Katana 450

API Documentation

Katana Native Interface

Document No: 233559
Version 0.0.4

Neuronics AG, Technoparkstrasse 1, CH-8005 Zürich
Tel: +41 44 445 16 40, Fax: +41 44 445 16 44
www.neuronics.ch
info@neuronics.ch

Contents

1 Overview

The Katana Native Interface KNI is an open source software library for controlling the Katana robot. KNI is written in C++ and structured so that it can easily be ported to other languages and frameworks. The code is non-platform-specific and can be compiled under both Windows (with the MS Visual C++ Compiler) and Linux (with the GNU Compiler Toolchain).

Since the KNI abstracts the underlying layers, applications can be written for the Katana without having to become involved in the details of the system. It takes just a few function calls to connect and initialise the robot. The protocol for controlling the robot from the PC is abstracted in its entirety. The KNI features an implementation of robot kinematics and path calculation routines for the synchronous control of all axes and the traversing of paths in space with the end effector.

The openness of the common sources also makes the KNI the ideal tool for research and training, since the entire implementation can be traced, as well as modified and adapted at will.

2 Module Index

2.1 Modules

Here is a list of all modules:

3 Directory Hierarchy

3.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

4 Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

5 Class Index

5.1 Class Hierarchy

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

6 Class Index

6.1 Class List

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

7 File Index

7.1 File List

Here is a list of all files with brief descriptions:

8 Module Documentation

8.1 Exceptions

Classes

- struct Context
- class Exception
- class CannotOpenPortException

  Failed to open the serial communication device.

- class CannotGetSetPortAttributesException

  Could not set or get the attributes for the given serial communication device.

- class PortNotOpenException

  The port was not open.

- class DeviceReadException

  Reading from the serial communication device failed.

- class DeviceWriteException

  Writing to the serial communication device failed.

- class ReadWriteNotCompleteException

  This exception is the base for the WriteNotComplete and ReadNotCompleteException.

- class WriteNotCompleteException

  Not all bytes could be written to the serial communication device.

- class ReadNotCompleteException

  The Katana didn't answer correctly within the given timeout.

- class ErrorException

  The Katana returned an error string.

- class WrongCRCException

  CRC check for the answer package failed.

- class FirmwareException

  Exception reported by the firmware.

- class SlaveErrorException

  Slave error occurred.

- class ParameterReadingException

  There was an error while reading a parameter from the robot.

- class ParameterWritingException

  The data you wanted to send to the robot was invalid.

- class WrongParameterException

  The given parameter was wrong.

- class MotorOutOfRangeException

  The encoders for the given motor were out of range.

- class MotorTimeoutException

  The timeout elapsed for the given motor and target position.

- class MotorCrashException

  The requested motor crashed during the movement.

- class ConfigFileOpenException

  Accessing the given configuration file failed (may be: access denied or wrong path).

- class ConfigFileStateException

  The state of the configuration file wasn't "good".

- class ConfigFileSectionNotFoundException

  The requested section could not be found.

- class ConfigFileSubsectionNotFoundException

  The requested subsection could not be found.

- class ConfigFileEntryNotFoundException

  The requested entry could not be found.

- class ConfigFileSyntaxErrorException

  There was a syntax error in the configuration file.

- class KNI::NoSolutionException

  No solution found for the given cartesian coordinates.

- class JointSpeedException

  Joint speed too high.

- class WaitParameterException

  Wait parameter set to false.

- class CLMBase

  Linear movement Class.

Functions

- double CLMBase::totalTime (double distance, double acc, double dec, double vmax)

  Calculates time needed for movement over a distance.

- double CLMBase::relPosition (double reltime, double distance, double acc, double dec, double vmax)

  Calculates the relative position reached after the relative time given.

- void CLMBase::splineCoefficients (int steps, double *timearray, double *encoderarray, double *arr_p1, double *arr_p2, double *arr_p3, double *arr_p4)

  Calculates the spline coefficient and stores them in arr_p1 - arr_p4.

- bool CLMBase::checkJointSpeed (std::vector< int > lastsolution, std::vector< int > solution, double time)

  Checks if the joint speeds are below speed limit.

- int CLMBase::getSpeed (int distance, int acceleration, int time)

  Calculates speed from distance, acceleration and time for the movement.

- CLMBase::CLMBase ()
- void CLMBase::movLM (double X, double Y, double Z, double Al, double Be, double Ga, bool exactflag, double vmax, bool wait=true, int tolerance=100, long timeout=TM_ENDLESS)
- void CLMBase::movLM2P (double X1, double Y1, double Z1, double Al1, double Be1, double Ga1, double X2, double Y2, double Z2, double Al2, double Be2, double Ga2, bool exactflag, double vmax, bool wait=true, int tolerance=100, long timeout=TM_ENDLESS)

  Move linear from point to point using multiple splines.

- void CLMBase::movP2P (double X1, double Y1, double Z1, double Ph1, double Th1, double Ps1, double X2, double Y2, double Z2, double Ph2, double Th2, double Ps2, bool exactflag, double vmax, bool wait=true, long timeout=TM_ENDLESS)

  Move point to point using splines.

- void CLMBase::setMaximumLinearVelocity (double maximumVelocity)
- double CLMBase::getMaximumLinearVelocity () const
- void CLMBase::setActivatePositionController (bool activate)

  Re-Activate the position controller after the linear movement.

- bool CLMBase::getActivatePositionController ()

  Check if the position controller will be activated after the linear movement.

- void CLMBase::moveRobotLinearTo (double x, double y, double z, double phi, double theta, double psi, bool waitUntilReached=true, int waitTimeout=TM_ENDLESS)
- void CLMBase::moveRobotLinearTo (std::vector< double > coordinates, bool waitUntilReached=true, int waitTimeout=TM_ENDLESS)

  This method does the same as the one above and is mainly provided for convenience.

- void CLMBase::moveRobotTo (double x, double y, double z, double phi, double theta, double psi, bool waitUntilReached=true, int waitTimeout=TM_ENDLESS)

  Moves to robot to given cartesian coordinates and euler-angles.

- void CLMBase::moveRobotTo (std::vector< double > coordinates, bool waitUntilReached=true, int waitTimeout=TM_ENDLESS)

  This method does the same as the one above and is mainly provided for convenience.

Variables

- bool CLMBase::_activatePositionController

8.1.1 Function Documentation

8.1.1.1 bool CLMBase::checkJointSpeed (std::vector< int > *lastsolution*, std::vector< int > *solution*, double *time*) [private, inherited]

Checks if the joint speeds are below speed limit.

Maximum joint speed is 180enc / 10ms.

Author:

Jonas Haller

Parameters:

lastsolution encoder values of last point

solution encoder values of current point

time time difference between the points in s

Returns:

true if joint speeds ok, false if joint speed too high

8.1.1.2 CLMBase::CLMBase () [inline, inherited]

Definition at line 153 of file lmBase.h.

8.1.1.3 bool CLMBase::getActivatePositionController () [inherited]

Check if the position controller will be activated after the linear movement.

8.1.1.4 double CLMBase::getMaximumLinearVelocity () const [inherited]

8.1.1.5 int CLMBase::getSpeed (int *distance*, int *acceleration*, int *time*) [private, inherited]

Calculates speed from distance, acceleration and time for the movement.

Author:

Jonas Haller

Parameters:

distance absolute (positive) distance of the movement in encoder

acceleration acceleration and deceleration in enc / s^2

time time that can be used for the movement in s

Returns:

speed in enc / s to finish the movement on time

8.1.1.6 void CLMBase::moveRobotLinearTo (std::vector< double > *coordinates*, bool *waitUntilReached* = true, int *waitTimeout* = TM_ENDLESS) [inherited]

This method does the same as the one above and is mainly provided for convenience.

Note:

You can call this function in python using tuples: Example: katana.moveRobotLinearTo((x,y,z,phi,theta,psi))
If the size of the container is smaller than 6, it will throw an exception!

8.1.1.7 void CLMBase::moveRobotLinearTo (double *x*, double *y*, double *z*, double *phi*, double *theta*, double *psi*, bool *waitUntilReached* = true, int *waitTimeout* = TM_ENDLESS) [inherited]

Parameters:

waitUntilReached has to be true with new implementation of movLM2P

8.1.1.8 void CLMBase::moveRobotTo (std::vector< double > *coordinates*, bool *waitUntilReached* = true, int *waitTimeout* = TM_ENDLESS) [inherited]

This method does the same as the one above and is mainly provided for convenience.

Note:

You can call this function in python using tuples: Example: katana.moveRobotTo((x,y,z,phi,theta,psi))
If the size of the container is smaller than 6, it will throw an exception

Reimplemented from CikBase.

8.1.1.9 void CLMBase::moveRobotTo (double *x*, double *y*, double *z*, double *phi*, double *theta*, double *psi*, bool *waitUntilReached* = true, int *waitTimeout* = TM_ENDLESS) [inherited]

Moves to robot to given cartesian coordinates and euler-angles.

Note:

Instead of a given tolerance, a default tolerance is being used

Reimplemented from CikBase.

8.1.1.10 void CLMBase::movLM (double *X*, double *Y*, double *Z*, double *Al*, double *Be*, double *Ga*, bool *exactflag*, double *vmax*, bool *wait* = true, int *tolerance* = 100, long *timeout* = TM_ENDLESS) [inherited]

Parameters:

wait has to be true with new implementation of movLM2P

8.1.1.11 void CLMBase::movLM2P (double *X1*, double *Y1*, double *Z1*, double *Al1*, double *Be1*, double *Ga1*, double *X2*, double *Y2*, double *Z2*, double *Al2*, double *Be2*, double *Ga2*, bool *exactflag*, double *vmax*, bool *wait* = true, int *tolerance* = 100, long *timeout* = TM_ENDLESS) [inherited]

Move linear from point to point using multiple splines.

Author:

Jonas Haller

Parameters:

X1,Y1,Z1,Ph1,Th1,Ps1 X, Y, Z, Phi, Theta, Psi of actual position

X2,Y2,Z2,Ph2,Th2,Ps2 X, Y, Z, Phi, Theta, Psi of target position

exactflag activate the position controller after the movement

vmax maximum velocity of the movement in mm/s

wait wait for end of movement

tolerance tolerance for all motor encoders

timeout timeout for linear movement in ms

Exceptions:

NoSolutionException if no solution found for IK

JointSpeedException if joint speed too high

WaitParameterException if wait set to false

Returns:

void

8.1.1.12 void CLMBase::movP2P (double *X1*, double *Y1*, double *Z1*, double *Ph1*, double *Th1*, double *Ps1*, double *X2*, double *Y2*, double *Z2*, double *Ph2*, double *Th2*, double *Ps2*, bool *exactflag*, double *vmax*, bool *wait* = `true`, long *timeout* = `TM_ENDLESS`) `[inherited]`

Move point to point using splines.

Author:

Jonas Haller

Parameters:

X1,Y1,Z1,Ph1,Th1,Ps1 X, Y, Z, Phi, Theta, Psi of actual position

X2,Y2,Z2,Ph2,Th2,Ps2 X, Y, Z, Phi, Theta, Psi of target position

exactflag activate the position controller after the movement

vmax maximum velocity for motors

wait wait for end of movement

tolerance tolerance for all motor encoders

timeout timeout for movement in ms

Exceptions:

NoSolutionException if no solution found for IK

JointSpeedException if joint speed too high

WaitParameterException if wait set to false

Returns:

void

8.1.1.13 double CLMBase::relPosition (double *reltime*, double *distance*, double *acc*, double *dec*, double *vmax*) `[private, inherited]`

Calculates the relative position reached after the relative time given.

Author:

Jonas Haller

Parameters:

reltime relative time (fraction of totaltime)

distance distance of the movement in mm

acc acceleration at the beginning in mm/s^2

dec deceleration at the end in mm/s^2

vmax maximum velocity of the movement in mm/s

Returns:

relative distance (fraction of distance)

8.1.1.14 void CLMBase::setActivatePositionController (bool *activate*) [inherited]

Re-Activate the position controller after the linear movement.

Note:

This can result in a small movement after the movement

8.1.1.15 void CLMBase::setMaximumLinearVelocity (double *maximumVelocity*) [inherited]

8.1.1.16 void CLMBase::splineCoefficients (int *steps*, double * *timearray*, double * *encoderarray*, double * *arr_p1*, double * *arr_p2*, double * *arr_p3*, double * *arr_p4*) [private, inherited]

Calculates the spline coefficient and stores them in arr_p1 - arr_p4.

Boundary conditions are that f_1'=0 and f_n'=0 (zero velocity at beginning and end of the movement) and f_i"=P_(i+1)".

Author:

Jonas Haller

Parameters:

steps number of splines to calculate

timearray times of the points (length = steps + 1)

encoderarray encoder values of the points (length = steps + 1)

arr_p1 to return parameters 1 (length = steps)

arr_p2 to return parameters 2 (length = steps)

arr_p3 to return parameters 3 (length = steps)

arr_p4 to return parameters 4 (length = steps)

Returns:

void

8.1.1.17 double CLMBase::totalTime (double *distance*, double *acc*, double *dec*, double *vmax*) [private, inherited]

Calculates time needed for movement over a distance.

Author:

Jonas Haller

Parameters:

distance distance of the movement in mm

acc acceleration at the beginning in mm/s^2

dec deceleration at the end in mm/s^2

vmax maximum velocity of the movement in mm/s

Returns:

time needed for the movement in s

8.1.2 Variable Documentation

8.1.2.1 bool CLMBase::_activatePositionController [private, inherited]

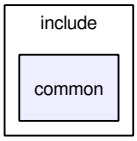
Definition at line 77 of file lmBase.h.

9 Directory Documentation

9.1 include/common/ Directory Reference

Files

- file dllexport.h
- file exception.h
- file MathHelperFunctions.h
- file Timer.h


include
common

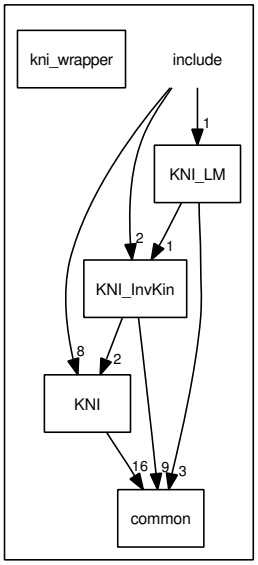
9.2 include/ Directory Reference

Directories

- directory common
- directory KNI
- directory KNI_InvKin
- directory KNI_LM
- directory kni_wrapper

Files

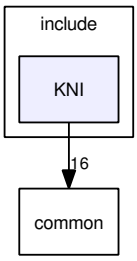
- file kniBase.h
- file libKinematics.h


kni_wrapper
include
1
KNI_LM
2
1
KNI_InvKin
8
2
KNI
16
9
3
common

9.3 include/KNI/ Directory Reference

Files

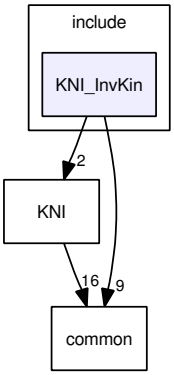
- file cdlBase.h
- file cdlCOM.h
- file cdlCOMExceptions.h
- file cdlSocket.h
- file cplBase.h
- file cplSerial.h
- file CRC.h
- file kmlBase.h
- file kmlCommon.h
- file kmlExt.h
- file kmlFactories.h
- file kmlMotBase.h
- file kmlSctBase.h


include
KNI
16
common

9.4 include/KNI_InvKin/ Directory Reference

Files

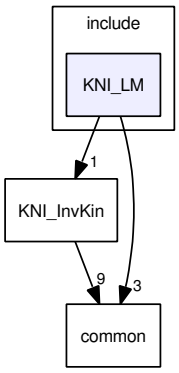
- file ikBase.h
- file KatanaKinematics.h
- file KatanaKinematics5M180.h
- file KatanaKinematics6M180.h
- file KatanaKinematics6M90G.h
- file KatanaKinematics6M90T.h
- file KatanaKinematicsDecisionAlgorithms.h


include
KNI_InvKin
2
KNI
16
9
common

9.5 include/KNI_LM/ Directory Reference

Files

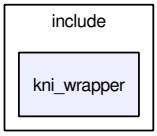
- file lmBase.h


include
KNI_LM
1
KNI_InvKin
9
3
common

9.6 include/kni_wrapper/ Directory Reference

Files

- file kni_wrapper.h


include
kni_wrapper

10 Namespace Documentation

10.1 KNI Namespace Reference

Classes

- class Timer

  Provides a stop-watch-like class with a resolution of milliseconds.

- class kmlFactory

  This class is for internal use only It may change at any time It shields the configuration file parsing.

- class NoSolutionException

  No solution found for the given cartesian coordinates.

- struct KinematicParameters

  To pass different parameters for the kinematic implementations.

- class KatanaKinematics

  The base class for all kinematic implementations.

- class KatanaKinematics5M180
- class KatanaKinematics6M180
- class KatanaKinematics6M90G
- class KatanaKinematics6M90T
- struct KinematicsDefaultEncMinAlgorithm

Functions

- void sleep (long time)

  This functions shields the platform specific implementation of the sleep function.

10.1.1 Function Documentation

10.1.1.1 void KNI::sleep (long *time*)

This functions shields the platform specific implementation of the sleep function.

10.2 KNI_MHF Namespace Reference

Classes

- struct unary_precalc_sin

  function-object which calculates sinus for n-elements of a container if used together with a STL algorithm

- struct unary_precalc_cos
- struct unary_rad2deg

  a function-object version of rad2deg

- struct unary_deg2rad

  a function-object version of rad2deg

Functions

- template<typename _T>
  short sign (_T x)
- template<typename _T>
  _T atan1 (_T in1, _T in2)
- template<typename _T>
  _T acotan (const _T in)
- template<typename _T>
  _T atan0 (const _T in1, const _T in2)
- template<typename _T>
  _T pow2 (const _T in)
- template<typename _T>
  _T rad2deg (const _T a)

  conversion from radian to degree

- template<typename _T>
  _T deg2rad (const _T a)

  conversion from degree to radian

- template<typename _T>
  _T anglereduce (const _T a)
- template<typename _angleT, typename _encT>
  _encT rad2enc (_angleT const &angle, _angleT const &angleOffset, _encT const &epc, _encT const &encOffset, _encT const &rotDir)

  converts absolute angles in radian to encoders.

- template<typename _angleT, typename _encT>
  _angleT enc2rad (_encT const &enc, _angleT const &angleOffset, _encT const &epc, _encT const &encOffset, _encT const &rotDir)

  converts encoders to absolute angles in radian

- double findFirstEqualAngle (double cosValue, double sinValue, double tolerance)

  Find the first equal angle.

10.2.1 Function Documentation

10.2.1.1 template<typename _T> _T KNI_MHF::acotan (const _T *in*) [inline]

Definition at line 82 of file MathHelperFunctions.h.

References M_PI.

10.2.1.2 template<typename _T> _T KNI_MHF::anglereduce (const _T *a*) [inline]

Definition at line 131 of file MathHelperFunctions.h.

References M_PI.

Referenced by findFirstEqualAngle().

10.2.1.3 template<typename _T> _T KNI_MHF::atan0 (const _T *in1*, const _T *in2*) [inline]

Definition at line 90 of file MathHelperFunctions.h.

References M_PI.

10.2.1.4 template<typename _T> _T KNI_MHF::atan1 (_T *in1*, _T *in2*) [inline]

Definition at line 62 of file MathHelperFunctions.h.

References M_PI, and sign().

Here is the call graph for this function:

10.2.1.5 template<typename _T> _T KNI_MHF::deg2rad (const _T *a*) [inline]

conversion from degree to radian

Definition at line 119 of file MathHelperFunctions.h.

References M_PI.

Referenced by KNI_MHF::unary_deg2rad< _T >::operator()().

10.2.1.6 template<typename _angleT, typename _encT> _angleT KNI_MHF::enc2rad (_encT const & *enc*, _angleT const & *angleOffset*, _encT const & *epc*, _encT const & *encOffset*, _encT const & *rotDir*) [inline]

converts encoders to absolute angles in radian

Definition at line 153 of file MathHelperFunctions.h.

References M_PI.

10.2.1.7 double KNI_MHF::findFirstEqualAngle (double *cosValue*, double *sinValue*, double *tolerance*) [inline]

Find the first equal angle.

You have to pass a cos and a sin Value

Definition at line 162 of file MathHelperFunctions.h.

References anglereduce(), and M_PI.

Here is the call graph for this function:

10.2.1.8 template<typename _T> _T KNI_MHF::pow2 (const _T *in*) [inline]

Definition at line 97 of file MathHelperFunctions.h.

10.2.1.9 template<typename _T> _T KNI_MHF::rad2deg (const _T *a*) [inline]

conversion from radian to degree

Definition at line 105 of file MathHelperFunctions.h.

References M_PI.

Referenced by KNI_MHF::unary_rad2deg< _T >::operator()().

10.2.1.10 template<typename _angleT, typename _encT> _encT KNI_MHF::rad2enc (_angleT const & *angle*, _angleT const & *angleOffset*, _encT const & *epc*, _encT const & *encOffset*, _encT const & *rotDir*) [inline]

converts absolute angles in radian to encoders.

Definition at line 139 of file MathHelperFunctions.h.

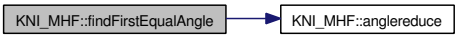
References M_PI.

10.2.1.11 template<typename _T> short KNI_MHF::sign (_T *x*) [inline]

Definition at line 37 of file MathHelperFunctions.h.

Referenced by atan1().


KNI_MHF::atan1
KNI_MHF::sign


KNI_MHF::findFirstEqualAngle
KNI_MHF::anglereduce

11 Class Documentation

11.1 CannotGetSetPortAttributesException Class Reference

Could not set or get the attributes for the given serial communication device.

`#include <cdlCOMExceptions.h>`

Inheritance diagram for CannotGetSetPortAttributesException: Collaboration diagram for CannotGetSetPortAttributesException:

Public Member Functions

- CannotGetSetPortAttributesException (const std::string &port) throw ()

11.1.1 Detailed Description

Could not set or get the attributes for the given serial communication device.

Note:

error_number=-11

Definition at line 56 of file cdlCOMExceptions.h.

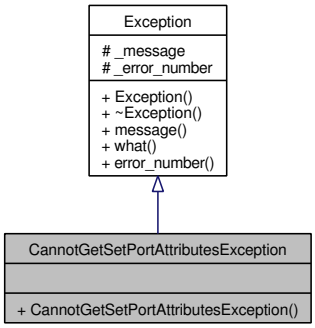
11.1.2 Constructor & Destructor Documentation

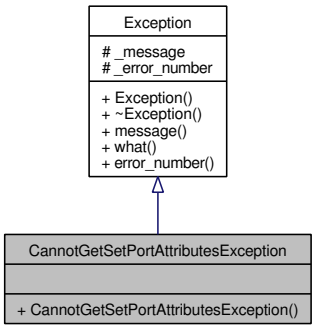
11.1.2.1 CannotGetSetPortAttributesException::CannotGetSetPortAttributesException (const std::string & *port*) throw () [inline]

Definition at line 58 of file cdlCOMExceptions.h.

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

- include/KNI/cdlCOMExceptions.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
CannotGetSetPortAttributesException
+ CannotGetSetPortAttributesException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
CannotGetSetPortAttributesException
+ CannotGetSetPortAttributesException()

11.2 CannotOpenPortException Class Reference

Failed to open the serial communication device.

`#include <cdlCOMExceptions.h>`

Inheritance diagram for CannotOpenPortException: Collaboration diagram for CannotOpenPortException:

Public Member Functions

- CannotOpenPortException (const std::string &port, const std::string os_msg) throw ()

11.2.1 Detailed Description

Failed to open the serial communication device.

Note:

error_number=-10
Linux only: You get also the direct error message from the system

Definition at line 47 of file cdlCOMExceptions.h.

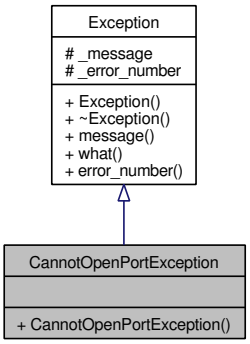
11.2.2 Constructor & Destructor Documentation

11.2.2.1 CannotOpenPortException::CannotOpenPortException (const std::string & *port*, const std::string *os_msg*) throw () [inline]

Definition at line 49 of file cdlCOMExceptions.h.

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

- include/KNI/cdlCOMExceptions.h



Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
CannotOpenPortException
+ CannotOpenPortException()

11.3 CCdlBase Class Reference

Abstract base class for devices.

`#include <cdlBase.h>`

Inheritance diagram for CCdlBase:

Public Member Functions

- virtual int send (const void *_buf, int _sz)=0
  Pure function to send data.
- virtual int recv (void *_buf, int _sz)=0
  Pure function to receive data.
- virtual ~CCdlBase ()
  destructor

11.3.1 Detailed Description

Abstract base class for devices.

This class is the base abstract class for devices; the abbreviation 'cdl' stands for 'Communication Device Layer'. By inheriting from this class different communication devices such a USB or a COM port can be handled easier.

Definition at line 47 of file cdlBase.h.

11.3.2 Constructor & Destructor Documentation

11.3.2.1 virtual CCdlBase::~CCdlBase () [inline, virtual]

destructor

This class is only an interface

Definition at line 69 of file cdlBase.h.

11.3.3 Member Function Documentation

11.3.3.1 virtual int CCdlBase::send (const void * *_buf*, int *_sz*) [pure virtual]

Pure function to send data.

This function is pure and should always be overwritten by classes inheriting from 'CCdlBase'. As the name proposes the function should contain a sending behavour from the device.

Implemented in CCdlCOM, and CCdlSocket.

11.3.3.2 virtual int CCdlBase::recv (void * *_buf*, int *_sz*) [pure virtual]

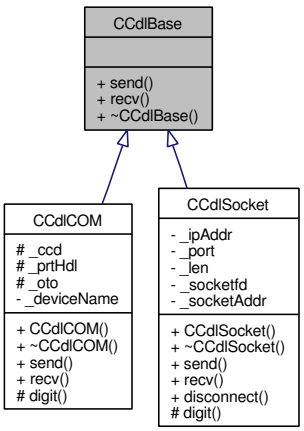
Pure function to receive data.

This function is pure and should always be overwritten by classes in- heriting from 'CCdlBase'. As the name proposes the function should contain a sending behavour from the device.

Implemented in CCdlCOM, and CCdlSocket.

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

- include/KNI/cdlBase.h


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
CCdlCOM
_ccd
_prtHdl
_oto
- _deviceName
+ CCdlCOM()
+ ~CCdlCOM()
+ send()
+ recv()
# digit()
CCdlSocket
- _ipAddr
- _port
- _len
- _socketfd
- _socketAddr
+ CCdlSocket()
+ ~CCdlSocket()
+ send()
+ recv()
+ disconnect()
# digit()

11.4 CCdlCOM Class Reference

Encapsulates the serial port device.

`#include <cdlCOM.h>`

Inheritance diagram for CCdlCOM: Collaboration diagram for CCdlCOM:

Public Member Functions

- CCdlCOM (TCdlCOMDesc ccd)
  Construct a CCdlCOM class.
- virtual ~CCdlCOM ()
  Destructs the class.
- virtual int send (const void *buf, int size)
  Sends data to the device.
- virtual int recv (void *buf, int size)
  Receives data from the device.

Static Protected Member Functions

- static char digit (const int _val)
  Converts an integer to a char.

Protected Attributes

- TCdlCOMDesc _ccd
  Stores the attributes of the serial port device.
- int _prtHdl
  port handle
- struct termios _oto
  old timeouts

Private Attributes

- std::string _deviceName

11.4.1 Detailed Description

Encapsulates the serial port device.

This class is responsible for direct communication with the serial port device. It builds the lowest layer for communication and uses the system API functions to get access the to the device.

Definition at line 73 of file cdlCOM.h.

11.4.2 Constructor & Destructor Documentation

11.4.2.1 CCdlCOM::CCdlCOM (TCdlCOMDesc *ccd*)

Construct a CCdlCOM class.

To this constructor a 'TCdlCOMDesc' parameter has to be given, which describes the desired serial port. An attempt to open a connection to the desired device will be tried.

11.4.2.2 virtual CCdlCOM::~CCdlCOM () [virtual]

Destructs the class.

11.4.3 Member Function Documentation

11.4.3.1 static char CCdlCOM::digit (const int *_val*) [inline, static, protected]

Converts an integer to a char.

Definition at line 99 of file cdlCOM.h.

11.4.3.2 virtual int CCdlCOM::send (const void * *buf*, int *size*) [virtual]

Sends data to the device.

Implements CCdlBase.

11.4.3.3 virtual int CCdlCOM::recv (void * *buf*, int *size*) [virtual]

Receives data from the device.

Implements CCdlBase.

11.4.4 Member Data Documentation

11.4.4.1 std::string CCdlCOM::_deviceName [private]

Definition at line 75 of file cdlCOM.h.

11.4.4.2 TCdlCOMDesc CCdlCOM::_ccd [protected]

Stores the attributes of the serial port device.

Definition at line 79 of file cdlCOM.h.

11.4.4.3 int CCdlCOM::_prtHdl [protected]

port handle

Definition at line 89 of file cdlCOM.h.

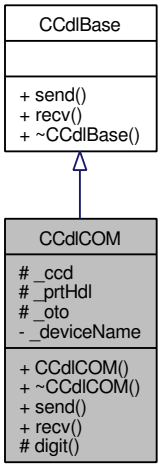
11.4.4.4 struct termios CCdlCOM::_oto [read, protected]

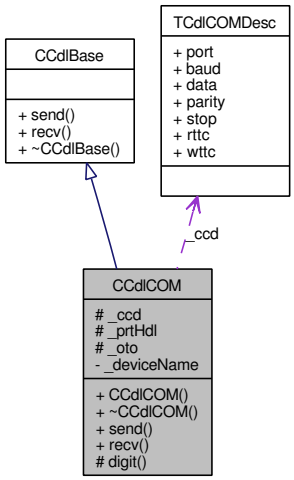
old timeouts

Definition at line 90 of file cdlCOM.h.

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

- include/KNI/cdlCOM.h


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
CCdlCOM
_ccd
_prtHdl
_oto
- _deviceName
+ CCdlCOM()
+ ~CCdlCOM()
+ send()
+ recv()
# digit()


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
TCdlCOMDesc
+ port
+ baud
+ data
+ parity
+ stop
+ rttc
+ wttc
_ccd
CCdlCOM
_ccd
_prtHdl
_oto
- _deviceName
+ CCdlCOM()
+ ~CCdlCOM()
+ send()
+ recv()
# digit()

11.5 CCdlSocket Class Reference

Encapsulates the socket communication device.

`#include <cdlSocket.h>`

Inheritance diagram for CCdlSocket: Collaboration diagram for CCdlSocket:

Public Member Functions

- CCdlSocket (char *adress, int port)
  Constructs a CCdlSocket object.
- virtual ~CCdlSocket ()
  Destructs the object.
- virtual int send (const void *_buf, int _size)
  Sends data to the socket.
- virtual int recv (void *_buf, int _size)
  Receives data from the socket.
- virtual int disconnect ()
  Terminates the socket connection.

Static Protected Member Functions

- static char digit (const int _val)
  Converts an integer to a char.

Private Attributes

- char * _ipAddr
  IP Address of the Robot or simulation environment.
- int _port
  Port number of the KNI communication socket.
- int _len
  Length of the message.
- int _socketfd
  File handler for the socket.
- struct sockaddr_in _socketAddr
  Structure to fill in the socket communication parameteres.

11.5.1 Detailed Description

Encapsulates the socket communication device.

This class is responsible for direct communication with the Katana robot or its simulation environment through sockets. It builds the lowest layer for KNI communication and uses the system API functions to get access to the socket.

Definition at line 61 of file cdlSocket.h.

11.5.2 Constructor & Destructor Documentation

11.5.2.1 CCdlSocket::CCdlSocket (char * *adress*, int *port*)

Constructs a CCdlSocket object.

To this constructor the socket's AF_INET address (for platform independence) and port number have to be given as parameters. An attempt to open a connection to the desired device will be tried and if successful, 'lastOP()' will return 'lopDONE', otherwise 'lopFAIL'.

11.5.2.2 virtual CCdlSocket::~CCdlSocket () [virtual]

Destructs the object.

11.5.3 Member Function Documentation

11.5.3.1 static char CCdlSocket::digit (const int *_val*) [inline, static, protected]

Converts an integer to a char.

Definition at line 93 of file cdlSocket.h.

11.5.3.2 virtual int CCdlSocket::send (const void * *_buf*, int *_size*) [virtual]

Sends data to the socket.

Implements CCdlBase.

11.5.3.3 virtual int CCdlSocket::recv (void * *_buf*, int *_size*) [virtual]

Receives data from the socket.

Implements CCdlBase.

11.5.3.4 virtual int CCdlSocket::disconnect () [virtual]

Terminates the socket connection.

11.5.4 Member Data Documentation

11.5.4.1 char* CCdlSocket::_ipAddr [private]

IP Address of the Robot or simulation environment.

Set to localhost or 127.0.0.1 if the simulation runs on the same machine

Definition at line 65 of file cdlSocket.h.

11.5.4.2 int CCdlSocket::_port [private]

Port number of the KNI communication socket.

Definition at line 67 of file cdlSocket.h.

11.5.4.3 int CCdlSocket::_len [private]

Length of the message.

Definition at line 69 of file cdlSocket.h.

11.5.4.4 int CCdlSocket::_socketfd [private]

File handler for the socket.

Definition at line 82 of file cdlSocket.h.

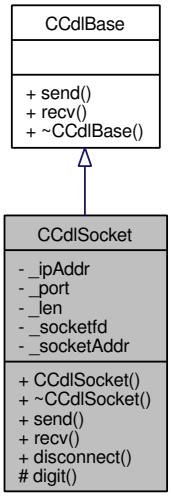
11.5.4.5 struct sockaddr_in CCdlSocket::_socketAddr [read, private]

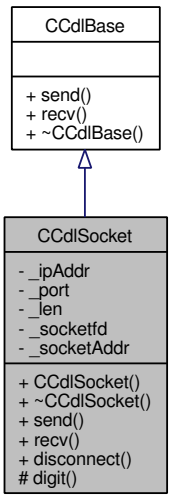
Structure to fill in the socket communication parameteres.

Definition at line 84 of file cdlSocket.h.

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

- include/KNI/cdlSocket.h


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
CCdlSocket
- _ipAddr
- _port
- _len
- _socketfd
- _socketAddr
+ CCdlSocket()
+ ~CCdlSocket()
+ send()
+ recv()
+ disconnect()
# digit()


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
CCdlSocket
- _ipAddr
- _port
- _len
- _socketfd
- _socketAddr
+ CCdlSocket()
+ ~CCdlSocket()
+ send()
+ recv()
+ disconnect()
# digit()

11.6 CCplBase Class Reference

Abstract base class for protocol definiton.

`#include <cplBase.h>`

Inheritance diagram for CCplBase: Collaboration diagram for CCplBase:

Public Member Functions

- virtual bool init (CCdlBase *_device, byte _kataddr=24)=0
  Basic initializing function.
- virtual void comm (const byte *pack, byte *buf, byte *size)=0
  Base communication function.
- virtual ~CCplBase ()
  destructor
- virtual void getMasterFirmware (short *fw, short *rev)=0
  Get the master firmware of the robot we are communicating with.

Protected Attributes

- CCdlBase * device
  communication device
- short mMasterVersion
  master version of robot we are communicating with
- short mMasterRevision
  master firmware revision

11.6.1 Detailed Description

Abstract base class for protocol definiton.

The robot can be controled by using different kind of protocols; this class has been introduced as an abstract base class to manage them gether; every protocol the robot should use in futur shoud be derived from this class.

Definition at line 47 of file cplBase.h.

11.6.2 Constructor & Destructor Documentation

11.6.2.1 virtual CCplBase::~CCplBase () `[inline, virtual]`

destructor

This class is only an interface

Definition at line 75 of file cplBase.h.

11.6.3 Member Function Documentation

11.6.3.1 virtual bool CCplBase::init (CCdlBase * *_device*, byte *_kataddr* = 24) [pure virtual]

Basic initializing function.

The children of this class should write their initializing part in that function.

Implemented in CCplSerialCRC.

11.6.3.2 virtual void CCplBase::comm (const byte * *pack*, byte * *buf*, byte * *size*) [pure virtual]

Base communication function.

The children of this class should write their main double way communication in this function.

Implemented in CCplSerialCRC.

11.6.3.3 virtual void CCplBase::getMasterFirmware (short * *fw*, short * *rev*) [pure virtual]

Get the master firmware of the robot we are communicating with.

Get master firmware read at initialization time.

Implemented in CCplSerialCRC.

11.6.4 Member Data Documentation

11.6.4.1 CCdlBase* CCplBase::device [protected]

communication device

Definition at line 50 of file cplBase.h.

11.6.4.2 short CCplBase::mMasterVersion [protected]

master version of robot we are communicating with

Definition at line 51 of file cplBase.h.

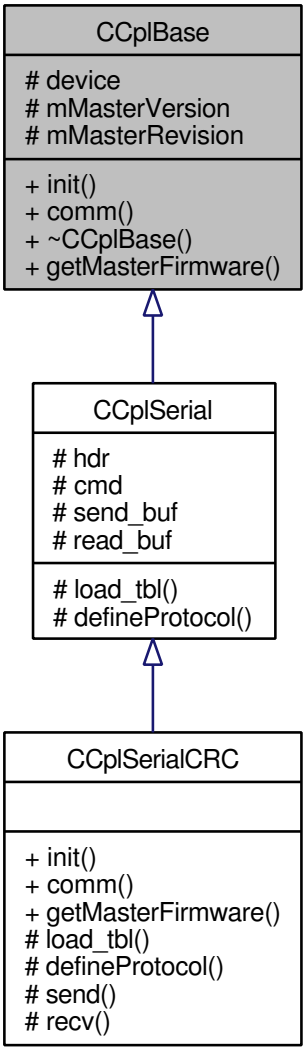
11.6.4.3 short CCplBase::mMasterRevision [protected]

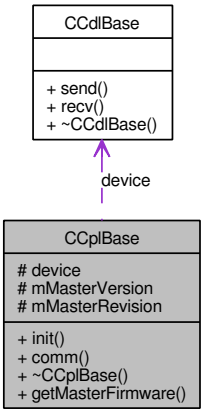
master firmware revision

Definition at line 52 of file cplBase.h.

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

- include/KNI/cplBase.h


CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
CCplSerial
hdr
cmd
send_buf
read_buf
load_tbl()
defineProtocol()
CCplSerialCRC
+ init()
+ comm()
+ getMasterFirmware()
# load_tbl()
# defineProtocol()
# send()
# recv()


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()

11.7 CCplSerial Class Reference

Base class of two different serial protocols.

`#include <cplSerial.h>`

Inheritance diagram for CCplSerial: Collaboration diagram for CCplSerial:

Protected Member Functions

- virtual bool load_tbl ()=0
  Loads the command table from the robot's firmware.
- virtual void defineProtocol (byte _kataddr)=0
  Defines the protocol's attributes.

Protected Attributes

- THeader hdr
  header
- TPacket cmd [256]
  command table
- byte send_buf [256]
  sending buffer
- byte read_buf [256]
  receive buffer

11.7.1 Detailed Description

Base class of two different serial protocols.

Definition at line 92 of file cplSerial.h.

11.7.2 Member Function Documentation

11.7.2.1 virtual bool CCplSerial::load_tbl () [protected, pure virtual]

Loads the command table from the robot's firmware.

Implemented in CCplSerialCRC.

11.7.2.2 virtual void CCplSerial::defineProtocol (byte *_kataddr*) [protected, pure virtual]

Defines the protocol's attributes.

Implemented in CCplSerialCRC.

11.7.3 Member Data Documentation

11.7.3.1 THeader CCplSerial::hdr [protected]

header

Definition at line 95 of file cplSerial.h.

11.7.3.2 TPacket CCplSerial::cmd[256] [protected]

command table

Definition at line 96 of file cplSerial.h.

11.7.3.3 byte CCplSerial::send_buf[256] [protected]

sending buffer

Definition at line 98 of file cplSerial.h.

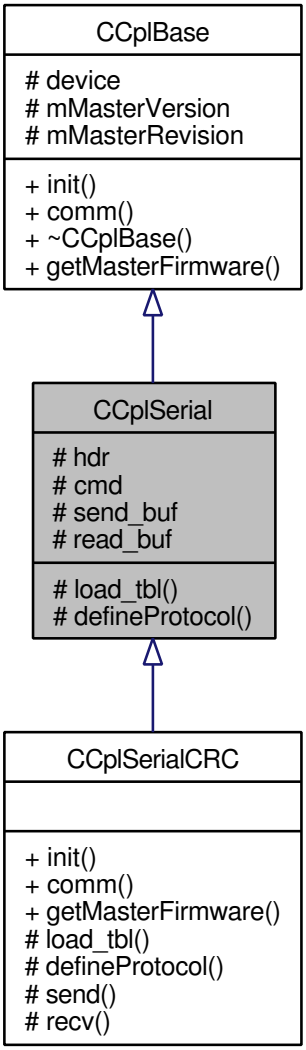
11.7.3.4 byte CCplSerial::read_buf[256] [protected]

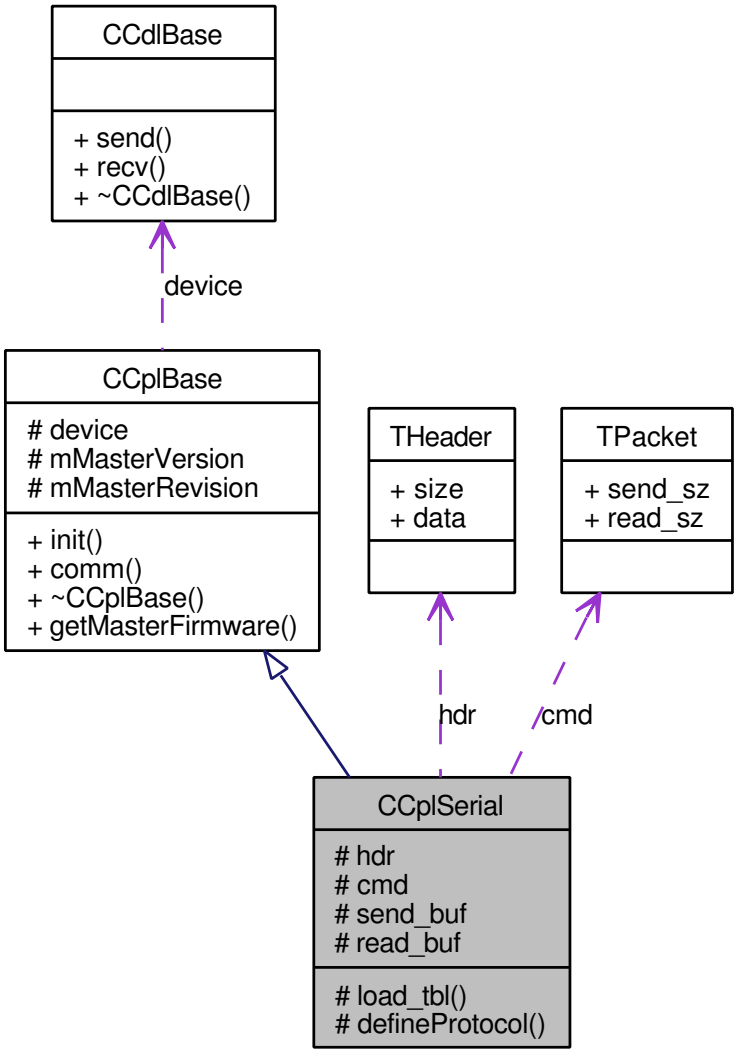
receive buffer

Definition at line 99 of file cplSerial.h.

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

- include/KNI/cplSerial.h


CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
CCplSerial
hdr
cmd
send_buf
read_buf
load_tbl()
defineProtocol()
CCplSerialCRC
+ init()
+ comm()
+ getMasterFirmware()
# load_tbl()
# defineProtocol()
# send()
# recv()


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
THeader
+ size
+ data
TPacket
+ send_sz
+ read_sz
hdr
cmd
CCplSerial
hdr
cmd
send_buf
read_buf
load_tbl()
defineProtocol()

11.8 CCplSerialCRC Class Reference

Implement the Serial-Zero protocol.

`#include <cplSerial.h>`

Inheritance diagram for CCplSerialCRC: Collaboration diagram for CCplSerialCRC:

Public Member Functions

- virtual bool init (CCdlBase *_device, byte _kataddr=24)
  Initializing function.
- virtual void comm (const byte *pack, byte *buf, byte *size)
  Communication function.
- virtual void getMasterFirmware (short *fw, short *rev)
  Get the master firmware of the robot we are communicating with.

Protected Member Functions

- virtual bool load_tbl ()
  Loads the command table from the robot's firmware.
- virtual void defineProtocol (byte _kataddr)
  Defines the protocol's attributes.
- virtual void send (byte *send_buf, byte write_sz, short retries=3)
- virtual void recv (byte *read_buf, byte read_sz, byte *size)

11.8.1 Detailed Description

Implement the Serial-Zero protocol.

Initializing function Init the protocols basic attributes.

Communication function Sends a communications packet and receives one from the robot.

Implement the Serial-CRC protocol

Definition at line 137 of file cplSerial.h.

11.8.2 Member Function Documentation

11.8.2.1 virtual bool CCplSerialCRC::load_tbl () [protected, virtual]

Loads the command table from the robot's firmware.

Implements CCplSerial.

11.8.2.2 virtual void CCplSerialCRC::defineProtocol (byte *_kataddr*) [protected, virtual]

Defines the protocol's attributes.

Implements CCplSerial.

11.8.2.3 virtual void CCplSerialCRC::send (byte * *send_buf*, byte *write_sz*, short *retries* = 3) [protected, virtual]

11.8.2.4 virtual void CCplSerialCRC::recv (byte * *read_buf*, byte *read_sz*, byte * *size*) [protected, virtual]

11.8.2.5 virtual bool CCplSerialCRC::init (CCdlBase * *_device*, byte *_kataddr* = 24) [virtual]

Initializing function.

Init the protocols basic attributes.

Implements CCplBase.

11.8.2.6 virtual void CCplSerialCRC::comm (const byte * *pack*, byte * *buf*, byte * *size*) [virtual]

Communication function.

Sends a communications packet and receives one from the robot.

Implements CCplBase.

11.8.2.7 virtual void CCplSerialCRC::getMasterFirmware (short * *fw*, short * *rev*) [virtual]

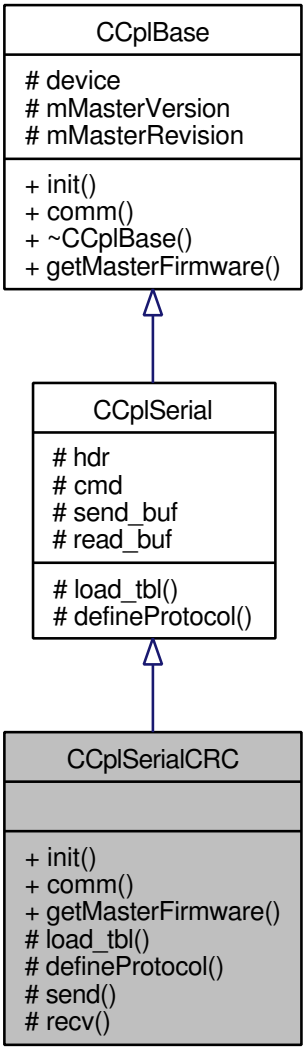
Get the master firmware of the robot we are communicating with.

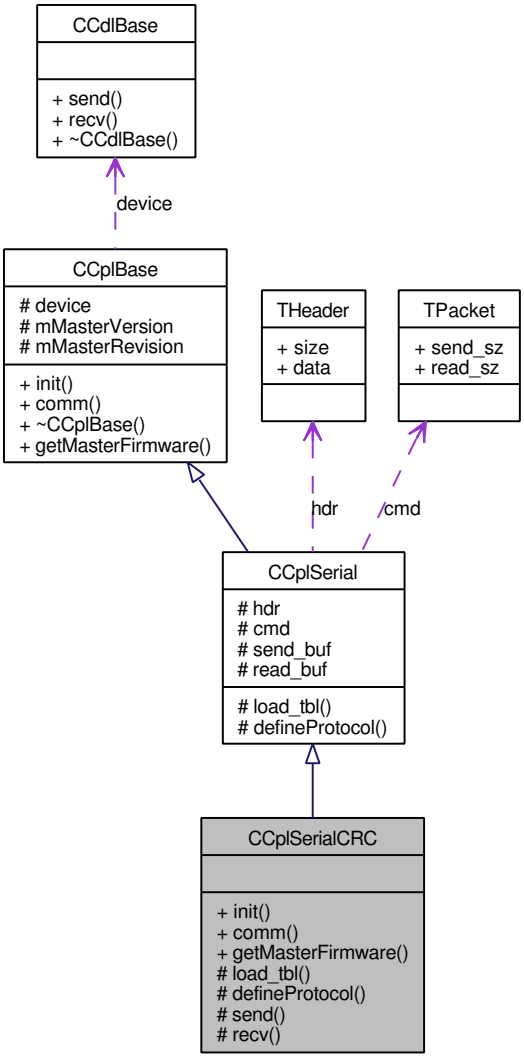
Get master firmware read at initialization time.

