ABB

ROBOTICS

Application manual
Externally Guided Motion

Application manual

Externally Guided Motion

RobotWare 6.15.08

Document ID: 3HAC073319-001

Revision: H

Table of contents

Table of contents

Overview of this manual

About this manual

This manual contains information about the RobotWare option Externally Guided Motion [689-1], often referred to as EGM.

Usage

This manual can be used to find out what Externally Guided Motion is and how to use it. The manual also provides information about RAPID components and system parameters related to Externally Guided Motion, and examples of how to use them.

Who should read this manual?

This manual is mainly intended for robot programmers.

Prerequisites

The reader should be familiar with:

- Industrial robots and their terminology
- The RAPID programming language
- System parameters and how to configure them

References

Reference	Document ID
Application manual - Controller software IRC5	*3HAC050798-001*
Operating manual - IRC5 with FlexPendant	*3HAC050941-001*
Operating manual - RobotStudio	*3HAC032104-001*
Technical reference manual - RAPID Overview	*3HAC050947-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC050917-001*
Technical reference manual - System parameters	*3HAC050948-001*

Revisions

Revision	Description
A	Released with RobotWare 6.10. The information in this manual was previously part of the manuals *Application manual - Controller software IRC5*, *Technical reference manual - RAPID Instructions, Functions and Data types*, and *Technical reference manual - System parameters*.
B	Released with RobotWare 6.10.01. • Changed default value for the parameter *Default Ramp Time on page 40*.

Continued

Revision	Description
C	Released with RobotWare 6.11. • Description of the *K factor* (Default proportional Position Gain) added in section *Input data on page 20*. • Note regarding unpredictable movement added in section *Default Low Pass Filter Bandwidth on page 42*. • Limitations updated for instructions *EGMRunJoint - Perform an EGM movement with a joint target on page 65* and *EGMRunPose - Perform an EGM movement with a pose target on page 68*. • Limitations updated in section *Overview on page 9*. • Removed the argument LATR from instructions EGMSetupAI, EGMSetupAO, and EGMSetupGI. This is no longer available. • New set-up section added in *Introduction to EGM Position Guidance on page 13*.
D	Released with RobotWare 6.12. • Section *Overview on page 27* updated with information about the EGM sensor protocol data structure. • Updated the section *Building an EGM sensor communication endpoint on page 33*.
E	Released with RobotWare 6.13. • Added moveIndex and CollisionIndex to *EgmRobot on page 30*. • The pose mode in *EGM Position Guidance* now also supports YuMi and SCARA robots.
F	Released with RobotWare 6.14. • *EGM Position Guidance* can now be used also for 4-axis palletizing robots. • Updated information about sending RAPID data and I/O signals. Added RAPIDfromRobot and RAPIDfromSensor to *EgmRobot on page 30*. Added the data type EGM_MAX_RAPID_DNUM, and arguments to the instructions EGMActJ and EGMActPose. • Added information for proportional controller gain, see *Input data on page 20*. • Path to EGM folder corrected in *The EGM sensor protocol on page 27*.
G	Released with RobotWare 6.15.07. • Correction done in section *Execution states on page 19*.
H	Released with RobotWare 6.15.08. • Minor corrections.

1 Introduction to Externally Guided Motion

1.1 Overview

Purpose

Externally Guided Motion (EGM) offers three different features:

- *EGM Position Stream*:

  The current and the planned positions of the mechanical units in a RAPID task are sent to an external equipment.

- *EGM Position Guidance*:

  The robot does not follow a programmed path in RAPID but a path generated by an external device.

- *EGM Path Correction*:

  The programmed robot path is modified/corrected using measurements provided by an external device.

EGM Position Stream

The purpose of EGM Position Stream is to provide external equipment with the current and planned positions of mechanical units that are controlled by the robot controller.

Some example of applications are:

- Laser Welding, where the Laser head is controlling the Laser beam dynamically.

- Any robot mounted equipment that controls the "robot"-TCP with an external controller.

EGM Position Guidance

The purpose of *EGM Position Guidance* is to use an external device to generate position data for one or several robots. The robots will be moved to that given position.

Some examples of applications are:

- Place an object (for example a car door or a window) at a location (for example a car body) that was given by an external sensor.

- Bin picking. Pick objects from a bin using an external sensor to identify the object and its position.

EGM Path Correction

The purpose of *EGM Path Correction* is to use external robot mounted devices to generate path correction data for one or several robots. The robots will be moved along the corrected path, which is the programmed path with added measured corrections.

Some examples of applications are:

- Seam tracking.

- Tracking of objects moving near a known path.

What is included

The RobotWare option *Externally Guided Motion* gives you access to:

- Instructions to start and stop EGM Position Stream.
- Instructions to set up, activate, and reset EGM Position Guidance.
- Instructions to set up, activate, and reset EGM Path Correction.
- Instructions to initiate EGM Position Guidance movements, synchronized with RAPID execution or not, and to stop them.
- Instructions to perform EGM Path Correction movements.
- A function to retrieve the current EGM state.
- System parameters to configure EGM and set default values.

