

Programmer's Guide PlatformCtrl

```

E:\User\Ivy\PlatformCtrl\PlatformCtrl_win_2265.exe
PrjPlatformCtrlMain.cpp      70   13:10:35   05:12:2006   Platform Control
version 2.265, Build 20060627 123   13:10:35   05:12:2006   image grabber is
IhndOppPltfCtrl.cpp          120   13:10:35   05:12:2006   available
IhndMotionPltfCtrl.cpp       120   13:10:35   05:12:2006   init pose read fr
on StartupPosition.dat, x:0  y:0  a:1.284
IhndCompPltfCtrl.cpp         57    13:10:35   05:12:2006   server started at
port 10002
IhndCompPltfCtrl.cpp         57    13:10:35   05:12:2006   server started at
port 10000
IhndMotionPltfCtrl.cpp       390    13:10:35   05:12:2006   load map: Platfor
m/Map/PltfMap.map
IhndMotionPltfCtrl.cpp       413    13:10:35   05:12:2006   load roadmap: Pla
tform/Roadmap/PltfRoadmap.rm
IhndMotionPltfCtrl.cpp       446    13:10:35   05:12:2006   system state: ini
tialized
  
```

Contents

1 Introduction.....	4
1.1 Contents of this document.....	4
1.2 Format of this document.....	4
2 Software Architecture.....	5
3 Installing the host computer.....	6
3.1 Windows operating system.....	6
Installing the Java Runtime Environment.....	6
Installing the GUI.....	6
3.2 Linux operating system.....	6
Installing the Java Runtime Environment.....	6
Installing the GUI.....	6
3.3 Setting up the network.....	6
4 PlatformCtrl.....	7
4.1 Features.....	7
4.2 Operation	7
4.3 Motion Modes.....	8
Mode No Motion.....	8
Mode Joystick (Hardware).....	8
Mode Joystick (Software).....	8
Mode Velocity.....	8
Mode Automatic.....	9
4.4 Platform configuration.....	9
4.5 Python task interpreter.....	10
5 PlatformCtrlGUI.....	11
5.1 Features.....	11
5.2 Connecting to the Platform.....	11
5.3 Map tools.....	12
5.4 Creating a Map.....	12
5.5 Creating a Roadmap.....	14
5.6 Starting a Motion Mode.....	15
5.7 Setting View Options.....	16
5.8 Hardware monitor.....	17
Motor.....	17
IOBoard.....	17
RelayBoard.....	17
USBoard.....	17
Radar.....	17
CAN Messages.....	17
TCP Client.....	17
TCP Platform	17
5.9 Setting platform parameters.....	18
Surveillance.....	18
Tracking & Detection.....	18
Ultrasonic.....	18
Mapping.....	19
Motion Control.....	19
Planning.....	19
Collision.....	19
Geometry.....	19
Interpolation.....	20

Localization.....	20
6 Platform Socket API.....	21
6.1 Overview.....	21
6.2 Message Format.....	23
6.3 Commands.....	23
CAN Communication Commands.....	23
Platform Operation Commands.....	25
Request Status Commands.....	26
Motion Commands.....	26
Platform Data Commands.....	27
Special Functions Commands.....	30
Debug Commands.....	30
6.4 Reply Telegrams.....	31
6.5 XML-Documents.....	39
Map.....	39
Roadmap.....	41
Program.....	42
Task.....	42
7 Legal notes.....	49

1 Introduction

1.1 Contents of this document

This guide explains how to use both Neobotix' platform controller software and the graphical user interface. For researchers and customers with extraordinary applications, this document also contains detailed information on how to communicate and interact with the control software.

There are two possibilities to use the software: non-simulation and simulation mode. In simulation mode no platform hardware is necessary. The original platform controller software is running on the local computer and all sensor signals are simulated.

In detail you will learn how to:

- Start and initialise the Java user interface application "PlatformCtrlGUI" and the C++ controller application "PlatformCtrl"
- Establish a socket connection between these two programs
- Create a map
- Create a roadmap
- Move the robot within a simulated or its real-world environment

All these actions and most of the GUI's features are described in detail. So for most customers and applications this document should provide all the information necessary to set up the mobile platform.

If any further information, for example when doing research or installing very specialised applications, is required please contact Neobotix.

1.2 Format of this document

Once the hardware has been set up according to the operating manual or the technical description, the platform will operate very safely. Changing any settings in the GUI will not cause danger to the platform hardware.

Nevertheless, highest safety can only be achieved if all parameters are set correctly! To help setting up a safe and convenient application the following icons mark important paragraphs:



The warning triangle marks paragraphs that concern the danger of injuries, damages or both. Please make sure to read these paragraphs very carefully!



The light bulb marks paragraphs which deal with common problems, misunderstandings and errors and is meant to be a help in troubleshooting.



The black board is used to highlight examples which demonstrate extraordinary functions or difficulties.

2 Software Architecture

The layered architecture of the Neobotix software is shown in the figure below.

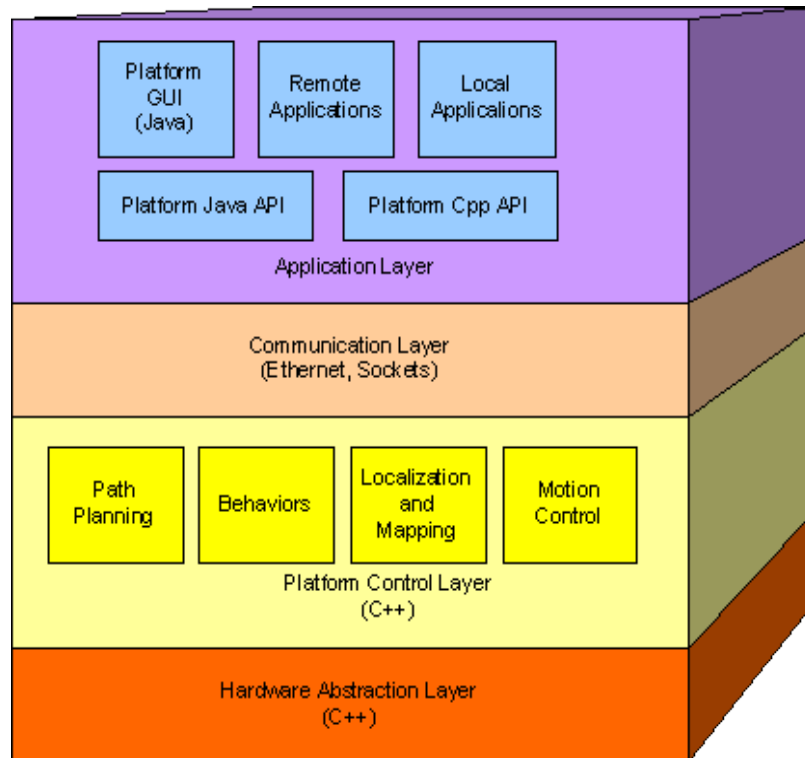


Figure 1: Architecture of the Neobotix Software Package

The software is highly modular and allows an easy adoption to the customers needs.

■ Hardware Abstraction Layer

A C++ driver library supports the Neobotix platform hardware and the integrated sensors and actors. The interface is open and available for Windows and Linux. The drivers for Linux are under the LGPL and need a kernel version 2.6.

■ Platform Control Layer

The robot platform controller is for efficient execution fully implemented in C++. The features comprise all navigation and control tasks needed for autonomous mobile robots.

■ Communication Layer

The platform interface of the platform control layer is mapped to socket commands. Additionally, all low-level sensor signals and actor commands are directly accessible by the platform socket interface, so that customers are able to integrate their own control software. By this way applications can be reside local on the platform PC or remote on any computer in the network. There is also no limitation towards the programming language or operating system.

■ Application Layer

The platform is ready to use with the graphical user interface. It is completely written in JAVA and delivers a comfortable interface to work with the robot platform. A customized application may directly use the socket interface to control the platform. But for applications written in JAVA or CPP the socket communication is already implemented and can be used by the specific platform API.

3 Installing the host computer

A common PC system is sufficient to run the Neobotix graphical user interface.

3.1 Windows operating system

Installing the Java Runtime Environment

The Neobotix GUI is programmed in Java and thus platform independent. This also means that a Java virtual machine and a library for 3D-calculations must be installed before using the GUI.

If there is none or an older version on the host computer use the files in the “Java” folder on the CD that was delivered together with the robot. Run both “.exe”-files and follow the instructions on the screen.

Installing the GUI

The graphical user interface does not need to be installed. Simply copy the folder “PltfGUI” from the CD to your hard drive.

To run the GUI just double-click on “start.bat” or create a shortcut on your desktop by right-dragging the “start.bat”-icon onto the desktop and selecting “Create shortcut here” from the pop-up-menu.

3.2 Linux operating system

Installing the Java Runtime Environment

Most common Linux distributions like Debian/Ubuntu or OpenSuse already come with an installed Java Runtime Environment. Please make sure that you installed Java 6 Runtime Environment from Sun Microsystems. For Debian/Ubuntu distributions installation starts after entering “apt-get install sun-java6-jre” on command line as root. For other distributions please use your package manager.

Installing the GUI

The graphical user interface does not need to be installed. Instead simply copy the folder “PltfGUI” from the CD to your hard drive.

To run the GUI open a terminal, change to the location of “PltfGUI” and enter “java -jar PlatformCtrlGUI.jar”

3.3 Setting up the network

If ordered, a wireless LAN-device was delivered together with the platform to allow easy connecting to the robot. In case the settings of this device need to be changed, please use the software on the according driver disc and work directly on the platform’s onboard computer as described in chapter “Maintenance – Software”.

The robot was set up with the network settings stated in “Configuration at delivery”. Make sure that both platform and host PC are in the same subnet. Please refer to the system administrator of the local computer network.

The last thing that has to be done after setting up the network is telling the GUI how to connect to the platform. Start the PlatformCtrlGUI and use the main menu to add a new address (*Socket* → *Open Socket At* → <<new>>). After clicking “OK” the program will immediately try to establish a connection to the platform. To reconnect to the robot chose either the last used address (*Socket* → *Open Socket* > *LastUsed (IP-address)*) or select one of the connections that are already known to the GUI (*Socket* → *Open Socket At* → *KnownAddress (IP-address)*).

4 PlatformCtrl

4.1 Features

The software program 'Neobotix Platform Control' (PlatformCtrl.exe) implements the hardware abstraction and the platform control layer. The features are:

- **Motion Control**

The interpolator generates smooth and continues movements. The automatic controller handles holonomic and non-holonomic platform kinematics and is highly accurate in path-tracking and posture stabilization.