Implements CCplBase.

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

- include/KNI/cplSerial.h


CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
CCplSerial
hdr
cmd
send_buf
read_buf
load_tbl()
defineProtocol()
CCplSerialCRC
+ init()
+ comm()
+ getMasterFirmware()
# load_tbl()
# defineProtocol()
# send()
# recv()


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
THeader
+ size
+ data
TPacket
+ send_sz
+ read_sz
hdr
cmd
CCplSerial
hdr
cmd
send_buf
read_buf
load_tbl()
defineProtocol()
CCplSerialCRC
+ init()
+ comm()
+ getMasterFirmware()
# load_tbl()
# defineProtocol()
# send()
# recv()

11.9 CikBase Class Reference

`#include <ikBase.h>`

Inheritance diagram for CikBase: Collaboration diagram for CikBase:

Public Member Functions

- CikBase ()
- ~CikBase ()
- void getKinematicsVersion (std::vector< int > &version)

  Returns the version number of the kinematics used.

- void setTcpOffset (double xoff, double yoff, double zoff, double psioff)

  Set the offset from the flange to the desired tcp.

- void DKApos (double *position)

  Returns the current position of the robot in cartesian coordinates.

- void getCoordinates (double &x, double &y, double &z, double &phi, double &theta, double &psi, bool refreshEncoders=true)

  Returns the current position of the robot in cartesian coordinates.

- void getCoordinatesFromEncoders (std::vector< double > &pose, const std::vector< int > &encs)

  Returns the position of the robot corresponting to the given encoders in cartesian coordinates.

- void IKCalculate (double X, double Y, double Z, double Al, double Be, double Ga, std::vector< int >::iterator solution_iter)

  Calculates a set of encoders for the given coordinates.

- void IKCalculate (double X, double Y, double Z, double Al, double Be, double Ga, std::vector< int >::iterator solution_iter, const std::vector< int > &actualPosition)

  Calculates a set of encoders for the given coordinates.

- void IKGoto (double X, double Y, double Z, double Al, double Be, double Ga, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Moves to robot to given cartesian coordinates and euler-angles.

- void moveRobotTo (double x, double y, double z, double phi, double theta, double psi, bool waitUntilReached=false, int waitTimeout=TM_ENDLESS)

  Moves to robot to given cartesian coordinates and euler-angles.

- void moveRobotTo (std::vector< double > coordinates, bool waitUntilReached=false, int waitTimeout=TM_ENDLESS)

  This method does the same as the one above and is mainly provided for convenience.

Private Member Functions

- void _initKinematics ()

Private Attributes

- std::auto_ptr< KNI::KatanaKinematics > _kinematicsImpl
- bool _kinematicsIsInitialized

11.9.1 Detailed Description

Definition at line 49 of file ikBase.h.

11.9.2 Constructor & Destructor Documentation

11.9.2.1 CikBase::CikBase () [inline]

Definition at line 58 of file ikBase.h.

11.9.2.2 CikBase::~CikBase ()

11.9.3 Member Function Documentation

11.9.3.1 void CikBase::_initKinematics () [private]

11.9.3.2 void CikBase::getKinematicsVersion (std::vector< int > & *version*)

Returns the version number of the kinematics used.

Parameters:

version vector to write in version (major, minor, revision)

Note:

integrated analytical kinematics returns version 0.1.0
kinematics library returns versions >= 1.0.0

11.9.3.3 void CikBase::setTcpOffset (double *xoff*, double *yoff*, double *zoff*, double *psioff*)

Set the offset from the flange to the desired tcp.

Parameters:

xoff offset in x direction of flange coordinate system in m

yoff offset in y direction of flange coordinate system in m

zoff offset in z direction of flange coordinate system in m

psioff angle offset around x-axis of flange coordinate system in rad

11.9.3.4 void CikBase::DKApos (double * *position*)

Returns the current position of the robot in cartesian coordinates.

Note:

This method is deprecated, please use getCoordinates(...) instead

11.9.3.5 void CikBase::getCoordinates (double & *x*, double & *y*, double & *z*, double & *phi*, double & *theta*, double & *psi*, bool *refreshEncoders* = `true`)

Returns the current position of the robot in cartesian coordinates.

Parameters:

refreshEncoders With this parameter you can determine if the method reads the actual encoders from the robot or if it will use the cached ones

Note:

This function returns a tuple in python

11.9.3.6 void CikBase::getCoordinatesFromEncoders (std::vector< double > & *pose*, const std::vector< int > & *encs*)

Returns the position of the robot corresponting to the given encoders in cartesian coordinates.

11.9.3.7 void CikBase::IKCalculate (double *X*, double *Y*, double *Z*, double *Al*, double *Be*, double *Ga*, std::vector< int >::iterator *solution_iter*)

Calculates a set of encoders for the given coordinates.

This method reads the current encoders from the robot and involves therefore also communication to the robot

11.9.3.8 void CikBase::IKCalculate (double *X*, double *Y*, double *Z*, double *Al*, double *Be*, double *Ga*, std::vector< int >::iterator *solution_iter*, const std::vector< int > & *actualPosition*)

Calculates a set of encoders for the given coordinates.

For this method you have to pass an actualPosition too. No communication with the robot will be done here.

11.9.3.9 void CikBase::IKGoto (double *X*, double *Y*, double *Z*, double *Al*, double *Be*, double *Ga*, bool *wait* = `false`, int *tolerance* = `100`, long *timeout* = `TM_ENDLESS`)

Moves to robot to given cartesian coordinates and euler-angles.

Note:

This method is deprecated, please use moveRobotTo(...) instead

11.9.3.10 void CikBase::moveRobotTo (double *x*, double *y*, double *z*, double *phi*, double *theta*, double *psi*, bool *waitUntilReached* = `false`, int *waitTimeout* = `TM_ENDLESS`)

Moves to robot to given cartesian coordinates and euler-angles.

Note:

Instead of a given tolerance, a default tolerance is being used

Reimplemented in CLMBase.

11.9.3.11 void CikBase::moveRobotTo (std::vector< double > *coordinates*, bool *waitUntilReached* = false, int *waitTimeout* = TM_ENDLESS)

This method does the same as the one above and is mainly provided for convenience.

Note:

You can call this function in python using tuples: Example: katana.moveRobotTo((x,y,z,phi,theta,psi))
If the size of the container is smaller than 6, it will throw an exception

Reimplemented in CLMBase.

11.9.4 Member Data Documentation

11.9.4.1 std::auto_ptr<KNI::KatanaKinematics> CikBase::_kinematicsImpl [private]

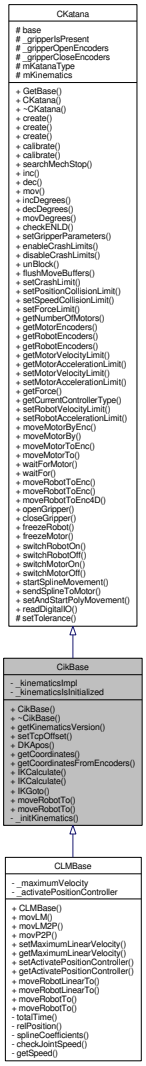
Definition at line 52 of file ikBase.h.

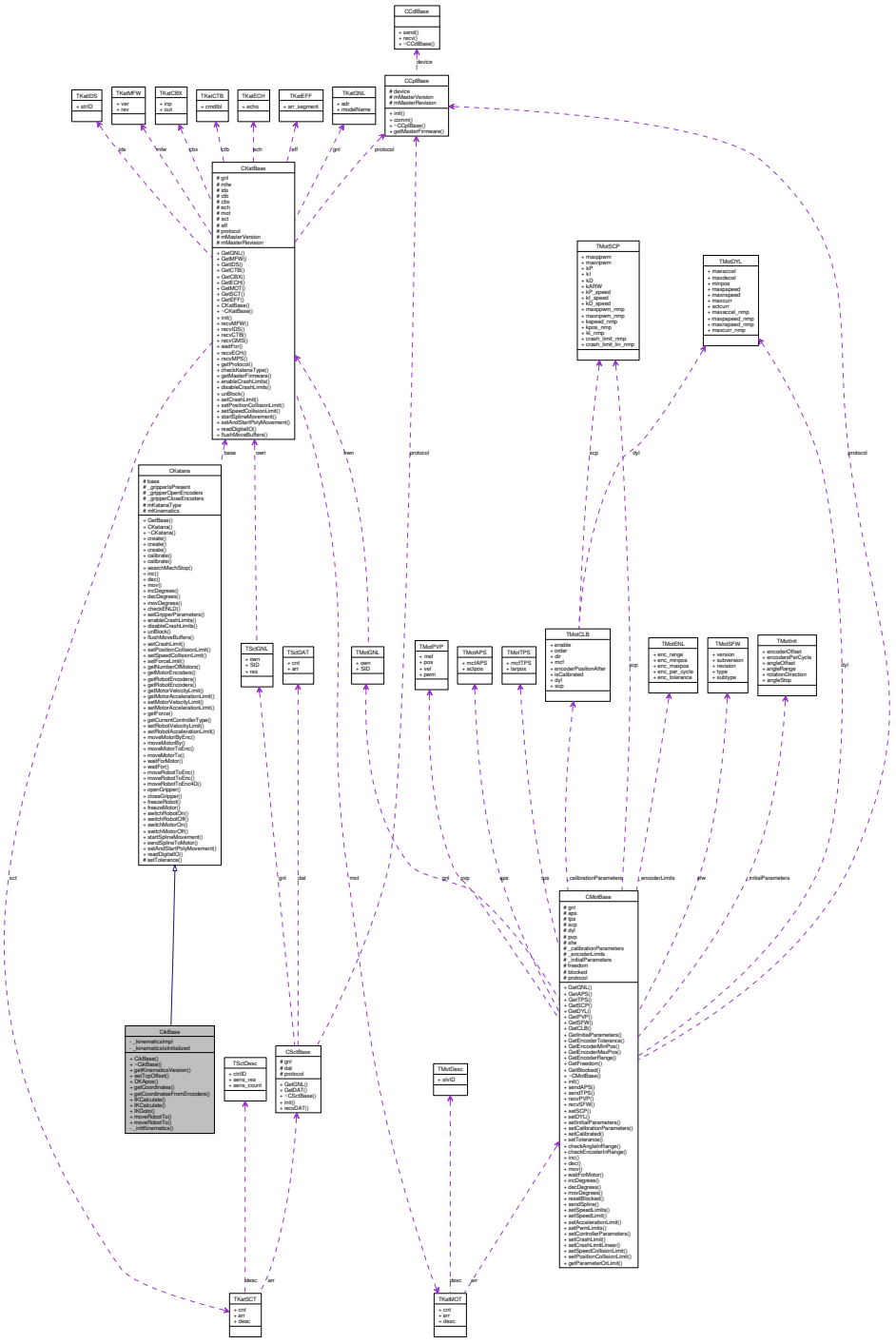
11.9.4.2 bool CikBase::_kinematicsIsInitialized [private]

Definition at line 53 of file ikBase.h.

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

- include/KNI_InvKin/ikBase.h


CKatana
base
_gripperIsPresent
_gripperOpenEncoders
_gripperCloseEncoders
mKatanaType
mKinematics
+ GetBase()
+ CKatana()
+ ~CKatana()
+ create()
+ create()
+ create()
+ calibrate()
+ calibrate()
+ searchMechStop()
+ inc()
+ dec()
+ mov()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ checkENLD()
+ setGripperParameters()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ flushMoveBuffers()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ setForceLimit()
+ getNumberOfMotors()
+ getMotorEncoders()
+ getRobotEncoders()
+ getRobotEncoders()
+ getMotorVelocityLimit()
+ getMotorAccelerationLimit()
+ setMotorVelocityLimit()
+ setMotorAccelerationLimit()
+ getForce()
+ getCurrentControllerType()
+ setRobotVelocityLimit()
+ setRobotAccelerationLimit()
+ moveMotorByEnc()
+ moveMotorBy()
+ moveMotorToEnc()
+ moveMotorTo()
+ waitForMotor()
+ waitFor()
+ moveRobotToEnc()
+ moveRobotToEnc()
+ moveRobotToEnc4D()
+ openGripper()
+ closeGripper()
+ freezeRobot()
+ freezeMotor()
+ switchRobotOn()
+ switchRobotOff()
+ switchMotorOn()
+ switchMotorOff()
+ startSplineMovement()
+ sendSplineToMotor()
+ setAndStartPolyMovement()
+ readDigitalIO()
# setTolerance()
CikBase
- _kinematicsImpl
- _kinematicsIsInitialized
+ CikBase()
+ ~CikBase()
+ getKinematicsVersion()
+ setTcpOffset()
+ DKApos()
+ getCoordinates()
+ getCoordinatesFromEncoders()
+ IKCalculate()
+ IKCalculate()
+ IKGoto()
+ moveRobotTo()
+ moveRobotTo()
- _initKinematics()
CLMBase
- _maximumVelocity
- _activatePositionController
+ CLMBase()
+ movLM()
+ movLM2P()
+ movP2P()
+ setMaximumLinearVelocity()
+ getMaximumLinearVelocity()
+ setActivatePositionController()
+ getActivatePositionController()
+ moveRobotLinearTo()
+ moveRobotLinearTo()
+ moveRobotTo()
+ moveRobotTo()
- totalTime()
- relPosition()
- splineCoefficients()
- checkJointSpeed()
- getSpeed()



11.10 CKatana Class Reference

Extended Katana class with additional functions.

`#include <kmlExt.h>`

Inheritance diagram for CKatana: Collaboration diagram for CKatana:

Public Member Functions

- CKatBase * GetBase ()

  Returns pointer to 'CKatBase'.*

- CKatana ()

  Constructor.

- ~CKatana ()

  Destructor.

- void create (const char *configurationFile, CCplBase *protocol)

  Create routine.

- void create (KNI::kmlFactory *infos, CCplBase *protocol)
- void create (TKatGNL &gnl, TKatMOT &mot, TKatSCT &sct, TKatEFF &eff, CCplBase *protocol)

  Create routine.

- void calibrate ()
- void calibrate (long idx, TMotCLB clb, TMotSCP scp, TMotDYL dyl)
- void searchMechStop (long idx, TSearchDir dir, TMotSCP scp, TMotDYL dyl)
- void inc (long idx, int dif, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Increments the motor specified by an index postion in encoders.

- void dec (long idx, int dif, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Decrements the motor specified by an index postion in encoders.

- void mov (long idx, int tar, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Moves the motor specified by an index to a given target position in encoders.

- void incDegrees (long idx, double dif, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Increments the motor specified by an index postion in degree units.

- void decDegrees (long idx, double dif, bool wait=false, int tolerance=100, long timeout=TM_-ENDLESS)

  Decrements the motor specified by an index postion in degree units.

- void movDegrees (long idx, double tar, bool wait=false, int tolerance=100, long timeout=TM_-ENDLESS)

  Moves the motor specified by an index to a given target position in degree units.

- bool checkENLD (long idx, double degrees)

  Check if the absolute position in degrees is out of range.

- void setGripperParameters (bool isPresent, int openEncoders, int closeEncoders)

Tell the robot about the presence of a gripper.

- void enableCrashLimits ()
  crash limits enable
- void disableCrashLimits ()
  crash limits disable
- void unBlock ()
  unblock robot after a crash
- void flushMoveBuffers ()
  flush move buffers
- void setCrashLimit (long idx, int limit)
  unblock robot after a crash
- void setPositionCollisionLimit (long idx, int limit)
  set collision position limits
- void setSpeedCollisionLimit (long idx, int limit)
  set collision speed limits
- void setForceLimit (int axis, int limit)
  Set the force limit for the current controller.
- short getNumberOfMotors () const
- int getMotorEncoders (short number, bool refreshEncoders=true) const
- std::vector< int >::iterator getRobotEncoders (std::vector< int >::iterator start, std::vector< int >::const_iterator end, bool refreshEncoders=true) const
  Write the cached encoders into the container.
- std::vector< int > getRobotEncoders (bool refreshEncoders=true) const
  Get the current robot encoders as a vector-container.
- short getMotorVelocityLimit (short number) const
- short getMotorAccelerationLimit (short number) const
- void setMotorVelocityLimit (short number, short velocity)
- void setMotorAccelerationLimit (short number, short acceleration)
- short getForce (int axis)
  Get the current force.
- int getCurrentControllerType (int axis)
  Get the axis controller type.
- void setRobotVelocityLimit (short velocity)
- void setRobotAccelerationLimit (short acceleration)
  Set the velocity of all motors together.
- void moveMotorByEnc (short number, int encoders, bool waitUntilReached=false, int waitTimeout=0)
- void moveMotorBy (short number, double radianAngle, bool waitUntilReached=false, int waitTimeout=0)
- void moveMotorToEnc (short number, int encoders, bool waitUntilReached=false, int encTolerance=100, int waitTimeout=0)

- void moveMotorTo (short number, double radianAngle, bool waitUntilReached=false, int waitTimeout=0)
- void waitForMotor (short number, int encoders, int encTolerance=100, short mode=0, int waitTimeout=5000)
- void waitFor (TMotStsFlg status, int waitTimeout)
- void moveRobotToEnc (std::vector< int >::const_iterator start, std::vector< int >::const_iterator end, bool waitUntilReached=false, int encTolerance=100, int waitTimeout=0)

  Move to robot to given encoders.

- void moveRobotToEnc (std::vector< int > encoders, bool waitUntilReached=false, int encTolerance=100, int waitTimeout=0)

  Move to robot to given encoders in the vector-container.

- void moveRobotToEnc4D (std::vector< int > target, int velocity=180, int acceleration=1, int encTolerance=100)

  Move to robot to given target in the vector-container with the given velocity, acceleration and tolerance.

- void openGripper (bool waitUntilReached=false, int waitTimeout=100)
- void closeGripper (bool waitUntilReached=false, int waitTimeout=100)
- void freezeRobot ()
- void freezeMotor (short number)
- void switchRobotOn ()
- void switchRobotOff ()
- void switchMotorOn (short number)
- void switchMotorOff (short number)
- void startSplineMovement (bool exactflag, int moreflag=1)

  Start a spline movement.

- void sendSplineToMotor (unsigned short number, short targetPosition, short duration, short p1, short p2, short p3, short p4)

  Send one spline to the motor.

- void setAndStartPolyMovement (std::vector< short > polynomial, bool exactflag, int moreflag)

  Send polynomials to all motors and start movement.

- int readDigitalIO ()

  Read The Digital I/Os.

Protected Member Functions

- void setTolerance (long idx, int enc_tolerance)

  Sets the tolerance range in encoder units for the robots movements.

Protected Attributes

- CKatBase * base

  base katana

- bool _gripperIsPresent
- int _gripperOpenEncoders

- int _gripperCloseEncoders
- int mKatanaType

  The type of KatanaXXX (300 or 400).

- int mKinematics

  The kinematics implementation: 0 for Analytical, 1 for RobAnaGuess.

11.10.1 Detailed Description

Extended Katana class with additional functions.

This class uses the 'CKatBase* base' object to refer to a Katana robot.

Definition at line 64 of file kmlExt.h.

11.10.2 Constructor & Destructor Documentation

11.10.2.1 CKatana::CKatana () [inline]

Constructor.

Definition at line 88 of file kmlExt.h.

11.10.2.2 CKatana::~CKatana () [inline]

Destructor.

Definition at line 91 of file kmlExt.h.

11.10.3 Member Function Documentation

11.10.3.1 void CKatana::setTolerance (long *idx*, int *enc_tolerance*) [protected]

Sets the tolerance range in encoder units for the robots movements.

11.10.3.2 CKatBase* CKatana::GetBase () [inline]

Returns pointer to 'CKatBase*'.

Definition at line 83 of file kmlExt.h.

11.10.3.3 void CKatana::create (const char * *configurationFile*, CCplBase * *protocol*)

Create routine.

11.10.3.4 void CKatana::create (KNI::kmlFactory * *infos*, CCplBase * *protocol*)

11.10.3.5 void CKatana::create (TKatGNL & *gnl*, TKatMOT & *mot*, TKatSCT & *sct*, TKatEFF & *eff*, CCplBase * *protocol*)

Create routine.

Parameters:

gnl katana initial attributes

mot motor initial attributes

sct sensor controller initial attributes

eff end effector initial attributes

protocol protocol to be used

11.10.3.6 void CKatana::calibrate ()

11.10.3.7 void CKatana::calibrate (long *idx*, TMotCLB *clb*, TMotSCP *scp*, TMotDYL *dyl*)

Parameters:

idx motor index

clb calibration struct for one motor

scp static controller parameters

dyl dynamic controller parameters

11.10.3.8 void CKatana::searchMechStop (long *idx*, TSearchDir *dir*, TMotSCP *scp*, TMotDYL *dyl*)

Parameters:

idx motor index

dir search direction

scp static controller parameters

dyl dynamic controller parameters

11.10.3.9 void CKatana::inc (long *idx*, int *dif*, bool *wait* = `false`, int *tolerance* = `100`, long *timeout* = `TM_ENDLESS`)

Increments the motor specified by an index postion in encoders.

11.10.3.10 void CKatana::dec (long *idx*, int *dif*, bool *wait* = `false`, int *tolerance* = `100`, long *timeout* = `TM_ENDLESS`)

Decrements the motor specified by an index postion in encoders.

11.10.3.11 void CKatana::mov (long *idx*, int *tar*, bool *wait* = `false`, int *tolerance* = `100`, long *timeout* = `TM_ENDLESS`)

Moves the motor specified by an index to a given target position in encoders.

11.10.3.12 void CKatana::incDegrees (long *idx*, double *dif*, bool *wait* = `false`, int *tolerance* = `100`, long *timeout* = `TM_ENDLESS`)

Increments the motor specified by an index postion in degree units.

11.10.3.13 void CKatana::decDegrees (long *idx*, double *dif*, bool *wait* = false, int *tolerance* = 100, long *timeout* = TM_ENDLESS)

Decrements the motor specified by an index postion in degree units.

11.10.3.14 void CKatana::movDegrees (long *idx*, double *tar*, bool *wait* = false, int *tolerance* = 100, long *timeout* = TM_ENDLESS)

Moves the motor specified by an index to a given target position in degree units.

11.10.3.15 bool CKatana::checkENLD (long *idx*, double *degrees*)

Check if the absolute position in degrees is out of range.

11.10.3.16 void CKatana::setGripperParameters (bool *isPresent*, int *openEncoders*, int *closeEncoders*)

Tell the robot about the presence of a gripper.

Parameters:

openEncoders Which encoders should be used as target positions for opening the gripper

closeEncoders Dito for closing the gripper

11.10.3.17 void CKatana::enableCrashLimits ()

crash limits enable

11.10.3.18 void CKatana::disableCrashLimits ()

crash limits disable

11.10.3.19 void CKatana::unBlock ()

unblock robot after a crash

11.10.3.20 void CKatana::flushMoveBuffers ()

flush move buffers

11.10.3.21 void CKatana::setCrashLimit (long *idx*, int *limit*)

unblock robot after a crash

11.10.3.22 void CKatana::setPositionCollisionLimit (long *idx*, int *limit*)

set collision position limits

11.10.3.23 void CKatana::setSpeedCollisionLimit (long *idx*, int *limit*)

set collision speed limits

11.10.3.24 void CKatana::setForceLimit (int *axis*, int *limit*)

Set the force limit for the current controller.

Parameters:

axis axis number 1-6. If 0, send limit to all axes

11.10.3.25 short CKatana::getNumberOfMotors () const

11.10.3.26 int CKatana::getMotorEncoders (short *number*, bool *refreshEncoders* = `true`) const

11.10.3.27 std::vector<int>::iterator CKatana::getRobotEncoders (std::vector< int >::iterator *start*, std::vector< int >::const_iterator *end*, bool *refreshEncoders* = `true`) const

Write the cached encoders into the container.

Set refreshEncoders=true if the KNI should fetch them from the robot. If m=distance(start, end) is smaller than the number of motors, only the first m motors will be written to the container, the function will not throw an exception because of this. The return value will point to one element after the last one.

11.10.3.28 std::vector<int> CKatana::getRobotEncoders (bool *refreshEncoders* = `true`) const

Get the current robot encoders as a vector-container.

This method is mainly provided for convenience. It is easier than the other getRobotEncoders method but probably not so efficient. It is much easier to use via the wrappers.

11.10.3.29 short CKatana::getMotorVelocityLimit (short *number*) const

11.10.3.30 short CKatana::getMotorAccelerationLimit (short *number*) const

11.10.3.31 void CKatana::setMotorVelocityLimit (short *number*, short *velocity*)

11.10.3.32 void CKatana::setMotorAccelerationLimit (short *number*, short *acceleration*)

11.10.3.33 short CKatana::getForce (int *axis*)

Get the current force.

11.10.3.34 int CKatana::getCurrentControllerType (int *axis*)

Get the axis controller type.

Returns:

0 for position controller, 1 for current controller

11.10.3.35 void CKatana::setRobotVelocityLimit (short *velocity*)

11.10.3.36 void CKatana::setRobotAccelerationLimit (short *acceleration*)

Set the velocity of all motors together.

This does not set the velocity of the TCP.

11.10.3.37 void CKatana::moveMotorByEnc (short *number*, int *encoders*, bool *waitUntilReached* = false, int *waitTimeout* = 0)

11.10.3.38 void CKatana::moveMotorBy (short *number*, double *radianAngle*, bool *waitUntilReached* = false, int *waitTimeout* = 0)

11.10.3.39 void CKatana::moveMotorToEnc (short *number*, int *encoders*, bool *waitUntilReached* = false, int *encTolerance* = 100, int *waitTimeout* = 0)

11.10.3.40 void CKatana::moveMotorTo (short *number*, double *radianAngle*, bool *waitUntilReached* = false, int *waitTimeout* = 0)

11.10.3.41 void CKatana::waitForMotor (short *number*, int *encoders*, int *encTolerance* = 100, short *mode* = 0, int *waitTimeout* = 5000)

11.10.3.42 void CKatana::waitFor (TMotStsFlg *status*, int *waitTimeout*)

11.10.3.43 void CKatana::moveRobotToEnc (std::vector< int >::const_iterator *start*, std::vector< int >::const_iterator *end*, bool *waitUntilReached* = false, int *encTolerance* = 100, int *waitTimeout* = 0)

Move to robot to given encoders.

You can provide less values than the number of motors. In that case only the given ones will be moved. This can be usefull in cases where you want to move the robot but you don't want to move the gripper.

11.10.3.44 void CKatana::moveRobotToEnc (std::vector< int > *encoders*, bool *waitUntilReached* = false, int *encTolerance* = 100, int *waitTimeout* = 0)

Move to robot to given encoders in the vector-container.

This method is mainly provided for convenience. Catch by value (and not by reference) is intended to avoid nasty wrapping code.

11.10.3.45 void CKatana::moveRobotToEnc4D (std::vector< int > *target*, int *velocity* = 180, int *acceleration* = 1, int *encTolerance* = 100)

Move to robot to given target in the vector-container with the given velocity, acceleration and tolerance.

11.10.3.46 void CKatana::openGripper (bool *waitUntilReached* = false, int *waitTimeout* = 100)

11.10.3.47 void CKatana::closeGripper (bool *waitUntilReached* = false, int *waitTimeout* = 100)

11.10.3.48 void CKatana::freezeRobot ()

11.10.3.49 void CKatana::freezeMotor (short *number*)

11.10.3.50 void CKatana::switchRobotOn ()

11.10.3.51 void CKatana::switchRobotOff ()

11.10.3.52 void CKatana::switchMotorOn (short *number*)

11.10.3.53 void CKatana::switchMotorOff (short *number*)

11.10.3.54 void CKatana::startSplineMovement (bool *exactflag*, int *moreflag* = 1)

Start a spline movement.

Parameters:

exactflag Set it to true if you want the position controller activated after the movement

moreflag 0 = start moving more following, 1 = last or a single polynomial movement, 2 = do not start moving yet more following

11.10.3.55 void CKatana::sendSplineToMotor (unsigned short *number*, short *targetPosition*, short *duration*, short *p1*, short *p2*, short *p3*, short *p4*)

Send one spline to the motor.

Parameters:

duration Duration has to be given in 10ms units

11.10.3.56 void CKatana::setAndStartPolyMovement (std::vector< short > *polynomial*, bool *exactflag*, int *moreflag*)

Send polynomials to all motors and start movement.

Parameters:

polynomial time, target pos and coefficients of the motor polynomials

exactflag exactflag

moreflag 0 = start moving more following, 1 = last or a single polynomial movement, 2 = do not start moving yet more following

11.10.3.57 int CKatana::readDigitalIO ()

Read The Digital I/Os.

11.10.4 Member Data Documentation

11.10.4.1 CKatBase* CKatana::base [protected]

base katana

Definition at line 67 of file kmlExt.h.

11.10.4.2 bool CKatana::_gripperIsPresent [protected]

Definition at line 69 of file kmlExt.h.

11.10.4.3 int CKatana::_gripperOpenEncoders [protected]

Definition at line 70 of file kmlExt.h.

11.10.4.4 int CKatana::_gripperCloseEncoders [protected]

Definition at line 71 of file kmlExt.h.

11.10.4.5 int CKatana::mKatanaType [protected]

The type of KatanaXXX (300 or 400).

Definition at line 73 of file kmlExt.h.

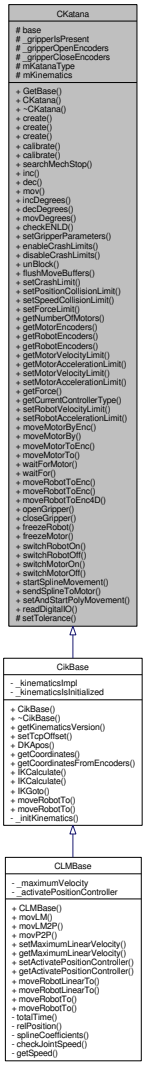
11.10.4.6 int CKatana::mKinematics [protected]

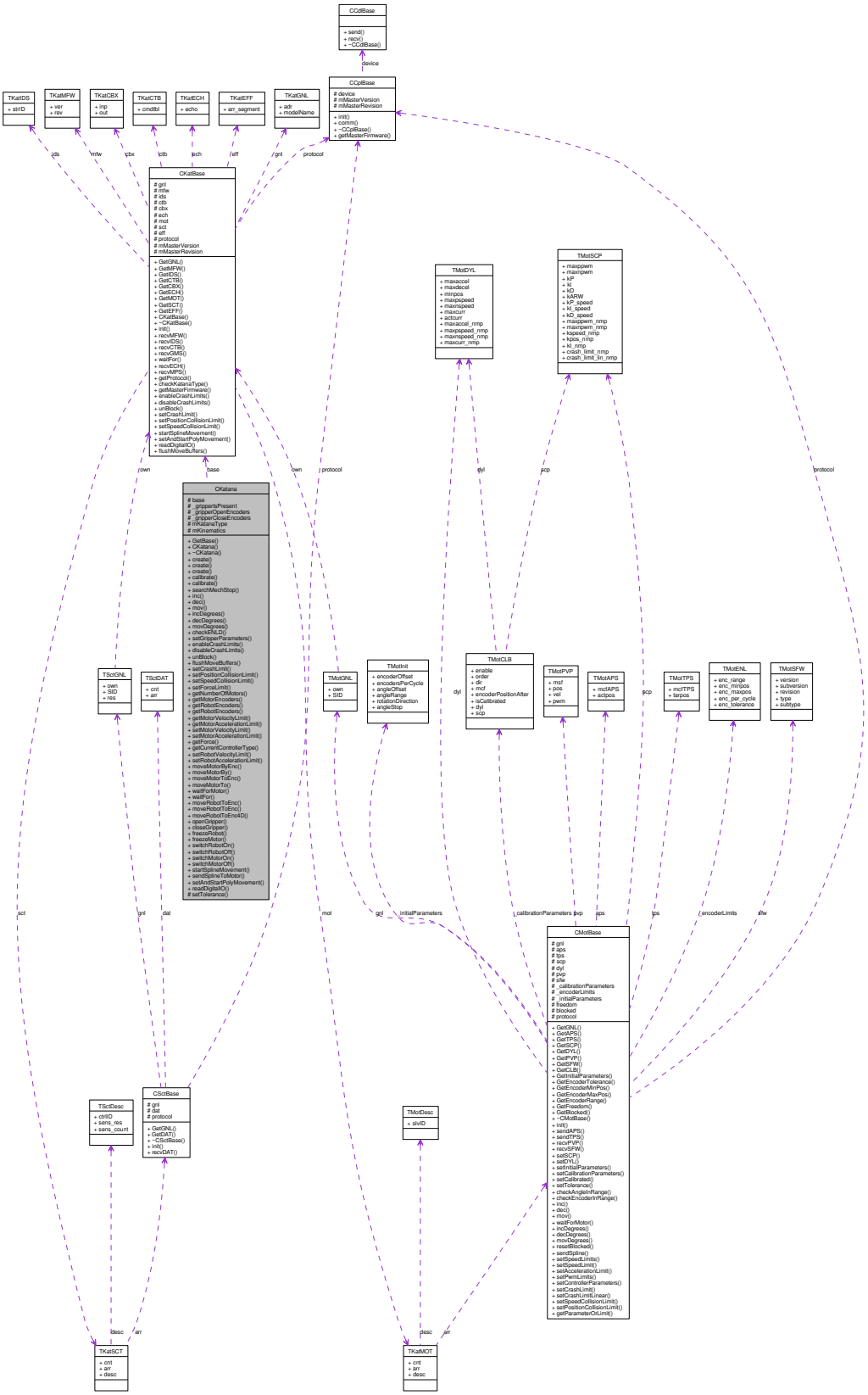
The kinematics implementation: 0 for Analytical, 1 for RobAnaGuess.

Definition at line 75 of file kmlExt.h.

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

- include/KNI/kmlExt.h


CKatana
base
_gripperIsPresent
_gripperOpenEncoders
_gripperCloseEncoders
mKatanaType
mKinematics
+ GetBase()
+ CKatana()
+ ~CKatana()
+ create()
+ create()
+ create()
+ calibrate()
+ calibrate()
+ searchMechStop()
+ inc()
+ dec()
+ mov()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ checkENLD()
+ setGripperParameters()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ flushMoveBuffers()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ setForceLimit()
+ getNumberOfMotors()
+ getMotorEncoders()
+ getRobotEncoders()
+ getRobotEncoders()
+ getMotorVelocityLimit()
+ getMotorAccelerationLimit()
+ setMotorVelocityLimit()
+ setMotorAccelerationLimit()
+ getForce()
+ getCurrentControllerType()
+ setRobotVelocityLimit()
+ setRobotAccelerationLimit()
+ moveMotorByEnc()
+ moveMotorBy()
+ moveMotorToEnc()
+ moveMotorTo()
+ waitForMotor()
+ waitFor()
+ moveRobotToEnc()
+ moveRobotToEnc()
+ moveRobotToEnc4D()
+ openGripper()
+ closeGripper()
+ freezeRobot()
+ freezeMotor()
+ switchRobotOn()
+ switchRobotOff()
+ switchMotorOn()
+ switchMotorOff()
+ startSplineMovement()
+ sendSplineToMotor()
+ setAndStartPolyMovement()
+ readDigitalIO()
# setTolerance()
CikBase
- _kinematicsImpl
- _kinematicsIsInitialized
+ CikBase()
+ ~CikBase()
+ getKinematicsVersion()
+ setTcpOffset()
+ DKApos()
+ getCoordinates()
+ getCoordinatesFromEncoders()
+ IKCalculate()
+ IKCalculate()
+ IKGoto()
+ moveRobotTo()
+ moveRobotTo()
- _initKinematics()
CLMBase
- _maximumVelocity
- _activatePositionController
+ CLMBase()
+ movLM()
+ movLM2P()
+ movP2P()
+ setMaximumLinearVelocity()
+ getMaximumLinearVelocity()
+ setActivatePositionController()
+ getActivatePositionController()
+ moveRobotLinearTo()
+ moveRobotLinearTo()
+ moveRobotTo()
+ moveRobotTo()
- totalTime()
- relPosition()
- splineCoefficients()
- checkJointSpeed()
- getSpeed()


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
TKatIDS
+ strID
TKatIMFW
+ ver
+ rev
TKatICBX
+ inp
+ out
TKatICTB
+ cmdBbl
TKatIECH
+ echo
TKatIEFF
+ arr_segment
TKatIGNL
+ adr
+ modelName
ids
mfw
cbx
ctb
ech
eff
gnl
protocol
CKatBase
gnl
mfw
ids
ctb
cbx
ech
mot
scl
eff
protocol
mMasterVersion
mMasterRevision
+ GetGNL()
+ GetMFW()
+ GetIDS()
+ GetCTB()
+ GetCBX()
+ GetECH()
+ GetMOT()
+ GetSCT()
+ GetEFF()
+ CKatBase()
+ ~CKatBase()
+ init()
+ recvMFW()
+ recvIDS()
+ recvCTB()
+ recvGMS()
+ waitFor()
+ recvECH()
+ recvMPS()
+ getProtocol()
+ checkKatanaType()
+ getMasterFirmware()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ startSplineMovement()
+ setAndStartPolyMovement()
+ readDigitalIO()
+ flushMoveBuffers()
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxrspeed
+ maxcurr
+ acbcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxrspeed_nmp
+ maxcurr_nmp
TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kARW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ kI_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
own
base
own
protocol
dyl
scp
protocol
CKatana
base
_gripperIsPresent
_gripperOpenEncoders
_gripperCloseEncoders
mKatanaType
mKinematics
+ GetBase()
+ CKatana()
+ ~CKatana()
+ create()
+ create()
+ create()
+ calibrate()
+ calibrate()
+ searchMechStop()
+ inc()
+ dec()
+ mov()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ checkENLD()
+ setGripperParameters()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ flushMoveBuffers()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ setForceLimit()
+ getNumberOfMotors()
+ getMotorEncoders()
+ getRobotEncoders()
+ getRobotEncoders()
+ getMotorVelocityLimit()
+ getMotorAccelerationLimit()
+ setMotorVelocityLimit()
+ setMotorAccelerationLimit()
+ getForce()
+ getCurrentControllerType()
+ setRobotVelocityLimit()
+ setRobotAccelerationLimit()
+ moveMotorByEnc()
+ moveMotorBy()
+ moveMotorToEnc()
+ moveMotorTo()
+ waitForMotor()
+ waitFor()
+ moveRobotToEnc()
+ moveRobotToEnc()
+ moveRobotToEnc4D()
+ openGripper()
+ closeGripper()
+ freazeRobot()
+ freazeMotor()
+ switchRobotOn()
+ switchRobotOff()
+ switchMotorOn()
+ switchMotorOff()
+ startSplineMovement()
+ sendSplineToMotor()
+ setAndStartPolyMovement()
+ readDigitalIO()
# setTolerance()
TSctGNL
+ own
+ SID
+ res
TSctDAT
+ cnt
+ arr
TMotGNL
+ own
+ SID
TMotInit
+ encoderOffset
+ encodersPerCycle
+ angleOffset
+ angleRange
+ rotationDirection
+ angleStop
TMotCLB
+ enable
+ order
+ cbr
+ mcf
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp
dyl
TMotPVP
+ mot
+ pos
+ vel
+ pwm
TMotAPS
+ mctAPS
+ actpos
scp
TMotTPS
+ mctTPS
+ tarpos
TMotENL
+ enc_range
+ enc_minpos
+ enc_maxpos
+ enc_per_cycle
+ enc_tolerance
TMotSFW
+ version
+ subversion
+ revision
+ type
+ subtype
sct
gnl
dat
mot
gpl
initialParameters
calibrationParameters
pvp
aps
tps
encoderLimits
sfw
CMotBase
gnl
aps
tps
scp
dyl
pvp
stw
_calibrationParameters
_encoderLimits
_initialParameters
freedom
blocked
protocol
+ GetGNL()
+ GetAPS()
+ GetTPS()
+ GetSCP()
+ GetDYL()
+ GetPVP()
+ GetSFW()
+ GetCLB()
+ GetInitialParameters()
+ GetEncoderTolerance()
+ GetEncoderMinPos()
+ GetEncoderMaxPos()
+ GetEncoderRange()
+ GetFreedom()
+ GetBlocked()
+ ~CMotBase()
+ init()
+ sendAPS()
+ sendTPS()
+ recvPVP()
+ recvSFW()
+ setSCP()
+ setDYL()
+ setInitialParameters()
+ setCalibrationParameters()
+ setCalibrated()
+ setTolerance()
+ checkAngleInRange()
+ checkEncoderInRange()
+ inc()
+ dec()
+ mov()
+ waitForMotor()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ resetBlocked()
+ sendSpline()
+ setSpeedLimits()
+ setSpeedLimit()
+ setAccelerationLimit()
+ setPwmLimits()
+ setControllerParameters()
+ setCrashLimit()
+ setCrashLimitLinear()
+ setSpeedCollisionLimit()
+ setPositionCollisionLimit()
+ getParameterOrLimit()
TSctDesc
+ ctrlID
+ sens_res
+ sens_count
CSctBase
gnl
dat
protocol
+ GetGNL()
+ GetDAT()
+ ~CSctBase()
+ init()
+ recvDAT()
TMotDesc
+ slvID
ldesc
arr
ldesc
arr
TKatISCT
+ cnt
+ arr
+ desc
TKatIMOT
+ cnt
+ arr
+ desc

11.11 CKatBase Class Reference

Base Katana class.

`#include <kmlBase.h>`

Collaboration diagram for CKatBase:

Public Member Functions

- const TKatGNL * GetGNL ()
  Get a pointer to the desired structure.
- const TKatMFW * GetMFW ()
  Get a pointer to the desired structure.
- const TKatIDS * GetIDS ()
  Get a pointer to the desired structure.
- const TKatCTB * GetCTB ()
  Get a pointer to the desired structure.
- const TKatCBX * GetCBX ()
  Get a pointer to the desired structure.
- const TKatECH * GetECH ()
  Get a pointer to the desired structure.
- const TKatMOT * GetMOT ()
  Get a pointer to the desired structure.
- const TKatSCT * GetSCT ()
  Get a pointer to the desired structure.
- TKatEFF * GetEFF ()
  Get a pointer to the desired structure.
- CKatBase ()
- virtual ~CKatBase ()
  destructor
- virtual bool init (const TKatGNL _gnl, const TKatMOT _mot, const TKatSCT _sct, const TKatEFF _eff, CCplBase *_protocol)
- void recvMFW ()
  receive data
- void recvIDS ()
  receive data
- void recvCTB ()
  receive data
- void recvGMS ()

receive data

- void waitFor (TMotStsFlg status, int waitTimeout, bool gripper)
  wait for motor on all motors
- void recvECH ()
  receive data
- void recvMPS ()
  read all motor positions simultaneously
- CCplBase * getProtocol ()
  get a handle of the protocol, used in CKatana
- int checkKatanaType (int type)
  checks for a K300 or K400
- void getMasterFirmware (short *fw, short *rev)
  Get the master firmware of the robot we are communicating with.
- void enableCrashLimits ()
  crash limits enable
- void disableCrashLimits ()
  crash limits disable
- void unBlock ()
  unblock robot after a crash
- void setCrashLimit (long idx, int limit)
  set collision limits
- void setPositionCollisionLimit (long idx, int limit)
  set collision position limits
- void setSpeedCollisionLimit (long idx, int limit)
  set collision speed limits
- void startSplineMovement (int exactflag, int moreflag=1)
  Start a spline movement.
- void setAndStartPolyMovement (std::vector< short > polynomial, int exactflag, int moreflag)
  Send polynomials to all motors and start movement.
- int readDigitalIO ()
  get digital I/O data from Katana400:
- int flushMoveBuffers ()
  flush move buffers on all motors:

Protected Attributes

- TKatGNL gnl
  katana general
- TKatMFW mfw
  master's firmware version/revision
- TKatIDS ids
  ID string.
- TKatCTB ctb
  cmd table
- TKatCBX cbx
  connector box
- TKatECH ech
  echo
- TKatMOT mot
  motors
- TKatSCT sct
  sensor controllers
- TKatEFF eff
  end effector
- CCplBase * protocol
  protocol interface
- short mMasterVersion
  master version of robot we are communicating with
- short mMasterRevision
  master firmware revision

11.11.1 Detailed Description

Base Katana class.

This class is the main object controlling the whole katana; to use it, it has to be initilized by using it's init function; those function expects a initilized protocol class, which in turn expects an initilized device! after the initialization, it does not mean that the coordinates (encoder values) of the motors have been set correctly; for that a calibration is needen; that calibration can be executed either by using the CKatana class in the 'kmlExt' module (which encapsulates this class) or by writing your own calibrations function..

Definition at line 132 of file kmlBase.h.

11.11.2 Constructor & Destructor Documentation

11.11.2.1 CKatBase::CKatBase () [inline]

Definition at line 172 of file kmlBase.h.

11.11.2.2 virtual CKatBase::~CKatBase () [inline, virtual]

destructor

Definition at line 175 of file kmlBase.h.

11.11.3 Member Function Documentation

11.11.3.1 const TKatGNL* CKatBase::GetGNL () [inline]

Get a pointer to the desired structure.

Definition at line 152 of file kmlBase.h.

11.11.3.2 const TKatMFW* CKatBase::GetMFW () [inline]

Get a pointer to the desired structure.

Definition at line 154 of file kmlBase.h.

11.11.3.3 const TKatIDS* CKatBase::GetIDS () [inline]

Get a pointer to the desired structure.

Definition at line 156 of file kmlBase.h.

11.11.3.4 const TKatCTB* CKatBase::GetCTB () [inline]

Get a pointer to the desired structure.

Definition at line 158 of file kmlBase.h.

11.11.3.5 const TKatCBX* CKatBase::GetCBX () [inline]

Get a pointer to the desired structure.

Definition at line 160 of file kmlBase.h.

11.11.3.6 const TKatECH* CKatBase::GetECH () [inline]

Get a pointer to the desired structure.

Definition at line 162 of file kmlBase.h.

11.11.3.7 const TKatMOT* CKatBase::GetMOT () [inline]

Get a pointer to the desired structure.

Definition at line 165 of file kmlBase.h.

11.11.3.8 const TKatSCT* CKatBase::GetSCT () [inline]

Get a pointer to the desired structure.

Definition at line 167 of file kmlBase.h.

11.11.3.9 TKatEFF* CKatBase::GetEFF () [inline]

Get a pointer to the desired structure.

Definition at line 169 of file kmlBase.h.

11.11.3.10 virtual bool CKatBase::init (const TKatGNL *_gnl*, const TKatMOT *_mot*, const TKatSCT *_sct*, const TKatEFF *_eff*, CCplBase * *_protocol*) [virtual]

Parameters:

- ***_gnl*** general attributes
- ***_mot*** motor attributes
- ***_sct*** sensor controller attributes
- ***_eff*** end effector attributes
- ***_protocol*** desired protocol

11.11.3.11 void CKatBase::recvMFW ()

receive data

11.11.3.12 void CKatBase::recvIDS ()

receive data

11.11.3.13 void CKatBase::recvCTB ()

receive data

11.11.3.14 void CKatBase::recvGMS ()

receive data

11.11.3.15 void CKatBase::waitFor (TMotStsFlg *status*, int *waitTimeout*, bool *gripper*)

wait for motor on all motors

11.11.3.16 void CKatBase::recvECH ()

receive data

11.11.3.17 void CKatBase::recvMPS ()

read all motor positions simultaneously

11.11.3.18 CCplBase* CKatBase::getProtocol () [inline]

get a handle of the protocol, used in CKatana

Definition at line 200 of file kmlBase.h.

11.11.3.19 int CKatBase::checkKatanaType (int *type*)

checks for a K300 or K400

11.11.3.20 void CKatBase::getMasterFirmware (short * *fw*, short * *rev*)

Get the master firmware of the robot we are communicating with.

Get master firmware read at initialization time.

