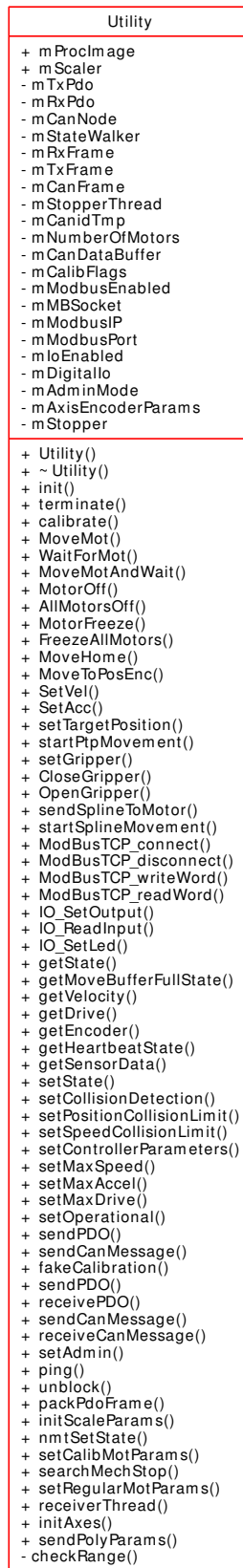


4.27 Katana Class Reference

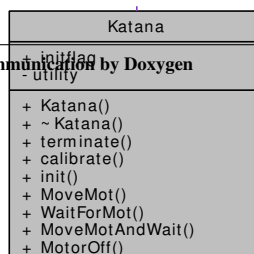
Python communication interface for [Katana 1.2](#).

```
#include <katana.h>
```

Collaboration diagram for Katana:



↑ utility



Public Member Functions

- [Katana](#) ()
constructor.
- [~Katana](#) ()
destructor
- [int terminate](#) ()
closes the [Katana](#), all the RT threads and the RT environment
- [int calibrate](#) (int axis, bool force)
calibrates the [Katana](#) (at startup)
- [int init](#) ()
initializes the communication & RT
- [int MoveMot](#) (int axis, int enc)
PTP movement.
- [int WaitForMot](#) (int axis, int targetpos, int tolerance, int mode)
waits until the axis is back in hold state
- [int MoveMotAndWait](#) (int axis, int targetpos, int tolerance)
calls [MoveMot\(\)](#) and [WaitForMot\(\)](#)
- [int MotorOff](#) (int axis)
Switches an axis off.
- [int AllMotorsOff](#) ()
Switches all axes off.
- [int MotorFreeze](#) (int axis)
Freezes an axis.
- [int FreezeAllMotors](#) ()
Put all axes into hold state.
- [int MoveHome](#) ()
Moves into the home position.
- [int MoveToPosEnc](#) (int enc1, int enc2, int enc3, int enc4, int enc5, int enc6, int velocity, int acceleration, int tolerance)
Moves all axes to a target encoder value.
- [int SetVel](#) (int axis, int value)
sets the velocity
- [int SetAcc](#) (int axis, int value)
sets the acceleration

- int `setGripper` (bool hasGripper)
sets or unsets whether the [Katana](#) has a Gripper
- int `CloseGripper` ()
closes the gripper if available
- int `OpenGripper` ()
opens the gripper if available
- int `sendSplineToMotor` (int axis, int targetpos, int duration, int p0, int p1, int p2, int p3)
sends a single polynomial to an axis (G)
- int `startSplineMovement` (int contd, int exactflag)
starts the linear movement (G+I28)
- int `ModBusTCP_connect` (char *hostIP, int timeout=1000, int port=502, int outputOffset=0)
connects to the modbusTCP server
- int `ModBusTCP_disconnect` ()
disconnects from the modbusTCP server
- int `ModBusTCP_writeWord` (int address, int value)
writes a value to the register 'address'
- int `ModBusTCP_readWord` (int address)
reads a value from the register 'address'
- int `IO_SetOutput` (int outputNr, int value)
sets an output of the digital I/Os
- int `IO_ReadInput` (int inputNr)
reads an input from the digital I/O
- int `IO_SetLed` (char state)
sets the LED on the back panel
- int `getState` (int axis)
gets the axis state
- int `getMoveBufferFullState` (int axis)
returns the fill state of the movebuffer
- int `getVelocity` (int axis)
gets the velocity
- int `getDrive` (int axis)
gets the pwm
- int `getEncoder` (int axis)

