MECADEMIC

INDUSTRIAL ROBOTICS

MC-OM-MCS500

Revision number: 11.1.43

Mecademic Robotics

July 17, 2025

Contents

MecaPortal Operating Manual for the MCS500 Industrial Robot

For firmware version: 11.1

Document revision: A

Online release date: July 17, 2025

Document ID: MC-OM-MCS500

About this manual

This user manual describes how to use Mecademic's MecaPortal, in the case of the Mcs500 robot.

Symbol definitions

The following table lists the symbols that may be used in Mecademic documents to denote certain conditions. Particular attention must be paid to the warning and danger messages in this manual.

> ℹ️ **Note**
>
> Identifies information that requires special consideration.

> ⚠️ **Warning**
>
> Provides indications that must be respected in order to avoid equipment or work (data) on the system being damaged or lost.

> ☢️ **Danger**
>
> Provides indications that must be respected in order to avoid a potentially hazardous situation, which could result in injury.

Revision history

The firmware that is installed on Mecademic products has the following numbering convention:

{major}.{minor}.{patch}.{build}

Each Mecademic manual is written for a specific {major}.{minor}.{*}.{*} firmware version. On a regular basis, we revise each manual, adding further information and improving certain explanations. We only provide the latest revision for each {major}.{minor}.{*}.{*} firmware version. Below is a summary of the changes made in each revision.

Revision	Date	Comments
A	March 17, 2025	This version

The document ID for each Mecademic manual in a particular language is the same, regardless of the firmware version and the revision number.

Introduction

MCS500's web interface, called MecaPortal, is more or less the equivalent of the teach pendant's interface of a traditional industrial robot. The interface is essentially an HTML 5 web page with JavaScript and WebGL scripts. All of these files reside in the robot's controller, so you do not need to install anything on your computer, but Google Chrome or another web browser.

The interface basically translates your mouse clicks, joystick movements and keyboard entries into proprietary commands that are sent to the robot's controller. These are primarily the same commands described in the Programming Manual that you will eventually start sending from your own application, written in C++, Java, Python or any other modern programming language. In addition, the web interface displays the feedback messages received from the robot and the 3D model of the actual robot.

The MecaPortal is intended mainly for testing and writing simple programs. You must create your own software application or program if you intend to use the robot for complex tasks, such as interacting with inputs and outputs (in which case you also need a third-party I/O module).

The web interface is also used for updating the firmware of your robot.

> ☢ **Danger**
>
> Before using the MecaPortal, you must read your robot's user manual (MC-UM-MCS500)

Privacy

Although the MecaPortal runs in a web browser, it does not require an internet connection to operate your robot. By default, it remains offline. The only times the MecaPortal connects to the internet are when you click on the Mecademic logo in the top-right corner or when you use clearly marked links to our website (for example, to view tutorials or documentation). If you do not use these links, no internet connection is made.

Overview

The following figure shows the main elements of the MecaPortal web interface. These are:

- The menu bar
- The code editor panel
- The jogging panel
- The 3D view panel
- The log panel
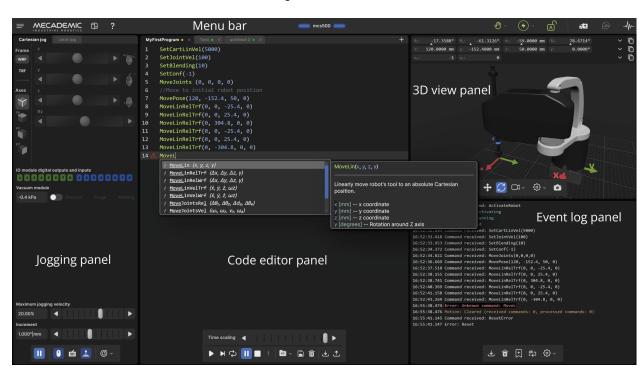- You can show/hide each of the four panels from the button ⊞, in the menu bar.

Figure 1: Overview of the MecaPortal

The menu bar

From left to right, the menu bar features

- The configuration menu button, ≡
- The panels show/hide menu button, ⊞
- The show/hide tooltips button, ?
- The robot name surrounded by state bars (see Table 1)
- The connection state selector, ⚬⚬ (disconnect) or ◎ (monitoring) or ✋ (control)
- The robot state selector, ⏻ (deactivate) or ⚡ (activate) or ⌂ (home)
- The recovery mode button, 🔧
- The simulation mode button, 🎭
- The end-of-arm tooling state icon, 🔧

> ℹ **Note**
>
> Until you get to know the MecaPortal, we recommend that you activate the tooltips using the button ? , in the menu bar. Most tooltips are disabled by default.