Limitations

Limitations for EGM Position Stream

- EGM Position Stream is available with UdpUc communication only.
- Tool data and load data cannot be changed dynamically during an active position stream.
- Absolute Accuracy is not supported if streaming is started using `EGMStreamStart`, but it is supported if it started using `EGMActXXX\StreamStart`.
- EGM Position Stream is not compatible with EGM Path Correction.
- It is not allowed to activate or deactivate mechanical units if EGM Position Stream is active.

Limitations for EGM Position Guidance

- Has to start and to end in a fine point.
- The first movement that is performed after a controller restart cannot be an EGM movement.
- Pose mode supports 6-axis robots, 4-axis palletizer robots, YuMi robots, and SCARA robots.
- It is not possible to perform linear movements using EGM Position Guidance, since EGM Position Guidance does not contain interpolator functionality. The actual path of the robot will depend on the robot configuration, the start position, and the generated position data.
- There is a limitation of one mechanical unit per motion task.
- It is not possible to use EGM Position Guidance to guide a mechanical unit in a moving work object.
- If the robot ends up near a singularity, i.e. when two robot axes are nearly parallel, the robot movement will be stopped with an error message. In that situation the only way is to jog the robot out of the singularity.
- When EGM is active, Motion Supervision can behave differently than during normal movements. The recommended action after a collision is to disable EGM and start the EGM sequence from the beginning.

Limitations for EGM Path Correction

- Supports only 6-axis robots.

- Has to start and to end in a fine point.

- The external device has to be robot mounted.

- Corrections can only be applied in the path coordinate system.

- Only position correction in y and z can be performed. It is not possible to perform orientation corrections, nor corrections in x (which is the path direction/tangent).

- When EGM is active, Motion Supervision can behave differently than during normal movements. The recommended action after a collision is to disable EGM and start the EGM sequence from the beginning.

Limitation for RRI

- Only supports 6- and 4-axis robots in the first RAPID task (T_ROB1).

1.2 Introduction to EGM Position Stream

What is EGM Position Stream

EGM Position Stream is available for UdpUc communication only. It provides the possibility to periodically send planned and actual mechanical unit (e.g. robot, positioner, track motion …) position data from the robot controller. The message contents is specified by the Google Protobuf definition file *egm.proto*. The cyclic communication channel (UDP) can be executed in the high-priority network environment of the robot controller which ensures a stable data exchange up to 250 Hz. There has to be one communication channel for each motion task.

EGM Position Stream may be used together with EGM Position Guidance.

1.3 Introduction to EGM Position Guidance

What is EGM Position Guidance

EGM Position Guidance is designed for advanced users and provides a low level interface to the robot controller, by by-passing the path planning that can be used when highly responsive robot movements are needed. EGM Position Guidance can be used to read positions from and write positions to the motion system at a high rate. This can be done every 4 ms with a control lag of 10–20 ms depending on the robot type. The references can either be specified using joint values or a pose. The pose can be defined in any work object that is not moved during the EGM Position Guidance movement.

> **Note**
>
> In all further descriptions of EGM and RRI the actual sampling time is 4.032 ms on a real robot system and approximately 4 ms in a virtual robot system.

All necessary filtering, supervision of references, and state handling is handled by EGM Position Guidance. Examples of state handling are program start/stop, emergency stop, etc.

The main advantage of EGM Position Guidance is the high rate and low delay/latency compared to other means of external motion control. The time between writing a new position until that given position starts to affect the actual robot position, is usually around 20 ms.

EGM handles *Absolute Accuracy*.

EGM Position Guidance can be combined with Logical settings (setting I/O:s etc.) or enabling other control modes e.g. using Force Control assembly instructions. This can be achieved by using optional argument to not wait for EGM convergence (\NoWaitCond).

What EGM Position Guidance does not do

EGM goes directly into the motor reference generation, i.e. it does not provide any path planning. This means that you cannot order a movement to a pose target and expect a linear movement. It is not possible either to order a movement with a specified speed or order a movement that is supposed to take a specified time.

For ordering such movements path planning is needed and we refer you to the standard movement instructions in RAPID, i.e. `MoveL`, `MoveJ`, etc.

> **WARNING**
>
> Since the path planning is by-passed by EGM in the robot controller, the robot path is created directly from user input. It is therefore important to make sure that the stream of position references sent to the controller is as smooth as possible. The robot will react quickly to all position references sent to the controller, also faulty ones.

Setting up EGM Position Guidance

When EGM Position Guidance is set up correctly, the external device communicates with the controller via a protocol and thereby controls the movement of the robot. The following image displays a schematic overview of a possible set-up of an EGM application:
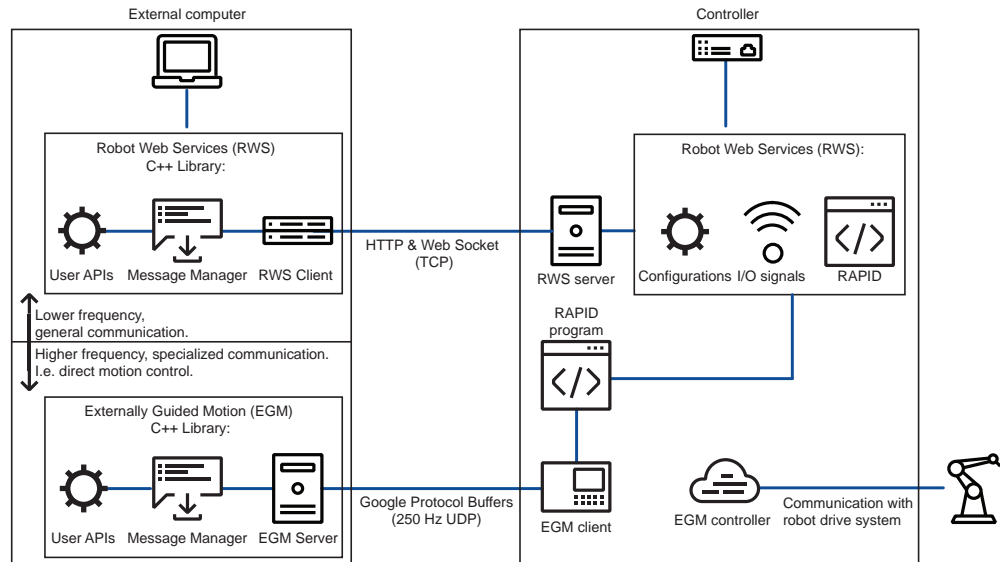
xx2000000152