gets the position

- `int getHeartbeatState` (int axis)
gets the heartbeat state of an axis
- `int getSensorData` (int sensor, int channel)
Gets the sensor data.
- `int setState` (int axis, int state)
puts the axis into the specified state
- `int setCollisionDetection` (int axis, bool state)
sets the collision detection on the axes.
- `int setPositionCollisionLimit` (int axis, int limit)
set the position collision limit axis 0 = all axes
- `int setSpeedCollisionLimit` (int axis, int threshold, int treshold_lin)
set the speed collision limit axis 0 = all axes
- `int setControllerParameters` (int axis, int ki, int kspeed, int kpos)
sets the controller parameters
- `int setMaxSpeed` (int axis, int value)
sets the max.
- `int setMaxAccel` (int axis, int value)
sets the max.
- `int setMaxDrive` (int axis, int value)
sets the PWM
- `int setOperational` (int axis)
Puts the axis into operational state.
- `int setTargetPosition` (int axis, int encoder)
sets the target encoder
- `int startPtpMovement` (int axis)
starts a PTP movement
- `int sendPDO` (int pdoNo, int axis, int length, char *data)
This sends a PDO.
- `int sendCanMessage` (int canId, int length, char *data)
This sends a raw CAN message.
- `int fakeCalibration` (int axis)
Sets encoder offsets without actually calibrating. For axes without HW stop.

- int [sendPDO](#) (struct [Pdo](#) &pdo)
NOT part of the K4D interface.
- int [receivePDO](#) (struct [Pdo](#) &pdo)
NOT part of the K4D interface.
- int [sendCanMessage](#) (struct can_frame &frame)
NOT part of the K4D interface.
- int [receiveCanMessage](#) (struct can_frame &frame, int timeout=20000)
NOT part of the K4D interface.
- int [setAdmin](#) (int passCode)
This unlocks critical funktions.
- int [ping](#) (int axis)
checks the alive state of an axis
- void [unblock](#) ()
unblocks the robot after collision/instantstop

Public Attributes

- bool [initflag](#)
is set to false if init fails:

Private Attributes

- [Utility](#) * [utility](#)
Utility class.

4.27.1 Detailed Description

Python communication interface for [Katana](#) 1.2.

Exports calls for all [Katana](#) Axis communication to Python (after swigging) Needed for standalone capability

Definition at line 28 of file katana.h.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 [Katana::Katana](#) ()

constructor.

This calls [init\(\)](#) on the [Katana](#) and [Utility](#) classes. May fail depending on the config file

4.27.2.2 **Katana::~~Katana ()**

destructor

4.27.3 Member Function Documentation

4.27.3.1 **int Katana::terminate ()**

closes the [Katana](#), all the RT threads and the RT environment

Returns:

returns -1 on failure, 1 if successful

4.27.3.2 **int Katana::calibrate (int *axis*, bool *force*)**

calibrates the [Katana](#) (at startup)

Parameters:

axis The axis to send the command to

force Force calibration even if already done

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

4.27.3.3 **int Katana::init ()**

initializes the communication & RT

Returns:

returns -1 on failure, 1 if successful

4.27.3.4 **int Katana::MoveMot (int *axis*, int *enc*)**

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

4.27.3.5 int Katana::WaitForMot (int axis, int targetpos, int tolerance, int mode)

waits until the axis is back in hold state

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

4.27.3.6 int Katana::MoveMotAndWait (int axis, int targetpos, int tolerance)

calls [MoveMot\(\)](#) and [WaitForMot\(\)](#)

Parameters:

axis The axis to send the command to

tolerance,: ignored if 0

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

4.27.3.7 int Katana::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

4.27.3.8 int Katana::AllMotorsOff ()

Switches all axes off.

Returns:

returns -1 on failure, 1 if successful

4.27.3.9 int Katana::MotorFreeze (int *axis*)

Freezes an axis.

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

4.27.3.10 int Katana::FreezeAllMotors ()

Put all axes into hold state.

Returns:

returns -1 on failure, 1 if successful

4.27.3.11 int Katana::MoveHome ()

Moves into the home 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

4.27.3.12 int Katana::MoveToPosEnc (int *enc1*, int *enc2*, int *enc3*, int *enc4*, int *enc5*, int *enc6*, int *velocity*, int *acceleration*, int *tolerance*)

Moves all axes to a target encoder value.

Parameters:

enc (encX) the target positions

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

4.27.3.13 int Katana::SetVel (int *axis*, int *value*)

sets the velocity

Parameters:

axis The axis to send the command to

value 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, ERR_RANGE_MISMATCH if the target position is out of range 1 if successful

4.27.3.14 int Katana::SetAcc (int axis, int value)

sets the acceleration

Parameters:

axis The axis to send the command to

value 1-4 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

4.27.3.15 int Katana::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

4.27.3.16 int Katana::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

4.27.3.17 int Katana::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

4.27.3.18 int Katana::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 argumant is out of range, ERR_STATE_MISMATCH if the commeand was given to a wrong state, ERR_RANGE_MISMATCH if the target position is out of range 1 if successful

4.27.3.19 int Katana::startSplineMovement (int contd, int exactflag)

starts the linear movement (G+128)

