RHA

Generated by Doxygen 1.8.11

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Module Documentation

6.1 Register Group

Enumerations

- enum {
  MODEL_NUMBER_L, **MODEL_NUMBER_H**, **VERSION**, **ID**,
  BAUD_RATE, **RETURN_DELAY_TIME**, **CW_ANGLE_LIMIT_L**, **CW_ANGLE_LIMIT_H**,
  CCW_ANGLE_LIMIT_L, **CCW_ANGLE_LIMIT_H**, **RESERVED1**, **LIMIT_TEMPERATURE**,
  DOWN_LIMIT_VOLTAGE, **UP_LIMIT_VOLTAGE**, **MAX_TORQUE_L**, **MAX_TORQUE_H**,
  STATUS_RETURN_LEVEL, **ALARM_LED**, **ALARM_SHUTDOWN**, **RESERVED2**,
  DOWN_CALIBRATION_L, **DOWN_CALIBRATION_H**, **UP_CALIBRATION_L**, **UP_CALIBRATION_H**,
  TORQUE_ENABLE, **LED**, **CW_COMPLIANCE_MARGIN**, **CCW_COMPLIANCE_MARGIN**,
  CW_COMPLIANCE_SLOPE, **CCW_COMPLIANCE_SLOPE**, **GOAL_POSITION_L**, **GOAL_POSITION_H**,
  MOVING_SPEED_L, **MOVING_SPEED_H**, **TORQUE_LIMIT_L**, **TORQUE_LIMIT_H**,
  PRESENT_POSITION_L, **PRESENT_POSITION_H**, **PRESENT_SPEED_L**, **PRESENT_SPEED_H**,
  PRESENT_LOAD_L, **PRESENT_LOAD_H**, **PRESENT_VOLTAGE**, **PRESENT_TEMPERATURE**,
  REGISTERED_INSTRUCTION, **RESERVE3**, **MOVING**, **LOCK**,
  PUNCH_L, **PUNCH_H** }
- enum {
  MODEL_NUMBER_L, **MODEL_NUMBER_H**, **VERSION**, **ID**,
  BAUD_RATE, **RETURN_DELAY_TIME**, **CW_ANGLE_LIMIT_L**, **CW_ANGLE_LIMIT_H**,
  CCW_ANGLE_LIMIT_L, **CCW_ANGLE_LIMIT_H**, **RESERVED1**, **LIMIT_TEMPERATURE**,
  DOWN_LIMIT_VOLTAGE, **UP_LIMIT_VOLTAGE**, **MAX_TORQUE_L**, **MAX_TORQUE_H**,
  STATUS_RETURN_LEVEL, **ALARM_LED**, **ALARM_SHUTDOWN**, **RESERVED2**,
  DOWN_CALIBRATION_L, **DOWN_CALIBRATION_H**, **UP_CALIBRATION_L**, **UP_CALIBRATION_H**,
  TORQUE_ENABLE, **LED**, **CW_COMPLIANCE_MARGIN**, **CCW_COMPLIANCE_MARGIN**,
  CW_COMPLIANCE_SLOPE, **CCW_COMPLIANCE_SLOPE**, **GOAL_POSITION_L**, **GOAL_POSITION_H**,
  MOVING_SPEED_L, **MOVING_SPEED_H**, **TORQUE_LIMIT_L**, **TORQUE_LIMIT_H**,
  PRESENT_POSITION_L, **PRESENT_POSITION_H**, **PRESENT_SPEED_L**, **PRESENT_SPEED_H**,
  PRESENT_LOAD_L, **PRESENT_LOAD_H**, **PRESENT_VOLTAGE**, **PRESENT_TEMPERATURE**,
  REGISTERED_INSTRUCTION, **RESERVE3**, **MOVING**, **LOCK**,
  PUNCH_L, **PUNCH_H** }

6.1.1 Detailed Description

Register directions in servo memory for each parameter listed

6.2 Error Group

Macros

- #define **SERROR_PING** 0X0000
- #define **SERROR_INPUTVOLTAGE** 0X0001
- #define **SERROR_ANGLELIMIT** 0X0002
- #define **SERROR_OVERHEATING** 0X0004
- #define **SERROR_RANGE** 0X0008
- #define **SERROR_CHECKSUM** 0X0010
- #define **SERROR_OVERLOAD** 0X0020
- #define **SERROR_INSTRUCTION** 0X0040
- #define **SERROR_PACKETLOST** 0X0100
- #define **SERROR_WRONGHEADER** 0X0200
- #define **SERROR_IDMISMATCH** 0X0400
- #define **SERROR_CHECKSUMERROR** 0X0800

- #define **SERROR_PING** 0X0000
- #define **SERROR_INPUTVOLTAGE** 0X0001
- #define **SERROR_ANGLELIMIT** 0X0002
- #define **SERROR_OVERHEATING** 0X0004
- #define **SERROR_RANGE** 0X0008
- #define **SERROR_CHECKSUM** 0X0010
- #define **SERROR_OVERLOAD** 0X0020
- #define **SERROR_INSTRUCTION** 0X0040
- #define **SERROR_PACKETLOST** 0X0100
- #define **SERROR_WRONGHEADER** 0X0200
- #define **SERROR_IDMISMATCH** 0X0400
- #define **SERROR_CHECKSUMERROR** 0X0800

6.2.1 Detailed Description

Defined to check error returned by servo (check as bit mask)

Chapter 7

Namespace Documentation

7.1 RHATypes Namespace Reference

Classes

- class FuzzyRegulator
- class FuzzyRegulatorNode
- struct Point3
- class Regulator

  Implements a standard PID regulator.
- struct SpeedGoal

  Data structure to store speed goal (servo id to send the goal, speed target, speed slope and direction to move).
- class Timer
- class TimerMicroseconds

7.1.1 Detailed Description

: Enrique Heredia Aguado <enheragu> : 22-Dec-2017 : RHA modified by: enheragu modified time: 22-Dec-2017

: Enrique Heredia Aguado <quique> : 17-Sep-2017 : RHA modified by: quique modified time: 29-Sep-2017

Chapter 8

Class Documentation

8.1 __freelist Struct Reference

Collaboration diagram for __freelist:

Public Attributes

- size_t **sz**
- struct __freelist ∗ **nx**

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

- lib/memory_free/MemoryFree.cpp

8.2 activateTimer Class Reference

```
#include <rha_types.h>
```

8.2.1 Detailed Description

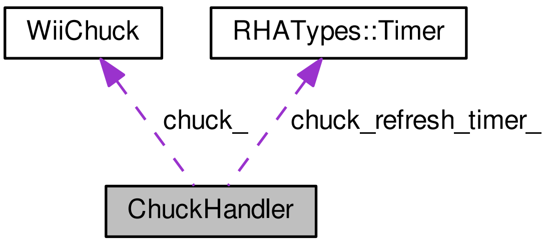
Implements the timer but in microseconds

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

- lib/rha_types/rha_types.h

8.3 ChuckHandler Class Reference

Collaboration diagram for ChuckHandler:



Public Member Functions

- void **begin** ()
- void setTimer (uint64_t timer)

  Sets period of chuck read axis refresh setTimer.
- void **printChuckValues** ()
- ChuckReadStruct readAxis ()

  Reads values from chuck and returns an X,Y,Z speed readAxis.

Protected Attributes

- WiiChuck **chuck_**
- RHATypes::Timer **chuck_refresh_timer_**

8.3.1 Member Function Documentation

8.3.1.1 ChuckReadStruct ChuckHandler::readAxis ( ) [inline]

Reads values from chuck and returns an X,Y,Z speed readAxis.

Returns

ChuckReadStruct is a struct with speed values in it (X,Y,Z) from -100 to 100 (direction and module)

8.3.1.2 void ChuckHandler::setTimer (uint64_t *timer*) [inline]

Sets period of chuck read axis refresh setTimer.

Parameters

timer	period in ms

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

- lib/chuck_handler/chuck_handler.h

8.4 ChuckReadStruct Struct Reference

Public Member Functions

- **ChuckReadStruct** (int _x=0, int _y=0, int _z=0, bool _updated=false)

Public Attributes

- int **X_**
- int **Y_**
- int **Z_**
- bool **updated_**

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

- lib/chuck_handler/chuck_handler.h

8.5 Cytron_G15_Servo Class Reference

Public Member Functions

- **Cytron_G15_Servo** (uint8_t servo_id, uint8_t rxpin, uint8_t txpin, uint8_t ctrlpin)
- **Cytron_G15_Servo** (uint8_t rxpin, uint8_t txpin, uint8_t ctrlpin)
- **Cytron_G15_Servo** (uint8_t ctrlpin)
- virtual void **init** (uint8_t servo_id, uint8_t rxpin, uint8_t txpin, uint8_t ctrlpin, uint32_t baudrate)
- void **init** (uint8_t servo_id, uint8_t rxpin, uint8_t txpin, uint8_t ctrlpin)
- virtual void **begin** (uint32_t baudrate)
- void **end** (void)
- uint16_t **setWheelMode** (void)
- uint16_t **exitWheelMode** (void)
- uint16_t **setWheelSpeed** (uint16_t speed, uint8_t direction, uint8_t Write_Reg)
- uint16_t **setPos** (uint16_t position, uint8_t Write_Reg)
- uint16_t **setPosAngle** (uint16_t angle, uint8_t Write_Reg)
- uint16_t **setPosSpeed** (uint16_t position, uint16_t speed, uint8_t Write_Reg)
- uint16_t **rotateCW** (uint16_t position, uint8_t Write_Reg)
- uint16_t **rotateCCW** (uint16_t position, uint8_t Write_Reg)
- uint16_t **setTorqueOnOff** (uint8_t onOff, uint8_t Write_Reg)
- uint16_t **setSpeed** (uint16_t speed, uint8_t Write_Reg)
- uint16_t **setTimeToGoal** (uint16_t time, uint8_t Write_Reg)

- uint16_t **setAngleLimit** (uint16_t cwAngle, uint16_t ccwAngle)
- uint16_t **setTorqueLimit** (uint16_t torqueLimit)
- uint16_t **setTemperatureLimit** (uint8_t temperature)
- uint16_t **setVoltageLimit** (uint8_t voltageLow, uint8_t voltageHigh)
- uint16_t **setID** (uint8_t newID)
- uint16_t **setLED** (uint8_t onOff, uint8_t Write_Reg)
- uint16_t **setAlarmLED** (uint8_t alarmLED)
- uint16_t **setAlarmShutDown** (uint8_t alarm)
- uint16_t **setMarginSlopePunch** (uint8_t CWMargin, uint8_t CCWMargin, uint8_t CWSlope, uint8_t CCW↩ Slope, uint16_t punch)
- uint16_t **setBaudRate** (uint32_t baudrate)
- uint16_t **factoryReset** ()
- uint16_t **ping** (uint8_t ∗data)
- virtual uint16_t **getPos** (uint8_t ∗data)
- virtual uint16_t **getSpeed** (uint8_t ∗data)
- virtual uint16_t **getLoad** (uint8_t ∗data)
- virtual uint16_t **getVoltage** (uint8_t ∗data)
- virtual uint16_t **getTemperature** (uint8_t ∗data)
- virtual uint16_t **getTorqueOnOff** (uint8_t ∗data)
- virtual uint16_t **isMoving** (uint8_t ∗data)
- void **setAction** (void)
- uint16_t **sendPacket** (uint8_t id, uint8_t instruction, uint8_t ∗data, uint8_t parameterLength)

Public Attributes

- uint8_t **servo_id_**

Protected Member Functions

- void **setRxMode** (void)
- void **setTxMode** (void)

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

- lib/cytron_g15_servo/cytron_g15_servo.h.txt
- lib/cytron_g15_servo/cytron_g15_servo.cpp.txt

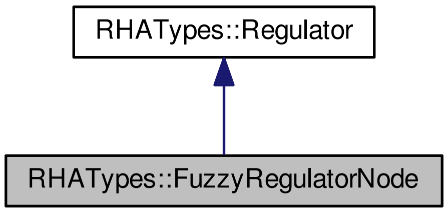
8.6 RHATypes::FuzzyRegulator Class Reference

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

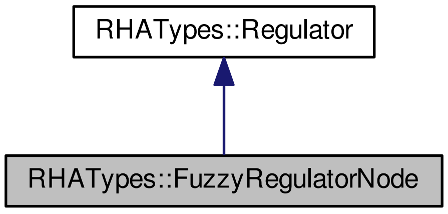
- lib/rha_types/fuzzy_regulator.h

8.7 RHATypes::FuzzyRegulatorNode Class Reference

Inheritance diagram for RHATypes::FuzzyRegulatorNode:



Collaboration diagram for RHATypes::FuzzyRegulatorNode:



Public Member Functions

- void resetRegulator ()

  Resets all regulator data to 0 resetRegulator.
- void setRegulator (float _aplication_point, float _kp, float _ki=0, float _kd=0)
- float regulator (float _error, float _derror=0, float _ierror=0)

  Calculates output of regulator to a set error regulator.
- float **getAplicationPoint** ()

8.7.1 Member Function Documentation

8.7.1.1 float RHATypes::FuzzyRegulatorNode::regulator (float *_error,* float *_derror* = 0, float *_ierror* = 0) [inline], [virtual]

Calculates output of regulator to a set error regulator.

