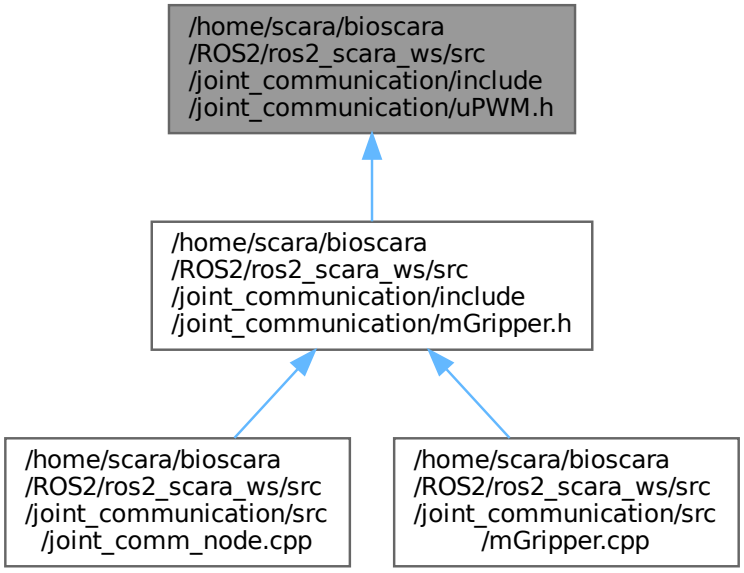
5.18 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/include/joint_communication/uPWM.h File Reference

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

Include dependency graph for uPWM.h:



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



Classes

- class RPI_PWM

5.19 uPWM.h

Go to the documentation of this file.

```
00001 /* I copied this from: https://github.com/berndporr/rpi_pwm/blob/main/rpi_pwm.h
00002 and slightly modified it*/
00003
00004 #ifndef __RPIPWM
00005 #define __RPIPWM
00006
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include <string.h>
00010 #include <unistd.h>
00011 #include <string>
00012 #include <iostream>
00013 #include <math.h>
00014
00018 class RPI_PWM
00019 {
00020 public:
00029     int start(int channel, int frequency, float duty_cycle = 0, int chip = 2)
00030     {
00031         chippath = "/sys/class/pwm/pwmchip" + std::to_string(chip);
00032         pwmpath = chippath + "/pwm" + std::to_string(channel);
00033         std::string p = chippath + "/export";
00034         FILE *const fp = fopen(p.c_str(), "w");
00035         if (NULL == fp)
00036         {
00037             std::cerr « "PWM device does not exist. Make sure to add 'dtoverlay=pwm-2chan' to
    /boot/firmware/config.txt.\n";
00038             return -1;
00039         }
00040         const int r = fprintf(fp, "%d", channel);
```

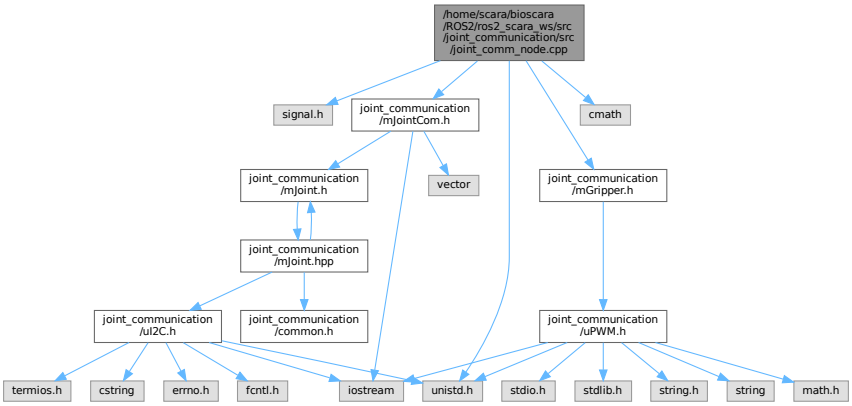
```
00041         fclose(fp);
00042         if (r < 0)
00043             return r;
00044         usleep(100000); // it takes a while till the PWM subdir is created
00045         per = (int)1E9 / frequency;
00046         setPeriod(per);
00047         setDutyCycle(duty_cycle);
00048         enable();
00049         return r;
00050     }
00051
00055     void stop()
00056     {
00057         disable();
00058     }
00059
00060     ~RPI_PWM()
00061     {
00062         disable();
00063     }
00064
00070     inline int setDutyCycle(float v) const
00071     {
00072         const int dc = (int)round((float)per * (v / 100.0));
00073         const int r = setDutyCycleNS(dc);
00074         return r;
00075     }
00076
00077 private:
00078     void setPeriod(int ns) const
00079     {
00080         writeSYS(pwmpath + "/" + "period", ns);
00081     }
00082
00083     inline int setDutyCycleNS(int ns) const
00084     {
00085         const int r = writeSYS(pwmpath + "/" + "duty_cycle", ns);
00086         return r;
00087     }
00088
00089     void enable() const
00090     {
00091         writeSYS(pwmpath + "/" + "enable", 1);
00092     }
00093
00094     void disable() const
00095     {
00096         writeSYS(pwmpath + "/" + "enable", 0);
00097     }
00098
00099     int per = 0;
00100
00101     std::string chippath;
00102     std::string pwmpath;
00103
00104     inline int writeSYS(std::string filename, int value) const
00105     {
00106         FILE *const fp = fopen(filename.c_str(), "w");
00107         if (NULL == fp)
00108         {
00109             return -1;
00110         }
00111         const int r = fprintf(fp, "%d", value);
00112         fclose(fp);
00113         return r;
00114     }
00115 };
00116
00117 #endif
```