11.11.3.21 void CKatBase::enableCrashLimits ()

crash limits enable

11.11.3.22 void CKatBase::disableCrashLimits ()

crash limits disable

11.11.3.23 void CKatBase::unBlock ()

unblock robot after a crash

11.11.3.24 void CKatBase::setCrashLimit (long *idx*, int *limit*)

set collision limits

//deprecated, use speed & position

11.11.3.25 void CKatBase::setPositionCollisionLimit (long *idx*, int *limit*)

set collision position limits

11.11.3.26 void CKatBase::setSpeedCollisionLimit (long *idx*, int *limit*)

set collision speed limits

11.11.3.27 void CKatBase::startSplineMovement (int *exactflag*, int *moreflag* = 1)

Start a spline movement.

Parameters:

exactflag Set it to 1 if you want the position controller activated after the movement, 0 otherwise, add 2 to set motor6_follow (no gripper)

moreflag 0 = start moving more following, 1 = last or a single polynomial movement, 2 = do not start moving yet more following

11.11.3.28 void CKatBase::setAndStartPolyMovement (std::vector< short > *polynomial*, int *exactflag*, int *moreflag*)

Send polynomials to all motors and start movement.

Parameters:

polynomial time, target pos and coefficients of the motor polynomials

exactflag exactflag

moreflag 0 = start moving more following, 1 = last or a single polynomial movement, 2 = do not start moving yet more following

11.11.3.29 int CKatBase::readDigitalIO ()

get digital I/O data from Katana400:

11.11.3.30 int CKatBase::flushMoveBuffers ()

flush move buffers on all motors:

11.11.4 Member Data Documentation

11.11.4.1 TKatGNL CKatBase::gnl [protected]

katana general

Definition at line 135 of file kmlBase.h.

Referenced by CSctBase::GetGNL(), and CMotBase::GetGNL().

11.11.4.2 TKatMFW CKatBase::mfw [protected]

master's firmware version/revision

Definition at line 136 of file kmlBase.h.

11.11.4.3 TKatIDS CKatBase::ids [protected]

ID string.

Definition at line 137 of file kmlBase.h.

11.11.4.4 TKatCTB CKatBase::ctb [protected]

cmd table

Definition at line 138 of file kmlBase.h.

11.11.4.5 TKatCBX CKatBase::cbx [protected]

connector box

Definition at line 139 of file kmlBase.h.

11.11.4.6 TKatECH CKatBase::ech [protected]

echo

Definition at line 140 of file kmlBase.h.

11.11.4.7 TKatMOT CKatBase::mot [protected]

motors

Definition at line 142 of file kmlBase.h.

11.11.4.8 TKatSCT CKatBase::sct [protected]

sensor controllers

Definition at line 143 of file kmlBase.h.

11.11.4.9 TKatEFF CKatBase::eff [protected]

end effector

Definition at line 144 of file kmlBase.h.

11.11.4.10 CCplBase* CKatBase::protocol [protected]

protocol interface

Definition at line 146 of file kmlBase.h.

11.11.4.11 short CKatBase::mMasterVersion [protected]

master version of robot we are communicating with

Definition at line 147 of file kmlBase.h.

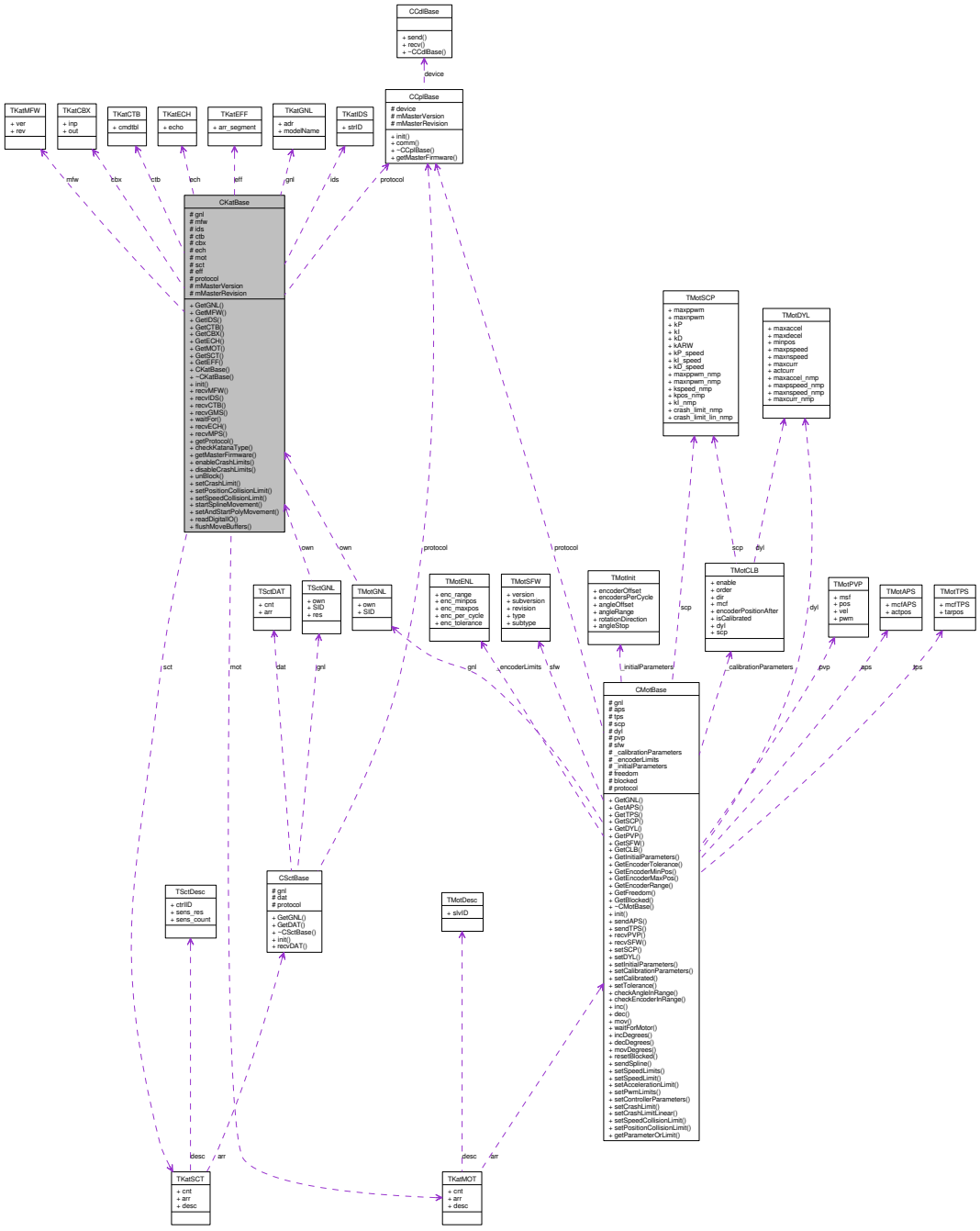
11.11.4.12 short CKatBase::mMasterRevision [protected]

master firmware revision

Definition at line 148 of file kmlBase.h.

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

- include/KNI/kmlBase.h


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
protocol
TKatMFW
+ ver
+ rev
TKatCBX
+ inp
+ out
TKatCTB
+ cmdtbl
TKatECH
+ echo
TKatEFF
+ arr_segment
TKatGNL
+ adr
+ modelName
TKatIDS
+ strID
mfw
cbx
ctb
ech
eff
gnl
ids
CKatBase
gnl
mfw
ids
ctb
cbx
ech
mot
sct
eff
protocol
mMasterVersion
mMasterRevision
+ GetGNL()
+ GetMFW()
+ GetIDS()
+ GetCTB()
+ GetCBX()
+ GetECH()
+ GetMOT()
+ GetSCT()
+ GetEFF()
+ CKatBase()
+ ~CKatBase()
+ init()
+ recvMFW()
+ recvIDS()
+ recvCTB()
+ recvGMS()
+ waitFor()
+ recvECH()
+ recvMPS()
+ getProtocol()
+ checkKatanaType()
+ getMasterFirmware()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ startSplineMovement()
+ setAndStartPolyMovement()
+ readDigitalIO()
+ flushMoveBuffers()
TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kARW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ kI_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxnspeed
+ maxcurr
+ actcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxnspeed_nmp
+ maxcurr_nmp
own
own
protocol
protocol
scp
dyl
TSctDAT
+ cnt
+ arr
TSctGNL
+ own
+ SID
+ res
TMotGNL
+ own
+ SID
TMotENL
+ enc_range
+ enc_minpos
+ enc_maxpos
+ enc_per_cycle
+ enc_tolerance
TMotSFW
+ version
+ subversion
+ revision
+ type
+ subtype
TMotInit
+ encoderOffset
+ encodersPerCycle
+ angleOffset
+ angleRange
+ rotationDirection
+ angleStop
TMotCLB
+ enable
+ order
+ dir
+ mcf
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp
TMotPVP
+ msf
+ pos
+ vel
+ pwm
TMotAPS
+ mctAPS
+ actpos
TMotTPS
+ mctTPS
+ tarpos
scp
dyl
sct
mot
dat
gnl
gnl
encoderLimits
sfw
_initialParameters
_calibrationParameters
pvp
aps
tps
CMotBase
gnl
aps
tps
scp
dyl
pvp
sfw
_calibrationParameters
_encoderLimits
_initialParameters
freedom
blocked
protocol
+ GetGNL()
+ GetAPS()
+ GetTPS()
+ GetSCP()
+ GetDYL()
+ GetPVP()
+ GetSFW()
+ GetCLB()
+ GetInitialParameters()
+ GetEncoderTolerance()
+ GetEncoderMinPos()
+ GetEncoderMaxPos()
+ GetEncoderRange()
+ GetFreedom()
+ GetBlocked()
+ ~CMotBase()
+ init()
+ sendAPS()
+ sendTPS()
+ recvPVP()
+ recvSFW()
+ setSCP()
+ setDYL()
+ setInitialParameters()
+ setCalibrationParameters()
+ setCalibrated()
+ setTolerance()
+ checkAngleInRange()
+ checkEncoderInRange()
+ inc()
+ dec()
+ mov()
+ waitForMotor()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ resetBlocked()
+ sendSpline()
+ setSpeedLimits()
+ setSpeedLimit()
+ setAcceleratorLimit()
+ setPwmLimits()
+ setControllerParameters()
+ setCrashLimit()
+ setCrashLimitLinear()
+ setSpeedCollisionLimit()
+ setPositionCollisionLimit()
+ getParameterOrLimit()
TSctDesc
+ ctrlID
+ sens_res
+ sens_count
CSctBase
gnl
dat
protocol
+ GetGNL()
+ GetDAT()
+ ~CSctBase()
+ init()
+ recvDAT()
TMotDesc
+ slvID
desc
arr
desc
arr
TKatSCT
+ cnt
+ arr
+ desc
TKatMOT
+ cnt
+ arr
+ desc

11.12 CLMBase Class Reference

Linear movement Class.

`#include <lmBase.h>`

Inheritance diagram for CLMBase: Collaboration diagram for CLMBase:

Public Member Functions

- CLMBase ()
- void movLM (double X, double Y, double Z, double Al, double Be, double Ga, bool exactflag, double vmax, bool wait=true, int tolerance=100, long timeout=TM_ENDLESS)
- void movLM2P (double X1, double Y1, double Z1, double Al1, double Be1, double Ga1, double X2, double Y2, double Z2, double Al2, double Be2, double Ga2, bool exactflag, double vmax, bool wait=true, int tolerance=100, long timeout=TM_ENDLESS)

  Move linear from point to point using multiple splines.

- void movP2P (double X1, double Y1, double Z1, double Ph1, double Th1, double Ps1, double X2, double Y2, double Z2, double Ph2, double Th2, double Ps2, bool exactflag, double vmax, bool wait=true, long timeout=TM_ENDLESS)

  Move point to point using splines.

- void setMaximumLinearVelocity (double maximumVelocity)
- double getMaximumLinearVelocity () const
- void setActivatePositionController (bool activate)

  Re-Activate the position controller after the linear movement.

- bool getActivatePositionController ()

  Check if the position controller will be activated after the linear movement.

- void moveRobotLinearTo (double x, double y, double z, double phi, double theta, double psi, bool waitUntilReached=true, int waitTimeout=TM_ENDLESS)
- void moveRobotLinearTo (std::vector< double > coordinates, bool waitUntilReached=true, int waitTimeout=TM_ENDLESS)

  This method does the same as the one above and is mainly provided for convenience.

- void moveRobotTo (double x, double y, double z, double phi, double theta, double psi, bool waitUntilReached=true, int waitTimeout=TM_ENDLESS)

  Moves to robot to given cartesian coordinates and euler-angles.

- void moveRobotTo (std::vector< double > coordinates, bool waitUntilReached=true, int waitTimeout=TM_ENDLESS)

  This method does the same as the one above and is mainly provided for convenience.

Private Member Functions

- double totalTime (double distance, double acc, double dec, double vmax)

  Calculates time needed for movement over a distance.

- double relPosition (double reltime, double distance, double acc, double dec, double vmax)

  Calculates the relative position reached after the relative time given.

- void splineCoefficients (int steps, double *timearray, double *encoderarray, double *arr_p1, double *arr_p2, double *arr_p3, double *arr_p4)

  Calculates the spline coefficient and stores them in arr_p1 - arr_p4.

- bool checkJointSpeed (std::vector< int > lastsolution, std::vector< int > solution, double time)

  Checks if the joint speeds are below speed limit.

- int getSpeed (int distance, int acceleration, int time)

  Calculates speed from distance, acceleration and time for the movement.

Private Attributes

- double _maximumVelocity
- bool _activatePositionController

11.12.1 Detailed Description

Linear movement Class.

This class allows to do linear movements with the Katana robot.

Definition at line 73 of file lmBase.h.

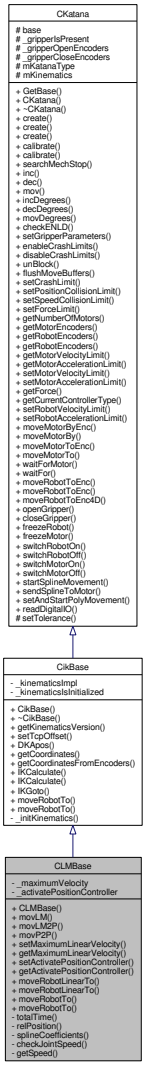
11.12.2 Member Data Documentation

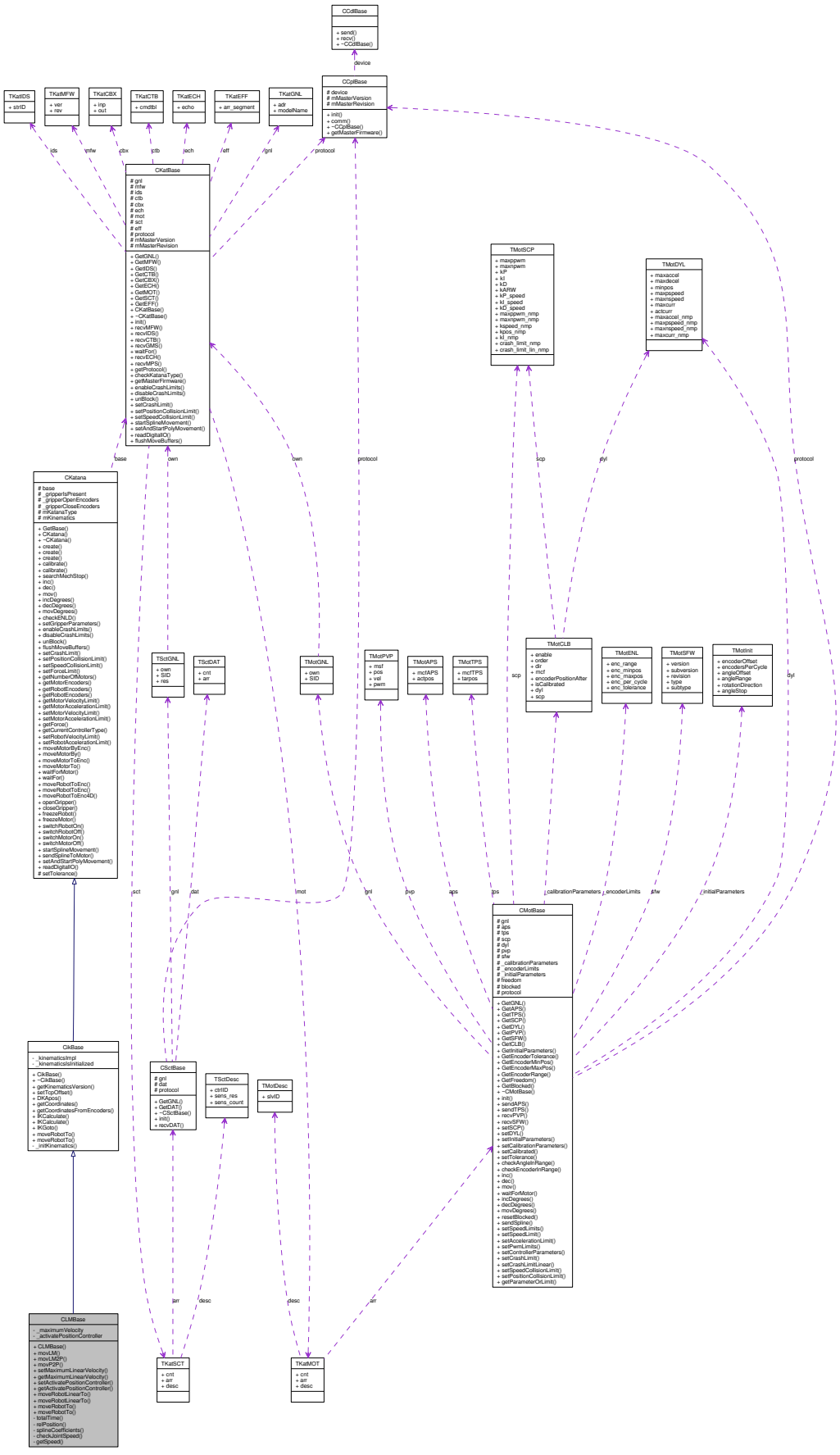
11.12.2.1 double CLMBase::_maximumVelocity [private]

Definition at line 76 of file lmBase.h.

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

- include/KNI_LM/lmBase.h


CKatana
base
_gripperIsPresent
_gripperOpenEncoders
_gripperCloseEncoders
mKatanaType
mKinematics
+ GetBase()
+ CKatana()
+ ~CKatana()
+ create()
+ create()
+ create()
+ calibrate()
+ calibrate()
+ searchMechStop()
+ inc()
+ dec()
+ mov()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ checkENLD()
+ setGripperParameters()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ flushMoveBuffers()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ setForceLimit()
+ getNumberOfMotors()
+ getMotorEncoders()
+ getRobotEncoders()
+ getRobotEncoders()
+ getMotorVelocityLimit()
+ getMotorAccelerationLimit()
+ setMotorVelocityLimit()
+ setMotorAccelerationLimit()
+ getForce()
+ getCurrentControllerType()
+ setRobotVelocityLimit()
+ setRobotAccelerationLimit()
+ moveMotorByEnc()
+ moveMotorBy()
+ moveMotorToEnc()
+ moveMotorTo()
+ waitForMotor()
+ waitFor()
+ moveRobotToEnc()
+ moveRobotToEnc()
+ moveRobotToEnc4D()
+ openGripper()
+ closeGripper()
+ freezeRobot()
+ freezeMotor()
+ switchRobotOn()
+ switchRobotOff()
+ switchMotorOn()
+ switchMotorOff()
+ startSplineMovement()
+ sendSplineToMotor()
+ setAndStartPolyMovement()
+ readDigitalIO()
# setTolerance()
CikBase
- _kinematicsImpl
- _kinematicsIsInitialized
+ CikBase()
+ ~CikBase()
+ getKinematicsVersion()
+ setTcpOffset()
+ DKApos()
+ getCoordinates()
+ getCoordinatesFromEncoders()
+ IKCalculate()
+ IKCalculate()
+ IKGoto()
+ moveRobotTo()
+ moveRobotTo()
- _initKinematics()
CLMBase
- _maximumVelocity
- _activatePositionController
+ CLMBase()
+ movLM()
+ movLM2P()
+ movP2P()
+ setMaximumLinearVelocity()
+ getMaximumLinearVelocity()
+ setActivatePositionController()
+ getActivatePositionController()
+ moveRobotLinearTo()
+ moveRobotLinearTo()
+ moveRobotTo()
+ moveRobotTo()
- totalTime()
- relPosition()
- splineCoefficients()
- checkJointSpeed()
- getSpeed()


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
protocol
TKatIDS
+ strID
TKatIMFW
+ ver
+ rev
TKatICBX
+ inp
+ out
TKatICTB
+ cmdtbl
TKatIECH
+ echo
TKatIEFF
+ arr_segment
TKatIGNL
+ adr
+ modeName
ids
imfw
icbx
ictb
iech
ieff
ignl
CKatBase
gnl
imfw
ids
ictb
icbx
iech
mot
sct
eff
protocol
mMasterVersion
mMasterRevision
+ GetGNL()
+ GetMFW()
+ GetIDS()
+ GetCTB()
+ GetCBX()
+ GetECH()
+ GetMOT()
+ GetSCT()
+ GetEFF()
+ CKatBase()
+ ~CKatBase()
+ init()
+ recvMFW()
+ recvIDS()
+ recvCTB()
+ recvGMS()
+ waitFor()
+ recvECH()
+ recvMPS()
+ getProtocol()
+ checkKatanaType()
+ getMasterFirmware()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ startSplineMovement()
+ setAndStartPolyMovement()
+ readDigitalIO()
+ flushMoveBuffers()
TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kARW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ kI_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxnspeed
+ maxcurr
+ actcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxnspeed_nmp
+ maxcurr_nmp
base
own
own
protocol
scp
dyl
protocol
CKatana
base
_gripperIsPresent
_gripperOpenEncoders
_gripperCloseEncoders
mKatanaType
mKinematics
+ GetBase()
+ CKatana()
+ ~CKatana()
+ create()
+ create()
+ create()
+ calibrate()
+ calibrate()
+ searchMechStop()
+ inc()
+ dec()
+ mov()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ checkENLD()
+ setGripperParameters()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ flushMoveBuffers()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ setForceLimit()
+ getNumberOfMotors()
+ getMotorEncoders()
+ getRobotEncoders()
+ getRobotEncoders()
+ getMotorVelocityLimit()
+ getMotorAccelerationLimit()
+ setMotorVelocityLimit()
+ setMotorAccelerationLimit()
+ getForce()
+ getCurrentControllerType()
+ setRobotVelocityLimit()
+ setRobotAccelerationLimit()
+ moveMotorByEnc()
+ moveMotorBy()
+ moveMotorToEnc()
+ moveMotorTo()
+ waitForMotor()
+ waitFor()
+ moveRobotToEnc()
+ moveRobotToEnc()
+ moveRobotToEnc4DI()
+ openGripper()
+ closeGripper()
+ freezeRobot()
+ freezeMotor()
+ switchRobotOn()
+ switchRobotOff()
+ switchMotorOn()
+ switchMotorOff()
+ startSplineMovement()
+ sendSplineToMotor()
+ setAndStartPolyMovement()
+ readDigitalIO()
# setTolerance()
TSctGNL
+ own
+ SID
+ res
TSctDAT
+ cnt
+ arr
TMotGNL
+ own
+ SID
TMotPVP
+ msf
+ pos
+ vel
+ pwm
TMotAPS
+ mctAPS
+ actpos
TMotTPS
+ mctTPS
+ tarpos
TMotCLB
+ enable
+ order
+ dir
+ mcr
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp
scp
TMotENL
+ enc_range
+ enc_minpos
+ enc_maxpos
+ enc_per_cycle
+ enc_tolerance
TMotSFW
+ version
+ subversion
+ revision
+ type
+ subtype
TMotInit
+ encoderOffset
+ encodersPerCycle
+ angleOffset
+ angleRange
+ rotationDirection
+ angleStop
dyl
sct
gnl
dat
mot
gnl
pvp
aps
tps
calibrationParameters
encoderLimits
sfw
initialParameters
CMotBase
gnl
aps
tps
scp
dyl
pvp
sfw
_calibrationParameters
_encoderLimits
_initialParameters
freedom
blocked
protocol
+ GetGNL()
+ GetAPS()
+ GetTPS()
+ GetSCP()
+ GetDYL()
+ GetPVP()
+ GetSFW()
+ GetCLB()
+ GetInitialParameters()
+ GetEncoderTolerance()
+ GetEncoderMinPos()
+ GetEncoderMaxPos()
+ GetEncoderRange()
+ GetFreedom()
+ GetBlocked()
+ ~CMotBase()
+ init()
+ sendAPS()
+ sendTPS()
+ recvPVP()
+ recvSFW()
+ setSCP()
+ setDYL()
+ setInitialParameters()
+ setCalibrationParameters()
+ setCalibrated()
+ setTolerance()
+ checkAngleInRange()
+ checkEncoderInRange()
+ inc()
+ dec()
+ mov()
+ waitForMotor()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ resetBlocked()
+ sendSpline()
+ setSpeedLimits()
+ setSpeedLimit()
+ setAccelerationLimit()
+ setPwmLimits()
+ setControllerParameters()
+ setCrashLimit()
+ setCrashLimitLinear()
+ setSpeedCollisionLimit()
+ setPositionCollisionLimit()
+ getParameterOfLimit()
CIkBase
- _kinematicsImpl
- _kinematicsIsInitialized
+ CIkBase()
+ ~CIkBase()
+ getKinematicsVersion()
+ setTcpOffset()
+ DKApos()
+ getCoordinates()
+ getCoordinatesFromEncoders()
+ IKCalculate()
+ IKCalculate()
+ IKGoto()
+ moveRobotTo()
+ moveRobotTo()
- _initKinematics()
CSctBase
gnl
dat
protocol
+ GetGNL()
+ GetDAT()
+ ~CSctBase()
+ init()
+ recvDAT()
TSctDesc
+ ctrlID
+ sens_res
+ sens_count
TMotDesc
+ slvID
CLMBase
- _maximumVelocity
- _activatePositionController
+ CLMBase()
+ movLIN()
+ movLMZP()
+ movP2P()
+ setMaximumLinearVelocity()
+ getMaximumLinearVelocity()
+ setActivatePositionController()
+ getActivatePositionController()
+ moveRobotLinearTo()
+ moveRobotLinearTo()
+ moveRobotTo()
+ moveRobotTo()
- totalTime()
- relPosition()
- splineCoefficients()
- checkJointSpeed()
- getSpeed()
arr
desc
desc
arr
TKatSCT
+ cnt
+ arr
+ desc
TKatMOT
+ cnt
+ arr
+ desc

11.13 CMotBase Class Reference

Motor class.

`#include <kmlMotBase.h>`

Collaboration diagram for CMotBase:

Public Member Functions

- const TMotGNL * GetGNL ()
- const TMotAPS * GetAPS ()
- const TMotTPS * GetTPS ()
- const TMotSCP * GetSCP ()
- const TMotDYL * GetDYL ()
- const TMotPVP * GetPVP ()
- const TMotSFW * GetSFW ()
- const TMotCLB * GetCLB ()
- const TMotInit * GetInitialParameters ()
- const int GetEncoderTolerance ()
- const int GetEncoderMinPos ()

  Returns the min Position of the Encoder.

- const int GetEncoderMaxPos ()

  Returns the max Position of the Encoder.

- const int GetEncoderRange ()

  Returns Encoder Range of the Encoder.

- const bool GetFreedom ()

  Get the value of the freedom property.

- const bool GetBlocked ()

  Get the value of the blocked property.

- virtual ~CMotBase ()
- bool init (CKatBase *_own, const TMotDesc _motDesc, CCplBase *protocol)
- void sendAPS (const TMotAPS *_aps)

  send data

- void sendTPS (const TMotTPS *_tps)

  send data

- void recvPVP ()

  receive data

- void recvSFW ()

  receive data

- void setSCP (TMotSCP _scp)
- void setDYL (TMotDYL _dyl)
- void setInitialParameters (double angleOffset, double angleRange, int encodersPerCycle, int encoderOffset, int rotationDirection)

- void setCalibrationParameters (bool doCalibration, short order, TSearchDir direction, TMotCmdFlg motorFlagAfter, int encoderPositionAfter)
- void setCalibrated (bool calibrated)
- void setTolerance (int tolerance)
- bool checkAngleInRange (double angle)

  check limits in encoder values

- bool checkEncoderInRange (int encoder)
- void inc (int dif, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Increments the motor specified by an index postion in encoder units.

- void dec (int dif, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Decrements the motor specified by an index postion in encoder units.

- void mov (int tar, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Moves the motor specified by an index to a given target position in encoder units.

- void waitForMotor (int tar, int encTolerance=100, short mode=0, int waitTimeout=TM_ENDLESS)

  Waits until the Motor has reached the given targen position.

- void incDegrees (double dif, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Increments the motor specified by an index postion in degrees.

- void decDegrees (double dif, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Decrements the motor specified by an index postion in degrees.

- void movDegrees (double tar, bool wait=false, int tolerance=100, long timeout=TM_ENDLESS)

  Moves the motor specified by an index to a given target position in degrees.

- void resetBlocked ()

  unblock the motor.

- void sendSpline (short targetPosition, short duration, short p1, short p2, short p3, short p4)

  Send one spline to the motor.

- void setSpeedLimits (short positiveVelocity, short negativeVelocity)
- void setSpeedLimit (short velocity)
- void setAccelerationLimit (short acceleration)

  Set the acceleration limits.

- void setPwmLimits (byte maxppwm, byte maxnpwm)

  Set the PWM limits (for the drive controller).

- void setControllerParameters (byte kSpeed, byte kPos, byte kI)

  Set the controller parameters.

- void setCrashLimit (int limit)

  Set the crash limit.

- void setCrashLimitLinear (int limit_lin)

  Set the crash limit linear.

- void setSpeedCollisionLimit (int limit)

  Set the collision limit.

- void setPositionCollisionLimit (int limit)

  Set the collision limit.

- void getParameterOrLimit (int subcommand, byte *R1, byte *R2, byte *R3)

  Get parameters or limits.

Protected Attributes

- TMotGNL gnl

  motor generals

- TMotAPS aps

  actual position

- TMotTPS tps

  target position

- TMotSCP scp

  static controller parameters

- TMotDYL dyl

  dynamic limits

- TMotPVP pvp

  reading motor parameters

- TMotSFW sfw

  slave firmware

- TMotCLB _calibrationParameters

  calibration structure

- TMotENL _encoderLimits

  motor limits in encoder values

- TMotInit _initialParameters
- bool freedom

  if it is set, it will move on a parallel movement

- bool blocked

  true if the motor was blocked due to a crash of the robot

- CCplBase * protocol

  protocol interface

Friends

- class CKatBase

11.13.1 Detailed Description

Motor class.

This class allows to control one motor; to control a motor it has to be initialized by using the init function. And the usage the internal allocated resources should be deallocated by using the 'free' method.

Definition at line 220 of file kmlMotBase.h.

11.13.2 Constructor & Destructor Documentation

11.13.2.1 virtual CMotBase::~CMotBase () [inline, virtual]

Definition at line 265 of file kmlMotBase.h.

11.13.3 Member Function Documentation

11.13.3.1 const TMotGNL* CMotBase::GetGNL () [inline]

Definition at line 241 of file kmlMotBase.h.

References CKatBase::gnl.

11.13.3.2 const TMotAPS* CMotBase::GetAPS () [inline]

Definition at line 242 of file kmlMotBase.h.

11.13.3.3 const TMotTPS* CMotBase::GetTPS () [inline]

Definition at line 243 of file kmlMotBase.h.

11.13.3.4 const TMotSCP* CMotBase::GetSCP () [inline]

Definition at line 244 of file kmlMotBase.h.

11.13.3.5 const TMotDYL* CMotBase::GetDYL () [inline]

Definition at line 245 of file kmlMotBase.h.

11.13.3.6 const TMotPVP* CMotBase::GetPVP () [inline]

Definition at line 246 of file kmlMotBase.h.

11.13.3.7 const TMotSFW* CMotBase::GetSFW () [inline]

Definition at line 247 of file kmlMotBase.h.

11.13.3.8 const TMotCLB* CMotBase::GetCLB () [inline]

Definition at line 248 of file kmlMotBase.h.

11.13.3.9 const TMotInit* CMotBase::GetInitialParameters () [inline]

Definition at line 250 of file kmlMotBase.h.

11.13.3.10 const int CMotBase::GetEncoderTolerance () [inline]

Definition at line 251 of file kmlMotBase.h.

11.13.3.11 const int CMotBase::GetEncoderMinPos () [inline]

Returns the min Position of the Encoder.

Definition at line 252 of file kmlMotBase.h.

11.13.3.12 const int CMotBase::GetEncoderMaxPos () [inline]

Returns the max Position of the Encoder.

Definition at line 253 of file kmlMotBase.h.

11.13.3.13 const int CMotBase::GetEncoderRange () [inline]

Returns Encoder Range of the Encoder.

Definition at line 254 of file kmlMotBase.h.

11.13.3.14 const bool CMotBase::GetFreedom () [inline]

Get the value of the freedom property.

Definition at line 257 of file kmlMotBase.h.

11.13.3.15 const bool CMotBase::GetBlocked () [inline]

Get the value of the blocked property.

Definition at line 259 of file kmlMotBase.h.

11.13.3.16 bool CMotBase::init (CKatBase * *_own*, const TMotDesc *_motDesc*, CCplBase * *protocol*)

11.13.3.17 void CMotBase::sendAPS (const TMotAPS * *_aps*)

send data

11.13.3.18 void CMotBase::sendTPS (const TMotTPS * *_tps*)

send data

11.13.3.19 void CMotBase::recvPVP ()

receive data

11.13.3.20 void CMotBase::recvSFW ()

receive data

11.13.3.21 void CMotBase::setSCP (TMotSCP *_scp*) [inline]

Definition at line 279 of file kmlMotBase.h.

11.13.3.22 void CMotBase::setDYL (TMotDYL *_dyl*) [inline]

Definition at line 280 of file kmlMotBase.h.

11.13.3.23 void CMotBase::setInitialParameters (double *angleOffset*, double *angleRange*, int *encodersPerCycle*, int *encoderOffset*, int *rotationDirection*)

11.13.3.24 void CMotBase::setCalibrationParameters (bool *doCalibration*, short *order*, TSearchDir *direction*, TMotCmdFlg *motorFlagAfter*, int *encoderPositionAfter*)

11.13.3.25 void CMotBase::setCalibrated (bool *calibrated*)

11.13.3.26 void CMotBase::setTolerance (int *tolerance*)

11.13.3.27 bool CMotBase::checkAngleInRange (double *angle*)

check limits in encoder values

11.13.3.28 bool CMotBase::checkEncoderInRange (int *encoder*)

11.13.3.29 void CMotBase::inc (int *dif*, bool *wait* = false, int *tolerance* = 100, long *timeout* = TM_ENDLESS)

Increments the motor specified by an index postion in encoder units.

11.13.3.30 void CMotBase::dec (int *dif*, bool *wait* = false, int *tolerance* = 100, long *timeout* = TM_ENDLESS)

Decrements the motor specified by an index postion in encoder units.

11.13.3.31 void CMotBase::mov (int *tar*, bool *wait* = false, int *tolerance* = 100, long *timeout* = TM_ENDLESS)

Moves the motor specified by an index to a given target position in encoder units.

11.13.3.32 void CMotBase::waitForMotor (int *tar*, int *encTolerance* = 100, short *mode* = 0, int *waitTimeout* = TM_ENDLESS)

Waits until the Motor has reached the given targen position.

11.13.3.33 void CMotBase::incDegrees (double *dif*, bool *wait* = false, int *tolerance* = 100, long *timeout* = TM_ENDLESS)

Increments the motor specified by an index postion in degrees.

11.13.3.34 void CMotBase::decDegrees (double *dif*, bool *wait* = false, int *tolerance* = 100, long *timeout* = TM_ENDLESS)

Decrements the motor specified by an index postion in degrees.

11.13.3.35 void CMotBase::movDegrees (double *tar*, bool *wait* = false, int *tolerance* = 100, long *timeout* = TM_ENDLESS)

Moves the motor specified by an index to a given target position in degrees.

11.13.3.36 void CMotBase::resetBlocked ()

unblock the motor.

11.13.3.37 void CMotBase::sendSpline (short *targetPosition*, short *duration*, short *p1*, short *p2*, short *p3*, short *p4*)

Send one spline to the motor.

Parameters:

duration Duration has to be given in 10ms units

11.13.3.38 void CMotBase::setSpeedLimits (short *positiveVelocity*, short *negativeVelocity*)

Set speed limits

11.13.3.39 void CMotBase::setSpeedLimit (short *velocity*) [inline]

Definition at line 331 of file kmlMotBase.h.

11.13.3.40 void CMotBase::setAccelerationLimit (short *acceleration*)

Set the acceleration limits.

11.13.3.41 void CMotBase::setPwmLimits (byte *maxppwm*, byte *maxnpwm*)

Set the PWM limits (for the drive controller).

11.13.3.42 void CMotBase::setControllerParameters (byte *kSpeed*, byte *kPos*, byte *kI*)

Set the controller parameters.

11.13.3.43 void CMotBase::setCrashLimit (int *limit*)

Set the crash limit.

11.13.3.44 void CMotBase::setCrashLimitLinear (int *limit_lin*)

Set the crash limit linear.

11.13.3.45 void CMotBase::setSpeedCollisionLimit (int *limit*)

Set the collision limit.

11.13.3.46 void CMotBase::setPositionCollisionLimit (int *limit*)

Set the collision limit.

11.13.3.47 void CMotBase::getParameterOrLimit (int *subcommand*, byte * *R1*, byte * *R2*, byte * *R3*)

Get parameters or limits.

Parameters:

subcommand 255-249;245, see katana user manual chapter 8 firmware commands for details

R1 pointer to store first byte of answer

R2 pointer to store second byte of answer

R3 pointer to store third byte of answer

11.13.4 Friends And Related Function Documentation

11.13.4.1 friend class CKatBase [friend]

Definition at line 222 of file kmlMotBase.h.

11.13.5 Member Data Documentation

11.13.5.1 TMotGNL CMotBase::gnl [protected]

motor generals

Definition at line 226 of file kmlMotBase.h.

11.13.5.2 TMotAPS CMotBase::aps [protected]

actual position

Definition at line 227 of file kmlMotBase.h.

11.13.5.3 TMotTPS CMotBase::tps [protected]

target position

Definition at line 228 of file kmlMotBase.h.

11.13.5.4 TMotSCP CMotBase::scp [protected]

static controller parameters

Definition at line 229 of file kmlMotBase.h.

11.13.5.5 TMotDYL CMotBase::dyl [protected]

dynamic limits

Definition at line 230 of file kmlMotBase.h.

11.13.5.6 TMotPVP CMotBase::pvp [protected]

reading motor parameters

Definition at line 231 of file kmlMotBase.h.

11.13.5.7 TMotSFW CMotBase::sfw [protected]

slave firmware

Definition at line 232 of file kmlMotBase.h.

11.13.5.8 TMotCLB CMotBase::_calibrationParameters [protected]

calibration structure

Definition at line 233 of file kmlMotBase.h.

11.13.5.9 TMotENL CMotBase::_encoderLimits [protected]

motor limits in encoder values

Definition at line 234 of file kmlMotBase.h.

11.13.5.10 TMotInit CMotBase::_initialParameters [protected]

Definition at line 235 of file kmlMotBase.h.

11.13.5.11 bool CMotBase::freedom [protected]

if it is set, it will move on a parallel movement

Definition at line 236 of file kmlMotBase.h.

11.13.5.12 bool CMotBase::blocked [protected]

true if the motor was blocked due to a crash of the robot

Definition at line 237 of file kmlMotBase.h.

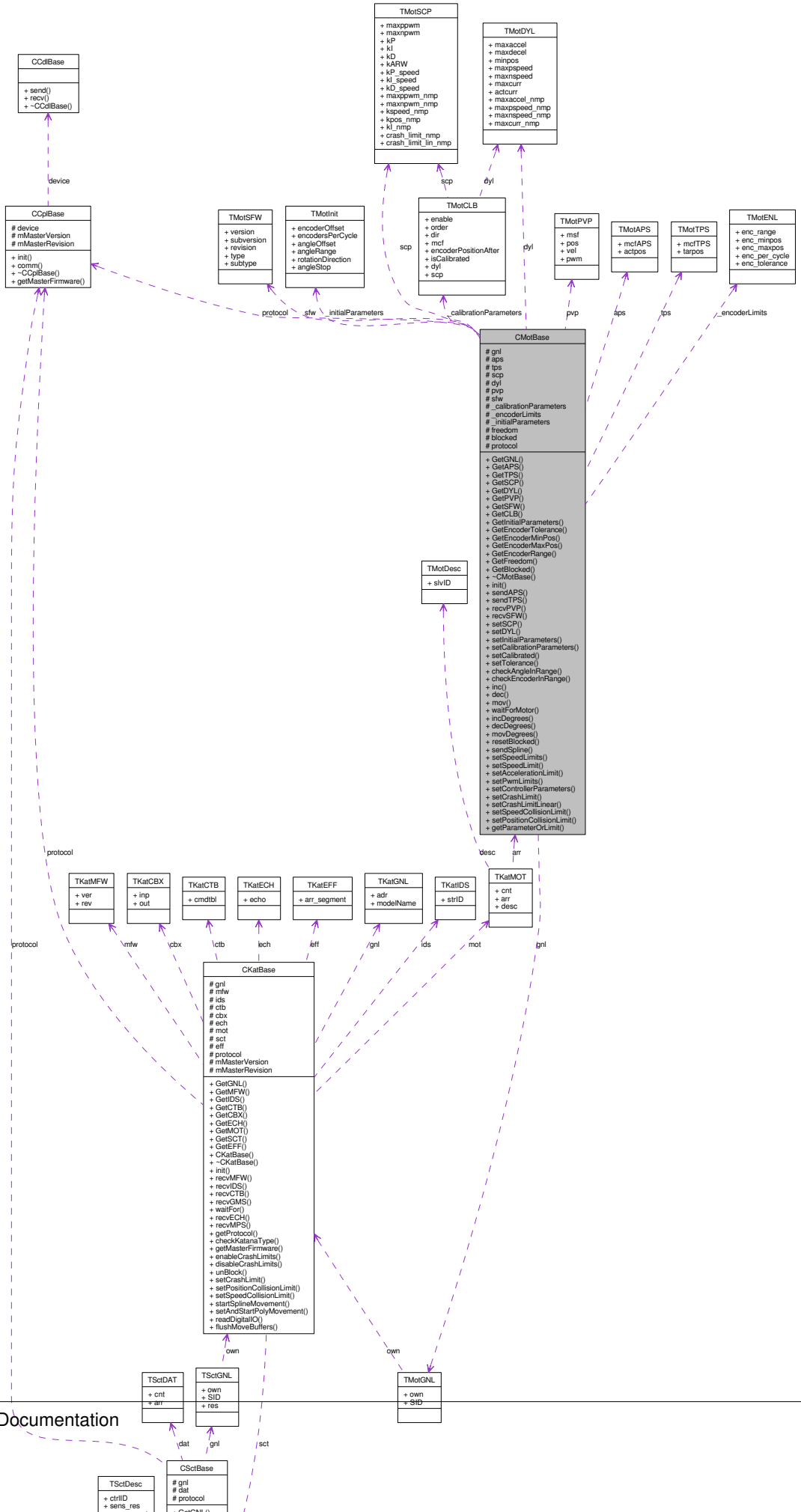
11.13.5.13 CCplBase* CMotBase::protocol [protected]

protocol interface

Definition at line 262 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h


TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kARW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ kI_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxnspeed
+ maxcurr
+ actcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxnspeed_nmp
+ maxcurr_nmp
CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
TMotSFW
+ version
+ subversion
+ revision
+ type
+ subtype
TMotInit
+ encoderOffset
+ encodersPerCycle
+ angleOffset
+ angleRange
+ rotationDirection
+ angleStop
TMotCLB
+ enable
+ order
+ dir
+ mcf
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp
TMotPVP
+ msf
+ pos
+ vel
+ pwm
TMotAPS
+ mcfAPS
+ actpos
TMotTPS
+ mcfTPS
+ tarpos
TMotENL
+ enc_range
+ enc_minpos
+ enc_maxpos
+ enc_per_cycle
+ enc_tolerance
scp
dyl
protocol
sfw
_initialParameters
_calibrationParameters
pvp
aps
tps
_encoderLimits
CMotBase
gnl
aps
tps
scp
dyl
pvp
sfw
_calibrationParameters
_encoderLimits
_initialParameters
freedom
blocked
protocol
+ GetGNL()
+ GetAPS()
+ GetTPS()
+ GetSCP()
+ GetDYL()
+ GetPVP()
+ GetSFW()
+ GetCLB()
+ GetInitialParameters()
+ GetEncoderTolerance()
+ GetEncoderMinPos()
+ GetEncoderMaxPos()
+ GetEncoderRange()
+ GetFreedom()
+ GetBlocked()
+ ~CMotBase()
+ init()
+ sendAPS()
+ sendTPS()
+ recvPVP()
+ recvSFW()
+ setSCP()
+ setDYL()
+ setInitialParameters()
+ setCalibrationParameters()
+ setCalibrated()
+ setTolerance()
+ checkAngleInRange()
+ checkEncoderInRange()
+ inc()
+ dec()
+ mov()
+ waitForMotor()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ resetBlocked()
+ sendSpline()
+ setSpeedLimits()
+ setSpeedLimit()
+ setAccelerationLimit()
+ setPwmLimits()
+ setControllerParameters()
+ setCrashLimit()
+ setCrashLimitLinear()
+ setSpeedCollisionLimit()
+ setPositionCollisionLimit()
+ getParameterOrLimit()
TMotDesc
+ slvID
desc
arr
protocol
TKatMFW
+ ver
+ rev
TKatCBX
+ inp
+ out
TKatCTB
+ cmdtbl
TKatECH
+ echo
TKatEFF
+ arr_segment
TKatGNL
+ adr
+ modelName
TKatIDS
+ strID
TKatMOT
+ cnt
+ arr
+ desc
mfw
cbx
ctb
ech
eff
gnl
ids
mot
CKatBase
gnl
mfw
ids
ctb
cbx
ech
mot
sct
eff
protocol
mMasterVersion
mMasterRevision
+ GetGNL()
+ GetMFW()
+ GetIDS()
+ GetCTB()
+ GetCBX()
+ GetECH()
+ GetMOT()
+ GetSCT()
+ GetEFF()
+ CKatBase()
+ ~CKatBase()
+ init()
+ recvMFW()
+ recvIDS()
+ recvCTB()
+ recvGMS()
+ waitFor()
+ recvECH()
+ recvMPS()
+ getProtocol()
+ checkKatanaType()
+ getMasterFirmware()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ startSplineMovement()
+ setAndStartPolyMovement()
+ readDigitalIO()
+ flushMoveBuffers()
own
own
TSctDAT
+ cnt
+ arr
TSctGNL
+ own
+ SID
+ res
TMotGNL
+ own
+ SID
dat
gnl
sct
TSctDesc
+ ctrlID
+ sens_res
CSctBase
gnl
dat
protocol

11.14 ConfigFileEntryNotFoundException Class Reference

The requested entry could not be found.

`#include <kmlFactories.h>`

Inheritance diagram for ConfigFileEntryNotFoundException: Collaboration diagram for ConfigFileEntryNotFoundException:

Public Member Functions

- ConfigFileEntryNotFoundException (const std::string &attribute) throw ()

11.14.1 Detailed Description

The requested entry could not be found.

Note:

error_number=-44

Definition at line 49 of file kmlFactories.h.

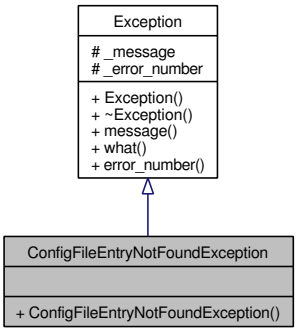
11.14.2 Constructor & Destructor Documentation

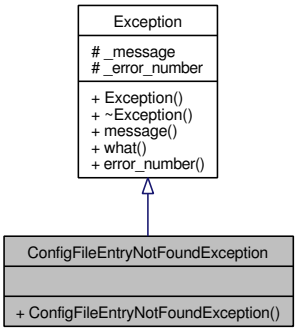
11.14.2.1 ConfigFileEntryNotFoundException::ConfigFileEntryNotFoundException (const std::string & *attribute*) throw () [inline]

Definition at line 51 of file kmlFactories.h.

