Migration to Firmware 8

Guide for Mecademic's existing clients

April 17, 2020

What's New?

From a user point of view, there are only a few major changes and improvements in firmware 8, but some modify the behaviour of the robot and may require that you slightly modify your programs, should you choose to upgrade. Of course, our technical support team is ready to assist you in the migration. We therefore recommend that you perform the migration, even if your robots are already in production.

Improved Management of Joint 6

One of the main problems in firmware 7 was the management of joint 6. While θ_6 was and continues to be constrained within the range ±180° when the robot executes a Cartesian-space command, in firmware 7, when θ_6 is outside this range at the beginning or exits it during the Cartesian-space motion, joint 6 is immediately "rewound" n360° (where n is a nonzero integer) until θ_6 is brought in the ±180° range. For example, in firmware 7, if you start linearly jogging your robot when $\theta_6 = 541°$, joint 6 will first quickly rotate to −179° before starting the linear movement. Similarly, if during the motion, θ_6 increments gradually and reaches 180°, for example, the robot stops, rewinds joint 6 to −179.999° and then continues to increment the new value of θ_6, instead of incrementing directly to 180.001°. In both cases, the TCP (tool center point) unexpectedly leaves the desired linear path, which is obviously unsafe.

In firmware 8, none of these two possible cases can occur. When executing a Cartesian-space command, joint 6 must already be in the ±180° range and must remain in that range at all times, or else the command will not be executed.

Because some of you did not mind these unexpected n360° rotations of joint 6 and have robots performing these movements in production, we decided to leave the "bug" uncorrected by default, if we detect that a robot is being upgraded from firmware 7. However, we strongly advise you to modify your programs so that you do not rely on such movements. To prevent these unexpected rotations, run the following command only once (no need to run it every time the robot is activated), while the robot is powered but not activated:

```
SetMotionOptions(1,0,0,0,0,0)
```

To re-introduce the unexpected rotations, run

```
SetMotionOptions(0,0,0,0,0,0)
```

Note that the above command is not described in the Programming Manual as it is aimed only at clients who need to upgrade from firmware 7.

Added Protection Against Large Joint Movements

In firmware 7, you can execute a Cartesian-space command where joints 4 and 6 rotate each more than 180° when you pass close to a wrist singularity (a robot configuration in which θ_5 is close to 0°). Here is an example:

```
SetTRF(0,0,0,0,0,0)
SetWRF(0,0,0,0,0,0)
MoveJoints(-60,0,0,95,60,-90)
MoveLinRelWRF(0,208,0,0,0,0)
```

Even worse, you can execute Cartesian-space commands where joint 6 rotates close to 360°. Here is one such example inspired by one of our customers, where joint 6 rotates more than 316°:

```
SetTRF(0,0,0,0,0,0)
SetWRF(0,0,0,0,0,0)
MoveJoints(70,48,-47,0,-91,-171)
MoveLinRelWRF(0,-420,0,0,0,175)
```

As of firmware 7, a single Cartesian-space motion in which joint 4 or joint 6 rotates more than 180° will no longer be allowed, for safety reasons. You would need to decompose such a motion into two (or more) Cartesian-space motions.

Again, for legacy reasons, if we detect that a robot is migrating from an older firmware to firmware 8, we will automatically disable this 180° limitation. The limit will be re-enabled, however, if you execute the command SetMotionOptions(1,0,0,0,0,0).

Improved Cycle Times

We have increased the linear velocity limit to 1000 mm/s and the angular velocity limit to 300 °/s. However, remember that these limits are for Cartesian-space motion commands only. In joint mode, the TCP might move at even higher speeds (e.g., if you fully stretch the robot arm horizontally, rotate joint 1 at full speed, and define a TCP that is far from the robot's flange).

Furthermore, improvements in our motion planner now allow the robot to accelerate in some situations up to six times quicker. That is why, the argument of the SetCartAcc command can now be up to 600. Thus, in firmware 8, 100% Cartesian acceleration corresponds to approximately the same maximum acceleration as in firmware 7 and before.

