Machine and Robot in Harmony

# TPS: Reference guide

# Trio Motion Technology

## Trio Robotics Series

First Edition • 2020

Revision: 1.4

Trio Programming Guides are designed to aid learning of the TrioBASIC language through description and examples. Each one will cover a particular topic and discuss which commands and parameters in the TrioBASIC are required to complete the task.

A general understanding of TrioBASIC is required and it is recommended to attend an introduction to TrioBASIC training course. The programming guides are not a replacement for the TrioBASIC help files which can be found in *Motion* Perfect as well as the manual which cover each command and parameter in more detail and should be referenced when required.

Any examples given in the programming guide will work and have been tested on an isolated controller. If you choose to use these examples on a machine, please take care that it will not cause damage or injury and that they are correctly included in the project changing parameters and values where required.



UK | USA | CHINA | INDIA

www.triomotion.com

### SAFETY WARNING

During the installation or use of a control system, users of Trio products must ensure there is no possibility of injury to any person, or damage to machinery.

Control systems, especially during installation, can malfunction or behave unexpectedly.

Bearing this in mind, users must ensure that even in the event of a malfunction or unexpected behaviour the safety of an operator or programmer is never compromised.

This document uses the following icons for your reference:

Information that relates to safety issues and critical software information.

Information to highlight key features or methods.

Useful tips and techniques.

Example programs.

# Contents

# 1 TPS Teach pendant

The Teach Programming System allows programming the robot in a managed and safe environment, using a real or virtual system.

The pendant is based around Trio's UNIPLAY HMI and can be used as a "real" or "virtual" on-screen pendant. The system includes extensive software and preconfigured motion control functions, which permit the controlling of all standard types of robot with reduced development time.

# 2 The device

## 2.1 Introduction

A teach pendant is a device by which an operator can jog a robot and create programs in a friendly and easy way. Thanks to its safety features and buttons it is completely secure to control a robot in a safe environment.


Figure 2-1: Teach pendant device

A virtual teach pendant has been developed that, along with other Motion Perfect robot tools, provides a virtual robot environment to design and debug robot solutions over PC.

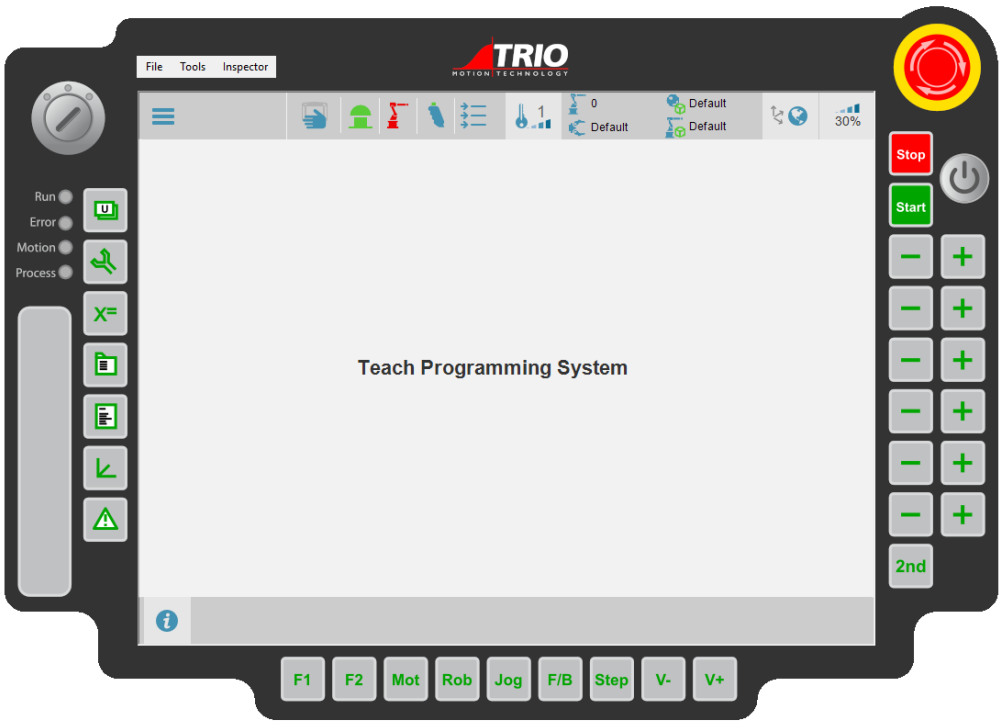



Figure 2-2: Virtual pendant

## 2.2 Technical specifications

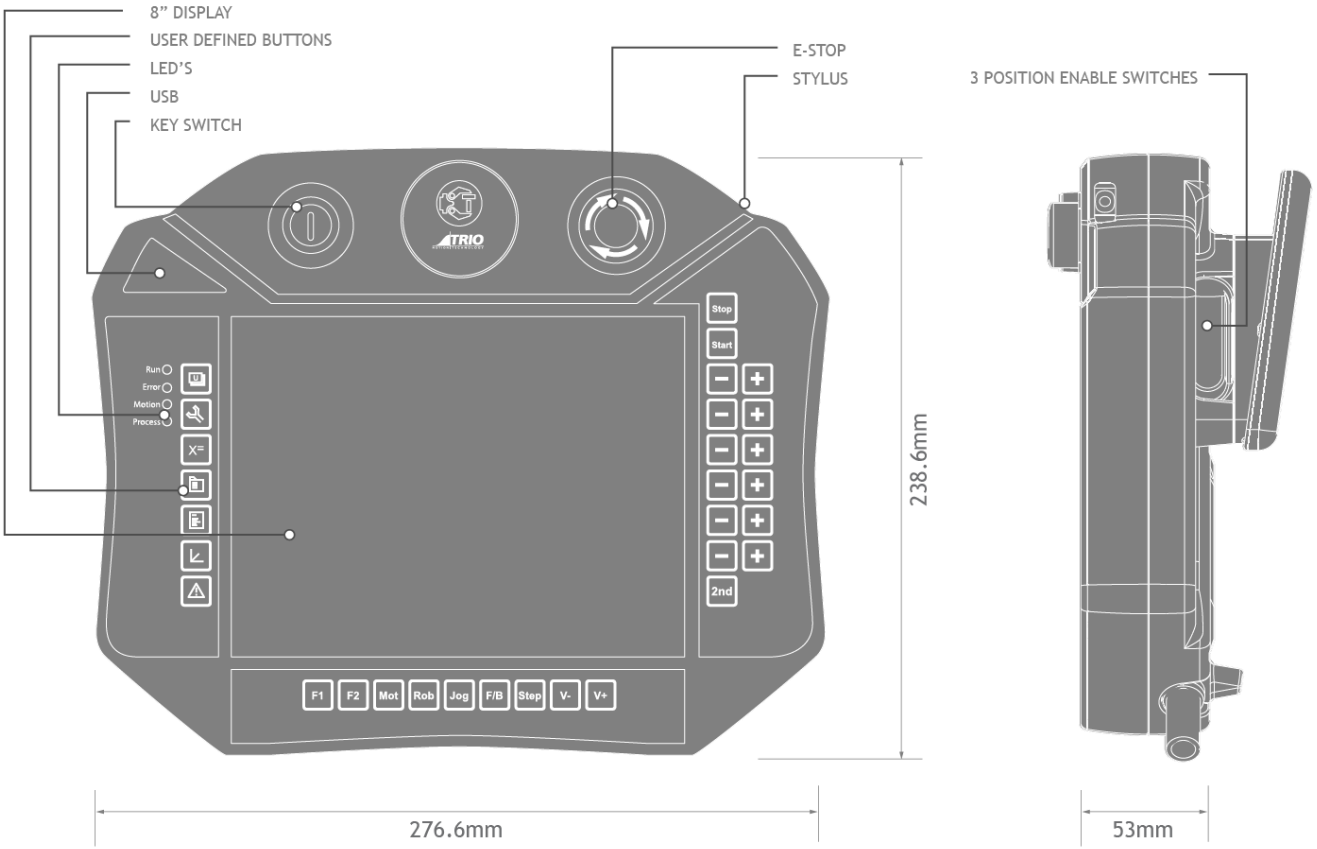



Figure 2-3: teach pendant device specs

| | | | |
|---|---|---|---|
| **Processor** | 32 bit RISC Cortex | **Safety Hardware** | Front mounted 3 position E-Stop. 2 x rear mounted 3 position enable switch. |
| **Memory** | 512 MB DDR | | |
| **Micro SD card** | Up to 32 Gb | **Protection Rating** | IP65 |
| **Display** | 8" (640 x 800) touch screen | **Input Voltage** | 24V DC via junction box (included) |
| **Pendant Software** | Windows CE Embedded 7 | **Size (w x d x h) mm** | 276.6 x 53 x 238.6 |
| **Membrane Keys** | 31 user definable keys (with *Motion Perfect 4*) | **Weight** | 2kg |
| **Communication** | Ethernet<br>USB 2.0 | **Cable** | 5m 26 pin shell type. RoHS compliant (connects to supplied junction box) |
| **Stylus** | Included | | |

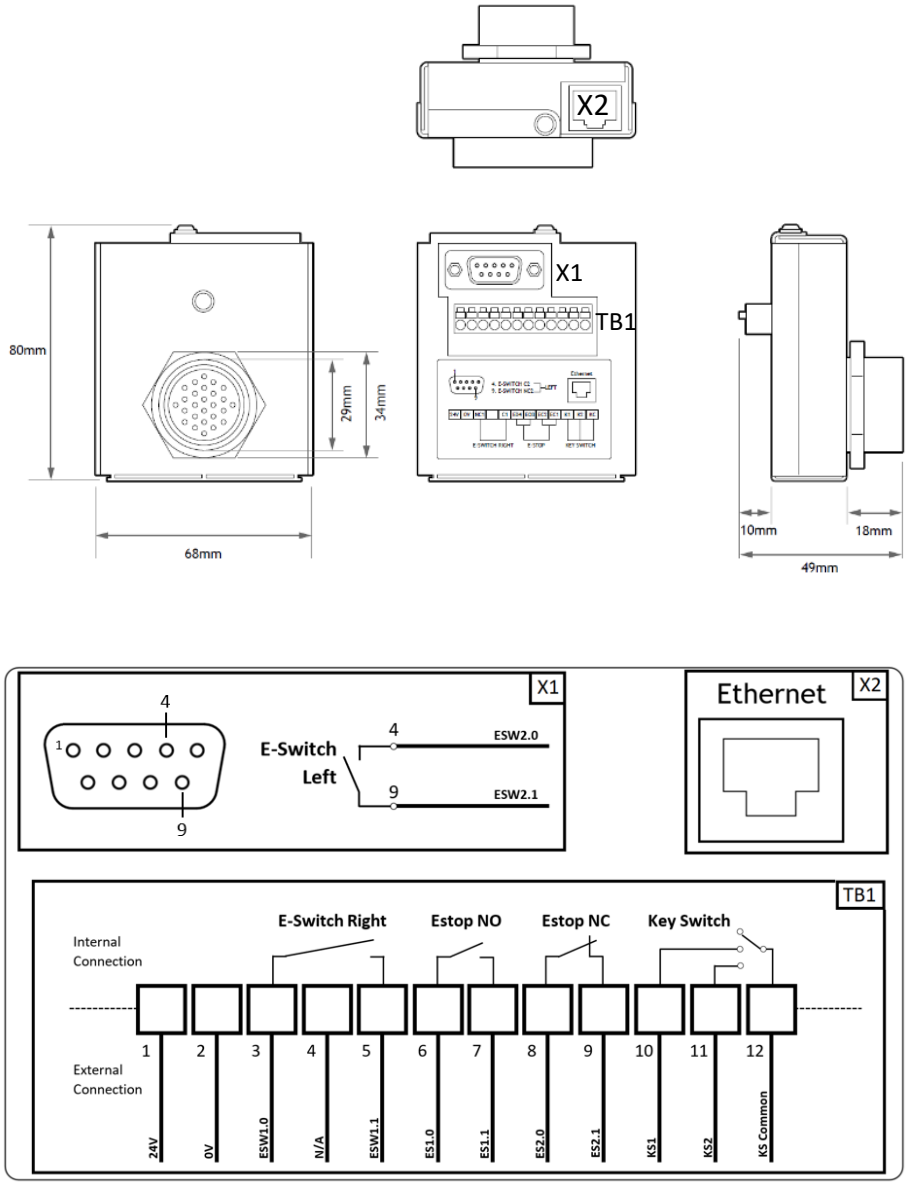



Figure 2-4: Junction Box

RPS has to be configured depending on what inputs had been used to connect the junction box with the controller.

Refer to section in IO config section of TPS, page 73 of this document, to set the inputs numbers into the system.

## 2.3 Safety components

This section describes safety components and procedures to be used when the Teach Programming System is operated.

However, it does not cover how to design for safety nor how to install safety related equipment.

**The safety components must be wired accordingly with safety standards that regulates automation and it has to be conducted by a qualified person.**

*In the case of using a Virtual Teach Pendant, the safety components wiring is done through a software casing mechanism. For more details go to* *Virtual Teach Pendant casing section, page 10.*

### 2.3.1 E-Stop

An emergency stop is a state that overrides any other robot control, disconnects drive power from the robot's motors, stops all moving parts and disconnects power from any potentially dangerous functions controlled by the robot system.

An emergency stop state means that all power is disconnected from the robot except for the manual brake release circuits. The E-Stop button can be configured in IO config section of TPS, page 73 of this document.

The E-Stop button can be found in the upper right corner of both real and virtual teach pendant and it has the following outlook:


Figure 2-5: E-Stop real Teach Pendant

Figure 2-6: E-Stop virtual Teach Pendant

It must be performed a recovery procedure in order to return to normal operation.

The recovery procedure is triggered through the input "Error Reset" that can be configured in IO config section of TPS, page 73 of this document.

### 2.3.2 Demand switch

Demand switch component is an enabling device, manually operated constant pressure three position push-button. The Teach Pendant is equipped with two demand switches, positioned at the left and right back side of the device, allowing for a left and right-handed operation.

The demand switch can be found at the back of the real Teach Pendant and at the front left side of the virtual one with the following outlook:

Figure 2-7: Demand switch real Teach Pendant

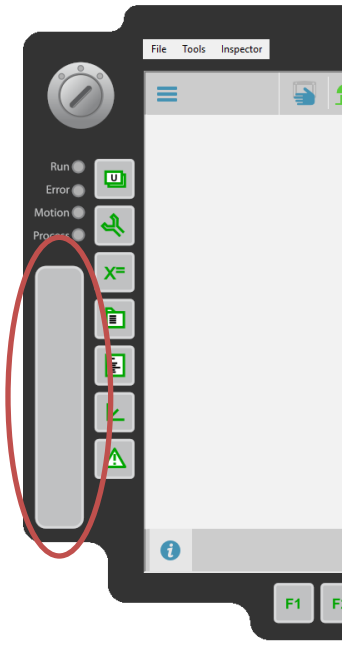



Figure 2-8: Demand switch virtual Teach Pendant

The demand switches can be in the following three positions:

| Position | Function | Demand switch | Contacts |
|---|---|---|---|
| 1 | Home position | Is not pressed | Enabling output are open |
| 2 | Enabling | Is pressed | Enabling output are close |
| 3 | Panic | Is pressed strong | Enabling output are open |

The Teach Pendant should be hold by the operator in a comfortable position, where demand switch can be pressed with one hand while programming or operating the keypad with the other hand.

A recommended way of holding it could be as the following picture:

Figure 2-9: Demand switch pressed

**When demand switch is in enabling function, the system is in active status and the motors are powered, allowing for movement through different functions but not initiating them. In any other position, the motors will not be powered and the system will be safely stopped.**

This component will enable the device only in manual mode. In auto mode Demand switch component is disabled and MOT button should be pressed to enable the system.

### 2.3.3 Key switch

There are three different modes where the system can be:

- **Manual**: to operate the system, demand switch has to be enabled. The maximum speed will be set as WORLD_POS_REDUCED_SPEED and WORLD_ORI_REDUCED_SPEED, set by the system integrator.
- **Auto**: in this case demand switch will be disabled and MOT button 

should be pressed to enabled the system.
- **Disable**: Teach Pendant cannot be operated.

Three contacts key switch gives the possibility of changing between those three different modes secured with its physical key.

The Key switch can be found in the upper left corner of both real and virtual teach pendant and it has the following outlook:

Figure 2-10: Key switch real Teach Pendant

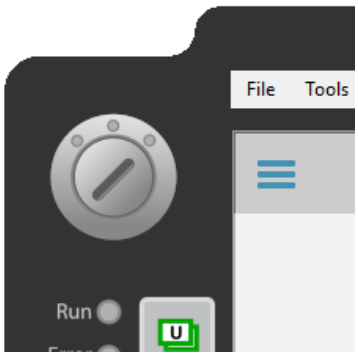
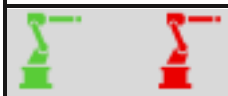



Figure 2-11: Key switch virtual Teach Pendant

All these safety components have an icon status representation in TPS. Once wired and configured in TPS (shown in section 7.2 IO Configuration) the icons will show the status of the buttons as follows:

| | |
|---|---|
|  | Manual, Auto, Disabled mode |
|  | Estop released, Estop pressed |
|  | Drive enabled, Drive disabled |
|  | Demand button released; Demand button pressed |

# 3 Declaration of conformity

This declaration of conformity is issued under the sole responsibility of the manufacturer.

We declare that the following product is in conformity with the essential requirements of the following European Council Directives.

The object of the declaration described above is in conformity with the relevant European Union harmonisation legislation in accordance with BS EN 50581:2012 and BS EN 17050-1:2010.

- Directive 2011/65/EU of the European Parliament and of the Council of 8 June 2011 on the restriction of the use of certain hazardous substances in electrical and electronic equipment
- Directive 2014/30/EU of the European Parliament and of the Council of 26 February 2014 on the harmonisation of the laws of the Member States relating to electromagnetic compatibility.
- EN55032:2015/AC: 2016 - Electromagnetic compatibility of multimedia equipment. Emission Requirements.
- EN55024:2010 - Information technology equipment. Immunity characteristics. Limits and methods of measurement.
- IEC 61000-4-2, 2nd Edition, Dec 2008 - Electromagnetic compatibility (EMC) - Part 4-2: Testing and measurement techniques - Electrostatic discharge immunity test.
- IEC 61000-4-3, 3.2 Edition, Console with AMD 2, April 2010 - Electromagnetic compatibility (EMC) - Part 4-3: Testing and measurement techniques - Radiated, radio-frequency, electromagnetic field immunity test.
- IEC 61000-4-4, 3rd Edition, Apr 2012 - Electromagnetic compatibility (EMC) - Part 4-4: Testing and measurement techniques - Electrical fast transient/burst immunity test.
- IEC 61000-4-5, 3rd Edition, May 2014 - Electromagnetic compatibility (EMC) - Part 4-5: Testing and measurement techniques - Surge immunity test.
- IEC 61000-4-6, 4th Edition, Oct 2013 - Electromagnetic compatibility (EMC) - Part 4-6: Testing and measurement techniques - Immunity to conducted disturbances, induced by radio-frequency fields.
- IEC 61000-4-8, 2nd Edition, Sept 2009 - Electromagnetic compatibility (EMC) - Part 4-8: Testing and measurement techniques - Power frequency magnetic field immunity test.

**The notified body:** Compliance Certification Services Inc. Kunshan Laboratory, Kunshan City, Jiangsu, China, duly issued the CE EMC Test certificate.

**Report Number:** C180913E14-ET

**Dated:** 27th September 2018

# 4 TPS: Teach Programming System

## 4.1 Virtual Teach Pendant casing

Virtual Teach Pendant with compact casing and key switch, E-Stop and demand buttons can simulate the same behaviour as real Teach Pendant. Key switch, E-Stop and demand buttons can be configured with a virtual or real output. In the case of real controller, those outputs should be wired to the inputs selected for the real buttons, so the actions will be triggered either with real or virtual Teach Pendant. In the case of Simulator controller, virtual inputs and outputs are assigned to the same numbers, triggering the correct actions using inputs or outputs.

The output configuration in virtual Teach Pendant can be found in the left top menu under “Tools -> Options”. Then, under “Casing” tab, “Compact casing for HTS-A013-1C6 with key, emergency and demand button” casing option can be selected. “Enable I/O mappings” has to be selected as well, along with the desired output numbers.

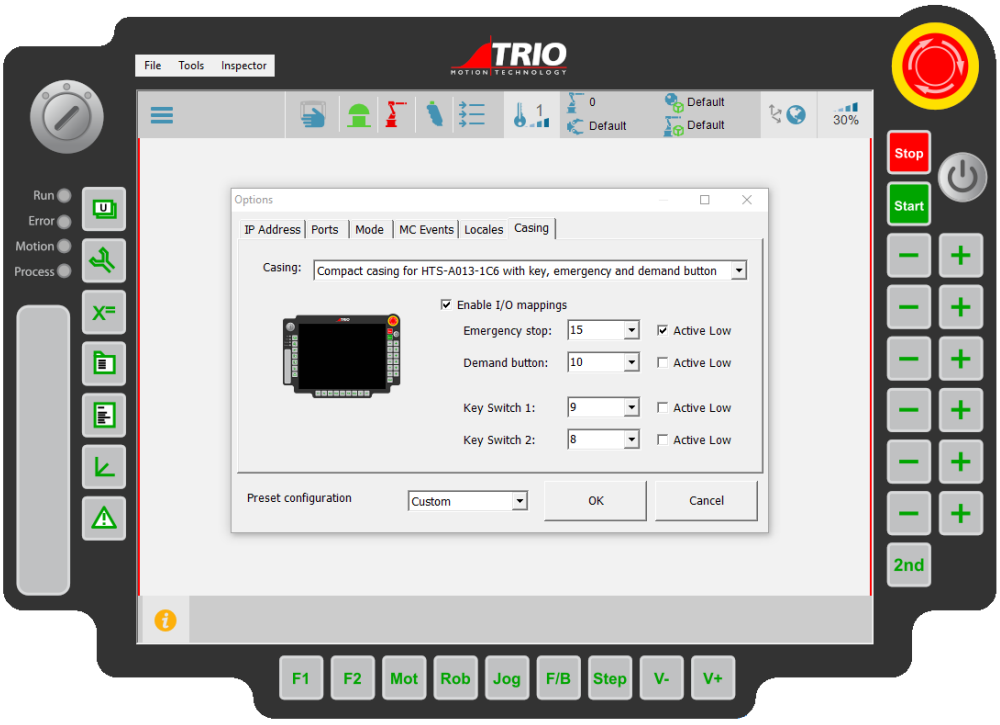



Figure 4-1: virtual pendant output mappings

The TPS needs to know what inputs have been configured for the safety components, for both cases real or virtual ones. For more details in how to do this configuration please, refer to IO config section, page 73 of this document.