Parameters

_error	error
_derror	derivative error
_ierror	integral error

Returns

returns output of regulator

Reimplemented from RHATypes::Regulator.

8.7.1.2 void RHATypes::FuzzyRegulatorNode::setRegulator (float *_aplication_point*, float *_kp*, float *_ki* = 0, float *_kd* = 0) [inline]

Sets Fuzzy regulator node constants and aplication point setRegulator

Parameters

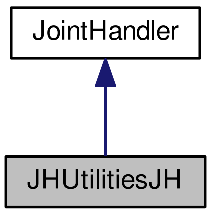
_aplication_point	Is the point in which this regulator is used
_kp	Proportional K
_ki	Integral K
_kd	Derivative K

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

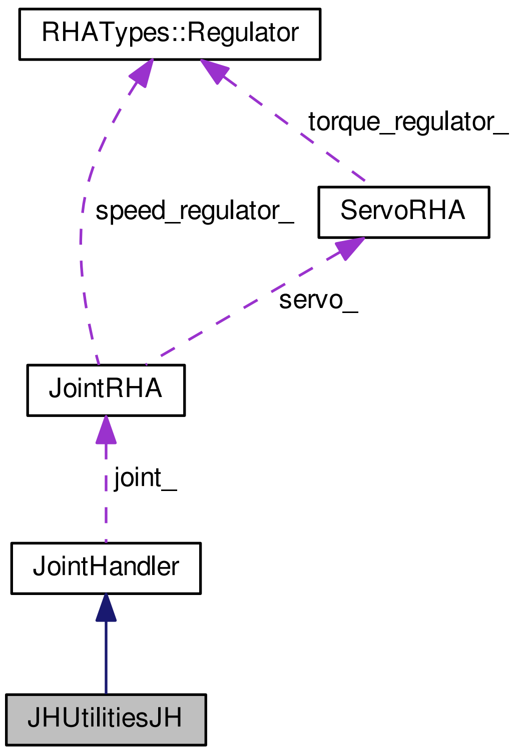
- lib/rha_types/fuzzy_regulator.h

8.8 JHUtilitiesJH Class Reference

Inheritance diagram for JHUtilitiesJH:



Collaboration diagram for JHUtilitiesJH:



Public Member Functions

- void **initJoints** (uint8_t _joint_to_test)
- void extractRegulatorData (uint8_t _joint_to_test)

  extractRegulatorData tests ServoRHA regulator and prints through serial monitor all info (python list style) to make a graphic. It can test one servo. Autodetects ID of the one connected extractRegulatorData
- void **extractStepSlopeData** (uint8_t __joint_to_test, uint8_t _option)
- void checkTimeGetInfo (uint8_t repetitions, uint8_t _joint_to_test)

  checkTimeInfo checks time spent sending and recieving packet with ServoRHA::updateInfo() . It can test one servo. Autodetects ID of the one connected
- void checkComSucces (uint16_t repetitions)

  This function is intended to test new baudrates and it's success comunication ratio checkPingSucces.
- void checkSpeed (uint8_t _joint_to_test)

  checkSpeed implements an encoder mode to measure real speed in RPM and check agains the measure returned by servo and torque value sent. It can test one servo. Autodetects ID of the one connected
- void **resetEncoder** ()
- void **updateEncoder** (uint8_t _joint_to_test)
- void **startEncoder** (uint8_t _joint_to_test)
- void **returnToStartPositionTest** (uint8_t _joint_to_test, uint8_t direction)

Additional Inherited Members

8.8.1 Member Function Documentation

8.8.1.1 void JHUtilitiesJH::checkComSucces (uint16_t *repetitions*)

This function is intended to test new baudrates and it's success comunication ratio checkPingSucces.

Parameters

repetitions	number of repetitions to perform

8.8.1.2 void JHUtilitiesJH::checkTimeGetInfo (uint8_t *repetitions,* uint8_t *_joint_to_test*)

checkTimeInfo checks time spent sending and recieving packet with ServoRHA::updateInfo() . It can test one servo. Autodetects ID of the one connected

Parameters

{long}	repetitions: num of repetitions the test is made (time is the average of this repetitions). Max of 255 (danger of memory overload)

See also

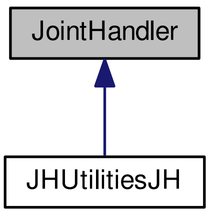
checkTimeSpeedRead(). Both are used together to compare speed rate in comunication.
averageChauvenet()

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

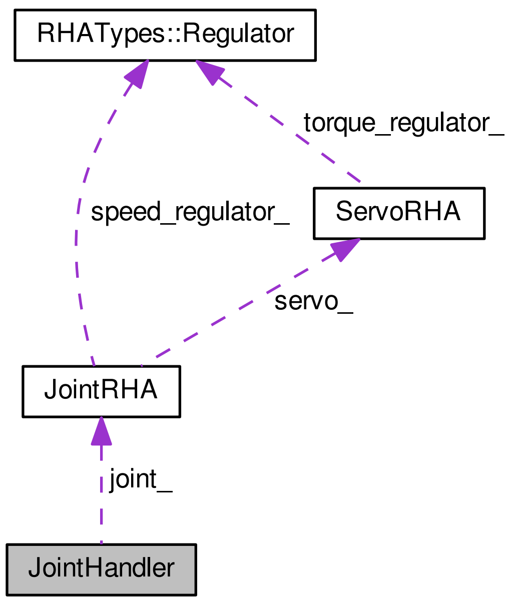
- lib/utilities/utilities.h
- lib/utilities/utilities.cpp

8.9 JointHandler Class Reference

Inheritance diagram for JointHandler:



Collaboration diagram for JointHandler:



Public Member Functions

- void **printCheckVar** ()
- JointHandler (uint64_t timer)

    Constructor with timer info JointHandler::JointHandler.
- virtual void initJoints ()

    Initialices joints with ID, up direction and potentiometer pin. Sets wheel mode to all servo JointHandler::initJoints.
- void setSpeedGoal (RHATypes::SpeedGoal _goal)

    Sets speed goal to a given joint (based on servo ID in goal) JointHandler::setSpeedGoal.
- void setTorqueControlTimer (uint64_t timer)

    Initialices timer from control_loop for speed in ServoRHA JointHandler::init.
- void setSpeedControlTimer (uint64_t timer)

    Initialices timer from control_loop for position in JointRHA JointHandler::init.
- virtual void controlLoopTorque ()

    controlLoopTorque() function handles control loop for servo speed (output of regulator is torque for servo)
- virtual void controlLoopSpeed ()

    controlLoopSpeed() function handles control loop for joint position (output of regulator is speed for ServoRHA) Joint←Handler::updateJointInfo
- bool checkJointSecurityAll ()

    Checks that is secure to move all joints checkJointSecurityAll.
- bool checkServoSecurityAll ()

    Checks that is secure to move all servo checkJointSecurityAll.
- void updateJointInfo ()

    Updates internal information from all joint. JointHandler::updateJointInfo.
- void updateJointErrorTorque ()

Updates all joints error to update torque goal JointHandler::updateJointErrorTorque.

- void sendJointTorques ()

  handles the packet construction and sent for all joint torques (torque goal saved in each servo) JointHandler::send↩ JointTorques
- void updateJointErrorSpeed ()

  Updates all joint position error to update speed goal sor ServoRHA JointHandler::updateJointErrorSpeed.
- void sendSpeedGoalAll ()

  Sends speed goal calculated to ServoRHA JointHandler::sendSpeedGoalAll.
- void sendSetWheelModeAll ()

  Sets wheel mode for all servo JointHandler::sendSetWheelModeAll.
- void sendExitWheelModeAll ()

  Exit wheel mode for all servo JointHandler::sendExitWheelModeAll.
- void sendSetTorqueLimitAll (uint16_t _torque_limit)

  Sets torque limit to all servo JointHandler::sendSetTorqueLimitAll.
- void sendSetWheelSpeedAll (uint16_t _speed=0, uint8_t _direction=0)

  Interface to send speed (in wheel mode this means torque) to servos. Params are by default 0, if this is the case servos work in a closed control loop, if not they use the speed set. JointHandler::sendSetWheelSpeedAll.
- void setReturnPacketOption (uint8_t _option)

  Sets retunr packet option for all joints JointHandler::setReturnPacketOption.
- bool checkConectionAll ()

  checks if can conect with all servo JointHandler::checkConectionAll
- uint8_t addToSyncPacket (uint8_t ∗_buffer, uint8_t ∗_data, uint8_t _num_bytes)

  Adds data to common buffer. Intended to put commands from all servo into one packet JointHandler::addToSync↩ Packet.
- void warpSyncPacket (uint8_t ∗_buffer, uint8_t _adress, uint8_t ∗_txBuffer, uint8_t _num_bytes, uint8_t _↩ num_servo)

  wrapPacket adds information needed once all servos had been aded (header, ID, instruction...). This function is used to send just one packet for all servos instead of each sending their respective information JointHandler::warpSync↩ Packet
- void warpSinglePacket (uint8_t _instruction, uint8_t ∗_buffer, uint8_t ∗_txBuffer)

  Warps packet info with the information needed for the comunication JointHandler::warpSinglePacket.
- uint16_t sendPacket (uint8_t ∗_buffer)

  Function to send to bus information contained in buffer param. Contains logic to read data in case it is needed JointHandler::sendPacket.
- bool isError ()
- **JointHandler** (uint8_t _rxpin, uint8_t _txpin, uint8_t _ctrlpin)
- **JointHandler** (uint8_t _ctrlpin)
- void initSerial ()

  Constructor with custom software serial. JointHandler::JointHandler.
- void begin ()

  Cpnfigures comunication at a set baudrate. Sets the serial port JointHandler::begin.
- void **end** (void)
- void **setTxMode** (void)
- void **setRxMode** (void)
- void **resetBuffer** (uint8_t buffer[ ])

Public Attributes

- JointRHA **joint_** [NUM_JOINT]

8.9.1 Member Function Documentation

8.9.1.1 uint8_t JointHandler::addToSyncPacket (uint8_t ∗ *_buffer,* uint8_t ∗ *_data,* uint8_t *_num_bytes*)

Adds data to common buffer. Intended to put commands from all servo into one packet JointHandler::addToSync←
Packet.

Parameters

{uint8←↩ _t	∗} buffer array to write all the info
{uint8←↩ _t	∗} data contains data to copy
{uint8←↩ _t}	num_bytes number of bytes tha have been written

Returns

returns length copied in bytes

8.9.1.2 void JointHandler::begin ( )

Cpnfigures comunication at a set baudrate. Sets the serial port JointHandler::begin.

Parameters

baudrate	Baudrate in which to communicate

8.9.1.3 bool JointHandler::checkConectionAll ( )

checks if can conect with all servo JointHandler::checkConectionAll

Returns

retunrs true if the conection with all servo was succesfull. Returns false if failed with any of them

8.9.1.4 bool JointHandler::checkJointSecurityAll ( )

Checks that is secure to move all joints checkJointSecurityAll.

Returns

returns true in case of safety, ralse otherwise

8.9.1.5 bool JointHandler::checkServoSecurityAll ( )

Checks that is secure to move all servo checkJointSecurityAll.

Returns

returns true in case of safety, ralse otherwise

8.9.1.6 void JointHandler::initSerial ( )

Constructor with custom software serial. JointHandler::JointHandler.