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

- include/KNI/kmlFactories.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileEntryNotFoundException
+ ConfigFileEntryNotFoundException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileEntryNotFoundException
+ ConfigFileEntryNotFoundException()

11.15 ConfigFileOpenException Class Reference

Accessing the given configuration file failed (may be: access denied or wrong path).

`#include <kmlExt.h>`

Inheritance diagram for ConfigFileOpenException: Collaboration diagram for ConfigFileOpenException:

Public Member Functions

- ConfigFileOpenException (const std::string &port) throw ()

11.15.1 Detailed Description

Accessing the given configuration file failed (may be: access denied or wrong path).

Note:

error_number=-40

Definition at line 45 of file kmlExt.h.

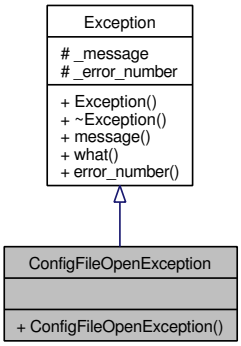
11.15.2 Constructor & Destructor Documentation

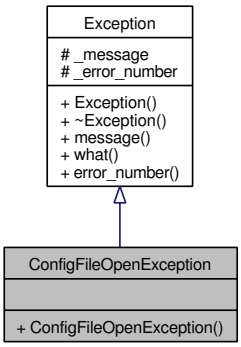
11.15.2.1 ConfigFileOpenException::ConfigFileOpenException (const std::string & *port*) throw () [inline]

Definition at line 47 of file kmlExt.h.

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

- include/KNI/kmlExt.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileOpenException
+ ConfigFileOpenException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileOpenException
+ ConfigFileOpenException()

11.16 ConfigFileSectionNotFoundException Class Reference

The requested section could not be found.

`#include <kmlFactories.h>`

Inheritance diagram for ConfigFileSectionNotFoundException: Collaboration diagram for ConfigFileSectionNotFoundException:

Public Member Functions

- ConfigFileSectionNotFoundException (const std::string &attribute) throw ()

11.16.1 Detailed Description

The requested section could not be found.

Note:

error_number=-42

Definition at line 31 of file kmlFactories.h.

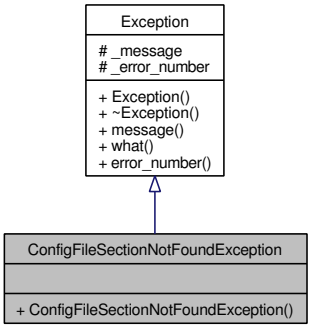
11.16.2 Constructor & Destructor Documentation

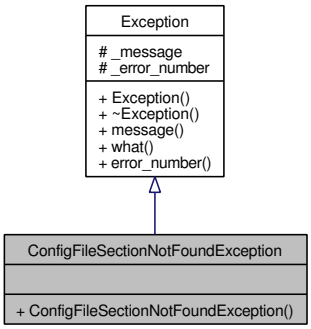
11.16.2.1 ConfigFileSectionNotFoundException::ConfigFileSectionNotFoundException (const std::string & *attribute*) throw () [inline]

Definition at line 33 of file kmlFactories.h.

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

- include/KNI/kmlFactories.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileSectionNotFoundException
+ ConfigFileSectionNotFoundException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileSectionNotFoundException
+ ConfigFileSectionNotFoundException()

11.17 ConfigFileStateException Class Reference

The state of the configuration file wasn't "good".

`#include <kmlFactories.h>`

Inheritance diagram for ConfigFileStateException: Collaboration diagram for ConfigFileStateException:

Public Member Functions

- ConfigFileStateException () throw ()

11.17.1 Detailed Description

The state of the configuration file wasn't "good".

Note:

error_number=-41

Definition at line 22 of file kmlFactories.h.

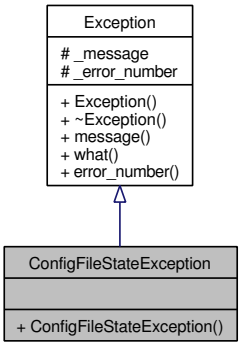
11.17.2 Constructor & Destructor Documentation

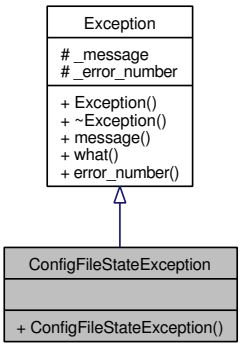
11.17.2.1 ConfigFileStateException::ConfigFileStateException () throw () [inline]

Definition at line 24 of file kmlFactories.h.

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

- include/KNI/kmlFactories.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileStateException
+ ConfigFileStateException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileStateException
+ ConfigFileStateException()

11.18 ConfigFileSubsectionNotFoundException Class Reference

The requested subsection could not be found.

`#include <kmlFactories.h>`

Inheritance diagram for ConfigFileSubsectionNotFoundException: Collaboration diagram for ConfigFileSubsectionNotFoundException:

Public Member Functions

- ConfigFileSubsectionNotFoundException (const std::string &attribute) throw ()

11.18.1 Detailed Description

The requested subsection could not be found.

Note:

error_number=-43

Definition at line 40 of file kmlFactories.h.

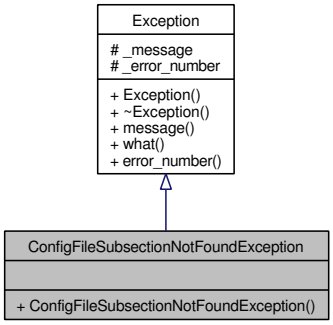
11.18.2 Constructor & Destructor Documentation

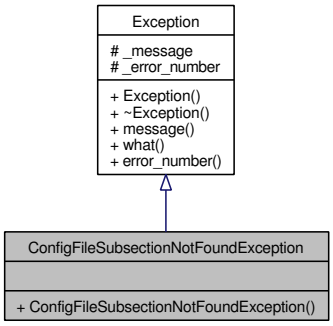
11.18.2.1 ConfigFileSubsectionNotFoundException::ConfigFileSubsectionNotFoundException (const std::string & *attribute*) throw () [inline]

Definition at line 42 of file kmlFactories.h.

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

- include/KNI/kmlFactories.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileSubsectionNotFoundException
+ ConfigFileSubsectionNotFoundException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileSubsectionNotFoundException
+ ConfigFileSubsectionNotFoundException()

11.19 ConfigFileSyntaxErrorException Class Reference

There was a syntax error in the configuration file.

`#include <kmlFactories.h>`

Inheritance diagram for ConfigFileSyntaxErrorException: Collaboration diagram for ConfigFileSyntaxErrorException:

Public Member Functions

- ConfigFileSyntaxErrorException (const std::string &line) throw ()

11.19.1 Detailed Description

There was a syntax error in the configuration file.

Note:

error_number=-45

Definition at line 58 of file kmlFactories.h.

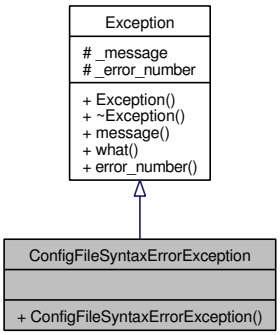
11.19.2 Constructor & Destructor Documentation

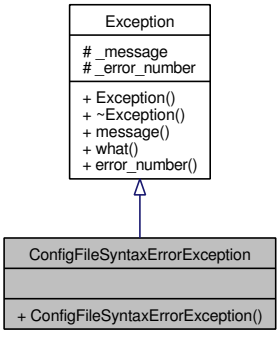
11.19.2.1 ConfigFileSyntaxErrorException::ConfigFileSyntaxErrorException (const std::string & *line*) throw () [inline]

Definition at line 60 of file kmlFactories.h.

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

- include/KNI/kmlFactories.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileSyntaxErrorException
+ ConfigFileSyntaxErrorException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ConfigFileSyntaxErrorException
+ ConfigFileSyntaxErrorException()

11.20 Context Struct Reference

`#include <exception.h>`

Public Member Functions

- Context (const char *)

11.20.1 Detailed Description

Definition at line 75 of file exception.h.

11.20.2 Constructor & Destructor Documentation

11.20.2.1 Context::Context (const char *) [inline]

Definition at line 76 of file exception.h.

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

- include/common/exception.h

11.21 CSctBase Class Reference

Sensor Controller class.

`#include <kmlSctBase.h>`

Collaboration diagram for CSctBase:

Public Member Functions

- const TSctGNL * GetGNL ()
- const TSctDAT * GetDAT ()
- virtual ~CSctBase ()
- bool init (CKatBase *_own, const TSctDesc _sctDesc, CCplBase *protocol)
- void recvDAT ()

  receive data

Protected Attributes

- TSctGNL gnl

  controller generals

- TSctDAT dat

  sensor data

- CCplBase * protocol

  protocol interface

Friends

- class CKatBase

11.21.1 Detailed Description

Sensor Controller class.

By using this class you can get access to the sensor data; to do so you should (after initialization) call 'recvDat()' to updated the internal 'TSctDAT dat' structure; after the updated you can read out the values by using the 'GetDAT()' function, which will return a constant pointer to the internal 'dat' structure.

Definition at line 72 of file kmlSctBase.h.

11.21.2 Constructor & Destructor Documentation

11.21.2.1 virtual CSctBase::~CSctBase () [inline, virtual]

Definition at line 88 of file kmlSctBase.h.

11.21.3 Member Function Documentation

11.21.3.1 const TSctGNL* CSctBase::GetGNL () [inline]

Definition at line 81 of file kmlSctBase.h.

References CKatBase::gnl.

11.21.3.2 const TSctDAT* CSctBase::GetDAT () [inline]

Definition at line 82 of file kmlSctBase.h.

11.21.3.3 bool CSctBase::init (CKatBase * *_own*, const TSctDesc *_sctDesc*, CCplBase * *protocol*)

11.21.3.4 void CSctBase::recvDAT ()

receive data

11.21.4 Friends And Related Function Documentation

11.21.4.1 friend class CKatBase [friend]

Definition at line 74 of file kmlSctBase.h.

11.21.5 Member Data Documentation

11.21.5.1 TSctGNL CSctBase::gnl [protected]

controller generals

Definition at line 77 of file kmlSctBase.h.

11.21.5.2 TSctDAT CSctBase::dat [protected]

sensor data

Definition at line 78 of file kmlSctBase.h.

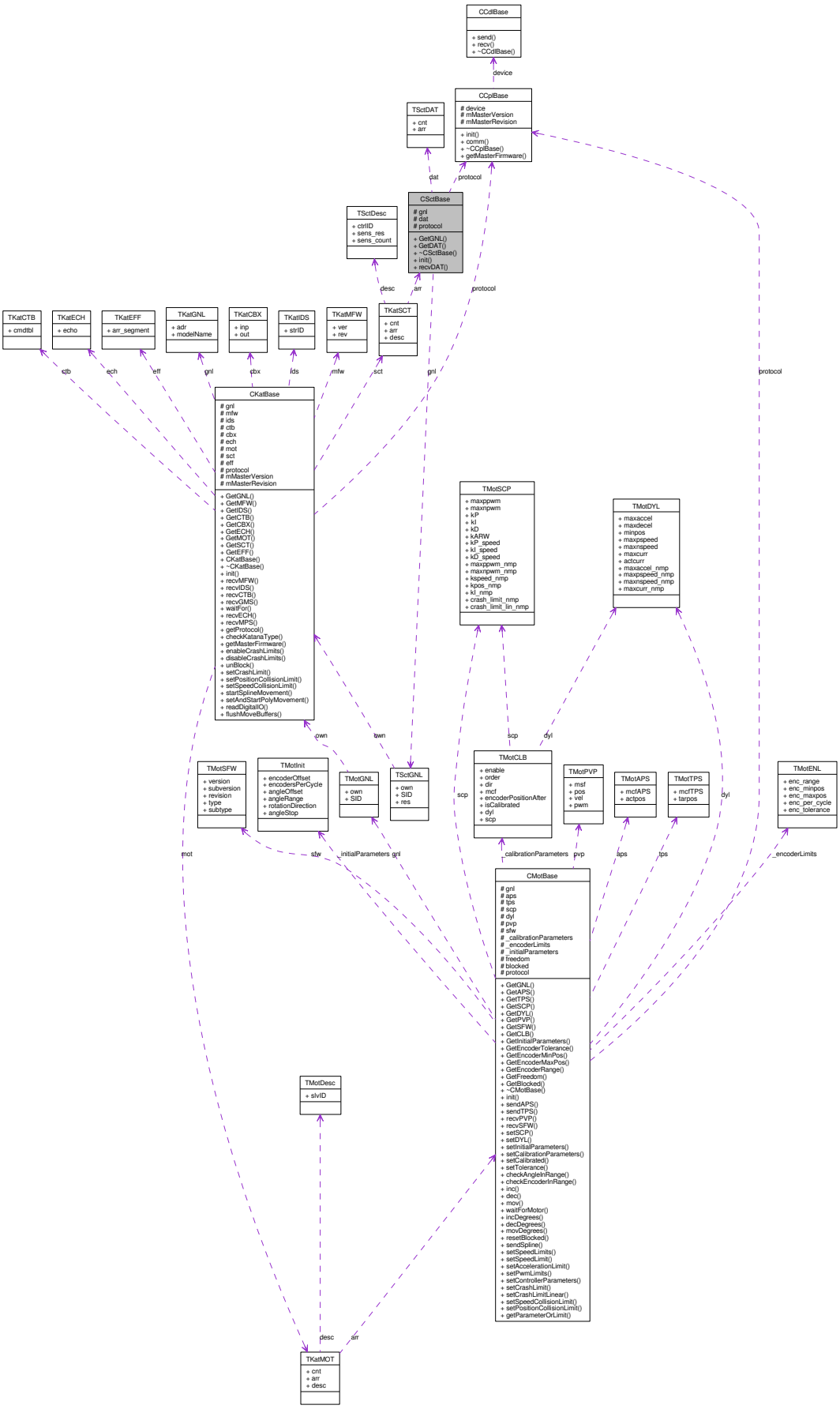
11.21.5.3 CCplBase* CSctBase::protocol [protected]

protocol interface

Definition at line 85 of file kmlSctBase.h.

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

- include/KNI/kmlSctBase.h


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
TSctDAT
+ cnt
+ arr
dat
protocol
CSctBase
gnl
dat
protocol
+ GetGNL()
+ GetDAT()
+ ~CSctBase()
+ init()
+ recvDAT()
TSctDesc
+ ctrlID
+ sens_res
+ sens_count
desc
arr
protocol
TKatCTB
+ cmdtbl
TKatECH
+ echo
TKatEFF
+ arr_segment
TKatGNL
+ adr
+ modelName
TKatCBX
+ inp
+ out
TKatIDS
+ strID
TKatMFW
+ ver
+ rev
TKatSCT
+ cnt
+ arr
+ desc
ctb
ech
eff
gnl
cbx
ids
mfw
sct
gnl
protocol
CKatBase
gnl
mfw
ids
ctb
cbx
ech
mot
sct
eff
protocol
mMasterVersion
mMasterRevision
+ GetGNL()
+ GetMFW()
+ GetIDS()
+ GetCTB()
+ GetCBX()
+ GetECH()
+ GetMOT()
+ GetSCT()
+ GetEFF()
+ CKatBase()
+ ~CKatBase()
+ init()
+ recvMFW()
+ recvIDS()
+ recvCTB()
+ recvGMS()
+ waitFor()
+ recvECH()
+ recvMPS()
+ getProtocol()
+ checkKatanaType()
+ getMasterFirmware()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ startSplineMovement()
+ setAndStartPolyMovement()
+ readDigitalIO()
+ flushMoveBuffers()
own
lown
TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kARW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ ki_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxnspeed
+ maxcurr
+ actcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxnspeed_nmp
+ maxcurr_nmp
scp
dyl
TMotSFW
+ version
+ subversion
+ revision
+ type
+ subtype
TMotInit
+ encoderOffset
+ encodersPerCycle
+ angleOffset
+ angleRange
+ rotationDirection
+ angleStop
TMotGNL
+ own
+ SID
TSctGNL
+ own
+ SID
+ res
TMotCLB
+ enable
+ order
+ dir
+ mcf
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp
TMotPVP
+ msf
+ pos
+ vel
+ pwm
TMotAPS
+ mcfAPS
+ actpos
TMotTPS
+ mcfTPS
+ tarpos
TMotENL
+ enc_range
+ enc_minpos
+ enc_maxpos
+ enc_per_cycle
+ enc_tolerance
mot
sfw
initialParameters
gnl
scp
_calibrationParameters
pvp
aps
tps
dyl
_encoderLimits
CMotBase
gnl
aps
tps
scp
dyl
pvp
sfw
_calibrationParameters
_encoderLimits
_initialParameters
freedom
blocked
protocol
+ GetGNL()
+ GetAPS()
+ GetTPS()
+ GetSCP()
+ GetDYL()
+ GetPVP()
+ GetSFW()
+ GetCLB()
+ GetInitialParameters()
+ GetEncoderTolerance()
+ GetEncoderMinPos()
+ GetEncoderMaxPos()
+ GetEncoderRange()
+ GetFreedom()
+ GetBlocked()
+ ~CMotBase()
+ init()
+ sendAPS()
+ sendTPS()
+ recvPVP()
+ recvSFW()
+ setSCP()
+ setDYL()
+ setInitialParameters()
+ setCalibrationParameters()
+ setCalibrated()
+ setTolerance()
+ checkAngleInRange()
+ checkEncoderInRange()
+ inc()
+ dec()
+ mov()
+ waitForMotor()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ resetBlocked()
+ sendSpline()
+ setSpeedLimits()
+ setSpeedLimit()
+ setAccelerationLimit()
+ setPwmLimits()
+ setControllerParameters()
+ setCrashLimit()
+ setCrashLimitLinear()
+ setSpeedCollisionLimit()
+ setPositionCollisionLimit()
+ getParameterOrLimit()
TMotDesc
+ slvID
desc
arr
TKatMOT
+ cnt
+ arr
+ desc

11.22 DeviceReadException Class Reference

Reading from the serial communication device failed.

`#include <cdlCOMExceptions.h>`

Inheritance diagram for DeviceReadException: Collaboration diagram for DeviceReadException:

Public Member Functions

- DeviceReadException (const std::string &port, const std::string os_msg) throw ()

11.22.1 Detailed Description

Reading from the serial communication device failed.

Note:

error_number=-13
Linux only: You get also the direct error message from the system

Definition at line 75 of file cdlCOMExceptions.h.

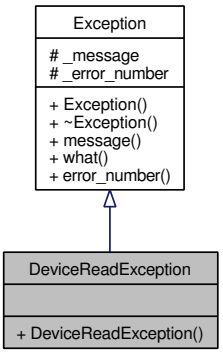
11.22.2 Constructor & Destructor Documentation

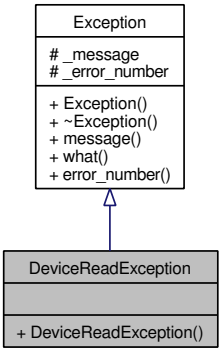
11.22.2.1 DeviceReadException::DeviceReadException (const std::string & *port*, const std::string *os_msg*) throw () [inline]

Definition at line 77 of file cdlCOMExceptions.h.

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

- include/KNI/cdlCOMExceptions.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
DeviceReadException
+ DeviceReadException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
DeviceReadException
+ DeviceReadException()

11.23 DeviceWriteException Class Reference

Writing to the serial communication device failed.

`#include <cdlCOMExceptions.h>`

Inheritance diagram for DeviceWriteException: Collaboration diagram for DeviceWriteException:

Public Member Functions

- DeviceWriteException (const std::string &port, const std::string os_msg) throw ()

11.23.1 Detailed Description

Writing to the serial communication device failed.

Note:

error_number=-14
Linux only: You get also the direct error message from the system

Definition at line 85 of file cdlCOMExceptions.h.

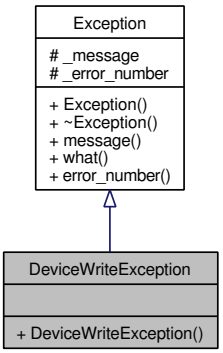
11.23.2 Constructor & Destructor Documentation

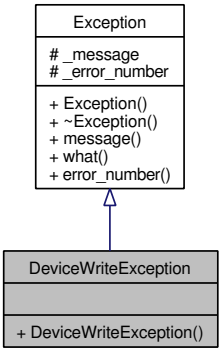
11.23.2.1 DeviceWriteException::DeviceWriteException (const std::string & *port*, const std::string *os_msg*) throw () [inline]

Definition at line 87 of file cdlCOMExceptions.h.

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

- include/KNI/cdlCOMExceptions.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
DeviceWriteException
+ DeviceWriteException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
DeviceWriteException
+ DeviceWriteException()

11.24 ErrorException Class Reference

The Katana returned an error string.

`#include <cdlCOMExceptions.h>`

Inheritance diagram for ErrorException: Collaboration diagram for ErrorException:

Public Member Functions

- ErrorException (const std::string &error) throw ()

11.24.1 Detailed Description

The Katana returned an error string.

Note:

error_number=-16

Definition at line 121 of file cdlCOMExceptions.h.

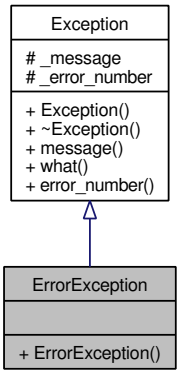
11.24.2 Constructor & Destructor Documentation

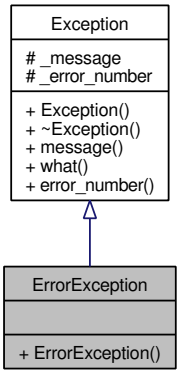
11.24.2.1 ErrorException::ErrorException (const std::string & *error*) throw () [inline]

Definition at line 123 of file cdlCOMExceptions.h.

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

- include/KNI/cdlCOMExceptions.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ErrorException
+ ErrorException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ErrorException
+ ErrorException()

11.25 Exception Class Reference

`#include <exception.h>`

Inheritance diagram for Exception:

Public Member Functions

- Exception (const std::string &message, const int error_number) throw ()
- virtual ~Exception () throw ()
- std::string message () const throw ()
- const char * what () const throw ()
- const int error_number () const throw ()

Protected Attributes

- const std::string _message
- const int _error_number

11.25.1 Detailed Description

Definition at line 79 of file exception.h.

11.25.2 Constructor & Destructor Documentation

11.25.2.1 Exception::Exception (const std::string & *message*, const int *error_number*) throw () [inline]

Definition at line 85 of file exception.h.

11.25.2.2 virtual Exception::~Exception () throw () [inline, virtual]

Definition at line 90 of file exception.h.

11.25.3 Member Function Documentation

11.25.3.1 std::string Exception::message () const throw () [inline]

Definition at line 93 of file exception.h.

11.25.3.2 const char* Exception::what () const throw () [inline]

Definition at line 96 of file exception.h.

11.25.3.3 const int Exception::error_number () const throw () [inline]

Definition at line 100 of file exception.h.

11.25.4 Member Data Documentation

11.25.4.1 const std::string Exception::_message [protected]

Definition at line 81 of file exception.h.

11.25.4.2 const int Exception::_error_number [protected]

Definition at line 82 of file exception.h.

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

- include/common/exception.h

11.26 FirmwareException Class Reference

Exception reported by the firmware.

`#include <cplSerial.h>`

Inheritance diagram for FirmwareException: Collaboration diagram for FirmwareException:

Public Member Functions

- FirmwareException (const std::string &error, const int error_number, const int axis, const char command) throw ()
- const int axis_number () const throw ()
- const char command_char () const throw ()

Protected Attributes

- int _axis_number

  axis number, if any

- char _command_char

  the command that caused the error

11.26.1 Detailed Description

Exception reported by the firmware.

Definition at line 52 of file cplSerial.h.

11.26.2 Constructor & Destructor Documentation

11.26.2.1 FirmwareException::FirmwareException (const std::string & *error*, const int *error_number*, const int *axis*, const char *command*) throw () [inline]

Definition at line 57 of file cplSerial.h.

11.26.3 Member Function Documentation

11.26.3.1 const int FirmwareException::axis_number () const throw () [inline]

Definition at line 61 of file cplSerial.h.

References _axis_number.

11.26.3.2 const char FirmwareException::command_char () const throw () [inline]

Definition at line 64 of file cplSerial.h.

References _command_char.

11.26.4 Member Data Documentation

11.26.4.1 int FirmwareException::_axis_number [protected]

axis number, if any

Definition at line 54 of file cplSerial.h.

Referenced by axis_number().

11.26.4.2 char FirmwareException::_command_char [protected]

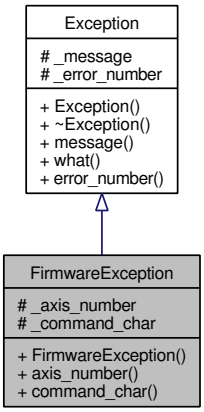
the command that caused the error

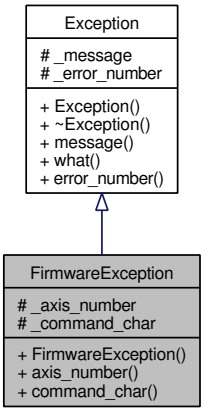
Definition at line 55 of file cplSerial.h.

Referenced by command_char().

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

- include/KNI/cplSerial.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
FirmwareException
_axis_number
_command_char
+ FirmwareException()
+ axis_number()
+ command_char()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
FirmwareException
_axis_number
_command_char
+ FirmwareException()
+ axis_number()
+ command_char()

11.27 FloatVector Struct Reference

`#include <libKinematics.h>`

Public Attributes

- int length
- float data [MaxDof]

11.27.1 Detailed Description

Definition at line 37 of file libKinematics.h.

11.27.2 Member Data Documentation

11.27.2.1 int FloatVector::length

Definition at line 39 of file libKinematics.h.

11.27.2.2 float FloatVector::data[MaxDof]

Definition at line 40 of file libKinematics.h.

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

- include/libKinematics.h

11.28 IntVector Struct Reference

`#include <libKinematics.h>`

Public Attributes

- int length
- int data [MaxDof]

11.28.1 Detailed Description

Definition at line 31 of file libKinematics.h.

11.28.2 Member Data Documentation

11.28.2.1 int IntVector::length

Definition at line 33 of file libKinematics.h.

11.28.2.2 int IntVector::data[MaxDof]

Definition at line 34 of file libKinematics.h.

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

- include/libKinematics.h

11.29 JointSpeedException Class Reference

Joint speed too high.

`#include <lmBase.h>`

Inheritance diagram for JointSpeedException: Collaboration diagram for JointSpeedException:

Public Member Functions

- JointSpeedException () throw ()

11.29.1 Detailed Description

Joint speed too high.

Note:

error_number = -70

Definition at line 52 of file lmBase.h.

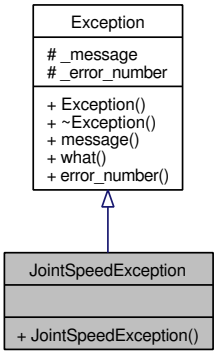
11.29.2 Constructor & Destructor Documentation

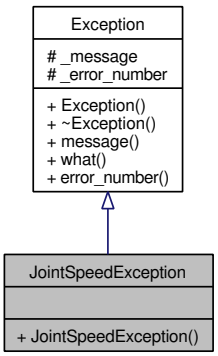
11.29.2.1 JointSpeedException::JointSpeedException () throw () [inline]

Definition at line 54 of file lmBase.h.

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

- include/KNI_LM/lmBase.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
JointSpeedException
+ JointSpeedException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
JointSpeedException
+ JointSpeedException()

11.30 KNI::KatanaKinematics Class Reference

The base class for all kinematic implementations.

`#include <KatanaKinematics.h>`

Inheritance diagram for KNI::KatanaKinematics:

Public Types

- typedef std::vector< KinematicParameters > parameter_container
- typedef std::vector< double > angles

  Being used to store angles (in radian).

- typedef std::vector< double > coordinates

  To store coordinates.

- typedef std::vector< double > metrics

  To store metrics, 'aka' the length's of the different segments of the robot.

- typedef std::vector< int > encoders

  To store encoders.

Public Member Functions

- virtual ~KatanaKinematics ()
- virtual void init (metrics const &length, parameter_container const &parameters)=0

  Initialize the parameters for the calculations.

- virtual void DK (coordinates &solution, encoders const &current_encoders) const =0

  Direct Kinematic.

- virtual void IK (encoders::iterator solution, coordinates const &pose, encoders const &cur_angles) const =0

  Inverse Kinematic.

11.30.1 Detailed Description

The base class for all kinematic implementations.

Definition at line 63 of file KatanaKinematics.h.

11.30.2 Member Typedef Documentation

11.30.2.1 typedef std::vector<KinematicParameters> KNI::KatanaKinematics::parameter_container

Definition at line 67 of file KatanaKinematics.h.

11.30.2.2 typedef std::vector<double> KNI::KatanaKinematics::angles

Being used to store angles (in radian).

Definition at line 71 of file KatanaKinematics.h.

11.30.2.3 typedef std::vector<double> KNI::KatanaKinematics::coordinates

To store coordinates.

Definition at line 74 of file KatanaKinematics.h.

11.30.2.4 typedef std::vector<double> KNI::KatanaKinematics::metrics

To store metrics, 'aka' the length's of the different segments of the robot.

Definition at line 77 of file KatanaKinematics.h.

11.30.2.5 typedef std::vector<int> KNI::KatanaKinematics::encoders

To store encoders.

Definition at line 80 of file KatanaKinematics.h.

11.30.3 Constructor & Destructor Documentation

11.30.3.1 virtual KNI::KatanaKinematics::~KatanaKinematics () [inline, virtual]

Definition at line 65 of file KatanaKinematics.h.

11.30.4 Member Function Documentation

11.30.4.1 virtual void KNI::KatanaKinematics::init (metrics const & *length*, parameter_container const & *parameters*) [pure virtual]

Initialize the parameters for the calculations.

This is needed to validate the calculated angles and to choose an appropriate solution You have to provide 5 or 6 length's and parameters, depending on you robot type

Implemented in KNI::KatanaKinematics5M180, KNI::KatanaKinematics6M180, KNI::KatanaKinematics6M90G, and KNI::KatanaKinematics6M90T.

11.30.4.2 virtual void KNI::KatanaKinematics::DK (coordinates & *solution*, encoders const & *current_encoders*) const [pure virtual]

Direct Kinematic.

Calculates the actual position in cartesian coordinates using the given encoders

Parameters:

solution This is where the algorithm will store the solution to (in cartesian coordinates)

current_encoders The encoder values which are being used for the calculation

Note:

strong guarantee provided

Implemented in KNI::KatanaKinematics5M180, KNI::KatanaKinematics6M180, KNI::KatanaKinematics6M90G, and KNI::KatanaKinematics6M90T.

11.30.4.3 virtual void KNI::KatanaKinematics::IK (encoders::iterator *solution*, coordinates const & *pose*, encoders const & *cur_angles*) const [pure virtual]

Inverse Kinematic.

Calculates one set of encoders (=one solution) for the given cartesian coordinates. You also have to provide the current encoders to allow the algorithm to choose between different valid solutions.

Parameters:

solution This is where the algorithm will store the solution to (in encoders)

pose The target position in cartesian coordinates plus the euler angles for the direction of the gripper

cur_angles The current angles (in encoders) of the robot

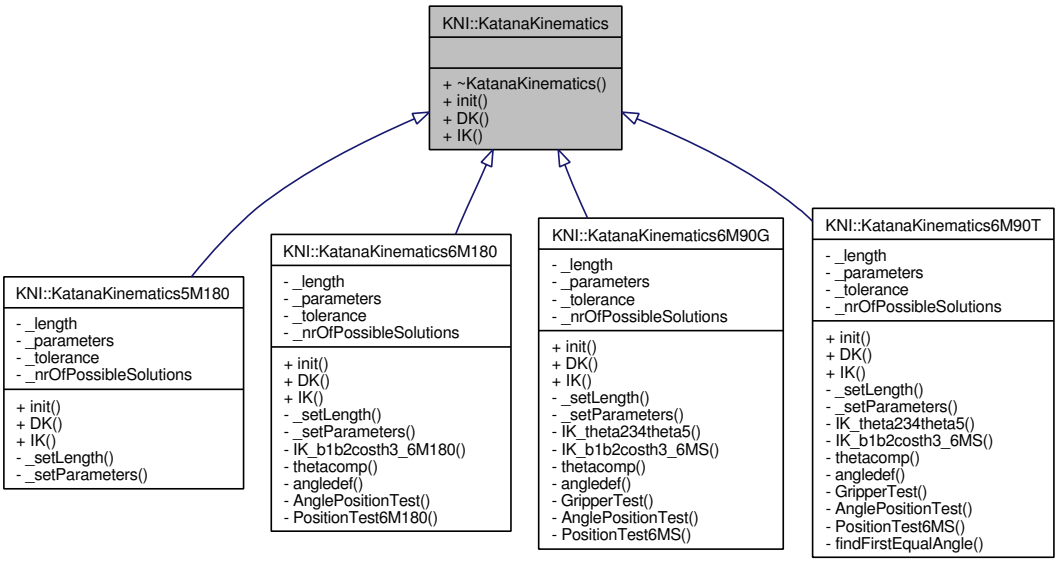
Note:

strong guarantee provided

Implemented in KNI::KatanaKinematics5M180, KNI::KatanaKinematics6M180, KNI::KatanaKinematics6M90G, and KNI::KatanaKinematics6M90T.

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

- include/KNI_InvKin/KatanaKinematics.h


KNI::KatanaKinematics
+ ~KatanaKinematics()
+ init()
+ DK()
+ IK()
KNI::KatanaKinematics5M180
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
KNI::KatanaKinematics6M180
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
- IK_b1b2costh3_6M180()
- thetacomp()
- angledef()
- AnglePositionTest()
- PositionTest6M180()
KNI::KatanaKinematics6M90G
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
- IK_theta234theta5()
- IK_b1b2costh3_6MS()
- thetacomp()
- angledef()
- GripperTest()
- AnglePositionTest()
- PositionTest6MS()
KNI::KatanaKinematics6M90T
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
- IK_theta234theta5()
- IK_b1b2costh3_6MS()
- thetacomp()
- angledef()
- GripperTest()
- AnglePositionTest()
- PositionTest6MS()
- findFirstEqualAngle()

11.31 KNI::KatanaKinematics5M180 Class Reference

`#include <KatanaKinematics5M180.h>`

Inheritance diagram for KNI::KatanaKinematics5M180: Collaboration diagram for KNI::KatanaKinematics5M180:

Public Member Functions

- void init (metrics const &length, parameter_container const &parameters)
  Initialize the parameters for the calculations.
- void DK (coordinates &solution, encoders const &current_encoders) const
  Direct Kinematic.
- void IK (encoders::iterator solution, coordinates const &pose, encoders const &cur_angles) const
  Inverse Kinematic.

Private Types

- typedef std::vector< angles_calc > angles_container

Private Member Functions

- void _setLength (metrics const &length)
- void _setParameters (parameter_container const &parameters)

Private Attributes

- metrics _length
- parameter_container _parameters

Static Private Attributes

- static const double _tolerance
- static const int _nrOfPossibleSolutions

Classes

- struct angles_calc
- struct position

11.31.1 Detailed Description

Author:

Tiziano Mueller <tiziano.mueller@neuronics.ch>
Christoph Voser <christoph.voser@neuronics.ch>

Definition at line 39 of file KatanaKinematics5M180.h.

11.31.2 Member Typedef Documentation

11.31.2.1 typedef std::vector<angles_calc> KNI::KatanaKinematics5M180::angles_container [private]

Definition at line 70 of file KatanaKinematics5M180.h.

11.31.3 Member Function Documentation

11.31.3.1 void KNI::KatanaKinematics5M180::init (metrics const & *length*, parameter_container const & *parameters*) [virtual]

Initialize the parameters for the calculations.

This is needed to validate the calculated angles and to choose an appropriate solution You have to provide 5 or 6 length's and parameters, depending on you robot type

Implements KNI::KatanaKinematics.

11.31.3.2 void KNI::KatanaKinematics5M180::DK (coordinates & *solution*, encoders const & *current_encoders*) const [virtual]

Direct Kinematic.

Calculates the actual position in cartesian coordinates using the given encoders

Parameters:

solution This is where the algorithm will store the solution to (in cartesian coordinates)

current_encoders The encoder values which are being used for the calculation

Note:

strong guarantee provided

Implements KNI::KatanaKinematics.

11.31.3.3 void KNI::KatanaKinematics5M180::IK (encoders::iterator *solution*, coordinates const & *pose*, encoders const & *cur_angles*) const [virtual]

Inverse Kinematic.

Calculates one set of encoders (=one solution) for the given cartesian coordinates. You also have to provide the current encoders to allow the algorithm to choose between different valid solutions.

Parameters:

solution This is where the algorithm will store the solution to (in encoders)

pose The target position in cartesian coordinates plus the euler angles for the direction of the gripper

cur_angles The current angles (in encoders) of the robot

Note:

strong guarantee provided

Implements KNI::KatanaKinematics.

11.31.3.4 void KNI::KatanaKinematics5M180::_setLength (metrics const & *length*) [inline, private]

Definition at line 78 of file KatanaKinematics5M180.h.

11.31.3.5 void KNI::KatanaKinematics5M180::_setParameters (parameter_container const & *parameters*) [inline, private]

Definition at line 79 of file KatanaKinematics5M180.h.

11.31.4 Member Data Documentation

11.31.4.1 metrics KNI::KatanaKinematics5M180::_length [private]

Definition at line 72 of file KatanaKinematics5M180.h.

11.31.4.2 parameter_container KNI::KatanaKinematics5M180::_parameters [private]

Definition at line 73 of file KatanaKinematics5M180.h.

11.31.4.3 const double KNI::KatanaKinematics5M180::_tolerance [static, private]

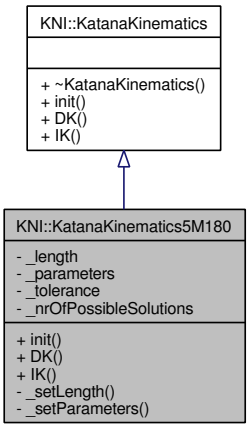
Definition at line 75 of file KatanaKinematics5M180.h.

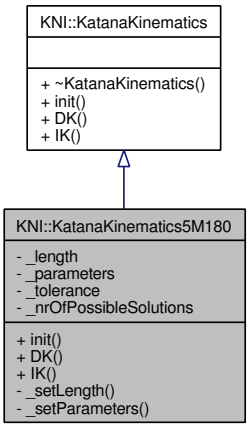
11.31.4.4 const int KNI::KatanaKinematics5M180::_nrOfPossibleSolutions [static, private]

Definition at line 76 of file KatanaKinematics5M180.h.

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

- include/KNI_InvKin/KatanaKinematics5M180.h


KNI::KatanaKinematics
+ ~KatanaKinematics()
+ init()
+ DK()
+ IK()
KNI::KatanaKinematics5M180
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()


KNI::KatanaKinematics
+ ~KatanaKinematics()
+ init()
+ DK()
+ IK()
KNI::KatanaKinematics5M180
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()

11.32 KNI::KatanaKinematics5M180::angles_calc Struct Reference

Public Attributes

- double theta1
- double theta2
- double theta3
- double theta4
- double theta5
- double theta234
- double b1
- double b2
- double costh3

11.32.1 Detailed Description

Definition at line 58 of file KatanaKinematics5M180.h.

11.32.2 Member Data Documentation

11.32.2.1 double KNI::KatanaKinematics5M180::angles_calc::theta1

Definition at line 59 of file KatanaKinematics5M180.h.

11.32.2.2 double KNI::KatanaKinematics5M180::angles_calc::theta2

Definition at line 60 of file KatanaKinematics5M180.h.

11.32.2.3 double KNI::KatanaKinematics5M180::angles_calc::theta3

Definition at line 61 of file KatanaKinematics5M180.h.

11.32.2.4 double KNI::KatanaKinematics5M180::angles_calc::theta4

Definition at line 62 of file KatanaKinematics5M180.h.

11.32.2.5 double KNI::KatanaKinematics5M180::angles_calc::theta5

Definition at line 63 of file KatanaKinematics5M180.h.

11.32.2.6 double KNI::KatanaKinematics5M180::angles_calc::theta234

Definition at line 64 of file KatanaKinematics5M180.h.

11.32.2.7 double KNI::KatanaKinematics5M180::angles_calc::b1

Definition at line 65 of file KatanaKinematics5M180.h.

11.32.2.8 double KNI::KatanaKinematics5M180::angles_calc::b2

Definition at line 66 of file KatanaKinematics5M180.h.

11.32.2.9 double KNI::KatanaKinematics5M180::angles_calc::costh3

Definition at line 67 of file KatanaKinematics5M180.h.

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

- include/KNI_InvKin/KatanaKinematics5M180.h

11.33 KNI::KatanaKinematics5M180::position Struct Reference

Public Attributes

- double x
- double y
- double z

11.33.1 Detailed Description

Definition at line 52 of file KatanaKinematics5M180.h.

11.33.2 Member Data Documentation

11.33.2.1 double KNI::KatanaKinematics5M180::position::x

Definition at line 53 of file KatanaKinematics5M180.h.

11.33.2.2 double KNI::KatanaKinematics5M180::position::y

Definition at line 54 of file KatanaKinematics5M180.h.

11.33.2.3 double KNI::KatanaKinematics5M180::position::z

Definition at line 55 of file KatanaKinematics5M180.h.

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

- include/KNI_InvKin/KatanaKinematics5M180.h

11.34 KNI::KatanaKinematics6M180 Class Reference

`#include <KatanaKinematics6M180.h>`

Inheritance diagram for KNI::KatanaKinematics6M180: Collaboration diagram for KNI::KatanaKinematics6M180:

Public Member Functions

- void init (metrics const &length, parameter_container const &parameters)

  Initialize the parameters for the calculations.

- void DK (coordinates &solution, encoders const &current_encoders) const

  Direct Kinematic.

- void IK (encoders::iterator solution, coordinates const &pose, encoders const &cur_angles) const

  Inverse Kinematic.

Private Types

- typedef std::vector< angles_calc > angles_container

Private Member Functions

- void _setLength (metrics const &length)
- void _setParameters (parameter_container const &parameters)
- void IK_b1b2costh3_6M180 (angles_calc &a, const position &p) const
- void thetacomp (angles_calc &a, const position &p_m) const
- bool angledef (angles_calc &a) const
- bool AnglePositionTest (const angles_calc &a) const
- bool PositionTest6M180 (const angles_calc &a, const position &p) const

Private Attributes

- metrics _length
- parameter_container _parameters

Static Private Attributes

- static const double _tolerance
- static const int _nrOfPossibleSolutions

Classes

- struct angles_calc
- struct position

11.34.1 Detailed Description

Author:

Tiziano Mueller <tiziano.mueller@neuronics.ch>
Christoph Voser <christoph.voser@neuronics.ch>

Definition at line 40 of file KatanaKinematics6M180.h.

11.34.2 Member Typedef Documentation

11.34.2.1 typedef std::vector<angles_calc> KNI::KatanaKinematics6M180::angles_container [private]

Definition at line 71 of file KatanaKinematics6M180.h.

11.34.3 Member Function Documentation

11.34.3.1 void KNI::KatanaKinematics6M180::init (metrics const & *length*, parameter_container const & *parameters*) [virtual]

Initialize the parameters for the calculations.

This is needed to validate the calculated angles and to choose an appropriate solution You have to provide 5 or 6 length's and parameters, depending on you robot type

Implements KNI::KatanaKinematics.

11.34.3.2 void KNI::KatanaKinematics6M180::DK (coordinates & *solution*, encoders const & *current_encoders*) const [virtual]

Direct Kinematic.

Calculates the actual position in cartesian coordinates using the given encoders

Parameters:

solution This is where the algorithm will store the solution to (in cartesian coordinates)

current_encoders The encoder values which are being used for the calculation

Note:

strong guarantee provided

Implements KNI::KatanaKinematics.

11.34.3.3 void KNI::KatanaKinematics6M180::IK (encoders::iterator *solution*, coordinates const & *pose*, encoders const & *cur_angles*) const [virtual]

Inverse Kinematic.

Calculates one set of encoders (=one solution) for the given cartesian coordinates. You also have to provide the current encoders to allow the algorithm to choose between different valid solutions.

Parameters:

solution This is where the algorithm will store the solution to (in encoders)

pose The target position in cartesian coordinates plus the euler angles for the direction of the gripper

cur_angles The current angles (in encoders) of the robot

Note:

strong guarantee provided

Implements KNI::KatanaKinematics.

11.34.3.4 void KNI::KatanaKinematics6M180::_setLength (metrics const & *length*) [inline, private]

Definition at line 79 of file KatanaKinematics6M180.h.

11.34.3.5 void KNI::KatanaKinematics6M180::_setParameters (parameter_container const & *parameters*) [inline, private]

Definition at line 80 of file KatanaKinematics6M180.h.

11.34.3.6 void KNI::KatanaKinematics6M180::IK_b1b2costh3_6M180 (angles_calc & *a*, const position & *p*) const [private]

11.34.3.7 void KNI::KatanaKinematics6M180::thetacomp (angles_calc & *a*, const position & *p_m*) const [private]

11.34.3.8 bool KNI::KatanaKinematics6M180::angledef (angles_calc & *a*) const [private]

11.34.3.9 bool KNI::KatanaKinematics6M180::AnglePositionTest (const angles_calc & *a*) const [private]

11.34.3.10 bool KNI::KatanaKinematics6M180::PositionTest6M180 (const angles_calc & *a*, const position & *p*) const [private]

11.34.4 Member Data Documentation

11.34.4.1 metrics KNI::KatanaKinematics6M180::_length [private]

Definition at line 73 of file KatanaKinematics6M180.h.

11.34.4.2 parameter_container KNI::KatanaKinematics6M180::_parameters [private]

Definition at line 74 of file KatanaKinematics6M180.h.

11.34.4.3 const double KNI::KatanaKinematics6M180::_tolerance [static, private]

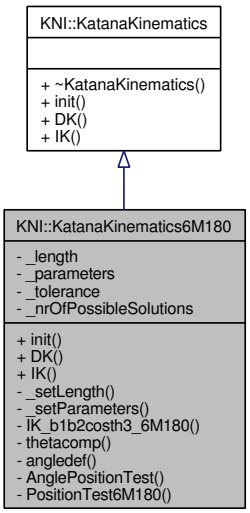
Definition at line 76 of file KatanaKinematics6M180.h.

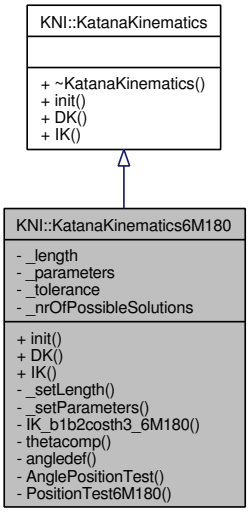
11.34.4.4 const int KNI::KatanaKinematics6M180::_nrOfPossibleSolutions [static, private]

Definition at line 77 of file KatanaKinematics6M180.h.

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

- include/KNI_InvKin/KatanaKinematics6M180.h


KNI::KatanaKinematics
+ ~KatanaKinematics()
+ init()
+ DK()
+ IK()
KNI::KatanaKinematics6M180
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
- IK_b1b2costh3_6M180()
- thetacomp()
- angledef()
- AnglePositionTest()
- PositionTest6M180()


KNI::KatanaKinematics
+ ~KatanaKinematics()
+ init()
+ DK()
+ IK()
KNI::KatanaKinematics6M180
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
- IK_b1b2costh3_6M180()
- thetacomp()
- angledef()
- AnglePositionTest()
- PositionTest6M180()

11.35 KNI::KatanaKinematics6M180::angles_calc Struct Reference

Public Attributes

- double theta1
- double theta2
- double theta3
- double theta4
- double theta5
- double theta234
- double b1
- double b2
- double costh3

11.35.1 Detailed Description

Definition at line 59 of file KatanaKinematics6M180.h.

11.35.2 Member Data Documentation

11.35.2.1 double KNI::KatanaKinematics6M180::angles_calc::theta1

Definition at line 60 of file KatanaKinematics6M180.h.

11.35.2.2 double KNI::KatanaKinematics6M180::angles_calc::theta2

Definition at line 61 of file KatanaKinematics6M180.h.

11.35.2.3 double KNI::KatanaKinematics6M180::angles_calc::theta3

Definition at line 62 of file KatanaKinematics6M180.h.

11.35.2.4 double KNI::KatanaKinematics6M180::angles_calc::theta4

Definition at line 63 of file KatanaKinematics6M180.h.