Table 1: The robot state bars in the MecaPortal

Visual state	Explanation
Gray, solid	The robot is deactivated.
Yellow, solid	The robot is activated but not homed, or is in recovery mode but not moving.
Yellow, blinking	The robot is in recovery mode and executing motion commands.
Green, solid	The robot is homed.
Green, blinking	The robot is homed and executing motion commands.
Blue, solid	The robot motion is paused.
Red, blinking	The robot is in error mode.

The connection and robot state selectors

Normally, once the web user interface is loaded, you have not only established an HTTP connection with the robot, but also activated the socket messaging which is the only communication channel between the web interface and the robot. By default, the web interface connects to the robot in monitoring mode only (as seen in `fig:MecaPortal-ConnectionButton`). To control the robot click on the ◎ icon and select ✋. If the robot is in error when you connect to the robot, the connection state button will show a red blinking ⚠ icon. You can still press that button and select to only monitor or control the robot. The monitoring option can be used to see in real-time the motion of the robot when

another client (e.g., another MecaPortal, a Python program, or a Profinet application running on a PLC) is controlling the actual robot.

Updating the robot's firmware

Make sure that you have the latest firmware installed on the robot and that you read the manuals corresponding to that firmware. Go here, and download the latest firmware (a zip file). Unzip it. Then, once you have followed the procedures of *installing-the-robot-system*, *first-time-use*, and *power-up-procedure* of your robot user manual, proceed to updating the robot firmware as follows:

- First, take control of the robot using the connection menu button 🖐.

- Then, open the configuration menu ≡ and navigate to the "Update firmware" section.

- Browse and select the file mcs500*.update that you just extracted.

- Finally, click on the Upload button.

A pop-up window will appear, showing the progress of each step in the process. Wait a few minutes for the update to complete. Once finished, the robot will reboot, and the new web interface will reload.

Now that you have installed the latest robot firmware, make sure to read the documentation that corresponds to that firmware. You are currently reading the documentation for firmware 11.1.

Next, you need to activate the robot by selecting the ⏻ button. A list with two options will unroll. Click the ⚡ icon to activate the robot. (In case you are familiar with our Meca500 six-axis robot, note that the MCS500 does not need homing, contrary to the Meca500.)

The code editor panel

The code editor is used mainly for writing and executing simple programs, i.e., for testing. These programs are sequences of the proprietary commands described in the Programming Manual. The robot's command interface does not support conditionals, loops, or other flow control statements, nor variables. Comments are entered in C/C++ style (e.g., *//* and */* */*).

For complex tasks, you must write a program outside the web interface (e.g., in your preferred integrated development environment) that parses the robot's feedback, controls the robot, and handles all flow control logic. You can use any language that supports communication over TCP/IP (e.g., C/C++, C#, Python, Java or even Structured Text). Note that we offer a Python API on our GitHub account.

Using programs

You can create a new program using the $+$ button in the upper right corner of the panel. The program name can be changed and the program saved to the robot's non-volatile memory by either double-clicking on the program tab or by clicking the icon ▤ at the bottom of the code editor panel, which is equivalent to pressing the shortcut `Ctrl+Shift+s`. You can also save the program directly using the shortcut `Ctrl+s`.

Programs can be saved in different folders. To specify a folder, and simultaneously create it if it does not exist, simply type the name of the folder followed by a slash, immediately before the name of the program. You can specify multiple layers of folders. Naturally, the program names in a given folder must be unique, as well as the sub-folder names in a given folder.

Program and folder names are case sensitive and must contain a maximum of 63 characters among the 62 alphanumerical (A..Z, a..z, 0..9), the underscore, the hyphen and the period.

A yellow dot to the right of the program's name indicates that the program has been changed and needs to be saved. If a syntax error is found in the code, during the saving, the yellow dot will turn red, and a red ⚠ will appear in front of each line of code containing a syntax error. Note that the syntax validation is performed by the robot (and not by the MecaPortal) when the program is saved. Local changes won't be validated until the program is saved again.