## 4.2 Keypad

Figure 4-2: menu keypad

| | | |
|---|---|---|
| | User interface | Enter the home page of user defined interface |
| | Settings | Enter the settings interface page |
| | | |
| | Program list | Prompt the list of programs in the current project |
| | Program editor | Enter the program editor page |
| | Positions | Prompt axis position window |
| | Messages / Log | Prompt the messages and log interface |


Figure 4-3: function keypad

| | | |
|---|---|---|
| F1 | | |
| F2 | | |
| Mot | Motor enable | Servo ON and OFF (Auto mode only) |
| Rob | Robot selection | Select different robot |
| Jog | Coordinate system selection | Change the jog coordinate system |
| F/B | | |
| Step | Operation mode | Change the program running mode between step and continuous mode |
| V- | Jog speed minus | Decrease the global jog speed |
| V+ | Jog speed plus | Increase the global jog speed |


Figure 4-4: axis keypad

| | | |
|---|---|---|
| Stop | Stop program | Stop program, robot execution and clear move buffers. Refer to section Program state manipulation for more information about stop program. |
| Start | Execute program | Start program execution. Refer to section Program state manipulation for more information about the different execution modes. |
| − | Jog axis negative | Under servo ON condition, jog axis backward |
| + | Jog axis positive | Under servo ON condition, jog axis forward |
| 2nd | | |

## 4.3 Home screen

System status windows will show warnings and errors that happens in every robot axis. It will show the axis number, error code and the message itself.

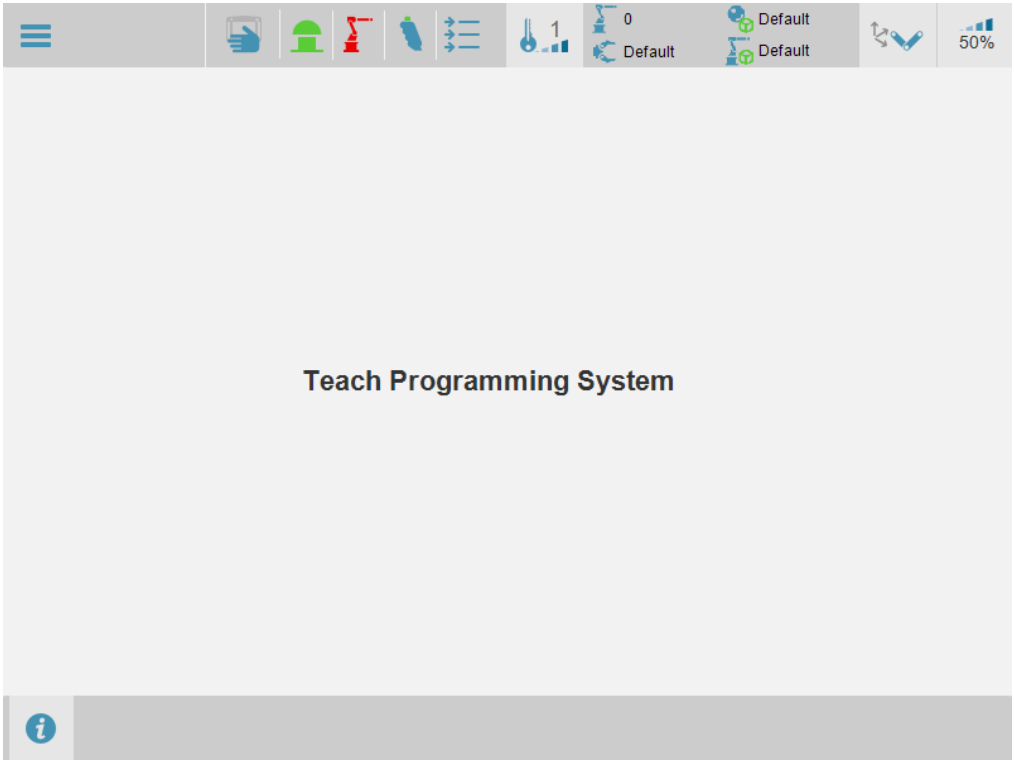



Figure 4-5: Home page

## 4.4 Warning / error window

If an error or warning exists, system log and status button will change its colour to yellow warning.

### 4.4.1 Status

The status is divided in System and Joints. General System errors or warnings will be shown under System tab. Individual Joint errors or warnings will be shown under each Joint tab.

Errors can try to be cleared by pressing refresh button.

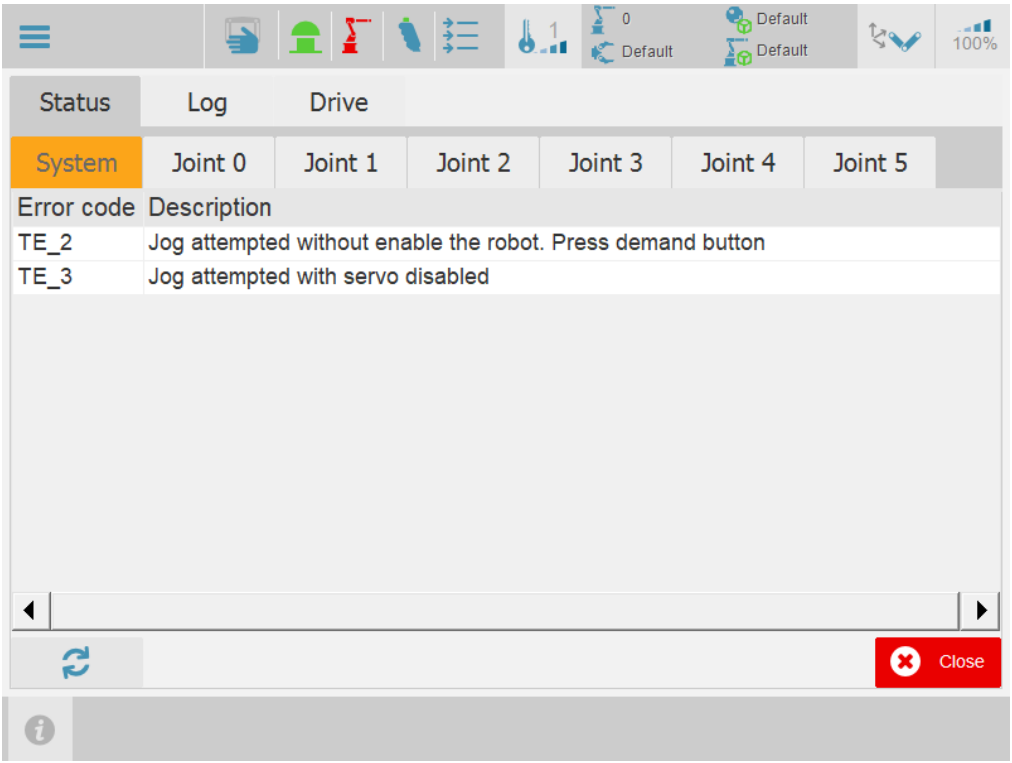



Figure 4-6: System status window

Please, refer to Error and warning codes section for a full list.

### 4.4.2 Log

The log window will show and store warning and error messages. Up to 2048 messages will be stored in flash memory. The messages will be compound by date and time of message generation, number of seconds the controller is active and the message itself.

The whole group of messages can be cleared by clicking trash button.

A special log entry is stored every time the controller is powered up, such as:

600:00014@1ms [0.05 2.0296000 $0000 $0c000003]

600 = Controller type

00014 = Controller serial number

@1ms = Servo period cycle time

0.05 = Bootloader version

2.0296000 = Firmware version

$0000 = FPGA number

$0c000003 = Feature Enabled Codes

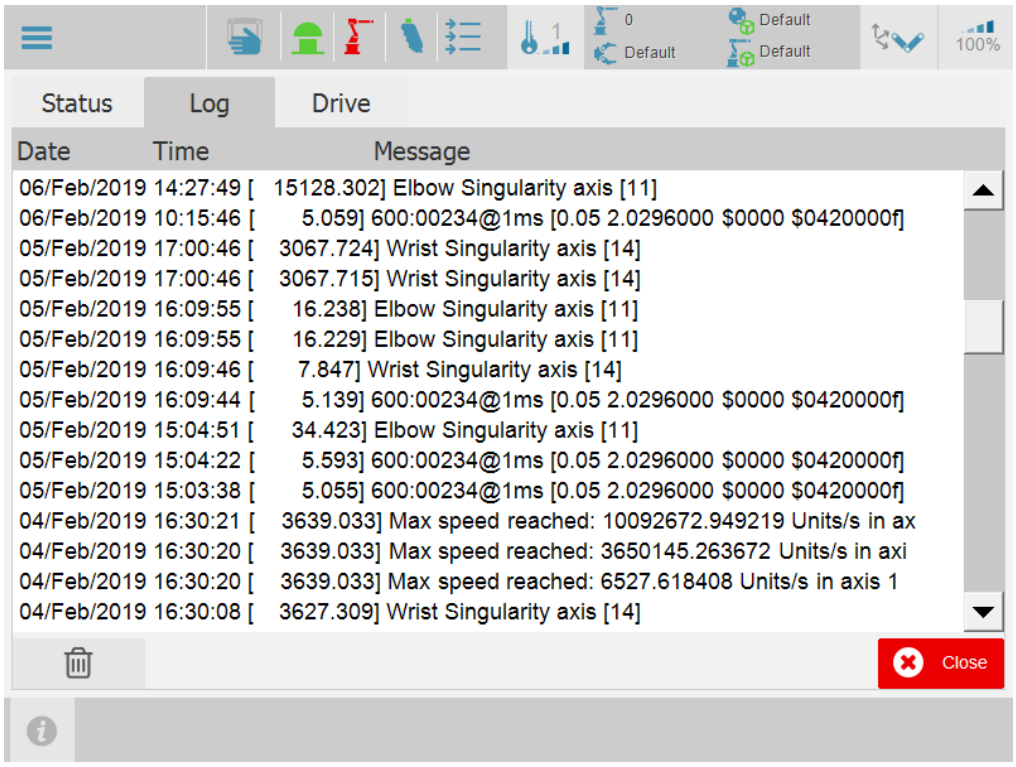



Figure 4-7: System log window

### 4.4.3 Drive

Users can develop their own logic to catch Drive errors and send messages to the teach pendant under Drive tab.

To show a user drive error on teach pendant, VR(DriveError) has to be = 1.

To reset it to its normal status, VR(DriveError) has to be = 0.

To trigger the events, VR(DriveMessageTrigger) has to be incremented.

The messages have to be stored in VRs and have been divided by axis: VR(DriveMessageAxis1), VR(DriveMessageAxis2), VR(DriveMessageAxis3), VR(DriveMessageAxis4), VR(DriveMessageAxis5) and VR(DriveMessageAxis6).

To clear the messages from the Drive window, the refresh button has to be pressed. This action will turn the flag VR(DriveErrorReset) = 1. The user will have to handle in his logic the drive error reset procedure and clear all VRs error messages. The window will be clear automatically after press refresh button, so VR(DriveMessageTrigger) should be triggered if some errors persist after the reset procedure.

*Example:*

```
INCLUDE "RPS_A_INITIALISE_VARIABLES"

'Trigger error flag, so it will highlight status icon on pendant
VR(driveerror) = 1
'Set error messages:
VR(drivemessageaxis1) = "Error in drive 1"
VR(drivemessageaxis3) = "Error in drive 3"

'Trigger drive errors:
VR(drivemessagetrigger) = VR(drivemessagetrigger) + 1

'Reset errors:
'press refresh button -> VR(DriveErrorReset) = 1
IF VR(driveerrorreset) = 1 THEN
    VR(driveerrorreset) = 0

    'Reset logic:
    'error drive 1 persist
    VR(drivemessageaxis1) = "Error in drive 1 persist"
    'error drive 3 cleared
    VR(drivemessageaxis3) = 0

    'Trigger drive errors
    VR(drivemessagetrigger) = VR(drivemessagetrigger) + 1
ENDIF
```

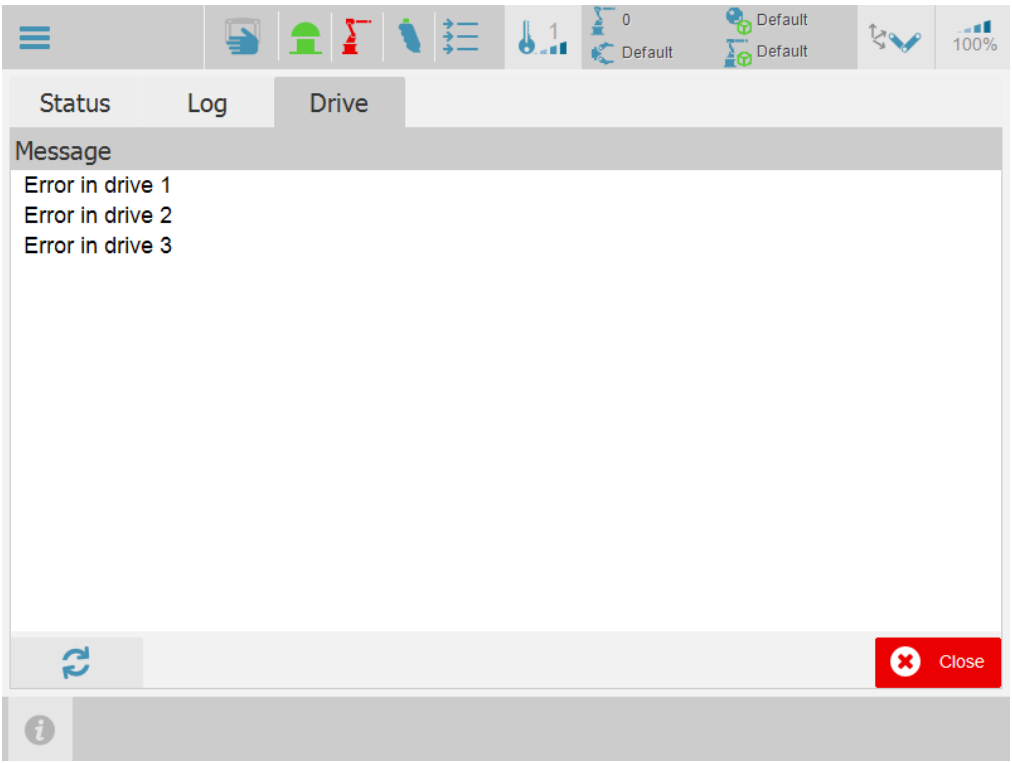



Figure 4-8: Drive error window

## 4.5 Speed

Jog speed can be set in percentages by Speed window.

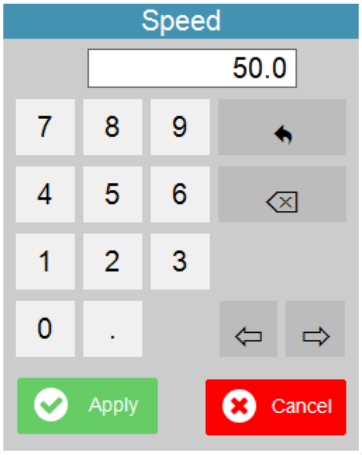



Figure 4-9: Speed window

## 4.6 Jog modes

There are five different jog modes selectable sequentially by pressing Jog mode button. The button will change its icon accordingly with selected mode.

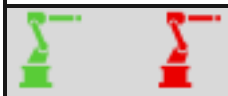
| | | |
|---|---|---|
|  | Joint | Each axis moves independently. The robot arm moves around joint under demand range. The position of end-effector is determined by joint orientation and position. |
|  | World | The end-effector moves straight along the world coordinate system. The orientation uses extrinsic rotations. All the position of end-effector is defined by world coordinate. It is possible that singularity point will occur when using world coordinate. |
|  | Base | The end-effector moves straight along the base coordinate system. The orientation uses extrinsic rotations. The origin point of the system is on the robot base. |
|  | Tool | The end-effector moves straight along the tool coordinate system. The orientation uses intrinsic rotation. The origin point of the system is on the tool like gripper. |
|  | Object Frame | The end-effector moves straight along the active object frame. The orientation uses extrinsic rotations. |

## 4.7 Status bar

Status bar will show the robot, tool, object frame and robot frame selected. All of them are selectable in their specific pages.


Figure 4-10: status bar

It is also shown the status of pendant (manual, auto, disabled), Estop, Servo and Demand buttons.

| | |
|---|---|
|  | Manual, Auto, Disabled mode |
|  | Estop released, Estop pressed |
|  | Drive enabled, Drive disabled |

| | |
|---|---|
|  | Demand button released; Demand button pressed |
|  | Step mode |
|  | Continuous mode |

## 4.8 Behaviour in different modes

The behaviour of the robot will change depending on what mode it is being operated in. The behaviour is listed below.

**Manual Mode:** In this mode, Jogging is allowed. The speed of the robot will be limited to WORLD_POS_REDUCED_SPEED and WORLD_ORI_REDUCED_SPEED and the global override will be applied to the reduced speeds. The robot will be enabled only using the demand button. The button needs to be held pressed.

**Auto Mode:** In this mode, Jogging is prohibited. The speed of the robot will only be limited be the set maximum speed and the global override will be applied to the program speed. The robot will be enabled by closing the safety door (if that option is enabled) and pressing the MOT button.

**Disabled Mode:** In this mode, jogging and running programs is prohibited. The robot will not enable in this mode.

## 4.9 User levels

The system will enable or disable different features depending on the selected user level. There are four user levels as described below:

**Level 1:** Highest level with all authority in the system, all the functions and features are available. It has been designed mainly for developers.

**Level 2:** Manufacturing level. For now, level 2 and level 1 share almost the same authority in the system.

**Level 3:** Customer engineering level. In this level users will not be allowed to configure system information or change system configuration.

**Level 4**: This level is designed for operators who work around the robot. Only run edited programs and start/stop/shut down the system or programs functions are available.

| Feature | Level 1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|
| Status and log | Y | Y | Y | Y |
| Jog Speed | Y | Y | Y | N |
| Jog Modes | Y | Y | Y | N |
| GTAs | Y | Y | Y | N |
| Tools Dimensions | Y | Y | Y | N |
| Tools Collision | Y | Y | Y | N |
| Object Frames | Y | Y | Y | N |
| Robot Frames | Y | Y | Y | N |
| Collision Objects | Y | Y | Y | N |
| Project Manager | Y | Y | Y | Y |
| Program Editor(Debug Buttons) | Y | Y | Y | Y |
| Program Editor(Program edition Buttons) | Y | Y | Y | N |
| Program Types and Projects | Y | Y | Y | Y |
| Instructions Set | Y | Y | Y | N |
| Settings | Y | Y | N | N |
| IOs settings | Y | Y | N | N |

User level status. It is a button that shows the current user level and shows the user level window when pressed.

Figure 4-11: user level status button

The levels can be set through user level window. Higher level can set a lower level without password. To set a higher level than the current one the correct password must be set.

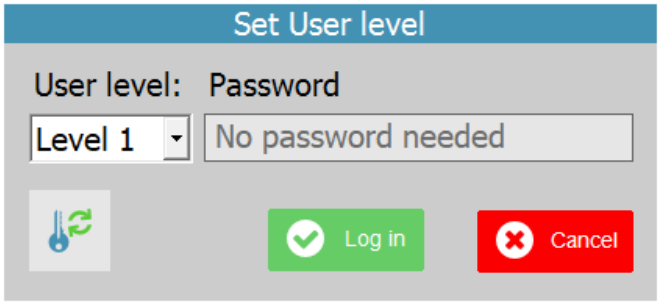



Figure 4-12: user level window

To change the password of a level, the button  must be pressed.

It is only possible to change the password of a level from its level or higher.

Default passwords are as follows:

- Level 1: level1
- Level 2: level2
- Level 3: level3
- Level 4: level4

Characters allowed are the ASCII table.

To reset all passwords to default values, COORDINATOR_DATA(68) instruction has to be executed over the terminal through Motion Perfect.

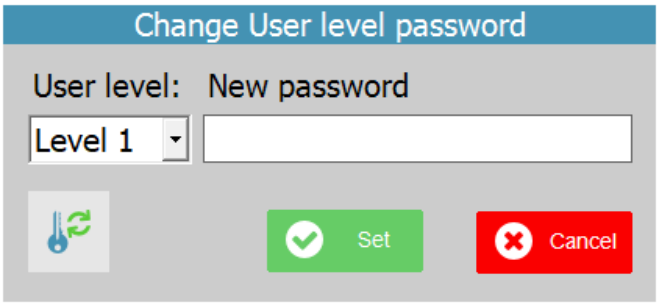



Figure 4-13: change level password window

## 4.10 Main menu

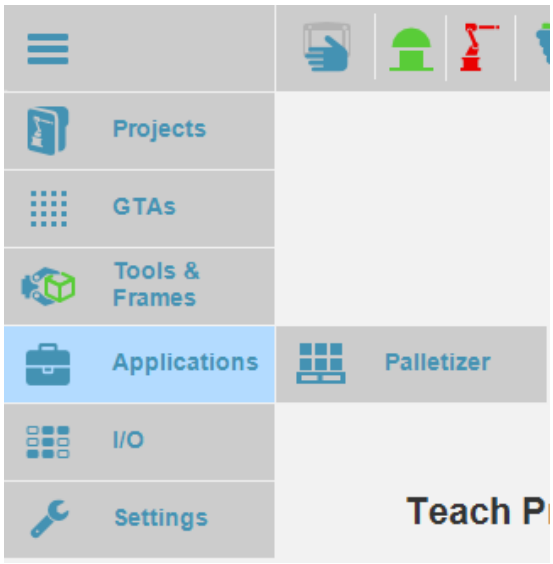
Main menu button will drop the main menu:


| |
|---|
| Projects manager and program editor |
| GTA page |
| Tools and Frames menu |
| Applications menu |
| Inputs / outputs page |
| Setting page |

Projects:

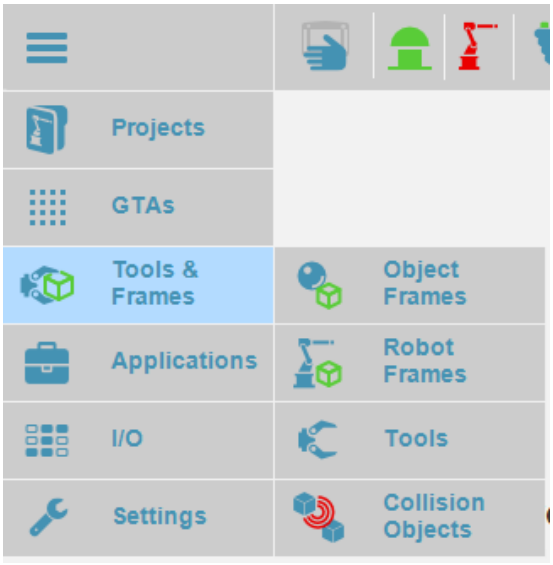

| |
|---|
| Project manager window |
| Program editor page |
| Function editor page |

Applications:



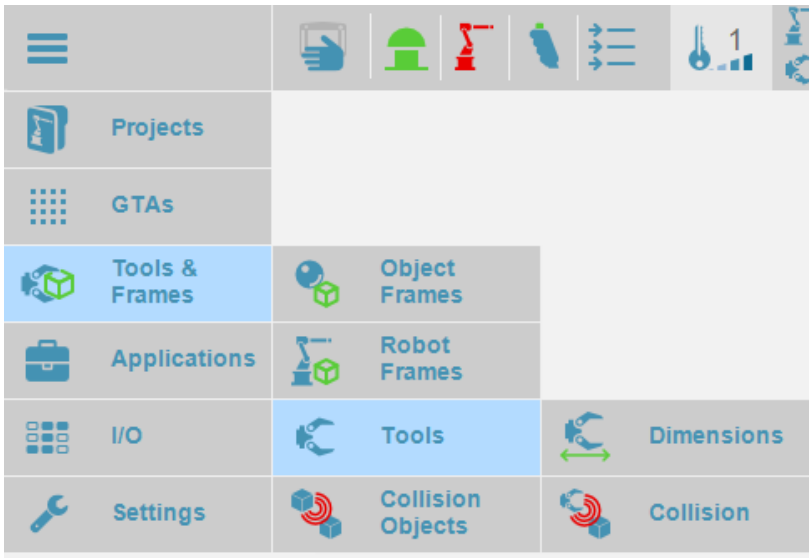

| |
|---|
| Palletizer page |

Tools, frames and collision objects:




| |
|---|
| Object Frames page |
| Robot Frames page |
| Tools menu |
| Collision Objects page |

Tools dimensions and collision:




| |
|---|
| Tools dimension page |
| Tools collision page |

## 4.11 GTAs / GTAJs

GTA is the Global Targets Array which contains an array of globally available TARGET points.