- **Path-Planning**

A global planner finds the shortest path to the target position while a local planner smooths the path and ensures obstacle avoidance. The planning algorithms are efficient and applicable in dynamic environments.

- **Localization and Mapping**

The position of the platform is reliably estimated with stochastic methods. A powerful user interface exists for an easy installation.

- **Behaviours**

Numerous behaviours of the platform such as "move on a path", "move to target", "obstacle avoidance", etc. are coded in motion modes. They give a substantial basis for many applications. On request, behaviours can be extended and customized to specific applications. Additionally, all low-level sensor signals and actor commands are directly accessible by the platform socket interface, so that customers are able to integrate their own behaviours in the application layer.

- **Tasks**

The integrated python task interpreter can be used to realize applications. The task interpreter can be programmed with an easy-to-use editor.

- **Simulation**

The platform dynamics and all platform sensors can be fully simulated. The simulation gives a test environment for the implemented algorithms and applications

4.2 Operation

After the start of the program "*PlatformCtrl*" a socket server is established on the platform PC and listens for clients on the socket ports 10000 and 10002. For each port one client is able to connect to the platform.

All commands for the platform are received from the socket communication. They will cause a state transition depending on the state chart shown in the figure below.

The initialization process starts with the command `CMD_CONNECT_CAN` and will cause an initialization of the platform's CAN-bus and a test of the CAN communication to the several CAN nodes. The command `CMD_INIT_CAN` initializes all hardware components and executes a platform self test.



If a hardware emergency stop occurs, the platform switches into a secure state. After the safety system has been reset, the platform will enter the state 'Ready' again.

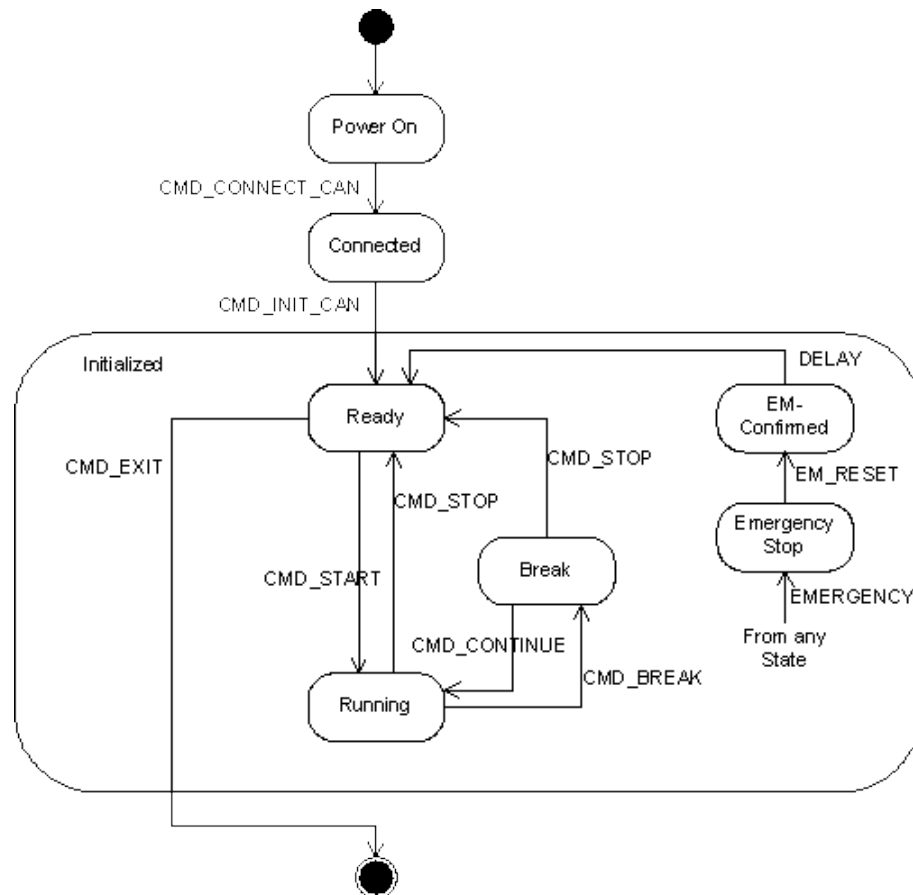


Figure 2: State chart of the operation states

In the state “Initialized” the platform can be started, stopped, paused or made to continue working on the last active command. A movement of the platform is only possible in the state ‘Running’ and its behaviour depends on the selected motion mode. For changing the motion mode the platform has to be in the state ‘Ready’.

4.3 Motion Modes

Mode No Motion

All move commands have no effects.

Mode Joystick (Hardware)

The platform is controlled with a wireless remote controller, which is included in delivery.

Mode Joystick (Software)

The desired translation and rotation velocities are set to the platform using the command `CMD_MOVE_VEL`.

Other move commands have no effect.

Mode Velocity

The desired velocities are set to the platform using the commands `CMD_MOVE_VEL_WHEEL` or `CMD_MOVE_VEL`.

The platform keeps the last set velocity until a new velocity value is set or a watchdog time expired. The watchdog time is chosen in the parameter file.

Mode Automatic

In the motion mode 'Automatic' several move commands are executed. If the target is reached, the platform stops and waits for a new move command. If a new move command is set while the platform is moving, the execution of the old command will be stopped immediately and the new command will be started.

CMD_MOVE_ABS:

The platform drives to a specified frame. If planning is disabled, the platform will drive directly on an interpolated path to the target. In front of obstacles the platform stops until the obstacle disappears again. If planning is enabled a shortest path to the target will be calculated. While driving the path will be smoothed and optimized. Obstacles will be recognized and the platform will move around them.

CMD_MOVE_PATH:

The platform drives on a specified path to the target. If planning is disabled the platform will first drive to the beginning of the path on the shortest possible way. With enabled planning the shortest path to the specified path is calculated. On the path the platform follows the path to the end and turns to the specified end orientation. If an obstacle appears while the platform is driving on the target path, the platform will stop and wait a few seconds. If the obstacle does not disappear, the platform will drive around the obstacle and the target path will be entered again at the next reachable place.

CMD_MOVE_TO_STATION:

For an application a roadmap can be defined. A roadmap specifies working stations and a network of connecting paths. The shortest path on the roadmap from the current platform position to the target station is calculated. Following the command CMD_MOVE_PATH is executed.



The behaviour of the several commands can be influenced by parameters set in the parameter file or by the command CMD_SET_PROPERTY. For a more detailed description refer to the Platform Socket API and the comments in the parameter files.

4.4 Platform configuration

The platform is configured with several parameter files:

```
CanCtrl.ini
CinneoE.ini
Interpolation.ini
Localization.ini
MotionCtrl.ini
Particlefilter.ini
UltraSonic.ini
Platform.ini
```

They are located in the folder '/SoftwareCtrl/bin/Platform'. The parameter files are read once at start-up and the values are valid until shutdown. Some selected parameters (called properties) can be set online using the Socket API. Follow the comments in the parameter files for a more detailed description of the parameters.

4.5 Python task interpreter

The task interpreter can be used to program applications of the platform. A task is a list of commands, which define actions or behaviours of the platform. The order of execution can be controlled by conditioned and unconditioned branch commands.

Usually, an application needs more than one task. For that, several tasks can be defined. The execution of the task will wait until the condition is satisfied.



The task execution has a local and a remote mode. In the local mode the program execution is done directly on the platform PC. In the remote mode separate computer executes the program and sends the socket commands to the platform via wireless Ethernet. This might be the case if a higher-level control software has to coordinate the mobile platform and other machines.

Further information and a list of all supported commands can be found in the “Python Client – Programmer’s Guide”.

5 PlatformCtrlGUI

5.1 Features

The software program Neobotix PlatformCtrlGUI is a Java application for controlling the Neobotix platform.

This chapter will cover the following topics:

- Visualization of the platform state and sensor data
- Using the tools for building a map of the environment
- Using the tools for building a roadmap for an application
- Control of the platform
- Configuration of selected platform parameters

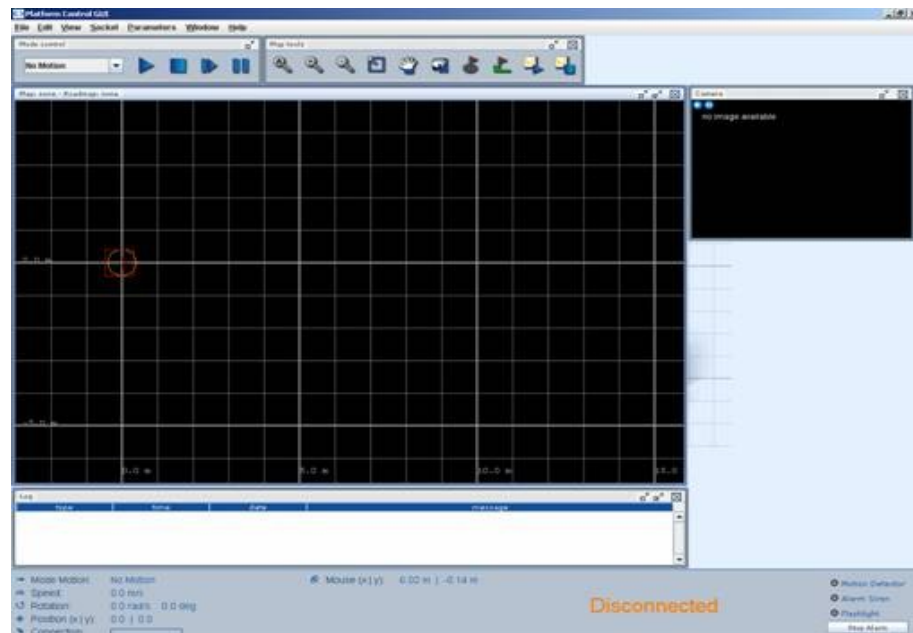


Figure 3: Snapshot of the GUI after start

5.2 Connecting to the Platform

To connect the GUI to the platform select the desired server in the server combo box. If the server is not listed in the combo box, extend the file 'PltfGUI\res\inifiles.Network.ini' with the missing server.

Then use the menu item 'Socket/Open Socket'.



Figure 4: Connecting to a socket

The socket connection to the platform will be established.