The programs that appear in the code editor panel are simply those opened in the MecaPortal and not necessarily in the robot's memory. The MecaPortal will locally save (in the browser's local storage) the opened programs, whether or not they are saved on the robot. To open a program that is already in the robot's memory, click on the ⟨⟩ icon. You can also delete programs from the same menu. Alternatively, you can delete a program that is opened and in focus, by clicking the 🗑 icon. To simply close an open program, click the × on the tab with the program name.

A program can be called in another program with the StartProgram command, as in the following example:

```
StartProgram("my_folder/my_subfolder/program123")
```

To save and load programs from the robot, you must be connected to it in control mode, i.e., the connection state icon displayed in the menu bar must be ✋.

You can also save the contents of the active code editor tab to your computer's Downloads folder by clicking the ⭳ icon. The file will be saved with the .mxprog extension. In addition, you can load a program from your computer, by clicking the ⭱ icon.

Writing programs

The code editor provides syntax highlighting. In addition, once you start typing a command, a list of suggestions appears, allowing auto-completion. Furthermore, a brief description of the command appears, as shown in Figure 1. Additionally, pressing the secondary mouse button when your mouse cursor is over the code editor text field brings a context menu with all available commands (Figure 2). The most important group is in "TO CURRENT POSITION" which inserts the selected command at the text cursor with the arguments corresponding to the current robot position, the current tool reference frame (TRF), and the current world reference frame (WRF). To cancel the context menu without inserting a command, press *Esc* or click away.
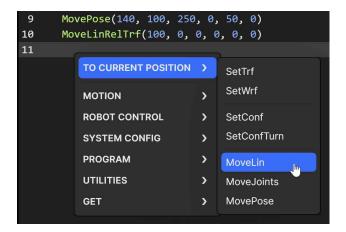
Figure 2: Context menu in the code editor

The code editor includes various editing features for productive source code editing. The shortcuts for accessing these features are presented in Table 2. Note that you can select a block of text, a box of text (column select), or even have multiple cursors. Each line that has a text cursor or a selection of text becomes highlighted in gray and is called an *active line* (page 31). For example, in Figure 2, line 11 is an active line.

Table 2: Shortcuts for course code editing

Shortcut	Action
Ctrl+Space	Open the auto-complete popup menu
Ctrl+Shift+Space	Open the command help popup
Ctrl+a	Select all
Ctrl+d	Add selection to next find match
Ctrl+f	Open the find and replace dialog
Ctrl+g	Open the "go to line" dialog
Ctrl+/	Comment/uncomment active line(s)
Ctrl+Home	Move cursor to the beginning of the first line
Ctrl+End	Move cursor to the end of the last line
Ctrl+→	Move cursor to next group
Ctrl+←	Move cursor to previous group
Double click	Select group
Triple click	Select line
Alt+↑	Move active line(s) up
Alt+↓	Move active line(s) down
Shift+Alt+↑	Duplicate active line(s) and insert result above
Shift+Alt+↓	Duplicate active line(s) and insert result below
Alt + click & drag	Select a box of text (column select)
Ctrl + click	Place multiple cursors for simultaneous editing

Executing programs

You can execute the complete current program (even if it is not saved yet) by pressing the ▶ button at the bottom of the code editor panel. Alternatively, you can execute only the active line(s), by pressing the ▶| button, or by pressing *Ctrl+Enter*.

Once the robot starts executing motion commands, the ■ button will become highlighted in yellow, and the green status bars in the menu panel will start blinking. You can press that button (or *Alt+;*) at any time to stop the robot and clear its motion queue (i.e., to send the ClearMotion command to the robot). Alternatively, you can press the ‖ button (or *Alt+.*) to pause the execution of motion commands (i.e., to send the PauseMotion command to the robot). To resume the execution of motion commands, press the ‖ button again (which sends the ResumeMotion command).
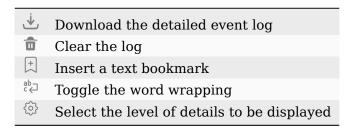
At any time, if a motion error occurs (either while the robot is executing a command or is idle), the button will become red. Pressing that button, or clicking the "Reset error" link in the red popup that will appear, acknowledges and resets the error (i.e., the MecaPortal sends the ResetError command to the robot). Once the error was reset, you need to ResumeMotion, by clicking the ! button (now blue).

Finally, recall that you can execute other programs, using the command StartProgram. With that command, you can only execute programs that have already been saved on the robot.