Parameters

rxpin	RX pin for serial comunication
txpin	TX pin for serial comunication
ctrlpin	control pin for serial comunication Constructor with default hardwareSerial (RX in pin 0, TX in pin 1) and with set control pin JointHandler::JointHandler
ctrlpin	control pin for serial comunication Method to set serial data (rx, tx, ctrlpin and baudrate) to init communication JointHandler::initSerial
rxpin	RX pin for serial comunication
txpin	TX pin for serial comunication
ctrlpin	control pin for serial comunication
baudrate	Baudrate in which to communicate

See also

JointHandler::begin()

8.9.1.7 bool JointHandler::isError ( ) [inline]

Returns whether theres any error or not

8.9.1.8 uint16_t JointHandler::sendPacket (uint8_t ∗ *_txBuffer*)

Function to send to bus information contained in buffer param. Contains logic to read data in case it is needed JointHandler::sendPacket.

Parameters

{uint8←_t	∗} buffer array with all the information to send, if info is read it will be copied here

Returns

error in comunication

8.9.1.9 void JointHandler::sendSetTorqueLimitAll (uint16_t *_torque_limit*)

Sets torque limit to all servo JointHandler::sendSetTorqueLimitAll.

Parameters

torque_limit	torque limit to set

8.9.1.10 void JointHandler::sendSetWheelSpeedAll (uint16_t _speed = 0, uint8_t _direction = 0)

Interface to send speed (in wheel mode this means torque) to servos. Params are by default 0, if this is the case servos work in a closed control loop, if not they use the speed set. JointHandler::sendSetWheelSpeedAll.

Parameters

speed	speed/torque to set
direction	direction CW (clocwise) or CCW (counterclockwise)

8.9.1.11 void JointHandler::setReturnPacketOption (uint8_t _option)

Sets retunr packet option for all joints JointHandler::setReturnPacketOption.

Parameters

_option	[description]

8.9.1.12 void JointHandler::setSpeedControlTimer (uint64_t _timer)

Initialices timer from control_loop for position in JointRHA JointHandler::init.

Parameters

timer	time in ms for control loop

8.9.1.13 void JointHandler::setSpeedGoal (RHATypes::SpeedGoal _goal)

Sets speed goal to a given joint (based on servo ID in goal) JointHandler::setSpeedGoal.

Parameters

goal	Goal containing speed, speed_slope and ID

8.9.1.14 void JointHandler::setTorqueControlTimer (uint64_t _timer)

Initialices timer from control_loop for speed in ServoRHA JointHandler::init.

Parameters

timer	time in ms for control loop

8.9.1.15 void JointHandler::updateJointInfo ( )

Updates internal information from all joint. JointHandler::updateJointInfo.

See also

JointRHA::updateJointInfo

8.9.1.16 void JointHandler::warpSinglePacket (uint8_t *_instruction*, uint8_t ∗ *_buffer*, uint8_t ∗ *_txBuffer*)

Warps packet info with the information needed for the comunication JointHandler::warpSinglePacket.

Parameters

instruction	Instruction of how to access servo register (iREAD_DATA, iREG_WRITE, iWRITE_DATA...)
{uint8_t	∗} buffer data to warp
{uint8_t	∗} txBuffer data warped and ready to send

See also

JointHandler::sendPacket()

8.9.1.17 void JointHandler::warpSyncPacket (uint8_t ∗ *_buffer*, uint8_t *_adress*, uint8_t ∗ *_txBuffer*, uint8_t *_num_bytes*, uint8_t *_num_servo*)

wrapPacket adds information needed once all servos had been aded (header, ID, instruction...). This function is used to send just one packet for all servos instead of each sending their respective information JointHandler↩ ::warpSyncPacket

Parameters

{uint8↩ _t	∗} buffer is the data that have been completed by each servo (by reference)
{uint8↩ _t}	adress Direction of servo register in which to write/read...
{uint8↩ _t	∗} txBuffer data warped and ready to send
{uint8↩ _t}	num_bytes is the length of data
{uint8↩ _t}	num_servo how many servos had been added to this packet

See also

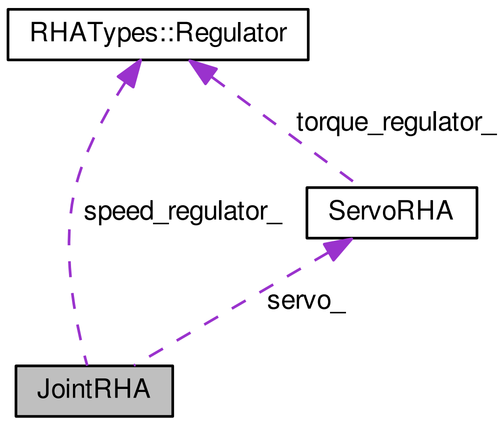
JointHandler::sendPacket()

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

- lib/joint_handler/joint_handler.h
- lib/joint_handler/joint_handler.cpp

8.10 JointRHA Class Reference

Collaboration diagram for JointRHA:



Public Member Functions

- void **printCheckVar** ()
- JointRHA (uint8_t _servo_id, uint8_t _up_direction, uint8_t _potentiometer=NO_POTENTIOMETER)

  Cunstructor of JointRHA class.
- ∼JointRHA ()

  ∼JointRHA destructor of JointRHA class.
- void init (uint8_t _servo_id, uint8_t _up_direction, float _zero_compensation=0, uint8_t _potentiometer=N↩O_POTENTIOMETER)

  Initialization for JointRHA default constructor.
- void setPotRelation (float _relation=1)
- void **initPotMeasurment** (uint32_t _pot_min_value, uint32_t _pot_max_value, uint8_t _angle_min_value, uint8_t _angle_max_value)
- uint8_t **setSpeedGoal** (RHATypes::SpeedGoal _goal)
- void updatePosition ()

  Updates position reading from potentiometer if there is a pot to read (not 255). Updates joint angle position JointR↩HA::updatePosition.
- void updateInfo (uint8_t ∗_data, uint16_t _error)

  Updates all the information of servo object information and position feedback of joint to use it in next control iteration (in control loop) JointRHA::updateInfo.
- void setPositionGoal (int _position)

  Sets a goal position for this joint setPositionGoal.
- void posError ()

  Calculates error to send to servo regulator.
- void calculateSpeed (float _error=0, float _derror=0, float _ierror=0)

  calculates speed from pos error using regulator. Params are by default 0, it is only used with params for testing pourposes JointRHA::calculateTorque
- void updateServoSpeedGoal ()

Updates ServoRHA speed goal JointRHA::updateServoSpeedGoal.

- bool checkSecurity ()

  checks that everithing goes as espected. If not it stops the servo checkSecurity

- bool reachedGoalPosition ()

  returns true if goal position is reached JointRHA::reachedGoalPosition

- float **getPosition** ()
- float **getGoalSpeed** ()
- int **getPosTarget** ()
- float **getError** ()
- float **getDError** ()
- float **getIError** ()
- int **getAnalogReadPot** ()
- int **getPotentiometerPin** ()

Public Attributes

- RHATypes::Regulator **speed_regulator_**
- ServoRHA **servo_**

8.10.1 Constructor & Destructor Documentation

8.10.1.1 JointRHA::JointRHA (uint8_t *_servo_id*, uint8_t *_up_direction*, uint8_t *_potentiometer* = NO_POTENTIOMETER)

Cunstructor of JointRHA class.

Parameters

{uint8↩_t}	servo_id servo id controlled by this joint
{uint8↩_t}	up_direction direction in which the servo has to move (CW or CCW) so the joint moves up.
{uint8↩_t}	potentiometer pin in which the potentiometer for this joint is connected. If there is no realim for this joint value will be 255

8.10.2 Member Function Documentation

8.10.2.1 void JointRHA::calculateSpeed (float *_error* = 0, float *_derror* = 0, float *_ierror* = 0)

calculates speed from pos error using regulator. Params are by default 0, it is only used with params for testing pourposes JointRHA::calculateTorque

Parameters

error	pos error
derror	derivative of pos error
ierror	integral of pos error

8.10.2.2 bool JointRHA::checkSecurity ( )

checks that everithing goes as espected. If not it stops the servo checkSecurity

Returns

Returns true when theres no problem, false otherwise

8.10.2.3 void JointRHA::init (uint8_t *_servo_id,* uint8_t *_up_direction,* float *_zero_compensation =* `0`*,* uint8_t *_potentiometer =* `NO_POTENTIOMETER`)

Initialization for JointRHA default constructor.

Parameters

{uint8↩_t}	servo_id servo id controlled by this joint
{uint8↩_t}	up_direction direction in which the servo has to move (CW or CCW) so the joint moves up.
{uint8↩_t}	potentiometer pin in which the potentiometer for this joint is connected. If there is no realim for this joint value will be 255

8.10.2.4 bool JointRHA::reachedGoalPosition ( )

returns true if goal position is reached JointRHA::reachedGoalPosition

Returns

[description]

8.10.2.5 void JointRHA::setPositionGoal (int *_position*)

Sets a goal position for this joint setPositionGoal.

Parameters

position	position to go

8.10.2.6 void JointRHA::setPotRelation (float *_relation =* `1`)

Sets the relation between the potentiometer angle (in grads) and the joint angle JointRHA::setPotRelation

Parameters

_relation	relation between measures. diameter of pot gear / diameter of bar gear

8.10.2.7 void JointRHA::updateInfo (uint8_t ∗ *_data,* uint16_t *_error*)

Updates all the information of servo object information and position feedback of joint to use it in next control iteration (in control loop) JointRHA::updateInfo.

Parameters

{uint8↩ _t	∗} data data with servo information to pass to it
{uint16↩ _t	∗} error error in communication with servo

8.10.2.8 void JointRHA::updatePosition ( )

Updates position reading from potentiometer if there is a pot to read (not 255). Updates joint angle position Joint↩ RHA::updatePosition.

Returns

returns position value in joint reference

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

- lib/joint_rha/joint_rha.h
- lib/joint_rha/joint_rha.cpp

8.11 RHATypes::Point3 Struct Reference

Public Member Functions

- **Point3** (uint8_t _x, int16_t _y, int16_t _z)

Public Attributes

- float **x**
- float **y**
- float **z**

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

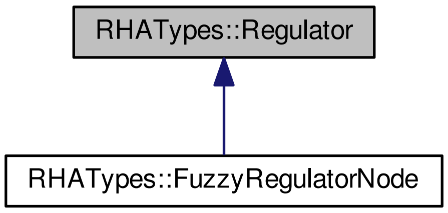
- lib/rha_types/rha_types.h

8.12 RHATypes::Regulator Class Reference

Implements a standard PID regulator.

```
#include <pid_regulator.h>
```

Inheritance diagram for RHATypes::Regulator:



Public Member Functions

- virtual void resetRegulator ()

  Resets all regulator data to 0 resetRegulator.
- virtual void setKRegulator (float _kp, float _ki=0, float _kd=0)
- virtual float regulator (float _error, float _derror=0, float _ierror=0)

  Calculates output of regulator to a set error regulator.
- float **getKp** ()
- float **getKi** ()
- float **getKd** ()

8.12.1 Detailed Description

Implements a standard PID regulator.

8.12.2 Member Function Documentation

8.12.2.1 virtual float RHATypes::Regulator::regulator (float *_error*, float *_derror* = 0, float *_ierror* = 0) [inline], [virtual]

Calculates output of regulator to a set error regulator.

Parameters

_error	error
_derror	derivative error
_ierror	integral error

Returns

returns output of regulator

Reimplemented in RHATypes::FuzzyRegulatorNode.