The controls have following meanings:

Open Socket	The GUI connects to the last connected PlatformControl.
Open Socket At	You can chose one of the server mentioned in 'PltfGUI\res\inifiles.Network.ini'.
Close Socket	The platform will be disconnected. The operation mode of the platform will not be affected. In order to reconnect the platform, use the item 'Open Socket'.
Shutdown Platform	Shutdown connected PlatformControl.

5.3 Map tools

The "Map Tools" toolbar is used to refresh, zoom, pan and set target positions for a platform on a map. The toolbar will appear when click "*View → Map*". The use of each tool is described below:



Refresh map.



Zoom all visible objects



Zoom in



Zoom out



Select the area and zoom.



Scroll the map. Right-click to finish.



The platform can be shifted by drag and drop. The frame is rotated with a right-click.



Selecting a target frame. Used in motion mode "*Automatic*", the platform will move to a specified location. When click on the tool, a red flag appears which can be shifted by drag and drop. Rotate the target direction with right click.



Selecting a target station. Used in motion mode "*Automatic*", the platform will move to a specified station. Select target station. The Robot will move along the edges to the selected station.

5.4 Creating a Map

To create a map open the menu item "*Edit → Edit Map*". The toolbar for editing the map will be opened. Tooltips of each tool are displayed at the bottom of the toolbar. For more details of how to use each tool, please refer to the PlatformCtrlGUI – Operating Manual.

The functions of the buttons are as follows:



Load a map from the hard disk of the PC hosting the Platform GUI program. For selecting a file a browser window will be opened.



Create a new map.



Save the current map to the local harddrive.



Save a map as a new file.



Load the current map from the platform into the map editor.



Send the current map of the editor to the platform. The map becomes active.



Clear the map.



Undo



Redo



Edit objects in the map by drag and drop.



Edit vertices of a polygon.



Delete objects on a map.



Add a line. Left-click to begin and drag to draw a line.



Add a reflector. Left-click to begin and drag to draw a reflector line.



Add a circle. Left-click to begin and drag to draw a circle.



Define a workspace for an application by drawing a polygon.



Define prohibited zones by drawing a polygon.



Insert a hidden Polygon for indicating special areas in the map. These objects have no effect on the platform.



Insert a label to indicate special areas for better understanding the map. Labels do not affect localization of the platform.



Trim a line in respect of another line. Click the line you want to trim and then select another line as a reference line, which will trim the first one.



Extend a line to reach another line. Click the line you want to extend, then click on a reference line that the first line will be extended to.



Make lines parallel. Click on the line you want to be parallel to the reference line and click the reference line to make the first selected line parallel to.



Make lines orthogonal. Click on the line you want to make orthogonal to another one. Click on another line as a reference line so that the first selected one is made orthogonal to the line.



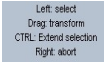
Invert direction of the line. The Scanner only recognizes the "front" of the line.



Select all objects in map.



Convert line from scan.



Brief guideline of how to use each tool is shown at the bottom of the toolbar.



Closes the toolbar and asks for saving if necessary.

An example of a map and an explanation of the graphical symbols are given in the following figure.



Figure 5: Example of objects on a map and meaning of each symbols

5.5 Creating a Roadmap

To create or edit a roadmap go to the menu item “*Edit → Edit Roadmap*”. The toolbar for editing the roadmap should appear. It looks similar to the toolbar of the map editor.



Load a map from the hard disk of the PC hosting the Platform GUI program. For selecting a file a browser window will be opened.



Make a new roadmap.



Save the current roadmap to harddisk.



Save a roadmap as...












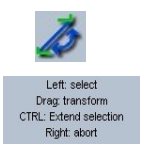

Load the current map from the platform into the map editor.



Send the current map of the editor to the platform. The map becomes active.



Clear the map.

	Undo
	Redo
	Edit objects in the map by drag and drop.
	Edit vertices of a polygon.
	Delete objects on a map.
	Insert a station. Renaming is possible by clicking on the name.
	Insert an edge between two stations. Attention: this is an one-way!
	Insert a bidirectional edge between two stations.
	Invert an unidirectional edge.
	A brief guideline of how to use each tool is shown at the bottom of the toolbar. Left: select Drag: transform CTRL: Extend selection Right: abort
	Close the toolbar and asks for saving if necessary.

An example of a roadmap is shown in the following figure. The edges and the path points specify the path.

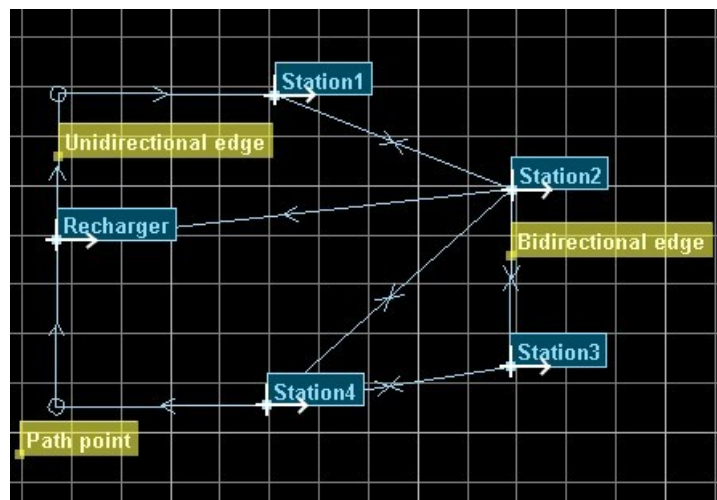


Figure 6: Example of stations and edges on a roadmap

5.6 Starting a Motion Mode

For starting a motion mode the platform has to be in the state "Ready". Select in to motion mode combo box the desired motion mode.

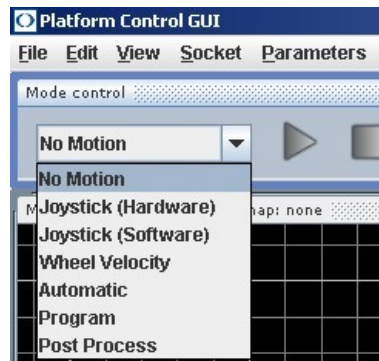






Figure 7: Motion mode menu

The motion modes are described in the PlatformCtrlGUI – Operating Manual.
The following buttons are used to control each motion mode:

-  Start motion
-  Stop motion
-  Continue motion
-  Break motion

5.7 Setting View Options

You can change view options such as switching the language English/German, hiding grid or showing only certain objects on a map using the menu “View Options”

To change the options, select the menu item “View → View Options” to open the window. For each selected category an entry mask appears on the right side of the window as shown in the figure below.

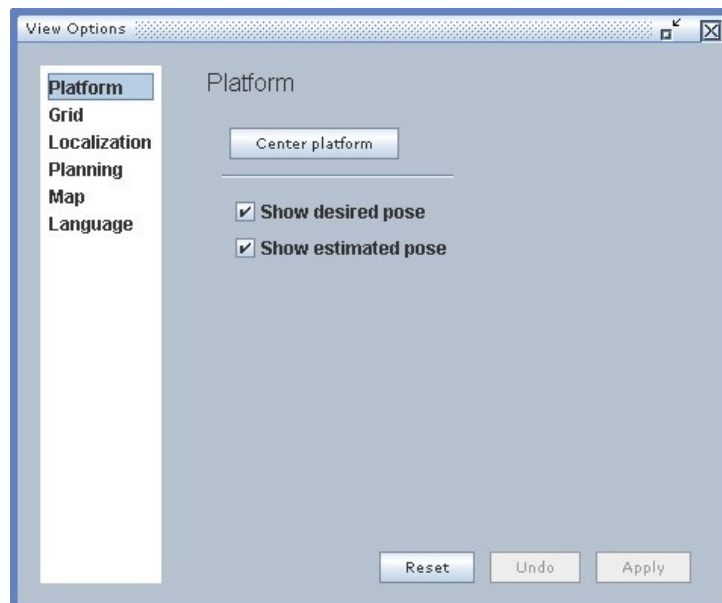


Figure 8: View options window

When changing the language, the GUI must be restarted to apply the change. You will be asked to close the program after clicking “Apply”.

5.8 Hardware monitor

Select the menu item “View → Hardware Monitor” to open the hardware control window. Each tab controls a CAN-node or visualizes hardware information.

Motor

The panel allows the user to monitor the condition of the left and right motors of the platform and configure them such as enabling and disabling brakes.

IOBoard

The IOBoard is a general purpose interface. The 16 channels of “Digital in” and 8 channels of “Analog in” are for general purpose and therefore optional. “Joy X” and “Joy Y” functions are obsoleted as they are now implemented on the Relay-board, however kept for compatibility.

RelayBoard

The panel displays the condition of the platform's Relay-board. Users can configure the board in this tab.

The functions “Relay out” and “Digital in” are for general purpose and optional. In surveillance robots, the second channel of “Relay out” is attached to a camera. The “Keypad” is associated with the keypad on the platform.

USBoard

The USBoard panel shows the status of 16 sensors on the board. These can be enabled and disabled individually through the command “start US”. The “Analog in” shows the status of passive infrared sensors of robots with surveillance mode. Otherwise, they are optional.

Radar

The panel display the readings of the radar sensors.

CAN Messages

The CAN-message-panel is not implemented at the moment.

TCP Client

The panel displays the status of the data transfer of the client.

TCP Platform

The panel displays the status of the data transfer of the platform.

5.9 Setting platform parameters

Select the menu item “*Parameter → Platform parameters*” to view the property panel of the platform.

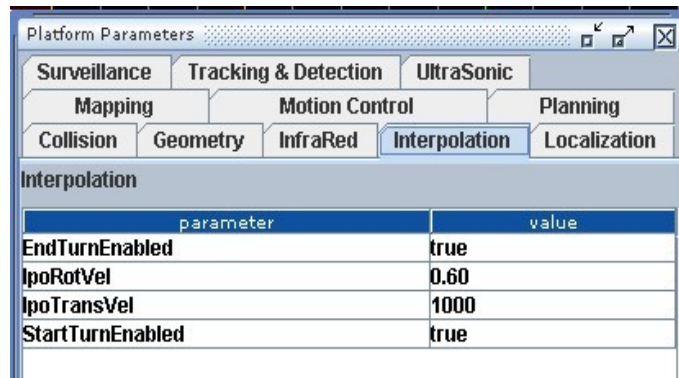


Figure 9: Platform parameters window: Interpolation

A property can be selected by a left-click and edit in an additional text field. The button “Set” transmits the value to the platform.

The definitions of each parameter is as described in the following chapters.