The log panel

In this panel, various events are listed such as commands sent, responses received, robot state changes, etc. You can control the level of details to be displayed.

The control buttons at the bottom of the log panel are self-explanatory:

⤓	Download the detailed event log
🗑	Clear the log
⊞	Insert a text bookmark
ab↵	Toggle the word wrapping
⚙	Select the level of details to be displayed

Other robot logs can also be downloaded from the configuration menu, ☰. Read *troubleshoot-prog* of the Programming Manual.

The 3D view panel

The 3D view window shows a perspective projection of the robot in its current position, as well as the current *WRF* (page 34) (World Reference Frame) and *TRF* (page 33) (Tool Reference Frame). To zoom in and out, place your mouse cursor over the 3D view window and use your mouse wheel. To change the view angle, press the primary mouse button (if the ↻ button is highlighted) or the secondary mouse button (if the ✛ button is highlighted) inside the 3D view window, hold it down and drag the mouse. To pan, press the primary mouse button (if the ✛ button is highlighted) or the middle mouse button (if the ↻ button is highlighted) inside the 3D view window, hold it down and drag the mouse. You can also select one of several preset views from the 🧊 menu.

Finally, you can show/hide the TRF, WRF, or the floor, and switch between perspective and orthographic projection from the ⚙ menu. You can also download a high-resolution screen capture of the current 3D view by pressing the 📷 button.

The robot position display

As soon as the MecaPortal appears, the current robot position appears at the top of the 3D view panel. By default, we show:

- the four joint positions, θ_1, θ_2, and θ_4, in degrees, and d_3, in mm,

- the position (x, y, z) and orientation (γ) of the TRF with respect to the WRF, in mm and degrees, respectively, and

- the robot configuration parameter.

One of the main limitations, when operating a robot, is its joint limits, which are different for each joint. We have therefore added various visual cues (patent pending), that represent the approximate joint position, relative to the corresponding joint limits. These cues are the up-pointing triangles under each joint position text field. The horizontal position of each triangle relative to the left and right limits of the corresponding text field, is proportional to the position of the corresponding joint relative to the joint limits. Furthermore, when the joint position is within 5° of the joint limit, the corresponding text field is highlighted in yellow, while if the joint limit is attained, the text field turns red.

Similarly, when the angle of joint 2 is close to or equal to 0° (elbow singularity), the corresponding text field is highlighted in yellow.

Another set of highly useful features can be accessed in the selection menus on the right of each of the three rows of data. You can select to either copy the data that is shown in the corresponding row by pressing the icon on the right (the icon is changed to ⧉) or insert a specific command with arguments corresponding to the current robot position (the icon is changed to </>). In addition, in the case of the first row of data, you can choose to show either the joints calculated or actual positions, the joints calculated or actual velocities, or even the joints torques. Similarly, in the case of the second row of data, you can choose to show either the calculated or actual pose of the TRF with respect to the WRF, or the calculated or actual Cartesian velocity of the TRF with respect to the WRF.

Table 3 shows the default keyboard shortcut for copying data from the robot position display or inserting instructions with that data at the current cursor in the code editor.

Table 3: Default keyboard shortcuts related to current robot position and reference frame definitions

Shortcut	Action
Alt+j	Insert MoveJoints command with current joint set as arguments
Alt+l	Insert MoveLin command with current TRF pose as arguments
Alt+p	Insert MovePose command with current TRF pose as arguments
Alt+c	Insert SetConf command with current posture configuration parameters as arguments
Alt+w	Insert SetWrf command with current WRF pose as arguments
Alt+t	Insert SetTrf command with current TRF pose as arguments
Alt+Shift+j	Copy current joint set
Alt+Shift+p	Copy current TRF pose with respect to the WRF
Alt+Shift+c	Copy the current posture configuration parameters
Alt+Shift+w	Copy the current definition of the WRF
Alt+Shift+t	Copy the current definition of the TRF