It stores information of position and orientation in 3D space in the case of GTA and joint positions in degrees for GTAJ. The TARGET data type represents a set of 6 values:

GTA (white back ground):

- X, Y, Z – for the coordinates of the point in 3D space in millimetres
- U, V, W – for the angular orientation in degrees

GTAJ (orange back ground):

- X, Y, Z, U, V, W – Up to six angular joint positions

An array of 1000 GTAs is available for use in all programs. In addition to the 6 coordinates GTAs can have name assigned which can be used to reference them in programs.

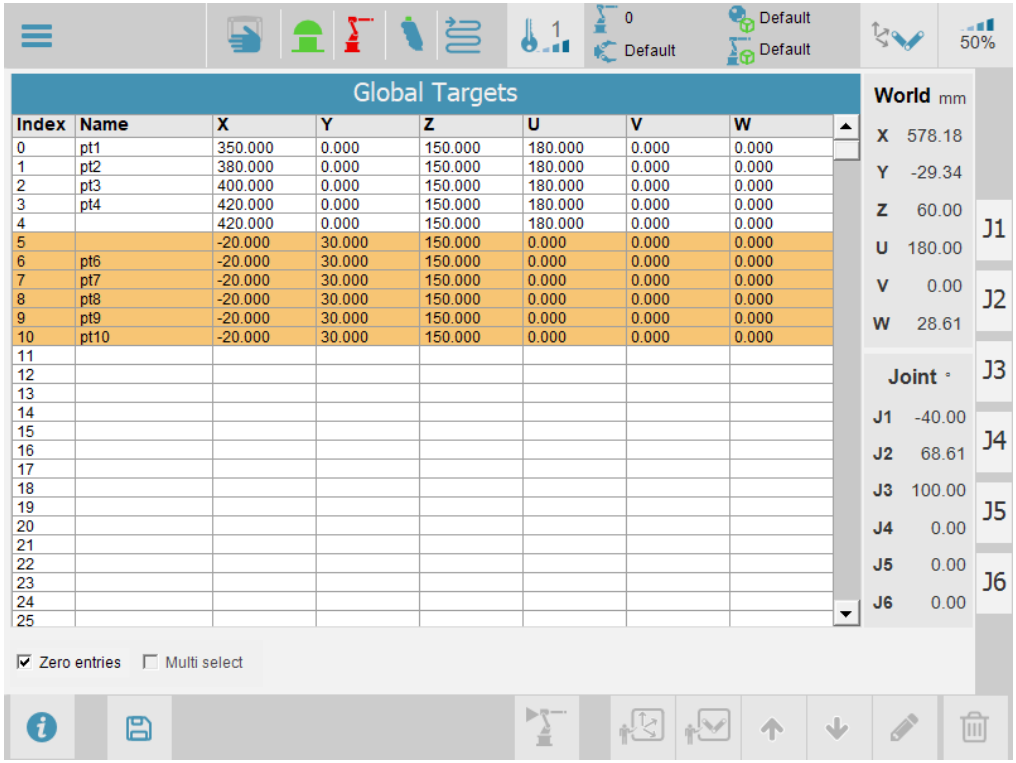



Figure 4-14: GTA page

| | |
|---|---|
| | Save |

On this screen you can see the range of GTAs that are in controller flash memory at the moment of entry in this page. All the changes will be done in flash memory, so GTAs will never be lost even if controller firmware is updated.

It is possible to save the whole table in the robot basic file “ROBOT_GLOBAL_TARGETS” (the program will be overwritten with the new values). If a GTA is set by another program while GTA page is active, it will be refreshed automatically.

| | |
|---|---|
|  | Move to GTA |

Move to GTA button will move the robot to the selected GTA with the active tool and frames, at the selected speed and mode. It will only be active in manual mode.

| | |
|---|---|
|  | Teach Cartesian |

Teach cartesian button will store the current cartesian position in the selected GTA with a given name. The position can be changed by jogging the robot with jog buttons of the Teach Pendant.

| | |
|---|---|
|  | Teach Cartesian |

Teach joints button will store the current joints position in the selected GTAJ with a given name. The position can be changed by jogging the robot with jog buttons of the Teach Pendant.

| | |
|---|---|
|  | Shift up |
|  | Shift down |

Shift up and down buttons will effectively shift a selected GTA on position in the list. This will change the index of the GTA.

| | |
|---|---|
|  | Edit |

Edit button will prompt an editor by which the operator can directly type-in a new GTA or modify the coordinates or the name of a previously defined entry.

| | |
|---|---|
|  | Delete |

It is possible to delete a GTA (or a range of GTAs by having *Multi select* check box checked) by clicking in delete button. That entry will become empty in controller flash memory and GTAs screen but controller program will still be having the entry until save button is pressed. This process will not delete entries declared out of “ROBOT_GLOBAL_TARGETS” robot basic file.

Zero entries check box will collapse or expand the empty entries for a more compact representation.

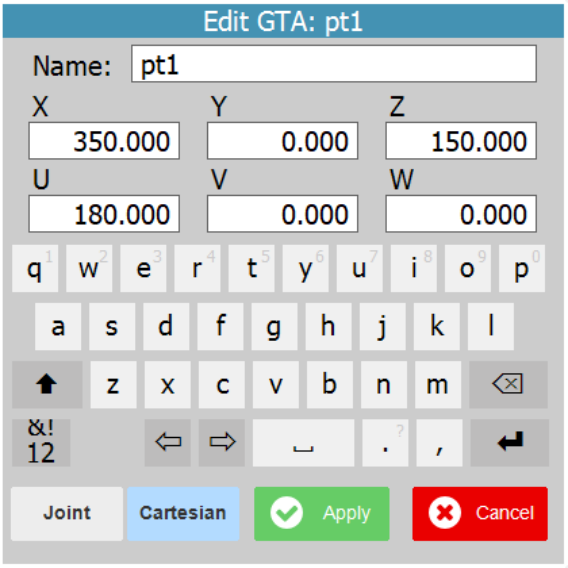



Figure 4-15: Edit GTAs window

## 4.12 Tools dimensions

The tool offset is a transformation between the end-effector and the Tool Centre Point. It sets a distance and orientation of a tool from the end effector to the TCP.

Similar to the target points it is presented as a set of 6 values:

- X, Y, Z – for the coordinates of the offset in millimetres
- U, V, W – for the angular orientation of the tip in degrees.

An array of 31 tool definitions is available for use to all programs. Unique name can be assigned to each tool to be used to identify and reference it in programs.

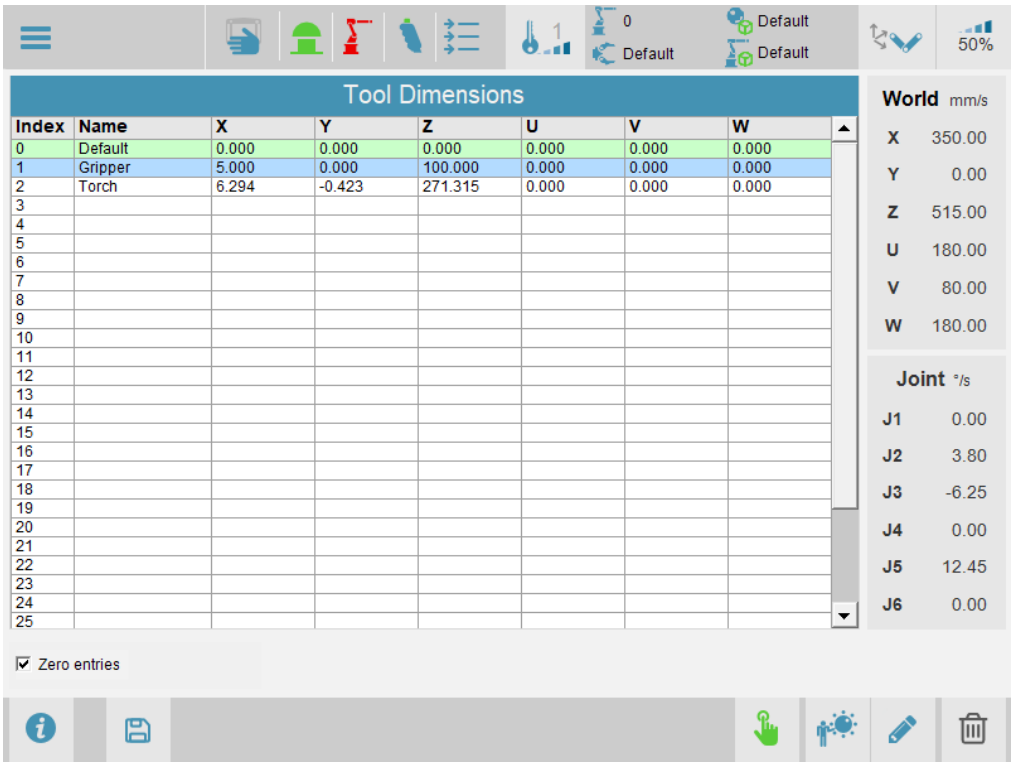



Figure 4-16: Tools page

| | |
|---|---|
| | Save |

On this screen you can see the range of Tools that are in controller flash memory at the moment of entry in this page. All the changes will be done in flash memory, so Tools will never be lost even if controller firmware is updated.

It is possible to save the whole table in the robot basic file “ROBOT_TOOLS_AND_FRAMES” (the program will be overwritten with the new values). If a Tool is set by another program while Tools page is active, it will be refreshed automatically.

| | |
|---|---|
| | Select |

It is possible to activate the selected Tool by clicking Select button. The selected tool will be highlighted in green colour. Tool offset 0 is active by default.

| | |
|---|---|
| | Edit |

Edit button will prompt an editor by which the operator can directly type-in a new Tool dimension or modify the coordinates or the name of a previously defined entry.

| | |
|---|---|
| | Delete |

It is possible to delete a Tool (or a range of Tools by having *Multi select* check box checked) by clicking in delete button. That entry will become empty in controller flash memory and Tools screen but controller program will still be having the entry until save button is pressed. This process will not delete entries declared out of "ROBOT_TOOLS_AND_FRAMES" robot basic file.

| | |
|---|---|
| | Calibrate |

Calibrate button will lead to Calibrate page. In this page it will be possible to calculate the dimensions of a tool performing a calibration procedure.

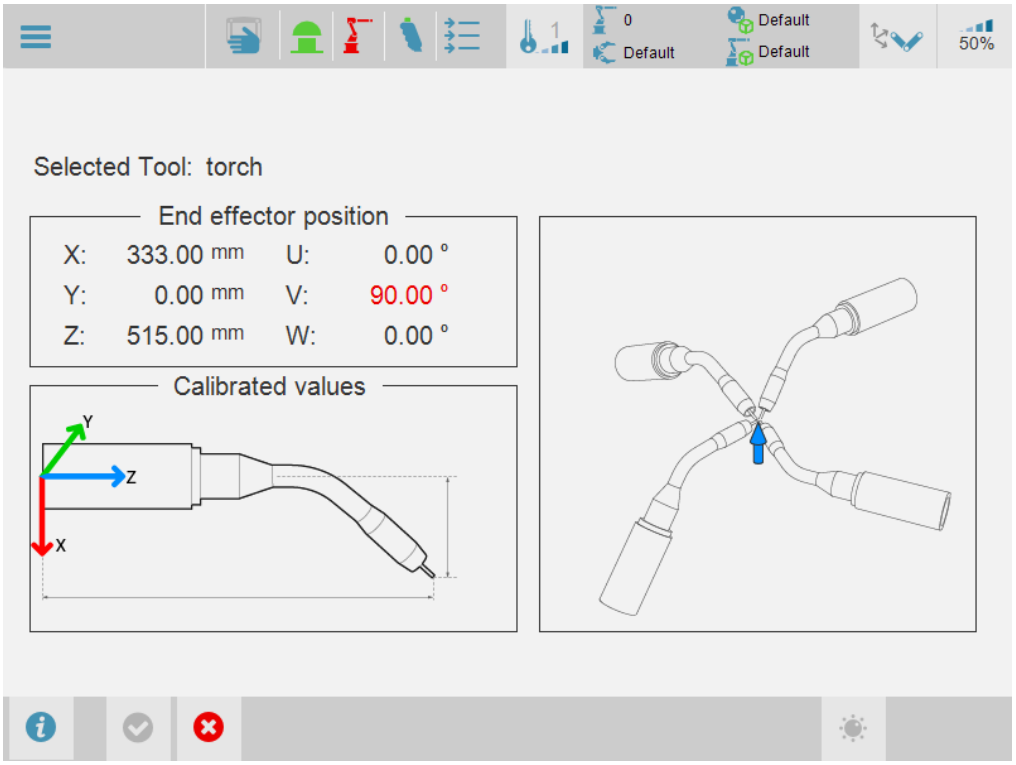



Figure 4-17: Tool calibration page

| | |
|---|---|
| | Calibrate |
| | Accept |
| | Cancel |

During the calibration procedure the robot must be moved to different positions and orientations trying to keep the tip of the tool as close as possible to a pre-selected point in 3D space. Usually a calibration object is fixed in a stable position relative the robot so that it can be used as a reference point.

The operator should jog the robot position and orientation until the tip goes as close as possible to the reference object.

By pressing one of the grey tools images the values will be saved for that position and it will become green, showing that position has been stored.

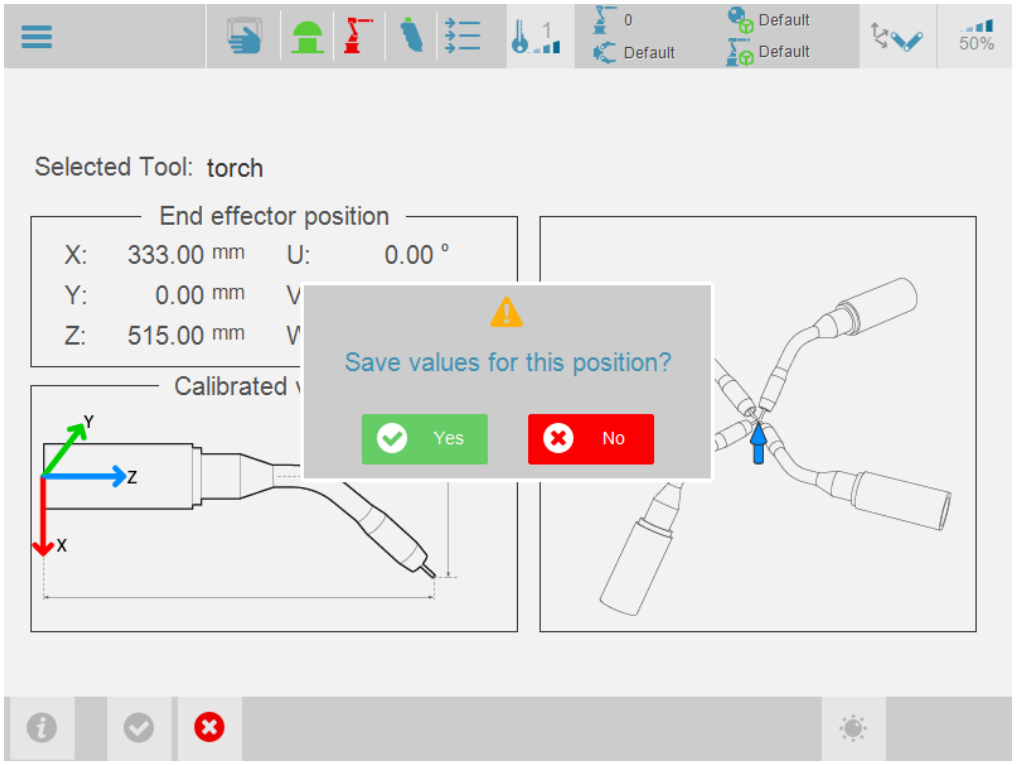



Figure 3-18: Tool calibration save position window

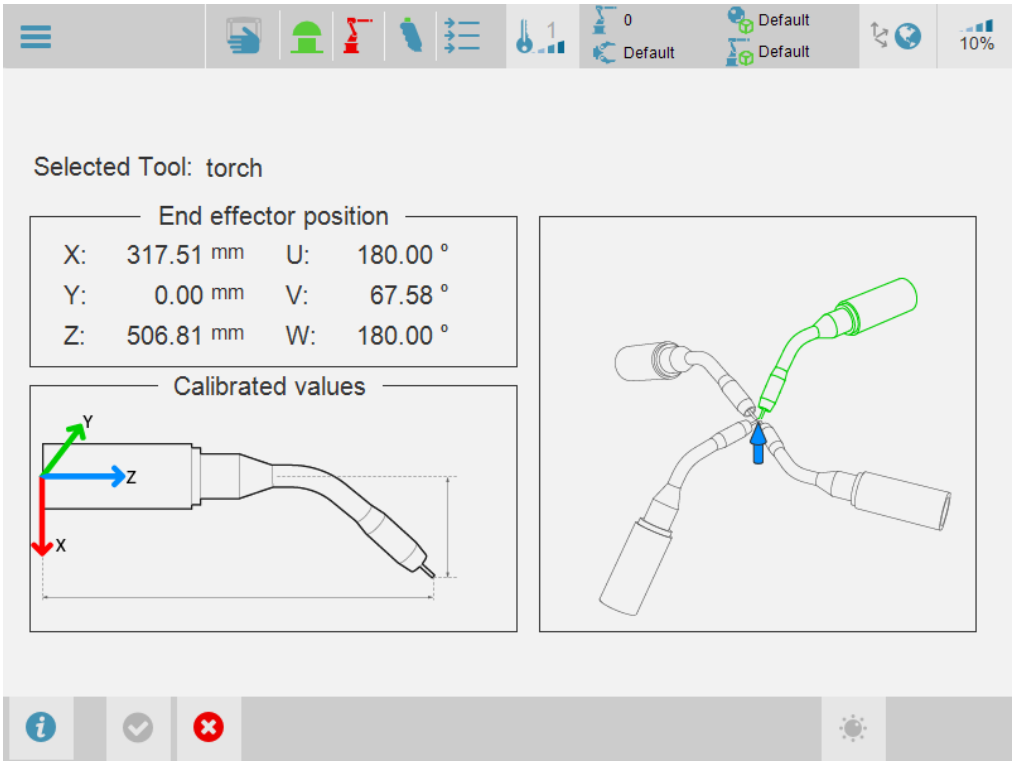



Figure 3-19: Tool calibration one position saved

Then the operator can continue by capturing the next point.

Once four points are captured the system will enable calibration button and, after this button had been pressed, calibration algorithm will attempt to calculate the tool offset based on the input points that are collected.

A deviation value (*Serror*) is displayed as well. It can be used as an indication for the quality of the point and the operator can decide to recapture some of the points in order to improve the calibration.

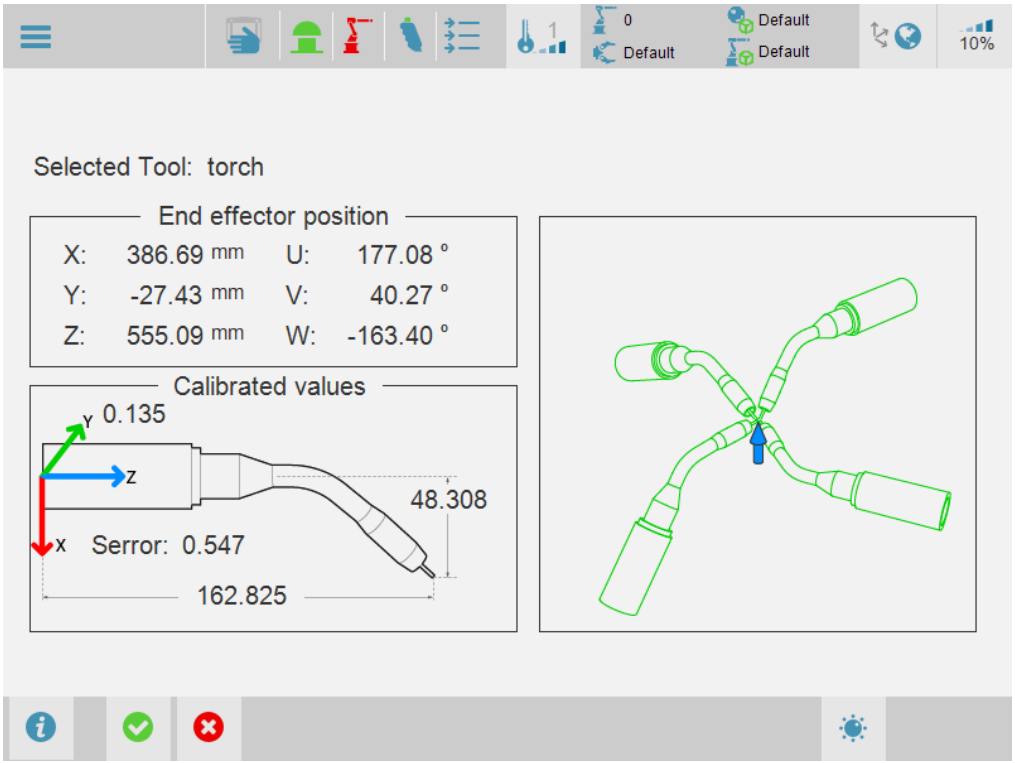



Figure 4-20: Tool calibration process done

By pressing accept button the tool data will be stored to the controller and the new values will appear in Tools page.