For this to work, the following steps must be taken:

1   Set up your RobotWare system, including the RobotWare option *Externally Guided Motion*.

> **Note**
>
> See *Limitations on page 10* for information regarding what types of robots and options can be included in the set-up.

2   Set up your UdpUc device with an EGM sensor protocol so that it can communicate with the robot controller. See *The EGM sensor protocol on page 27* for basic instructions on how to build an EGM sensor communication endpoint.

> **Note**
>
> Code examples are available in the RobotWare distribution. See *UdpUc code examples on page 113*.

3   Set up the RAPID code defining the details of the EGM Position Guidance functionality. See *Basic approach on page 17* for a description of the basic elements needed in the code, and *Using EGM Position Guidance with an UdpUc device on page 102* for a complete example of RAPID code.

4   The device which provides the input data for EGM has to be configured as a UdpUc device. This configuration is made in RobotStudio, using the system

Application manual - Externally Guided Motion
3HAC073319-001 Revision: H

parameters of type *Transmission Protocol* in topic *Communication*. Define the name and the IP address of the device, and set transmission protocol to UdpUc. See *Configuring UdpUc devices on page 34*.

Note

After this configuration change, the controller has to be restarted. After the restart, the device can be used by EGM to guide a robot.

1.4 Introduction to EGM Path Correction

What is EGM Path Correction

EGM Path Correction gives the user the possibility to correct a programmed robot path. The device or sensor that is used to measure the actual path has to be mounted on the tool flange of the robot and it must be possible to calibrate the sensor frame.

The corrections are performed in the path coordinate system, which gets its x-axis from the tangent of the path, the y-axis is the cross product of the path tangent, and the z-direction of the active tool frame and the z-axis is the cross product of x-axis and y-axis.

EGM Path correction has to start and end in a fine point. The sensor measurements can be provided at multiples of about 48 ms.

2 Using Externally Guided Motion

2.1 Basic approach

Basic approach for EGM Position Stream

EGM Position Stream is available if UdpUc is used for communication with the external equipment. It is possible to start EGM Position Stream in two different ways. One is to use `EGMStreamStart` and the other `EGMActJoint\StreamStart` or `EGMActPose\StreamStart`. EGM Position Stream is automatically stopped by `EGMStop`, `EGMReset` and when an `EGMRunJoint` or `EGMRunPose` instruction is completed. There is also a specific instruction, `EGMStreamStop`, to stop the data stream.

Position streaming does not support dynamic change of tool or load. If `EGMStreamStart` is used to start the position stream, the active tool and load are passed to the controller. If `EGMActJoint` or `EGMActPose` is used, the active tool and load (or if specified, the specified tool and/or load) are passed to the controller. Those tool and load data are then used by EGM for calculation of positions until the position stream is stopped. For each motion task, a separate position stream has to be started.

	Action
1	Register an EGM client and get an EGM identity. This identity is then used to link setup, activation, movement, deactivation etc. to a certain EGM usage. The EGM state changes to `EGM_STATE_CONNECTED`.
2	Call the EGM setup instruction `EGMSetupUC` to set up the external equipment using UdpUc protocol connection.

3	Either:
	  A Start the position stream with the instruction `EGMStreamStart`.
	  B Start the position stream using `EGMActJoint` or `EGMActPose` with the optional argument `\StreamStart`.
	Now the EGM state is `EGM_STATE_RUNNING`.

4	EGM Position Stream will be active, sending actual and planned position, until it is stopped.

5	A If started with `EGMStreamStart`:
	    Stop the position stream with `EGMStreamStop`.
	B If started with `EGMActJoint` or `EGMActPose`:
	    Stop the position stream with `EGMStop` or `EGMReset`.
	The EGM state changes back to `EGM_STATE_CONNECTED`.

Basic approach for EGM Position Guidance

This is the general approach to move/guide a robot using an external device (sensor) to give the target for the movement.

	Action
1	Move the robot to a fine point.
2	Register an EGM client and get an EGM identity. This identity is then used to link setup, activation, movement, deactivation etc. to a certain EGM usage. The EGM state changes to `EGM_STATE_CONNECTED`.