11.35.2.5 double KNI::KatanaKinematics6M180::angles_calc::theta5

Definition at line 64 of file KatanaKinematics6M180.h.

11.35.2.6 double KNI::KatanaKinematics6M180::angles_calc::theta234

Definition at line 65 of file KatanaKinematics6M180.h.

11.35.2.7 double KNI::KatanaKinematics6M180::angles_calc::b1

Definition at line 66 of file KatanaKinematics6M180.h.

11.35.2.8 double KNI::KatanaKinematics6M180::angles_calc::b2

Definition at line 67 of file KatanaKinematics6M180.h.

11.35.2.9 double KNI::KatanaKinematics6M180::angles_calc::costh3

Definition at line 68 of file KatanaKinematics6M180.h.

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

- include/KNI_InvKin/KatanaKinematics6M180.h

11.36 KNI::KatanaKinematics6M180::position Struct Reference

Public Attributes

- double x
- double y
- double z

11.36.1 Detailed Description

Definition at line 53 of file KatanaKinematics6M180.h.

11.36.2 Member Data Documentation

11.36.2.1 double KNI::KatanaKinematics6M180::position::x

Definition at line 54 of file KatanaKinematics6M180.h.

11.36.2.2 double KNI::KatanaKinematics6M180::position::y

Definition at line 55 of file KatanaKinematics6M180.h.

11.36.2.3 double KNI::KatanaKinematics6M180::position::z

Definition at line 56 of file KatanaKinematics6M180.h.

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

- include/KNI_InvKin/KatanaKinematics6M180.h

11.37 KNI::KatanaKinematics6M90G Class Reference

`#include <KatanaKinematics6M90G.h>`

Inheritance diagram for KNI::KatanaKinematics6M90G: Collaboration diagram for KNI::KatanaKinematics6M90G:

Public Member Functions

- void init (metrics const &length, parameter_container const &parameters)

  Initialize the parameters for the calculations.

- void DK (coordinates &solution, encoders const &current_encoders) const

  Direct Kinematic.

- void IK (encoders::iterator solution, coordinates const &pose, encoders const &cur_angles) const

  Inverse Kinematic.

Private Types

- typedef std::vector< angles_calc > angles_container

Private Member Functions

- void _setLength (metrics const &length)
- void _setParameters (parameter_container const &parameters)
- void IK_theta234theta5 (angles_calc &angle, const position &p_gr) const
- void IK_b1b2costh3_6MS (angles_calc &a, const position &p) const
- void thetacomp (angles_calc &a, const position &p_m) const
- bool angledef (angles_calc &a) const
- bool GripperTest (const position &p_gr, const angles_calc &angle) const
- bool AnglePositionTest (const angles_calc &a) const
- bool PositionTest6MS (const angles_calc &a, const position &p) const

Private Attributes

- metrics _length
- parameter_container _parameters

Static Private Attributes

- static const double _tolerance
- static const int _nrOfPossibleSolutions

Classes

- struct angles_calc
- struct position

11.37.1 Detailed Description

Author:

Tiziano Mueller <tiziano.mueller@neuronics.ch>
Christoph Voser <christoph.voser@neuronics.ch>

Definition at line 39 of file KatanaKinematics6M90G.h.

11.37.2 Member Typedef Documentation

11.37.2.1 typedef std::vector<angles_calc> KNI::KatanaKinematics6M90G::angles_container [private]

Definition at line 70 of file KatanaKinematics6M90G.h.

11.37.3 Member Function Documentation

11.37.3.1 void KNI::KatanaKinematics6M90G::init (metrics const & *length*, parameter_container const & *parameters*) [virtual]

Initialize the parameters for the calculations.

This is needed to validate the calculated angles and to choose an appropriate solution You have to provide 5 or 6 length's and parameters, depending on you robot type

Implements KNI::KatanaKinematics.

11.37.3.2 void KNI::KatanaKinematics6M90G::DK (coordinates & *solution*, encoders const & *current_encoders*) const [virtual]

Direct Kinematic.

Calculates the actual position in cartesian coordinates using the given encoders

Parameters:

solution This is where the algorithm will store the solution to (in cartesian coordinates)

current_encoders The encoder values which are being used for the calculation

Note:

strong guarantee provided

Implements KNI::KatanaKinematics.

11.37.3.3 void KNI::KatanaKinematics6M90G::IK (encoders::iterator *solution*, coordinates const & *pose*, encoders const & *cur_angles*) const [virtual]

Inverse Kinematic.

Calculates one set of encoders (=one solution) for the given cartesian coordinates. You also have to provide the current encoders to allow the algorithm to choose between different valid solutions.

Parameters:

solution This is where the algorithm will store the solution to (in encoders)

pose The target position in cartesian coordinates plus the euler angles for the direction of the gripper

cur_angles The current angles (in encoders) of the robot

Note:

strong guarantee provided

Implements KNI::KatanaKinematics.

11.37.3.4 void KNI::KatanaKinematics6M90G::_setLength (metrics const & *length*) [inline, private]

Definition at line 78 of file KatanaKinematics6M90G.h.

11.37.3.5 void KNI::KatanaKinematics6M90G::_setParameters (parameter_container const & *parameters*) [inline, private]

Definition at line 79 of file KatanaKinematics6M90G.h.

11.37.3.6 void KNI::KatanaKinematics6M90G::IK_theta234theta5 (angles_calc & *angle*, const position & *p_gr*) const [private]

11.37.3.7 void KNI::KatanaKinematics6M90G::IK_b1b2costh3_6MS (angles_calc & *a*, const position & *p*) const [private]

11.37.3.8 void KNI::KatanaKinematics6M90G::thetacomp (angles_calc & *a*, const position & *p_m*) const [private]

11.37.3.9 bool KNI::KatanaKinematics6M90G::angledef (angles_calc & *a*) const [private]

11.37.3.10 bool KNI::KatanaKinematics6M90G::GripperTest (const position & *p_gr*, const angles_calc & *angle*) const [private]

11.37.3.11 bool KNI::KatanaKinematics6M90G::AnglePositionTest (const angles_calc & *a*) const [private]

11.37.3.12 bool KNI::KatanaKinematics6M90G::PositionTest6MS (const angles_calc & *a*, const position & *p*) const [private]

11.37.4 Member Data Documentation

11.37.4.1 metrics KNI::KatanaKinematics6M90G::_length [private]

Definition at line 72 of file KatanaKinematics6M90G.h.

11.37.4.2 parameter_container KNI::KatanaKinematics6M90G::_parameters [private]

Definition at line 73 of file KatanaKinematics6M90G.h.

11.37.4.3 const double KNI::KatanaKinematics6M90G::_tolerance [static, private]

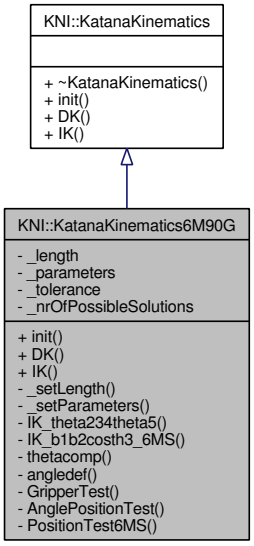
Definition at line 75 of file KatanaKinematics6M90G.h.

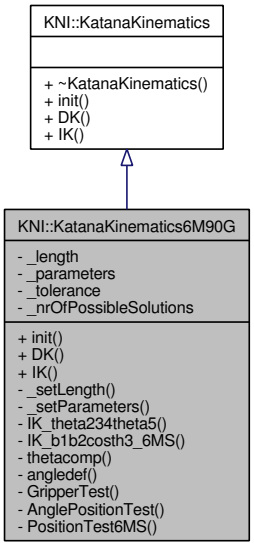
11.37.4.4 const int KNI::KatanaKinematics6M90G::_nrOfPossibleSolutions [static, private]

Definition at line 76 of file KatanaKinematics6M90G.h.

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

- include/KNI_InvKin/KatanaKinematics6M90G.h


KNI::KatanaKinematics
+ ~KatanaKinematics()
+ init()
+ DK()
+ IK()
KNI::KatanaKinematics6M90G
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
- IK_theta234theta5()
- IK_b1b2costh3_6MS()
- thetacomp()
- angledef()
- GripperTest()
- AnglePositionTest()
- PositionTest6MS()


KNI::KatanaKinematics
+ ~KatanaKinematics()
+ init()
+ DK()
+ IK()
KNI::KatanaKinematics6M90G
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
- IK_theta234theta5()
- IK_b1b2costh3_6MS()
- thetacomp()
- angledef()
- GripperTest()
- AnglePositionTest()
- PositionTest6MS()

11.38 KNI::KatanaKinematics6M90G::angles_calc Struct Reference

Public Attributes

- double theta1
- double theta2
- double theta3
- double theta4
- double theta5
- double theta234
- double b1
- double b2
- double costh3

11.38.1 Detailed Description

Definition at line 58 of file KatanaKinematics6M90G.h.

11.38.2 Member Data Documentation

11.38.2.1 double KNI::KatanaKinematics6M90G::angles_calc::theta1

Definition at line 59 of file KatanaKinematics6M90G.h.

11.38.2.2 double KNI::KatanaKinematics6M90G::angles_calc::theta2

Definition at line 60 of file KatanaKinematics6M90G.h.

11.38.2.3 double KNI::KatanaKinematics6M90G::angles_calc::theta3

Definition at line 61 of file KatanaKinematics6M90G.h.

11.38.2.4 double KNI::KatanaKinematics6M90G::angles_calc::theta4

Definition at line 62 of file KatanaKinematics6M90G.h.

11.38.2.5 double KNI::KatanaKinematics6M90G::angles_calc::theta5

Definition at line 63 of file KatanaKinematics6M90G.h.

11.38.2.6 double KNI::KatanaKinematics6M90G::angles_calc::theta234

Definition at line 64 of file KatanaKinematics6M90G.h.

11.38.2.7 double KNI::KatanaKinematics6M90G::angles_calc::b1

Definition at line 65 of file KatanaKinematics6M90G.h.

11.38.2.8 double KNI::KatanaKinematics6M90G::angles_calc::b2

Definition at line 66 of file KatanaKinematics6M90G.h.

11.38.2.9 double KNI::KatanaKinematics6M90G::angles_calc::costh3

Definition at line 67 of file KatanaKinematics6M90G.h.

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

- include/KNI_InvKin/KatanaKinematics6M90G.h

11.39 KNI::KatanaKinematics6M90G::position Struct Reference

Public Attributes

- double x
- double y
- double z

11.39.1 Detailed Description

Definition at line 52 of file KatanaKinematics6M90G.h.

11.39.2 Member Data Documentation

11.39.2.1 double KNI::KatanaKinematics6M90G::position::x

Definition at line 53 of file KatanaKinematics6M90G.h.

11.39.2.2 double KNI::KatanaKinematics6M90G::position::y

Definition at line 54 of file KatanaKinematics6M90G.h.

11.39.2.3 double KNI::KatanaKinematics6M90G::position::z

Definition at line 55 of file KatanaKinematics6M90G.h.

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

- include/KNI_InvKin/KatanaKinematics6M90G.h

11.40 KNI::KatanaKinematics6M90T Class Reference

`#include <KatanaKinematics6M90T.h>`

Inheritance diagram for KNI::KatanaKinematics6M90T: Collaboration diagram for KNI::KatanaKinematics6M90T:

Public Member Functions

- void init (metrics const &length, parameter_container const &parameters)
  Initialize the parameters for the calculations.
- void DK (coordinates &solution, encoders const &current_encoders) const
  Direct Kinematic.
- void IK (encoders::iterator solution, coordinates const &pose, encoders const &cur_angles) const
  Inverse Kinematic.

Private Types

- typedef std::vector< angles_calc > angles_container

Private Member Functions

- void _setLength (metrics const &length)
- void _setParameters (parameter_container const &parameters)
- void IK_theta234theta5 (angles_calc &angle, const position &p_gr) const
- void IK_b1b2costh3_6MS (angles_calc &a, const position &p) const
- void thetacomp (angles_calc &a, const position &p_m, const coordinates &pose) const
- bool angledef (angles_calc &a) const
- bool GripperTest (const position &p_gr, const angles_calc &angle) const
- bool AnglePositionTest (const angles_calc &a) const
- bool PositionTest6MS (const double &theta1, const double &theta2, const double &theta3, const double &theta234, const position &p) const
- double findFirstEqualAngle (const angles &v1, const angles &v2) const

Private Attributes

- metrics _length
- parameter_container _parameters

Static Private Attributes

- static const double _tolerance
- static const int _nrOfPossibleSolutions

Classes

- struct angles_calc
- struct position

11.40.1 Detailed Description

Author:

Tiziano Mueller <tiziano.mueller@neuronics.ch>
Christoph Voser <christoph.voser@neuronics.ch>

Definition at line 39 of file KatanaKinematics6M90T.h.

11.40.2 Member Typedef Documentation

11.40.2.1 typedef std::vector<angles_calc> KNI::KatanaKinematics6M90T::angles_container [private]

Definition at line 71 of file KatanaKinematics6M90T.h.

11.40.3 Member Function Documentation

11.40.3.1 void KNI::KatanaKinematics6M90T::init (metrics const & *length*, parameter_container const & *parameters*) [virtual]

Initialize the parameters for the calculations.

This is needed to validate the calculated angles and to choose an appropriate solution You have to provide 5 or 6 length's and parameters, depending on you robot type

Implements KNI::KatanaKinematics.

11.40.3.2 void KNI::KatanaKinematics6M90T::DK (coordinates & *solution*, encoders const & *current_encoders*) const [virtual]

Direct Kinematic.

Calculates the actual position in cartesian coordinates using the given encoders

Parameters:

solution This is where the algorithm will store the solution to (in cartesian coordinates)

current_encoders The encoder values which are being used for the calculation

Note:

strong guarantee provided

Implements KNI::KatanaKinematics.

11.40.3.3 void KNI::KatanaKinematics6M90T::IK (encoders::iterator *solution*, coordinates const & *pose*, encoders const & *cur_angles*) const [virtual]

Inverse Kinematic.

Calculates one set of encoders (=one solution) for the given cartesian coordinates. You also have to provide the current encoders to allow the algorithm to choose between different valid solutions.

Parameters:

solution This is where the algorithm will store the solution to (in encoders)

pose The target position in cartesian coordinates plus the euler angles for the direction of the gripper

cur_angles The current angles (in encoders) of the robot

Note:

strong guarantee provided

Implements KNI::KatanaKinematics.

11.40.3.4 void KNI::KatanaKinematics6M90T::_setLength (metrics const & *length*) [inline, private]

Definition at line 79 of file KatanaKinematics6M90T.h.

11.40.3.5 void KNI::KatanaKinematics6M90T::_setParameters (parameter_container const & *parameters*) [inline, private]

Definition at line 80 of file KatanaKinematics6M90T.h.

11.40.3.6 void KNI::KatanaKinematics6M90T::IK_theta234theta5 (angles_calc & *angle*, const position & *p_gr*) const [private]

11.40.3.7 void KNI::KatanaKinematics6M90T::IK_b1b2costh3_6MS (angles_calc & *a*, const position & *p*) const [private]

11.40.3.8 void KNI::KatanaKinematics6M90T::thetacomp (angles_calc & *a*, const position & *p_m*, const coordinates & *pose*) const [private]

11.40.3.9 bool KNI::KatanaKinematics6M90T::angledef (angles_calc & *a*) const [private]

11.40.3.10 bool KNI::KatanaKinematics6M90T::GripperTest (const position & *p_gr*, const angles_calc & *angle*) const [private]

11.40.3.11 bool KNI::KatanaKinematics6M90T::AnglePositionTest (const angles_calc & *a*) const [private]

11.40.3.12 bool KNI::KatanaKinematics6M90T::PositionTest6MS (const double & *theta1*, const double & *theta2*, const double & *theta3*, const double & *theta234*, const position & *p*) const [private]

11.40.3.13 double KNI::KatanaKinematics6M90T::findFirstEqualAngle (const angles & *v1*, const angles & *v2*) const [private]

11.40.4 Member Data Documentation

11.40.4.1 metrics KNI::KatanaKinematics6M90T::_length [private]

Definition at line 73 of file KatanaKinematics6M90T.h.

11.40.4.2 parameter_container KNI::KatanaKinematics6M90T::_parameters [private]

Definition at line 74 of file KatanaKinematics6M90T.h.

11.40.4.3 const double KNI::KatanaKinematics6M90T::_tolerance [static, private]

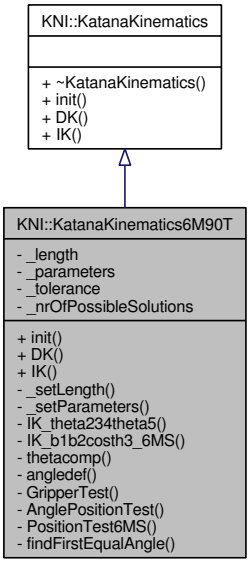
Definition at line 76 of file KatanaKinematics6M90T.h.

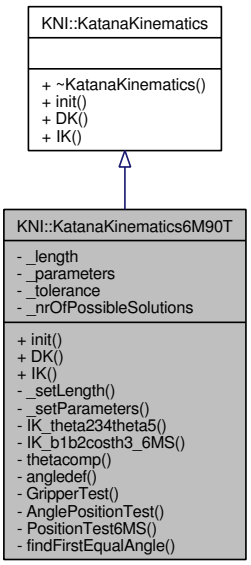
11.40.4.4 const int KNI::KatanaKinematics6M90T::_nrOfPossibleSolutions [static, private]

Definition at line 77 of file KatanaKinematics6M90T.h.

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

- include/KNI_InvKin/KatanaKinematics6M90T.h


KNI::KatanaKinematics
+ ~KatanaKinematics()
+ init()
+ DK()
+ IK()
KNI::KatanaKinematics6M90T
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
- IK_theta234theta5()
- IK_b1b2costh3_6MS()
- thetacomp()
- angledef()
- GripperTest()
- AnglePositionTest()
- PositionTest6MS()
- findFirstEqualAngle()


KNI::KatanaKinematics
+ ~KatanaKinematics()
+ init()
+ DK()
+ IK()
KNI::KatanaKinematics6M90T
- _length
- _parameters
- _tolerance
- _nrOfPossibleSolutions
+ init()
+ DK()
+ IK()
- _setLength()
- _setParameters()
- IK_theta234theta5()
- IK_b1b2costh3_6MS()
- thetacomp()
- angledef()
- GripperTest()
- AnglePositionTest()
- PositionTest6MS()
- findFirstEqualAngle()

11.41 KNI::KatanaKinematics6M90T::angles_calc Struct Reference

Public Attributes

- double theta1
- double theta2
- double theta3
- double theta4
- double theta5
- double theta6
- double theta234
- double b1
- double b2
- double costh3

11.41.1 Detailed Description

Definition at line 58 of file KatanaKinematics6M90T.h.

11.41.2 Member Data Documentation

11.41.2.1 double KNI::KatanaKinematics6M90T::angles_calc::theta1

Definition at line 59 of file KatanaKinematics6M90T.h.

11.41.2.2 double KNI::KatanaKinematics6M90T::angles_calc::theta2

Definition at line 60 of file KatanaKinematics6M90T.h.

11.41.2.3 double KNI::KatanaKinematics6M90T::angles_calc::theta3

Definition at line 61 of file KatanaKinematics6M90T.h.

11.41.2.4 double KNI::KatanaKinematics6M90T::angles_calc::theta4

Definition at line 62 of file KatanaKinematics6M90T.h.

11.41.2.5 double KNI::KatanaKinematics6M90T::angles_calc::theta5

Definition at line 63 of file KatanaKinematics6M90T.h.

11.41.2.6 double KNI::KatanaKinematics6M90T::angles_calc::theta6

Definition at line 64 of file KatanaKinematics6M90T.h.

11.41.2.7 double KNI::KatanaKinematics6M90T::angles_calc::theta234

Definition at line 65 of file KatanaKinematics6M90T.h.

11.41.2.8 double KNI::KatanaKinematics6M90T::angles_calc::b1

Definition at line 66 of file KatanaKinematics6M90T.h.

11.41.2.9 double KNI::KatanaKinematics6M90T::angles_calc::b2

Definition at line 67 of file KatanaKinematics6M90T.h.

11.41.2.10 double KNI::KatanaKinematics6M90T::angles_calc::costh3

Definition at line 68 of file KatanaKinematics6M90T.h.

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

- include/KNI_InvKin/KatanaKinematics6M90T.h

11.42 KNI::KatanaKinematics6M90T::position Struct Reference

Public Attributes

- double x
- double y
- double z

11.42.1 Detailed Description

Definition at line 52 of file KatanaKinematics6M90T.h.

11.42.2 Member Data Documentation

11.42.2.1 double KNI::KatanaKinematics6M90T::position::x

Definition at line 53 of file KatanaKinematics6M90T.h.

11.42.2.2 double KNI::KatanaKinematics6M90T::position::y

Definition at line 54 of file KatanaKinematics6M90T.h.

11.42.2.3 double KNI::KatanaKinematics6M90T::position::z

Definition at line 55 of file KatanaKinematics6M90T.h.

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

- include/KNI_InvKin/KatanaKinematics6M90T.h

11.43 KNI::KinematicParameters Struct Reference

To pass different parameters for the kinematic implementations.

`#include <KatanaKinematics.h>`

Public Attributes

- double angleOffset
- double angleStop
- int epc
- int encOffset
- int rotDir

11.43.1 Detailed Description

To pass different parameters for the kinematic implementations.

These parameters are used for "reducing" different solutions to valid angles and to to check angles against given limits (angleOffset, angleStop)

Definition at line 53 of file KatanaKinematics.h.

11.43.2 Member Data Documentation

11.43.2.1 double KNI::KinematicParameters::angleOffset

Definition at line 54 of file KatanaKinematics.h.

11.43.2.2 double KNI::KinematicParameters::angleStop

Definition at line 55 of file KatanaKinematics.h.

11.43.2.3 int KNI::KinematicParameters::epc

Definition at line 56 of file KatanaKinematics.h.

11.43.2.4 int KNI::KinematicParameters::encOffset

Definition at line 57 of file KatanaKinematics.h.

11.43.2.5 int KNI::KinematicParameters::rotDir

Definition at line 58 of file KatanaKinematics.h.

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

- include/KNI_InvKin/KatanaKinematics.h

11.44 KNI::KinematicsDefaultEncMinAlgorithm Struct Reference

`#include <KatanaKinematicsDecisionAlgorithms.h>`

Public Types

- typedef std::vector< int > encoders
- typedef encoders::const_iterator c_iter
- typedef std::vector< encoders >::const_iterator t_iter

Public Member Functions

- t_iter operator() (t_iter targetEnc_begin, t_iter targetEnc_end, c_iter currentEnc_begin, c_iter currentEnc_end)

11.44.1 Detailed Description

Definition at line 30 of file KatanaKinematicsDecisionAlgorithms.h.

11.44.2 Member Typedef Documentation

11.44.2.1 typedef std::vector<int> KNI::KinematicsDefaultEncMinAlgorithm::encoders

Definition at line 31 of file KatanaKinematicsDecisionAlgorithms.h.

11.44.2.2 typedef encoders::const_iterator KNI::KinematicsDefaultEncMinAlgorithm::c_iter

Definition at line 32 of file KatanaKinematicsDecisionAlgorithms.h.

11.44.2.3 typedef std::vector< encoders >::const_iterator KNI::KinematicsDefaultEncMinAlgorithm::t_iter

Definition at line 33 of file KatanaKinematicsDecisionAlgorithms.h.

11.44.3 Member Function Documentation

11.44.3.1 t_iter KNI::KinematicsDefaultEncMinAlgorithm::operator() (t_iter *targetEnc_begin*, t_iter *targetEnc_end*, c_iter *currentEnc_begin*, c_iter *currentEnc_end*)

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

- include/KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h

11.45 KNI::kmlFactory Class Reference

This class is for internal use only It may change at any time It shields the configuration file parsing.

`#include <kmlFactories.h>`

Public Member Functions

- kmlFactory ()
- bool openFile (const char *filepath)
- TKatGNL getGNL ()
- TKatMOT getMOT ()
- TKatSCT getSCT ()
- TKatEFF getEFF ()
- TMotDesc * getMotDesc (short count)
- TSctDesc * getSctDesc (short count)
- TMotCLB getMotCLB (short number)
- TMotSCP getMotSCP (short number)
- TMotDYL getMotDYL (short number)
- int getType ()

  returns the Katana type

- int getKinematics ()

  returns the Kinematics to use

- TMotInit getMotInit (short number)
- void getGripperParameters (bool &isPresent, int &openEncoders, int &closeEncoders)

Private Member Functions

- void _readEntry (char *dest, int destsz, const char *section, const char *subsection, const char *entry)

Private Attributes

- std::ifstream _configfile

11.45.1 Detailed Description

This class is for internal use only It may change at any time It shields the configuration file parsing.

Definition at line 75 of file kmlFactories.h.

11.45.2 Constructor & Destructor Documentation

11.45.2.1 KNI::kmlFactory::kmlFactory ()

11.45.3 Member Function Documentation

11.45.3.1 void KNI::kmlFactory::_readEntry (char * *dest*, int *destsz*, const char * *section*, const char * *subsection*, const char * *entry*) [private]

11.45.3.2 bool KNI::kmlFactory::openFile (const char * *filepath*) [inline]

Definition at line 83 of file kmlFactories.h.

11.45.3.3 TKatGNL KNI::kmlFactory::getGNL ()

11.45.3.4 TKatMOT KNI::kmlFactory::getMOT ()

11.45.3.5 TKatSCT KNI::kmlFactory::getSCT ()

11.45.3.6 TKatEFF KNI::kmlFactory::getEFF ()

11.45.3.7 TMotDesc* KNI::kmlFactory::getMotDesc (short *count*)

11.45.3.8 TSctDesc* KNI::kmlFactory::getSctDesc (short *count*)

11.45.3.9 TMotCLB KNI::kmlFactory::getMotCLB (short *number*)

11.45.3.10 TMotSCP KNI::kmlFactory::getMotSCP (short *number*)

11.45.3.11 TMotDYL KNI::kmlFactory::getMotDYL (short *number*)

11.45.3.12 int KNI::kmlFactory::getType ()

returns the Katana type

Returns:

300 for Katana300, 400 for Katana400, 450 for Katana450

11.45.3.13 int KNI::kmlFactory::getKinematics ()

returns the Kinematics to use

Returns:

0 for Analytical, 1 for RobAnaGuess

11.45.3.14 TMotInit KNI::kmlFactory::getMotInit (short *number*)

11.45.3.15 void KNI::kmlFactory::getGripperParameters (bool & *isPresent*, int & *openEncoders*, int & *closeEncoders*)

11.45.4 Member Data Documentation

11.45.4.1 std::ifstream KNI::kmlFactory::_configfile [private]

Definition at line 77 of file kmlFactories.h.

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

- include/KNI/kmlFactories.h

11.46 MotorCrashException Class Reference

The requested motor crashed during the movement.

`#include <kmlCommon.h>`

Inheritance diagram for MotorCrashException: Collaboration diagram for MotorCrashException:

Public Member Functions

- MotorCrashException () throw ()

11.46.1 Detailed Description

The requested motor crashed during the movement.

Note:

error_number=-37

Definition at line 89 of file kmlCommon.h.

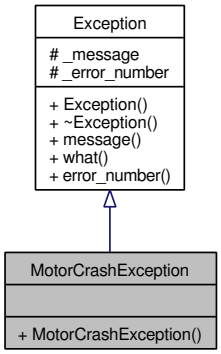
11.46.2 Constructor & Destructor Documentation

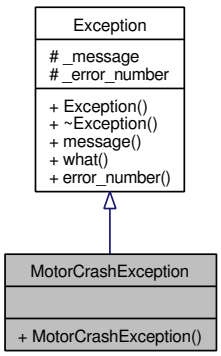
11.46.2.1 MotorCrashException::MotorCrashException () throw () [inline]

Definition at line 91 of file kmlCommon.h.

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

- include/KNI/kmlCommon.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
MotorCrashException
+ MotorCrashException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
MotorCrashException
+ MotorCrashException()

11.47 MotorOutOfRangeException Class Reference

The encoders for the given motor were out of range.

`#include <kmlCommon.h>`

Inheritance diagram for MotorOutOfRangeException: Collaboration diagram for MotorOutOfRangeException:

Public Member Functions

- MotorOutOfRangeException () throw ()

11.47.1 Detailed Description

The encoders for the given motor were out of range.

Note:

error_number=-35

Definition at line 71 of file kmlCommon.h.

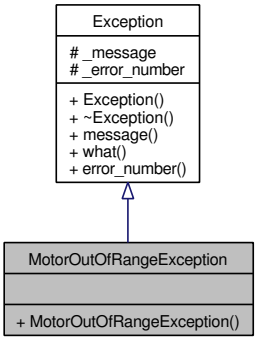
11.47.2 Constructor & Destructor Documentation

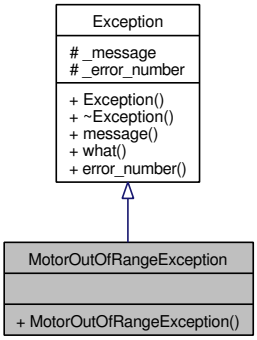
11.47.2.1 MotorOutOfRangeException::MotorOutOfRangeException () throw () [inline]

Definition at line 73 of file kmlCommon.h.

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

- include/KNI/kmlCommon.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
MotorOutOfRangeException
+ MotorOutOfRangeException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
MotorOutOfRangeException
+ MotorOutOfRangeException()

11.48 MotorTimeoutException Class Reference

The timeout elapsed for the given motor and target position.

`#include <kmlCommon.h>`

Inheritance diagram for MotorTimeoutException: Collaboration diagram for MotorTimeoutException:

Public Member Functions

- MotorTimeoutException () throw ()

11.48.1 Detailed Description

The timeout elapsed for the given motor and target position.

Note:

error_number=-36

Definition at line 80 of file kmlCommon.h.

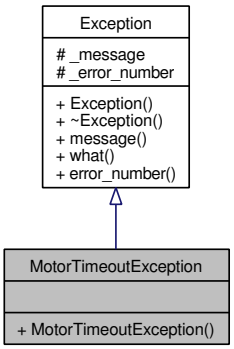
11.48.2 Constructor & Destructor Documentation

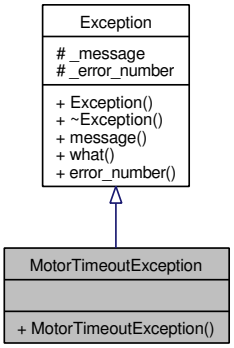
11.48.2.1 MotorTimeoutException::MotorTimeoutException () throw () [inline]

Definition at line 82 of file kmlCommon.h.

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

- include/KNI/kmlCommon.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
MotorTimeoutException
+ MotorTimeoutException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
MotorTimeoutException
+ MotorTimeoutException()

11.49 KNI::NoSolutionException Class Reference

No solution found for the given cartesian coordinates.

`#include <KatanaKinematics.h>`

Inheritance diagram for KNI::NoSolutionException: Collaboration diagram for KNI::NoSolutionException:

Public Member Functions

- NoSolutionException () throw ()

11.49.1 Detailed Description

No solution found for the given cartesian coordinates.

Note:

error_number=-60

Definition at line 39 of file KatanaKinematics.h.

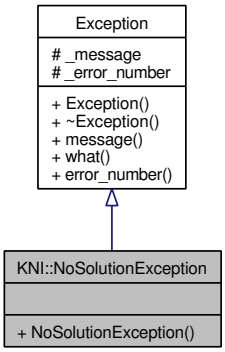
11.49.2 Constructor & Destructor Documentation

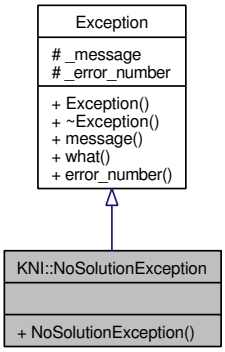
11.49.2.1 KNI::NoSolutionException::NoSolutionException () throw () [inline]

Definition at line 41 of file KatanaKinematics.h.

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

- include/KNI_InvKin/KatanaKinematics.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
KNI::NoSolutionException
+ NoSolutionException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
KNI::NoSolutionException
+ NoSolutionException()

11.50 ParameterReadingException Class Reference

There was an error while reading a parameter from the robot.

`#include <kmlCommon.h>`

Inheritance diagram for ParameterReadingException: Collaboration diagram for ParameterReadingException:

Public Member Functions

- ParameterReadingException (const std::string &para) throw ()

11.50.1 Detailed Description

There was an error while reading a parameter from the robot.

Note:

error_number=-32

Definition at line 44 of file kmlCommon.h.

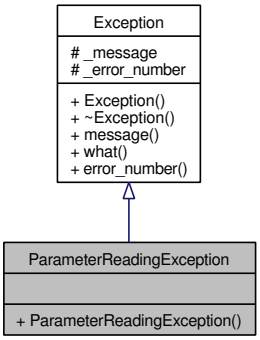
11.50.2 Constructor & Destructor Documentation

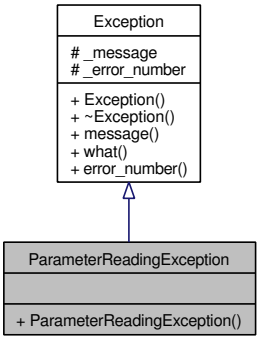
11.50.2.1 ParameterReadingException::ParameterReadingException (const std::string & *para*) throw () [inline]

Definition at line 46 of file kmlCommon.h.

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

- include/KNI/kmlCommon.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ParameterReadingException
+ ParameterReadingException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ParameterReadingException
+ ParameterReadingException()

11.51 ParameterWritingException Class Reference

The data you wanted to send to the robot was invalid.

`#include <kmlCommon.h>`

Inheritance diagram for ParameterWritingException: Collaboration diagram for ParameterWritingException:

Public Member Functions

- ParameterWritingException (const std::string &para) throw ()

11.51.1 Detailed Description

The data you wanted to send to the robot was invalid.

Note:

error_number=-33

Definition at line 53 of file kmlCommon.h.

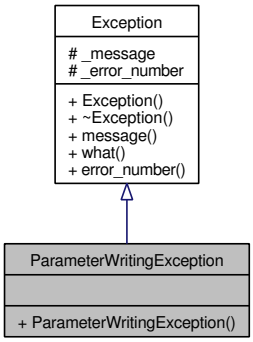
11.51.2 Constructor & Destructor Documentation

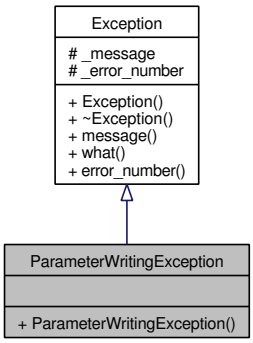
11.51.2.1 ParameterWritingException::ParameterWritingException (const std::string & *para*) throw () [inline]

Definition at line 55 of file kmlCommon.h.

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

- include/KNI/kmlCommon.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ParameterWritingException
+ ParameterWritingException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ParameterWritingException
+ ParameterWritingException()

11.52 PortNotOpenException Class Reference

The port was not open.

`#include <cdlCOMExceptions.h>`

Inheritance diagram for PortNotOpenException: Collaboration diagram for PortNotOpenException:

Public Member Functions

- PortNotOpenException (const std::string &port) throw ()

11.52.1 Detailed Description

The port was not open.

Note:

error_number=-12

Definition at line 65 of file cdlCOMExceptions.h.

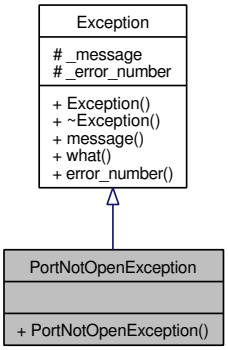
11.52.2 Constructor & Destructor Documentation

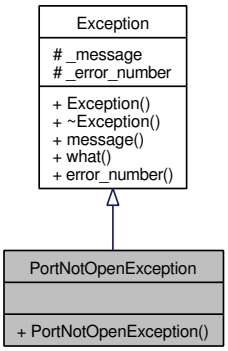
11.52.2.1 PortNotOpenException::PortNotOpenException (const std::string & *port*) throw () [inline]

Definition at line 67 of file cdlCOMExceptions.h.

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

- include/KNI/cdlCOMExceptions.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
PortNotOpenException
+ PortNotOpenException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
PortNotOpenException
+ PortNotOpenException()

11.53 ReadNotCompleteException Class Reference

The Katana didn't answer correctly within the given timeout.

`#include <cdlCOMExceptions.h>`

Inheritance diagram for ReadNotCompleteException: Collaboration diagram for ReadNotCompleteException:

Public Member Functions

- ReadNotCompleteException (const std::string &port) throw ()

11.53.1 Detailed Description

The Katana didn't answer correctly within the given timeout.

Note:

error_number=-16

Definition at line 112 of file cdlCOMExceptions.h.

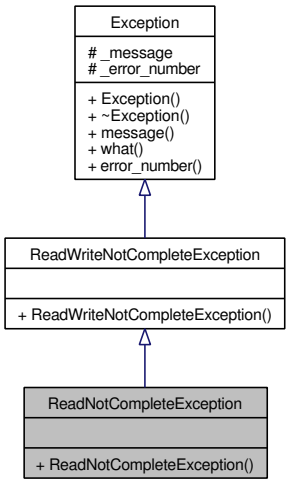
11.53.2 Constructor & Destructor Documentation

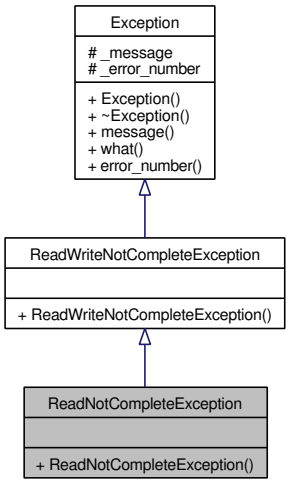
11.53.2.1 ReadNotCompleteException::ReadNotCompleteException (const std::string & *port*) throw () [inline]

Definition at line 114 of file cdlCOMExceptions.h.

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

- include/KNI/cdlCOMExceptions.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ReadWriteNotCompleteException
+ ReadWriteNotCompleteException()
ReadNotCompleteException
+ ReadNotCompleteException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ReadWriteNotCompleteException
+ ReadWriteNotCompleteException()
ReadNotCompleteException
+ ReadNotCompleteException()

11.54 ReadWriteNotCompleteException Class Reference

This exception is the base for the WriteNotComplete and ReadNotCompleteException.

`#include <cdlCOMExceptions.h>`

Inheritance diagram for ReadWriteNotCompleteException: Collaboration diagram for ReadWriteNotCompleteException:

Public Member Functions

- ReadWriteNotCompleteException (const std::string &errstr, const int error_number) throw ()

11.54.1 Detailed Description

This exception is the base for the WriteNotComplete and ReadNotCompleteException.

Definition at line 94 of file cdlCOMExceptions.h.

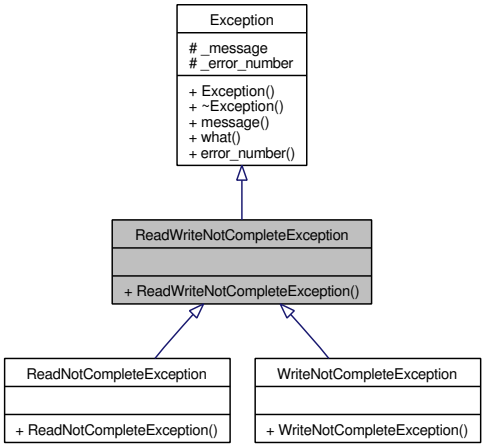
11.54.2 Constructor & Destructor Documentation

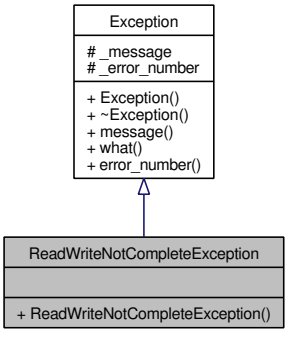
11.54.2.1 ReadWriteNotCompleteException::ReadWriteNotCompleteException (const std::string & *errstr*, const int *error_number*) throw () [inline]

Definition at line 96 of file cdlCOMExceptions.h.

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

- include/KNI/cdlCOMExceptions.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ReadWriteNotCompleteException
+ ReadWriteNotCompleteException()
ReadNotCompleteException
+ ReadNotCompleteException()
WriteNotCompleteException
+ WriteNotCompleteException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ReadWriteNotCompleteException
+ ReadWriteNotCompleteException()

11.55 SlaveErrorException Class Reference

Slave error occurred.

`#include <kmlCommon.h>`

Inheritance diagram for SlaveErrorException: Collaboration diagram for SlaveErrorException:

Public Member Functions

- SlaveErrorException () throw ()

11.55.1 Detailed Description

Slave error occurred.

Note:

error_number=-31

Definition at line 35 of file kmlCommon.h.

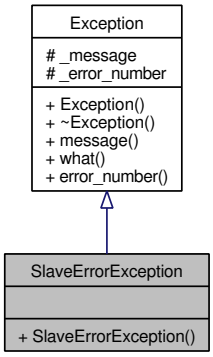
11.55.2 Constructor & Destructor Documentation

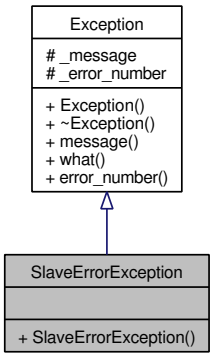
11.55.2.1 SlaveErrorException::SlaveErrorException () throw () [inline]

Definition at line 37 of file kmlCommon.h.

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

- include/KNI/kmlCommon.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
SlaveErrorException
+ SlaveErrorException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
SlaveErrorException
+ SlaveErrorException()

11.56 TCdlCOMDesc Struct Reference

This structrue stores the attributes for a serial port device.

`#include <cdlCOM.h>`

Public Attributes

- int port
  serial port number
- int baud
  baud rate of port
- int data
  data bit
- int parity
  parity bit
- int stop
  stop bit
- int rttc
  read total timeout
- int wttc
  write total timeout

11.56.1 Detailed Description

This structrue stores the attributes for a serial port device.

Definition at line 53 of file cdlCOM.h.

11.56.2 Member Data Documentation

11.56.2.1 int TCdlCOMDesc::port

serial port number

Definition at line 54 of file cdlCOM.h.

11.56.2.2 int TCdlCOMDesc::baud

baud rate of port

Definition at line 55 of file cdlCOM.h.

11.56.2.3 int TCdlCOMDesc::data

data bit

Definition at line 56 of file cdlCOM.h.

11.56.2.4 int TCdlCOMDesc::parity

parity bit

Definition at line 57 of file cdlCOM.h.

11.56.2.5 int TCdlCOMDesc::stop

stop bit

Definition at line 58 of file cdlCOM.h.

11.56.2.6 int TCdlCOMDesc::rttc

read total timeout

Definition at line 59 of file cdlCOM.h.

11.56.2.7 int TCdlCOMDesc::wttc

write total timeout

Definition at line 60 of file cdlCOM.h.

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

- include/KNI/cdlCOM.h

11.57 TCurrentMot Struct Reference

structure for the currently active axis

`#include <kni_wrapper.h>`

Public Attributes

- int idx
- bool running
- bool dir

11.57.1 Detailed Description

structure for the currently active axis

Definition at line 74 of file kni_wrapper.h.

11.57.2 Member Data Documentation

11.57.2.1 int TCurrentMot::idx

Definition at line 75 of file kni_wrapper.h.

11.57.2.2 bool TCurrentMot::running

Definition at line 76 of file kni_wrapper.h.

11.57.2.3 bool TCurrentMot::dir

Definition at line 77 of file kni_wrapper.h.

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

- include/kni_wrapper/kni_wrapper.h

11.58 THeader Struct Reference

Header of a communication packet.

`#include <cplSerial.h>`

Public Attributes

- byte size
  header size
- byte data [256]
  data part: 16x zero, 1x one, 1x katadr

11.58.1 Detailed Description

Header of a communication packet.

Definition at line 75 of file cplSerial.h.

11.58.2 Member Data Documentation

11.58.2.1 byte THeader::size

header size

Definition at line 76 of file cplSerial.h.

11.58.2.2 byte THeader::data[256]

data part: 16x zero, 1x one, 1x katadr

Definition at line 77 of file cplSerial.h.

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

- include/KNI/cplSerial.h

11.59 KNI::Timer Class Reference

Provides a stop-watch-like class with a resolution of milliseconds.

`#include <Timer.h>`

Public Member Functions

- Timer ()
- Timer (long timeout)
- void Set (long timeout)
- void Start ()
- void Set_And_Start (long timeout)
- bool Elapsed () const

  Returns true if timer is elapsed.

- long ElapsedTime () const

  Returns the elapsed time.

- void WaitUntilElapsed () const

  Block until time's up.

Private Member Functions

- long _ElapsedTime () const

  Platform specific implementation of ElapsedTime().

Private Attributes

- long _timeout
- struct timeval _ct

11.59.1 Detailed Description

Provides a stop-watch-like class with a resolution of milliseconds.

Definition at line 41 of file Timer.h.

11.59.2 Constructor & Destructor Documentation

11.59.2.1 KNI::Timer::Timer ()

11.59.2.2 KNI::Timer::Timer (long *timeout*)

11.59.3 Member Function Documentation

11.59.3.1 long KNI::Timer::_ElapsedTime () const [private]

Platform specific implementation of ElapsedTime().

11.59.3.2 void KNI::Timer::Set (long *timeout*)

11.59.3.3 void KNI::Timer::Start ()

11.59.3.4 void KNI::Timer::Set_And_Start (long *timeout*)

11.59.3.5 bool KNI::Timer::Elapsed () const

Returns true if timer is elapsed.

11.59.3.6 long KNI::Timer::ElapsedTime () const

Returns the elapsed time.

11.59.3.7 void KNI::Timer::WaitUntilElapsed () const

Block until time's up.

11.59.4 Member Data Documentation

11.59.4.1 long KNI::Timer::_timeout [private]

Definition at line 43 of file Timer.h.

11.59.4.2 struct timeval KNI::Timer::_ct [read, private]

Definition at line 48 of file Timer.h.

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

- include/common/Timer.h

11.60 TKatCBX Struct Reference

[CBX] connector box

`#include <kmlBase.h>`

Public Attributes

- bool inp [2]
  input: green & red LED
- bool out [2]
  output: green & red LED

11.60.1 Detailed Description

[CBX] connector box

Definition at line 93 of file kmlBase.h.

11.60.2 Member Data Documentation

11.60.2.1 bool TKatCBX::inp[2]

input: green & red LED

Definition at line 94 of file kmlBase.h.

11.60.2.2 bool TKatCBX::out[2]

output: green & red LED

Definition at line 95 of file kmlBase.h.

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

- include/KNI/kmlBase.h

11.61 TKatCTB Struct Reference

[CTB] command table defined in the firmware

`#include <kmlBase.h>`

Public Attributes

- byte cmdtbl [256]
  command table

11.61.1 Detailed Description

[CTB] command table defined in the firmware

Definition at line 87 of file kmlBase.h.

11.61.2 Member Data Documentation

11.61.2.1 byte TKatCTB::cmdtbl[256]

command table

Definition at line 88 of file kmlBase.h.

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

- include/KNI/kmlBase.h

11.62 TKatECH Struct Reference

[ECH] echo

`#include <kmlBase.h>`

Public Attributes

- byte echo

  echo answer

11.62.1 Detailed Description

[ECH] echo

Definition at line 100 of file kmlBase.h.

11.62.2 Member Data Documentation

11.62.2.1 byte TKatECH::echo

echo answer

Definition at line 101 of file kmlBase.h.

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

- include/KNI/kmlBase.h

11.63 TKatEFF Struct Reference

Inverse Kinematics structure of the endeffektor.

`#include <kmlBase.h>`

Public Attributes

- double arr_segment [4]

  length of the Katana segments

11.63.1 Detailed Description

Inverse Kinematics structure of the endeffektor.

This structure describes the properties of the endeffector and it's used for the inverse kinematic calculations. An endeffector is a point where the attributes of this structure belong to. Please remember that the actual inverse kinematic calculations have been set up **only** for the Katana **6M** robot! So do not be astonished if you get strange behaviour with a Katana **5M**.

Definition at line 113 of file kmlBase.h.

11.63.2 Member Data Documentation

11.63.2.1 double TKatEFF::arr_segment[4]

length of the Katana segments

Definition at line 114 of file kmlBase.h.