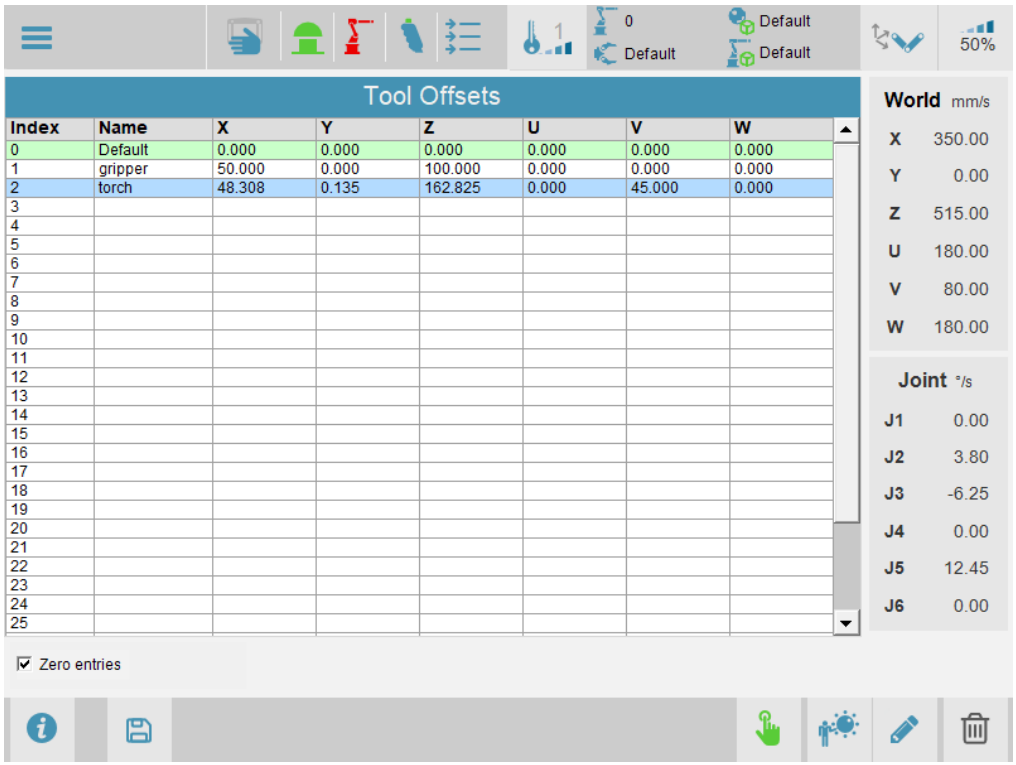



Figure 4-21: Tools calibration process done

## 4.13 Tools collision

The tool collision is an Oriented Bounding Box around the physical tool. It sets the centre position of the OBB and the dimensions needed for the collision algorithm.

It is presented as a set of 6 values:

- CX, CY, CZ – centre position of the OBB on the object in millimetres
- DX, DY, DZ – half distances measured in every vector of the OBB in millimetres.

Tools collision is information stored in tool data, which means rules of tools offset apply here. An array of 31 definitions is available for use to all programs. A unique name can be assigned to each tool to be used to identify and reference it in programs.

For more information please, refer to RPS manual.

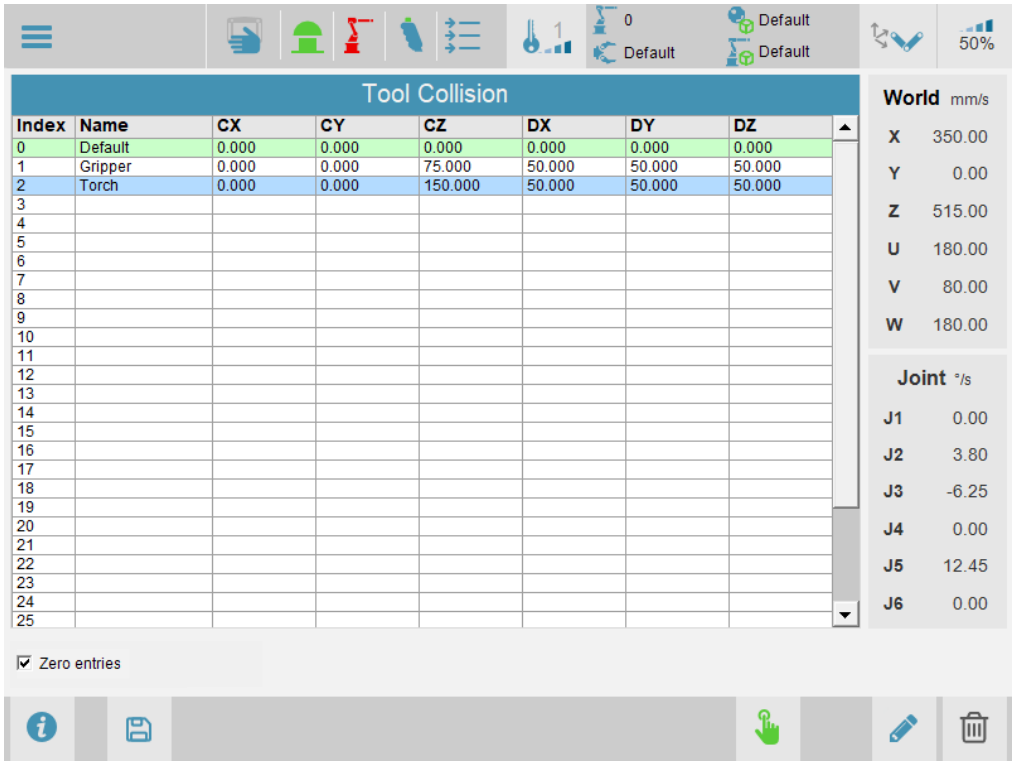



Figure 4-22: Tool Collision page

## 4.14 Object Frames

The object frame is a transformation between the global coordinate system and the coordinate system of an object that is manipulated by the robot. By using object frames it is possible for target points to be defined with respect in the coordinate system of the object. Like the target points it is presented as a set of 6 values:

- X, Y, Z – for the coordinates of the offset in millimetres
- U, V, W – for the angular orientation offset in degrees.

An array of 31 object frame definitions is available for use to all programs. A unique name can be assigned to each object frame to be used to identify and reference it in programs.

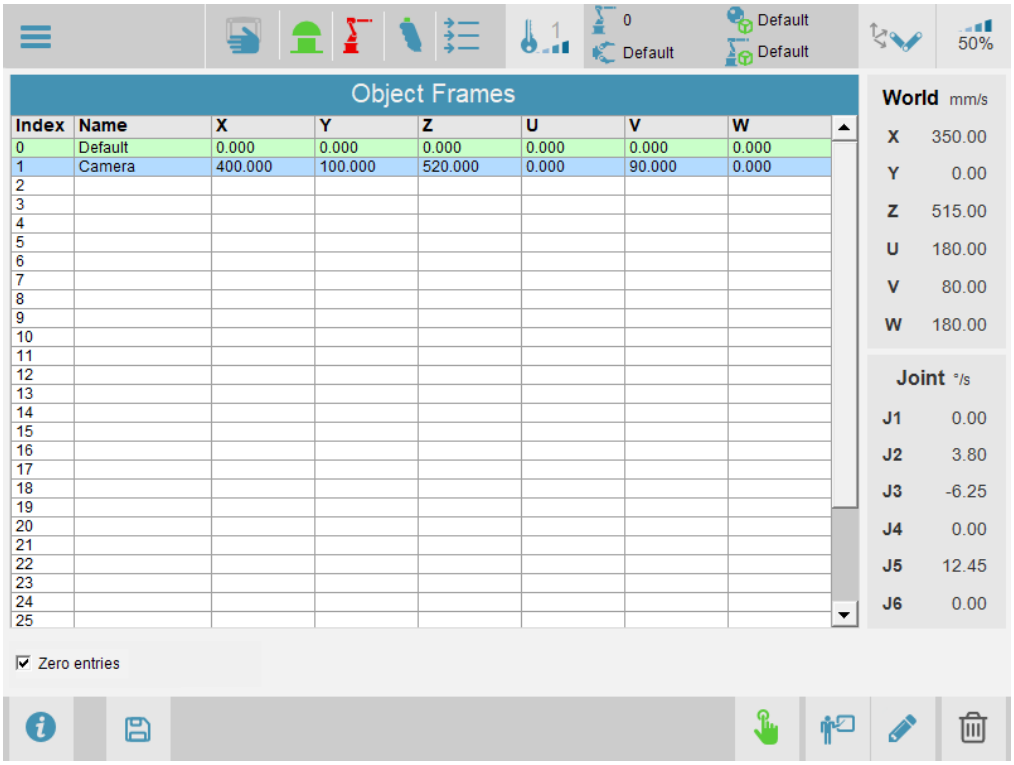



Figure 4-23: Object Frames page

| | |
|---|---|
| | Save |

On this screen you can see the range of Object Frames that are in controller flash memory at the moment of entry in this page. All the changes will be done in flash memory, so Object Frames will never be lost even if controller firmware is updated.

It is possible to save the whole table in the robot basic file “ROBOT_TOOLS_AND_FRAMES” (the program will be overwritten with the new values). If an Object Frame is set by another program while Object Frames page is active, it will be refreshed automatically.

| | |
|---|---|
| | Select |

It is possible to activate the selected Object Fame by clicking Select button. The selected object frame will be highlighted in green colour. Object frame 0 is active by default.

| | |
|---|---|
| | Edit |

Edit button will prompt an editor by which the operator can directly type-in a new Object Frames or modify the coordinates or the name of a previously defined entry.

| | |
|---|---|
| | Delete |

It is possible to delete an Object Frame (or a range of Object Frames by having *Multi select* check box checked) by clicking in delete button. That entry will become empty in controller flash memory and Object Frames screen but controller program will still be having the entry until save button is pressed. This process will not delete entries declared out of "ROBOT_TOOLS_AND_FRAMES" robot basic file.

| | |
|---|---|
| | Teach |

Teach button will prompt the constructor window. In Object Frame constructor window, it will be able to build an Object Frame by teaching three points as follows:

- First point should be at the base or the origin of the object coordinate system.
- Second point should on the X axis of the object coordinate system.
- Third point should on the Y axis of the object coordinate system.

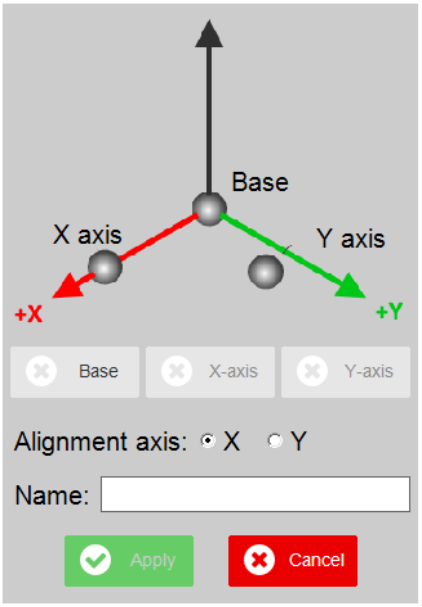



Figure 4-24: construct Object Frame window

Points are captures using Teach Pendant jog buttons.

The operator should adjust the robot position and orientation until the tip goes as close as possible to the desired target point.

The corresponding button changes its state if the captured point is correct.

When three points are captured, the Object Frame is ready to be stored and the operator can select a unique name for it so that it can be referenced in programs.

Edit button will prompt an editor by which the operator can directly type-in a new Object Frame or modify the coordinates or the name of a previously defined entry.

Zero entries check box will collapse or expand the empty entries for a more compact representation.

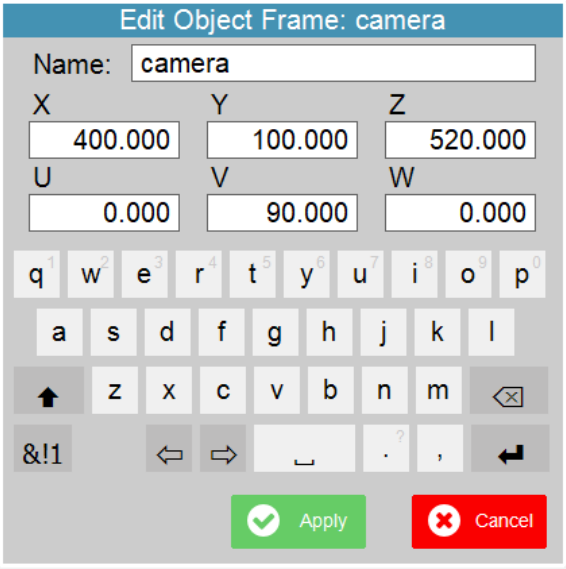



Figure 4-25: Edit Object Frames window

## 4.15 Robot Frames

The robot frame is a transformation between the global coordinate system and the coordinate system of the robot. By using Robot Frames, it is possible for multiple robots to be positioned about a common global coordinate system. Like the target points robot frames are presented as a set of 6 values:

- X, Y, Z – for the coordinates of the offset in millimetres
- U, V, W – for the angular orientation offset in degrees.

An array of 31 robot frame definitions is available for use to all programs. Unique name can be assigned to each robot frame to be used to identify and reference it in programs.

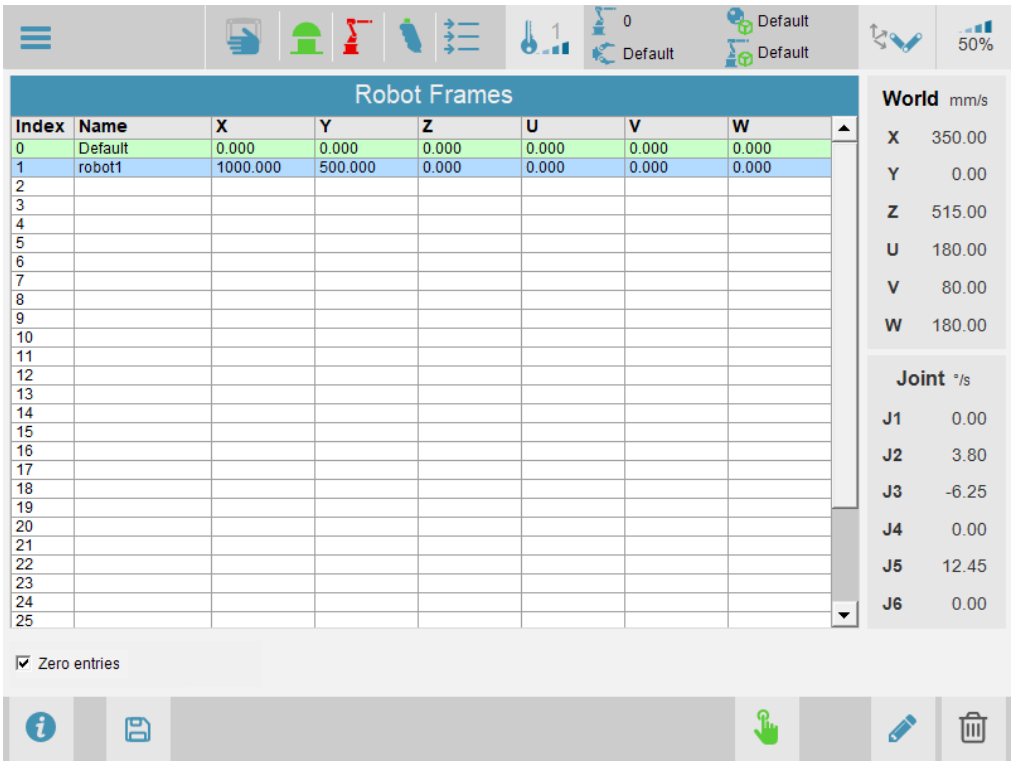



Figure 4-26: Robot Frames page

| | |
|---|---|
| | Save |

On this screen you can see the range of Robot Frames that are in controller flash memory at the moment of entry in this page. All the changes will be done in flash memory, so Robot Frames will never be lost even if controller firmware is updated.

It is possible to save the whole table in the robot basic file “ROBOT_TOOLS_AND_FRAMES” (the program will be overwritten with the new values). If a Robot Frame is set by another program while Robot Frames page is active, it will be refreshed automatically.

| | |
|---|---|
| | Select |

It is possible to activate the selected Robot Fame by clicking Select button. The selected Robot frame will be highlighted in green colour. Robot frame 0 is active by default.

| | |
|---|---|
| | Edit |

Edit button will prompt an editor by which the operator can directly type-in a new Robot Frame or modify the coordinates or the name of a previously defined entry.

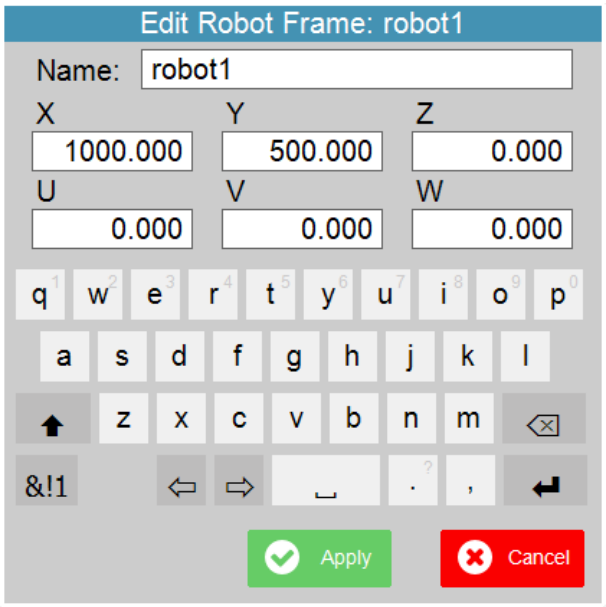



Figure 4-27: Edit Robot Frames window

| | |
|---|---|
| | Delete |

It is possible to delete a Robot Frame (or a range of Robot Frames by having *Multi select* check box checked) by clicking in delete button. That entry will become empty in controller flash memory and Robot Frames screen but controller program will still be having the entry until save button is pressed. This process will not delete entries declared out of "ROBOT_TOOLS_AND_FRAMES" robot basic file.

Zero entries check box will collapse or expand the empty entries for a more compact representation.

## 4.16 Collision objects

Collision objects are an Oriented Bounding Boxes around the physical objects present in the scenario. It sets the centre position and orientation of the OBB and the dimensions needed for the collision algorithm.

It is presented as a set of 9 values:

- X, Y, Z – centre position of the OBB on the object in millimetres
- U, V, W – centre orientation of the OBB on the object in degrees
- DX, DY, DZ – half distances measured in every vector of the OBB in millimetres.

An array of 32 definitions is available for use to all programs. A unique name can be assigned to each object to be used to identify.

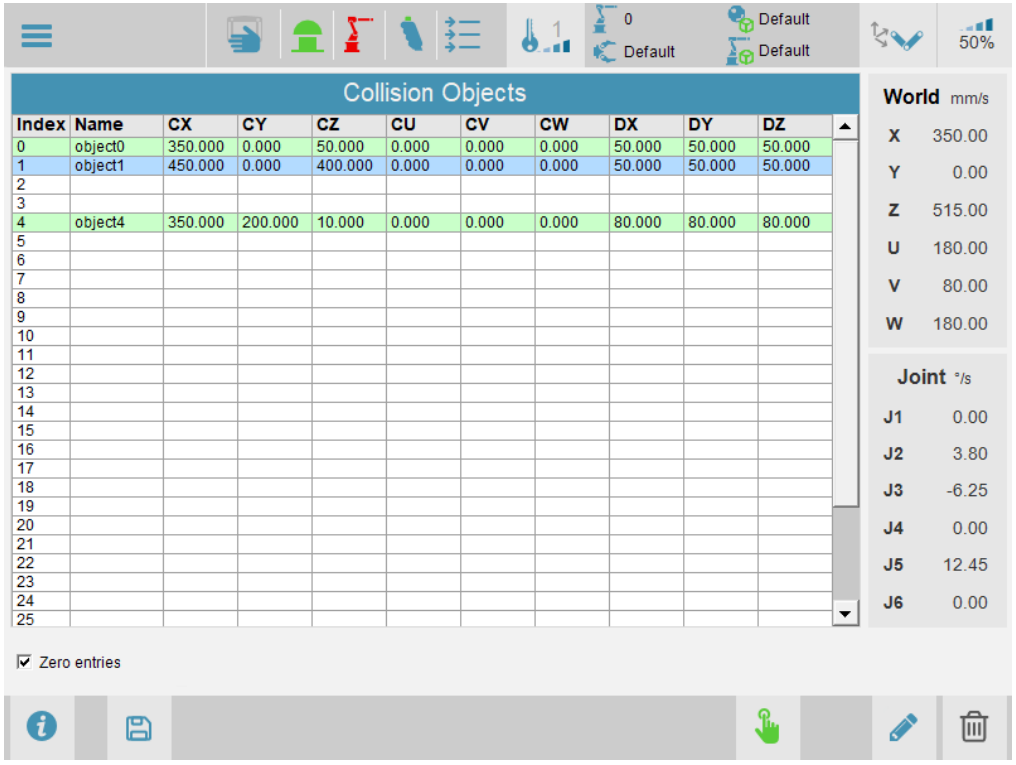



Figure 4-28: Collision Objects.

| | |
|---|---|
| | Save |

On this screen you can see the range of Collision Objects that are in controller flash memory at the moment of entry in this page. All the changes will be done in flash memory, so Collision Objects will never be lost even if controller firmware is updated.

It is possible to save the whole table in the robot basic file “ROBOT_TOOLS_AND_FRAMES” (the program will be overwritten with the new values). If a Collision Objects is set by another program while Collision Objects page is active, it will be refreshed automatically.

| | |
|---|---|
| | Select |

It is possible to activate the selected Collision Objects by clicking Select button. The selected Collision Objects will be highlighted in green colour.

Although is possible to define 32 Collision Objects, just 10 can be active at the same time.

Each object is activated per robot defined. It means an object can be active for a robot and not for other existing one in the system.

| | |
|---|---|
| | Edit |

Edit button will prompt an editor by which the operator can directly type-in a new Collision Objects or modify the coordinates or the name of a previously defined entry.

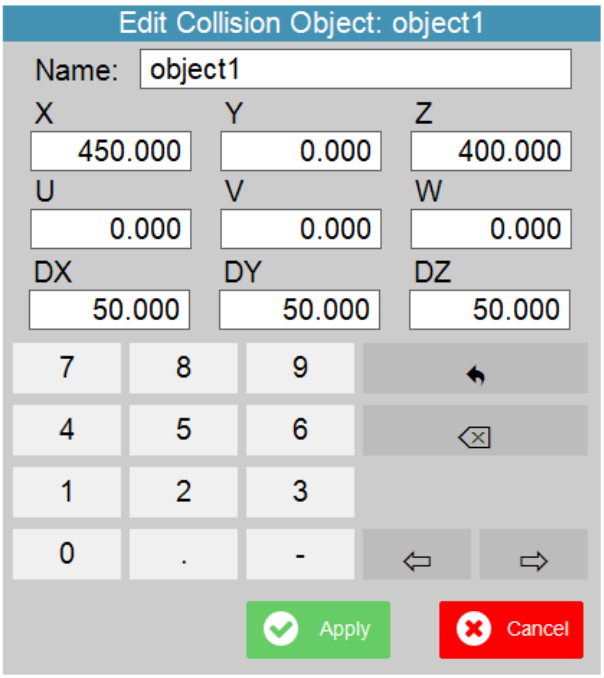



Figure 4-29: Edit Collision Object window.

| | |
|---|---|
| | Delete |

It is possible to delete a Collision Object (or a range of Collision Objects by having *Multi select* check box checked) by clicking in delete button. That entry will become empty in controller flash memory and Collision Objects screen but controller program will still be having the entry until save button is pressed. This process will not delete entries declared out of "ROBOT_TOOLS_AND_FRAMES" robot basic file.

## 4.17 Applications: Palletizer

This function can create a palletizer. It separates the building process into several steps.

The first step builds a plane with size of three dimensions.