2 Using Externally Guided Motion

	Action
3	Call an EGM setup instruction to set up the position data source using signals or UdpUc protocol connection.
4	Choose if the position is given as joint values or as a pose and give the position convergence criteria, i.e. when the position is considered to be reached.
5	If pose was chosen, define which frames are used to define the target position and in which frame the movement is to be applied.
6	Give the stop mode, an optional time-out and perform the movement itself. Now the EGM state is `EGM_STATE_RUNNING`. This is when the robot is moving.
7	The EGM movement will stop when the position is considered to be reached, i.e. the convergence criteria is fulfilled. Now the EGM state has changed back to `EGM_STATE_CONNECTED`.

Basic approach for EGM Path Correction

This is the general approach to correct a programmed path with EGM Path Correction.

	Action
1	Move the robot to a fine point.
2	Register an EGM client and get an EGM identity. This identity is then used to link setup, activation, movement, deactivation etc. to a certain EGM usage. The EGM state changes to `EGM_STATE_CONNECTED`.
3	Call an EGM setup instruction to set up the position data source using signals or UdpUc protocol connection.
4	Define the sensor correction frame, which always is a tool frame.
5	Perform the movement itself. Now the EGM state is `EGM_STATE_RUNNING`.
	At the end point EGM will return to the state `EGM_STATE_CONNECTED`.
6	To free an EGM identity for use with another sensor you have to reset EGM, which returns EGM to the state `EGM_STATE_DISCONNECTED`.

2.2 Execution states

Execution states description

These are the execution states of EGM on the robot controller:

Value	Description
EGM_STATE_DISCONNECTED	The EGM state of the specific process is undefined. No EGM correction is connected/active.
EGM_STATE_CONNECTED	Setup has been made, but no EGM movement is active.
EGM_STATE_RUNNING	The specified EGM process is running. The EGM movement is active, i.e. the robot is moved.

Transitions between the different states are according to the figure below.
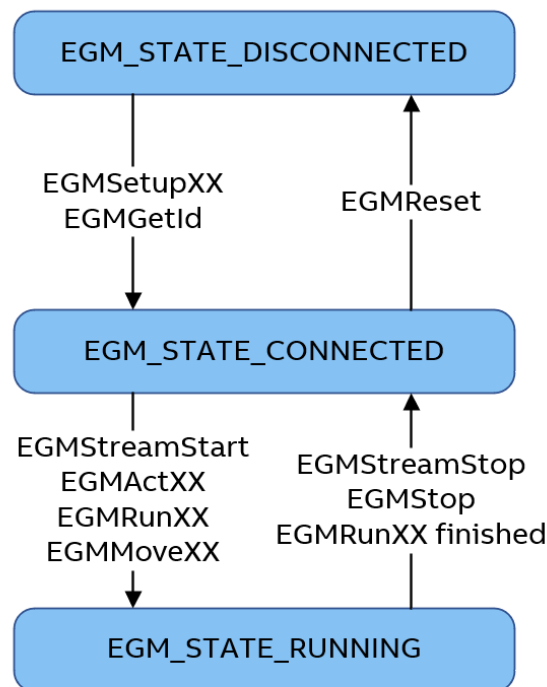
xx1400001082

The RAPID instructions `EGMRunJoint` and `EGMRunPose` change the state from `EGM_STATE_CONNECTED` to `EGM_STATE_RUNNING` as long as the convergence criteria for the target position have not been met and the time-out time has not expired. When one of these conditions is met, the EGM state is changed to `EGM_STATE_CONNECTED` again and the instruction ends, i.e. RAPID execution continues to the next instruction.

If EGM has the state `EGM_STATE_RUNNING` and RAPID execution is stopped, EGM enters the state `EGM_STATE_CONNECTED`. At program restart, EGM returns to the state `EGM_STATE_RUNNING`.

If the program pointer is moved using **PP to Main** or **PP to cursor**, the EGM state is changed to `EGM_STATE_CONNECTED`, if the state was `EGM_STATE_RUNNING`.

2.3 Input data

Input data for EGM Position Guidance

The source for input data is selected using the EGM setup instructions. The three first instructions select a signal interface and the last instruction a UdpUc interface (*User Datagram Protocol Unicast Communication*).

Instructions	Description
EGMSetupAI	Setup analog input signals for EGM
EGMSetupAO	Setup analog output signals for EGM
EGMSetupGI	Setup group input signals for EGM
EGMSetupUC	Setup the UdpUc protocol for EGM

Input data for EGM contain mainly position data either as joints or as a pose, i.e. Cartesian position plus orientation.

The data flow for the signal interface is illustrated below:
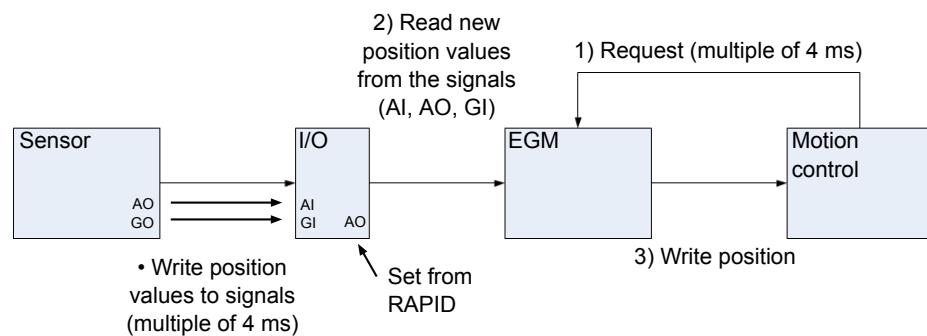
xx1400002016

1   Motion control calls EGM.

2   EGM reads the position values from the signals.

3   EGM writes the position data to motion control.

•   The sensor writes position data to the signals.