Surveillance

Parameter	Description	Value
FlashlightEnabled	Turning on or off the flashlight on a platform.	true: on false: off
PIRSensorsEnabled	Enable PIR Sensor.	true: enable false: disable
RadarSensorsEnabled	Enable radar sensor.	true: enable false: disable
SirenEnabled	Enable sirens	true: enable false: disable
SurveillanceModeEnabled	Enable the surveillance mode	true: enable false: disable

Tracking & Detection

Parameter	Description	Value
ChargeStationDetectEnabled	Enable function to detect a charge station.	true: enable false: disable
PersonDetectionEnabled	Enable function to detect a person.	true: enable false: disable
TrackingEnabled	Enable tracking of moved objects with the scanner.	true: enable false: disable

Ultrasonic

Configuration of the ultrasonic sensors. These values are used for safety purposes and therefore should not be changed.

Each ultrasonic sensor has four parameters:

- X-coordinate with respect to the platform frame in [mm]

- Y-coordinate with respect to the platform frame in [mm]
- Z-coordinate with respect to the platform frame in [mm]
- Angle with respect to x-axis in [degree]

Mapping

Parameter	Description	Value
MinLineLenAux	Minimum length of auxiliary lines [mm]: lines extracted from scan.	-
MinLineLenMap	Minimum length of mapped lines [mm]	-

Motion Control

These parameters affect the platform's movements. Note that the parameters "PathFollowParam1" and "PathFollowParam2" are correction values and should not be changed.

Parameter	Description	Value
PathFollowParam1	First parameter for path following function. (correction value)	-
PathFollowParam2	Second parameter for path following function. (correction value)	-
PositionStabilizationEnabled	Enable posture stabilization at the end of a move command.	true: enable false: disable

Planning

Parameter	Description	Value
Planning Enabled	Enable planning in the mode automatic	true: enable false: disable
SimuOnlineEnabledPostDis	Enable online simulation	true: enable false: disable
ViewPlanObstEnabled	View the obstacles used for planning (red lines in the platform GUI).	true: enable false: disable

Collision

Parameter	Description	Value
CollCheckEnabled	Enable the collision check with all sensors	true: enable false: disable
CollLaserMinDist	Minimum distance to the obstacle [mm]	-

Geometry

This tab displays the dimensions of the platform that are used for collision avoidance and path planning. These values are hardware dependent and should not be changed.

Parameter	Description	Value
DistPltfCenterToWheelAxis	Distance of the center of the platform to its wheel axis. [mm]	-
DistWheels	Distance between wheels [mm]	-

PltfLength	Length of the platform [mm]	-
RadiusWheel	Wheel radius of the platform [mm]	-

Interpolation

Parameter	Description	Value
EndTurnEnabled	Allows the platform to turn on the target position to face the target direction. Otherwise the platform will drive a curve.	true: enable false: disable
IpoRotvel	Rotational velocity of the interpolated frame in [rad/s].	Default: 0.6 rad/s
IpoTransVel	Translational velocity of the interpolated frame [mm/s]. If there is no obstacle and a path is straight, the platform will accelerate up to the specified velocity.	Default: 1000 mm/s
StartTurnEnabled	In the mode automatic with disabled planning the path is interpolated with a Bezier-curve. Enable this function to make the platform turn first into the direction of the path before driving to the target.	true: enable false: disable

Localization

Parameter	Description	Value
GlobalLocalization	Makes the platform use the whole map to localize its possible position depending on the scan.	true: enable false: disable
LineExtractMaxCov	Maximum uncertainty of extracted lines	
LocalLocalization	Enable the platform to correct its possible position using the laser scanner.	true: enable false: disable
MappingMode	Not used at the moment.	

6 Platform Socket API

6.1 Overview

The platform control software provides an interface based on sockets. This interface accepts commands and returns data only on request.

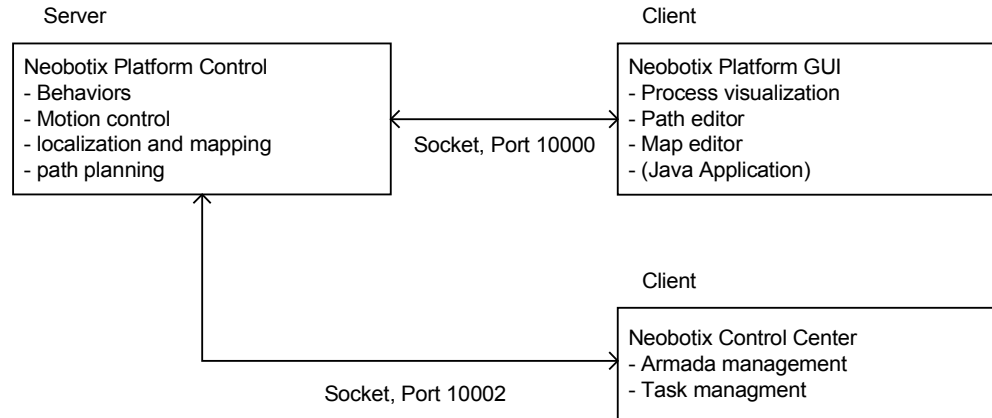


Figure 10: Command and data exchange between server and clients

After the start of 'PlatformCtrl' a socket server is established on the platform PC and listens for clients on the socket ports 10000 and 10002. For each port one client is able to connect to the platform.

An overview of the supported commands is given in the following table:

Command	Description	Page
CAN Communication Commands		
CMD_CONNECT_TO_PLTF	Connect all CAN-nodes of the platform to the main controller	23
CMD_SHUTDOWN_PLTF	Disconnect the CAN nodes	23
CMD_INIT_PLTF	Initialize the CAN-nodes	24
CMD_SET_BRAKE	Change the state of the drive brakes	24
CMD_SET_DIGOUT	Set the values of the digital output	24
CMD_START_ULTRASONIC	Start a periodic measurement of the ultrasonic sensors	24
CMD_STOP_ULTRASONIC	Stop the ultrasonic measurement	24
CMD_ADD_CAN_LISTENING_ID	Add a CAN-ID for listing	24
CMD_REMOVE_CAN_LISTENING_ID	Remove a CAN-ID from the listening list	24
CMD_SEND_CAN_MESSAGE	Send a CAN message	25
CMD_GET_CAN_MESSAGES	Request received CAN-messages	25
Platform Operation Commands		
CMD_START	Start a motion mode	25
CMD_BREAK	Break motion	25
CMD_CONTINUE	Continue motion after a break	25
CMD_STOP	Stop a motion mode	25
CMD_SET_MODE	Change a motion mode	25
CMD_EXIT	Shuts the platform down	26
CMD_CLOSE_SOCKET	Close the socket connection	26
Request Status Commands		
CMD_GET_STATUS	Request the status of the platform	26
CMD_GET_POSITION	Request Position and Velocity	26

Motion Commands		
CMD_MOVE_PATH	Move on a specified path to a target	26
CMD_MOVE_TO_STATION	Move to a working station	26
CMD_MOVE_ABS	Move to a frame given in world coordinates	26
CMD_MOVE_REL	Move to a frame given in platform coordinates	27
CMD_MOVE_VEL_WHEEL	Move with a specified velocity of the drive wheels	27
CMD_MOVE_VEL	Move with a specified translation and rotation velocity	27
Platform Data Commands		
CMD_SET_MAP	Set an active map	27
CMD_GET_MAP	Request to send the active map	27
CMD_LOAD_MAP	Load a map from the hard disk of the platform	27
CMD_SAVE_MAP	Save a map to the hard disk of the platform	27
CMD_MAPPING_START	Start the platform's mapping mode	28
CMD_MAPPING_RESET	Reset mapping	28
CMD_GET_MAPPING_STOP	Stop the platform's mapping mode	28
CMD_SET_ROADMAP	Set an active roadmap	28
CMD_GET_ROADMAP	Request to send an active roadmap	28
CMD_LOAD_ROADMAP	Load a roadmap from the hard disk of the platform	28
CMD_SAVE_ROADMAP	Save a roadmap to the hard disk of the platform	28
CMD_SET_FRM_PLTF	Set the estimated frame of the platform	28
CMD_SET_PLTF_PARAM	Change a parameter value	29
CMD_GET_PLTF_PARAM	Request to send the actual value of a single parameter	29
CMD_GET_PLTF_PARAM_VEC	Request to send the complete parameter list	29
CMD_SET_PROGRAM	Set a program of the task interpreter	29
CMD_GET_PROGRAM	Request to send the actual program	29
CMD_LOAD_PROGRAM	Load a program from the hard disk of the platform	29
CMD_SAVE_PROGRAM	Save a program to the hard disk of the platform	29
Special Functions Commands		
CMD_SPEAK_TEXT	Send a text to the text-to-speech converter	30
CMD_PLAY_SOUND	Play a sound file	30
CMD_GRAB_IMAGE	Grab an image from the mounted camera	30
CMD_SET_POS_HEAD_DRIVE	Set the position of the head drive	30
CMD_SET_VEL_HEAD_DRIVE	Set the velocity of the head drive	30
CMD_SET_POS_PAN_TILT_CAMERA	Set the position of the mounted pan-tilt camera	30
Debug Commands		
CMD_GET_DEBUG_DATA	Tell the platform to send its debug data	30
CMD_DEBUG1 – CMD_DEBUG4	Variable used for debug commands	30
CMD_MOUSE_MOVE	Set a mouse cursor as an obstacle	31
CMD_SET_LOG_ENABLED	Enable/Disable logging	31
CMD_SET_POST_PROCESS_FILENAME	Set the post process file name	31
CMD_SET_EVENT	Set an event	31
CMD_GET_CAMERA	Tell the platform to send information about the mounted camera	31

6.2 Message Format

In the general format of a message telegram, the length of the message is sent followed by the data bytes.

Byte	Content
1	Number of data bytes (N); byte 0 (MSB)
2	Number of data bytes (N); byte 1
3	Number of data bytes (N); byte 2
4	Number of data bytes (N); byte 3
5	1. Data byte
6	2. Data byte
...	...
N+4	N. Data byte

The data types are sent in the Motorola format with the most significant byte (MSB) first. The coding rules for simple data types are summarized in the following table:

Type	Length
BYTE	1 Byte
INT	4 Bytes
DOUBLE	8 Bytes
STRING	A string of characters. Each character is represented by a byte. A string is of arbitrary length.
XML	Same as STRING, but with the clue, that the data is to be interpreted as XML data.