8.12.2.2 virtual void RHATypes::Regulator::setKRegulator (float *_kp*, float *_ki* = 0, float *_kd* = 0) [inline], [virtual]

Sets PID regulator constants setKRegulator

Parameters

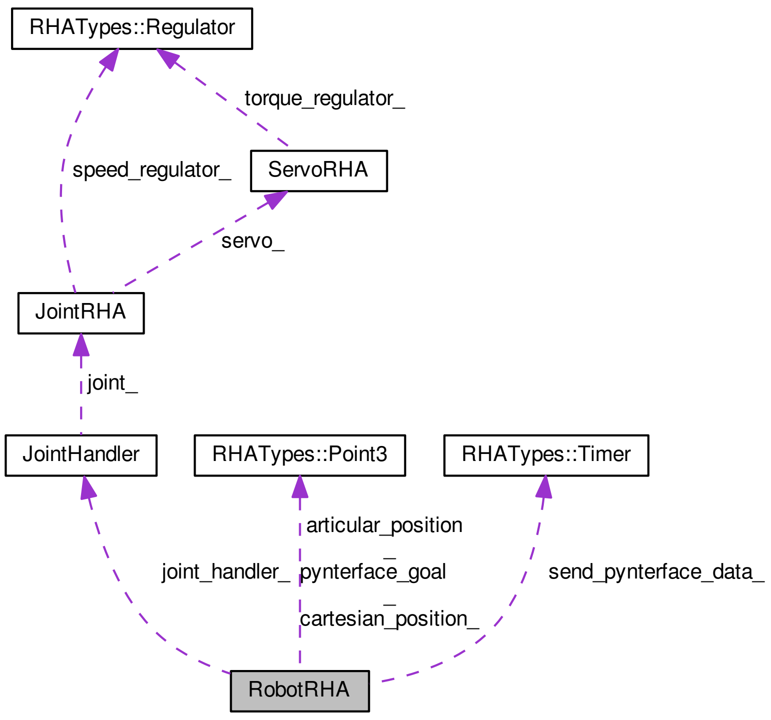
_kp	Proportional K
_ki	Integral K
_kd	Derivative K

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

- lib/rha_types/pid_regulator.h

8.13 RobotRHA Class Reference

Collaboration diagram for RobotRHA:



Public Member Functions

- void initJointHandler ()
  Inits Joint handler timer, serial and joints RobotRHA::initJointHandler.
- void **initChuckHandler** ()
- void **handleRobot** ()
- void setCartesianSpeedGoal (float _speed_x, float _speed_y, float _speed_z)
  Inits chuck handler timer RobotRHA::initChuckHandler.
- void setSpeedToServos (float _speed, uint8_t _servo_id)
  Sends speed goal RobotRHA::setSpeedToServos.
- void **initPynterface** ()
- void **handleWithPynterface** ()
- bool **sendPackage** ()
- void **getPackage** ()
- void **handleWithChuck** ()
- void handleWithSerialPort ()
  Handles all GDL with nunchuck input. This method calls chuck reading ,sets speed goals and calls Joint handler controlLoop RobotRHA::handleWithChuck.
- int getGoalFromSerialInput (int _joint_target)
  Asks a position to go through the serial port and returns it RobotRHA::getGoalFromSerialInput.
- void **updateInfo** ()
- void **goToCartesianPos** (RHATypes::Point3 _cartesian_pos)
- void **goToArticularPos** (RHATypes::Point3 _articular_pos)
- bool **checkError** ()
- RHATypes::Point3 **getCartesianPos** ()
- RHATypes::Point3 **forwardKinematics** (RHATypes::Point3 _articular_pos)
- RHATypes::Point3 **inverseKinematics** (RHATypes::Point3 _cartesian_pos)
- bool **isError** ()

Public Attributes

- RHATypes::Timer **send_pynterface_data_**
- JointHandler **joint_handler_**
- RHATypes::Point3 **articular_position_**
- RHATypes::Point3 **cartesian_position_**
- RHATypes::Point3 **pynterface_goal_**

8.13.1 Member Function Documentation

8.13.1.1 int RobotRHA::getGoalFromSerialInput (int *_joint_target*)

Asks a position to go through the serial port and returns it RobotRHA::getGoalFromSerialInput.

Parameters

_joint_target	joint for which the goal is intended

8.13.1.2 void RobotRHA::handleWithSerialPort ( )

Handles all GDL with nunchuck input. This method calls chuck reading ,sets speed goals and calls Joint handler controlLoop RobotRHA::handleWithChuck.

Handles all GDL with serial port input. Ask Goal position for all GDL and then goes to it RobotRHA::handleWith↩ SerialPort

8.13.1.3 void RobotRHA::setCartesianSpeedGoal (float *_speed_x,* float *_speed_y,* float *_speed_z*)

Inits chuck handler timer RobotRHA::initChuckHandler.

Sets X, Y, X speed to Joints applyin transformation RobotRHA::setCartesianSpeedGoal

Parameters

_speed↩ _x	speed in X
_speed↩ _y	speed in Y
_speed↩ _z	speed in Z

8.13.1.4 void RobotRHA::setSpeedToServos (float *_speed,* uint8_t *_servo_id*)

Sends speed goal RobotRHA::setSpeedToServos.

Parameters

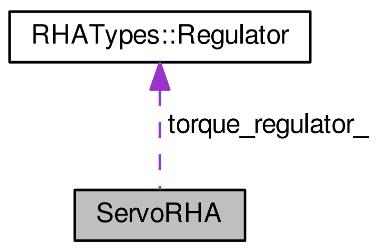
_speed	speed to set
_servo↩ _id	target servo for this goal speed

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

- lib/robot_rha/robot_rha.h
- lib/robot_rha/robot_rha.cpp

8.14 ServoRHA Class Reference

Collaboration diagram for ServoRHA:



Public Member Functions

- void **printCheckVar** ()
- ServoRHA (uint8_t _servo_id)

    Constructor of ServoRHA class.
- void init (uint8_t _servo_id)

    Handles the inicialization of all ServoRHA internal parameters when default constructor is used.
- void **init** ()
- void updateInfo (uint8_t ∗_data, uint16_t _error)

    Asks the servo for all the information to be updated by class servo. ServoRHA::updateInfo.
- void addReturnOptionToPacket (uint8_t ∗_buffer, uint8_t _option)

    Saves in buffer the package return level of servo (error information for each command sent) ServoRHA::addReturn←OptionToPacket.
- void addUpadteInfoToPacket (uint8_t ∗_buffer)

    adds to buffer packet with the uptade info command ServoRHA::addUpadteInfoToPacket
- bool addTorqueToPacket (uint8_t ∗_buffer)

    Adds this servo torque command to a buffer with his own information. This function is used to send just one packet for all servos instead of each sending their respective information ServoRHA::addTorqueToPacket.
- void setTorqueOnOfToPacket (uint8_t ∗_buffer, uint8_t _onOff)

    Adds to buffer information about the torque option (on or off) ServoRHA::setTorqueOnOfToPacket.
- void setWheelModeToPacket (uint8_t ∗_buffer)

    Adds to buffer information to set wheel mode for servo ServoRHA::setWheelModeToPacket.
- void exitWheelModeToPacket (uint8_t ∗_buffer)

    Adds to buffer information to exit wheel mode for servo ServoRHA::exitWheelModeToPacket.
- void wheelModeToPacket (uint8_t ∗_buffer, uint16_t _CW_angle, uint16_t _CCW_angle)

    Adds to buffer information to set/exit wheel mode for servo. Common function for exit and set functions.
- void addToPacket (uint8_t ∗_buffer, uint8_t ∗_packet, uint8_t _packet_len)

    addToPacket adds this servo to a buffer with his own information (id, goal, etc). This function is used to send just one packet for all servos instead of each sending their respective information
- void pingToPacket (uint8_t ∗_buffer)

    Arranges data array to ping action ServoRHA::pingToPacket.
- bool checkSecurity ()

checks that everithing goes as espected. If not it stops the servo checkSecurity

- uint8_t setSpeedGoal (RHATypes::SpeedGoal _goal)

  Sets speed goal to achieve with speed slope.
- void speedError ()

  Calculates error to send to servo regulator ServoRHA::speedError.
- void calculateTorque (float _error=0, float _derror=0, float _ierror=0)

  calculates torque from speed error using regulator. Params are by default 0, it is only used with params for testing pourposes ServoRHA::calculateTorque
- void setTorqueLimitToPacket (uint8_t ∗_buffer, uint16_t _torque_limit)

  Arranges data packet with torque limit ServoRHA::setTorqueLimitToPacket.
- void setWheelSpeedToPacket (uint8_t ∗_buffer, uint16_t _torque_limit, uint8_t _direction)

  Makes packet with speed goal with set direction ServoRHA::setWheelSpeedToPacket.
- virtual uint8_t **getID** ()
- virtual float **getSpeed** ()
- virtual uint16_t **getSpeedDir** ()
- virtual uint16_t **getPosition** ()
- virtual uint16_t **getLoad** ()
- virtual uint16_t **getLoadDir** ()
- virtual uint16_t **getGoalTorque** ()
- uint16_t **getSpeedTarget** ()
- uint8_t **getDirectionTarget** ()
- float **getError** ()
- float **getDError** ()
- float **getIError** ()
- uint16_t **getSpeedWithDir** ()
- uint16_t **getTorqueWithDir** ()

Public Attributes

- RHATypes::Regulator **torque_regulator_**

Protected Attributes

- uint8_t **empty_var**
- uint8_t **empty_var_2**
- volatile uint8_t **servo_id_**
- uint16_t **speed_dir_**
- uint16_t **position_**
- uint16_t **load_**
- uint16_t **load_dir_**
- uint16_t **error_comunication_**
- float **speed_**
- uint16_t **goal_torque_**
- uint8_t **direction_target_**
- uint16_t **speed_target_**
- uint64_t **time_last_**
- uint64_t **time_last_error_**
- float **error_**
- float **last_error_**
- float **derror_**
- float **ierror_**

8.14.1 Constructor & Destructor Documentation

8.14.1.1 ServoRHA::ServoRHA (uint8_t *servo_id*) `[explicit]`

Constructor of ServoRHA class.

Parameters

{uint8↩ _t}	servo_id servo id controlled by this object

8.14.2 Member Function Documentation

8.14.2.1 void ServoRHA::addReturnOptionToPacket (uint8_t ∗ *_buffer,* uint8_t *_option*)

Saves in buffer the package return level of servo (error information for each command sent) ServoRHA::add↩ ReturnOptionToPacket.

Parameters

{uint8_t∗}	buffer array in which add the information
{uint8_t}	option RETURN_PACKET_ALL -> servo returns packet for all commands sent; RETURN_PACKET_NONE -> servo never retunrs state packet; RETURN_PACKET_READ_INSTRUCTIONS -> servo answer packet state when a READ command is sent (to read position, temperature, etc)

See also

addToPacket()

8.14.2.2 void ServoRHA::addToPacket (uint8_t ∗ *_buffer,* uint8_t ∗ *_packet,* uint8_t *_packet_len*)

addToPacket adds this servo to a buffer with his own information (id, goal, etc). This function is used to send just one packet for all servos instead of each sending their respective information

Parameters

{uint8↩ _t	∗} buffer is the buffer in which the information will be added (by reference)
{uint8↩ _t	∗} packet small packet to add. Note that it can be speed, torque, position... It can be a combination (go to an X position with an Y speed) (by reference)
{uint8↩ _t}	packet_len length of the small packet to add (uint8_ts)

8.14.2.3 bool ServoRHA::addTorqueToPacket (uint8_t ∗ *_buffer*)

Adds this servo torque command to a buffer with his own information. This function is used to send just one packet for all servos instead of each sending their respective information ServoRHA::addTorqueToPacket.

Parameters

{uint8↩ _t	∗} buffer is the buffer in which the information will be added (by reference)

See also

addToPacket()

8.14.2.4 void ServoRHA::addUpadteInfoToPacket (uint8_t * *_buffer*)

adds to buffer packet with the uptade info command ServoRHA::addUpadteInfoToPacket

Parameters

{uint8_t}*	buffer array in which add the information

8.14.2.5 void ServoRHA::calculateTorque (float *_error* = 0, float *_derror* = 0, float *_ierror* = 0)

calculates torque from speed error using regulator. Params are by default 0, it is only used with params for testing pourposes ServoRHA::calculateTorque

Parameters

error	speed error
derror	derivative of speed error
ierror	integral of speed error

8.14.2.6 bool ServoRHA::checkSecurity ( )

checks that everithing goes as espected. If not it stops the servo checkSecurity

Returns

Returns true when theres no problem, false otherwise

8.14.2.7 void ServoRHA::exitWheelModeToPacket (uint8_t * *_buffer*)

Adds to buffer information to exit wheel mode for servo ServoRHA::exitWheelModeToPacket.

Parameters

buffer	array in which add the information

See also

setWheelModeToPacket()
wheelModeToPacket()

8.14.2.8 void ServoRHA::init (uint8_t *_servo_id*)

Handles the inicialization of all ServoRHA internal parameters when default constructor is used.

Parameters

{uint8↩ _t}	servo_id servo id controlled by this object

8.14.2.9 void ServoRHA::pingToPacket (uint8_t ∗ *_buffer*)

Arranges data array to ping action ServoRHA::pingToPacket.