If signals are used as data source, the input is limited to 6 for the robot, i.e. 6 joint values or 3 Cartesian position values (x, y, z) plus 3 Euler angle values (rx, ry, rz), and up to 6 values for additional axes.

When using EGM joint mode with a 7-axis robot, then the first additional axis input provides the position for the additional robot axis.

Application manual - Externally Guided Motion
3HAC073319-001 Revision: H

The data flow for the UdpUc interface is illustrated below:
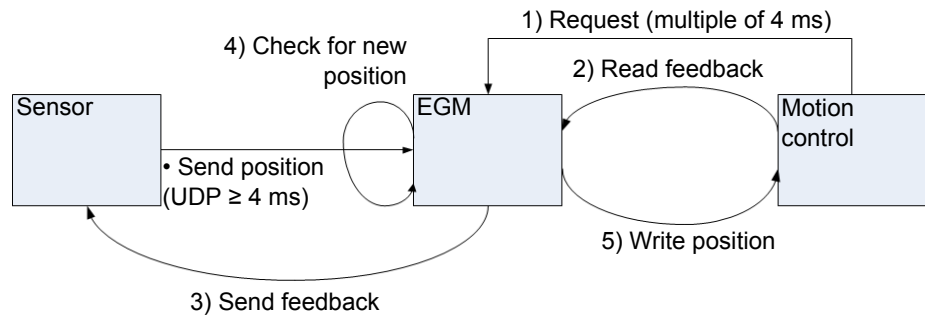
xx1400002017

1 Motion control calls EGM.

2 EGM reads feedback data from motion control.

3 EGM sends feedback data to the sensor.

4 EGM checks the UDP queue for messages from the sensor.

5 If there is a message, EGM reads the next message and step 5 writes the position data to motion control. If no position data had been sent, motion control continues to use the latest position data previously written by EGM.

• The sensor sends position data to the controller (EGM). Our recommendation is to couple this to step 3. Then the sensor will be in phase with the controller.

The control loop is based on the following relation between speed and position:

	k - factor (Default proportional Position Gain)
$speed = k * (pos\_ref – pos) + speed\_ref$	pos_ref - reference position
	pos - desired position
	$speed\_ref$ - reference speed

For instructions on how to implement the UdpUc protocol for an external device, see . There you will also find a description of input data.

Proportional controller gain k

The proportional controller gain k is used to calculate a velocity correction, used to drive the robot position towards the reference position *pos_ref*.

The input *speed_ref* is intended to be used as feedforward of a desired velocity, for instance when tracking a moving target or time-varying reference position. In such cases, input *speed_ref* should correspond to the desired current velocity of the moving target or time-varying reference position.

The resulting value speed is the resulting velocity, used internally by the robot controller to calculate joint references sent to the low-level servo control.

Input data for EGM Path Correction

The source for input data is selected using the EGM setup instructions. The three first instructions select a signal interface and the last instruction a UdpUc interface (*User Datagram Protocol Unicast Communication*).

Instructions	Description
EGMSetupAI	Setup analog input signals for EGM
EGMSetupAO	Setup analog output signals for EGM
EGMSetupGI	Setup group input signals for EGM
EGMSetupUC	Setup the UdpUc protocol for EGM

Input data for EGM contain mainly position data.

The data flow for the signal interface is illustrated below:
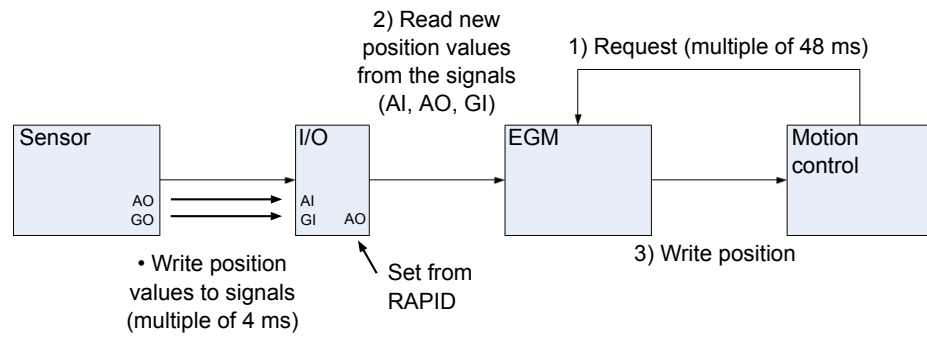
xx1400002016

1   Motion control calls EGM.

2   The measurement data (y- and z-values) are read from the signals or fetched from the sensor at multiples of about 48 ms.

3   EGM calculates the position correction and writes it to motion control. If the UdpUc protocol is used, feedback is sent to the sensor.

2.4 Output data

Description

Output data is only available for the UdpUc interface.

For instructions on how to implement the UdpUc protocol for an external device, see *The EGM sensor protocol on page 27*. There you will also find a description of output data.