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

- include/KNI/kmlBase.h

11.64 TKatGNL Struct Reference

[GNL] general robot attributes

`#include <kmlBase.h>`

Public Attributes

- byte adr
  jumper adress
- char modelName [255]
  model name

11.64.1 Detailed Description

[GNL] general robot attributes

Definition at line 67 of file kmlBase.h.

11.64.2 Member Data Documentation

11.64.2.1 byte TKatGNL::adr

jumper adress

Definition at line 68 of file kmlBase.h.

11.64.2.2 char TKatGNL::modelName[255]

model name

Definition at line 69 of file kmlBase.h.

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

- include/KNI/kmlBase.h

11.65 TKatIDS Struct Reference

[IDS] identification string

`#include <kmlBase.h>`

Public Attributes

- byte strID [256]
  id string

11.65.1 Detailed Description

[IDS] identification string

Definition at line 81 of file kmlBase.h.

11.65.2 Member Data Documentation

11.65.2.1 byte TKatIDS::strID[256]

id string

Definition at line 82 of file kmlBase.h.

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

- include/KNI/kmlBase.h

11.66 TKatMFW Struct Reference

[MFW] master firmware version/revision number

`#include <kmlBase.h>`

Public Attributes

- byte ver
  version
- byte rev
  revision

11.66.1 Detailed Description

[MFW] master firmware version/revision number

Definition at line 74 of file kmlBase.h.

11.66.2 Member Data Documentation

11.66.2.1 byte TKatMFW::ver

version

Definition at line 75 of file kmlBase.h.

11.66.2.2 byte TKatMFW::rev

revision

Definition at line 76 of file kmlBase.h.

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

- include/KNI/kmlBase.h

11.67 TKatMOT Struct Reference

[MOT] every motor's attributes

`#include <kmlMotBase.h>`

Collaboration diagram for TKatMOT:

Public Attributes

- short cnt
  count of motors
- CMotBase * arr
  array of motors
- TMotDesc * desc
  description[]

11.67.1 Detailed Description

[MOT] every motor's attributes

Definition at line 40 of file kmlMotBase.h.

11.67.2 Member Data Documentation

11.67.2.1 short TKatMOT::cnt

count of motors

Definition at line 41 of file kmlMotBase.h.

11.67.2.2 CMotBase* TKatMOT::arr

array of motors

Definition at line 42 of file kmlMotBase.h.

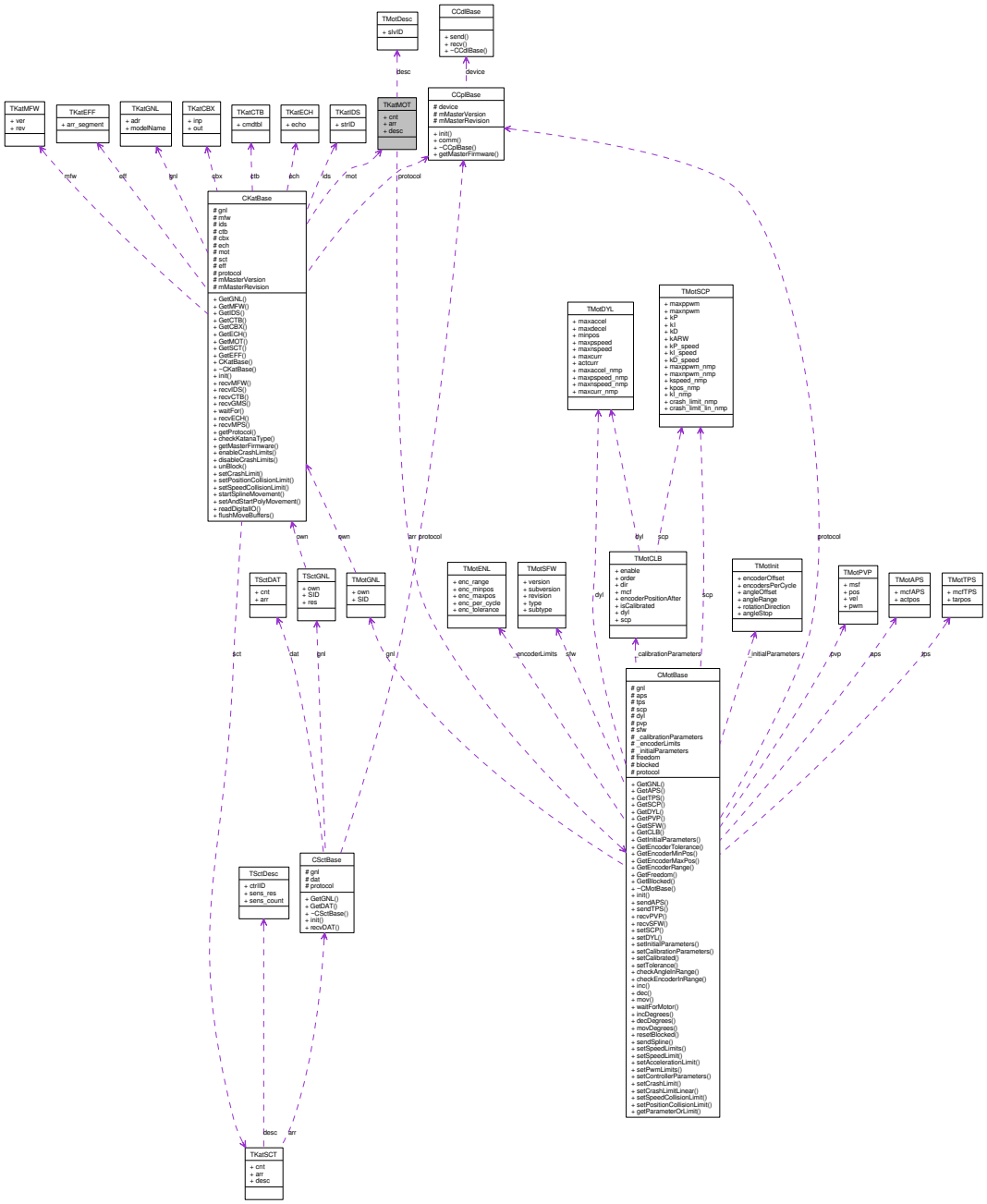
11.67.2.3 TMotDesc* TKatMOT::desc

description[]

Definition at line 43 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h


TMotDesc
+ slvID
CCdtBase
+ send()
+ recv()
+ ~CCdtBase()
idesc
device
TKatMFW
+ ver
+ rev
TKatEFF
+ arr_segment
TKatGNL
+ adr
+ modelName
TKatCBX
+ inp
+ out
TKatCTB
+ cmdtbl
TKatECH
+ echo
TKatIDS
+ strID
TKatMOT
+ cnt
+ arr
+ desc
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
mfw
eff
gnl
cbx
ctb
ech
ids
mot
protocol
CKatBase
gnl
mfw
ids
ctb
cbx
ech
mot
sct
eff
protocol
mMasterVersion
mMasterRevision
+ GetGNL()
+ GetMFW()
+ GetIDS()
+ GetCTB()
+ GetCBX()
+ GetECH()
+ GetMOT()
+ GetSCT()
+ GetEFF()
+ CKatBase()
+ ~CKatBase()
+ init()
+ recvMFW()
+ recvIDS()
+ recvCTB()
+ recvGMS()
+ waitFor()
+ recvECH()
+ recvMPS()
+ getProtocol()
+ checkKatanaType()
+ getMasterFirmware()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ startSplineMovement()
+ setAndStartPolyMovement()
+ readDigitalIO()
+ flushMoveBuffers()
own
own
arr
protocol
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxnspeed
+ maxcurr
+ actcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxnspeed_nmp
+ maxcurr_nmp
TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kAPW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ kI_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
dyl
scp
protocol
TSctDAT
+ cnt
+ arr
TSctGNL
+ own
+ SID
+ res
TMotGNL
+ own
+ SID
TMotENL
+ enc_range
+ enc_minpos
+ enc_maxpos
+ enc_per_cycle
+ enc_tolerance
TMotSFW
+ version
+ subversion
+ revision
+ type
+ subtype
TMotCLB
+ enable
+ order
+ dir
+ mcf
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp
TMotInit
+ encoderOffset
+ encodersPerCycle
+ angleOffset
+ angleRange
+ rotationDirection
+ angleStop
TMotPVP
+ msf
+ pos
+ vel
+ pwm
TMotAPS
+ mctAPS
+ actpos
TMotTPS
+ mctTPS
+ tarpos
dyl
scp
encoderLimits
sfw
calibrationParameters
initialParameters
pvp
aps
tps
sct
dat
gnl
gnl
CMotBase
gnl
aps
tps
scp
dyl
pvp
sfw
_calibrationParameters
_encoderLimits
_initialParameters
freedom
blocked
protocol
+ GetGNL()
+ GetAPS()
+ GetTPS()
+ GetSCP()
+ GetDYL()
+ GetPVP()
+ GetSFW()
+ GetCLB()
+ GetInitialParameters()
+ GetEncoderTolerance()
+ GetEncoderMinPos()
+ GetEncoderMaxPos()
+ GetEncoderRange()
+ GetFreedom()
+ GetBlocked()
+ ~CMotBase()
+ init()
+ sendAPS()
+ sendTPS()
+ recvPVP()
+ recvSFW()
+ setSCP()
+ setDYL()
+ setInitialParameters()
+ setCalibrationParameters()
+ setCalibrated()
+ setTolerance()
+ checkAngleInRange()
+ checkEncoderInRange()
+ inc()
+ dec()
+ mov()
+ waitForMotor()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ resetBlocked()
+ sendSpline()
+ setSpeedLimits()
+ setSpeedLimit()
+ setAccelerationLimit()
+ setPwmLimits()
+ setControllerParameters()
+ setCrashLimit()
+ setCrashLimitLinear()
+ setSpeedCollisionLimit()
+ setPositionCollisionLimit()
+ getParameterOrLimit()
TSctDesc
+ ctrlID
+ sens_res
+ sens_count
CSctBase
gnl
dat
protocol
+ GetGNL()
+ GetDAT()
+ ~CSctBase()
+ init()
+ recvDAT()
idesc
arr
TKatSCT
+ cnt
+ arr
+ desc

11.68 TKatSCT Struct Reference

[SCT] every sens ctrl's attributes

`#include <kmlSctBase.h>`

Collaboration diagram for TKatSCT:

Public Attributes

- short cnt
  count of sens ctrl's
- CSctBase * arr
  array of sens ctrl's
- TSctDesc * desc
  description[]

11.68.1 Detailed Description

[SCT] every sens ctrl's attributes

Definition at line 41 of file kmlSctBase.h.

11.68.2 Member Data Documentation

11.68.2.1 short TKatSCT::cnt

count of sens ctrl's

Definition at line 42 of file kmlSctBase.h.

11.68.2.2 CSctBase* TKatSCT::arr

array of sens ctrl's

Definition at line 43 of file kmlSctBase.h.

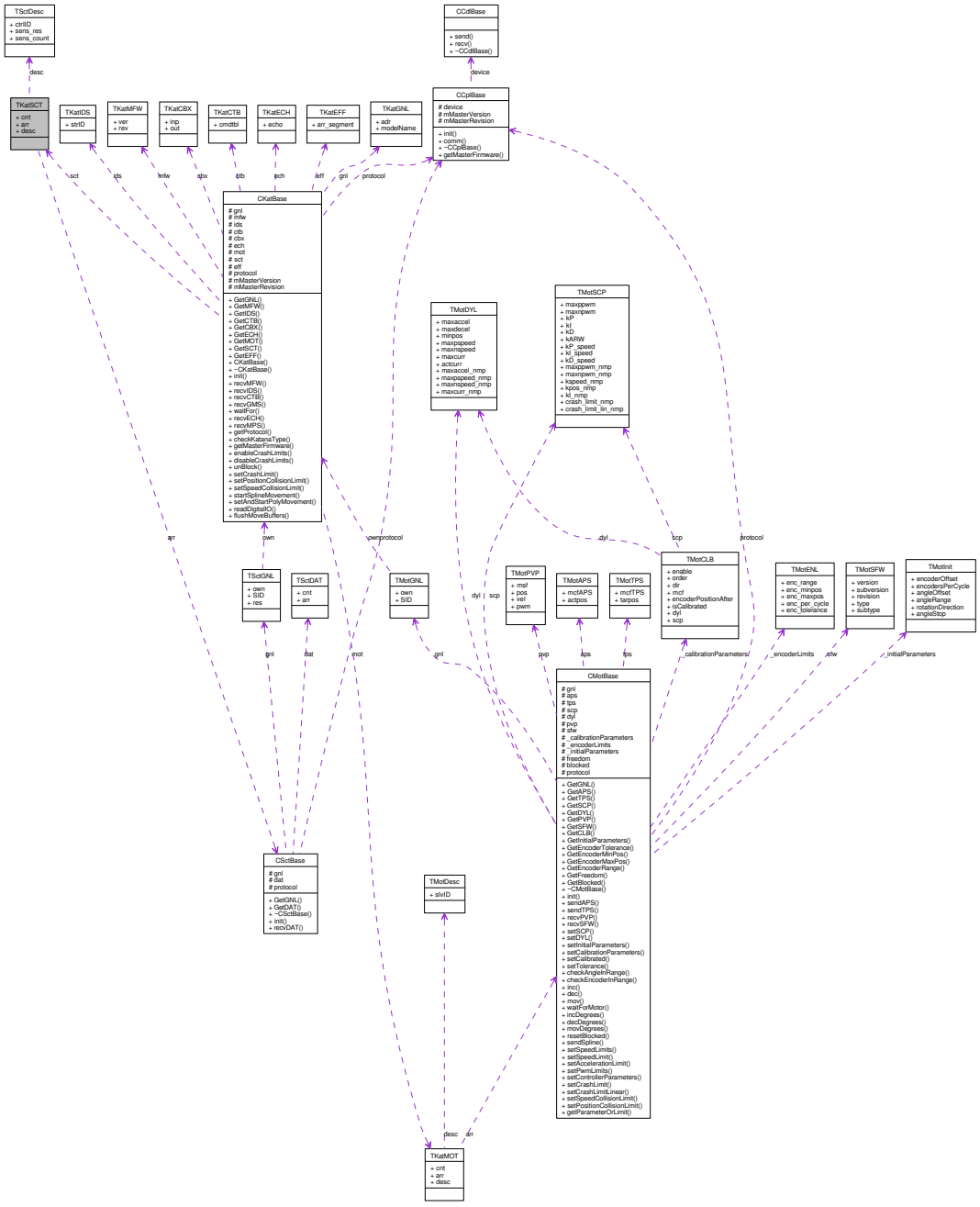
11.68.2.3 TSctDesc* TKatSCT::desc

description[]

Definition at line 44 of file kmlSctBase.h.

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

- include/KNI/kmlSctBase.h


TSctDesc
+ ctrlID
+ sens_res
+ sens_count
desc
CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
TKatSCT
+ cnt
+ arr
+ desc
TKatIDS
+ strID
TKatMFW
+ ver
+ rev
TKatCBX
+ inp
+ out
TKatCTB
+ cmdtbl
TKatECH
+ echo
TKatEFF
+ arr_segment
TKatGNL
+ adr
+ modelName
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
sct
ids
mfw
cbx
ctb
ech
eff
gnl
protocol
CKatBase
gnl
mfw
ids
ctb
cbx
ech
mot
sct
eff
protocol
mMasterVersion
mMasterRevision
+ GetGNL()
+ GetMFW()
+ GetIDS()
+ GetCTB()
+ GetCBX()
+ GetECH()
+ GetMOT()
+ GetSCT()
+ GetEFF()
+ CKatBase()
+ ~CKatBase()
+ init()
+ recvMFW()
+ recvIDS()
+ recvCTB()
+ recvGMS()
+ waitFor()
+ recvECH()
+ recvMPS()
+ getProtocol()
+ checkKatanaType()
+ getMasterFirmware()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ startSplineMovement()
+ setAndStartPolyMovement()
+ readDigitalIO()
+ flushMoveBuffers()
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxnspeed
+ maxcurr
+ actcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxnspeed_nmp
+ maxcurr_nmp
TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kARW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ kI_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
arr
own
ownprotocol
dyl
scp
protocol
TSctGNL
+ own
+ SID
+ res
TSctDAT
+ cnt
+ arr
TMotGNL
+ own
+ SID
dyl
scp
TMotPVP
+ msf
+ pos
+ vel
+ pwm
TMotAPS
+ mctAPS
+ actpos
TMotTPS
+ mcfTPS
+ tarpos
TMotCLB
+ enable
+ order
+ dir
+ mcr
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp
TMotENL
+ enc_range
+ enc_minpos
+ enc_maxpos
+ enc_per_cycle
+ enc_tolerance
TMotSFW
+ version
+ subversion
+ revision
+ type
+ subtype
TMotInit
+ encoderOffset
+ encodersPerCycle
+ angleOffset
+ angleRange
+ rotationDirection
+ angleStop
gnl
dat
mot
gnl
pvp
aps
tps
calibrationParameters
encoderLimits
sfw
initialParameters
CMotBase
gnl
aps
tps
scp
dyl
pvp
sfw
_calibrationParameters
_encoderLimits
_initialParameters
freedom
blocked
protocol
+ GetGNL()
+ GetAPS()
+ GetTPS()
+ GetSCP()
+ GetDYL()
+ GetPVP()
+ GetSFW()
+ GetCLB()
+ GetInitialParameters()
+ GetEncoderTolerance()
+ GetEncoderMinPos()
+ GetEncoderMaxPos()
+ GetEncoderRange()
+ GetFreedom()
+ GetBlocked()
+ ~CMotBase()
+ init()
+ sendAPS()
+ sendTPS()
+ recvPVP()
+ recvSFW()
+ setSCP()
+ setDYL()
+ setInitialParameters()
+ setCalibrationParameters()
+ setCalibrated()
+ setTolerance()
+ checkAngleInRange()
+ checkEncoderInRange()
+ inc()
+ dec()
+ mov()
+ waitForMotor()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ resetBlocked()
+ sendSpline()
+ setSpeedLimits()
+ setSpeedLimit()
+ setAccelerationLimit()
+ setPwmLimits()
+ setControllerParameters()
+ setCrashLimit()
+ setCrashLimitLinear()
+ setSpeedCollisionLimit()
+ setPositionCollisionLimit()
+ getParameterOrLimit()
CSctBase
gnl
dat
protocol
+ GetGNL()
+ GetDAT()
+ ~CSctBase()
+ init()
+ recvDAT()
TMotDesc
+ slvID
desc
arr
TKatMOT
+ cnt
+ arr
+ desc

11.69 TMotAPS Struct Reference

[APS] actual position

`#include <kmlMotBase.h>`

Public Attributes

- TMotCmdFlg mcfAPS
  motor command flag
- short actpos
  actual position

11.69.1 Detailed Description

[APS] actual position

Definition at line 96 of file kmlMotBase.h.

11.69.2 Member Data Documentation

11.69.2.1 TMotCmdFlg TMotAPS::mcfAPS

motor command flag

Definition at line 97 of file kmlMotBase.h.

11.69.2.2 short TMotAPS::actpos

actual position

Definition at line 98 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h

11.70 TMotCLB Struct Reference

Calibration structure for single motors.

`#include <kmlMotBase.h>`

Collaboration diagram for TMotCLB:

Public Attributes

- bool enable
  enable/disable
- short order
  order in which this motor will be calibrated. range: 0..5
- TSearchDir dir
  search direction for mech. stopper
- TMotCmdFlg mcf
  motor flag after calibration
- int encoderPositionAfter
- bool isCalibrated
- TMotDYL dyl
- TMotSCP scp

11.70.1 Detailed Description

Calibration structure for single motors.

Definition at line 182 of file kmlMotBase.h.

11.70.2 Member Data Documentation

11.70.2.1 bool TMotCLB::enable

enable/disable

Definition at line 183 of file kmlMotBase.h.

11.70.2.2 short TMotCLB::order

order in which this motor will be calibrated. range: 0..5

Definition at line 184 of file kmlMotBase.h.

11.70.2.3 TSearchDir TMotCLB::dir

search direction for mech. stopper

Definition at line 186 of file kmlMotBase.h.

11.70.2.4 TMotCmdFlg TMotCLB::mcf

motor flag after calibration

Definition at line 187 of file kmlMotBase.h.

11.70.2.5 int TMotCLB::encoderPositionAfter

Definition at line 189 of file kmlMotBase.h.

11.70.2.6 bool TMotCLB::isCalibrated

Definition at line 190 of file kmlMotBase.h.

11.70.2.7 TMotDYL TMotCLB::dyl

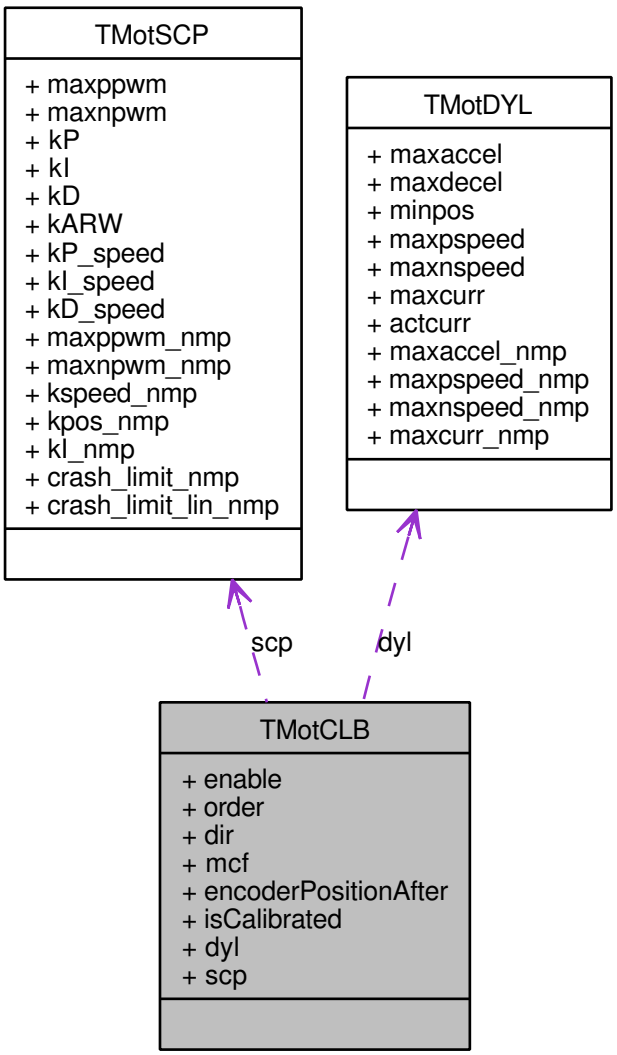
Definition at line 192 of file kmlMotBase.h.

11.70.2.8 TMotSCP TMotCLB::scp

Definition at line 193 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h


TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kARW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ kI_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxnspeed
+ maxcurr
+ actcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxnspeed_nmp
+ maxcurr_nmp
scp
dyl
TMotCLB
+ enable
+ order
+ dir
+ mcf
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp

11.71 TMotDesc Struct Reference

motor description (partly)

`#include <kmlMotBase.h>`

Public Attributes

- byte slvID
  slave number

11.71.1 Detailed Description

motor description (partly)

Definition at line 34 of file kmlMotBase.h.

11.71.2 Member Data Documentation

11.71.2.1 byte TMotDesc::slvID

slave number

Definition at line 35 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h

11.72 TMotDYL Struct Reference

[DYL] dynamic limits

`#include <kmlMotBase.h>`

Public Attributes

- byte maxaccel
  max acceleration
- byte maxdecel
  max deceleration
- short minpos
  not yet active
- short maxpspeed
  max. allowed forward speed
- short maxnspeed
  max. allowed reverse speed; pos!
- byte maxcurr
  max current
- byte actcurr
  actual current
- byte maxaccel_nmp
  Maximal acceleration and deceleration.
- short maxpspeed_nmp
  Max. allowed forward speed.
- short maxnspeed_nmp
  Max. allowed reverse speed.
- byte maxcurr_nmp
  set the maximal current

11.72.1 Detailed Description

[DYL] dynamic limits

Definition at line 138 of file kmlMotBase.h.

11.72.2 Member Data Documentation

11.72.2.1 byte TMotDYL::maxaccel

max acceleration

Definition at line 142 of file kmlMotBase.h.

11.72.2.2 byte TMotDYL::maxdecel

max deceleration

Definition at line 143 of file kmlMotBase.h.

11.72.2.3 short TMotDYL::minpos

not yet active

Definition at line 144 of file kmlMotBase.h.

11.72.2.4 short TMotDYL::maxpspeed

max. allowed forward speed

Definition at line 145 of file kmlMotBase.h.

11.72.2.5 short TMotDYL::maxnspeed

max. allowed reverse speed; pos!

Definition at line 146 of file kmlMotBase.h.

11.72.2.6 byte TMotDYL::maxcurr

max current

Definition at line 149 of file kmlMotBase.h.

11.72.2.7 byte TMotDYL::actcurr

actual current

Definition at line 150 of file kmlMotBase.h.

11.72.2.8 byte TMotDYL::maxaccel_nmp

Maximal acceleration and deceleration.

Definition at line 154 of file kmlMotBase.h.

11.72.2.9 short TMotDYL::maxpspeed_nmp

Max. allowed forward speed.

Definition at line 155 of file kmlMotBase.h.

11.72.2.10 short TMotDYL::maxnspeed_nmp

Max. allowed reverse speed.

Definition at line 156 of file kmlMotBase.h.

11.72.2.11 byte TMotDYL::maxcurr_nmp

set the maximal current

Definition at line 157 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h

11.73 TMotENL Struct Reference

[ENL] limits in encoder values (INTERNAL STRUCTURE!)

`#include <kmlMotBase.h>`

Public Attributes

- int enc_range
  motor's range in encoder values
- int enc_minpos
  motor's minimum position in encoder values
- int enc_maxpos
  motor's maximum position in encoder values
- int enc_per_cycle
  number of encoder units needed to complete 360 degrees;
- int enc_tolerance
  encoder units of tolerance to accept that a position has been reached

11.73.1 Detailed Description

[ENL] limits in encoder values (INTERNAL STRUCTURE!)

Definition at line 171 of file kmlMotBase.h.

11.73.2 Member Data Documentation

11.73.2.1 int TMotENL::enc_range

motor's range in encoder values

Definition at line 172 of file kmlMotBase.h.

11.73.2.2 int TMotENL::enc_minpos

motor's minimum position in encoder values

Definition at line 173 of file kmlMotBase.h.

11.73.2.3 int TMotENL::enc_maxpos

motor's maximum position in encoder values

Definition at line 174 of file kmlMotBase.h.

11.73.2.4 int TMotENL::enc_per_cycle

number of encoder units needed to complete 360 degrees;

Definition at line 175 of file kmlMotBase.h.

11.73.2.5 int TMotENL::enc_tolerance

encoder units of tolerance to accept that a position has been reached

Definition at line 176 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h

11.74 TMotGNL Struct Reference

[GNL] motor generals

`#include <kmlMotBase.h>`

Collaboration diagram for TMotGNL:

Public Attributes

- CKatBase * own
  parent robot
- byte SID
  slave ID

11.74.1 Detailed Description

[GNL] motor generals

Definition at line 79 of file kmlMotBase.h.

11.74.2 Member Data Documentation

11.74.2.1 CKatBase* TMotGNL::own

parent robot

Definition at line 80 of file kmlMotBase.h.

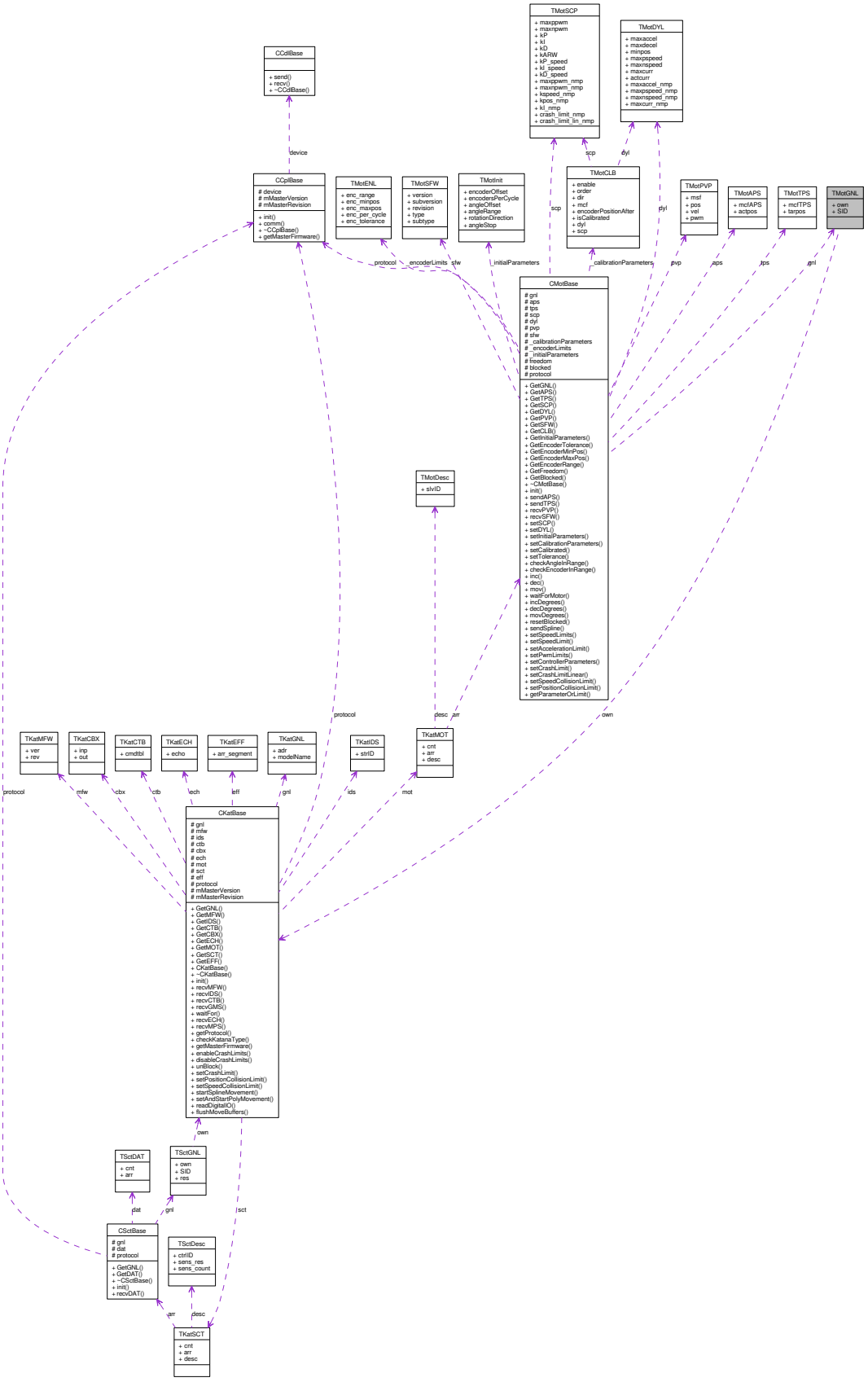
11.74.2.2 byte TMotGNL::SID

slave ID

Definition at line 81 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kARW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ kI_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxnspeed
+ maxcurr
+ actcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxnspeed_nmp
+ maxcurr_nmp
scp
dyl
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
TMotENL
+ enc_range
+ enc_minpos
+ enc_maxpos
+ enc_per_cycle
+ enc_tolerance
TMotSFW
+ version
+ subversion
+ revision
+ type
+ subtype
TMotInit
+ encoderOffset
+ encodersPerCycle
+ angleOffset
+ angleRange
+ rotationDirection
+ angleStop
TMotCLB
+ enable
+ order
+ dir
+ mcf
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp
TMotPVP
+ msf
+ pos
+ vel
+ pwm
TMotAPS
+ mcfAPS
+ actpos
TMotTPS
+ mcfTPS
+ tarpos
TMotGNL
+ own
+ SID
protocol
encoderLimits
shw
initialParameters
calibrationParameters
pvp
aps
tps
gnl
CMotBase
gnl
aps
tps
scp
dyl
pvp
shw
_calibrationParameters
_encoderLimits
_initialParameters
freedom
blocked
protocol
+ GetGNL()
+ GetAPS()
+ GetTPS()
+ GetSCP()
+ GetDYL()
+ GetPVP()
+ GetSFW()
+ GetCLB()
+ GetInitialParameters()
+ GetEncoderTolerance()
+ GetEncoderMinPos()
+ GetEncoderMaxPos()
+ GetEncoderRange()
+ GetFreedom()
+ GetBlocked()
+ ~CMotBase()
+ init()
+ sendAPS()
+ sendTPS()
+ recvPVP()
+ recvSFW()
+ setSCP()
+ setDYL()
+ setInitialParameters()
+ setCalibrationParameters()
+ setCalibrated()
+ setTolerance()
+ checkAngleInRange()
+ checkEncoderInRange()
+ inc()
+ dec()
+ movj()
+ waitForMotor()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ resetBlocked()
+ sendSpline()
+ setSpeedLimits()
+ setSpeedLimit()
+ setAccelerationLimit()
+ setPwmLimits()
+ setControllerParameters()
+ setCrashLimit()
+ setCrashLimitLinear()
+ setSpeedCollisionLimit()
+ setPositionCollisionLimit()
+ getParameterOrLimit()
TMotDesc
+ slvID
desc
arr
own
protocol
TKatMFW
+ ver
+ rev
TKatCBX
+ inp
+ out
TKatCTB
+ cmdtbl
TKatECH
+ echo
TKatEFF
+ arr_segment
TKatGNL
+ adr
+ modelName
TKatIDS
+ strID
TKatMOT
+ cnt
+ arr
+ desc
mfw
cbx
ctb
ech
eff
gnl
ids
mot
CKatBase
gnl
mfw
ids
ctb
cbx
ech
mot
sct
eff
protocol
mMasterVersion
mMasterRevision
+ GetGNL()
+ GetMFW()
+ GetIDS()
+ GetCTB()
+ GetCBX()
+ GetECH()
+ GetMOT()
+ GetSCT()
+ GetEFF()
+ CKatBase()
+ ~CKatBase()
+ init()
+ recvMFW()
+ recvIDS()
+ recvCTB()
+ recvGMS()
+ waitFor()
+ recvECH()
+ recvMPS()
+ getProtocol()
+ checkKatanaType()
+ getMasterFirmware()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ startSplineMovement()
+ setAndStartPolyMovement()
+ readDigitalIO()
+ flushMoveBuffers()
TSctDAT
+ cnt
+ arr
TSctGNL
+ own
+ SID
+ res
dat
sct
CSctBase
gnl
dat
protocol
+ GetGNL()
+ GetDAT()
+ ~CSctBase()
+ init()
+ recvDAT()
TSctDesc
+ ctrlID
+ sens_res
+ sens_count
TKatSCT
+ cnt
+ arr
+ desc

11.75 TMotInit Struct Reference

Initial motor parameters.

`#include <kmlMotBase.h>`

Public Attributes

- int encoderOffset
- int encodersPerCycle
- double angleOffset
- double angleRange
- int rotationDirection
- double angleStop

11.75.1 Detailed Description

Initial motor parameters.

Definition at line 199 of file kmlMotBase.h.

11.75.2 Member Data Documentation

11.75.2.1 int TMotInit::encoderOffset

Definition at line 200 of file kmlMotBase.h.

11.75.2.2 int TMotInit::encodersPerCycle

Definition at line 201 of file kmlMotBase.h.

11.75.2.3 double TMotInit::angleOffset

Definition at line 202 of file kmlMotBase.h.

11.75.2.4 double TMotInit::angleRange

Definition at line 203 of file kmlMotBase.h.

11.75.2.5 int TMotInit::rotationDirection

Definition at line 204 of file kmlMotBase.h.

11.75.2.6 double TMotInit::angleStop

Definition at line 207 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h

11.76 TMotPVP Struct Reference

[PVP] position, velocity, pulse width modulation

`#include <kmlMotBase.h>`

Public Attributes

- TMotStsFlg msf
  motor status flag
- short pos
  position
- short vel
  velocity
- byte pwm
  pulse with modulation

11.76.1 Detailed Description

[PVP] position, velocity, pulse width modulation

Definition at line 162 of file kmlMotBase.h.

11.76.2 Member Data Documentation

11.76.2.1 TMotStsFlg TMotPVP::msf

motor status flag

Definition at line 163 of file kmlMotBase.h.

11.76.2.2 short TMotPVP::pos

position

Definition at line 164 of file kmlMotBase.h.

11.76.2.3 short TMotPVP::vel

velocity

Definition at line 165 of file kmlMotBase.h.

11.76.2.4 byte TMotPVP::pwm

pulse with modulation

Definition at line 166 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h

11.77 TMotSCP Struct Reference

[SCP] static controller parameters

`#include <kmlMotBase.h>`

Public Attributes

- byte maxppwm
  max. val for pos. voltage
- byte maxnpwm
  max. val for neg. voltage; pos!
- byte kP
  prop. factor of pos comp
- byte kI
  not yet active
- byte kD
  derivate factor of pos comp
- byte kARW
  not yet active
- byte kP_speed
  Proportional factor of the speed compensator.
- byte kI_speed
  Integral factor of the speed compensator.
- byte kD_speed
  Derivative factor of the speed compensator.
- byte maxppwm_nmp
  Max. value for positive voltage (0 => 0%, +70 => 100%).
- byte maxnpwm_nmp
  Max. value for negative voltage (0 => 0%, +70 => 100%).
- byte kspeed_nmp
  Proportional factor of speed compensator.
- byte kpos_nmp
  Proportional factor of position compensator.
- byte kI_nmp
  Integral factor (1/kI) of control output added to the final control output.
- int crash_limit_nmp
  Limit of error in position.

- int crash_limit_lin_nmp

  Limit of error in position in linear movement.

11.77.1 Detailed Description

[SCP] static controller parameters

Definition at line 110 of file kmlMotBase.h.

11.77.2 Member Data Documentation

11.77.2.1 byte TMotSCP::maxppwm

max. val for pos. voltage

Definition at line 114 of file kmlMotBase.h.

11.77.2.2 byte TMotSCP::maxnpwm

max. val for neg. voltage; pos!

Definition at line 115 of file kmlMotBase.h.

11.77.2.3 byte TMotSCP::kP

prop. factor of pos comp

Definition at line 116 of file kmlMotBase.h.

11.77.2.4 byte TMotSCP::kI

not yet active

Definition at line 117 of file kmlMotBase.h.

11.77.2.5 byte TMotSCP::kD

derivate factor of pos comp

Definition at line 118 of file kmlMotBase.h.

11.77.2.6 byte TMotSCP::kARW

not yet active

Definition at line 119 of file kmlMotBase.h.

11.77.2.7 byte TMotSCP::kP_speed

Proportional factor of the speed compensator.

Definition at line 121 of file kmlMotBase.h.

11.77.2.8 byte TMotSCP::kI_speed

Integral factor of the speed compensator.

Definition at line 122 of file kmlMotBase.h.

11.77.2.9 byte TMotSCP::kD_speed

Derivative factor of the speed compensator.

Definition at line 123 of file kmlMotBase.h.

11.77.2.10 byte TMotSCP::maxppwm_nmp

Max. value for positive voltage (0 => 0%, +70 => 100%).

Definition at line 127 of file kmlMotBase.h.

11.77.2.11 byte TMotSCP::maxnpwm_nmp

Max. value for negative voltage (0 => 0%, +70 => 100%).

Definition at line 128 of file kmlMotBase.h.

11.77.2.12 byte TMotSCP::kspeed_nmp

Proportional factor of speed compensator.

Definition at line 129 of file kmlMotBase.h.

11.77.2.13 byte TMotSCP::kpos_nmp

Proportional factor of position compensator.

Definition at line 130 of file kmlMotBase.h.

11.77.2.14 byte TMotSCP::kI_nmp

Integral factor (1/kI) of control output added to the final control output.

Definition at line 131 of file kmlMotBase.h.

11.77.2.15 int TMotSCP::crash_limit_nmp

Limit of error in position.

Definition at line 132 of file kmlMotBase.h.

11.77.2.16 int TMotSCP::crash_limit_lin_nmp

Limit of error in position in linear movement.

Definition at line 133 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h

11.78 TMotSFW Struct Reference

[SFW] slave firmware

`#include <kmlMotBase.h>`

Public Attributes

- byte version
  firmware version number
- byte subversion
  firmware subversion number
- byte revision
  firmware revision number
- byte type
  controller type
- byte subtype
  slave subtype

11.78.1 Detailed Description

[SFW] slave firmware

Definition at line 86 of file kmlMotBase.h.

11.78.2 Member Data Documentation

11.78.2.1 byte TMotSFW::version

firmware version number

Definition at line 87 of file kmlMotBase.h.

11.78.2.2 byte TMotSFW::subversion

firmware subversion number

Definition at line 88 of file kmlMotBase.h.

11.78.2.3 byte TMotSFW::revision

firmware revision number

Definition at line 89 of file kmlMotBase.h.

11.78.2.4 byte TMotSFW::type

controller type

Definition at line 90 of file kmlMotBase.h.

11.78.2.5 byte TMotSFW::subtype

slave subtype

Definition at line 91 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h

11.79 TMotTPS Struct Reference

[TPS] target position

`#include <kmlMotBase.h>`

Public Attributes

- TMotCmdFlg mcfTPS
  motor command flag
- short tarpos
  target position

11.79.1 Detailed Description

[TPS] target position

Definition at line 103 of file kmlMotBase.h.

11.79.2 Member Data Documentation

11.79.2.1 TMotCmdFlg TMotTPS::mcfTPS

motor command flag

Definition at line 104 of file kmlMotBase.h.

11.79.2.2 short TMotTPS::tarpos

target position

Definition at line 105 of file kmlMotBase.h.

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

- include/KNI/kmlMotBase.h

11.80 TMovement Struct Reference

structure for the

`#include <kni_wrapper.h>`

Collaboration diagram for TMovement:

Public Attributes

- TPos pos

  The position, see above struct TPos.

- ETransition transition

  the transition to this position, PTP or LINEAR

- int velocity

  The velocitiy for this particular movement.

- int acceleration

  the acceleration for this particular movement

11.80.1 Detailed Description

structure for the

Definition at line 63 of file kni_wrapper.h.

11.80.2 Member Data Documentation

11.80.2.1 TPos TMovement::pos

The position, see above struct TPos.

Definition at line 65 of file kni_wrapper.h.

11.80.2.2 ETransition TMovement::transition

the transition to this position, PTP or LINEAR

Definition at line 67 of file kni_wrapper.h.

11.80.2.3 int TMovement::velocity

The velocitiy for this particular movement.

Definition at line 69 of file kni_wrapper.h.

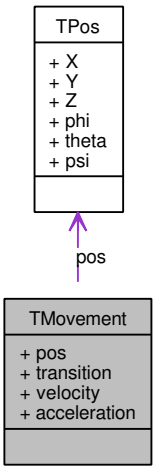
11.80.2.4 int TMovement::acceleration

the acceleration for this particular movement

Definition at line 71 of file kni_wrapper.h.

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

- include/kni_wrapper/kni_wrapper.h


TPos
+ X
+ Y
+ Z
+ phi
+ theta
+ psi
pos
TMovement
+ pos
+ transition
+ velocity
+ acceleration

11.81 TPacket Struct Reference

Communication packet.

`#include <cplSerial.h>`

Public Attributes

- byte send_sz
  send size of the packet
- byte read_sz
  read size of the packet

11.81.1 Detailed Description

Communication packet.

Definition at line 82 of file cplSerial.h.

11.81.2 Member Data Documentation

11.81.2.1 byte TPacket::send_sz

send size of the packet

Definition at line 83 of file cplSerial.h.

11.81.2.2 byte TPacket::read_sz

read size of the packet

Definition at line 84 of file cplSerial.h.

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

- include/KNI/cplSerial.h

11.82 TPos Struct Reference

extern C because we want to access these interfaces from anywhere:

`#include <kni_wrapper.h>`

Public Attributes

- double X

  The position in cartesian space.

- double Y
- double Z
- double phi

  the orientation of the TCP

- double theta
- double psi

11.82.1 Detailed Description

extern C because we want to access these interfaces from anywhere:

structure representing a point & orientation in space

Definition at line 49 of file kni_wrapper.h.

11.82.2 Member Data Documentation

11.82.2.1 double TPos::X

The position in cartesian space.

Definition at line 51 of file kni_wrapper.h.

11.82.2.2 double TPos::Y

Definition at line 51 of file kni_wrapper.h.

11.82.2.3 double TPos::Z

Definition at line 51 of file kni_wrapper.h.

11.82.2.4 double TPos::phi

the orientation of the TCP

Definition at line 53 of file kni_wrapper.h.

11.82.2.5 double TPos::theta

Definition at line 53 of file kni_wrapper.h.

11.82.2.6 double TPos::psi

Definition at line 53 of file kni_wrapper.h.

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

- include/kni_wrapper/kni_wrapper.h

11.83 TSctDAT Struct Reference

[DAT] sensor data

`#include <kmlSctBase.h>`

Public Attributes

- short cnt
  count of sensors
- short * arr
  sensor data

11.83.1 Detailed Description

[DAT] sensor data

Definition at line 57 of file kmlSctBase.h.

11.83.2 Member Data Documentation

11.83.2.1 short TSctDAT::cnt

count of sensors

Definition at line 58 of file kmlSctBase.h.

11.83.2.2 short* TSctDAT::arr

sensor data

Definition at line 59 of file kmlSctBase.h.

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

- include/KNI/kmlSctBase.h

11.84 TSctDesc Struct Reference

sensor controller description (partly)

`#include <kmlSctBase.h>`

Public Attributes

- byte ctrlID
  controller number (ID)
- short sens_res
  resolution: 8/12 bit
- short sens_count
  count of sensors

11.84.1 Detailed Description

sensor controller description (partly)

Definition at line 33 of file kmlSctBase.h.

11.84.2 Member Data Documentation

11.84.2.1 byte TSctDesc::ctrlID

controller number (ID)

Definition at line 34 of file kmlSctBase.h.

11.84.2.2 short TSctDesc::sens_res

resolution: 8/12 bit

Definition at line 35 of file kmlSctBase.h.

11.84.2.3 short TSctDesc::sens_count

count of sensors

Definition at line 36 of file kmlSctBase.h.

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

- include/KNI/kmlSctBase.h

11.85 TSctGNL Struct Reference

[GNL] controller generals

`#include <kmlSctBase.h>`

Collaboration diagram for TSctGNL:

Public Attributes

- CKatBase * own
  parent robot
- byte SID
  slave ID
- short res
  resolution: 8/12 bit

11.85.1 Detailed Description

[GNL] controller generals

Definition at line 49 of file kmlSctBase.h.

11.85.2 Member Data Documentation

11.85.2.1 CKatBase* TSctGNL::own

parent robot

Definition at line 50 of file kmlSctBase.h.

11.85.2.2 byte TSctGNL::SID

slave ID

Definition at line 51 of file kmlSctBase.h.