Parameters

buffer	Array in which to store the data

8.14.2.10 uint8_t ServoRHA::setSpeedGoal (RHATypes::SpeedGoal *_goal*)

Sets speed goal to achieve with speed slope.

Parameters

{uint16↩ _t}	speed_target speed to achieve
{uint16↩ _t}	direction_target move CW or CCW

8.14.2.11 void ServoRHA::setTorqueLimitToPacket (uint8_t ∗ *_buffer,* uint16_t *_torque_limit*)

Arranges data packet with torque limit ServoRHA::setTorqueLimitToPacket.

Parameters

buffer	Array in which to store the data
torque_limit	Torque limit to set

8.14.2.12 void ServoRHA::setTorqueOnOfToPacket (uint8_t ∗ *_buffer,* uint8_t *_onOff*)

Adds to buffer information about the torque option (on or off) ServoRHA::setTorqueOnOfToPacket.

Parameters

buffer	array in which add the information
onOff	ON = 1; OFF = 0;

8.14.2.13 void ServoRHA::setWheelModeToPacket (uint8_t ∗ *_buffer*)

Adds to buffer information to set wheel mode for servo ServoRHA::setWheelModeToPacket.

Parameters

buffer	array in which add the information

See also

exitWheelModeToPacket()
wheelModeToPacket()

8.14.2.14 void ServoRHA::setWheelSpeedToPacket (uint8_t ∗ *_buffer,* uint16_t *_speed,* uint8_t *_direction*)

Makes packet with speed goal with set direction ServoRHA::setWheelSpeedToPacket.

Parameters

buffer	Array in which to store the data
speed	Speed to set
direction	Direction in which servo will move

8.14.2.15 void ServoRHA::updateInfo (uint8_t ∗ *_data,* uint16_t *_error*)

Asks the servo for all the information to be updated by class servo. ServoRHA::updateInfo.

Parameters

{uint8_t	∗} data Array containing all the data
{uint16←_t}	error Error in comunication

Reads from register PRESENT_POSITION_L (0x24) to MOVING (0x2E). Position are bits 10 to 0 from register 0x24 and 0x25 Speed are bits 9 to 0 from register 0x26 and 0x27, 10th bit is direction Load are bits 9 to 0 from register 0x28 and 0x29, 10th bit is direction

To avoid spending too much time the following parameter have been commented as they are not used

Voltage is in register 0x2a Temperature is in register 0x2B Action registered (pending from activation) flag is in register 0x2C Moving flag is in register 0x2E

8.14.2.16 void ServoRHA::wheelModeToPacket (uint8_t ∗ *_buffer,* uint16_t *_CW_angle,* uint16_t *_CCW_angle*)

Adds to buffer information to set/exit wheel mode for servo. Common function for exit and set functions.

Parameters

buffer	array in which add the information
CW_angle	cw angle limit
CCW_angle	ccw angle limit

See also

setWheelModeToPacket()
exitWheelModeToPacket()

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

- lib/servo_rha/servo_rha.h
- lib/servo_rha/servo_rha.cpp

8.15 setTimer Class Reference

```
#include <rha_types.h>
```

8.15.1 Detailed Description

Class which implements a timer

Parameters

time_set	time duration

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

- lib/rha_types/rha_types.h

8.16 RHATypes::SpeedGoal Struct Reference

Data structure to store speed goal (servo id to send the goal, speed target, speed slope and direction to move).

```
#include <rha_types.h>
```

Public Member Functions

- **SpeedGoal** (uint8_t _id, int16_t _speed, uint8_t _direction)

Public Attributes

- uint8_t **servo_id**
- int16_t **speed**
- uint8_t **direction**

8.16.1 Detailed Description

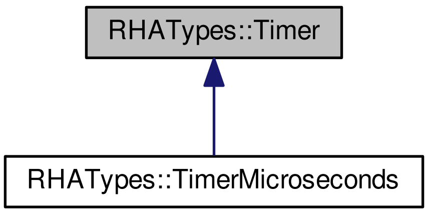
Data structure to store speed goal (servo id to send the goal, speed target, speed slope and direction to move).

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

- lib/rha_types/rha_types.h

8.17 RHATypes::Timer Class Reference

Inheritance diagram for RHATypes::Timer:



Public Member Functions

- void **setTimer** (uint32_t _time_set)
- virtual void activateTimer ()

    Activates the timer, it starts counting time activateTimer.
- virtual void checkWait ()

    Checks if time was reached. If no, it pauses the execution and waits for it to finish checkWait.
- virtual bool checkContinue ()

    Checks if time was reached. If not it returns a false and does not block the execution. checkContinue.
- uint64_t getInitTime ()

    Interface method to get time in which timer was activated getInitTime.

Protected Attributes

- uint32_t **time_**
- uint64_t **init_time_**

8.17.1 Member Function Documentation

8.17.1.1 virtual bool RHATypes::Timer::checkContinue ( ) [inline], [virtual]

Checks if time was reached. If not it returns a false and does not block the execution. checkContinue.

Returns

Returns true or false if it reached the time or not

Reimplemented in RHATypes::TimerMicroseconds.

8.17.1.2 uint64_t RHATypes::Timer::getInitTime ( ) [inline]

Interface method to get time in which timer was activated getInitTime.

Returns

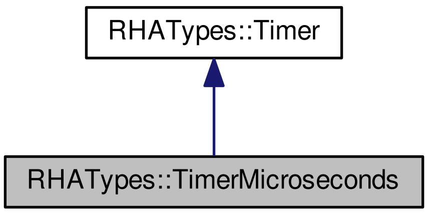
Returns last activation time

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

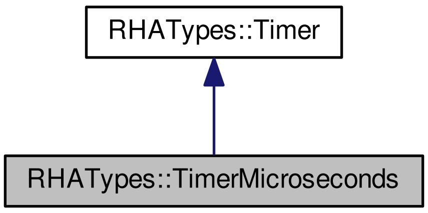
- lib/rha_types/rha_types.h

8.18 RHATypes::TimerMicroseconds Class Reference

Inheritance diagram for RHATypes::TimerMicroseconds:



Collaboration diagram for RHATypes::TimerMicroseconds:



Public Member Functions

- virtual void activateTimer ()
- virtual void checkWait ()
- virtual bool checkContinue ()

Additional Inherited Members

8.18.1 Member Function Documentation

8.18.1.1 virtual void RHATypes::TimerMicroseconds::activateTimer ( ) [inline], [virtual]

activateTimer

See also

Timer::activateTimer

Reimplemented from RHATypes::Timer.

8.18.1.2 virtual bool RHATypes::TimerMicroseconds::checkContinue ( ) [inline], [virtual]

checkContinue

See also

Timer::activateTimer

Reimplemented from RHATypes::Timer.

8.18.1.3 virtual void RHATypes::TimerMicroseconds::checkWait ( ) [inline], [virtual]

checkWait

See also

Timer::activateTimer

Reimplemented from RHATypes::Timer.

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

- lib/rha_types/rha_types.h

8.19 WiiChuck Class Reference

Public Member Functions

- void **begin** ()
- void **calibrateJoy** ()
- void **update** ()
- float **readAccelX** ()
- float **readAccelY** ()
- float **readAccelZ** ()
- bool **zPressed** ()
- bool **cPressed** ()
- bool **rightJoy** (int thresh=60)
- bool **leftJoy** (int thresh=60)
- int **readJoyX** ()
- int **readJoyY** ()
- int **readRoll** ()
- int **readPitch** ()

Public Attributes

- uint8_t **joyX**
- uint8_t **joyY**
- bool **buttonZ**
- bool **buttonC**

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

- lib/chuck_handler/WiiChuck.h

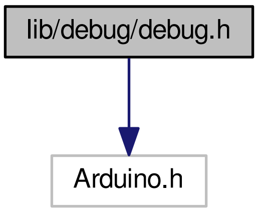
Chapter 9

File Documentation

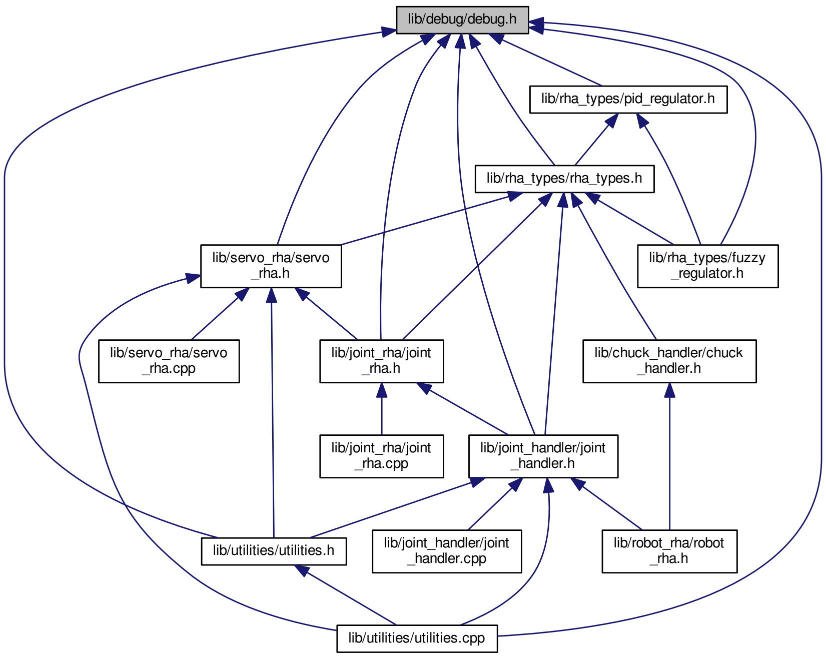
9.1 lib/debug/debug.h File Reference

Implements debugging macros with Serial printig that can be activated or not for each different librari or file.

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



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


lib/debug/debug.h
lib/rha_types/pid_regulator.h
lib/rha_types/rha_types.h
lib/servo_rha/servo _rha.h
lib/rha_types/fuzzy _regulator.h
lib/servo_rha/servo _rha.cpp
lib/joint_rha/joint _rha.h
lib/chuck_handler/chuck _handler.h
lib/joint_rha/joint _rha.cpp
lib/joint_handler/joint _handler.h
lib/utilities/utilities.h
lib/joint_handler/joint _handler.cpp
lib/robot_rha/robot _rha.h
lib/utilities/utilities.cpp

Macros

- #define **SERROR_PING** 0X0000
- #define **SERROR_INPUTVOLTAGE** 0X0001
- #define **SERROR_ANGLELIMIT** 0X0002
- #define **SERROR_OVERHEATING** 0X0004
- #define **SERROR_RANGE** 0X0008
- #define **SERROR_CHECKSUM** 0X0010
- #define **SERROR_OVERLOAD** 0X0020
- #define **SERROR_INSTRUCTION** 0X0040
- #define **SERROR_PACKETLOST** 0X0100
- #define **SERROR_WRONGHEADER** 0X0200
- #define **SERROR_IDMISMATCH** 0X0400
- #define **SERROR_CHECKSUMERROR** 0X0800
- #define **DEBUG_ROBOT_RHA**
- #define **PRINT_SERVO_ERROR_MSG** false
- #define DebugSerialG15Ln(a)
- #define **DebugSerialG15Ln2**(a, b)
- #define **DebugSerialG15Ln4**(a, b, c, d)
- #define DebugSerialSRHALn(a)
- #define **DebugSerialSRHALn2**(a, b)
- #define **DebugSerialSRHALn4**(a, b, c, d)
- #define **DebuSerialRHALnPrintServoStatus**(pos, speed, speed_dir, load, load_dir, voltage, temperature, error)
- #define DebugSerialJRHALn(a)
- #define **DebugSerialJRHALn2**(a, b)
- #define **DebugSerialJRHALn4**(a, b, c, d)

- #define DebugSerialJHLn(a)
- #define **DebugSerialJHLn2**(a, b)
- #define **DebugSerialJHLn4**(a, b, c, d)
- #define **DebugSerialJHLn4Error**(a, b) { printServoStatusError(a, b); }
- #define DebugSerialUtilitiesLn(a)
- #define **DebugSerialUtilitiesLn2**(a, b)
- #define **DebugSerialUtilities**(a)
- #define **DebugSerialUtilitiesLn4**(a, b, c, d)
- #define DebugSerialRHATypesLn(a)
- #define **DebugSerialRHATypesLn2**(a, b)
- #define **DebugSerialRHATypes**(a)
- #define **DebugSerialRHATypesLn4**(a, b, c, d)
- #define DebugSerialSeparation(a) { Serial.println("#===
  }
- #define DebugSerialRRHALn(a) { Serial.print(F("[DC] ROBOT_RHA::")); Serial.println(a); }
- #define **DebugSerialRRHALn2**(a, b) { Serial.print(F("[DC] ROBOT_RHA::")); Serial.print(a); Serial.println(b);
  }
- #define **DebugSerialRRHAn4**(a, b, c, d) { Serial.print(F("[DC] ROBOT_RHA::")); Serial.print(a); Serial.↩
  print(b); Serial.print(c); Serial.println(d); }
- #define DebugSerialTG15Ln(a)
- #define **DebugSerialTG15**(a)
- #define DebugSerialTSRHALn(a)
- #define **DebugSerialTSRHA**(a)
- #define DebugSerialTJRHALn(a)
- #define **DebugSerialTJRHALn2**(a, b)
- #define **DebugSerialTJRHALn4**(a, b, c, d)
- #define **DebugSerialTJRHA**(a)

Functions

- void printServoStatusError (uint16_t error, uint8_t ID)

  Analyses error and prints error msgs.
- void **printServoStatus** (uint16_t pos, uint16_t speed, uint8_t speed_dir, uint16_t load, uint8_t load_dir, uint8_t voltage, uint8_t temperature, uint16_t error)

9.1.1 Detailed Description

Implements debugging macros with Serial printig that can be activated or not for each different librari or file.