2.5 Configuration

Configuration for EGM Position Stream

The default configuration, which is predefined in RobotWare, works for any position streaming. It will also work with the same configuration as for EGM Position Guidance.

Configuration for EGM Position Guidance

EGM behavior can be influenced using the system parameters of type *External Motion Interface Data* topic *Motion*. For a description of all available EGM parameters, see .

Here follows a closer description of the two parameters that influence the EGM control loop. The figure shows a simplified view of the EGM control system.
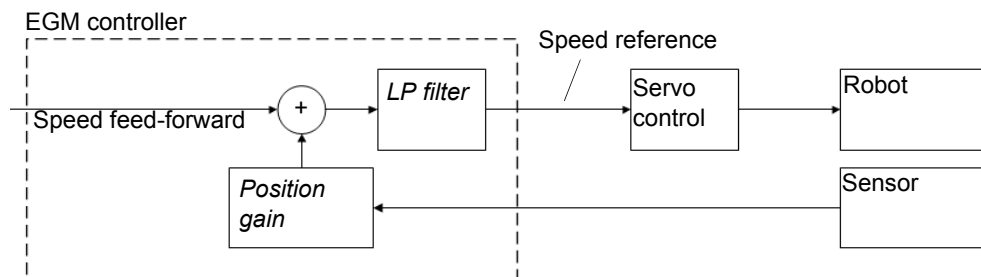
xx1400001083

Default proportional Position Gain	The parameter *Position gain* in the figure influences the responsiveness moving to the target position, given by the sensor, in relation to the current robot position. The higher the value, the faster the response.
Default Low Pass Filter Bandwith Time	The parameter *LP Filter* in the figure is the default value used to filter the speed contribution from EGM.

Configuration for EGM Path Correction

The configuration for EGM Path Correction has to have *Level* set to *Path*. None of the other values are used.

2.6 Frames

Frames for EGM Position Stream

Position streaming does not support dynamic change of tool or load. If `EGMStreamStart` is used to start the position stream, the active tool and load are passed to the controller. If `EGMActJoint` or `EGMActPose` is used, the active tool and load (or if specified, the specified tool and/or load) are passed to the controller. Those tool and load data are then used by EGM for calculation of positions until the position stream is stopped.

Frames for EGM Position Guidance

EGM can be run in two different modes, joint mode and pose mode. The following section applies to the EGM pose mode only.

For the joint mode there is no need for reference frames, because both sensor values and position values are axis angles given in degrees relative to the calibration position of each axis. But for the pose mode reference frames are necessary. Measurements from the sensor and directions for position change can only be given relative to reference frames.

The RAPID instruction `EGMActPose` defines all frames that are available in EGM:

Frame	Description
Tool	The tool data to be used for the EGM process is defined with the optional `\Tool` argument.
Work object	The work object data used for the EGM process is defined with the optional `\Wobj` argument.
Correction	The frame to be used to give the final movement direction is defined by the mandatory `CorrFrame` argument.
Sensor	The frame to be used to interpret the sensor data is defined by the mandatory `SensorFrame` argument.

Tools and work objects

The tool and the work object may be defined in two combinations only:

1. If the tool is attached to the robot, the work object has to be fixed.

2. If the tool is fixed, the work object has to be attached to the robot.

> **Note**
>
> It is not possible to use a work object or tool that is attached to any other mechanical unit than the EGM robot.

Predefined frame types

For the frames `CorrFrame` and `SensorFrame` it is also necessary to know what they are related to. This information is specified using the predefined frame types in the data type egmframetype:

Value	Description
EGM_FRAME_BASE	The frame is defined relative to the base frame (pose mode).
EGM_FRAME_TOOL	The frame is defined relative to `tool0` (pose mode).

Continues on next page

Value	Description
EGM_FRAME_WOBJ	The frame is defined relative to the active work object (pose mode).
EGM_FRAME_WORLD	The frame is defined relative to the world frame (pose mode).
EGM_FRAME_JOINT	The values are joint values (joint mode).

Frames for EGM Path Correction

EGM Path Correction can only be run in pose mode.

The RAPID instruction `EGMActMove` defines the only frame that is needed for EGM Path Correction. Tool and Work object are specified in `EGMMoveL` or `EGMMoveC`.

Tools and work objects

The tool has to be attached to the robot and the work object may be fixed or moved by another mechanical unit.

3 The EGM sensor protocol

3.1 Overview

3.1.1 EGM sensor protocol overview

Communication and transport protocol

The EGM sensor protocol is designed for high speed communication between a robot controller and a communication endpoint with minimum overhead.

The communication endpoint is typically a sensor, so *sensor* will be used from now on instead of communication endpoint. Sometimes the sensor is connected to a PC, and the PC then transfers the sensor data to the robot. The purpose of the sensor protocol is to communicate sensor data frequently between the robot controller and sensors. The EGM sensor protocol is using Google Protocol Buffers for encoding and UDP as a transport protocol. Google Protocol Buffers has been selected due to its speed and its language-neutrality. UDP has been chosen as a transport protocol since the data sent is *real-time* data sent with high frequency and if packets get lost it is useless to re-send the data.

Data structures and system parameters

The EGM sensor protocol data structures are defined by the EGM proto file. Sensor name, IP-address and port number of sensors are configured in the system parameters. A maximum of eight sensors can be configured.