Returns:

returns -1 on failure, 1 if successful

4.27.3.20 int Katana::ModBusTCP_connect (char * hostIP, int timeout = 1000, int port = 502, int outputOffset = 0)

connects to the modbusTCP server

Returns:

returns -1 on failure, 1 if successful

4.27.3.21 int Katana::ModBusTCP_disconnect ()

disconnects from the modbusTCP server

Returns:

returns -1 on failure, 1 if successful

4.27.3.22 int Katana::ModBusTCP_writeWord (int address, int value)

writes a value to the register 'address'

Returns:

returns -1 on failure, 1 if successful

4.27.3.23 int Katana::ModBusTCP_readWord (int address)

reads a value from the register 'address'

Returns:

returns -1 on failure, 1 if successful

4.27.3.24 int Katana::IO_SetOutput (int outputNr, int value)

sets an output of the digital I/Os

Returns:

returns -1 on failure, 1 if successful

4.27.3.25 int Katana::IO_ReadInput (int inputNr)

reads an input from the digital I/O

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

4.27.3.26 int Katana::IO_SetLed (char state)

sets the LED on the back panel

Parameters:

state 'r' for red, 'g' for green, 'o' for off

Returns:

returns -1 on failure, 1 if successful

4.27.3.27 int Katana::getState (int axis)

gets the axis state

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

4.27.3.28 int Katana::getMoveBufferFullState (int axis)

returns the fill state of the movebuffer

Parameters:

axis The axis to send the command to

Returns:

0 if not full, 1 if full 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

4.27.3.29 int Katana::getVelocity (int axis)

gets the velocity

Parameters:

axis The axis to send the command to

Returns:

returns -1 on failure, 1 if successful

4.27.3.30 int Katana::getDrive (int axis)

gets the pwm

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

4.27.3.31 int Katana::getEncoder (int axis)

gets the position

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

4.27.3.32 int Katana::getHeartbeatState (int axis)

gets the heartbeat 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.

4.27.3.33 int Katana::getSensorData (int sensor, int channel)

Gets the sensor data.

4.27.3.34 int Katana::setState (int axis, int state)

puts the axis into the specified state

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

4.27.3.35 int Katana::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

4.27.3.36 int Katana::setPositionCollisionLimit (int axis, int limit)

set the position collision limit axis 0 = all axes

4.27.3.37 int Katana::setSpeedCollisionLimit (int axis, int threshold, int threshold_lin)

set the speed collision limit axis 0 = all axes

4.27.3.38 int Katana::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

4.27.3.39 int Katana::setMaxSpeed (int axis, int value)

sets the max.

speed

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

4.27.3.40 int Katana::setMaxAccel (int axis, int value)

sets the max.

acceleration

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

4.27.3.41 int Katana::setMaxDrive (int axis, int value)

sets the PWM

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

4.27.3.42 int Katana::setOperational (int axis)

Puts the axis into operational state.

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

4.27.3.43 int Katana::setTargetPosition (int axis, int encoder)

sets the target encoder

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

4.27.3.44 int Katana::startPtpMovement (int axis)

starts a PTP movement

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

4.27.3.45 int Katana::sendPDO (int pdoNo, int axis, int length, char * data)

This sends a [PDO](#).

Parameters:

axis The axis to send the command to

Returns:

returns -1 on failure, 1 if successful

4.27.3.46 int Katana::sendCanMessage (int *canId*, int *length*, char * *data*)

This sends a raw CAN message.

Returns:

returns -1 on failure, 1 if successful

4.27.3.47 int Katana::fakeCalibration (int *axis*)

Sets encoder offsets without actually calibrating. For axes without HW stop.

4.27.3.48 int Katana::sendPDO (struct [Pdo](#) & *pdo*)

NOT part of the K4D interface.

This sends a [PDO](#)

Returns:

returns -1 on failure, 1 if successful

4.27.3.49 int Katana::receivePDO (struct [Pdo](#) & *pdo*)

NOT part of the K4D interface.

Receives a [PDO](#)

Returns:

returns -1 on failure, 1 if successful

4.27.3.50 int Katana::sendCanMessage (struct [can_frame](#) & *frame*)

NOT part of the K4D interface.

Sends a raw CAN message

Returns:

returns -1 on failure, 1 if successful

4.27.3.51 int Katana::receiveCanMessage (struct [can_frame](#) & *frame*, int *timeout* = 20000)

NOT part of the K4D interface.

receive a raw CAN message

Returns:

returns -1 on failure, 1 if successful

4.27.3.52 `int Katana::setAdmin (int passCode)`

This unlocks critical funktions.

Returns:

returns -1 on failure, 1 if successful

4.27.3.53 `int Katana::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.

4.27.3.54 `void Katana::unblock ()`

unblocks the robot after collision/instantstop

4.27.4 Member Data Documentation

4.27.4.1 `Utility*` `Katana::utility` [private]

`Utility` class.

Definition at line 31 of file `katana.h`.

4.27.4.2 `bool` `Katana::initflag`

is set to false if init fails:

Definition at line 35 of file `katana.h`.

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

- `inc/pykatana/katana.h`