The jogging panel

The jogging panel is used to jog (move) the robot in several different ways (in the Cartesian or joint space) and with several different input devices (mouse, keyboard shortcuts or joystick). Once a desired robot joint arrangement is reached, you can use the context menu in the program editor field to insert a motion command with current joint set or TRF pose.
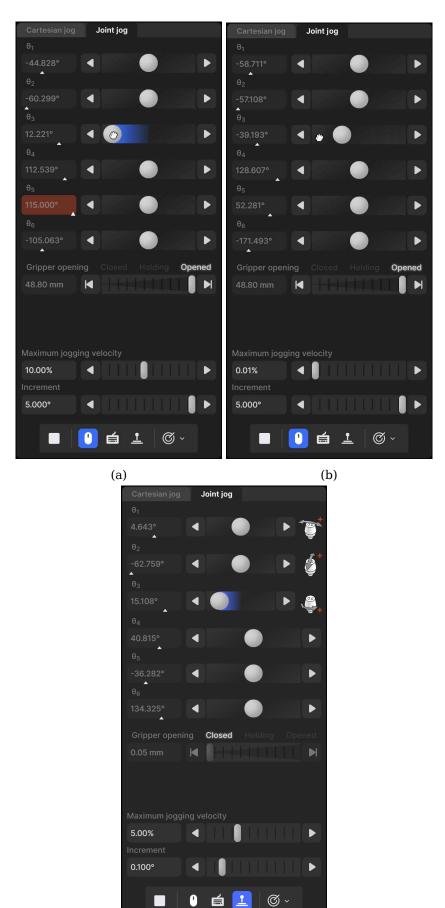
The jogging panel is enabled whenever the robot is ready to receive AND execute new motion commands. Every motion that can be commanded in the jogging panel can also be performed by sending commands from the code editor panel, though in a much less user-friendly manner. For safety reasons, when jogging, the robot can move only up to 50% of its maximum speed.

The jogging panel has two tabs that will be described in the following sections.

The joint jog tab

We use an innovative UI element (patent pending) for jogging each joint. That jogging bar is a mixture of a slider and a virtual single-axis joystick and has dual functionality, depending on which mouse button is used to interact with it (Figure 3). You can also jog each joint using keyboard shortcuts, as well as a three-axis or a six-axis joystick with at least two buttons. (Technically, you can use other joysticks, such as a PlayStation controller, but the icons displayed will be those for the SpaceMouse.)

By default, only the mouse is activated, and the button ▯ at the bottom of the jogging panel is highlighted in blue. To jog each axis as if using a joystick, click the primary mouse button onto the thumb of the corresponding jogging bar and drag it right or left (Figure 3a). The further you drag the thumb away from its central position, right or left, the faster the joint turns, in positive or negative direction, respectively. In other words, you are directly controlling the joint speed, rather than the joint position. Alternatively, you can directly click with the primary mouse button on either of the two arrow buttons of the jogging bar and hold, which makes the joint rotate at the specified maximum jogging velocity.

(a)   (b)

(c)

Figure 3: Joint jog tab when using (a) the primary mouse button, (b) the secondary mouse

You can also drag the thumbs of the jogging bars using the secondary mouse button (Figure 3b). Doing so, it is the mouse horizontal velocity (rather than relative position) that is mapped to the robot joint velocity. Additionally, clicking the left or right arrow buttons with the secondary mouse button commands a single incremental joint displacement, with the increment specified in the specified in the text field at the bottom of the jogging panel.

You can also directly specify the joint value by entering the joint angle in the text field next to each jogging bar. Note that each text field changes to yellow when the joint position is close to one of its limits or to a singularity (joints 3 and 5 only), or to red, when the joint position is at a limit or at a singularity, as is the case with the robot position display in the 3D view panel. In addition, the little triangle underneath each bar indicates the relative joint position with respect to the joint limits.

You can use keyboard shortcuts to jog the robot by activating the toggle button ⌨. Table 4 the default keyboard shortcuts for jogging the robot. The shortcuts are active only if the toggle button ⌨ is selected and the mouse cursor is over the jogging panel, in which case the panel is surrounded by a blue frame. You can change most of these shortcuts in the configuration menu, ≡.

Table 4: Keyboard shortcuts for jogging (focus must be on jogging panel)