Velocity Mode

As our robots are sometimes used in advanced applications such as micromanipulation, force control, or precision path following, we decided to introduce a new mode for displacing our robots. We call this mode the **velocity mode**, whereas we now call the traditional mode (MoveLin, MovePose, MoveJoints, etc.), the **position mode**. In velocity mode, you specify the six-dimensional velocity vector of the robot end-effector or of the robot joints by sending special new commands (MoveLinVelTRF, MoveLinVelWRF and MoveJointsVel). In contrast with the position mode where commands are queued, in velocity mode, as soon as a new velocity-mode command is received by the robot, it is immediately executed. If no new motion command is sent, the maximum duration of the last velocity-mode command is the one specified by the command SetVelTimeout. The Velocity mode is explained in detail in Section 2.2.5.

Checkpoint

In firmware 7, when you sent several motion commands, they were put in a queue and there was no way of knowing when a specific motion command was finally executed. To remedy this problem, we have introduced the **SetCheckpoint(n)** command, which is to be sent just before the motion command of interest. Immediately before executing the motion command in question, the robot will return a message with the checkpoint number.

Web Interface

The web interface undertook multiple cosmetic and usability improvements, but the main changes come from the introduction of the velocity mode. Jogging is now performed in velocity mode. We have introduced the possibility to control the jogging speed and we now offer the possibility to jog the robot with 3Dconnexion's **SpaceMouse**® (readily available from various online stores).

In velocity mode, the robot motion is significantly affected by the acceleration settings and the so-called velocity timeout. To keep the jogging performance consistent between sessions, we now have to set these settings to certain optimal values every time you wish to jog the robot. In addition, for safety reasons, we wanted to prevent the possibility to jog the robot while the latter is executing a program. This is why we introduced the **Jogging toggle button**. You now have to switch between jogging and programming mode. We understand that this adds an additional action that is to be repeated often, so we have added the shortcut Alt+T to toggle between the two modes. Note, however, that while in jogging mode, you can still use the context menu in the programming editor and add commands relative to the current robot position.

We kept the unique possibility to **jog in incremental mode**, though due to the nature of the velocity mode, you can no longer set an arbitrary increment. This limitation, however, will be lifted soon.

To improve usability, we introduced a few additional **shortcuts**. For example, you can close and open the gripper with Alt+G and run the command where the cursor is in the programming editor with Ctrl+Enter.

Because the performance of the **CAD model import** feature was inconsistent, we temporarily removed the Tool & Environment tab. In the meantime, you can use some of the robot simulation software products that support our robots.

Finally, we added a **Request Log tab**, which shows every command sent to the robot, except the velocity-mode commands sent (every few milliseconds) while jogging.

EtherCAT

Loosely speaking, the EtherCAT protocol is now a closer translation to our TCP/IP protocol. For example, all errors are now reported in a PDO with the same error number as in the TCP/IP protocol. Furthermore, PDOs changed significantly. We also added some new commands, such as one used to disable the brakes, and PDOs for the velocity, torque ratios, gripper status, and one of the accelerometers embedded in the robot.

Other Features

We also added the **SetNetworkOptions** command to allow controlling the number of TCP idle packets that can be lost before closing the connection.

In addition, **robot status** and events are now sent on the monitoring channel (port 10,001) every time there is a state change.

Finally, we fixed the problem with the occasional losses of the TCP connection and the problem with some missing EOB messages.

What’s Coming Up?

In the coming months, our R&D team will be working on the following new features.

Optimal Management of Joint 6

It will be possible to move with Cartesian-space motion commands while having joint 6 outside its ±180°, by specifying an additional configuration parameter. This will effectively enlarge the Cartesian workspace of our robots.

Possibility to Modify the Joint Limits

We plan to introduce two commands for modifying the lower and upper limits of all six joints.

Web Interface

New commands will be added to make it possible to jog in incremental mode, with any increment.

New Get Commands

For all Set commands (e.g., SetTRF, SetJointAcc), we will introduce a corresponding Get command (e.g., GetTRF, GetJointAcc). Indeed, currently, if you disconnect from a robot and connect to it from another client (i.e., from another computing device or simply from another program), there is no way to retrieve all the parameters that were set through the other client.

Ethernet/IP

We will add the Ethernet/IP protocol.

Any Suggestions?

We would appreciate greatly if you continue to send us feedback about our products.