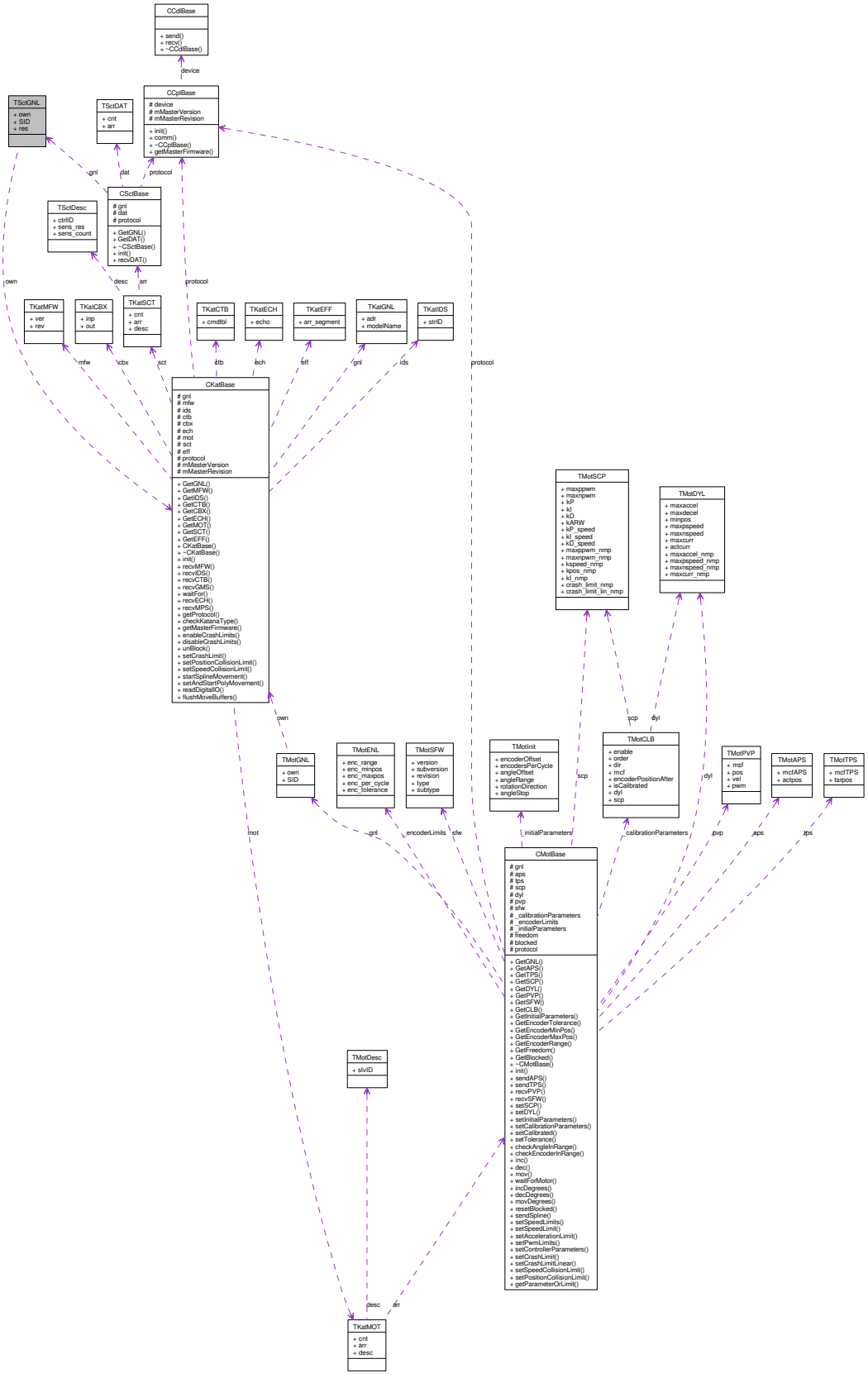
11.85.2.3 short TSctGNL::res

resolution: 8/12 bit

Definition at line 52 of file kmlSctBase.h.

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

- include/KNI/kmlSctBase.h


CCdlBase
+ send()
+ recv()
+ ~CCdlBase()
device
CCplBase
device
mMasterVersion
mMasterRevision
+ init()
+ comm()
+ ~CCplBase()
+ getMasterFirmware()
TSctGNL
+ own
+ SID
+ res
TSctDAT
+ cnt
+ arr
gnl
dat
protocol
CSctBase
gnl
dat
protocol
+ GetGNL()
+ GetDAT()
+ ~CSctBase()
+ init()
+ recvDAT()
TSctDesc
+ ctrlID
+ sens_res
+ sens_count
desc
arr
own
protocol
TKatMFW
+ ver
+ rev
TKatCBX
+ inp
+ out
TKatSCT
+ cnt
+ arr
+ desc
TKatCTB
+ cmdtbl
TKatECH
+ echo
TKatEFF
+ arr_segment
TKatGNL
+ adr
+ modelName
TKatIDS
+ strID
mfw
cbx
sct
ctb
ech
eff
gnl
ids
protocol
CKatBase
gnl
mfw
ids
ctb
cbx
ech
mot
sct
eff
protocol
mMasterVersion
mMasterRevision
+ GetGNL()
+ GetMFW()
+ GetIDS()
+ GetCTB()
+ GetCBX()
+ GetECH()
+ GetMOT()
+ GetSCT()
+ GetEFF()
+ CKatBase()
+ ~CKatBase()
+ init()
+ recvMFW()
+ recvIDS()
+ recvCTB()
+ recvGMS()
+ waitFor()
+ recvECHi()
+ recvMPS()
+ getProtocol()
+ checkKatanaType()
+ getMasterFirmware()
+ enableCrashLimits()
+ disableCrashLimits()
+ unBlock()
+ setCrashLimit()
+ setPositionCollisionLimit()
+ setSpeedCollisionLimit()
+ startSplineMovement()
+ setAndStartPolyMovement()
+ readDigitalIO()
+ flushMoveBuffers()
TMotSCP
+ maxppwm
+ maxnpwm
+ kP
+ kI
+ kD
+ kARW
+ kP_speed
+ kI_speed
+ kD_speed
+ maxppwm_nmp
+ maxnpwm_nmp
+ kspeed_nmp
+ kpos_nmp
+ kI_nmp
+ crash_limit_nmp
+ crash_limit_lin_nmp
TMotDYL
+ maxaccel
+ maxdecel
+ minpos
+ maxpspeed
+ maxnspeed
+ maxcurr
+ actcurr
+ maxaccel_nmp
+ maxpspeed_nmp
+ maxnspeed_nmp
+ maxcurr_nmp
own
TMotGNL
+ own
+ SID
TMotENL
+ enc_range
+ enc_minpos
+ enc_maxpos
+ enc_per_cycle
+ enc_tolerance
TMotSFW
+ version
+ subversion
+ revision
+ type
+ subtype
TMotInit
+ encoderOffset
+ encodersPerCycle
+ angleOffset
+ angleRange
+ rotationDirection
+ angleStop
TMotCLB
+ enable
+ order
+ dir
+ mcf
+ encoderPositionAfter
+ isCalibrated
+ dyl
+ scp
TMotPVP
+ mcf
+ pos
+ vel
+ pwm
TMotAPS
+ mcfAPS
+ actpos
TMotTPS
+ mcfTPS
+ tarpos
scp
dyl
mot
gnl
encoderLimits
sfw
initialParameters
calibrationParameters
pvp
aps
tps
CMotBase
gnl
aps
tps
scp
dyl
pvp
sfw
_calibrationParameters
_encoderLimits
_initialParameters
freedom
blocked
protocol
+ GetGNL()
+ GetAPS()
+ GetTPS()
+ GetSCP()
+ GetDYL()
+ GetPVP()
+ GetSFW()
+ GetCLB()
+ GetInitialParameters()
+ GetEncoderTolerance()
+ GetEncoderMinPos()
+ GetEncoderMaxPos()
+ GetEncoderRange()
+ GetFreedom()
+ GetBlocked()
+ ~CMotBase()
+ init()
+ sendAPS()
+ sendTPS()
+ recvPVP()
+ recvSFW()
+ setSCP()
+ setDYL()
+ setInitialParameters()
+ setCalibrationParameters()
+ setCalibrated()
+ setTolerance()
+ checkAngleInRange()
+ checkEncoderInRange()
+ inc()
+ dec()
+ mov()
+ waitForMotor()
+ incDegrees()
+ decDegrees()
+ movDegrees()
+ resetBlocked()
+ sendSpline()
+ setSpeedLimits()
+ setSpeedLimit()
+ setAccelerationLimit()
+ setPwmLimits()
+ setControllerParameters()
+ setCrashLimit()
+ setCrashLimitLinear()
+ setSpeedCollisionLimit()
+ setPositionCollisionLimit()
+ getParameterOrLimit()
TMotDesc
+ slvID
desc
arr
TKatMOT
+ cnt
+ arr
+ desc

11.86 KNI_MHF::unary_deg2rad< _T > Struct Template Reference

a function-object version of rad2deg

`#include <MathHelperFunctions.h>`

Public Member Functions

- _T operator() (const _T a)

11.86.1 Detailed Description

template<typename _T> struct KNI_MHF::unary_deg2rad< _T >

a function-object version of rad2deg

Definition at line 126 of file MathHelperFunctions.h.

11.86.2 Member Function Documentation

11.86.2.1 template<typename _T> _T KNI_MHF::unary_deg2rad< _T >::operator() (const _T *a*) [inline]

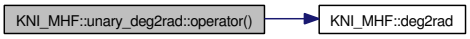
Definition at line 127 of file MathHelperFunctions.h.

References KNI_MHF::deg2rad().

Here is the call graph for this function:

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

- include/common/MathHelperFunctions.h


KNI_MHF::unary_deg2rad::operator()
KNI_MHF::deg2rad

11.87 KNI_MHF::unary_precalc_cos< _T > Struct Template Reference

`#include <MathHelperFunctions.h>`

Public Member Functions

- _T operator() (_T x)

11.87.1 Detailed Description

template<typename _T> struct KNI_MHF::unary_precalc_cos< _T >

See also:

unary_precalc_sin

Definition at line 53 of file MathHelperFunctions.h.

11.87.2 Member Function Documentation

11.87.2.1 template<typename _T> _T KNI_MHF::unary_precalc_cos< _T >::operator() (_T *x*) [inline]

Definition at line 54 of file MathHelperFunctions.h.

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

- include/common/MathHelperFunctions.h

11.88 KNI_MHF::unary_precalc_sin< _T > Struct Template Reference

function-object which calculates sinus for n-elements of a container if used together with a STL algorithm

`#include <MathHelperFunctions.h>`

Public Member Functions

- _T operator() (_T &x)

11.88.1 Detailed Description

template<typename _T> struct KNI_MHF::unary_precalc_sin< _T >

function-object which calculates sinus for n-elements of a container if used together with a STL algorithm

Definition at line 44 of file MathHelperFunctions.h.

11.88.2 Member Function Documentation

11.88.2.1 template<typename _T> _T KNI_MHF::unary_precalc_sin< _T >::operator() (_T & *x*) [inline]

Definition at line 45 of file MathHelperFunctions.h.

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

- include/common/MathHelperFunctions.h

11.89 KNI_MHF::unary_rad2deg< _T > Struct Template Reference

a function-object version of rad2deg

`#include <MathHelperFunctions.h>`

Public Member Functions

- _T operator() (const _T a)

11.89.1 Detailed Description

template<typename _T> struct KNI_MHF::unary_rad2deg< _T >

a function-object version of rad2deg

Definition at line 112 of file MathHelperFunctions.h.

11.89.2 Member Function Documentation

11.89.2.1 template<typename _T> _T KNI_MHF::unary_rad2deg< _T >::operator() (const _T *a*) [inline]

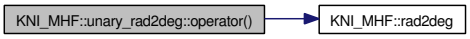
Definition at line 113 of file MathHelperFunctions.h.

References KNI_MHF::rad2deg().

Here is the call graph for this function:

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

- include/common/MathHelperFunctions.h


KNI_MHF::unary_rad2deg::operator()
KNI_MHF::rad2deg

11.90 WaitParameterException Class Reference

Wait parameter set to false.

`#include <lmBase.h>`

Inheritance diagram for WaitParameterException: Collaboration diagram for WaitParameterException:

Public Member Functions

- WaitParameterException () throw ()

11.90.1 Detailed Description

Wait parameter set to false.

Note:

error_number = -71

Definition at line 60 of file lmBase.h.

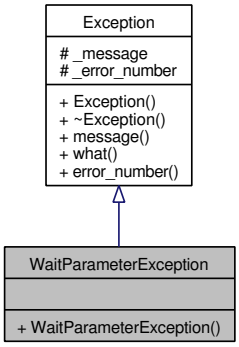
11.90.2 Constructor & Destructor Documentation

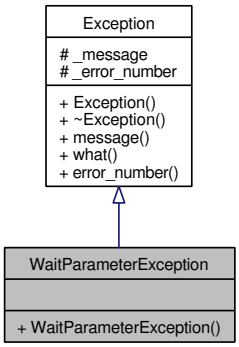
11.90.2.1 WaitParameterException::WaitParameterException () throw () [inline]

Definition at line 62 of file lmBase.h.

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

- include/KNI_LM/lmBase.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
WaitParameterException
+ WaitParameterException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
WaitParameterException
+ WaitParameterException()

11.91 WriteNotCompleteException Class Reference

Not all bytes could be written to the serial communication device.

`#include <cdlCOMExceptions.h>`

Inheritance diagram for WriteNotCompleteException: Collaboration diagram for WriteNotCompleteException:

Public Member Functions

- WriteNotCompleteException (const std::string &port) throw ()

11.91.1 Detailed Description

Not all bytes could be written to the serial communication device.

Note:

error_number=-15

Definition at line 103 of file cdlCOMExceptions.h.

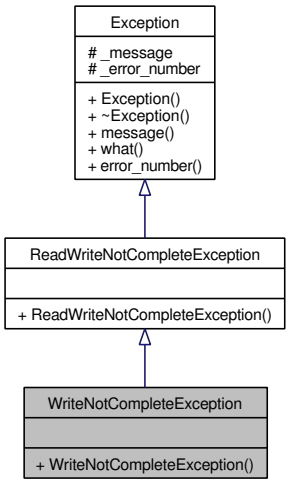
11.91.2 Constructor & Destructor Documentation

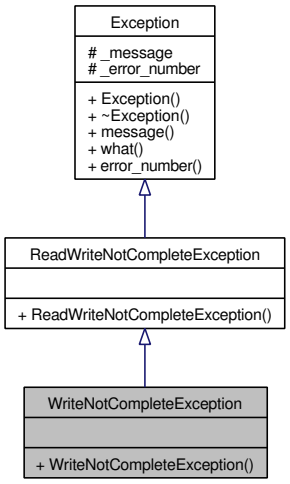
11.91.2.1 WriteNotCompleteException::WriteNotCompleteException (const std::string & *port*) throw () [inline]

Definition at line 105 of file cdlCOMExceptions.h.

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

- include/KNI/cdlCOMExceptions.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ReadWriteNotCompleteException
+ ReadWriteNotCompleteException()
WriteNotCompleteException
+ WriteNotCompleteException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
ReadWriteNotCompleteException
+ ReadWriteNotCompleteException()
WriteNotCompleteException
+ WriteNotCompleteException()

11.92 WrongCRCException Class Reference

CRC check for the answer package failed.

`#include <cplSerial.h>`

Inheritance diagram for WrongCRCException: Collaboration diagram for WrongCRCException:

Public Member Functions

- WrongCRCException () throw ()

11.92.1 Detailed Description

CRC check for the answer package failed.

Definition at line 44 of file cplSerial.h.

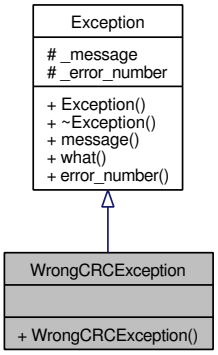
11.92.2 Constructor & Destructor Documentation

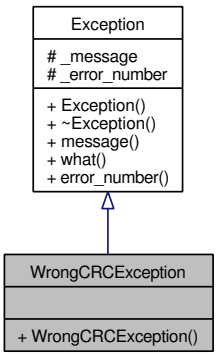
11.92.2.1 WrongCRCException::WrongCRCException () throw () [inline]

Definition at line 46 of file cplSerial.h.

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

- include/KNI/cplSerial.h


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
WrongCRCException
+ WrongCRCException()


Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
WrongCRCException
+ WrongCRCException()

11.93 WrongParameterException Class Reference

The given parameter was wrong.

`#include <kmlCommon.h>`

Inheritance diagram for WrongParameterException: Collaboration diagram for WrongParameterException:

Public Member Functions

- WrongParameterException (const std::string &para) throw ()

11.93.1 Detailed Description

The given parameter was wrong.

Note:

error_number=-34

Definition at line 62 of file kmlCommon.h.

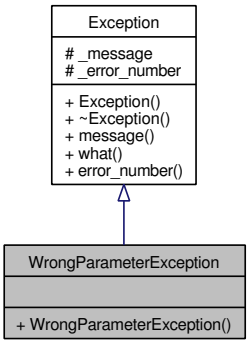
11.93.2 Constructor & Destructor Documentation

11.93.2.1 WrongParameterException::WrongParameterException (const std::string & *para*) throw () [inline]

Definition at line 64 of file kmlCommon.h.

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

- include/KNI/kmlCommon.h



Exception
_message
_error_number
+ Exception()
+ ~Exception()
+ message()
+ what()
+ error_number()
WrongParameterException
+ WrongParameterException()

12 File Documentation

12.1 include/common/dllexport.h File Reference

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

Defines

- #define DLLDIR
- #define DLLDIR_IK
- #define DLLDIR_LM

12.1.1 Define Documentation

12.1.1.1 #define DLLDIR

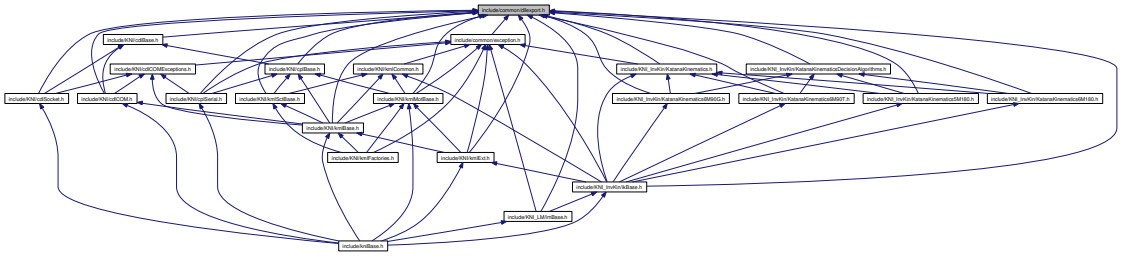
Definition at line 30 of file dllexport.h.

12.1.1.2 #define DLLDIR_IK

Definition at line 31 of file dllexport.h.

12.1.1.3 #define DLLDIR_LM

Definition at line 32 of file dllexport.h.



12.2 include/common/exception.h File Reference

```
#include <string>
```

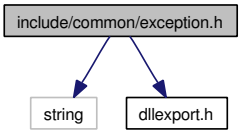
```
#include "dllexport.h"
```

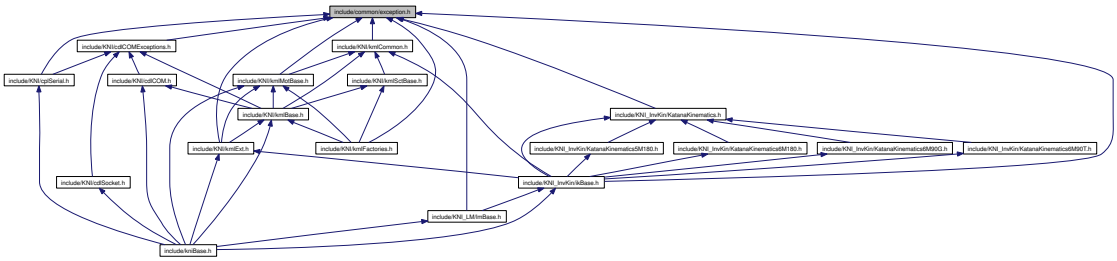
Include dependency graph for exception.h:

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

Classes

- struct Context
- class Exception


include/common/exception.h
string
dllexport.h



12.3 include/common/MathHelperFunctions.h File Reference

```
#include <cmath>
#include <vector>
#include <functional>
#include <cassert>
```

Include dependency graph for MathHelperFunctions.h:

Namespaces

- namespace KNI_MHF

Classes

- struct KNI_MHF::unary_precalc_sin< _T >
  function-object which calculates sinus for n-elements of a container if used together with a STL algorithm
- struct KNI_MHF::unary_precalc_cos< _T >
- struct KNI_MHF::unary_rad2deg< _T >
  a function-object version of rad2deg
- struct KNI_MHF::unary_deg2rad< _T >
  a function-object version of rad2deg

Defines

- #define M_PI 3.14159265358979323846

Functions

- template<typename _T>
  short KNI_MHF::sign (_T x)
- template<typename _T>
  _T KNI_MHF::atan1 (_T in1, _T in2)
- template<typename _T>
  _T KNI_MHF::acotan (const _T in)
- template<typename _T>
  _T KNI_MHF::atan0 (const _T in1, const _T in2)
- template<typename _T>
  _T KNI_MHF::pow2 (const _T in)
- template<typename _T>
  _T KNI_MHF::rad2deg (const _T a)
  conversion from radian to degree
- template<typename _T>
  _T KNI_MHF::deg2rad (const _T a)
  conversion from degree to radian

- template<typename _T>
  _T KNI_MHF::anglereduce (const _T a)
- template<typename _angleT, typename _encT>
  _encT KNI_MHF::rad2enc (_angleT const &angle, _angleT const &angleOffset, _encT const &epc, _encT const &encOffset, _encT const &rotDir)

  converts absolute angles in radian to encoders.

- template<typename _angleT, typename _encT>
  _angleT KNI_MHF::enc2rad (_encT const &enc, _angleT const &angleOffset, _encT const &epc, _encT const &encOffset, _encT const &rotDir)

  converts encoders to absolute angles in radian

- double KNI_MHF::findFirstEqualAngle (double cosValue, double sinValue, double tolerance)

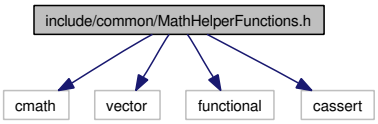
  Find the first equal angle.

12.3.1 Define Documentation

12.3.1.1 #define M_PI 3.14159265358979323846

Definition at line 21 of file MathHelperFunctions.h.

Referenced by KNI_MHF::acotan(), KNI_MHF::anglereduce(), KNI_MHF::atan0(), KNI_MHF::atan1(), KNI_MHF::deg2rad(), KNI_MHF::enc2rad(), KNI_MHF::findFirstEqualAngle(), KNI_MHF::rad2deg(), and KNI_MHF::rad2enc().


include/common/MathHelperFunctions.h
cmath
vector
functional
cassert

12.4 include/common/Timer.h File Reference

`#include <ctime>`

`#include <sys/time.h>`

Include dependency graph for Timer.h:

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

Namespaces

- namespace KNI

Classes

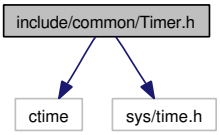
- class KNI::Timer

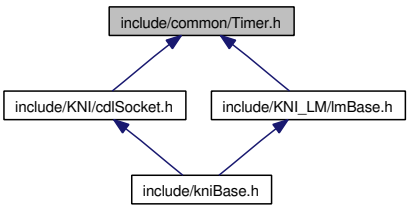
  Provides a stop-watch-like class with a resolution of milliseconds.

Functions

- void KNI::sleep (long time)

  This functions shields the platform specific implementation of the sleep function.


include/common/Timer.h
ctime
sys/time.h


include/common/Timer.h
include/KNI/cdlSocket.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.5 include/KNI/cdlBase.h File Reference

`#include "common/dllexport.h"`

Include dependency graph for cdlBase.h:

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

Classes

- class CCdlBase
  Abstract base class for devices.

Defines

- #define BYTE_DECLARED

Typedefs

- typedef unsigned char byte
  type specification (8 bit)

12.5.1 Define Documentation

12.5.1.1 #define BYTE_DECLARED

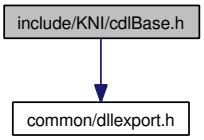
Definition at line 28 of file cdlBase.h.

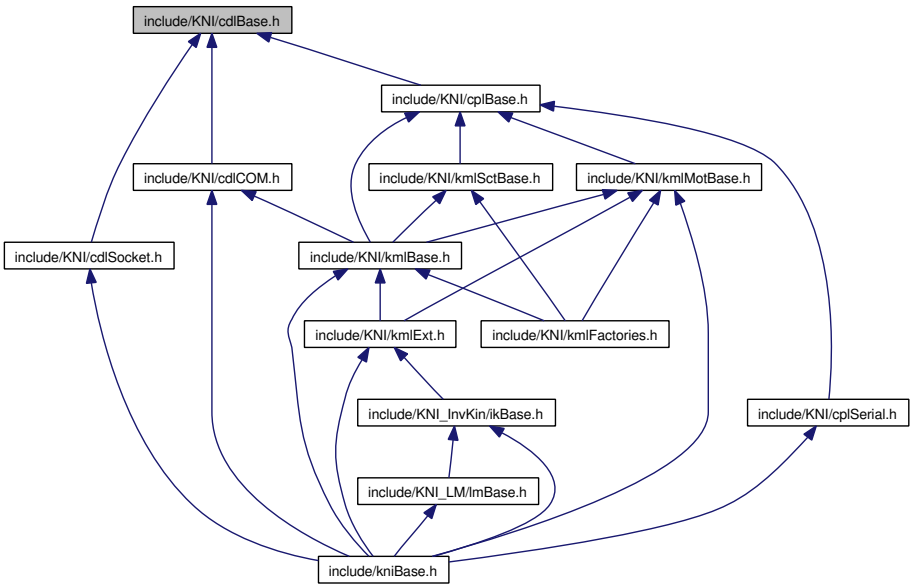
12.5.2 Typedef Documentation

12.5.2.1 typedef unsigned char byte

type specification (8 bit)

Definition at line 29 of file cdlBase.h.


include/KNI/cdlBase.h
common/dllexport.h


include/KNI/cdlBase.h
include/KNI/cplBase.h
include/KNI/cdlCOM.h
include/KNI/kmlSctBase.h
include/KNI/kmlMotBase.h
include/KNI/cdlSocket.h
include/KNI/kmlBase.h
include/KNI/kmlExt.h
include/KNI/kmlFactories.h
include/KNI_InvKin/ikBase.h
include/KNI/cplSerial.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.6 include/KNI/cdlCOM.h File Reference

```
#include "common/dllexport.h"
#include "KNI/cdlBase.h"
#include "KNI/cdlCOMExceptions.h"
#include <string>
#include <termios.h>
#include <fcntl.h>
#include <cerrno>
```

Include dependency graph for cdlCOM.h:

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

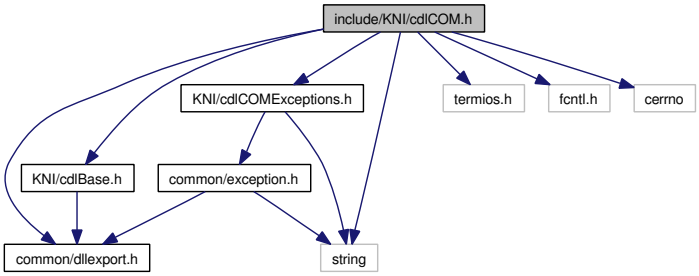
Classes

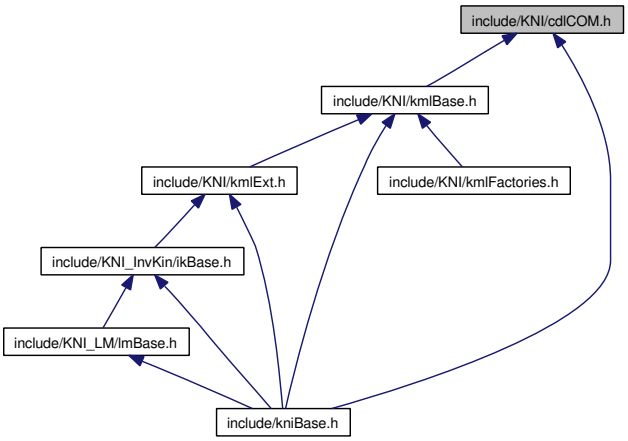
- struct TCdlCOMDesc

  This structrue stores the attributes for a serial port device.

- class CCdlCOM

  Encapsulates the serial port device.


include/KNI/cdlCOM.h
KNI/cdlCOMExceptions.h
termios.h
fcntl.h
cerrno
KNI/cdlBase.h
common/exception.h
common/dllexport.h
string


include/KNI/cdlCOM.h
include/KNI/kmlBase.h
include/KNI/kmlExt.h
include/KNI/kmlFactories.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.7 include/KNI/cdlCOMExceptions.h File Reference

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

```
#include <string>
```

Include dependency graph for cdlCOMExceptions.h:

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

Classes

- class CannotOpenPortException

  Failed to open the serial communication device.

- class CannotGetSetPortAttributesException

  Could not set or get the attributes for the given serial communication device.

- class PortNotOpenException

  The port was not open.

- class DeviceReadException

  Reading from the serial communication device failed.

- class DeviceWriteException

  Writing to the serial communication device failed.

- class ReadWriteNotCompleteException

  This exception is the base for the WriteNotComplete and ReadNotCompleteException.

- class WriteNotCompleteException

  Not all bytes could be written to the serial communication device.

- class ReadNotCompleteException

  The Katana didn't answer correctly within the given timeout.

- class ErrorException

  The Katana returned an error string.

Enumerations

- enum {

  ERR_FAILED = -1, ERR_INVALID_ARGUMENT = -2, ERR_STATE_MISMATCH = -3, ERR_TYPE_-MISMATCH = -4,

  ERR_RANGE_MISMATCH = -5, ERR_AXIS_HEARTBEAT = -6, ERR_AXIS_OPERATIONAL = -7, ERR_AXIS_MOVE = -8,

  ERR_AXIS_MOVE_POLY = -9, ERR_AXIS_COLLISION = -10, ERR_AXIS_ANY = -11, ERR_CRC = -12,

  ERR_PERIPHERAL = -13, ERR_MESSAGE = 192, ERR_MESSAGE_STRING = 193 }

  Error codes in error handling strings.

12.7.1 Enumeration Type Documentation

12.7.1.1 anonymous enum

Error codes in error handling strings.

Enumerator:

ERR_FAILED

ERR_INVALID_ARGUMENT

ERR_STATE_MISMATCH

ERR_TYPE_MISMATCH

ERR_RANGE_MISMATCH

ERR_AXIS_HEARTBEAT

ERR_AXIS_OPERATIONAL

ERR_AXIS_MOVE

ERR_AXIS_MOVE_POLY

ERR_AXIS_COLLISION

ERR_AXIS_ANY

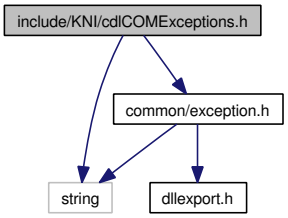
ERR_CRC

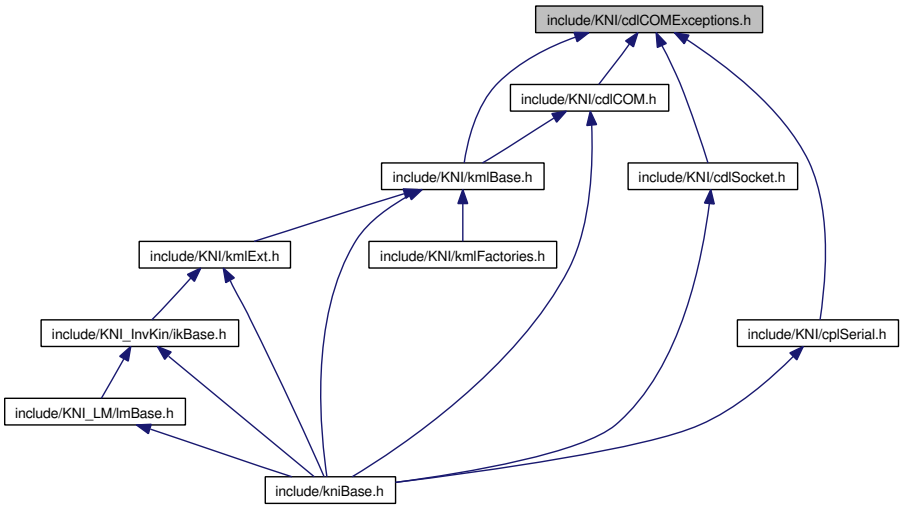
ERR_PERIPHERAL

ERR_MESSAGE

ERR_MESSAGE_STRING

Definition at line 20 of file cdlCOMExceptions.h.


include/KNI/cdlCOMExceptions.h
common/exception.h
string
dllexport.h


include/KNI/cdlCOMExceptions.h
include/KNI/cdlCOM.h
include/KNI/kmlBase.h
include/KNI/cdlSocket.h
include/KNI/kmlExt.h
include/KNI/kmlFactories.h
include/KNI_InvKin/ikBase.h
include/KNI/cplSerial.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.8 include/KNI/cdlSocket.h File Reference

```
#include "common/dllexport.h"
#include "common/Timer.h"
#include "KNI/cdlBase.h"
#include "KNI/cdlCOMExceptions.h"
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <cerrno>
#include <unistd.h>
#include <string>
```

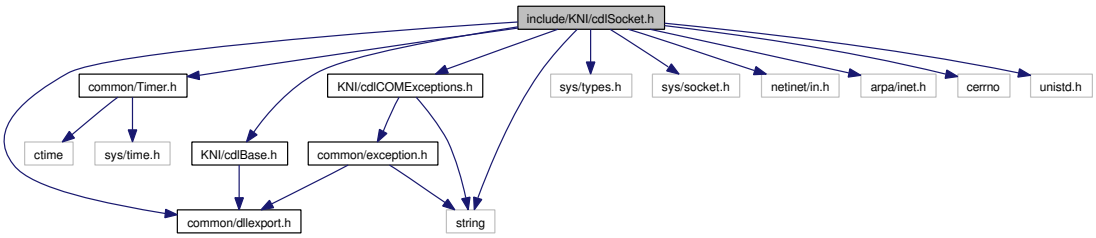
Include dependency graph for cdlSocket.h:

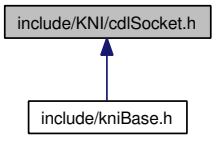
This graph shows which files directly or indirectly include this file:

Classes

- class CCdlSocket

  Encapsulates the socket communication device.


include/KNI/cdlSocket.h
common/Timer.h
KNI/cdlCOMExceptions.h
sys/types.h
sys/socket.h
netinet/in.h
arpa/inet.h
cerrno
unistd.h
ctime
sys/time.h
KNI/cdlBase.h
common/exception.h
common/dllexport.h
string


include/KNI/cdlSocket.h
include/kniBase.h

12.9 include/KNI/cplBase.h File Reference

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

```
#include "KNI/cdlBase.h"
```

Include dependency graph for cplBase.h:

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

Classes

- class CCplBase
  Abstract base class for protocol definiton.

Defines

- #define BYTE_DECLARED

Typedefs

- typedef unsigned char byte
  type specification (8 bit)

12.9.1 Define Documentation

12.9.1.1 #define BYTE_DECLARED

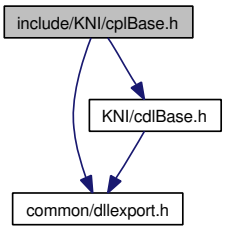
Definition at line 32 of file cplBase.h.

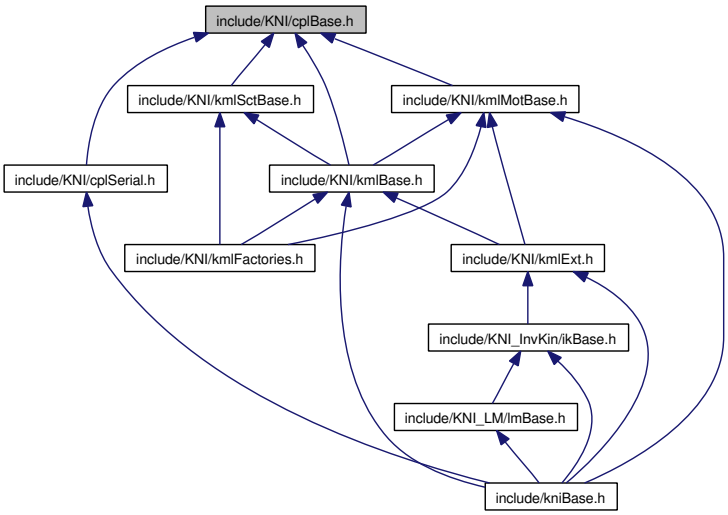
12.9.2 Typedef Documentation

12.9.2.1 typedef unsigned char byte

type specification (8 bit)

Definition at line 33 of file cplBase.h.


include/KNI/cplBase.h
KNI/cdlBase.h
common/dllexport.h


include/KNI/cplBase.h
include/KNI/kmlSctBase.h
include/KNI/kmlMotBase.h
include/KNI/cplSerial.h
include/KNI/kmlBase.h
include/KNI/kmlFactories.h
include/KNI/kmlExt.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.10 include/KNI/cplSerial.h File Reference

```
#include "common/dllexport.h"
#include "common/exception.h"
#include "KNI/cplBase.h"
#include "KNI/cdlCOMExceptions.h"
```

Include dependency graph for cplSerial.h:

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

Classes

- class WrongCRCException
  CRC check for the answer package failed.
- class FirmwareException
  Exception reported by the firmware.
- struct THeader
  Header of a communication packet.
- struct TPacket
  Communication packet.
- class CCplSerial
  Base class of two different serial protocols.
- class CCplSerialCRC
  Implement the Serial-Zero protocol.

Defines

- #define NUMBER_OF_RETRIES_SEND 3
- #define NUMBER_OF_RETRIES_RECV 3

Variables

- const int KATANA_ERROR_FLAG = 192
  defines the error flag number

12.10.1 Define Documentation

12.10.1.1 #define NUMBER_OF_RETRIES_RECV 3

Definition at line 32 of file cplSerial.h.

12.10.1.2 #define NUMBER_OF_RETRIES_SEND 3

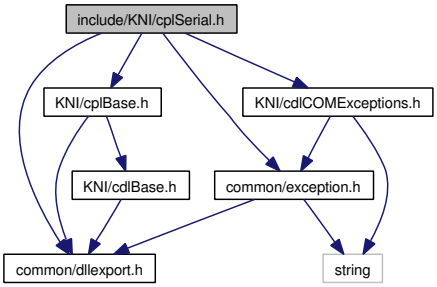
Definition at line 31 of file cplSerial.h.

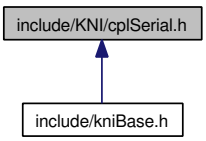
12.10.2 Variable Documentation

12.10.2.1 const int KATANA_ERROR_FLAG = 192

defines the error flag number

Definition at line 36 of file cplSerial.h.


include/KNI/cplSerial.h
KNI/cplBase.h
KNI/cdlCOMExceptions.h
KNI/cdlBase.h
common/exception.h
common/dllexport.h
string


include/KNI/cplSerial.h
include/kniBase.h

12.11 include/KNI/CRC.h File Reference

Defines

- #define uint8 unsigned char
  unsigned 8 bit
- #define uint16 unsigned short
  unsigned 16 bit

Functions

- uint16 CRC_CHECKSUM (uint8 *data, uint8 size_of_BYTE)

12.11.1 Define Documentation

12.11.1.1 #define uint16 unsigned short

unsigned 16 bit

Definition at line 28 of file CRC.h.

12.11.1.2 #define uint8 unsigned char

unsigned 8 bit

Definition at line 27 of file CRC.h.

12.11.2 Function Documentation

12.11.2.1 uint16 CRC_CHECKSUM (uint8 * *data*, uint8 *size_of_BYTE*)

12.12 include/KNI/kmlBase.h File Reference

```
#include "common/dllexport.h"
#include "KNI/cplBase.h"
#include "KNI/kmlCommon.h"
#include "KNI/kmlMotBase.h"
#include "KNI/kmlSctBase.h"
#include "KNI/cdlCOM.h"
#include "KNI/cdlCOMExceptions.h"
```

Include dependency graph for kmlBase.h:

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

Classes

- struct TKatGNL
  [GNL] general robot attributes
- struct TKatMFW
  [MFW] master firmware version/revision number
- struct TKatIDS
  [IDS] identification string
- struct TKatCTB
  [CTB] command table defined in the firmware
- struct TKatCBX
  [CBX] connector box
- struct TKatECH
  [ECH] echo
- struct TKatEFF
  Inverse Kinematics structure of the endeffektor.
- class CKatBase
  Base Katana class.

Defines

- #define K400_OLD_PROTOCOL_THRESHOLD 3
  The old protocol is only supported up to K400 version 0.x.x.
- #define BYTE_DECLARED
- #define TM_ENDLESS -1
  timeout symbol for 'endless' waiting

Typedefs

- typedef unsigned char byte
  type specification (8 bit)

12.12.1 Define Documentation

12.12.1.1 #define BYTE_DECLARED

Definition at line 45 of file kmlBase.h.

12.12.1.2 #define K400_OLD_PROTOCOL_THRESHOLD 3

The old protocol is only supported up to K400 version 0.x.x.

Definition at line 42 of file kmlBase.h.

12.12.1.3 #define TM_ENDLESS -1

timeout symbol for 'endless' waiting

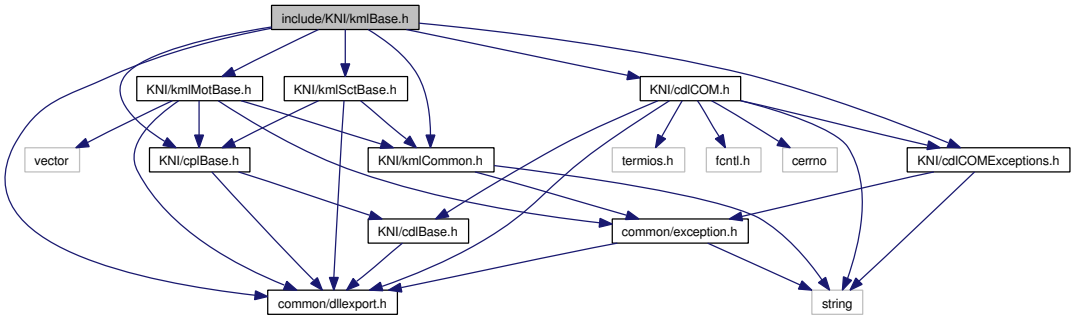
Definition at line 51 of file kmlBase.h.

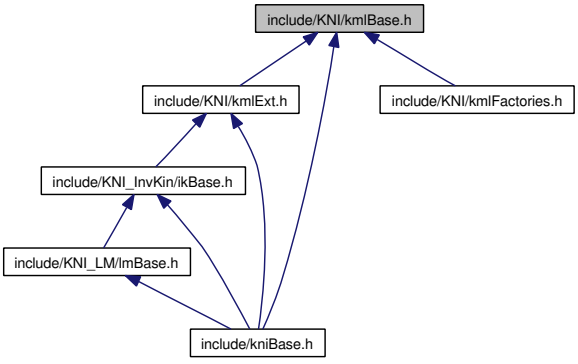
12.12.2 Typedef Documentation

12.12.2.1 typedef unsigned char byte

type specification (8 bit)

Definition at line 46 of file kmlBase.h.


include/KNI/kmlBase.h
KNI/kmlMotBase.h
KNI/kmlSctBase.h
KNI/cdlCOM.h
vector
KNI/cplBase.h
KNI/kmlCommon.h
termios.h
fcntl.h
cerrno
KNI/cdlCOMExceptions.h
KNI/cdlBase.h
common/exception.h
common/dllexport.h
string


include/KNI/kmlBase.h
include/KNI/kmlExt.h
include/KNI/kmlFactories.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.13 include/KNI/kmlCommon.h File Reference

`#include "common/exception.h"`

`#include <string>`

Include dependency graph for kmlCommon.h:

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

Classes

- class SlaveErrorException
  Slave error occurred.
- class ParameterReadingException
  There was an error while reading a parameter from the robot.
- class ParameterWritingException
  The data you wanted to send to the robot was invalid.
- class WrongParameterException
  The given parameter was wrong.
- class MotorOutOfRangeException
  The encoders for the given motor were out of range.
- class MotorTimeoutException
  The timeout elapsed for the given motor and target position.
- class MotorCrashException
  The requested motor crashed during the movement.

Defines

- #define TM_ENDLESS -1
  timeout symbol for 'endless' waiting
- #define BYTE_DECLARED

Typedefs

- typedef unsigned char byte
  type specification (8 bit)

12.13.1 Define Documentation

12.13.1.1 #define BYTE_DECLARED

Definition at line 22 of file kmlCommon.h.

12.13.1.2 #define TM_ENDLESS -1

timeout symbol for 'endless' waiting

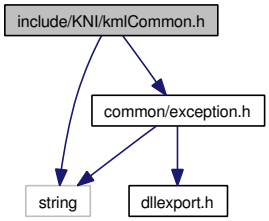
Definition at line 19 of file kmlCommon.h.

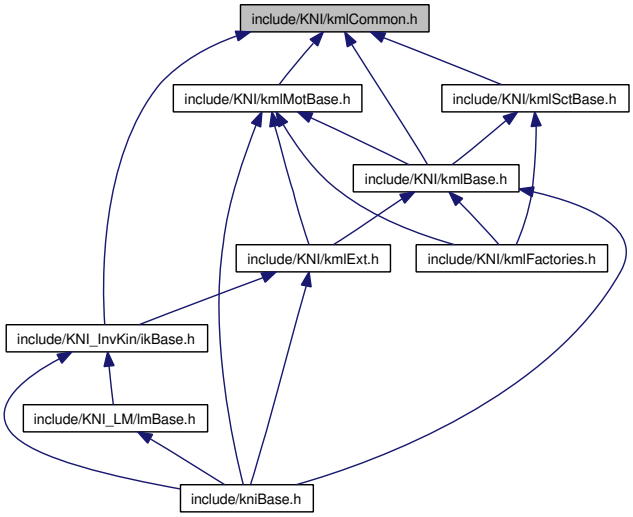
12.13.2 Typedef Documentation

12.13.2.1 typedef unsigned char byte

type specification (8 bit)

Definition at line 23 of file kmlCommon.h.


include/KNI/kmlCommon.h
common/exception.h
string
dllexport.h


include/KNI/kmlCommon.h
include/KNI/kmlMotBase.h
include/KNI/kmlSctBase.h
include/KNI/kmlBase.h
include/KNI/kmlExt.h
include/KNI/kmlFactories.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.14 include/KNI/kmlExt.h File Reference

```
#include "common/dllexport.h"
#include "common/exception.h"
#include "KNI/kmlBase.h"
#include "KNI/kmlMotBase.h"
#include <vector>
```

Include dependency graph for kmlExt.h:

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

Namespaces

- namespace KNI

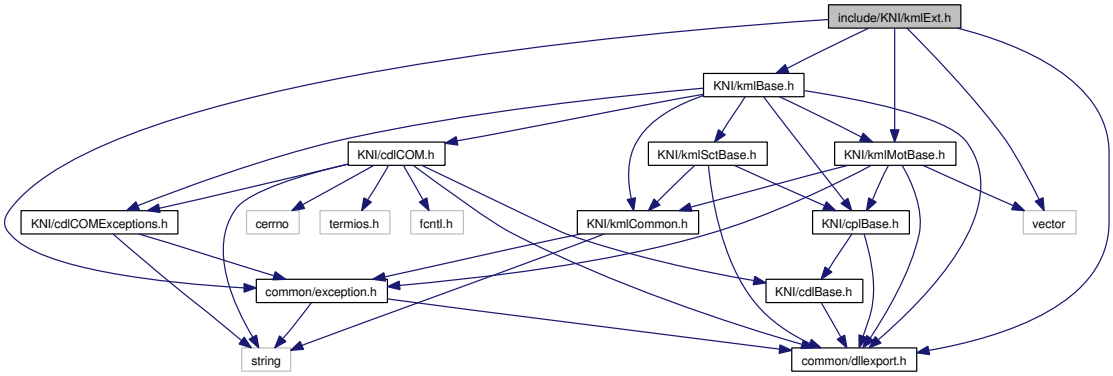
Classes

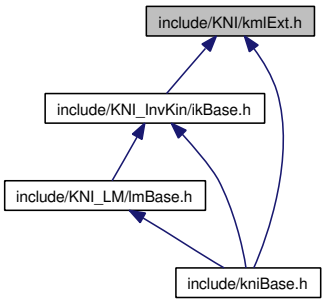
- class ConfigFileOpenException

  Accessing the given configuration file failed (may be: access denied or wrong path).

- class CKatana

  Extended Katana class with additional functions.


include/KNI/kmlExt.h
KNI/kmlBase.h
KNI/cdlCOM.h
KNI/kmlSctBase.h
KNI/kmlMotBase.h
KNI/cdlCOMExceptions.h
cerrno
termios.h
fcntl.h
KNI/kmlCommon.h
KNI/cplBase.h
vector
common/exception.h
KNI/cdlBase.h
string
common/dllexport.h


include/KNI/kmlExt.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.15 include/KNI/kmlFactories.h File Reference

```
#include "common/exception.h"
#include "KNI/kmlBase.h"
#include "KNI/kmlMotBase.h"
#include "KNI/kmlSctBase.h"
#include <string>
#include <fstream>
```

Include dependency graph for kmlFactories.h:

Namespaces

- namespace KNI

Classes

- class ConfigFileStateException

  The state of the configuration file wasn't "good".

- class ConfigFileSectionNotFoundException

  The requested section could not be found.

- class ConfigFileSubsectionNotFoundException

  The requested subsection could not be found.