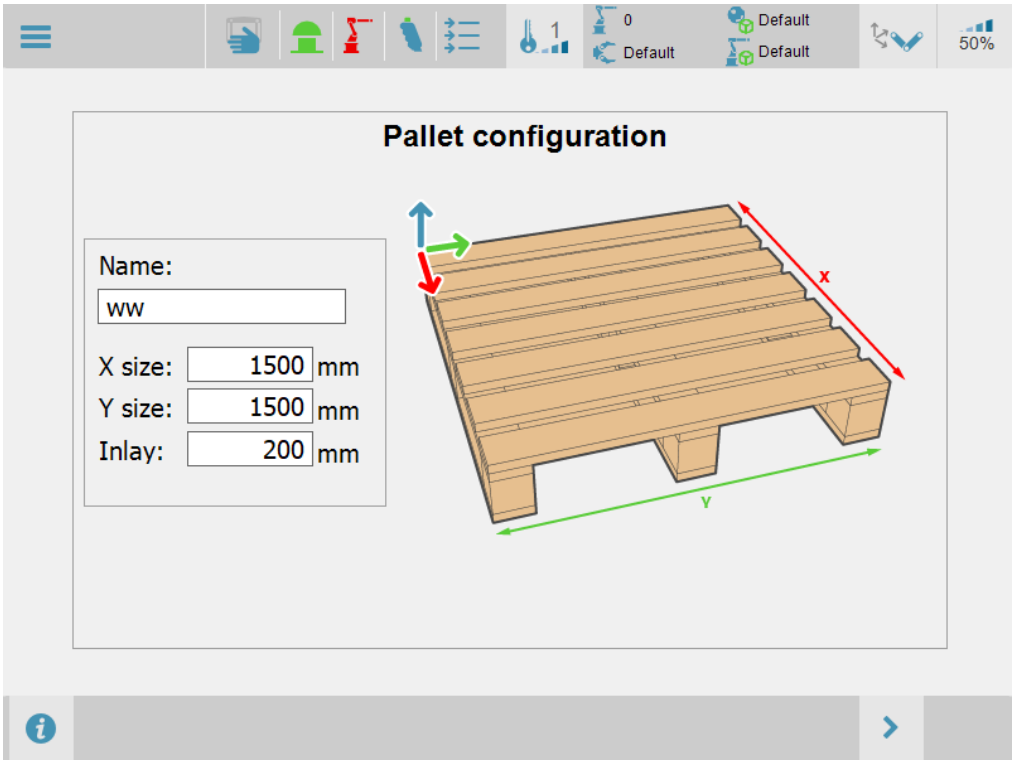



Figure 4-30: pallet configuration window.

The second step is to design item size and position. The dimension of item and position of tool target.

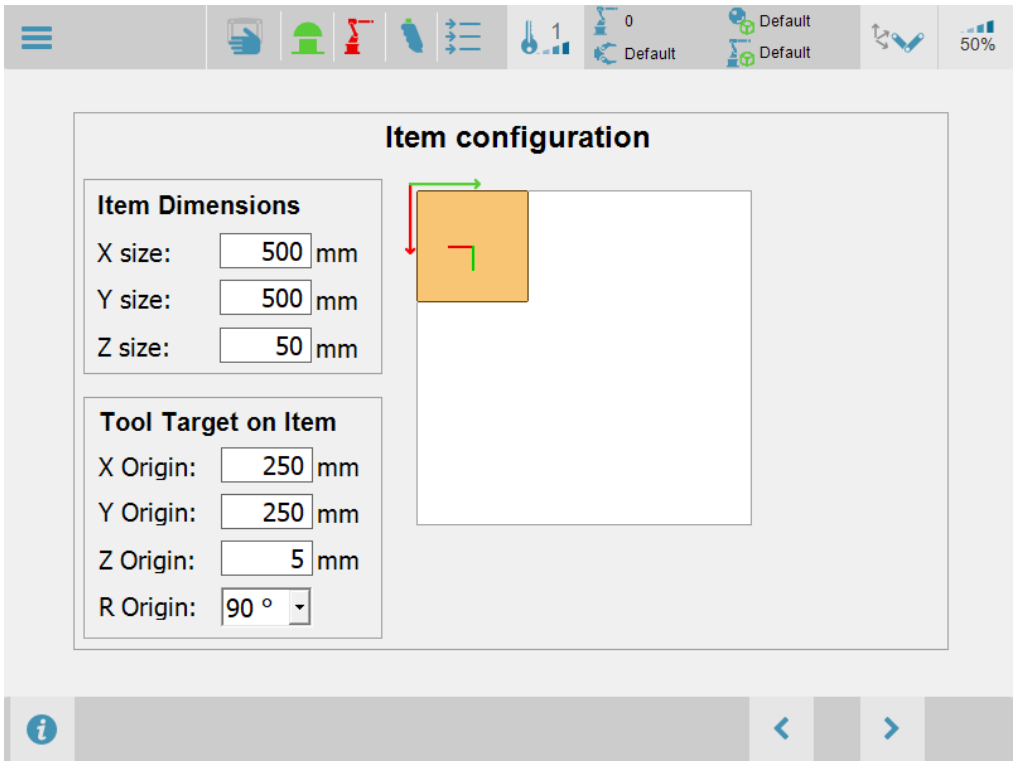



Figure 4-31: item configuration window

The third step is to design number items of each layer.

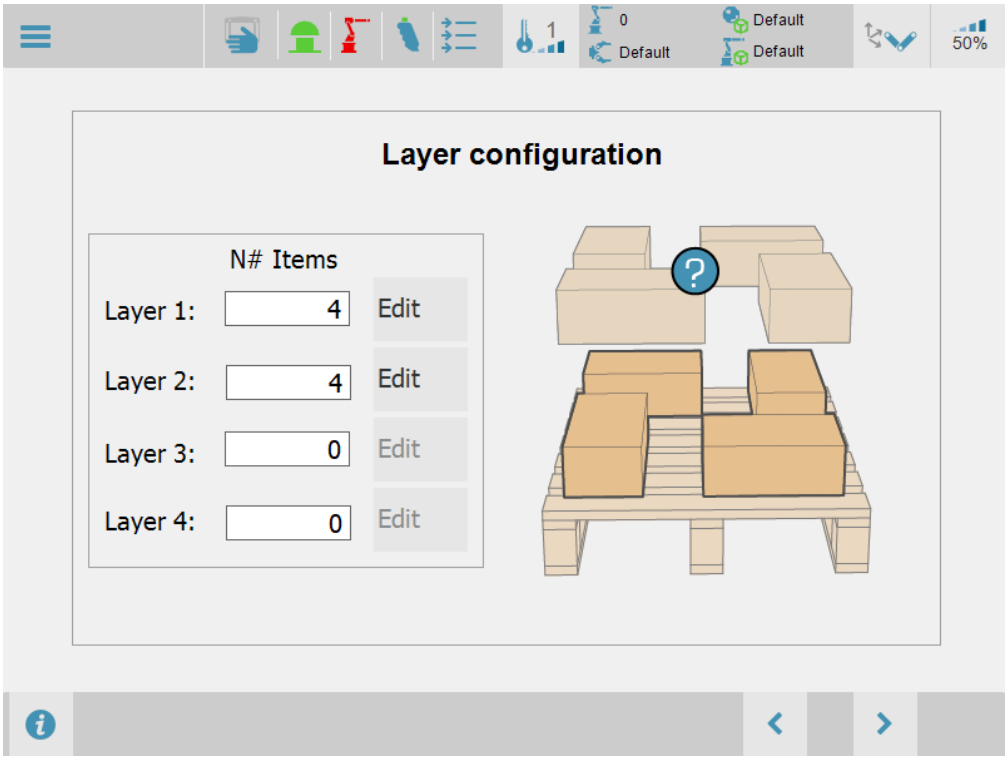



Figure 4-32: layer configuration step1 window

The objects on each layer can be organised by clicking Edit in Layer configuration page.

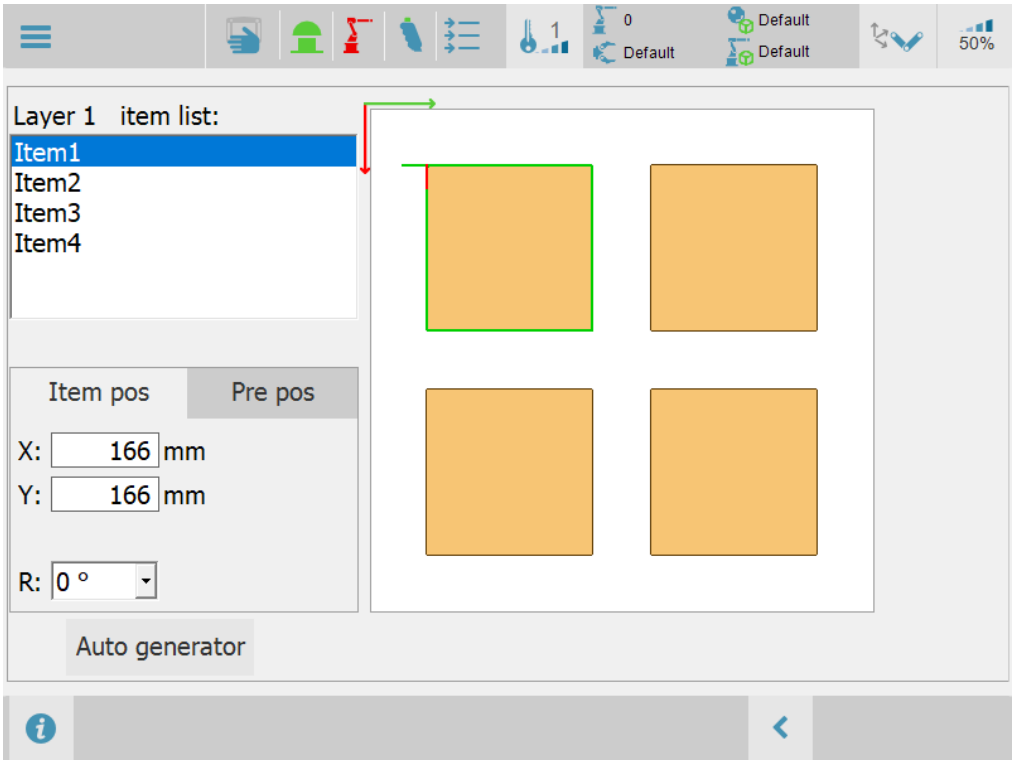



Figure 4-33: item organiser window

The next step can put items on a virtual tray.

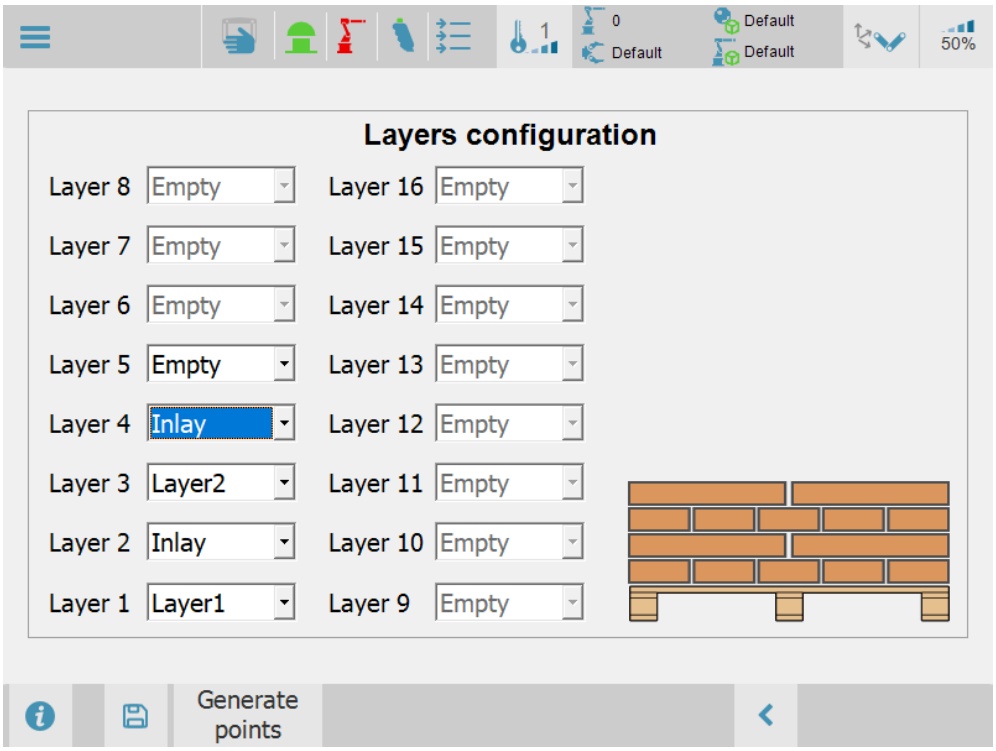
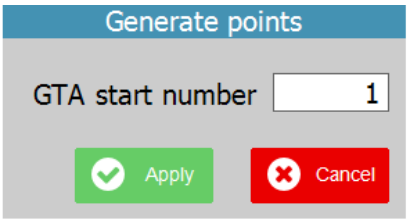



Figure 4-34: layer configuration step2 window

The Generate Points button can define the start number from GTAs list.




There is another application related to this palletizer application in pendant edit program list.

# 5 Projects and programs

## 5.1 Project manager

Projects and programs can be handled through project manager window.

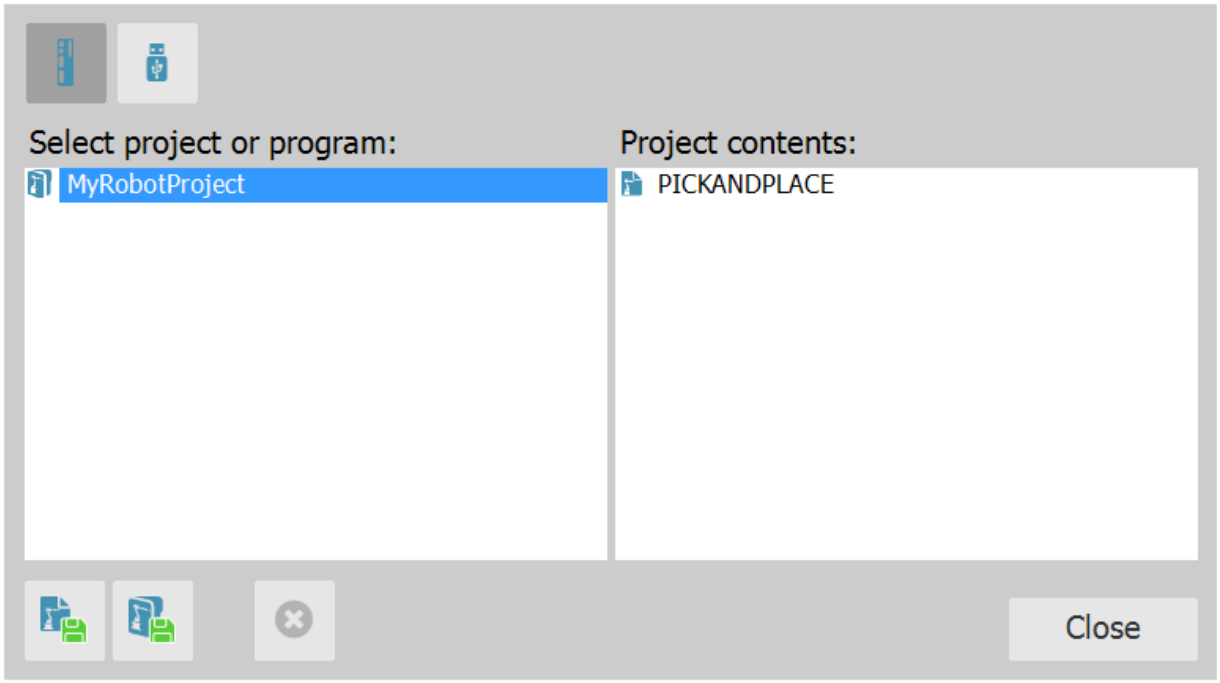



Figure 5-1: project manager window

In this manager it is possible to save programs and projects into an USB stick connected to the teach pendant if the controller button is selected.

If USB stick button is selected, its content is shown, and programs and projects can be loaded or deleted.

Only one project with multiples programs can be stored in controller memory. For multiples projects use an USB stick. Projects must be in root directory organised in folders.

## 5.2 Program editor

In program editor page is where a program can be edited, and it provides debugging facilities for robot programs.

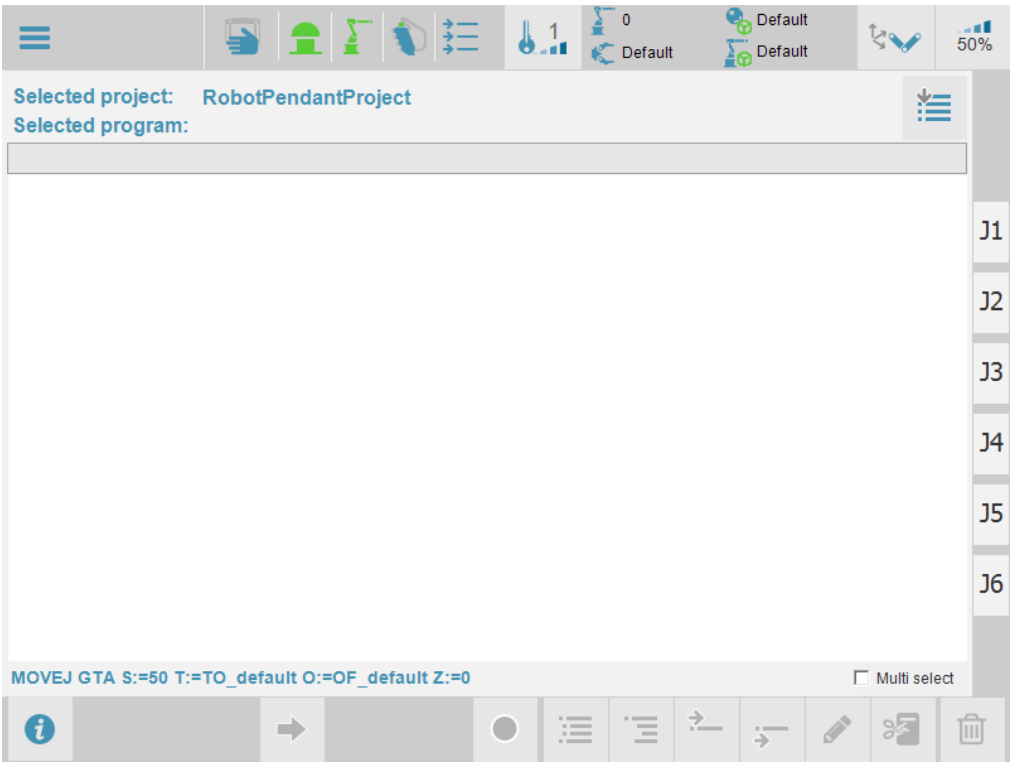



Figure 5-2: Program editor page

Debug buttons:

| | | |
|---|---|---|
|  | Set program pointer | Set the program pointer over the selected line. If the program is stop then it will start it, pausing it over the selected line (previous instructions will not be executed). |
|  | Break point | Toggle breakpoint on selected line. |

*Refer to section Program state manipulation for more information about how to start / stop programs.*

Program edition buttons:

| | | |
|---|---|---|
|  | Program list | Prompt the list of robot programs and the actions that can be done to them. |
|  | Indent | It will auto-format the program for a more readable form. |
|  | Comment | Comment and uncomment selected lines, group of lines or insert comment in line. |
|  | Insert above | Shows the instruction list to be inserted above the selected line. |
|  | Insert below | Shows the instruction list to be inserted below the selected line. |
|  | Edit | It will prompt the corresponding window depending on the selected instruction to edit. |
|  | Copy / Cut / Paste | The selected line or multiple lines can be copied, cut or pasted using this menu button. |
|  | Delete | Delete the selected line or multiple lines. |

## 5.3 Multi-line selection

On the right lower corner there is a **Multi select** check box that allows up to 20 lines be selected at the same time. It is possible to copy, cut, paste, delete and comment the selected group of lines at the same time.

## 5.4 Default move values

Default move values can be accessible by the left lower corner link where, in this case, says **MOVEJ S:=50 T:=TO_default O:=OF_default Z:=0.** These values will be used by teach instruction explained below.

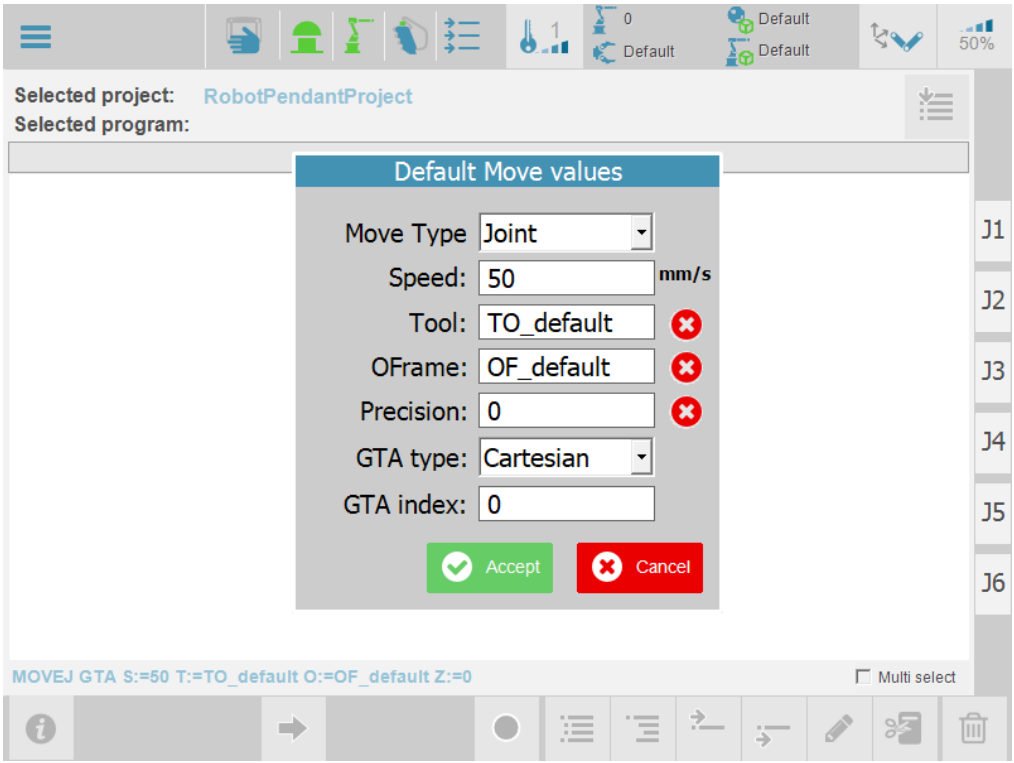



Figure 5-3: default move window

## 5.5 Program types and projects

The *program list* button  will prompt the program list window and what can be done to them. It shows in a list all robot programs available in controller flash memory.