In addition, every message consists of a header, which contains four characters (bytes) each. The four characters of the header have specific values as shown below:

```
m_cHeader[0] = (char)255;
m_cHeader[1] = (char)1;
m_cHeader[2] = (char)254;
m_cHeader[3] = (char)2;
```

Figure 11: Values of a header

6.3 Commands

CAN Communication Commands

CMD_CONNECT_TO_PLTF

The initialization process starts with the command CMD_CONNECT_TO_PLTF. The command causes an initialization of the CAN-field bus and a test of the CAN communication to the several CAN nodes.

Type	Description	Unit
BYTE	Command byte: CMD_CONNECT_TO_PLTF = 0	



Example: The byte stream including the message size is: 0x00, 0x00, 0x00, 0x01, 0x00

CMD_SHUTDOWN_PLTF

Used to shutdown the CAN communication.

Type	Description	Unit
BYTE	Command byte: CMD_SHUTDOWN_PLTF = 1	

CMD_INIT_PLTF

The command initializes all CAN-hardware components and executes a platform self test.

Type	Description	Unit
BYTE	Command byte: CMD_INIT_PLTF = 2	

CMD_SET_BRAKE

Sets the brakes of the drives

Type	Description	Unit
BYTE	Command byte: CMD_SET_BRAKE = 3	
	Index of the motor:	
BYTE	CANNODE_MOTORRIGHT = 1 CANNODE_MOTORLEFT = 2	
	Desired state of the brake	
BYTE	0: Brake enabled 1: Brake disabled	

CMD_SET_DIGOUT

Set one of the digital output channels.

Type	Description	Unit
BYTE	Command byte: CMD_SET_DIGOUT = 4	
	Board Type:	
INT	TYPE_IOBOARD = 0 TYPE_RELAYBOARD = 1	
INT	Channel address	
	Value	
BYTE	0: Output low 1: Output high	

CMD_START_ULTRASONIC

Start the ultrasonic sensor measurement.

Type	Description	Unit
BYTE	Command byte: CMD_START_ULTRASONIC = 5	
INT	Address of active channel	

CMD_STOP_ULTRASONIC

Stop the ultrasonic sensor measurement.

Type	Description	Unit
BYTE	Command byte: CMD_STOP_ULTRASONIC = 6	

CMD_ADD_CAN_LISTENING_ID

Add a CAN-ID to the receive list. All messages with CAN-IDs of the list will be buffered and sent on request.

Type	Description	Unit
BYTE	Command byte: CMD_ADD_CAN_LISTENING_ID = 7	
INT	CAN-ID	

CMD_REMOVE_CAN_LISTENING_ID

Removes a CAN-ID from the receive list.

Type	Description	Unit
BYTE	Command byte: CMD_REMOVE_CAN_LISTENING_ID = 8	
INT	CAN-ID	

CMD_SEND_CAN_MESSAGE

Send a CAN-Message.

Type	Description	Unit
BYTE	Command byte: CMD_SEND_CAN_MESSAGE = 9	
INT	CAN-ID	
BYTE	Data bytes of the CAN-Message (8 bytes)	

CMD_GET_CAN_MESSAGES

Request the received messages with CAN-ID of the receive list. The detail of reply telegrams is described in "Reply Telegram CMD_GET_CAN_MESSAGES".

Type	Description	Unit
BYTE	Command byte: CMD_GET_CAN_MESSAGES = 10	
	Status of message in the buffer	
BYTE	0: The message is kept 1: The received message will be deleted	

Platform Operation Commands

CMD_START

Start a platform motion mode. The operation state changes into the state "Moving".

Type	Description	Unit
BYTE	Command byte: CMD_START = 11	

CMD_BREAK

Break a platform motion mode. The operation state changes into the state "Break".

Type	Description	Unit
BYTE	Command byte: CMD_BREAK = 12	

CMD_CONTINUE

Continue the execution of a platform motion mode. The operation state changes into the state "Moving".

Type	Description	Unit
BYTE	Command byte: CMD_CONTINUE = 13	

CMD_STOP

Stop the motion of the platform. The operation state changes into the state "Ready" and waits for the next command.

Type	Description	Unit
BYTE	Command byte: CMD_STOP = 14	

CMD_SET_MODE

Set the motion mode of the platform. The operation state has to be in "Ready" before changing to another motion mode.

Type	Description	Unit
BYTE	Command byte: CMD_SET_MODE = 15	
INT	Motion mode: MODE_MOTION_NO = 0, MODE_MOTION_JOYHARD = 1, MODE_MOTION_JOYSOFT = 2, MODE_MOTION_VEL = 3, MODE_MOTION_AUTOMATIC = 4,	

MODE_MOTION_PROGRAM = 5,
MODE_MOTION_POST_PROCESS_REPLAY= 6

CMD_EXIT

Shut the platform down.

Type	Description	Unit
BYTE	Command byte: CMD_EXIT = 16	

CMD_CLOSE_SOCKET

Close the socket connection.

Type	Description	Unit
BYTE	Command byte: CMD_CLOSE_SOCKET = 17	

Request Status Commands

CMD_GET_STATUS

Request status data. The platform controller answers with a status telegram. For detailed description see "Reply Telegram CMD_GET_STATUS".

Type	Description	Unit
BYTE	Command byte: CMD_GET_STATUS = 18	

CMD_GET_POSITION

Request status data. The platform controller answers with a status telegram. For detailed description see "Reply Telegram CMD_GET_POSITION".

Type	Description	Unit
BYTE	Command byte: CMD_GET_STATUS = 74	

Motion Commands

CMD_MOVE_PATH

The platform moves on a specified path to the target.

Type	Description	Unit
BYTE	Command byte: CMD_MOVE_PATH = 19	
INT	Number of frames in the path	
INT	X-Coordinate of the first frame	[mm]
INT	Y-Coordinate of the first frame	[mm]
INT	Angle of the first frame	[1000*deg]

CMD_MOVE_TO_STATION

Move platform to a station

Type	Description	Unit
BYTE	Command byte: CMD_MOVE_TO_STATION = 20	
INT	Length of the station's name	
STRING	Name of the station	

CMD_MOVE_ABS

The platform moves to a specified frame according to the world coordinates.

Type	Description	Unit
BYTE	Command byte: CMD_MOVE_ABS = 21	
INT	X-Coordinate	[mm]
INT	Y-Coordinate	[mm]

INT	Angle	[1000*deg]
-----	-------	------------

CMD_MOVE_REL

The platform moves to a frame according to platform coordinates

Type	Description	Unit
BYTE	Command byte: CMD_MOVE_REL = 22	
INT	X-Coordinate	[mm]
INT	Y-Coordinate	[mm]
INT	Angle	[1000*deg]

CMD_MOVE_VEL_WHEEL

In the motion mode velocity the speed of the wheels is set by the command. The operation state has to be in “Moving” and in the motion mode “Wheel Velocity”.

Type	Description	Unit
BYTE	Command byte: CMD_MOVE_VEL_WHEEL = 23	
INT	Velocity of the right wheel	[mm/s]
INT	Velocity of the left wheel	[mm/s]

CMD_MOVE_VEL

The command is executed in the motion mode “Joystick (Software)”. If no velocity commands are received, the platform will decelerate to zero after few seconds.

Type	Description	Unit
BYTE	Command byte: CMD_MOVE_VEL = 24	
INT	Translational velocity	[mm/s]
INT	Rotational velocity in 1000*rad/s	[1000 * rad/s]

Platform Data Commands

CMD_SET_MAP

Set map for localization and planning. The map is specified in an XML-document.

Type	Description	Unit
BYTE	Command byte: CMD_SET_MAP = 25	
INT	Size of map name	
STRING	Name of the Map	
INT	Size of the XML-document in bytes	
XML	XML-document of the map	

CMD_GET_MAP

Request active map from platform. The reply telegram is described in “Reply Telegram CMD_GET_MAP”.

Type	Description	Unit
BYTE	Command byte: CMD_GET_MAP = 26	

CMD_LOAD_MAP

Load a stored map from the hard disk of the platform.

Type	Description	Unit
BYTE	Command byte: CMD_LOAD_MAP = 27	

CMD_SAVE_MAP

Save active map on the hard disk of the platform

Type	Description	Unit
------	-------------	------

BYTE	Command byte CMD_SAVE_MAP = 28
------	--------------------------------

CMD_MAPPING_START

Start mapping mode.

Type	Description	Unit
BYTE	Command byte CMD_MAPPING_START = 29	

CMD_MAPPING_RESET

Reset the map.

Type	Description	Unit
BYTE	Command byte CMD_MAPPING_RESET = 30	

CMD_GET_MAPPING_STOP

Stop mapping mode

Type	Description	Unit
BYTE	Command byte CMD_MAPPING_STOP = 31	

CMD_SET_ROADMAP

Set a roadmap which defines working stations and the connecting paths. The roadmap is specified in an XML-document.

Type	Description	Unit
BYTE	Command byte: CMD_SET_ROADMAP = 32	
INT	Size of roadmap name	
STRING	Name of the roadmap	
INT	Size of the XML-document in bytes	
XML	XML-document of the roadmap	

CMD_GET_ROADMAP

Request the platform to send a roadmap. The reply telegram is described in "Reply Telegram CMD_GET_ROADMAP".

Type	Description	Unit
BYTE	Command byte: CMD_GET_ROADMAP = 33	

CMD_LOAD_ROADMAP

Load a stored roadmap from the hard disk of the platform.

Type	Description	Unit
BYTE	Command byte: CMD_LOAD_ROADMAP = 34	

CMD_SAVE_ROADMAP

Save the active roadmap on the hard disk of the platform.

Type	Description	Unit
BYTE	Command byte CMD_SAVE_ROADMAP = 35	

CMD_SET_FRM_PLTF

Set the estimated frame of the mobile platform to a desired frame in global coordinates.

Type	Description	Unit
BYTE	Command byte: CMD_SET_FRM_PLTF = 36	
INT	X-Coordinate	[mm]
INT	Y-Coordinate	[mm]
INT	Angle	[1000 * deg]

CMD_SET_PLTF_PARAM

Change the value of a parameter.

Type	Description	Unit
BYTE	Command byte: CMD_SET_PLTF_PARAM = 37	
INT	Key length	
STRING	Key (property name)	
INT	Value length	
STRING	Value	

CMD_GET_PLTF_PARAM

Request the value of a single parameter. The reply telegram is described in “Reply Telegram CMD_GET_PLTF_PARAM”

Type	Description	Unit
BYTE	Command byte: CMD_GET_PLTF_PARAM = 38	
INT	Key length	
STRING	Key (property name)	

CMD_GET_PLTF_PARAM_VEC

Request the list of all supported parameters. The reply telegram is described in “Reply Telegram CMD_GET_PLTF_PARAM_VEC”.