Messages and queue handling

The sensor is acting as a server and it cannot send anything to the robot before it has received a first message from the robot controller. Messages can be sent independently of each other in both directions after that first message. Applications using the protocol may put restrictions on its usage but the protocol itself has no built-in synchronization of request responses or supervision of lost messages. There are no special connect or disconnect messages, only data which can flow in both directions independently of each other. The first message from the robot is a data message. One has also to keep in mind, that a sender of an UDP message continues to send even though the receiver's queue may be full. The receiver has to make sure, that its queue is emptied.

By default, the robot will send and read data from the sensor every 4 milliseconds, independently of when data is sent from the sensor. This cycle time can be changed to a multiple of 4 ms using the optional argument `\SampleRate` of the RAPID instructions `EGMStreamStart`, `EGMActJoint` or `EGMActPose`.

Each motion task requires its own UDP channel.

3.1.2 Google Protocol Buffers

Protobuf overview

Google Protocol Buffers or *Protobuf*, are a way to serialize/de-serialize data in a very efficient way. Protobuf is in general 10-100 times faster than XML. There is plenty of information on the Internet about Protobuf and the *Google overview* is a good start.

In short, message structures are described in a *.proto* file. The *.proto* file is then compiled. The compiler generates serialized/de-serialized code which is then used by the application. The application reads a message from the network, runs the de-serialization, creates a message, calls serialization method, and then sends the message.

It is possible to use Protobuf in most programming languages since Protobuf is language neutral. There are many different implementations depending on the language.

The main disadvantage with Protobuf is that Protobuf messages are serialized into a binary format which makes it more difficult to debug packages using a network analyzer.

Third party tools

Except for the *Google C++* tool, we have also verified the following third party tools and code:

- *Nanopb*, generates C-code and it does not require any dynamic memory allocations.
- *Protobuf-net*, a Google Protobuf .NET library.
- *Protobuf-csharp*, a Google Protobuf .NET library, the C# API is similar to the Google C++ API.

> **ℹ Note**
>
> Note that the code mentioned above is open source, which means that you have to check the license that the code is allowed to be used in your product.

3.1.3 EGM sensor protocol description

Data structure

The EGM sensor protocol is not a request/response protocol, the sensor can send data at any frequency after the sensor gets the first message from the robot.

The EGM sensor protocol has two main data structures, *EgmRobot* and *EgmSensor*. *EgmRobot* is sent from the robot and *EgmSensor* is sent from the sensor. All message fields in both data structures are defined as optional which means that a field may or may not be present in a message. Applications using *Google Protocol Buffers* must check if optional fields are present or not.

The Google protobuf data structure can include the *repeated* element, that is, a list of elements of the same type. The *repeated* element count is a maximum of six elements in the EGM sensor protocol, except for `EGMRAPIDdata` which has a maxiumum of 40.

See the *egm.proto* file for a description of `EgmRobot` and `EgmSensor`, *UdpUc code examples on page 113*.

Sensor protocol messages

This section describes some of the sensor protocol messages. Note that new optional fields that are not described in this manual can exist.

EgmHeader

The `EgmHeader` is common for both *EgmRobot* and *EgmSensor*.

```
message EgmHeader
{
  optional uint32 seqno = 1; // sequence number (to be able to find
      lost messages)
  optional uint32 tm = 2; // time stamp in milliseconds

  enum MessageType {
    MSGTYPE_UNDEFINED = 0;
    MSGTYPE_COMMAND = 1; // for future use
    MSGTYPE_DATA = 2; // sent by robot controller
    MSGTYPE_CORRECTION = 3; // sent by sensor for position guidance
    MSGTYPE_PATH_CORRECTION = 4; // sent by sensor for path
        correction
  }

  optional MessageType mtype = 3 [default = MSGTYPE_UNDEFINED];
}
```

Variable	Description
`seqno`	Sequence number. Applications shall increase the sequence number by one for each message they send. It makes it possible to check for lost messages in a series of messages.
`tm`	Timestamp in milliseconds. (Can be used for monitoring of delays).

Continues on next page

3 The EGM sensor protocol

3.1.3 EGM sensor protocol description
Continued

Variable	Description
mtype	Message type. **Shall be set to** MSGTYPE_CORRECTION **by the sensor, and is set to** MSGTYPE_DATA **by the robot controller.**

EgmRobot

```
message EgmRobot
{
    optional EgmHeader              header = 1;
    optional EgmFeedBack            feedBack = 2;
    optional EgmPlanned             planned = 3;

    optional EgmMotorState          motorState = 4;
    optional EgmMCIState            mciState = 5;
    optional bool                   mciConvergenceMet = 6;
    optional EgmTestSignals         testSignals = 7;
    optional EgmRapidCtrlExecState  rapidExecState = 8;
    optional EgmMeasuredForce       measuredForce = 9;
    optional double                 utilizationRate=10;
    optional uint32                 moveIndex=11;
    optional EgmCollisionInfo       CollisionInfo = 12;
    optional EgmRAPIDdata           RAPIDfromRobot = 13;
}
```