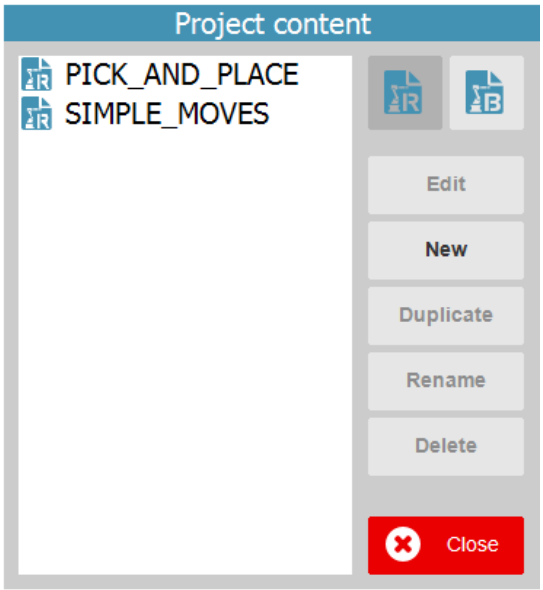



Figure 5-4: program list window

There are two different robot programs: Robot Programs (.ROB file extension) and Robot Basic Programs (.RBS file extension).

Both have the same functionality, but robot programs can only contain the list of functions available from pendant. Also, TPS can only edit robot programs. Robot basic programs can only be run and debugged.

To select what type of robot program will be shown in the program list just simply press the buttons:

Figure 5-4: robot program type

It is possible to edit, duplicate, rename or delete a robot program or robot basic program selecting it on the list and pressing the corresponding button.

To create a new robot program simply press **New** button and assign a unique name.

Robot basic program can only be seen if the system is in the correct level.

The picture below shows a blank new program called “PICK_AND_PLACE”:

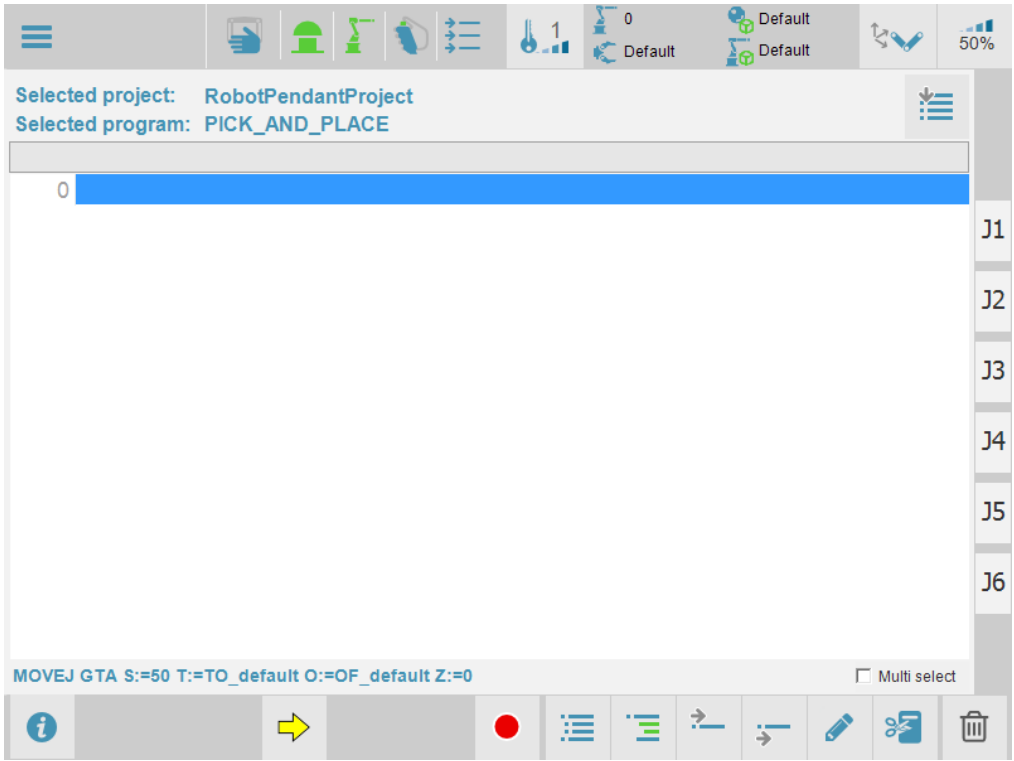



Figure 5-6: program PICK_AND_PLACE recently created

To change the name of the project just simply press over the current project name, “RobotPendantProject” in this case, and a keypad will be prompt.

## 5.6 Instructions set

A certain group of instructions can be inserted above or below of the selected line by clicking *insert above* and *insert below* buttons.

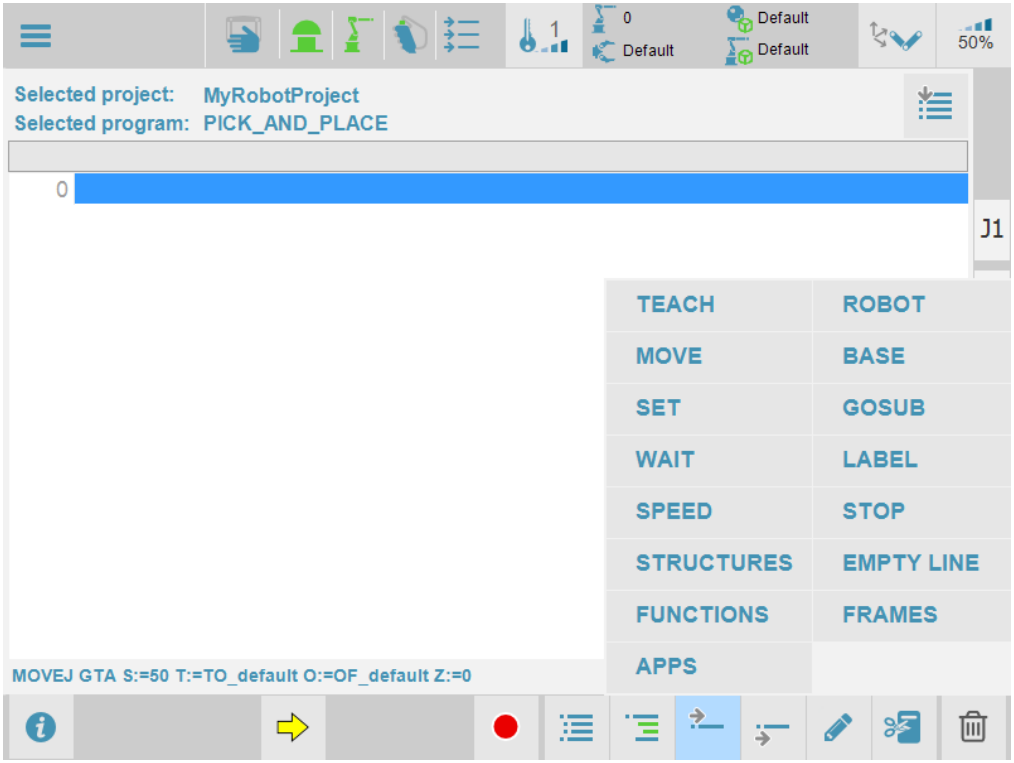



Figure 5-7: Instruction list menu

### 5.6.1 Teach

It is possible to speed up the process of inserting several moves of the same type and the same parameters. In this case the GTA coordinates are captured from the current robot position instead of from the GTA table. The coordinates are saved automatically in the next empty GTA starting from the selected index in default move values window.

### 5.6.2 Move

Up to three move instructions can be inserted with move instruction window.

**MOVEJ** is used to move the robot from one point to another along a non-linear path. All axes reach the destination position at the same time. It is the quickest type of movement due to the axes move the exact amount of degrees needed to reach the desired position.

**MOVEL** is used to move the robot from one point to another along a linear path. All axes reach the destination position at the same time.

**MOVEC** is used to move the robot from one point to another along a circular path. All axes reach the destination position at the same time. This type of move needs a middle point in the curve and the end point.

**MOVEJREL** is used to move the robot from one point to another, relative to the starting position, along a non-linear path. All axes reach the destination position at the same time. It is the quickest type of movement due to the axes move the exact amount of degrees needed to reach the desired position.

The speed of the move is an interpolated speed and NOT the speed of any individual axis or the end effector.

A single joint can be moved by simply set all values of the target as 0 except the desired joint to move.

**MOVELREL** is used to move the robot from one point to another, relative to the starting position, along a linear path. The end effector moves along the straight line between current point and target point in cartesian space. All axes reach the destination position at the same time.

A single vector can be moved by simply set all values of the target as 0 except the desired vector to move.

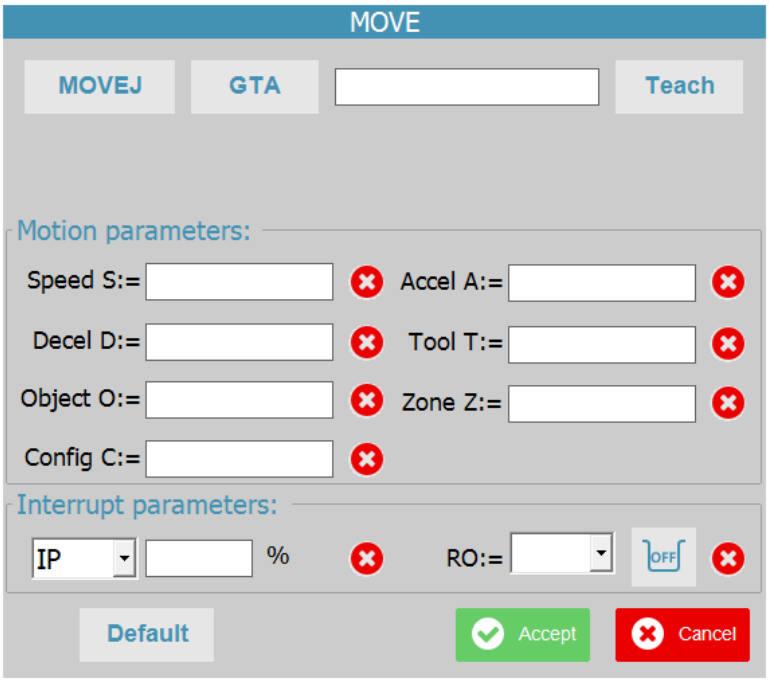



Figure 5-8: MOVE instruction window

It is possible to modify the behaviour of the movement according to some embedded parameters. These parameters only affect the belonging move instruction. If a move instruction has no parameter or just a few it will use the default values for any missing parameter. The available parameters are:

- S: Speed in degrees per second for joint moves and millimetres per second for linear and circular moves.
- A: Acceleration
- D: Decerelation
- T: Tool offset.
- O: Object Frame.
- C: Configuration (just for joint moves).

- Z: Precision (for linear and joint moves).

Use the current robot position storing the data into a GTA is possible through Teach button. If a GTA is selected, the system will reteach it with the new axis values, otherwise the system will store the values in a no active GTA, starting from the selected default index (4.4 Default move values).

After applying the changes, the system will store the data in both, controller volatile memory and in ROBOT_GLOBAL_TARGETS file.

On the other hand, if cancel button is pressed, all the changes will be discarded.

If a GTA is already selected, Teach button will overwrite its value with the current robot position.

### 5.6.3 Robot

The ROBOT command is used to direct all subsequent motion instructions and robot parameter read/writes to a particular robot.

To select external axes BASE command has to be used.

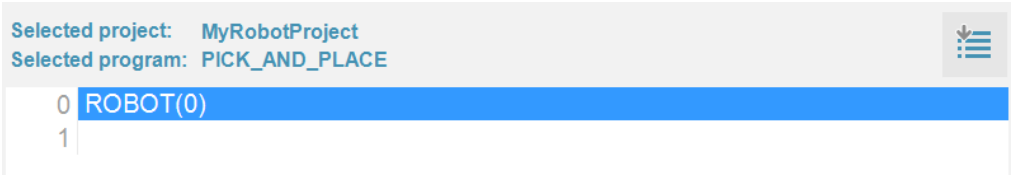



Figure 5-9: ROBOT instruction

### 5.6.4 Gosub

Stores the position of the line after the GOSUB command and then branches to the label specified. Upon reaching the RETURN statement, control is returned to the stored line.

GOSUB structure can be nested up to 8 deeps in each program.

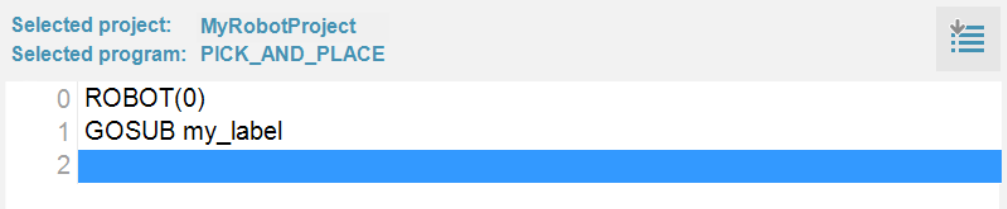



Figure 5-10: GOSUB instruction

### 5.6.5 Label

Labels are used as destinations for GOSUB commands and also to aid readability of code.

With a label RETURN instruction is inserted automatically as well. RETURN instruction can be inserted as its own by selecting RETURN radio button on LABEL window.


Figure 5-11: Label instruction

It is recommended to insert STOP instruction above any LABEL-RETURN structure to avoid execution errors.

### 5.6.6 Stop

STOP instruction will stop the program execution at its current line.


Figure 5-12: Stop instruction

### 5.6.7 Empty line

It will introduce an empty line to aid readability of code.

### 5.6.8 Set

Sets either digital or analog outputs to a given value, assign values to a VR or an already declared variable or declare a variable.

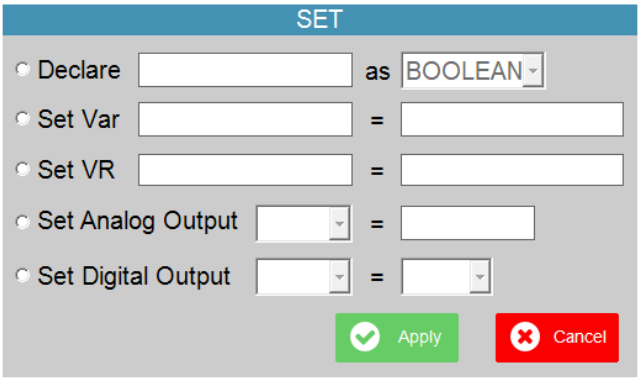



Figure 5-13: Set window

Analog output has to set as 12 bits (+/- 10v)

Only pre-configured Digital Outputs will be shown on the Digital Outputs combo box.

VR is an array of real numbers stored in flash memory. The size of the array depends of controller model.

The type of possible variables that can be declare are the next ones:

- Boolean: 1bit binary value (TRUE or FALSE).
- Float: 64bit floating point number.
- Integer: 64bit signed integer value.
- String: ASCII text (1024 characters maximum).

String data type require size as an extra parameter.

Multiple variables can be declared in one instruction separated by commas.


Figure 5-14: multiple variable declaration

If an invalid symbol is inserted or entry ends with a comma the window will show a message and *Apply* button will be disabled.

Assign values or variables value to a variable is also possible in set window. The variable list will be accessible when *Set Var* is selected or through the button *Var* in set variable value window when setting variable or VR value. If the selected variable is STRING datatype it will display a QWERTY keyboard window at the moment of setting its value.

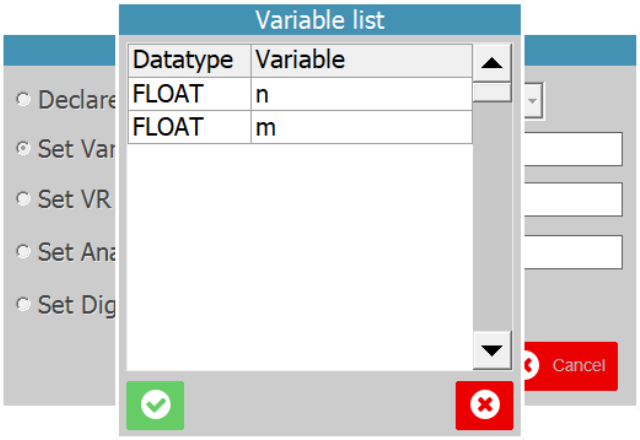



Figure 5-16: Variable list window


Figure 5-15: Set variable value window

The return value of the a robot function can be assigned to a variable of the same return datatype through 'Func' button.

This will prompt a list of avaliable functions for the slected variable datatype.

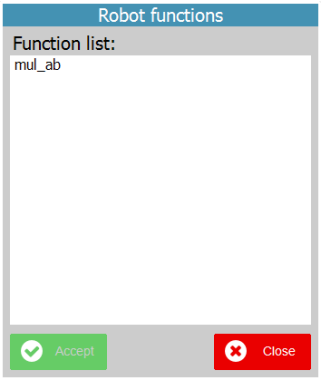



Figure 5-17: Function list window

### 5.6.9 Wait

Three types of wait are possible to set:

- Wait IDLE: wait until all motion in buffer of selected robot or axis is finished.
  It is possible to add time in Wait IDLE, so the robot will hold for the number of milliseconds specified after all motion in buffer is finished.
- Wait: WA() will hold up program execution for the number of milliseconds specified.
- Wait until: wait for selected input is ON or OFF.

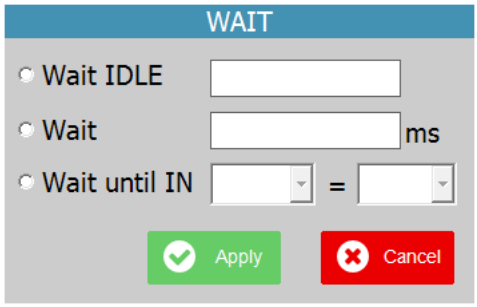



Figure 5-18: Wait window

### 5.6.10 Structures

Structures instructions are compound by WHILE…WEND, REPEAT…UNTIL, IF…ELSE and FOR…NEXT.

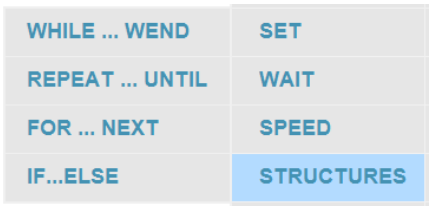



Figure 5-19: Structures submenu

The commands contained in the WHILE…WEND loop are continuously executed until the condition becomes false. Execution then continues after the WEND. If the condition is false when the WHILE is first executed, then the loop will be skipped.


Figure 5-20: While...Wend structure window

The REPEAT…UNTIL structure allows a block of instructions to be continuously repeated until an expression becomes TRUE. REPEAT…UNTIL loops can be nested without limit.

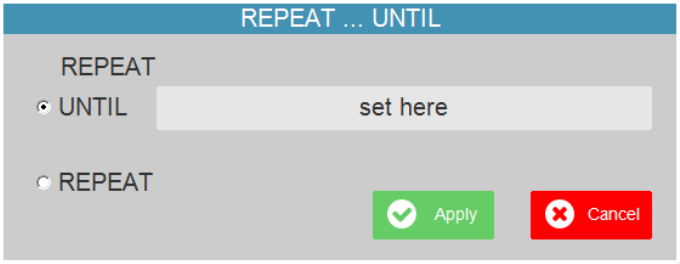



Figure 5-21: Repeat...Until structure window

An IF program structure is used to execute a block of code after a valid expression. The structure will execute only one block of instructions depending on the conditions. If multiple expressions are valid then the first will have its instructions executed. If no expressions are valid and an ELSE is present the instructions under the ELSE will be executed.


Figure 5-22: If...Elseif...Else structure window

WHILE…WEND, REPEAT…UNTIL, IF…ELSE structure instructions can be set by bool condition builder window. It is a wizard that helps users set the condition for structure instructions. The conditions are built in the following format:

variable - relational operator – variable

logical operator

variable - relational operator - variable

…


Figure 5-23: logical operators

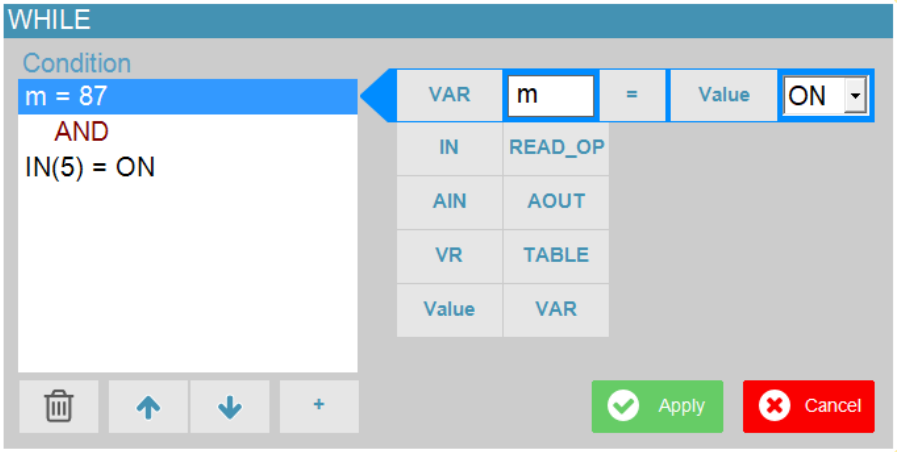



Figure 5-25: Condition builder window


Figure 5-24: relational operators

A FOR…NEXT structure is used to execute a block of code a number of times.

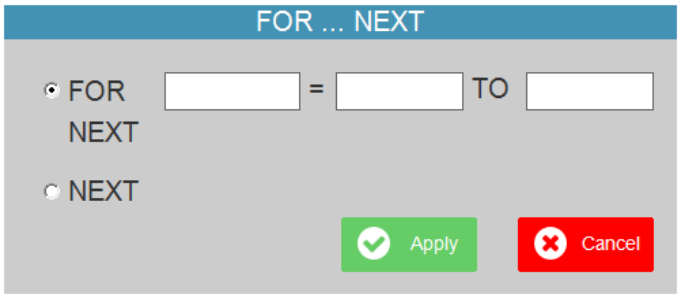



Figure 5-27: For…Next structure window

On entering this structure, the variable (previously declared) is initialised to the value of start and the block of instructions is then executed. Upon reaching the NEXT command, the variable defined is incremented. If the value of the variable is less than or equal to the end parameter, then the block of instructions is repeatedly executed. Once the variable is greater than the end value the program drops out of the FOR...NEXT.

FOR...NEXT loops can be nested up to 8 deeps in each program.

### 5.6.11 Robot Functions

Already defined Robot functions can be called in robot programs to do specific tasks or to have a cleaner code.


Figure 5-28: Function list window

This will prompt a list of avaliable functions for the slected variable datatype.

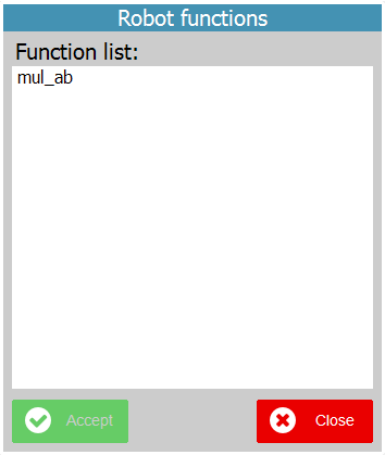



Figure 5-29: Function menu

Assign the return value of a function to a variable is possible through 'Set' window (refer to section 5.6.8).