Each set of macros has a define option, if it's been defined all debugging options in the set will be printed. If it's not defined Debug commands en that file will be ignored. Each set has different macros which admit different number of parameters to print.

: Enrique Heredia Aguado <enheragu> : 2017_Sep_08 : RHA : debug.h modified by: quique modified time: 29-Oct-2017

9.1.2 Macro Definition Documentation

9.1.2.1 #define DebugSerialG15Ln(*a*)

DEBUG_CYTRON_G15_SERVO implements debug macros for cytron_g15_servo.h and .cpp files

9.1.2.2 #define DebugSerialJHLn(*a*)

DEBUG_JOINT_HANDLER implements debug macros for servo_rha.h and .cpp files

9.1.2.3 #define DebugSerialJRHALn(*a*)

DEBUG_JOINT_RHA implements debug macros for joint_rha.h and .cpp files

9.1.2.4 #define DebugSerialRHATypesLn(*a*)

DEBUG_RHA_TYPES implements debug macros for rha_types.h file

9.1.2.5 #define DebugSerialRRHALn(*a*) { Serial.print(F("[DC] ROBOT_RHA::")); Serial.println(a); }

DEBUG_ROBOT_RHA implements debug macros for robot_rha.h and .cpp files

9.1.2.6 #define DebugSerialSeparation(*a*) { Serial.←println("#==#"); }

DebugSerialSeparation prints a horizontal line to separate different set of debug information

9.1.2.7 #define DebugSerialSRHALn(*a*)

DEBUG_SERVO_RHA implements debug macros for servo_rha.h and .cpp files

9.1.2.8 #define DebugSerialTG15Ln(*a*)

DEBUG_TEST_CYTRON_G15_SERVO implements debug macros for test_cytron_g15_servo.cpp file

9.1.2.9 #define DebugSerialTJRHALn(*a*)

DEBUG_TEST_SERVO_RHA_REAL implements debug macros for test_servo_real.cpp file

9.1.2.10 #define DebugSerialTSRHALn(*a*)

DEBUG_TEST_SERVO_RHA_MOCK implements debug macros for test_servo_mock.cpp file

9.1.2.11 #define DebugSerialUtilitiesLn(*a*)

DEBUG_UTILITIES implements debug macros for utilities.h file

9.1.3 Function Documentation

9.1.3.1 void printServoStatusError (uint16_t *error,* uint8_t *ID*)

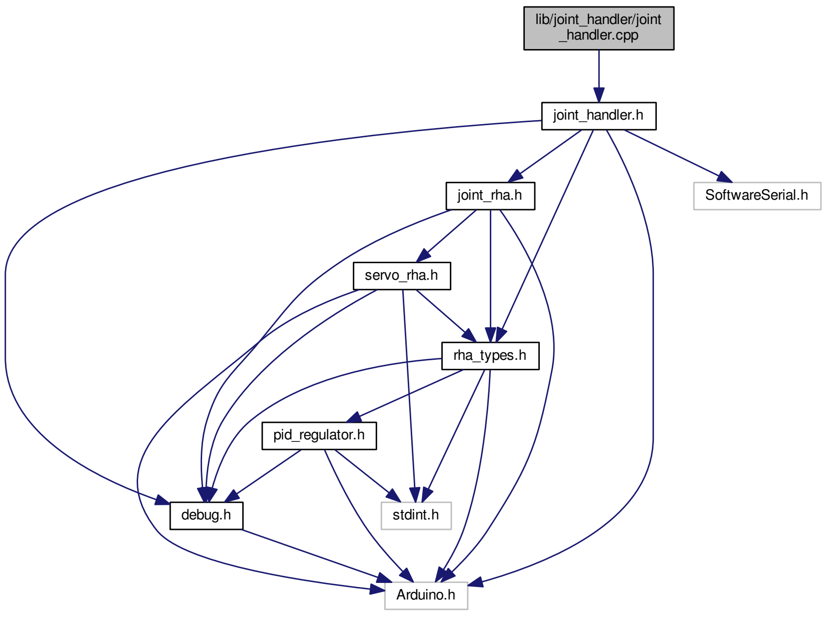
Analyses error and prints error msgs.

: Enrique Heredia Aguado <quique> : 21-Sep-2017 : RHA modified by: quique modified time: 28-Sep-2017

9.2 lib/joint_handler/joint_handler.cpp File Reference

Implements JointHandler functions defined in joint_handler.h.

```
#include "joint_handler.h"
```
Include dependency graph for joint_handler.cpp:



Variables

- boolean **hardwareSerial_** = false
- SoftwareSerial ∗ **G15Serial_**

9.2.1 Detailed Description

Implements JointHandler functions defined in joint_handler.h.

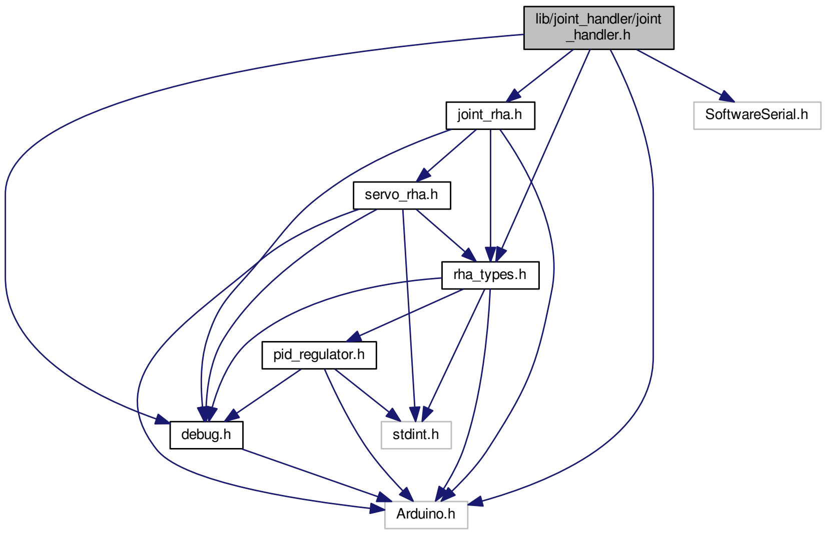
: Enrique Heredia Aguado <enheragu> : 2017_Sep_08 : RHA : joint_handler.cpp modified by: enheragu modified time: 31_Oct_2017

9.3 lib/joint_handler/joint_handler.h File Reference

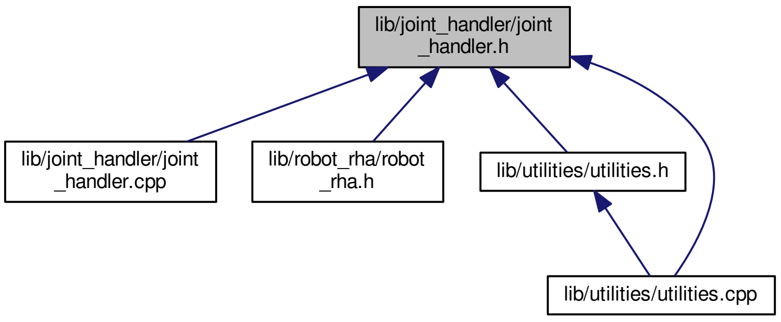
Implements JointHandler class. This object is in charge to sync all joints.

```
#include "debug.h"
#include "rha_types.h"
#include "joint_rha.h"
#include <SoftwareSerial.h>
#include "Arduino.h"
```

Include dependency graph for joint_handler.h:



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



Classes

- class JointHandler

Macros

- #define **__AVR_ATmega1280__**
- #define **Serial_G15_lib** Serial2
- #define **CHECK_MEGA_HARDWARESERIAL**(rx, tx) (rx == 17 && tx == 16)
- #define **iPING** 0x01
- #define **iREAD_DATA** 0x02
- #define **iWRITE_DATA** 0x03
- #define **iREG_WRITE** 0x04
- #define **iACTION** 0x05
- #define **iRESET** 0x06
- #define **iSYNC_WRITE** 0x83
- #define **SerialTimeOut** 100L
- #define **TxMode** LOW
- #define **RxMode** HIGH
- #define **ConvertAngleToPos**(angle) (uint16_t)((uint16_t)(angle) $*$ 1088UL / 360UL)
- #define **ConvertPosToAngle**(position) static_cast<float>((position) $*$ 360.0 / 1088.0)
- #define **ConvertTime**(time) (uint16_t)(time $*$ 10UL)
- #define ALL_SERVO 0xFE
- #define **DEFAULT_ID** 0x01
- #define **G15_BAUDRATE** 460800
- #define **NUM_JOINT** 3
- #define **BUFFER_LEN** 20
- #define **EEMPROM_WRITE_DELAY** 25
- #define **G15_BAUDRATE** 460800
- #define **G15_RX_PIN** 17
- #define **G15_TX_PIN** 16
- #define **G15_CONTRL_PIN** 8

9.3.1 Detailed Description

Implements JointHandler class. This object is in charge to sync all joints.

: Enrique Heredia Aguado <enheragu> : 2017_Sep_08 : RHA : joint_handler.h modified by: enheragu modified time: 31_Oct_2017

9.3.2 Macro Definition Documentation

9.3.2.1 #define ALL_SERVO 0xFE

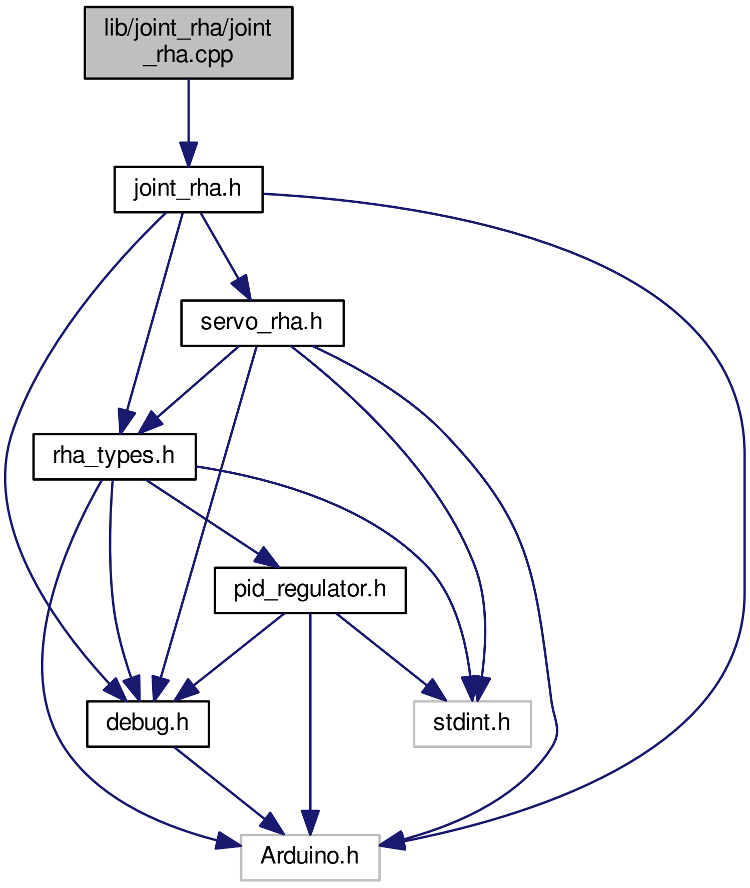
ALL_SERVO is ID to broadcast to all servo in bus.