Shortcut	Action
Shift+q[1] or *q*	Rotate joint 1 or move end-effector along x-axis, in negative direction
Shift+w or *w*	Rotate joint 1 or move end-effector along x-axis, in positive direction
Shift+a or *a*	Rotate joint 2 or move end-effector along y-axis, in negative direction
Shift+s or *s*	Rotate joint 2 or move end-effector along y-axis, in positive direction
Shift+z or *z*	Rotate joint 3 or move end-effector along z-axis, in negative direction
Shift+x or *x*	Rotate joint 3 or move end-effector along z-axis, in positive direction
Shift+e or *e*	Rotate joint 4 or rotate end-effector about x-axis, in negative direction
Shift+r or *r*	Rotate joint 4 or rotate end-effector about x-axis, in positive direction
Shift+d or *d*	Rotate joint 5 or rotate end-effector about y-axis, in negative direction
Shift+f or *f*	Rotate joint 5 or rotate end-effector about y-axis, in positive direction
Shift+c or *c*	Rotate joint 6 or rotate end-effector about z-axis, in negative direction
Shift+v or *v*	Rotate joint 6 or rotate end-effector about z-axis, in positive direction
Shift+arrow or *arrow*	Jog along each of the four directions chosen in the jogging pad
1	Open Cartesian jog panel and select TRF mode
2	Open Cartesian jog panel and select WRF mode
3	Open joint jog panel
.	Toggle between TRF and WRF mode, if Cartesian jog panel open
-	Reduce jog velocity
=	Increase jog velocity

Finally, you can jog the robot using either a 3-axis USB precision joystick (Figure 4a) or the SpaceMouse® 6-axis joystick from 3Dconnexion (Figure 4b). To use a 3-axis or a 6-axis joystick, you need to activate the button ⊥. For the MecaPortal to detect the joystick, you will need to press one of the device's buttons or, occasionally, unplug and replug the device. Once the joystick is detected you will see the respective joystick icons next to each slider (Figure 3c).

---

[1] Hold Shift for continuous jog, otherwise robot will jog in increments.

> ☢ **Danger**
>
> Contrary to the keyboard shortcuts for jogging the robot, the joystick is active even if the mouse cursor is not over the jogging menu, as long as the focus is on the MecaPortal.

> ☢ **Danger**
>
> *Having multiple input modes for jogging activated simultaneously is unsafe.* You should enable only one input mode at a time: the mouse, keyboard shortcuts, or the joystick.

> ⓘ **Note**
>
> Do not install the driver and the software that come with the SpaceMouse as these will interfere with the desired functioning of the device in the MecaPortal. Also, if you opt for the wireless model, keep the device close to the universal USB receiver and remember to recharge the device regularly.

You can deactivate certain joystick directions by clicking on the respective joystick icon, to the right of each slider bar. You can even customize the mapping between the joystick "axes" and the jogging directions in the configuration menu, ≡.

(a) an example of a 3-axis USB precision joystick (b) SpaceMouse® 6-axis joystick from 3Dconnexion

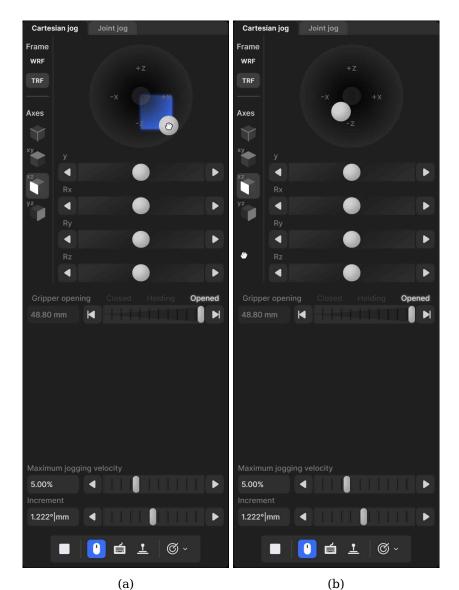Figure 4: USB joysticks that can be used with the MecaPortal

To select between joint jog, Cartesian jog with respect to the TRF or Cartesian jog with respect to the WRF, you can either use your mouse or press (up to two times if necessary) the right button of the SpaceMouse or of the 3-axis joystick until the desired selection is obtained.

For much more precise jogging, we strongly recommend the use of a 3-axis precision USB joystick such as from APEM.

Finally, for convenience, the same stop button ▣ available at the bottom of the code editor is also provided at the bottom of both jogging menus. Lastly, a list of predefined robot positions (essentially, joint sets) are stored and available in the ⊘ menu button. Clicking on one of these robot positions moves the robot to that joint set, in joint mode.