### 5.6.12 APPS

This application can design a conveyer system compare with former palletizer system. There are several blocks to determine some parameters needed in the conveyer and palletizer system.

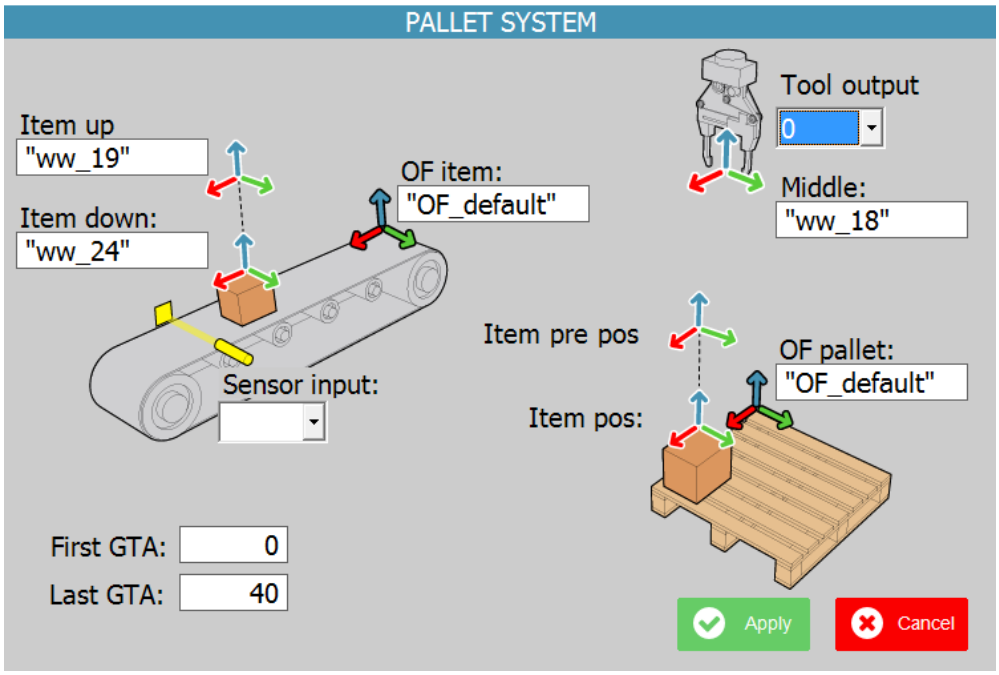



Figure 5-30: pallet system

Description of each parameters:

- Item_up: The position pick item up
- Item_down: The position pick item down
- First GTA: First GTA in the system
- Last GTA: Last GTA in the system
- Tool Output: The output tool like gripper
- OF item: Offset point on the conveyer
- OF pallet: Offset point on the pallet
- Sensor input: Vision or light sensor input of system
- Middle: The middle point of tool.

# 6 Robot functions

Users can define robot functions and call them in robot programs, giving a lot of programming possibilities. Robot has to be defined in Robot Function Libraries. These ones can be Robot Functions or Robot Basic Functions depending on user level.

Robot functions can be defined in functions editor which is accessible through Main menu -> Projects -> Function editor.


Figure 6-1: Function Editor access

The function editor aspect is very similar to program editor.

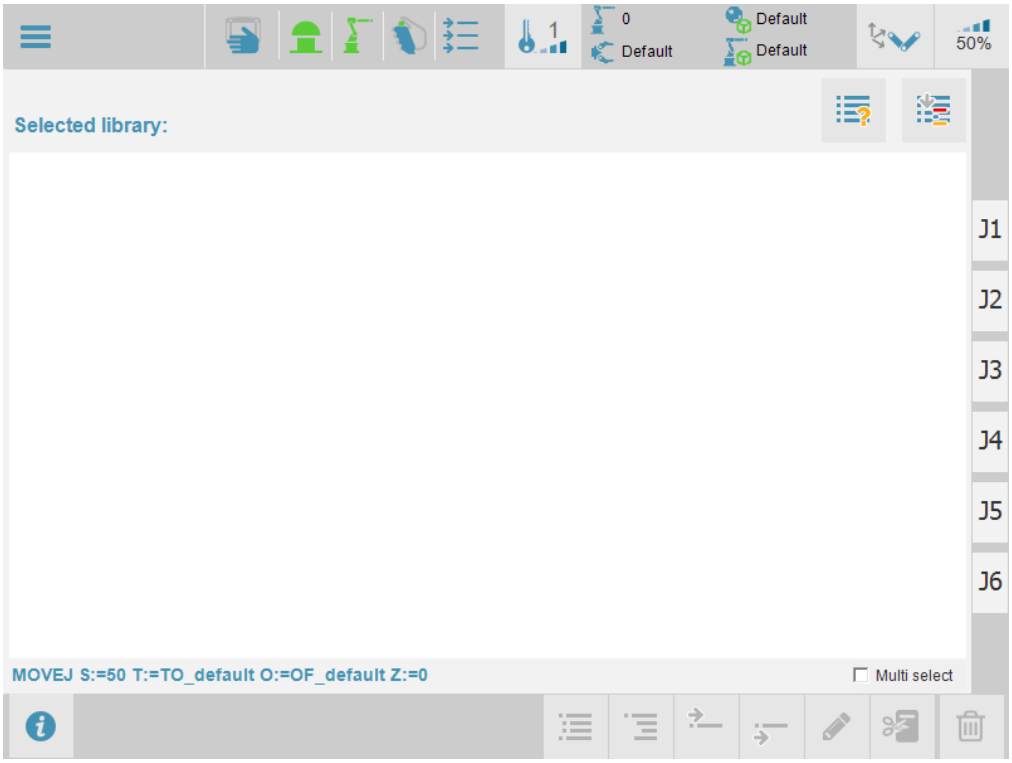



Figure 6-2: Function Editor

Program edition buttons:

| | | |
|---|---|---|
|  | Compile | Forces compilation of the selected Robot Function Library. |
|  | Function library list | Prompt the list of Robot Function Libraries and the actions that can be done to them. |
|  | Indent | It will auto-format the program for a more readable form. |
|  | Comment | Comment and uncomment selected lines, group of lines or insert comment in line. |
|  | Insert above | Shows the instruction list to be inserted above the selected line. |
|  | Insert below | Shows the instruction list to be inserted below the selected line. |
|  | Edit | It will prompt the corresponding window depending on the selected instruction to edit. |
|  | Copy / Cut / Paste | The selected line or multiple lines can be copied, cut or pasted using this menu button. |
|  | Delete | Delete the selected line or multiple lines. |

## 6.1 Multi-line selection

On the right lower corner there is a **Multi select** check box that allows up to 20 lines be selected at the same time. It is possible to copy, cut, paste, delete and comment the selected group of lines at the same time.

## 6.2 Default move values

Default move values can be accessible by the left lower corner link where, in this case, says **MOVEJ S:=50 T:=TO_default O:=OF_default Z:=0.** These values will be used by teach instruction explained below.

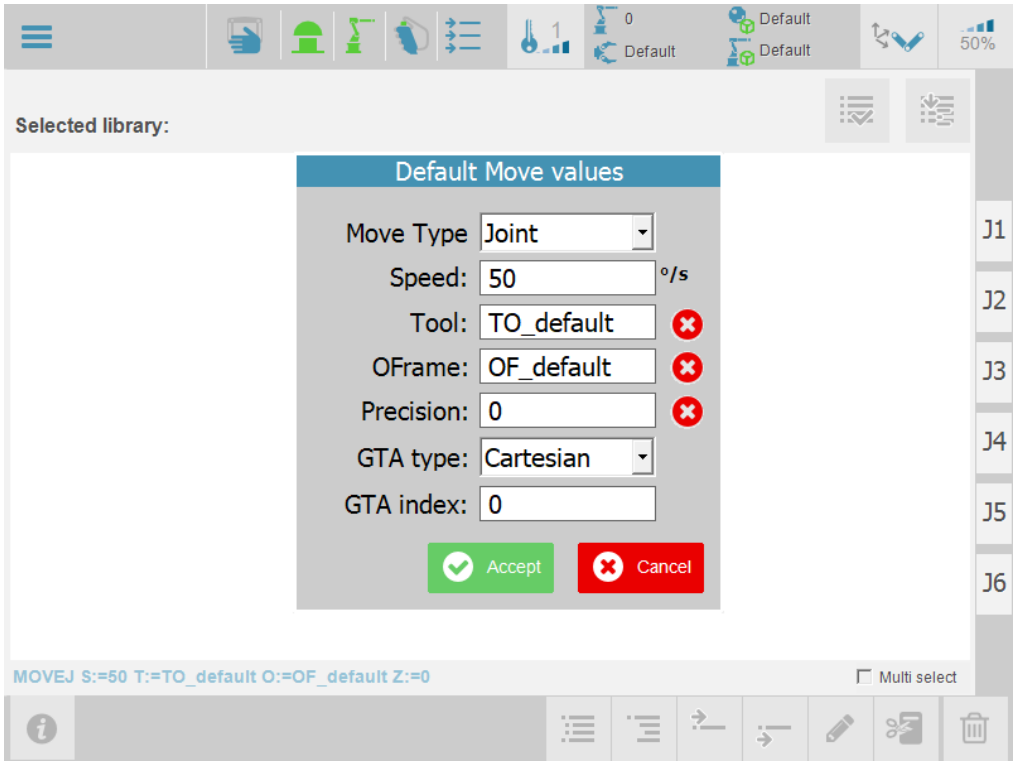



Figure 6-3: default move window

## 6.3 Library and function types

The function library list button will prompt the library and function list window and what can be done to them. It shows two list, all robot function libraries and function list associated to the selected library, all of them available in controller flash memory.

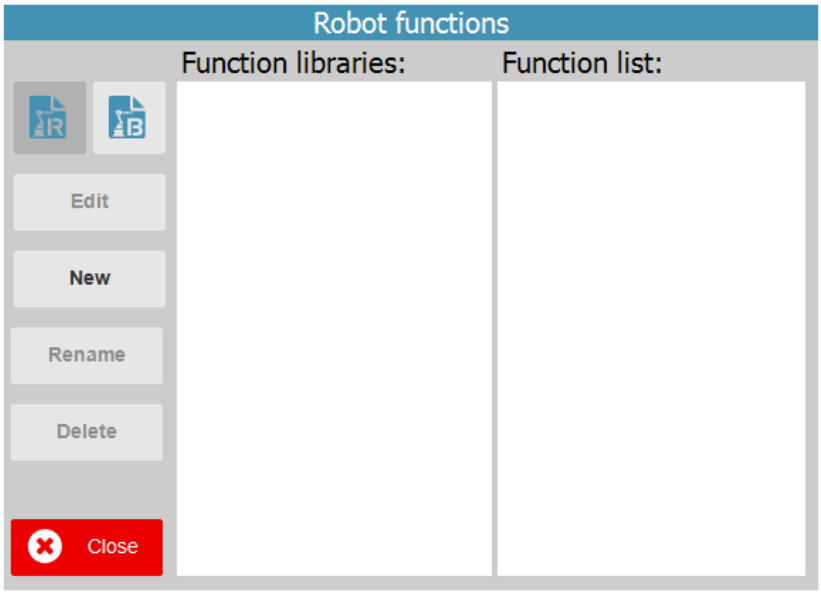



Figure 6-4: program list window

There are two different robot libraries: Robot Libraries (.ROBLIB extension) and Robot Basic Libraries (.RBSLIB file extension).

Both have the same functionality, but robot libraries can only contain the list of functions available from pendant. Also, TPS can only edit robot libraries. Robot basic libraries functions and their functions can only be called.

To select what type of robot library will be shown in the library list just simply press the buttons:

Figure 6-5: robot library type

It is possible to edit, duplicate, rename or delete a robot library selecting it on the list and pressing the corresponding button.

To create a new robot library simply press New button and assign a unique name.

Robot basic library can only be seen if the system is in the correct level.

The picture below shows a blank new library called “ROB_LIB”:

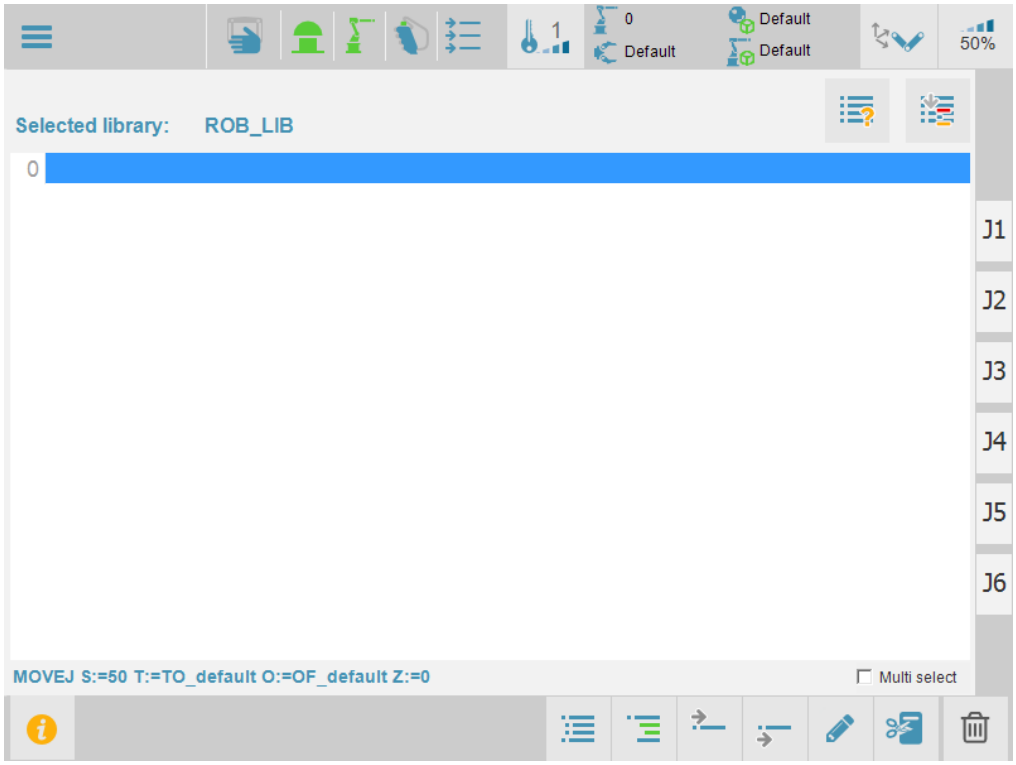



Figure 6-7: library ROB_LIB recently created

## 6.4 Functions: create and insert

Function editor is very similar to Program editor with the difference that it is possible to create and insert functions (Program editor can only insert functions).

To create a function, click over 'insert above' or 'insert below' button , then 'Functions' and then 'Create'.

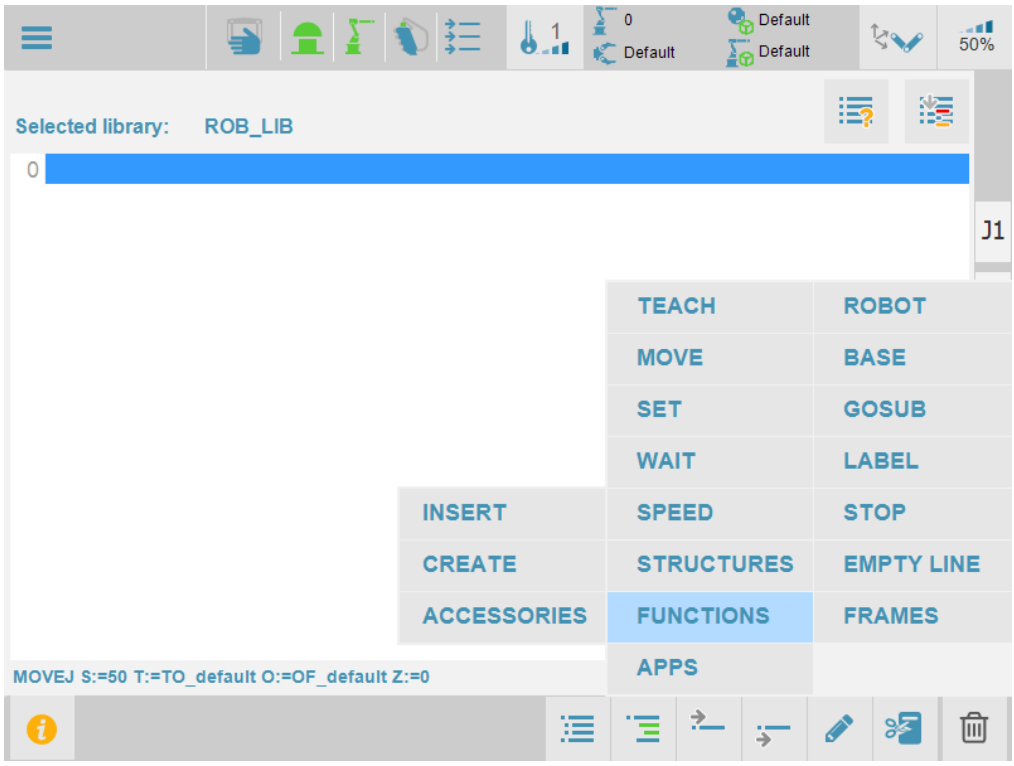



Figure 6-8: functions menu

The Robot functions creator window will prompt.

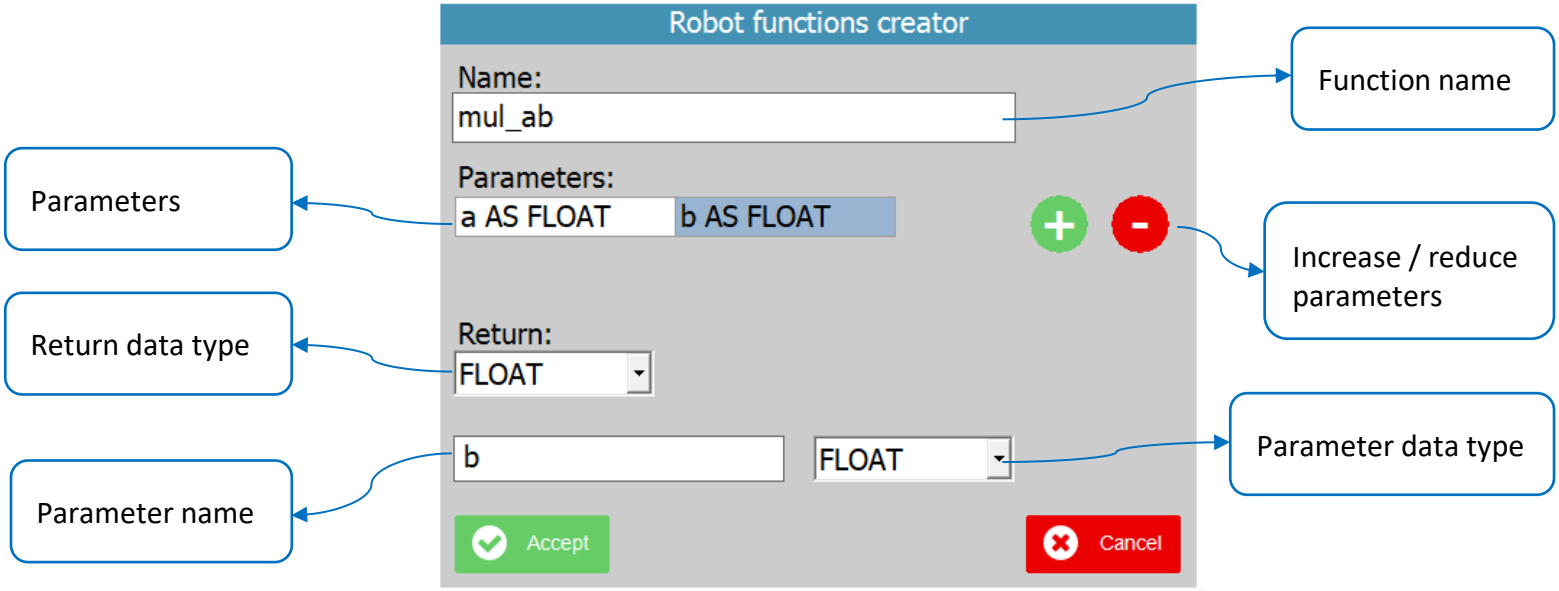



Figure 6-9: Robot functions creator

'Function name' is the only compulsory entry, 'Parameters' and 'Return' are optional.

To add parameters just simply click over increase / reduce parameters buttons  .

Then, select the parameter, select the parameter data type through the combo box and click over parameter name. A window will prompt to insert a name for it.

After click over 'Accept' button, the example function 'mul_ab' will be created as the following image shows:

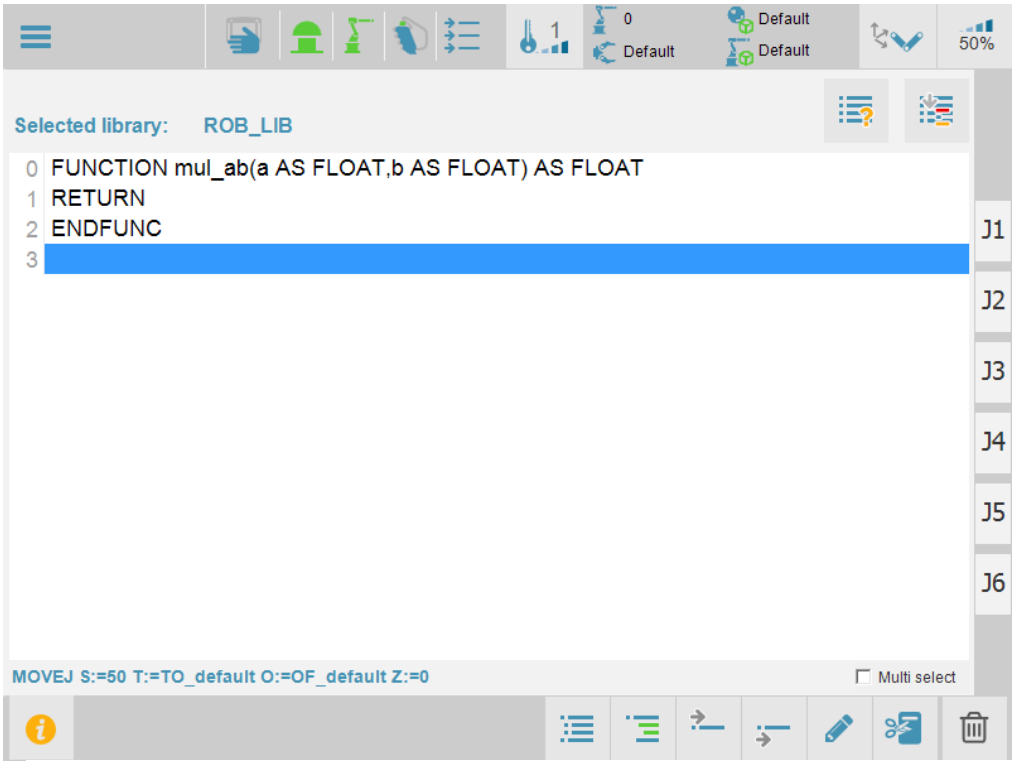



Figure 6-10: mul_ab function recently created

***Pay attention to 'Compile' button. In this case the library has not been compiled. Meaning that a program will not be editable or run.***

It is possible to declare variables and return their values in the function.