Type	Description	Unit
BYTE	Command byte: CMD_GET_PLTF_PARAM_VEC = 39	

CMD_SET_PROGRAM

Set a program of the task interpreter. The program is sent in an XML-document.

Type	Description	Unit
BYTE	Command byte: CMD_SET_PROGRAM = 40	
INT	Size of the XML-document in bytes.	
XML	XML-document of the program.	

CMD_GET_PROGRAM

Request to send the actual program. The detail of the reply telegrams are described in “Reply Telegram CMD_GET_PROGRAM”.

Type	Description	Unit
BYTE	Command byte: CMD_GET_PROGRAM = 41	

CMD_LOAD_PROGRAM

Load a program from the hard disk of the platform.

Type	Description	Unit
BYTE	Command byte: CMD_LOAD_PROGRAM = 42	

CMD_SAVE_PROGRAM

Save a program to the hard disk of the platform.

Type	Description	Unit
BYTE	Command byte: CMD_SAVE_PROGRAM = 43	

Special Functions Commands

CMD_SPEAK_TEXT

Send a text to the text-to-speech converters.

Type	Description	Unit
BYTE	Command byte: CMD_SPEAK_TEXT = 44	
INT	Text length	
STRING	Text to speak	

CMD_PLAY_SOUND

Play a sound file.

Type	Description	Unit
BYTE	Command byte: CMD_PLAY_SOUND = 45	
INT	Length of sound file name	
STRING	File name of the sound file.	

CMD_GRAB_IMAGE

Grab an image from the mounted surveillance camera.

Type	Description	Unit
BYTE	Command byte: CMD_GRAB_IMAGE = 46	
INT	Length of image file name	
STRING	File name of image to grab.	

CMD_SET_POS_HEAD_DRIVE

Set the position of the head drive.

Type	Description	Unit
BYTE	Command byte: CMD_SET_POS_HEAD_DRIVE = 47	
INT	Angle: position of the head drive	[deg]

CMD_SET_VEL_HEAD_DRIVE

Set the velocity of the head drive.

Type	Description	Unit
BYTE	Command byte: CMD_SET_VEL_HEAD_DRIVE = 48	
INT	Velocity of the head drive	[1/100*deg/s]

CMD_SET_POS_PAN_TILT_CAMERA

Set the position of the mounted pan tilt camera.

Type	Description	Unit
BYTE	Command byte: CMD_SET_POS_PAN_TILT_CAMERA = 49	
INT	Length of the position string	
STRING	Position string	

Debug Commands

CMD_GET_DEBUG_DATA

Request all debug data from the platform. For detail of the reply telegrams, please refer to "Reply Telegram CMD_GET_DEBUG_DATA".

Type	Description	Unit
BYTE	Command byte: CMD_GET_DEBUG_DATA = 50	

CMD_DEBUG1 – CMD_DEBUG4

These commands can be used to receive individual debug data from the platform.

CMD_MOUSE_MOVE

Set the mouse obstacle.

Type	Description	Unit
BYTE	Command byte: CMD_MOUSE_MOVE = 55	
INT	X-Coordinate of the mouse obstacle	[mm]
INT	Y-Coordinate of the mouse obstacle	[mm]
The button state:		
INT	EV_MOUSE_RBUTTON_DOWN = 2	
	EV_MOUSE_RBUTTON_UP = 5	

CMD_SET_LOG_ENABLED

Enable or disable logging. *Not used at the moment.*

Type	Description	Unit
BYTE	Command byte: CMD_SET_LOG_ENABLED = 56	

CMD_SET_POST_PROCESS_FILENAME

Not used at the moment.

Type	Description	Unit
BYTE	Command byte: CMD_SET_POST_PROCESS_FILENAME = 57	

CMD_SET_EVENT

Type	Description	Unit
BYTE	Command byte: CMD_SET_EVENT = 58	
The event to set:		
	EV_START_POST_PROCESS_RECORDING = 1	
	EV_STOP_POST_PROCESS_RECORDING = 2	
INT	EV_START_POST_PROCESS_REPLAY = 3	
	EV_STOP_POST_PROCESS_REPLAY = 4	
	EV_BREAK_POST_PROCESS_REPLAY = 5	
	EV_CONTINUE_POST_PROCESS_REPLAY = 6	

CMD_GET_CAMERA

For details of the reply telegram, please refer to “Reply Telegram CMD_GET_CAMERA”.

Type	Description	Unit
BYTE	Command byte: CMD_GET_CAMERA = 59	

6.4 Reply Telegrams

CMD_GET_CAN_MESSAGES

The telegram is the reply of the command CMD_GET_CAN_MESSAGES.

Type	Description	Unit
BYTE	Telegram identifier: CMD_GET_CAN_MESSAGES = 10	
INT	The number of CAN messages	
DOUBLE	The time-stamp of the CAN message	
INT	The ID of the CAN message	
BYTE	The byte #1 - #8 of the CAN message	

CMD_GET_STATUS

The telegram is the reply of the command CMD_GET_STATUS.

Type	Description	Unit
BYTE	Telegram identifier CMD_GET_STATUS = 18	

Commanded platform frame		
INT	X-Coordinate	[mm]
INT	Y-Coordinate	[mm]
INT	Angle	[deg*1000]
Estimated platform frame		
INT	X-Coordinate	[mm]
INT	Y-Coordinate	[mm]
INT	Angle	[deg*1000]
Covariance of the estimated platform frame		
INT	Deviation in principal axis direction	[mm]
INT	Deviation in minor axis direction	[mm]
INT	Covariance of the angle	[deg*1000]
INT	Angle of the principal axis	[deg*1000]
Observed scan		
n times {	INT	Number of scan points (n)
	INT16	X-Coordinate of a scan point [mm]
	INT16	Y-Coordinate of a scan point [mm]
Reflectors observed		
n times {	INT	Number of reflector points (n)
	INT	X-Coordinate of the reflector [mm]
	INT	Y-Coordinate of the reflector [mm]
Lines observed		
n times {	INT	Number of lines (n)
	INT	X-Coordinate of the target from the platform's frame [mm]
	INT	Y-Coordinate of the target from the platform's frame [mm]
	INT	X-Coordinate of the source from the platform's frame [mm]
	INT	Y-Coordinate of the source from the platform's frame [mm]
Circles observed		
n times {	INT	Number of circles (n)
	INT	X-Coordinate of the center of the circle [mm]
	INT	Y-Coordinate of the center of the circle [mm]
	INT	Radius [mm]
Corners observed		
n times {	INT	Number of corners (n)
	INT	X-Coordinate of the corner [mm]
	INT	Y-Coordinate of the corner [mm]
Interpolated path		
n times {	INT	Number of Frames
	INT	X-Coordinate [mm]
	INT	Y-Coordinate [mm]
Joystick Buttons		
n times {	INT	Number of Buttons (n)
	BOOL	Pressed / not pressed
Current commanded velocities		
INT	Translation velocity	[mm/s]
INT	Rotation velocity	[1/1000*rad/s]
Data from relay board		
INT	Digital In	
INT	Status of board	
8xINT	Analogue in #1- #8	
4xINT	Optosensor Distance	
IO-Board		
INT	Battery voltage	

INT	Digital in	
INT	Digital out	
INT	Status of board	
8xINT	Analogue in #1 - #8	
Motor Control Board		
INT	Status of Motor Control Board on right side	
INT	Status of Motor Control Board on left side	
INT	Temperature of Motor Control Board on right side	
INT	Temperature of Motor Control Board on left side	
Measured distances of the ultrasonic sensors		
16xINT	Distance of sensor #1 - #16	[mm]
Analogue data from ultrasonic board		
8xINT	Analogue in data #1- #8	
INT	Status of ultrasonic board	
Measured velocities from the radar board		
4xINT	Velocity of sensor 1 - 4	[mm/s]
INT	Status of radar board	
GPS DATA		
DOUB	Latitude	
DOUB	Longitude	
Gyro board		
3xINT	Acceleration sensor #1 - #3 multiplied by 10000	
INT	Angle of gyro board	[rad*10000]
INT	Temperature of gyro board	
INT	Status of gyro board	
Miscellaneous status information		
Platform initialization		
BYTE	0: Not initialized 1: Initialized	
Platform target reached		
BYTE	0: Target not reached 1: Target reached	
BYTE	Active motion mode. Coding see CMD_SET_MODE	
Active operation state:		
BYTE	ST_OP_READY = 0 ST_OP_RUNNING = 1 ST_OP_BROKEN = 2 ST_OP_NOT_INITIALIZED = 3	
Active emergency state:		
INT	ST_EM_NOT_ACTIVE = 0 ST_EM_ACTIVE = 1 ST_EM_RESET = 2	
BYTE	Application mode of a local application; not used at the moment	
Planner state		
INT	PLANNER_STANDBY = 0 PLANNER_PLANNING = 1 PLANNER_NO_PATH = 2 PLANNER_PATH_FOUND = 3 PLANNER_COLLISION_START = 4 PLANNER_COLLISION_TARGET = 5	
INT	Message time stamp	
Error code of applications:		
INT	APP_ERR_NONE = 0 APP_ERR_DOCKING = 1	

BYTE	Docked to charging station 0: Not docked 1: Docked
BYTE	Localization status 0: Localization not lost 1: Localization lost
BYTE	Collision 0: No collision 1: Collision
BYTE	Percent done while calculating lookup data 0: finished / currently not calculating 1-99: actual percentage
Surveillance information	
4xBYTE	State of IR sensor #1 - #4 0: Inactive 1: Active
BYTE	State of alarm siren 0: Disabled 1: Enabled
BYTE	State of flash light 0: Disabled 1: Enabled
Position of head drive	
INT	Position of head drive [deg]
Message check	
INT	Value is always -1

CMD_GET_ROADMAP

The telegram is the reply of the command CMD_GET_ROADMAP.

Type	Description	Unit
BYTE	Telegram identifier: CMD_GET_ROADMAP = 33	
BYTE	Number of characters in map	
INT	Length of the XML-document in bytes	
XML	XML-document of the roadmap	
INT	Length of the string name	
STRING	Length of the filename	

CMD_GET_MAP

The telegram is the reply of the command CMD_GET_MAP.

Type	Description	Unit
BYTE	Telegram identifier: CMD_GET_MAP = 26	
BYTE	Number of characters in map	
INT	Length of the XML-document in bytes	
XML	XML-document of the roadmap	
INT	Length of the string name	
STRING	Length of the filename	

CMD_GET_PROGRAM

The telegram is the reply on the command CMD_GET_PROGRAM.