9.4 lib/joint_rha/joint_rha.cpp File Reference

Implements JointRHA functions defined in joint_rha.h : Enrique Heredia Aguado <enheragu> : 2017_Sep_08 : RHA : joint_rha.cpp modified by: quique modified time: 29-Sep-2017.

```
#include "joint_rha.h"
```

Include dependency graph for joint_rha.cpp:



9.4.1 Detailed Description

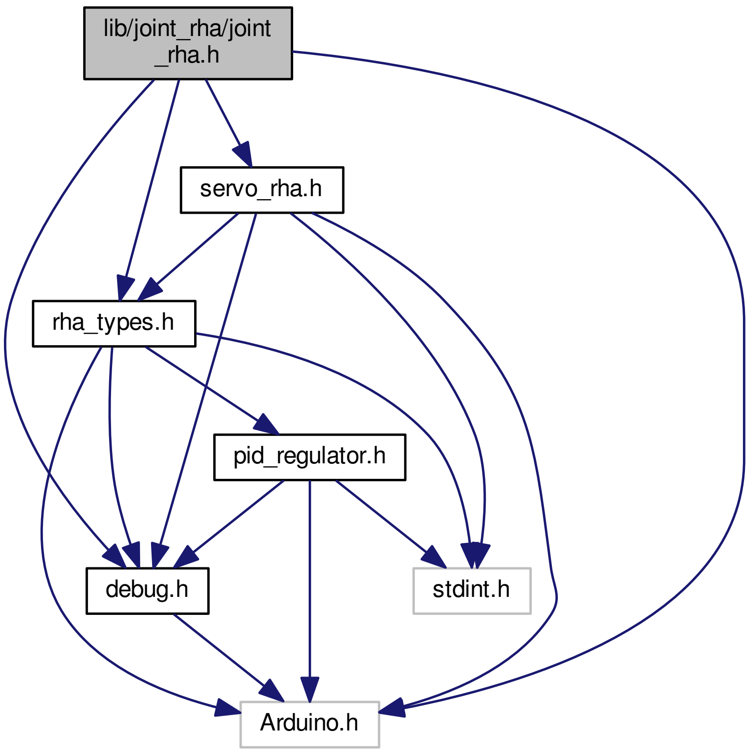
Implements JointRHA functions defined in joint_rha.h : Enrique Heredia Aguado <enheragu> : 2017_Sep_08 : RHA : joint_rha.cpp modified by: quique modified time: 29-Sep-2017.

9.5 lib/joint_rha/joint_rha.h File Reference

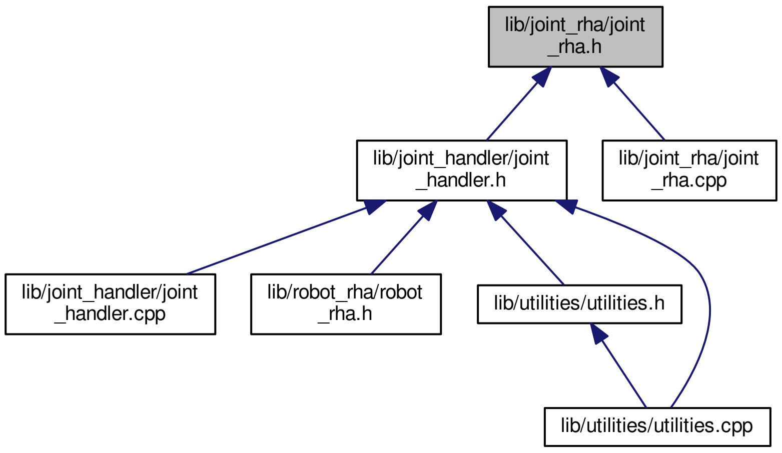
Implements JointRHA class. This object combines potentiometer with ServoRHA object readings to enhance it's functionality.

```
#include "servo_rha.h"
#include "rha_types.h"
#include "debug.h"
#include "Arduino.h"
```

Include dependency graph for joint_rha.h:



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


lib/joint_rha/joint _rha.h
lib/joint_handler/joint _handler.h
lib/joint_rha/joint _rha.cpp
lib/joint_handler/joint _handler.cpp
lib/robot_rha/robot _rha.h
lib/utilities/utilities.h
lib/utilities/utilities.cpp

Classes

- class JointRHA

Macros

- #define **ANGLE_TOLERANCE** 4
- #define **NO_POTENTIOMETER** 255
- #define ERROR_MOVING_MARGIN 5
- #define **KP** 0.35
- #define **KD** 1
- #define **KI** 1

9.5.1 Detailed Description

Implements JointRHA class. This object combines potentiometer with ServoRHA object readings to enhance it's functionality.

: Enrique Heredia Aguado <enheragu> : 2017_Sep_08 : RHA : joint_rha.h modified by: quique modified time: 29-Sep-2017

9.5.2 Macro Definition Documentation

9.5.2.1 #define ERROR_MOVING_MARGIN 5

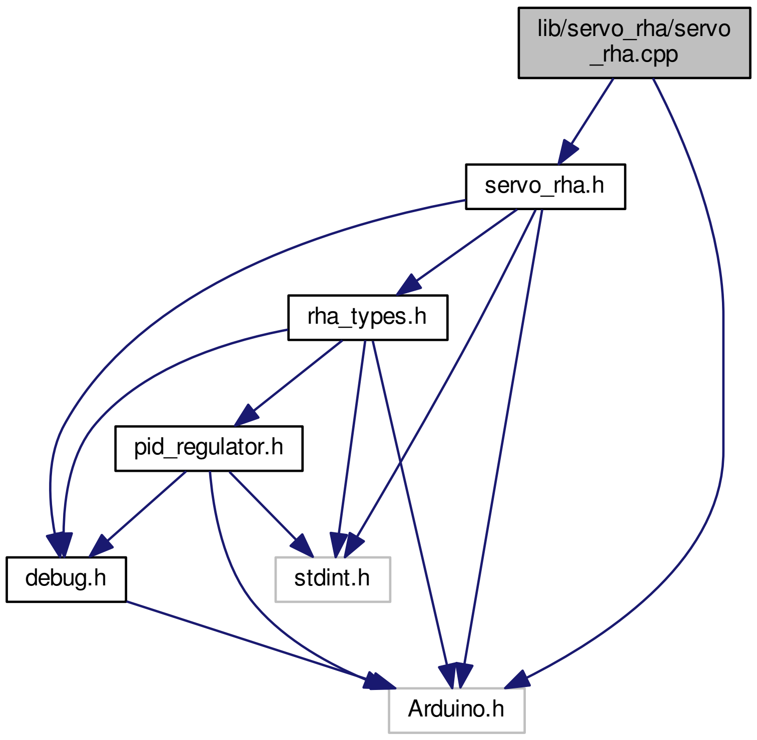
It the servo moves but not the joint for more than ERROR_MOVING_MARGIN cicles it raiseses an error

9.6 lib/servo_rha/servo_rha.cpp File Reference

Implements ServoRHA functions defined in servo_rha.h.

```
#include "servo_rha.h"
#include "Arduino.h"
```

Include dependency graph for servo_rha.cpp:



Functions

- uint8_t compareAngles (float _angle1, float _angle2, float _angle_margin)

  compareAngles function compares two angles with a margin set.
- uint8_t compareSpeed (float _speed1, float _speed2, float _speed_margin)

  compareSpeed function compares two speeds with a margin set.
- float **floatMap** (float x, float in_min, float in_max, float out_min, float out_max)

9.6.1 Detailed Description

Implements ServoRHA functions defined in servo_rha.h.

: Enrique Heredia Aguado <enheragu> : 2017_Sep_08 : RHA : servo_rha.cpp modified by: quique modified time: 30-Sep-2017

9.6.2 Function Documentation

9.6.2.1 uint8_t compareAngles (float *_angle1,* float *_angle2,* float *_angle_margin*)

compareAngles function compares two angles with a margin set.

Parameters

{uint16↩_t}	angle1 angle to compare
{uint16↩_t}	angle2 angle used in the comparison
{uint8_t}	angle_margin margin in which the angle1 will be considered to be equal to angle2 [angle2-angle_margin, angle2+angle_margin]

Returns

{uint8_t} Returns enumeration defined in servo_rha.h -> LESS_THAN, GREATER_THAN or EQUAL

9.6.2.2 uint8_t compareSpeed (float *_speed1,* float *_speed2,* float *_speed_margin*)

compareSpeed function compares two speeds with a margin set.

Parameters

{uint16↩_t}	speed1 speed to compare
{uint16↩_t}	speed2 speed used in the comparison
{uint8_t}	speed_margin margin in which the speed will be considered to be equal to speed2 [speed2-speed_margin, speed2+speed_margin]

Returns

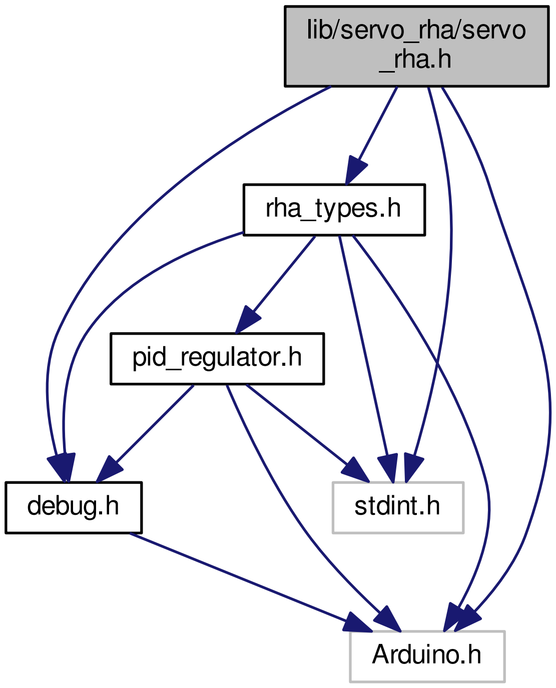
{uint8_t} Returns enumeration defined in servo_rha.h -> LESS_THAN, GREATER_THAN or EQUAL

9.7 lib/servo_rha/servo_rha.h File Reference

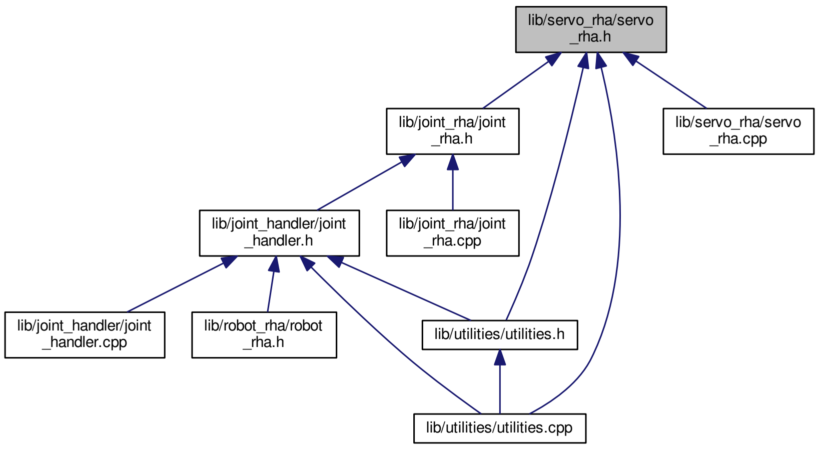
Implements ServoRHA class. This object inherits from CytronG15Servo object to enhance its capabilities.

```
#include "debug.h"
#include "rha_types.h"
#include "Arduino.h"
#include <stdint.h>
```

Include dependency graph for servo_rha.h:



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



Classes

- class ServoRHA

Macros

- #define **DELAY1** 500
- #define **TORQUE_CALIBRATION_INTERVAL** 5
- #define **MIN_TORQUE_CALIBRATION** 0
- #define **MAX_TORQUE_CALIBRATION** 800
- #define MARGIN_SPEED_COMPARISON 5
- #define MARGIN_ANGLE_COMPARISON 5
- #define **RETURN_PACKET_ALL** 0x02
- #define **RETURN_PACKET_NONE** 0x00
- #define **RETURN_PACKET_READ_INSTRUCTIONS** 0x01
- #define **TORQUE_OFFSET** 80
- #define **TORQUE_PREALIMENTATION_SLOPE** 1.2
- #define **MAX_TORQUE_VALUE** 1023
- #define **MAX_SPEED_VALUE** 30
- #define **CW** 1
- #define **CCW** 0
- #define **ON** 1
- #define **OFF** 0
- #define **KP** 10
- #define **KD** 0
- #define **KI** 0

Enumerations

- enum { **LESS_THAN**, **EQUAL**, **GREATER_THAN** }
- enum {
  MODEL_NUMBER_L, **MODEL_NUMBER_H**, **VERSION**, **ID**,
  BAUD_RATE, **RETURN_DELAY_TIME**, **CW_ANGLE_LIMIT_L**, **CW_ANGLE_LIMIT_H**,
  CCW_ANGLE_LIMIT_L, **CCW_ANGLE_LIMIT_H**, **RESERVED1**, **LIMIT_TEMPERATURE**,
  DOWN_LIMIT_VOLTAGE, **UP_LIMIT_VOLTAGE**, **MAX_TORQUE_L**, **MAX_TORQUE_H**,
  STATUS_RETURN_LEVEL, **ALARM_LED**, **ALARM_SHUTDOWN**, **RESERVED2**,
  DOWN_CALIBRATION_L, **DOWN_CALIBRATION_H**, **UP_CALIBRATION_L**, **UP_CALIBRATION_H**,
  TORQUE_ENABLE, **LED**, **CW_COMPLIANCE_MARGIN**, **CCW_COMPLIANCE_MARGIN**,
  CW_COMPLIANCE_SLOPE, **CCW_COMPLIANCE_SLOPE**, **GOAL_POSITION_L**, **GOAL_POSITION_H**,
  MOVING_SPEED_L, **MOVING_SPEED_H**, **TORQUE_LIMIT_L**, **TORQUE_LIMIT_H**,
  PRESENT_POSITION_L, **PRESENT_POSITION_H**, **PRESENT_SPEED_L**, **PRESENT_SPEED_H**,
  PRESENT_LOAD_L, **PRESENT_LOAD_H**, **PRESENT_VOLTAGE**, **PRESENT_TEMPERATURE**,
  REGISTERED_INSTRUCTION, **RESERVE3**, **MOVING**, **LOCK**,
  PUNCH_L, **PUNCH_H** }

Functions

- uint8_t compareAngles (float angle1, float angle2, float angle_margin=0)

  compareAngles function compares two angles with a margin set.
- uint8_t compareSpeed (float speed1, float speed2, float speed_margin=0)

  compareSpeed function compares two speeds with a margin set.
- float **floatMap** (float x, float in_min, float in_max, float out_min, float out_max)

9.7.1 Detailed Description

Implements ServoRHA class. This object inherits from CytronG15Servo object to enhance its capabilities.

: Enrique Heredia Aguado <enheragu> : 2017_Sep_08 : RHA : servo_rha.h modified by: quique modified time: 30-Sep-2017

9.7.2 Macro Definition Documentation

9.7.2.1 #define MARGIN_ANGLE_COMPARISON 5

MARGIN_ANGLE_COMPARISON defines an interval in which two angle values will be considered as the same value when compared

9.7.2.2 #define MARGIN_SPEED_COMPARISON 5

MARGIN_ANGLE_COMPARISON defines an interval in which two speed values will be considered as the same value when compared

9.7.3 Function Documentation

9.7.3.1 uint8_t compareAngles (float *_angle1,* float *_angle2,* float *_angle_margin*)

compareAngles function compares two angles with a margin set.

Parameters

{uint16←_t}	angle1 angle to compare
{uint16←_t}	angle2 angle used in the comparison
{uint8_t}	angle_margin margin in which the angle1 will be considered to be equal to angle2 [angle2-angle_margin, angle2+angle_margin]

Returns

{uint8_t} Returns enumeration defined in servo_rha.h -> LESS_THAN, GREATER_THAN or EQUAL

9.7.3.2 uint8_t compareSpeed (float *_speed1,* float *_speed2,* float *_speed_margin*)

compareSpeed function compares two speeds with a margin set.

Parameters

{uint16←_t}	speed1 speed to compare
{uint16←_t}	speed2 speed used in the comparison
{uint8_t}	speed_margin margin in which the speed will be considered to be equal to speed2 [speed2-speed_margin, speed2+speed_margin]

Returns

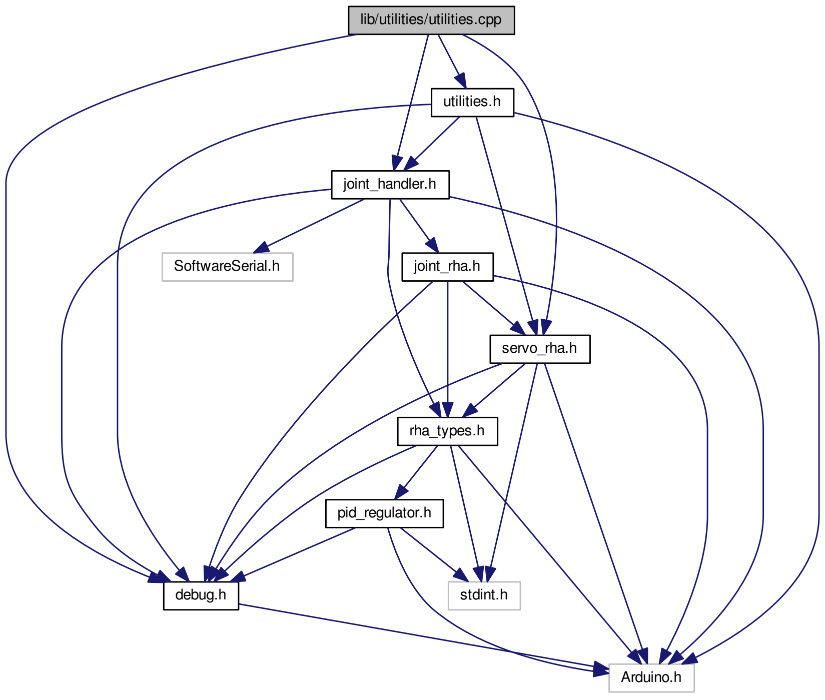
{uint8_t} Returns enumeration defined in servo_rha.h -> LESS_THAN, GREATER_THAN or EQUAL

9.8 lib/utilities/utilities.cpp File Reference

Implements a set of utilities to measure, experimentally, some interesting parameters.

```
#include "debug.h"
#include "servo_rha.h"
#include "utilities.h"
#include "joint_handler.h"
```

Include dependency graph for utilities.cpp:



Macros

- #define **SPEED_TARGET** 80
- #define **KP_REGULATOR** 150
- #define **LOOP_FREQUENCY** 100
- #define **BAUD_RATE_G15** 460800
- #define **CHAUVENET_REPETITIONS** 50
- #define **KN** 1.54
- #define **ENCODER_ANGLE_MARGIN** 15

Functions

- void MeasureUtilities::averageChauvenet (uint32_t ∗data, uint8_t n, float &arithmetic_average, float &standard_deviation)

  Calculates the average applying chauvenets criterion averageChauvenet.
- void **blinkLed** (uint8_t pin_led, uint8_t time_blink)
- void **testOnProcess** (uint8_t true_false)

9.8.1 Detailed Description

Implements a set of utilities to measure, experimentally, some interesting parameters.

Measures real speed of servo, time spent with packet handling, etc

: Enrique Heredia Aguado <enheragu> : 2017_Sep_13 : RHA : utilities.h modified by: quique modified time: 29-Oct-2017

9.8.2 Function Documentation

9.8.2.1 void MeasureUtilities::averageChauvenet (uint32_t ∗ *data,* uint8_t *n,* float & *arithmetic_average,* float & *standard_deviation*)

Calculates the average applying chauvenets criterion averageChauvenet.

Parameters

data	data to calculate the average
n	amount of data (max of 255)

Returns

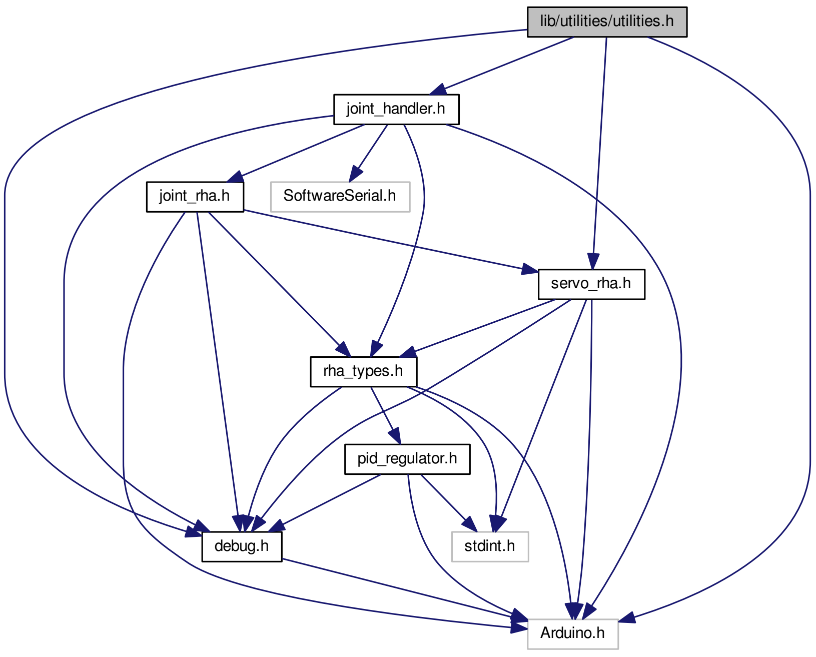
Returns the average

9.9 lib/utilities/utilities.h File Reference

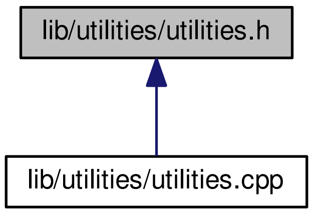
Implements a set of utilities to measure, experimentally, some interesting parameters.

```
#include "debug.h"
#include "servo_rha.h"
#include "joint_handler.h"
#include <Arduino.h>
```

Include dependency graph for utilities.h:



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



Classes

- class JHUtilitiesJH

Macros

- #define **LED** 13
- #define **SAMPLE_REGULATOR** 500
- #define **SPEED_REGULATOR_TEST** 120

- #define **SAMPLE_KP** 3
- #define **KP_SAMPLES** {2, 2, 2};
- #define **KD_SAMPLES** {0, 0.5, 0};
- #define **KI_SAMPLES** {0, 0, 0.1};
- #define **STEP** 0
- #define **SLOPE** 1
- #define **SAMPLE_STEP** 300
- #define **SAMPLE_TEST_STEP** 20
- #define **STEP_SPEED** 1023
- #define **SAMPLE_SLOPE** 800
- #define **SAMPLE_TEST_SLOPE** 20
- #define **SLOPE_SPEED** 0.15
- #define **SPEED** 1023
- #define **UP** CCW
- #define **DOWN** CW
- #define **LED_ROJO** 3
- #define **LED_VERDE** 4
- #define **PULSADOR** 5

Functions

- void MeasureUtilities::averageChauvenet (uint32_t ∗data, uint8_t n, float &arithmetic_average, float &standard_deviation)

    Calculates the average applying chauvenets criterion averageChauvenet.
- void **ServoUtilities::setServoId** (uint8_t new_id)
- void **ServoUtilities::fullFactoryResetBR** ()

9.9.1 Detailed Description

Implements a set of utilities to measure, experimentally, some interesting parameters.

Measures real speed of servo, time spent with packet handling, etc

: Enrique Heredia Aguado <enheragu> : 2017_Sep_08 : RHA : utilities.h modified by: quique modified time: 30-Sep-2017

9.9.2 Function Documentation

9.9.2.1 void MeasureUtilities::averageChauvenet (uint32_t ∗ *data,* uint8_t *n,* float & *arithmetic_average,* float & *standard_deviation*)

Calculates the average applying chauvenets criterion averageChauvenet.

Parameters

data	data to calculate the average
n	amount of data (max of 255)

Returns

Returns the average