To do that, first insert a variable declaration inside the Function structure, accessible through 'SET' button in insert above / below menu  :

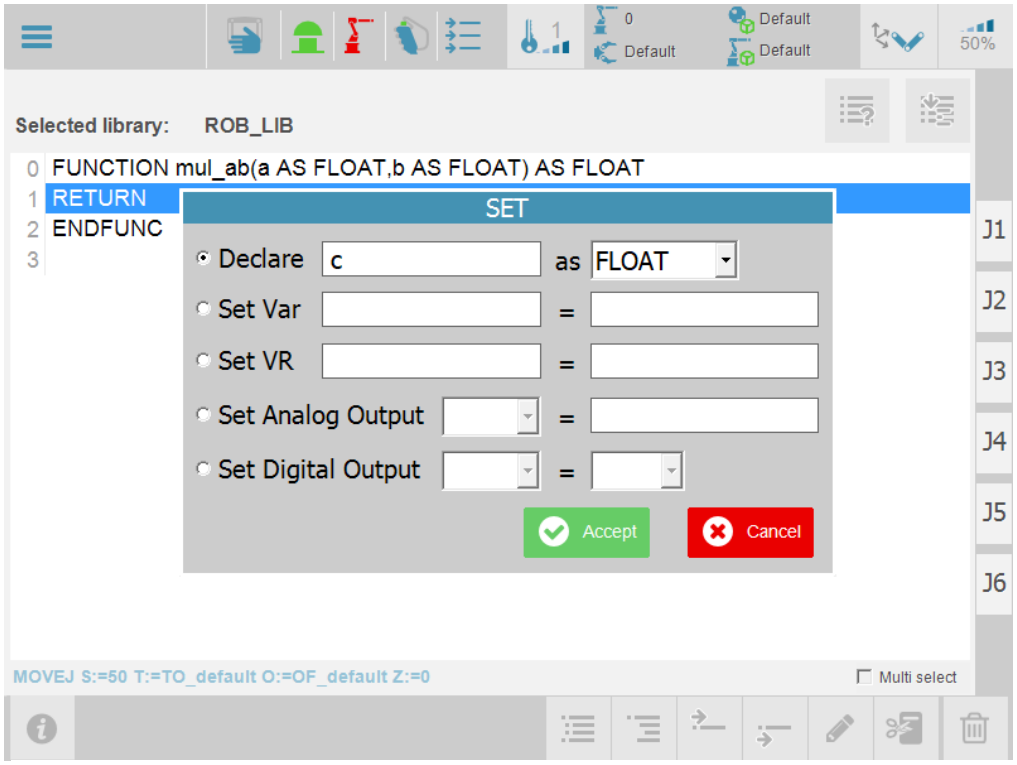



Figure 6-11: insert variable 'c' declaration

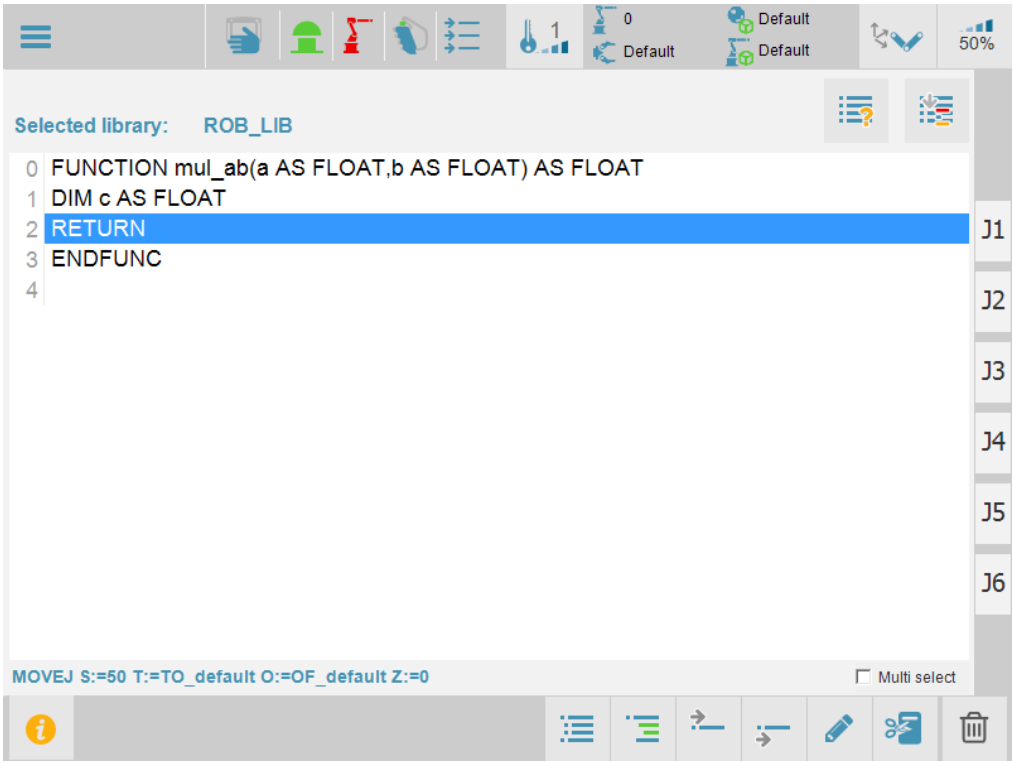



Figure 6-12: 'c' variable declared

Now we can set the new variable 'c' as the return value for our function by simply edit the 'RETURN' instruction using 'edit' button after select the line:

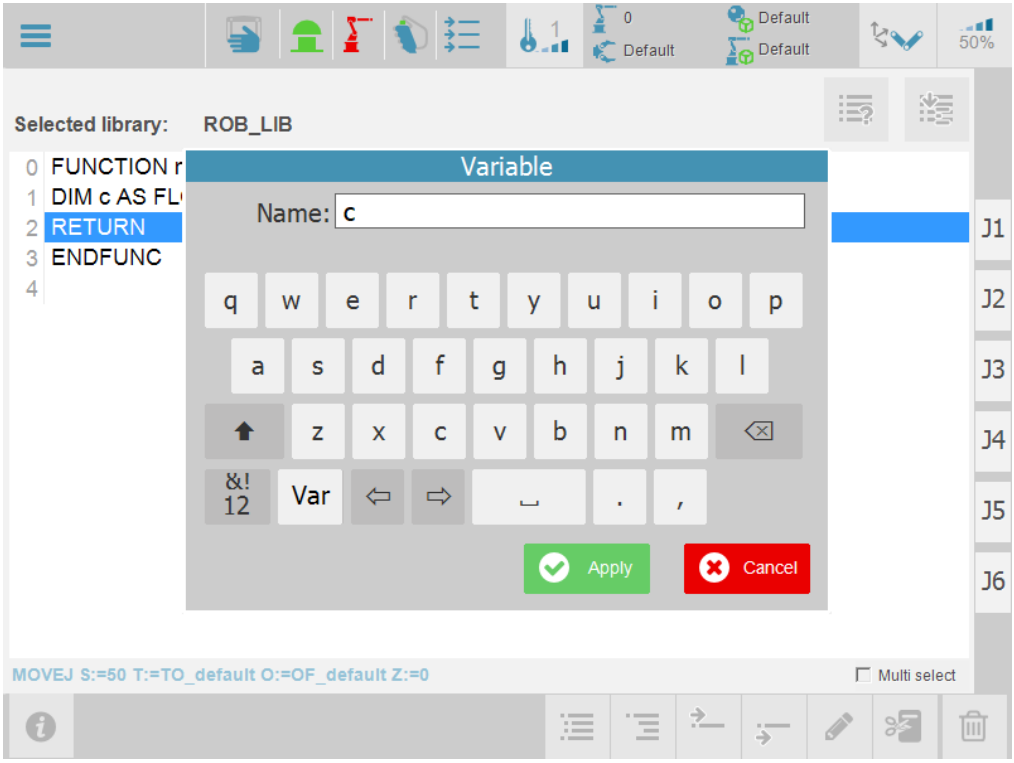



Figure 6-13: return 'c' variable

After this, the library can be compiled by pressing ‘compile’ button  . It will not return any value due to ‘c’ variable has not being used but the function syntax is correct. ‘Compile’ button will change if the library has been compiled successfully. 

A simple operation can be done to finalise our ‘mul_ab’ function. Let’s multiply parameters ‘a’ and ‘b’ and assign it to ‘c’ variable. This can be done using ‘SET’ window as before.

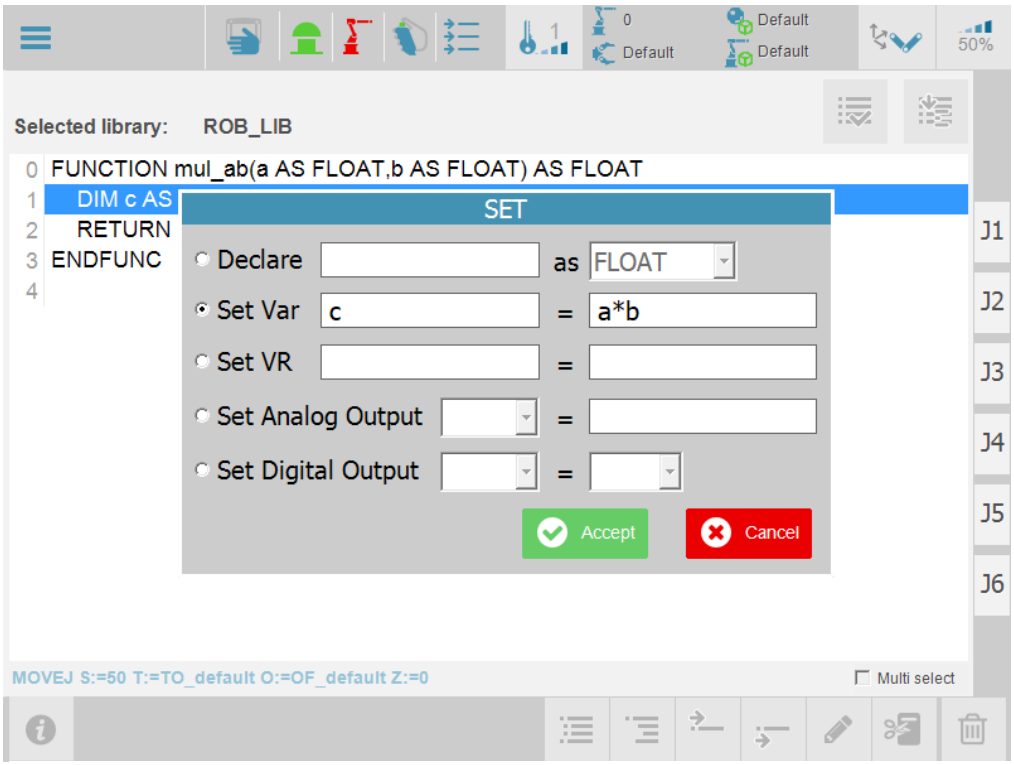



Figure 6-14: set ‘c’ variable as ‘a’ * ‘b’

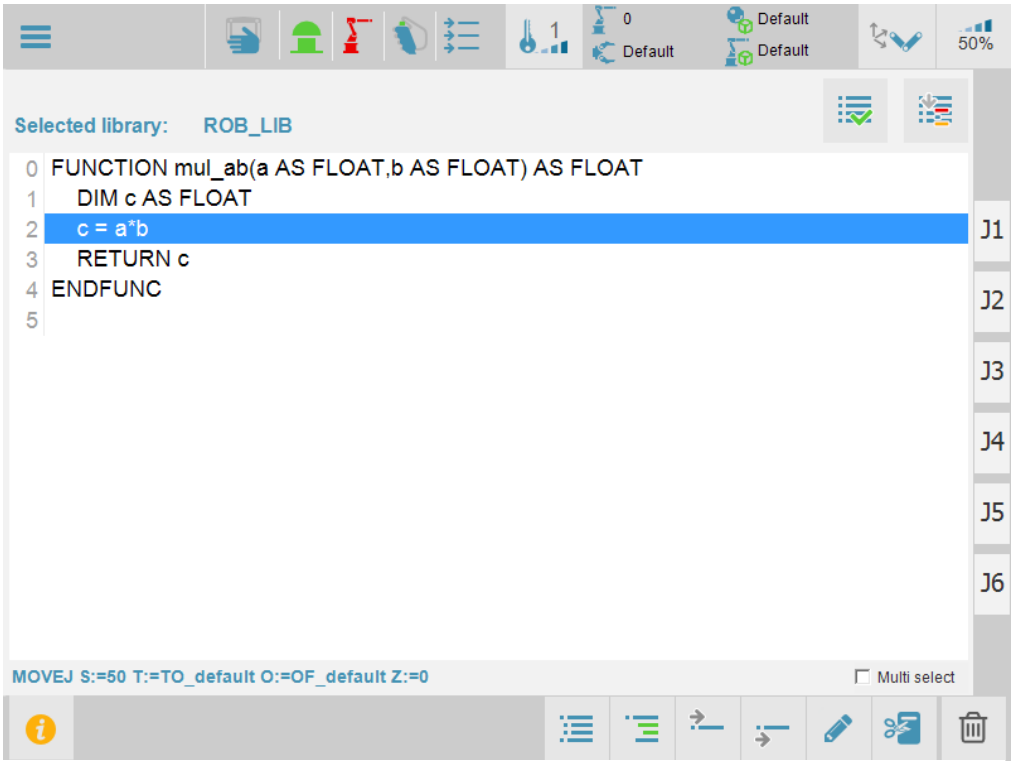



Figure 6-15: 'c' = 'a' * 'b'

After indent our library and compile it, our 'mul_ab' is ready to be called from programs.

# 7 Program state manipulation

The program state can be changed by the start 

and stop 

keys.

Once a program is selected (refer to section Projects and programs to know how to select or create a program) it could be started in different modes depending on how the start button is operated, the selected execution mode and the selected TPS mode.

TPS can be in Manual mode or Auto mode.

- If the system is in Manual mode  then the demand switch has to be enabled, otherwise the system will prompt an error if intended to run a program. Refer to Demand switch section to know how to enable demand switch.
- If the system is in Auto  mode, MOT button 

should be pressed to enable the system.

Once the system is enabled, a drive enable status icon  will be depicted in green. Otherwise it will be red.

The following table describes the program execution behaviour:

| TPS mode | Program execution behaviour |
|---|---|
| Manual | **Step mode**: program will step one line when the button is pressed and, if the instruction has not been completed, stopped when released. If the instruction has finished before release start button then the program will remain paused until stop button is pressed.<br>**Continuous mode**: program will start running when the button pressed and hold, and will pause when released. The program will stop when stop button is pressed. |
| Auto | **Step mode**: program will step one line when start button is pressed, no matter when the button is released.<br>**Continuous mode**: program will start running when start button pressed and it will stop when stop button is pressed. |

To change the operation mode, step button 

has to be pressed to toggle between step mode  and continuous mode  .

*This section is only applicable to TPS. The program execution behaviour will be different if it is performed through Motion Perfect. Refer to RPS documentation for more detailed information.*

# 8 Settings

## 8.1 About

In this section it is shown information about the version of the system: controller version, serial number, controller type, UniPlay version and RPS version.

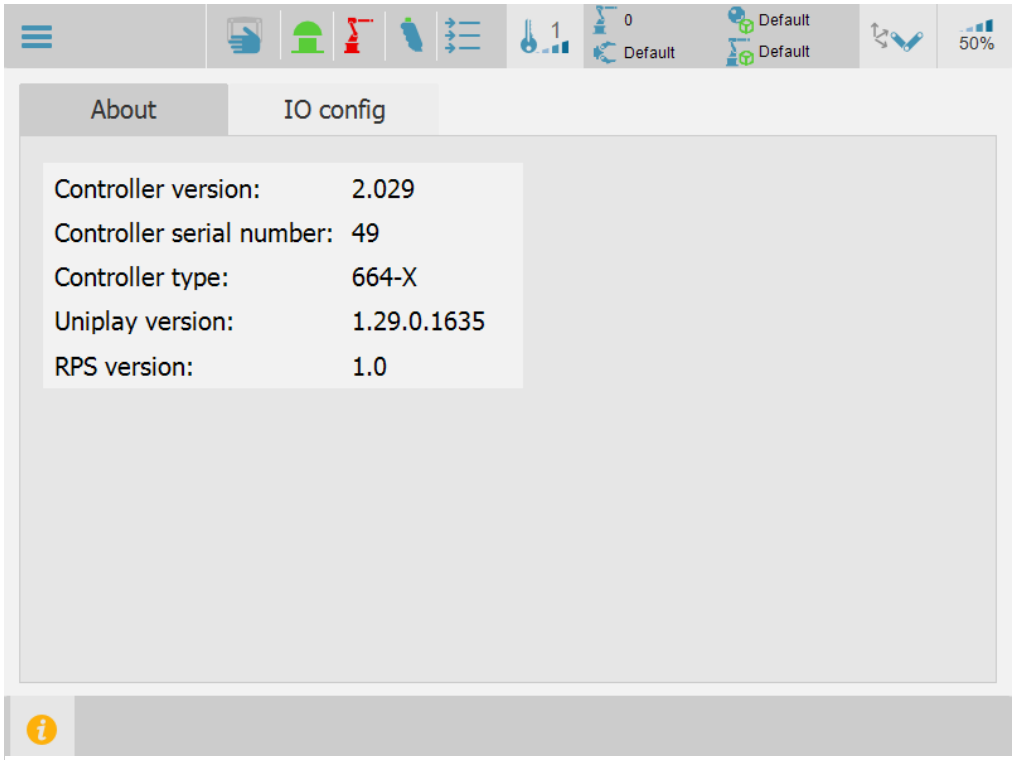



Figure 8-1: About page

## 8.2 IO configuration

Every system has different IO configuration. In this page it will be possible to address the different physical IOs with RPS.

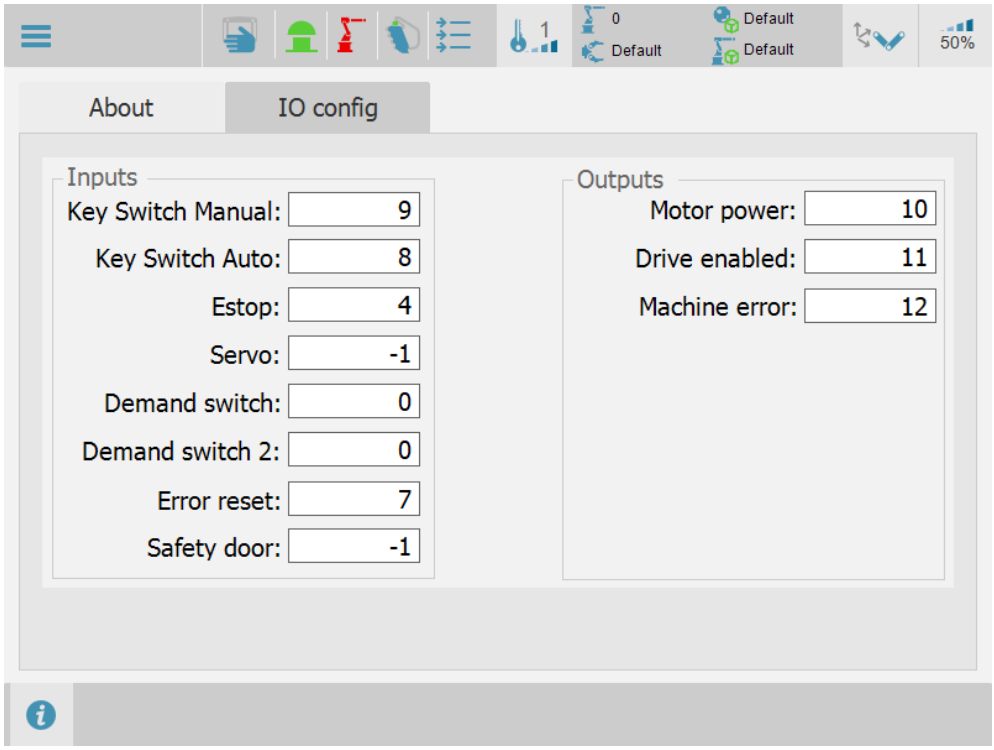



Figure 8-2: IO configuration

The inputs should be set as per physical wiring of the components.
It could be the possibility Servo, Error reset and Safety door do not exist in the real system. For those ones, a -1 value has to be set. This will tell to TPS no real input has been set for that particular component.

*In the case of Error reset, if there is not real input and -1 is set, Refresh button has to be pressed to reset the errors in case they occur (refer to section 4.4 Warning Error window).*

# 9 Error and warning codes

## 9.1 Axis status codes

| | |
|---|---|
| AS_2 | Communications error to remote drive |
| AS_3 | Remote drive error |
| AS_4 | In forward hardware limit |
| AS_5 | In reverse hardware limit |
| AS_8 | Following error exceeds limit |
| AS_9 | FS_LIMIT active |
| AS_10 | RS_LIMIT active |
| AS_12 | Pulse output axis over-speed |
| AS_16 | AXIS_FS_LIMIT active |
| AS_17 | AXIS_RS_LIMIT active |
| AS_21 | FEC 26: Robotics runtime 1-hour free limit. Reset the controller |

## 9.2 Robot status codes

| | |
|---|---|
| RS_0 | WORLD_FS_LIMIT active |
| RS_1 | WORLD_RS_LIMIT active |
| RS_2 | ROBOT_FS_LIMIT active |
| RS_3 | ROBOT_RS_LIMIT active |
| RS_4 | TCP_FS_LIMIT active |

| | |
|---|---|
| RS_5 | TCP_RS_LIMIT active |
| RS_7 | Robot following error exceeds limit |
| RS_8 | Wrist singularity |
| RS_9 | Alignment singularity |
| RS_10 | Elbow singularity |
| RS_11 | Max speed limit |
| RS_12 | Robot collided |

## 9.3 TPS system codes

| | |
|---|---|
| TE_0 | Jog attempted out of Manual mode. Select Manual mode |
| TE_1 | Jog attempted while E-Stop is pressed. Release E-Stop and enable robot |
| TE_2 | Jog attempted without enable the robot. Press demand button |
| TE_3 | Jog attempted with servo disabled |
| TE_4 | Jog attempted while a program is running |
| TE_5 | Jog attempted while move buffers are not empty |
| TE_6 | Jog attempted in linear while in singularity. Jog in joint mode to go out of singularity |
| TE_7 | Attempting to run a program while E-Stop is pressed. Release E-Stop and enable robot |
| TE_8 | Attempting to run a program without enable the robot. Press demand button |
| TE_9 | Attempting to run a program without enable the robot. Press MOT button |
| TE_10 | Failed to enable the robot |

| | |
|---|---|
| TE_11 | Jog speed 0% |

## 9.4 RPS Architecture codes

| | |
|---|---|
| RA_0 | Error status because E-Stop. Release E-Stop and press reset button |
| RA_1 | Error status because MOTION_ERROR. Press reset button to clear the error |
| RA_2 | Error status because SYSTEM_ERROR. Press reset button to clear the error |
| RA_3 | Error status because MOTOR output is off. Press reset button to clear the error |
| RA_4 | Error status because WDOG turned off unexpectedly. Press reset button to clear the error |