Type	Description	Unit
BYTE	Telegram identifier: CMD_GET_PROGRAM = 41	
INT	Size of a program	
STRING	Name of a program	

CMD_GET_PLTF_PARAM

The telegram is the reply of the command CMD_GET_PLTF_PARAM.

Type	Description	Unit
BYTE	Telegram identifier: CMD_GET_PLTF_PARAM = 38	
INT	Length of key string	
STRING	Key (parameter name)	
INT	Length of value string	
STRING	Value	
	The category the parameter belongs to:	
	GEOM = 0, PLANNING = 1, IPO = 2, LOCALIZATION = 3, TRACKING_DETECTION = 4, COLLISION = 5, SURVEILLANCE = 6, MOTION_CTRL = 7, MAPPING = 8, USONIC = 9, INFRARED = 10	
	The data type of the parameter:	
	BOOLEAN = 0, INTEGER = 1, DOUBLE = 2	
INT	The minimum value of this parameter	
INT	The maximum value of this parameter	

CMD_GET_PLTF_PARAM_VEC

The telegram is the reply of the command CMD_GET_PLTF_PARAM_VEC.

Type	Description	Unit
BYTE	Telegram identifier: CMD_GET_PLTF_PARAM_VEC = 39	
INT	Number of parameters (n)	
INT	Length of key string	
STRING	Key (property name)	
INT	Length of value string	
STRING	Value	
	The category the parameter belongs to:	
	GEOM = 0, PLANNING = 1, IPO = 2, LOCALIZATION = 3, TRACKING_DETECTION = 4, COLLISION = 5, SURVEILLANCE = 6, MOTION_CTRL = 7, MAPPING = 8, USONIC = 9, INFRARED = 10	
	The data type of the parameter:	
	BOOLEAN = 0, INTEGER = 1, DOUBLE = 2	
INT	The minimum value of this parameter	

n times

INT	The maximum value of this parameter
-----	-------------------------------------

CMD_GET_DEBUG_DATA

The telegram is the reply of the command CMD_GET_DEBUG_DATA.

Type	Description	Unit
BYTE	Telegram identifier: CMD_GET_DEBUG_DATA = 50	
Validation gate		
INT	Size of validation gate vector	
INT	X/Y-Coordinate pairs the four points of a validation gate.	
Mapping mode		
	Indicates if the platform is in the mapping mode.	
BYTE	0: Disabled. The next data already belongs to the interpolated path 1: Enabled	
Segments mapped		
INT	Number of segments mapped	
INT	X-Coordinate of source	[mm]
INT	Y-Coordinate of source	[mm]
INT	X-Coordinate of target	[mm]
INT	Y-Coordinate of target	[mm]
Explored segments mapped		
INT	Number of explored segments being mapped	
INT	X-Coordinate of source	[mm]
INT	Y-Coordinate of source	[mm]
INT	X-Coordinate of target	[mm]
INT	Y-Coordinate of target	[mm]
Global localization		
INT	Size of the global localization vector	
INT	X-Coordinate of the global localization frame	[mm]
INT	Y-Coordinate of the global localization frame	[mm]
INT	Angle of the n th global localization frame	[rad]
INT	Size of the visible lines vector	
INT	X-Coordinate of line source	[mm]
INT	Y-Coordinate of line source	[mm]
INT	X-Coordinate of line target	[mm]
INT	Y-Coordinate of line target	[mm]
INT	Size of the validation gate in global localization	
INT	X/Y-Coordinates of the four points of the validation gate in global localization.	[mm]
INT	Size of path	
INT	X-Coordinates of path	[mm]
INT	Y-Coordinates of path	[mm]
INT	Angle	[rad]
Graphical objects vector		
INT	Size of graphical object vector	
INT	Type of the graphical object: TYPE_POINT_CROSS = 0 TYPE_POINT_RECT = 1 TYPE_POINT_TRIA = 2 TYPE_POINT_SMALLDOT = 3 TYPE_POINT_LARGEDOT = 4 TYPE_LINE_THIN = 5 TYPE_LINE_THICK = 6 TYPE_LINE_ARROW = 7 TYPE_LINE_DOUBLEDOT = 8	

	TYPE_FRAME_RECT = 9	
	TYPE_FRAME_CIRC = 10	
	TYPE_FRAME_TRIA = 11	
	TYPE_FRAME_PLTF = 12	
	TYPE_CIRCLE = 13	
	TYPE_STRING = 14	
	TYPE_ELLIPSE = 15	
	TYPE_NUM_LABEL = 16	
	TYPE_POINT_PIXEL = 17	
	TYPE_POINT_PERM = 18	
	Color of the graphical object:	
	COLOR_RED = 20	
	COLOR_GREEN = 21	
	COLOR_BLUE = 22	
	COLOR_PINK = 23	
	COLOR_MAGENTA = 24	
INT	COLOR_WHITE = 25	
	COLOR_YELLOW = 26	
	COLOR_GRAY = 27	
	COLOR_ORANGE = 28	
	COLOR_DARKGRAY = 29	
	COLOR_CYAN = 30	
INT	X-Coordinate of the graphical object	[mm]
INT	Y-Coordinate of the graphical object	[mm]
INT	Angle	[deg]

Three variables of type INT are used to send data of objects. These variables are used differently, depending on category of graphical objects.

	Object category	Content of each integer
	Point	All variables are 0.
	Line/Circle	Variable #1: X-coordinate [mm] Variable #2: Y-coordinate [mm] Variable #3: 0
INT	Frame	Variable #1: 0 Variable #2: 0 Variable #3: Angle [deg * 1000]
	Ellipse	Variable #1: X-coordinate [mm] Variable #2: Y-coordinate [mm] Variable #3: Angle [deg * 1000]
	Label	Variable #1: X-coordinate [mm] Variable #2: 0 Variable #3: 0

INT	0	
Variables (not used at the moment)		
INT	Number of variables	
INT	ID of the variable	
INT	Length of the variables value string	
STRING	String of the variable's value	
Moving clusters		
INT	Number of moving clusters	
INT	X-Coordinate of the moving cluster	[mm]
INT	Y-Coordinate of the moving cluster	[mm]
INT	Index of the moving cluster	
Person clusters		

INT	Number of moving clusters	
INT	The X-Coordinate of the person cluster	[mm]
INT	The Y-Coordinate of the person cluster	[mm]
INT	The index of the person cluster	
Ball cluster		
INT	Number of ball clusters	
INT	X-Coordinate of the ball cluster	[mm]
INT	Y-Coordinate of the ball cluster	[mm]
INT	Index the ball cluster	
Protocol queue		
INT	Number of entries in the protocol queue	
INT	Length of the queue entries string	
STRING	The queue entry	
TCP/IP statistics		
INT	Average receive time	[s]
INT	Percentage of receive time	[%]
INT	Average send time	[s]
INT	Percentage of send time	[%]
INT	Received bytes per second	[Byte/s]
INT	Sent bytes per second	[Byte/s]
INT	Received telegrams per second	[s]
INT	Sent telegrams per second	[s]
INT	Average wait time for telegram	[s]
INT	Percentage of wait time for telegram	[%]
Safe Velocities		
INT	Safe velocity for forward laser	[mm/s]
INT	Safe velocity for backward laser	[mm/s]
INT	Safe velocity for forward US	[mm/s]
INT	Safe velocity for backward US	[mm/s]
INT	Safe velocity for optocoupler	[mm/s]
Message Control		
INT	-1	

CMD_GET_CAMERA

The telegram is the reply of the command CMD_GET_CAMERA.

Type	Description	Unit
BYTE	Telegram identifier: CMD_GET_CAMERA = 59	
INT	The length of the IP string of the camera	
STRING	The IP address of the camera	
INT	The length of the ID string of the camera	
STRING	The ID of the camera	

CMD_GET_POSITION

The telegram is the reply of the command CMD_GET_POSITION.

Type	Description	Unit
BYTE	Telegram identifier: CMD_GET_POSITION = 74	
INT	The estimated X-Position of the platform	
INT	The estimated Y-Position of the platform	
DOUBLE	The angle of the platform to x axis	
DOUBLE	The translational velocity of the platform	
DOUBLE	The rotational velocity of the platform	

6.5 XML-Documents

Map

Element: <map>

Description: Defines the map for localisation and planning.

Child elements: (layer*)

Attributes:

Attribute	Type	Description
Version	NMTOKEN	The version of the map DTD the XML document is using

Element: <layer>

Description: A map consists of several layers.

Child elements: (workarea?), (circle*) | (line*) | (auxline*) | (linemapped*) | (auxlinemapped*) | (reflmark*) | (prohibit*) | (hiddenpolygon*) | (hiddenlabel*)

Attributes:

Attribute	Type	Description
index	Integer	Index of the layer

Element: <circle>

Description: Defines a circle in the map.

Child elements: (center), (radius)

Attributes: None

Element: <line>

Description: Defines a line in the map.

Child elements: (point), (point)

Attributes: None

Element: <auxline>

Description: Defines an auxiliary line in the map.

Child elements: (point), (point)

Attributes: None

Element: <linemapped>

Description: Defines a line that has been mapped

Child elements: (point), (point)

Attributes: None

Element: <auxlinemapped>

Description: Defines an auxiliary line that has been mapped.

Child elements: (point), (point)

Attributes: None

Element: <reflmark>

Description: Defines reflectors in the map.

Child elements: (point), (point)

Attributes: None

Element: <prohibit>

Description: Defines areas in the map which the robot is not allowed to enter.

Child elements: (point), (point), (point+)

Attributes: None

Element: <workarea>

Description: Defines the area in the map which the robot is not allowed to leave.

Child elements: (point), (point), (point+)

Attributes: None

Element: <hiddenpolygon>

Description: Defines areas in the map which are used to make the GUI usage easier and do not affect the platform's behaviour.

Child elements: (point), (point), (point+)

Attributes: None

Element: <center>

Description: Defines the centre of a circle.

Child elements: None

Attributes:

Attribute	Type	Description
x	NMTOKEN	X-Coordinate in mm.
y	NMTOKEN	Y-Coordinate in mm

Element: <radius>

Description: Defines the radius of a circle.

Child elements: None

Attributes:

Attribute	Type	Description
r	NMTOKEN	Radius in mm.

Element: <point>

Description: Defines a single point.