- class ConfigFileEntryNotFoundException

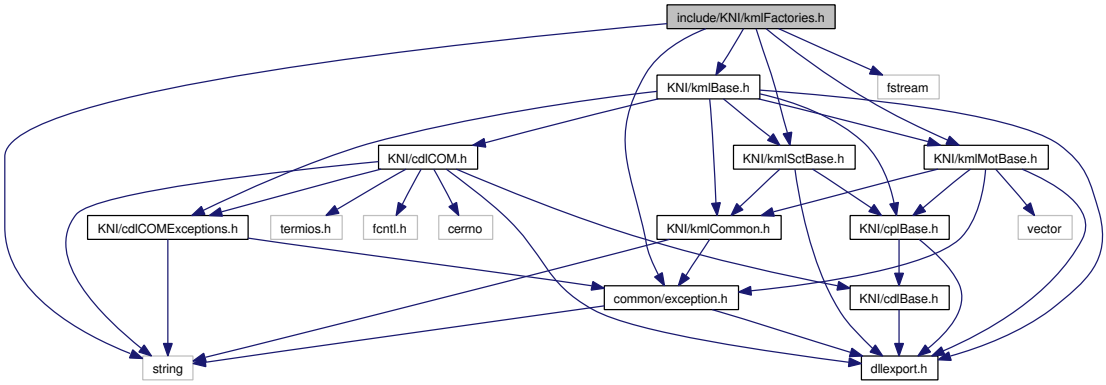
  The requested entry could not be found.

- class ConfigFileSyntaxErrorException

  There was a syntax error in the configuration file.

- class KNI::kmlFactory

  This class is for internal use only It may change at any time It shields the configuration file parsing.


include/KNI/kmlFactories.h
KNI/kmlBase.h
fstream
KNI/cdlCOM.h
KNI/kmlSctBase.h
KNI/kmlMotBase.h
KNI/cdlCOMExceptions.h
termios.h
fcntl.h
cermo
KNI/kmlCommon.h
KNI/cplBase.h
vector
common/exception.h
KNI/cdlBase.h
string
dllexport.h

12.16 include/KNI/kmlMotBase.h File Reference

```
#include "common/exception.h"
#include "common/dllexport.h"
#include "KNI/kmlCommon.h"
#include "KNI/cplBase.h"
#include <vector>
```

Include dependency graph for kmlMotBase.h:

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

Classes

- struct TMotDesc
  motor description (partly)
- struct TKatMOT
  [MOT] every motor's attributes
- struct TMotGNL
  [GNL] motor generals
- struct TMotSFW
  [SFW] slave firmware
- struct TMotAPS
  [APS] actual position
- struct TMotTPS
  [TPS] target position
- struct TMotSCP
  [SCP] static controller parameters
- struct TMotDYL
  [DYL] dynamic limits
- struct TMotPVP
  [PVP] position, velocity, pulse width modulation
- struct TMotENL
  [ENL] limits in encoder values (INTERNAL STRUCTURE!)
- struct TMotCLB
  Calibration structure for single motors.
- struct TMotInit
  Initial motor parameters.
- class CMotBase
  Motor class.

Enumerations

- enum TMotCmdFlg {
  MCF_OFF = 0, MCF_CALIB = 4, MCF_FREEZE = 8, MCF_ON = 24,
  MCF_CLEAR_MOVEBUFFER = 32 }
  command flags
- enum TMotStsFlg {
  MSF_MECHSTOP = 1, MSF_MAXPOS = 2, MSF_MINPOS = 4, MSF_DESPOS = 8,
  MSF_NORMOPSTAT = 16, MSF_MOTCRASHED = 40, MSF_NLINMOV = 88, MSF_LINMOV = 152,
  MSF_NOTVALID = 128 }
  status flags
- enum TSearchDir { DIR_POSITIVE, DIR_NEGATIVE }

12.16.1 Enumeration Type Documentation

12.16.1.1 enum TMotCmdFlg

command flags

Enumerator:

MCF_OFF set the motor off

MCF_CALIB calibrate

MCF_FREEZE freeze the motor

MCF_ON set the motor on

MCF_CLEAR_MOVEBUFFER clear the movebuffer

Definition at line 48 of file kmlMotBase.h.

12.16.1.2 enum TMotStsFlg

status flags

Enumerator:

MSF_MECHSTOP mechanical stopper reached, new: unused (default value)

MSF_MAXPOS max. position was reached, new: unused

MSF_MINPOS min. position was reached, new: calibrating

MSF_DESPOS in desired position, new: fixed, state holding

MSF_NORMOPSTAT trying to follow target, new: moving (polymb not full)

MSF_MOTCRASHED motor has crashed, new: collision

MSF_NLINMOV non-linear movement ended, new: poly move finished

MSF_LINMOV linear movement ended, new: moving poly, polymb full

MSF_NOTVALID motor data not valid

Definition at line 58 of file kmlMotBase.h.

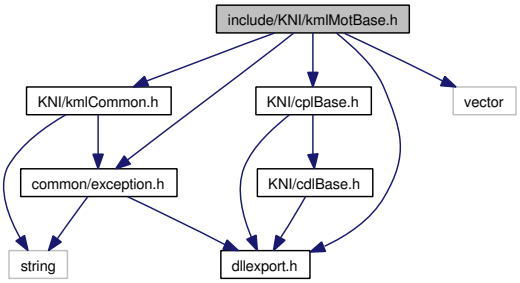
12.16.1.3 enum TSearchDir

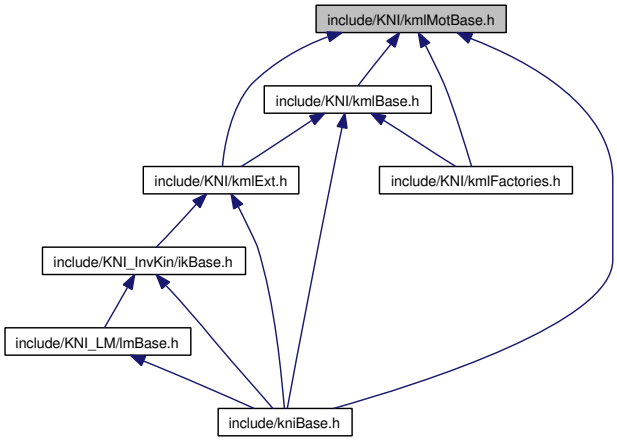
Enumerator:

DIR_POSITIVE search direction for the meachanical stopper

DIR_NEGATIVE

Definition at line 69 of file kmlMotBase.h.


include/KNI/kmlMotBase.h
KNI/kmlCommon.h
KNI/cplBase.h
vector
common/exception.h
KNI/cdlBase.h
string
dllexport.h


include/KNI/kmlMotBase.h
include/KNI/kmlBase.h
include/KNI/kmlExt.h
include/KNI/kmlFactories.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.17 include/KNI/kmlSctBase.h File Reference

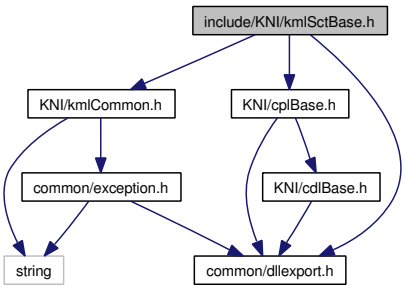
```
#include "common/dllexport.h"
#include "KNI/kmlCommon.h"
#include "KNI/cplBase.h"
```

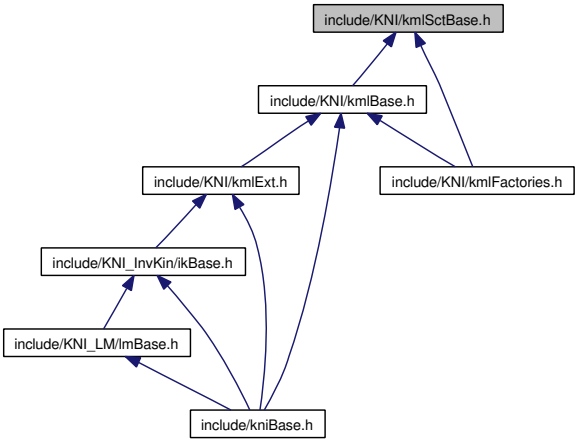
Include dependency graph for kmlSctBase.h:

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

Classes

- struct TSctDesc
  sensor controller description (partly)
- struct TKatSCT
  [SCT] every sens ctrl's attributes
- struct TSctGNL
  [GNL] controller generals
- struct TSctDAT
  [DAT] sensor data
- class CSctBase
  Sensor Controller class.


include/KNI/kmlSctBase.h
KNI/kmlCommon.h
KNI/cplBase.h
common/exception.h
KNI/cdlBase.h
string
common/dllexport.h


include/KNI/kmlSctBase.h
include/KNI/kmlBase.h
include/KNI/kmlExt.h
include/KNI/kmlFactories.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.18 include/KNI_InvKin/ikBase.h File Reference

```
#include "common/exception.h"
#include "common/dllexport.h"
#include "KNI/kmlExt.h"
#include "KNI/kmlCommon.h"
#include "KNI_InvKin/KatanaKinematics.h"
#include "KNI_InvKin/KatanaKinematics6M90G.h"
#include "KNI_InvKin/KatanaKinematics6M90T.h"
#include "KNI_InvKin/KatanaKinematics6M180.h"
#include "KNI_InvKin/KatanaKinematics5M180.h"
#include <vector>
#include <memory>
#include <cmath>
```

Include dependency graph for ikBase.h:

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

Classes

- class CikBase

Defines

- #define TM_ENDLESS -1

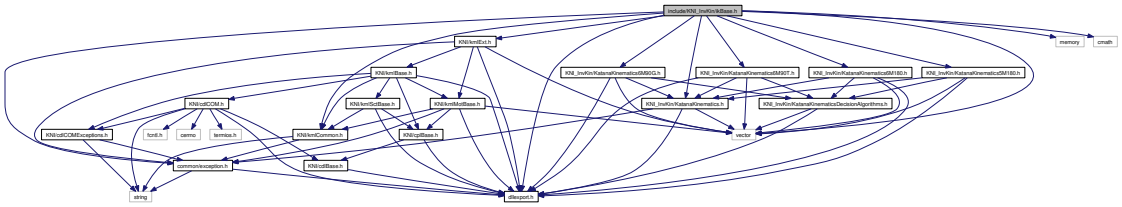
  timeout symbol for 'endless' waiting

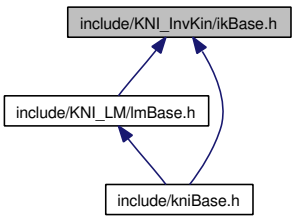
12.18.1 Define Documentation

12.18.1.1 #define TM_ENDLESS -1

timeout symbol for 'endless' waiting

Definition at line 45 of file ikBase.h.




include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.19 include/KNI_InvKin/KatanaKinematics.h File Reference

```
#include "common/dllexport.h"
#include "common/exception.h"
#include <vector>
```

Include dependency graph for KatanaKinematics.h:

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

Namespaces

- namespace KNI

Classes

- class KNI::NoSolutionException

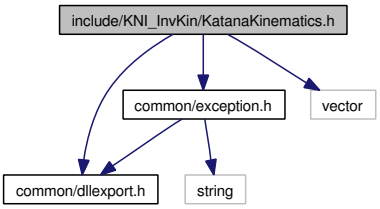
  No solution found for the given cartesian coordinates.

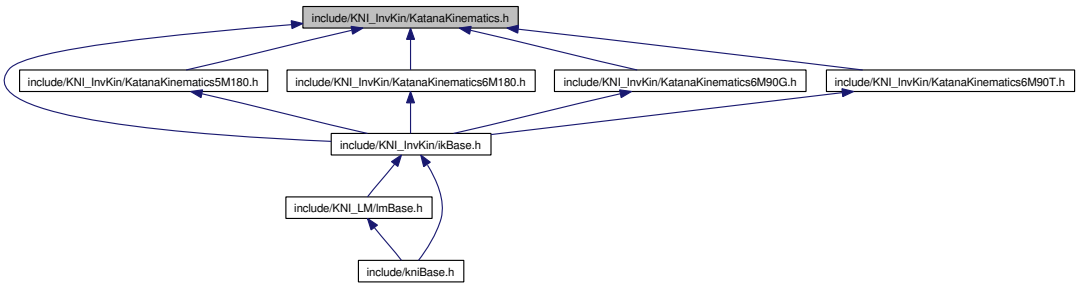
- struct KNI::KinematicParameters

  To pass different parameters for the kinematic implementations.

- class KNI::KatanaKinematics

  The base class for all kinematic implementations.


include/KNI_InvKin/KatanaKinematics.h
common/exception.h
vector
common/dllexport.h
string


include/KNI_InvKin/KatanaKinematics.h
include/KNI_InvKin/KatanaKinematics5M180.h
include/KNI_InvKin/KatanaKinematics6M180.h
include/KNI_InvKin/KatanaKinematics6M90G.h
include/KNI_InvKin/KatanaKinematics6M90T.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.20 include/KNI_InvKin/KatanaKinematics5M180.h File Reference

```
#include "common/dllexport.h"
#include "KNI_InvKin/KatanaKinematics.h"
#include "KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h"
#include <vector>
```

Include dependency graph for KatanaKinematics5M180.h:

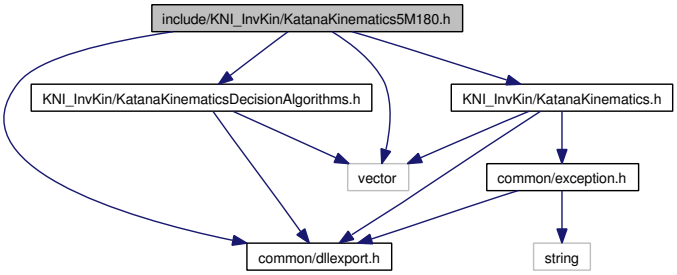
This graph shows which files directly or indirectly include this file:

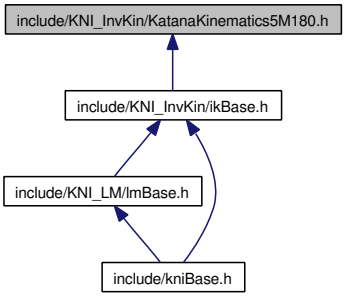
Namespaces

- namespace KNI

Classes

- class KNI::KatanaKinematics5M180
- struct KNI::KatanaKinematics5M180::position
- struct KNI::KatanaKinematics5M180::angles_calc


include/KNI_InvKin/KatanaKinematics5M180.h
KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h
KNI_InvKin/KatanaKinematics.h
vector
common/exception.h
common/dllexport.h
string


include/KNI_InvKin/KatanaKinematics5M180.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.21 include/KNI_InvKin/KatanaKinematics6M180.h File Reference

```
#include "common/dllexport.h"
#include "KNI_InvKin/KatanaKinematics.h"
#include "KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h"
#include <vector>
```

Include dependency graph for KatanaKinematics6M180.h:

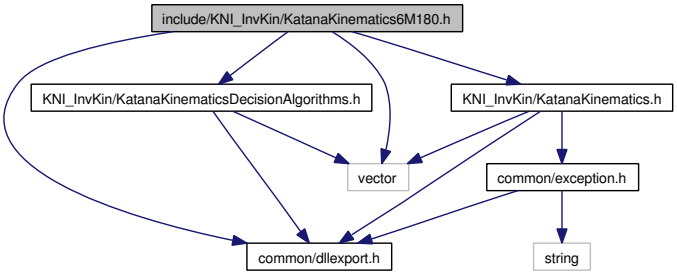
This graph shows which files directly or indirectly include this file:

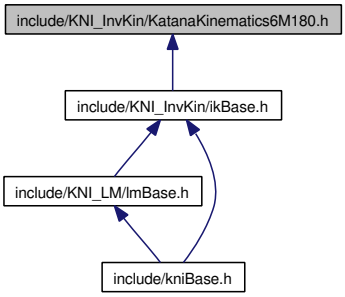
Namespaces

- namespace KNI

Classes

- class KNI::KatanaKinematics6M180
- struct KNI::KatanaKinematics6M180::position
- struct KNI::KatanaKinematics6M180::angles_calc


include/KNI_InvKin/KatanaKinematics6M180.h
KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h
KNI_InvKin/KatanaKinematics.h
vector
common/exception.h
common/dllexport.h
string


include/KNI_InvKin/KatanaKinematics6M180.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.22 include/KNI_InvKin/KatanaKinematics6M90G.h File Reference

```
#include "common/dllexport.h"
#include "KNI_InvKin/KatanaKinematics.h"
#include "KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h"
#include <vector>
```

Include dependency graph for KatanaKinematics6M90G.h:

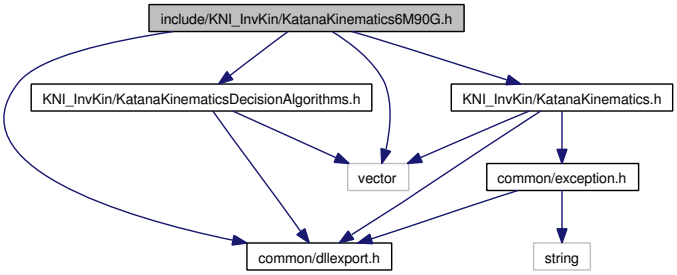
This graph shows which files directly or indirectly include this file:

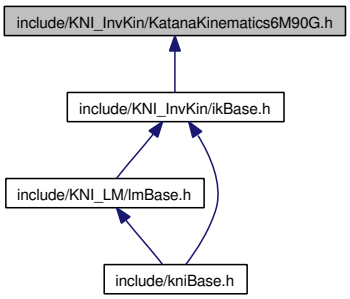
Namespaces

- namespace KNI

Classes

- class KNI::KatanaKinematics6M90G
- struct KNI::KatanaKinematics6M90G::position
- struct KNI::KatanaKinematics6M90G::angles_calc


include/KNI_InvKin/KatanaKinematics6M90G.h
KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h
KNI_InvKin/KatanaKinematics.h
vector
common/exception.h
common/dllexport.h
string


include/KNI_InvKin/KatanaKinematics6M90G.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.23 include/KNI_InvKin/KatanaKinematics6M90T.h File Reference

```
#include "common/dllexport.h"
#include "KNI_InvKin/KatanaKinematics.h"
#include "KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h"
#include <vector>
```

Include dependency graph for KatanaKinematics6M90T.h:

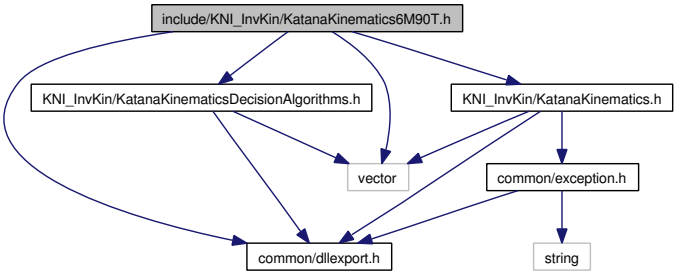
This graph shows which files directly or indirectly include this file:

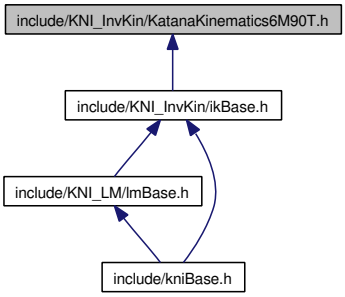
Namespaces

- namespace KNI

Classes

- class KNI::KatanaKinematics6M90T
- struct KNI::KatanaKinematics6M90T::position
- struct KNI::KatanaKinematics6M90T::angles_calc


include/KNI_InvKin/KatanaKinematics6M90T.h
KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h
KNI_InvKin/KatanaKinematics.h
vector
common/exception.h
common/dllexport.h
string


include/KNI_InvKin/KatanaKinematics6M90T.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.24 include/KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h File Reference

```
#include "common/dllexport.h"
#include <vector>
```

Include dependency graph for KatanaKinematicsDecisionAlgorithms.h:

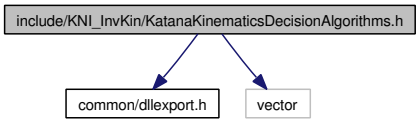
This graph shows which files directly or indirectly include this file:

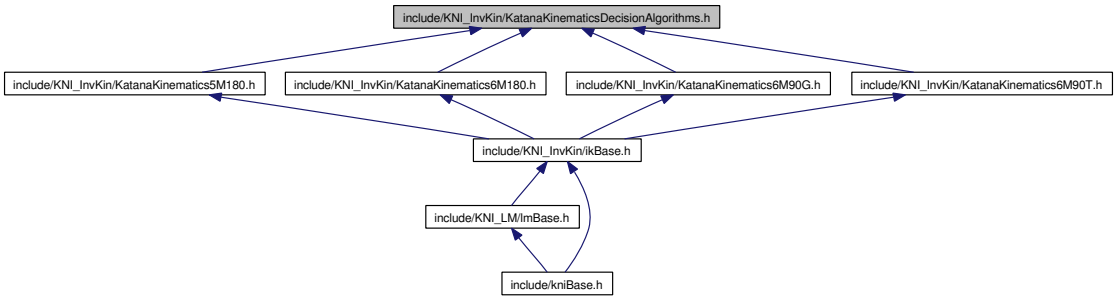
Namespaces

- namespace KNI

Classes

- struct KNI::KinematicsDefaultEncMinAlgorithm


include/KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h
common/dllexport.h
vector


include/KNI_InvKin/KatanaKinematicsDecisionAlgorithms.h
include/KNI_InvKin/KatanaKinematics5M180.h
include/KNI_InvKin/KatanaKinematics6M180.h
include/KNI_InvKin/KatanaKinematics6M90G.h
include/KNI_InvKin/KatanaKinematics6M90T.h
include/KNI_InvKin/ikBase.h
include/KNI_LM/lmBase.h
include/kniBase.h

12.25 include/KNI_LM/lmBase.h File Reference

```
#include "common/dllexport.h"
#include "common/Timer.h"
#include "KNI_InvKin/ikBase.h"
#include "common/exception.h"
#include <vector>
#include <cmath>
#include "boost/numeric/ublas/matrix.hpp"
#include "boost/numeric/ublas/vector.hpp"
#include "boost/numeric/ublas/vector_proxy.hpp"
#include "boost/numeric/ublas/triangular.hpp"
#include "boost/numeric/ublas/lu.hpp"
#include "boost/numeric/ublas/io.hpp"
```

Include dependency graph for lmBase.h:

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

Classes

- class JointSpeedException

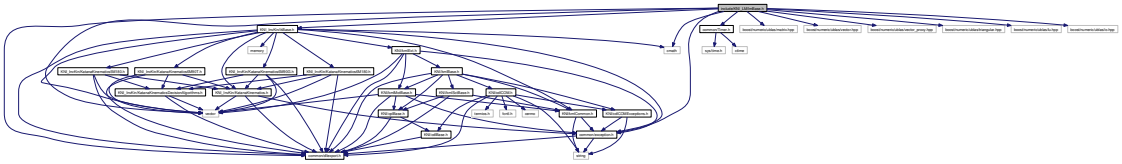
  Joint speed too high.

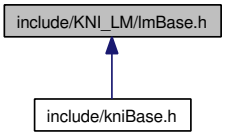
- class WaitParameterException

  Wait parameter set to false.

- class CLMBase

  Linear movement Class.




include/KNI_LM/lmBase.h
include/kniBase.h

12.26 include/kni_wrapper/kni_wrapper.h File Reference

Classes

- struct TPos

  extern C because we want to access these interfaces from anywhere:

- struct TMovement

  structure for the

- struct TCurrentMot

  structure for the currently active axis

Defines

- #define DLLEXPORT

Enumerations

- enum { ERR_NONE = 0, ERR_SUCCESS = 1 }
- enum ETransition { PTP = 1, LINEAR = 2 }

  the transition types for a movement

Functions

- DLLEXPORT int allMotorsOff ()

  Switches all axes off.

- DLLEXPORT int allMotorsOn ()

  Put all axes into hold state.

- DLLEXPORT int calibrate (int axis)

  closes the Katana session

- DLLEXPORT int clearMoveBuffers ()

  clears the movebuffers

- DLLEXPORT int closeGripper ()

  closes the gripper if available

- DLLEXPORT int deleteMovementFromStack (char ∗name, int index)

  deletes a movement from the stack

- DLLEXPORT int deleteMovementStack (char ∗name)

  deletes a movemnt stack

- DLLEXPORT int executeConnectedMovement (struct TMovement ∗movement, struct TPos ∗startPos, bool first, bool last)

execute a connected movement

- DLLEXPORT int executeMovement (struct TMovement *movement)
  execute a movement
- DLLEXPORT int flushMoveBuffers ()
  flush all the movebuffers
- DLLEXPORT int getAxisFirmwareVersion (int axis, char value[])
  gets the axis firmware version and returns it in the value argument
- DLLEXPORT int getDrive (int axis, int &value)
  gets the pwm and returns it in the value argument
- DLLEXPORT int getEncoder (int axis, int &value)
  gets the position and returns it in the value argument
- DLLEXPORT int getNumberOfMotors ()
  returns the number of motors configured
- DLLEXPORT int getPosition (struct TPos *pos)
  gets a position
- DLLEXPORT int getVelocity (int axis, int &value)
  gets the velocity and returns it in the value argument
- DLLEXPORT int getVersion (char value[])
  gets the controlboard firmware version and returns it in the value argument
- DLLEXPORT int initKatana (char *configFile, char *ipaddress)
  This initializes the Katana (communication etc).
- DLLEXPORT int IO_readInput (int inputNr, int &value)
  reads an input from the digital I/O and returns it in the value argument
- DLLEXPORT int IO_setOutput (char output, int value)
  sets an output of the digital I/Os
- DLLEXPORT int ModBusTCP_readWord (int address, int &value)
  reads a value from the register 'address' (if not connected, connect to the IP in katana.conf)
- DLLEXPORT int ModBusTCP_writeWord (int address, int value)
  writes a value to the register 'address' (if not connected, connect to the IP in katana.conf)
- DLLEXPORT int motorOff (int axis)
  Switches an axis off.
- DLLEXPORT int motorOn (int axis)
  Switches an axis on.
- DLLEXPORT int moveMot (int axis, int enc, int speed, int accel)
  PTP movement.

- DLLEXPORT int moveMotAndWait (int axis, int targetpos, int tolerance)
  calls MoveMot() and WaitForMot()
- DLLEXPORT int moveToPos (struct TPos *pos, int velocity, int acceleration)
  moves in IK
- DLLEXPORT int moveToPosEnc (int enc1, int enc2, int enc3, int enc4, int enc5, int enc6, int velocity, int acceleration, int tolerance, bool _wait)
  Moves all axes to a target encoder value.
- DLLEXPORT int moveToPosLin (struct TPos *targetPos, int velocity, int acceleration)
  moves in LM
- DLLEXPORT int openGripper ()
  opens the gripper if available
- DLLEXPORT int ping (int axis)
  checks the alive state of an axis
- DLLEXPORT int pushMovementToStack (struct TMovement *movement, char *name)
  pushes a movement onto a stack
- DLLEXPORT int runThroughMovementStack (char *name, int loops)
  Runs through the movement stack, executes the movements.
- DLLEXPORT int sendSplineToMotor (int axis, int targetpos, int duration, int p0, int p1, int p2, int p3)
  sends a single polynomial to an axis (G)
- DLLEXPORT int setCollisionDetection (int axis, bool state)
  sets the collision detection on the axes.
- DLLEXPORT int setCollisionParameters (int axis, int position, int velocity)
  sets the collision parameters this function internally calls setPositionCollisionLimit and setVelocityCollisionLimit
- DLLEXPORT int setControllerParameters (int axis, int ki, int kspeed, int kpos)
  sets the controller parameters
- DLLEXPORT int setGripper (bool hasGripper)
  sets or unsets whether the Katana has a Gripper
- DLLEXPORT int setMaxAccel (int axis, int acceleration)
  sets the maximum acceleration (allowed values are only 1 and 2)
- DLLEXPORT int setMaxVelocity (int axis, int vel)
  sets the maximum velocity
- DLLEXPORT int setPositionCollisionLimit (int axis, int limit)
  set the position collision limit
- DLLEXPORT int setVelocityCollisionLimit (int axis, int limit)
  set the velocity collision limit

- DLLEXPORT int setForceLimit (int axis, int limit)
  set the force limit
- DLLEXPORT int getForce (int axis)
  set the current force
- DLLEXPORT int getCurrentControllerType (int axis)
  set the current controller limit
- DLLEXPORT int startSplineMovement (int contd, int exactflag)
  starts the linear movement (G+128)
- DLLEXPORT int unblock ()
  unblocks the robot after collision/instantstop
- DLLEXPORT int waitForMot (int axis, int targetpos, int tolerance)
  waits for a motor

Variables

- const double PI = 3.14159265358979323846

12.26.1 Define Documentation

12.26.1.1 #define DLLEXPORT

Definition at line 30 of file kni_wrapper.h.

12.26.2 Enumeration Type Documentation

12.26.2.1 anonymous enum

Enumerator:

ERR_NONE

ERR_SUCCESS

Definition at line 36 of file kni_wrapper.h.

12.26.2.2 enum ETransition

the transition types for a movement

Enumerator:

PTP Point-to-point movement.

LINEAR linear movement

Definition at line 56 of file kni_wrapper.h.

12.26.3 Function Documentation

12.26.3.1 DLLEXPORT int allMotorsOff ()

Switches all axes off.

Returns:

returns -1 on failure, 1 if successful

12.26.3.2 DLLEXPORT int allMotorsOn ()

Put all axes into hold state.

Returns:

returns -1 on failure, 1 if successful

12.26.3.3 DLLEXPORT int calibrate (int *axis*)

closes the Katana session

Returns:

returns -1 on failure, 1 if successful

12.26.3.4 DLLEXPORT int clearMoveBuffers ()

clears the movebuffers

12.26.3.5 DLLEXPORT int closeGripper ()

closes the gripper if available

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.6 DLLEXPORT int deleteMovementFromStack (char * *name*, int *index*)

deletes a movement from the stack

Parameters:

name the name of the stack

index the index of the movement to delete

12.26.3.7 DLLEXPORT int deleteMovementStack (char * *name*)

deletes a movemnt stack

12.26.3.8 DLLEXPORT int executeConnectedMovement (struct TMovement * *movement*, struct TPos * *startPos*, bool *first*, bool *last*)

execute a connected movement

Parameters:

movement a TMovement struct to execute

startPos a TPos struct with the start position, can be omitted if first=true

first if this is the first of the connected movements (start at current pos)

last if this is the last of the connected movements (wait for end of move)

12.26.3.9 DLLEXPORT int executeMovement (struct TMovement * *movement*)

execute a movement

Parameters:

movement a TMovement struct to execute, starting from the current position

12.26.3.10 DLLEXPORT int flushMoveBuffers ()

flush all the movebuffers

12.26.3.11 DLLEXPORT int getAxisFirmwareVersion (int *axis*, char *value*[ ])

gets the axis firmware version and returns it in the value argument

12.26.3.12 DLLEXPORT int getCurrentControllerType (int *axis*)

set the current controller limit

Returns:

0 for position controller, 1 for current controller

12.26.3.13 DLLEXPORT int getDrive (int *axis*, int & *value*)

gets the pwm and returns it in the value argument

Parameters:

axis The axis to send the command to

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.14 DLLEXPORT int getEncoder (int *axis*, int & *value*)

gets the position and returns it in the value argument

Parameters:

axis The axis to send the command to

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.15 DLLEXPORT int getForce (int *axis*)

set the current force

12.26.3.16 DLLEXPORT int getNumberOfMotors ()

returns the number of motors configured

12.26.3.17 DLLEXPORT int getPosition (struct TPos * *pos*)

gets a position

12.26.3.18 DLLEXPORT int getVelocity (int *axis*, int & *value*)

gets the velocity and returns it in the value argument

Parameters:

axis The axis to send the command to

Returns:

returns -1 on failure, 1 if successful

12.26.3.19 DLLEXPORT int getVersion (char *value*[ ])

gets the controlboard firmware version and returns it in the value argument

12.26.3.20 DLLEXPORT int initKatana (char * *configFile*, char * *ipaddress*)

This initializes the Katana (communication etc).

12.26.3.21 DLLEXPORT int IO_readInput (int *inputNr*, int & *value*)

reads an input from the digital I/O and returns it in the value argument

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.22 DLLEXPORT int IO_setOutput (char *output*, int *value*)

sets an output of the digital I/Os

Parameters:

ouputNr 1, or 2 for OutA or OutB

Returns:

returns -1 on failure, 1 if successful

12.26.3.23 DLLEXPORT int ModBusTCP_readWord (int *address*, int & *value*)

reads a value from the register 'address' (if not connected, connect to the IP in katana.conf)

Returns:

returns -1 on failure, the read value if successful

12.26.3.24 DLLEXPORT int ModBusTCP_writeWord (int *address*, int *value*)

writes a value to the register 'address' (if not connected, connect to the IP in katana.conf)

Returns:

returns -1 on failure, 1 if successful

12.26.3.25 DLLEXPORT int motorOff (int *axis*)

Switches an axis off.

Parameters:

axis The axis to send the command to

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.26 DLLEXPORT int motorOn (int *axis*)

Switches an axis on.

Parameters:

axis The axis to send the command to

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.27 DLLEXPORT int moveMot (int *axis*, int *enc*, int *speed*, int *accel*)

PTP movement.

Parameters:

axis The axis to send the command to

enc the target position

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.28 DLLEXPORT int moveMotAndWait (int *axis*, int *targetpos*, int *tolerance*)

calls MoveMot() and WaitForMot()

Parameters:

axis The axis to send the command to

tolerance in encoder values (0 means wait until reached)

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, ERR_RANGE_MISMATCH if the target position is out of range 1 if successful

12.26.3.29 DLLEXPORT int moveToPos (struct TPos * *pos*, int *velocity*, int *acceleration*)

moves in IK

12.26.3.30 DLLEXPORT int moveToPosEnc (int *enc1*, int *enc2*, int *enc3*, int *enc4*, int *enc5*, int *enc6*, int *velocity*, int *acceleration*, int *tolerance*, bool *_wait*)

Moves all axes to a target encoder value.

Parameters:

enc (encX) the target positions

tolerance in encoders. sent unscaled to axis and handled there. WaitForMot (and MoveMOtAndWait) checks the tolerance though.

wait wait for the movement to finish

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.31 DLLEXPORT int moveToPosLin (struct TPos * *targetPos*, int *velocity*, int *acceleration*)

moves in LM

12.26.3.32 DLLEXPORT int openGripper ()

opens the gripper if available

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.33 DLLEXPORT int ping (int *axis*)

checks the alive state of an axis

Parameters:

axis 0 = get all axes

Returns:

If axis 0: 1 if all axes are present, negative value is the inverted number of the first axis found failing, 0 if no data is available. If axis != 0: 1 if heartbeat found, -1 if failed, 0 if no data is available.

12.26.3.34 DLLEXPORT int pushMovementToStack (struct TMovement * *movement*, char * *name*)

pushes a movement onto a stack

Parameters:

movement a movement structure filled with position and movement parameters

name the name of the stack to push it onto

12.26.3.35 DLLEXPORT int runThroughMovementStack (char * *name*, int *loops*)

Runs through the movement stack, executes the movements.

Parameters:

name the name of the stack to run through

loops the number of loops to run

12.26.3.36 DLLEXPORT int sendSplineToMotor (int *axis*, int *targetpos*, int *duration*, int *p0*, int *p1*, int *p2*, int *p3*)

sends a single polynomial to an axis (G)

Parameters:

axis The axis to send the command to

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, ERR_RANGE_MISMATCH if the target position is out of range 1 if successful

12.26.3.37 DLLEXPORT int setCollisionDetection (int *axis*, bool *state*)

sets the collision detection on the axes.

Parameters:

state true = on

axis 0 = set all axes

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.38 DLLEXPORT int setCollisionParameters (int *axis*, int *position*, int *velocity*)

sets the collision parameters this function internally calls setPositionCollisionLimit and setVelocityCollision-Limit

Parameters:

axis 0 = set all axes

position range 1-10

velocity range 1-10

12.26.3.39 DLLEXPORT int setControllerParameters (int *axis*, int *ki*, int *kspeed*, int *kpos*)

sets the controller parameters

Parameters:

axis 0 = set all axes

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.40 DLLEXPORT int setForceLimit (int *axis*, int *limit*)

set the force limit

Parameters:

axis 0 = all axes

limit limit in percent

12.26.3.41 DLLEXPORT int setGripper (bool *hasGripper*)

sets or unsets whether the Katana has a Gripper

Parameters:

hasGripper set to true if a gripper is present. Default at startup: false

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.42 DLLEXPORT int setMaxAccel (int *axis*, int *acceleration*)

sets the maximum acceleration (allowed values are only 1 and 2)

Parameters:

axis 0 = set all axes

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.43 DLLEXPORT int setMaxVelocity (int *axis*, int *vel*)

sets the maximum velocity

Parameters:

axis 0 = set all axes

vel 1-180 are valid

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, 1 if successful

12.26.3.44 DLLEXPORT int setPositionCollisionLimit (int *axis*, int *limit*)

set the position collision limit

Parameters:

axis 0 = all axes

12.26.3.45 DLLEXPORT int setVelocityCollisionLimit (int *axis*, int *limit*)

set the velocity collision limit

Parameters:

axis 0 = all axes

12.26.3.46 DLLEXPORT int startSplineMovement (int *contd*, int *exactflag*)

starts the linear movement (G+128)

Returns:

returns -1 on failure, 1 if successful

12.26.3.47 DLLEXPORT int unblock ()

unblocks the robot after collision/instantstop

12.26.3.48 DLLEXPORT int waitForMot (int *axis*, int *targetpos*, int *tolerance*)

waits for a motor

Parameters:

axis The axis to wait for

targetpos (only relevant if mode is 0)

tolerance (only relevant if mode is 0) in encoder values

Returns:

returns ERR_FAILURE on failure, ERR_INVALID_ARGUMENT if an argument is out of range, ERR_-STATE_MISMATCH if the command was given to a wrong state, ERR_RANGE_MISMATCH if the target position is out of range 1 if successful

12.26.4 Variable Documentation

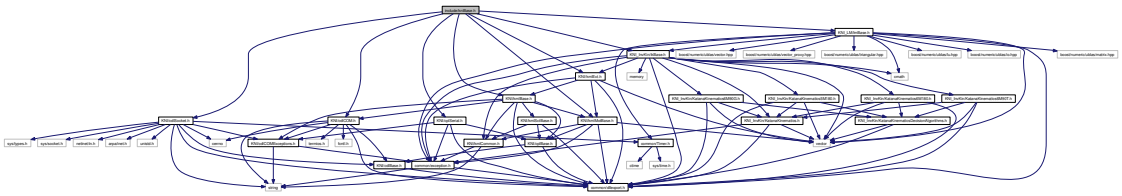
12.26.4.1 const double PI = 3.14159265358979323846

Definition at line 34 of file kni_wrapper.h.

12.27 include/kniBase.h File Reference

```
#include "KNI/cdlCOM.h"
#include "KNI/cplSerial.h"
#include "KNI/kmlBase.h"
#include "KNI/kmlExt.h"
#include "KNI/cdlSocket.h"
#include "KNI_InvKin/ikBase.h"
#include "KNI_LM/lmBase.h"
#include "KNI/kmlMotBase.h"
```

Include dependency graph for kniBase.h:



12.28 include/libKinematics.h File Reference

Classes

- struct IntVector
- struct FloatVector

Defines

- #define _declspec(dllexport) int kin_setType(int type)

  This sets the robot type.

Variables

- const int MaxDof = 10

  Maximum degree of freedom.

- FloatVector * d
- FloatVector FloatVector * a
- FloatVector FloatVector FloatVector * alpha
- FloatVector FloatVector FloatVector int typeNr = -1)
- FloatVector * angleMDH
- FloatVector * angleK4D
- FloatVector * angle
- IntVector * enc
- FloatVector * pose
- FloatVector * prev
- FloatVector FloatVector int maxBisection = 0)

12.28.1 Define Documentation

12.28.1.1 _declspec(dllexport)

This sets the robot type.

This calculates the inverse kinematics.

This calculates the direct kinematics.

This converts angles in mDH convention to encoders.

This converts encoders to angles in mDH convention.

This converts angles from mDH to Katana4D convention.

This converts angles from Katana4D to mDH convention.

This cleans the kinematics library.

This initializes the kinematic.

Get the version number of the library.

Get the tcp offset.

Get the angle range.

Get the angle offsets.

Get the rotation direction.

Get the encoder offsets.

Get the encoders per cycle.

Get the immobile flag of the last joint.

Get the modified Denavit-Hartenberg parameters.

Get the degree of mobility.

Get the degree of freedom.

Get the maximum degree of freedom.

Get the robot type.

This sets the tcp offset.

This sets the angle range.

This sets the angle offsets.

This sets the rotation direction.

This sets the encoder offsets.

This sets the encoders per cycle.

This sets the immobile flag of the last joint.

This sets the link length parameters (part of mDH-parameters).

This sets the modified Denavit-Hartenberg parameters.

Setting robot type includes setting mDH parameters, immobile flag, degree of freedom (dof), degree of mobility (dom) and angles.

type dof dom 6M90A_F 6 6 6M90A_G 6 5 6M180 5 5 6M90B_F 6 6 6M90B_G 6 5

Parameters:

type 0: 6M90A_F, 1: 6M90A_G, 2: 6M180, 3: 6M90B_F, 4: 6M90B_G

Returns:

0 if successful, $<$ 0 if failed

Transformation from previous to current link: move a along (previous) x-axis, rotate alpha about (previous) x-axis, move d along (current) z-axis, rotate theta about (current) z-axis. Setting the mDH parameters includes setting dof, adjusting immobile flag (all free) and dom (= dof).

The length of the vectors need to be the same.

Parameters:

theta angle about z-axis

d distance along x-axis

a distance previous to current along previous x-axis

alpha angle about previous x-axis

typeNr 0: 6M90A_F, 1: 6M90A_G, 2: 6M180, 3: 6M90B_F, 4: 6M90B_G, -1: other

Returns:

0 if successful, $<$ 0 if failed

Setting link length parameters requires the type to be set to 0 (6M90A_F), 1 (6M90A_G), 2 (6M180), 3 (6M90B_F) or 4 (6M90B_G). This is done using the setType or setModifiedDH function that set the mDH parameters so they can be adjusted. The length of the vector has to be four.

Parameters:

links length of the links in m

Returns:

0 if successful, $<$ 0 if failed

Flag for the last joint if it is free (0) or locked (1). Setting the immobile flag includes adjusting dom. Type or mDH parameters have to be set.

Parameters:

immobile last joint immobile flag: 0 for free, 1 for locked (immobile)

Returns:

0 if successful, $<$ 0 if failed

Number of encoders in a full cycle of each joint. Length of epc needs to be equal to DOM set with Type or mDH parameters.

Parameters:

epc encoders pec cycle

Returns:

0 if successful, $<$ 0 if failed

The encoder values at the calibration stops. Length of encOffset needs to be equal to DOM set with Type or mDH parameters.

Parameters:

encOffset encoder values at calibration stops

Returns:

0 if successful, $<$ 0 if failed

The rotation direction of the joints: +1 if encoders and angles (mDH convention) grow in same direction, -1 if encoders and angles grow in opposit direction. Length of rotDir needs to be equal to DOM set with Type or mDH parameters.

Parameters:

rotDir rotation direction of the joints

Returns:

0 if successful, $<$ 0 if failed

The angles at the calibration stops in mDH convention! Length of angleOffset needs to be equal to DOM set with Type or mDH parameters.

Parameters:

angleOffset angle values at calibration stops in mDH

Returns:

0 if successful, $<$ 0 if failed

The angle range from angle offset to angle stop in mDH convention (negative if angleOffset $>$ angleStop)! Length of angleRange needs to be equal to DOM set with Type or mDH parameters.

Parameters:

angleRange angle range from angle offset to angle stop

Returns:

0 if successful, $<$ 0 if failed

Offset from the flange to the effective tcp.

Parameters:

tcpOffset (x, y, z, psi) offset in m and rad respectively where psi is rotation about x-axis of tool coordinate system

Returns:

0 if successful, $<$ 0 if failed

Get the robot type

Returns:

0 for 6M90A_F, 1 for 6M90A_G, 2 for 6M180, 3 for 6M90B_F, 4 for 6M90B_G, -1 for other or not set yet

Get the maximum degree of freedom (length of array in the vectors)

Returns:

Maximum degree of freedom

Get the degree of freedom

Returns:

Degree of freedom or -1 if not set yet

Get the degree of mobility

Returns:

Degree of mobility or -1 if not set yet

Parameters:

theta vector to write in angle about z-axis

d vector to write in distance along x-axis

a vector to write in distance previous to current along previous x-axis

alpha vector to write in angle about previous x-axis

Returns:

0 if successful, $<$ 0 if failed
0 if mobile, 1 if immobile, $<$ 0 if failed

Parameters:

epc vector to write in encoders pec cycle

Returns:

0 if successful, $<$ 0 if failed

Parameters:

encOffset vector to write in encoder values at calibration stops

Returns:

0 if successful, $<$ 0 if failed

Parameters:

rotDir vector to write in rotation direction of the joints

Returns:

0 if successful, $<$ 0 if failed

Parameters:

angleOffset vector to write in angle values at calibration stops in mDH

Returns:

0 if successful, $<$ 0 if failed

Parameters:

angleRange vector to write in angle range from angle offset to angle stop

Returns:

0 if successful, $<$ 0 if failed

Parameters:

tcpOffset vector to write in (x, y, z, psi) offset

Returns:

0 if successful, $<$ 0 if failed

Parameters:

version vector to write in version (major, minor, revision)

Returns:

0 if successful, $<$ 0 if failed

Initialize the kinematic. Parameters mDH, dof, angle offset and angle range (SetType or SetMDH and SetAngleOffset and SetAngleRange) have to be set prior to initialization of the kinematic!

Returns:

0 if successful, $<$ 0 if failed

Free memory allocated in setType() / setMDH() before closing the library

Returns:

0 if successful, $<$ 0 if failed

Length of angleK4D needs to be equal to DOM set with Type or mDH parameters.

Parameters:

angleK4D angle in K4D convention

angleMDH angle in mDH convention

Returns:

0 if successful, $<$ 0 if failed

Length of angleMDH needs to be equal to DOM set with Type or mDH parameters.

Parameters:

angleMDH angle in mDH convention

angleK4D angle in K4D convention

Returns:

0 if successful, $<$ 0 if failed

Length of enc needs to be equal to DOM set with Type or mDH parameters.

Parameters:

enc encoders

angle angles in mDH convention

Returns:

0 if successful, $<$ 0 if failed

Length of angle needs to be equal to DOM set with Type or mDH parameters.

Parameters:

angle angle in mDH convention

enc encoders

Returns:

0 if successful, $<$ 0 if failed

Length of angle needs to be equal to DOM set with Type or mDH parameters.

Parameters:

angle angles in mDH convention

pose pose of the TCP [x, y, z, phi, theta, psi]

Returns:

0 if successful, $<$ 0 if failed

Length of prev needs to be equal to DOM set with Type or mDH parameters.

Parameters:

pose pose of the TCP [x, y, z, phi, theta, psi]

prev previous angles (starting point)

angle angles in mDH convention

maxBisection maximum number of bisections done, if no solution found

Returns:

0 if successful, $<$ 0 if failed

Definition at line 19 of file libKinematics.h.

12.28.2 Variable Documentation

12.28.2.1 FloatVector FloatVector * a

Definition at line 78 of file libKinematics.h.

12.28.2.2 FloatVector FloatVector FloatVector * alpha

Definition at line 78 of file libKinematics.h.

12.28.2.3 FloatVector FloatVector * angle

Definition at line 323 of file libKinematics.h.

12.28.2.4 FloatVector* angleK4D

Definition at line 313 of file libKinematics.h.

12.28.2.5 FloatVector* angleMDH

Definition at line 303 of file libKinematics.h.

12.28.2.6 FloatVector * d

Definition at line 78 of file libKinematics.h.

12.28.2.7 IntVector* enc

Definition at line 333 of file libKinematics.h.

12.28.2.8 FloatVector FloatVector int maxBisection = 0)

Definition at line 356 of file libKinematics.h.

12.28.2.9 const int MaxDof = 10

Maximum degree of freedom.

Definition at line 28 of file libKinematics.h.

12.28.2.10 FloatVector* pose

Definition at line 343 of file libKinematics.h.

12.28.2.11 FloatVector* prev

Definition at line 355 of file libKinematics.h.

12.28.2.12 FloatVector FloatVector FloatVector int typeNr = -1)

Definition at line 79 of file libKinematics.h.