The Cartesian jog tab

In the Cartesian jog tab (Figure 5), you can move the robot along (labels X, Y, Z) or about (labels Rx, Ry, or Rz) the axes of the TRF (if the TRF button is selected, as in the figure) or along or about the axes of a reference frame having the same orientation as the WRF but with origin at the TCP (if WRF is selected). As with the joint jog tab, the jogging bars can be controlled with both the primary (Figure 5a) and secondary (Figure 5b) mouse buttons, using the same logic. However, you can merge two jogging bars for linear movement (i.e., X and Y, Y and Z, or Z and X) into a 2D jogging pad using the buttons under the label "Axes". *Note that subsequent clicks on one of these buttons changes the jogging directions in the 2D jogging pad* (there are a total of eight combinations).

(a)   (b)

The other major difference is that there are no text fields showing the position and orientation of the TRF with respect to the WRF, next to each jogging bar, let alone visual cues for indicating whether Cartesian jogging in one direction might soon reach a workspace limit. This is mainly because, in general, jogging along and about the three axes of a reference frame have no one-to-one correlation to the position coordinates and Euler angles used to represent the pose of the TRF with respect to the WRF. Furthermore, other than mechanical interferences between non-adjacent links of the robot, the factors that limit the robot workspace are the joints limits and the singularities, which are already displayed in the robot position display, in the top right corner of the MecaPortal.

The keyboard shortcuts for Cartesian jogging are shown in Table 4.

Finally, you can jog in Cartesian mode using a joystick. Figure 5c shows the Cartesian jog tab in the case of the SpaceMouse.

> ☢ **Danger**
>
> *Having multiple input modes for jogging activated simultaneously is unsafe.* You should enable only one input mode at a time: the mouse, keyboard shortcuts, or the joystick.

The configuration menu

The configuration menu can be accessed from the top left corner of the MecaPortal window, ≡. It features the following sections:

- Robot information, such as the serial number and the complete version number of the firmware;

- Help, primarily links to documentation, the support website, and to the legacy web interface;

- Network configuration, such as entries of the IP address of the robot or the netmask,

- Joint limits, for reducing the joint ranges;

- Work zone limits and collision configuration, for preventing self-collisions and defining a work zone;

- Variables management (creation, deletion or modification of robot persistent variables);

- Update firmware, for upgrading (or downgrading) the firmware of the robot;

- Get logs and backup, for downloading a complete backup (robot logs, persistent configuration data, programs, etc.) or the latest robot logs only;

- Keyboard mapping, for listing all and changing certain keyboard shortcuts;

- Joystick mapping, for assigning robot jogging directions to joystick axes.

Terminology

Below is the list of terms used by us in our technical documentation.

active line: The line in the MecaPortal where the cursor is currently positioned.

BRF: Base Reference Frame.

Cartesian space: The four-dimensional space defined by the position (x, y, z) and orientation (γ) of the TRF with respect to the WRF.

control port: The TCP port 10000, over which commands to the robot and messages from the robot are sent.

data request commands: Commands used to request some data regarding the robot (e.g., GetTrf, GetBlending, GetJointVel). These commands are executed immediately and generally return values for parameters that have already been configured (sent and executed) with a Set* command (or the default values).

default value: There are different settings in the robot controller that can be configured using Set* commands (e.g., SetCartAcc). Many of these settings have default values. Every time the robot is powered up, these settings are initialized to their default values. In the case of motion commands settings, their values are also initialized to their default values every time the robot is deactivated. In contrast, some settings are persistent and their values are stored on an SD drive.

detailed event log: This file mirrors the content of the event log panel in the MecaPortal when in detailed mode. It can be downloaded from the MecaPortal (see `troubleshoot-prog` of the Programming Manual).

distal link: In the MCS500, this is the black-anodized body that holds the spline shaft.

EOAT: End-of-arm tooling.

EOB: End-of-block message, [3012][], sent by default every time the robot has stopped moving AND its motion queue is empty. You can disable this message with the command SetEob.

EOM: End-of-motion message, [3004][], sent by the robot whenever it has stopped moving for at least 1 ms, if this option is activated with SetEom.

error mode: The robot goes into error mode when it encounters an error while executing a command or a hardware problem (see `tab:error-messages`).

FCP: Flange Center Point. The origin of the FRF.