5.20 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/src/joint_comm_node.cpp File Reference

```
#include <signal.h>
#include <unistd.h>
#include "joint_communication/mJointCom.h"
#include "joint_communication/mGripper.h"
```

```
#include <cmath>
```

Include dependency graph for joint_comm_node.cpp:



Functions

- void INT_handler (int s)
- int main (int argc, char **argv)

Variables

- Joint_comms _Joints
- Gripper _Gripper

5.20.1 Function Documentation

5.20.1.1 INT_handler()

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

5.20.1.2 main()

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

5.20.2 Variable Documentation

5.20.2.1 _Gripper

```
Gripper _Gripper
```

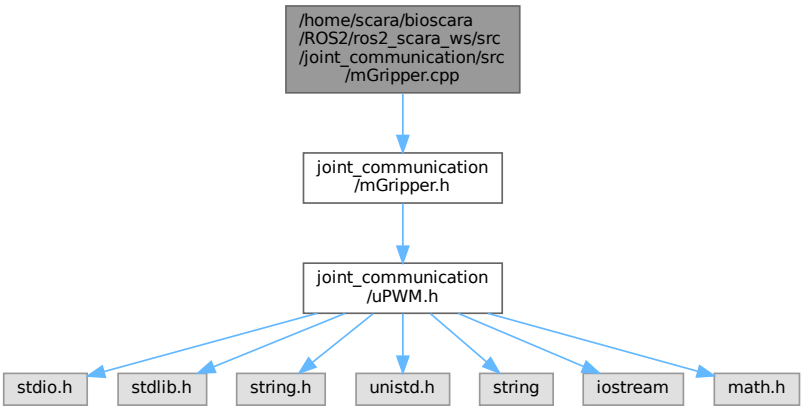
5.20.2.2 _Joints

```
Joint_comms _Joints
```

5.21 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_↩ communication/src/mGripper.cpp File Reference

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

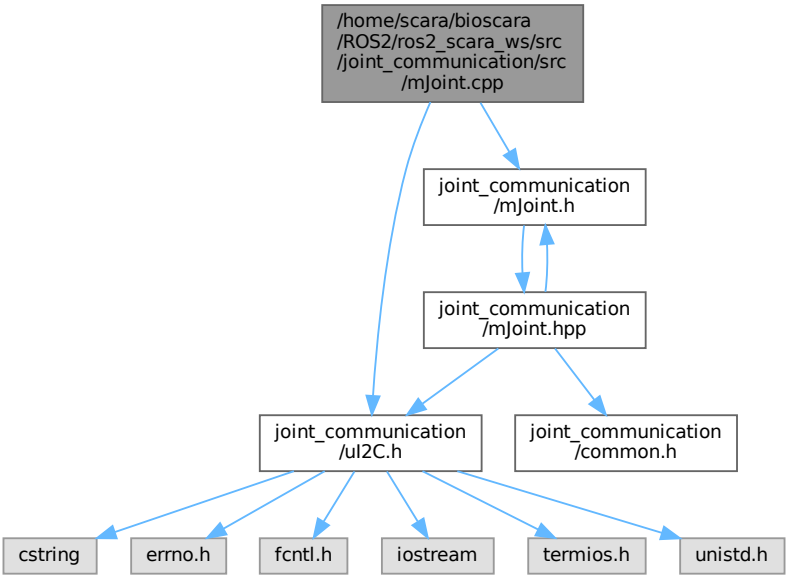
Include dependency graph for mGripper.cpp:



5.22 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_↩ communication/src/mJoint.cpp File Reference

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

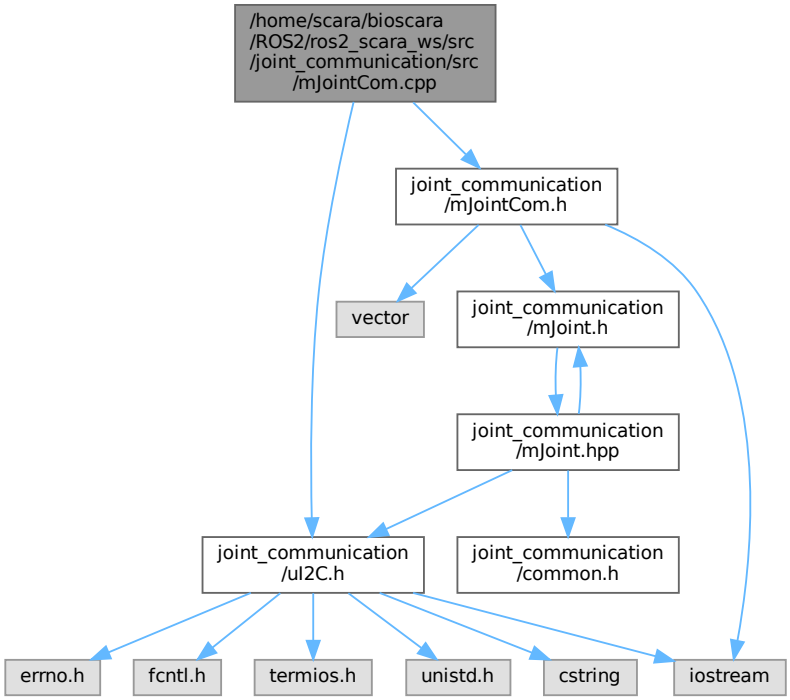
Include dependency graph for mJoint.cpp:



5.23 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/src/mJointCom.cpp File Reference

```
#include "joint_communication/uI2C.h"
#include "joint_communication/mJointCom.h"
```

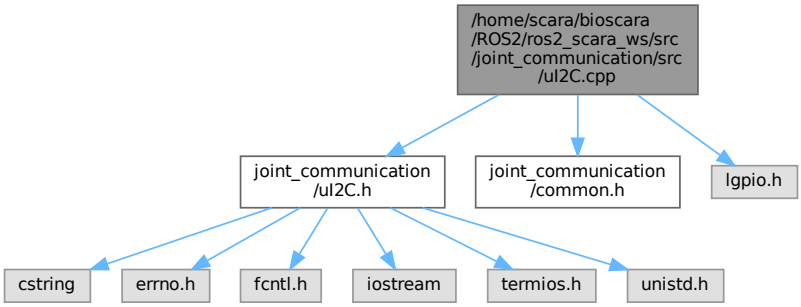
Include dependency graph for mJointCom.cpp:



5.24 /home/scara/bioscara/ROS2/ros2_scara_ws/src/joint_communication/src/uI2C.cpp File Reference

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

Include dependency graph for uI2C.cpp:



Functions

- 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 (const int dev_handle)
- u_int8_t generateChecksum (const u_int8_t *buffer, size_t length)

5.24.1 Function Documentation

5.24.1.1 closeI2CDevHandle()

```
int closeI2CDevHandle (
            const 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.

5.24.1.2 generateChecksum()

```
u_int8_t generateChecksum (
            const u_int8_t * buffer,
            size_t length )
```

Compute the two' complement checksum of the `buffer` according to SAE J1708

Parameters

buffer	Pointer to buffer to compute checksum off
length	Length of the buffer

Returns

Two's complement checksum.

5.24.1.3 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.

5.24.1.4 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.

5.24.1.5 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.

Index