Child elements: None

Attributes:

Attribute	Type	Description
x	NMTOKEN	X-Coordinate in mm.
y	NMTOKEN	Y-Coordinate in mm

Element: <hiddenlabel>

Description: Defines a label in the map that does not affect the platform but is meant to make GUI usage easier.

Child elements: (point)

Attributes:

Attribute	Type	Description
text	CDATA	Name of the label

Roadmap

Element: <roadmap>

Description: A roadmap defines target positions for the mobile platform as well as paths between these positions.

Child elements: (stations), (edges)

Attributes:

Attribute	Type	Description
Version	NMTOKEN	The version of the roadmap DTD the XML document is using

Element: <stations>

Description: Defines a list of target positions called stations.

Child elements: (station*)

Attributes: None

Element: <edges>

Description: Defines a list of paths connecting the stations.

Child elements: (edge*)

Attributes: None

Element: <station>

Description: Defines a target station for the platform.

Child elements: (point)

Attributes:

Attribute	Type	Description
id	ID	Unique identifier.
name	NMTOKEN	Name of the station
angle	NMTOKEN	Angle

Element: <edge>

Description: Defines a single path between two stations.

Child elements: (points)

Attributes:

Attribute	Type	Description
startstation	IDREF	Identifier of the start station
endstation	IDREF	Identifier of the end station

Element: <point>

Description: Defines a single point.

Child elements: None

Attributes:

Attribute	Type	Description
-----------	------	-------------

x	NMTOKEN	X-Coordinate of the point
y	NMTOKEN	Y-Coordinate of the point

Program

Element: <program>

Description: Defines a program that might consist of several tasks.

Child elements: (task*)

Attributes:

Attribute	Type	Description
version	NMTOKEN	The version of the program DTD the XML document is using

Note: Program.dtd has the entity "task.dtd".

Task

Element: <task>

Description: Defines a task in a program. A task is a sequence of commands including branch commands and labels.

Child elements: (ioevent*) | (goto*) | (label*) | (load*) | (log*) | (motion*) | (parameter*) | (surveillance*) | (audio*) | (turnhead*) | (wait*)

Attributes:

Attribute	Type	Description
name	NMTOKEN	The name of the task
id	CDATA	The ID of the task
priority	NMTOKEN	Priority of the task
version	NMTOKEN	The version of the task DTD the XML document is using

Element: <ioevent>

Description: Executes an IO-event on the platform.

Child elements: (digitalout)

Attributes: None

Element: <digitalout>

Description: Executes an IO-event on the platform.

Child elements: None

Attributes:

Attribute	Type	Description
D1	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D2	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D3	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D4	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D5	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D6	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D7	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on

D8	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D9	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D10	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D11	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on
D12	NMTOKEN	-1 if the state should not be changed, 0: off and 1: on

Element: <goto>

Description: When executed the program jumps to the line with the specified label.

Child elements: None

Attributes:

Attribute	Type	Description
label	CDATA	The name of the label to go to
condition	NMTOKEN	Must be true to execute goto

Element: <label>

Description: A marker as destination for a goto-jump.

Child elements: None

Attributes:

Attribute	Type	Description
name	CDATA	The name of the label.

Element: <load>

Description: Loads the map and roadmap with the specified names.

Child elements: None

Attributes:

Attribute	Type	Description
map	NMTOKEN	The name of the map file to load.
roadmap	NMTOKEN	The name of the roadmap file to load.

Element: <log>

Description: Sends a log message to PlatformCtrlGUI.

Child elements: None

Attributes:

Attribute	Type	Description
type	NMTOKEN	The type of the log message.
message	CDATA	The message string

Element: <motion>

Description: Sets a motion command for the platform.

Child elements: (settarget) | (moveto)

Attributes:

Attribute	Type	Description
-----------	------	-------------

stop	NMTOKEN	1, if the platform should stop motion
------	---------	---------------------------------------

Element: <settarget>

Description: Sets a target for the next platform motion.

Child elements: (coord) | (station)

Attributes: None

Element: (coord)

Description: Describes the coordinates for the (settarget) element.

Child elements: None

Attributes:

Attribute	Type	Description
x	NMTOKEN	The X-Coordinate of the frame
y	NMTOKEN	The Y-Coordinate of the frame
angle	NMTOKEN	The angle of the frame

Element: <station>

Description: Describes a station for the (settarget) element.

Child elements: None

Attributes:

Attribute	Type	Description
name	CDATA	The name of the station

Element: <moveto>

Description: Describes a “move to” command for the platform.

Child elements: (trackedperson) | (trackedball) | (trackedmovingobject) | (chargingstation)

Attributes:

Attribute	Type	Description
onErrorTask	CDATA	Names a task, that should be executed on error

Element: <trackedperson>

Description: The platform should move to a tracked person.

Child elements: None

Attributes: None

Element: <trackedball>

Description: The platform should move to a tracked ball.

Child elements: None

Attributes: None

Element: <trackedmovingobject>

Description: The platform should move to a tracked object.

Child elements: None

Attributes: None

Element: <chargingstation>

Description: The platform should move to the charging station.

Child elements: None

Attributes: None

Element: <parameter>

Description: Used to set a platform parameter.

Child elements: (set)

Attributes: None

Element: <set>

Description: Describes a parameter's key and its value.

Child elements: None

Attributes:

Attribute	Type	Description
key	NMTOKEN	The key of the parameter to be set
value	NMTOKEN	The value for the parameter key to be set

Element: <surveillance>

Description: Contains commands for surveillance options.

Child elements: (grab) | (movecamera)

Attributes: None

Element: <grab>

Description: Used to either grab a scan or an image from the camera or both.

Child elements: None

Attributes:

Attribute	Type	Description
filename	NMTOKEN	The name for the file(s) to grab (for image .jpg is appended, for scan .dat)
grabscan	true false	Whether to grab a scan (true) or not (false)
grabimage	true false	Whether to grab an image (true) or not (false)
autoobservation	true false	Whether the platform should observe automatically

Element: <movecamera>

Description: Moves the camera mounted onto the platform.

Child elements: None

Attributes:

Attribute	Type	Description
position	CDATA	The position string (camera dependent)

Element: <audio>

Description: Contains options for audio output.

Child elements: (speak) | (play)

Attributes: None

Element: <speak>

Description: Contains text for the platforms text-to-speech system.

Child elements: None

Attributes:

Attribute	Type	Description
text	CDATA	The text for the platforms text-to-speech system

Element: <play>

Description: Specifies the wav-file that is to be played by the platform.

Child elements: None

Attributes:

Attribute	Type	Description
filename	NMTOKEN	The name of the wav-file to be played

Element: <turnhead>

Description: Turns the platform's head into the given direction.

Child elements: None

Attributes:

Attribute	Type	Description
angle	NMTOKEN	The angle for the head to turn

Element: <wait>

Description: Makes the task wait until the specified event occurs.

Child elements: (collision?) | (movingobjects?) | (person?) | (ball?) | (lostlocalisation?) | (target?) | (infrared?) | (radar?) | (date?) | (delay?) | (userevent?) | (random?) | (batterycharge?)

Attributes:

Attribute	Type	Description
timeoutlabel	CDATA	The label to jump to, when a timeout occurs
timeout	NMTOKEN	The timeout limit
for	NMTOKEN	

Element: <collision>

Description: Makes the task wait for a collision.

Child elements: None

Attributes: None

Element: <movingobject>

Description: Makes the task wait until a moving object has been detected.

Child elements: None

Attributes: None

Element: <person>

Description: Makes the task wait until a person has been detected.

Child elements: None

Attributes: None

Element: <ball>

Description: Makes the task wait until a ball has been detected.

Child elements: None

Attributes: None

Element: <lostlocalization>

Description: Makes the task wait until the platform has lost its localisation.

Child elements: None

Attributes: None

Element: <target>

Description: Set a target for the platform to move to.

Child elements: None

Attributes: None

Element: <infrared>

Description: Makes the task wait until one of the infrared detectors detected something.

Child elements: None

Attributes:

Attribute	Type	Description
direction	NMTOKEN	The direction of the infrared detection.

Element: <radar>

Description: Makes the task wait until one of the radar sensors detected something.

Child elements: None

Attributes:

Attribute	Type	Description
direction	NMTOKEN	The direction of the radar detection.

Element: <date>

Description: Makes the task wait until the specified date and time are reached.

Child elements: None

Attributes:

Attribute	Type	Description
day	NMTOKEN	The day of the date specified

month	NMTOKEN	The month of the date specified
year	NMTOKEN	The year of the date specified
hour	NMTOKEN	The hour of the date specified
minute	NMTOKEN	The minute of the date specified
second	NMTOKEN	The second of the date specified

Element: <delay>

Description: Makes the task wait for a specified time.

Child elements: None

Attributes:

Attribute	Type	Description
amount	NMTOKEN	The amount of milliseconds to wait

Element <userevent>

Description: Makes the task wait until the specified user event occurs.

Child elements: None

Attributes:

Attribute	Type	Description
id	NMTOKEN	The ID of the user event

Element: <random>

Description: Makes the task wait for a randomly chosen time.

Child elements: None

Attributes:

Attribute	Type	Description
min	NMTOKEN	The minimum time to wait
max	NMTOKEN	The maximum time to wait

Element: <batterycharge>

Description: Makes the task wait until the battery level has reached the specified threshold.

Child elements: None

Attributes:

Attribute	Type	Description
limit	NMTOKEN	The specified threshold of the charging state

7 Legal notes

Version information

This document has been translated and is not the original. Please refer to the German version in case of uncertainties or questions.

Liability

Every care has been taken in the preparation of this manual which represents the state of technology at the time of its composing. However, inaccuracies or omissions might occur. Please inform Neobotix in case you notice any.

The GPS GmbH, Neobotix, cannot be held responsible for any technical or typographical errors and reserves the right to make changes to the product and manual without prior notice.

Neobotix makes no warranty of any kind with regard to the material contained within this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Neither the GPS GmbH nor Neobotix shall be liable or responsible for incidental or consequential damages in connection with the improper use of one or more of the products described in this manual.



Declaration of conformity

This product fulfils all relevant directives of the European Union. For further information please contact Neobotix.

Downloads and further information

Additional information, data sheets and documentations, also for the other products of Neobotix, can be found on our homepage www.neobotix.de/downloads.

Imprint

GPS GmbH, Abt. Neobotix
Holderäckerstr. 5, D-70839 Gerlingen
www.neobotix.de

Contact: Dipl.-Ing. Till May
Tel.: (+49) 7156 / 42 59-121
e-mail: may@neobotix.de