FRF: Flange Reference Frame.

instantaneous commands: These are commands that are executed immediately, as soon as received by the robot. All data request commands (Get*), all robot control commands, all work zone supervision and collision prevention commands and some optional accessories commands (*_Immediate) are instantaneous.

inverse kinematics: The problem of obtaining the robot joint sets that correspond to a desired end-effector pose. See *inverse-kinematics* of the Programming manual for more details.

joint position: The joint angle associated with a rotary joint or the position of joint 3.

joint set: The set of all joint positions.

joint space: The four-dimensional space defined by the positions of the robot joints.

monitoring port: The TCP port 10001, over which data is sent periodically from the robot.

motion commands: Commands used to construct the robot trajectory (e.g., Delay, MoveJoints, SetTRF, SetBlending). When a Mecademic robot receives a motion command, it places it in a motion queue. The command will be run once all preceding motion commands have been executed.

motion queue: The buffer where motion commands that were sent to the robot are stored and executed on a FIFO basis by the robot.

offline program: A sequence of commands saved in the internal memory of the robot. The term *offline* is often omitted and will eventually be removed altogether.

online mode programming: Programming the robot in online mode involves moving it directly to each desired robot position, typically using jogging controls.

PDO (**Process Data Object**): In EtherCAT, a Process Data Object (PDO) is a data structure used for exchanging real-time cyclic data between an EtherCAT master and its slave devices. PDOs can contain individual bits, bytes, or words.

persistent settings: Some settings in the robot controller have default values (e.g., the robot name set by the command SetRobotName), but when changed, their new values are written on an SD drive and persist even if the robot is powered off.

pose: The position and orientation of one reference frame with respect to another.

position mode: One of the two control modes, in which the robot's motion is generated by requesting a target end-effector pose or joint set (see *pos-vel-modes* of the Programming Manual).

robot posture configuration: The two-value (-1 or 1) parameter c_e that normally defines each of the two possible robot postures for a given pose of the robot's end-effector.

proximal link: This is the clear-anodized body between the base and the distal link.

queued commands: Commands that are placed in the motion queue, rather than executed immediately. All motion commands are queued commands, as well as some external-tool commands.

reach: The maximum distance between the axis of joint 1 and the axis of joint 4.

real-time data request commands: Commands used to request some real-time data regarding the current status of robot (e.g., GetRtTrf, GetRtCartPos, GetStatusRobot).

retaining ring: In the MCS500, this is each of the two circular clamps on both ends of the spline shaft.

robot control commands: Commands used to immediately control the robot, (e.g., ActivateRobot, PauseMotion, SetNetworkOptions). These commands are executed immediately, i.e., are instantaneous.

robot is ready for motion: The robot is considered *ready* to receive motion commands, i.e. when it is activated.

Note that if the robot is in error or if a safety stop condition is present, it will refuse motion commands, but it will still be considered *ready* since its motion queue remains initialized and retains the latest received settings (e.q., velocity, acceleration, blending, WRF, TRF, etc.).

robot log: This file is a more detailed version of the user log, intended primarily for our support team. It can be downloaded from the MecaPortal (see `troubleshoot-prog` of the Programming Manual).

robot position: A robot position is equivalent to either a joint set or the pose of the TRF relative to the WRF, along with the definitions of both reference frames, and the robot posture and last joint turn configuration parameters.

robot posture: The arrangement of the robot links. Equivalent to a joint set in which all joint angles are normalized, i.e. have been converted to the range $(-180°, 180°]$.

SDO (**Service Data Object**): In EtherCAT, a Service Data Object (SDO) is a data structure used for non-real-time communication between an EtherCAT master and its slave devices. SDOs are typically used to configure device parameters and access diagnostic information through the object dictionary. Unlike PDOs, SDOs exchange structured data rather than individual bits or bytes.

singularities: A robot posture where the robot's end-effector is blocked in some directions even if no joint is at a limit (see `singularities` of the Programming Manual).

spline shaft: This is the the groved reciprocating shaft.

TCP: Tool Center Point. The origin of the TRF. Not to be confused with Transmission Control Protocol.

TRF: Tool reference frame.

turn configuration parameter: Since the last joint of the robot can rotate multiple revolutions, the turn configuration parameter defines the revolution number.

user log: This file is a simplified log containing user-friendly traces of major events (e.g., robot activation, movement, E-Stop activation). It can be downloaded from the MecaPortal (see `troubleshoot-prog` of the Programming Manual).

velocity mode: One of the two control modes, in which the robot's motion is generated by requesting a target joint velocity vector or end-effector Cartesian velocity vector (see `pos-vel-modes` of the Programming Manual).

workspace: The Cartesian workspace of a robot is the set of all feasible poses of its TRF with respect to its WRF. Note that many of these poses can be attained with more than one set of

configuration parameters.

WRF: World reference frame.