Variable	Description
header	**Reference to** EgmHeader.
feedback	**Feedback position, that is, measured position for robot and additional axes. See the description of** EgmFeedBack **for more details.**
planned	**Reference position for robot and additional axes. See the description of** EgmPlanned **for more details.**
motorState	**Motors On state (On, Off).**
mciState[i]	**Internal EGM state (Running, Stopped, Error).**
mciConvergenceMet	**Boolean that shows if the convergence criteria defined in the EGM instruction is fulfilled**
testSignals	**For internal use only.**
rapidExecState	**RAPID program execution state (Running, Stopped).**
measuredForce	**Measured contact force. Will show raw sensor force when force control is not active.**
utilizationRate	**This value shows a percentage value of how much of the available robot performance is needed to perform the movement according to the latest sent** EgmSensor **package. If this value is over 100% then the controller will not be able to perform the full movement and EGM will modify the reference so that the robot will only perform a partial movement in the desired direction.** **If** EgmSensor **packages are missing or delayed the last sent** EgmPackage **will be used and this can also cause the** utilizationRate **temporarily to be very high.**
moveIndex	**Simple index counter that will indicate start of movement and change of target.**

Continues on next page

Variable	Description
CollisionInfo	Shows an indication of collision for an axis.
RAPIDfromRobot	I/O signal and array of `dnum` that are sent from the robot to the sensor. This data can be used in various ways and adds flexibility to the use of EGM.

i   The acronym MCI refers to the Motion Correction Interface which is the internal controller interface that EGM uses.

EgmSensor

```
message EgmSensor
{
  optional EgmHeader          header = 1;
  optional EgmPlanned         planned = 2;
  optional EgmSpeedRef        speedRef = 3;
  optional EgmRAPIDdata       RAPIDtoRobot = 4;
}
```

Variable	Description
header	Reference to `EgmHeader`.
planned	Reference position for robot and additional axes.
speedRef	Reference speed for robot and additional axes. The unit is mm/s or degrees/s depending on the type of joint.
RAPIDfromSensor	I/O signal and array of `dnum` that are sent from the sensor to the robot. This data can be used in various ways and adds flexibility to the use of EGM.

EgmPlanned

```
message EgmPlanned
{
optional EgmJoints            joints = 1;
optional EgmPose              cartesian = 2;
optional EgmJoints external   Joints = 3;
optional EgmClock             time = 4;
}
```

Variable	Description
joints	Planned joint position for Robot.
cartesian	Planned Cartesian position for Robot.
externalJoints	Planned position for external axes (6 values).
time	Timestamp for when the Robot and external axes will be in the planned position.

EgmFeedBack

```
message EgmFeedBack
{
optional EgmJoints            joints = 1;
optional EgmPose              cartesian = 2;
optional EgmJoints external   Joints = 3;
optional EgmClock             time = 4;
}
```

3.1.3 EGM sensor protocol description
Continued

Variable	Description
`joints`	**Measured joint position for Robot.**
`cartesian`	**Measured Cartesian position for Robot.**
`externalJoints`	**Measured position for external axes (6 values).**
`time`	**Timestamp for when the Robot and external axes was in the measured position.**

3.2 Building an EGM sensor communication endpoint

How to build an EGM sensor communication endpoint using .Net

This guide assumes that you build and compile using Visual Studio and are familiar with its operation.

Here is a short description on how to install and create a simple test application using *protobuf-csharp-port*.

	Action
1	In Visual Studio, create a C# application.
2	Select **Tools** and then **NuGet Package Manager**, and install *Google.Protobuf*.
3	In the **NuGet Package Manager**, also install *Google.Protobuf.Tools*.
4	Navigate to *Solution package\packages\Google.Protobuf.Tools.3.xx.x\tools\windows_x64*.
5	Copy the EGM folder from *C:\ProgramData\ABB\DistributionPackages\ABB.RobotWare-6.yy\RobotPackages\RobotWare_RPK_6.zz\utility\Template\" to "\packages\Google.Protobuf.Tools.3.xx.x\tools\windows_x64*.
6	Open a cmd line and run " *.\protoc .\egm\egm.proto --csharp_out=.\egm* ". => Egm.cs file is built ℹ️ **Note** The *egm.proto* syntax is `proto2`.
7	Add the generated file *egm.cs* to the Visual Studio project (add existing item).
8	Copy the example code into the Visual Studio Windows Console application file *(egm-sensor.cs)* and then compile, link and run.

3.3 Configuring UdpUc devices

How to configure UdpUc devices

UdpUc communicates with a maximum of eight devices over Udp. The devices act as servers, and the robot controller acts as a client. It is the robot controller that initiates the connection to the sensor.

Each UDP channel is defined as a device, i.e. you need to set up one device for each motion task where you want to use EGM.

System parameters

This is a brief description of the parameters used when configuring a device. For more information about the parameters, see *Technical reference manual - System parameters*.

These parameters belong to the type *Transmission Protocol* in topic *Communication*.

Parameter	Description
Name	The name of the transmission protocol. For example *EGMsensor*.
Type	The type of transmission protocol. It has to be *UDPUC*.
Serial Port	The name of the serial port that will be used for the sensor. This refers to the parameter *Name* in the type *Serial Port*. For IP based transmission protocols (i.e. *Type* has value TCP/IP, SOCKDEV, LTAPPTCP or UDPUC), *Serial Port* is not used and has the value N/A.
Remote Address	The IP address of the remote device.
Remote Port Number	The IP port number that the remote device has opened.

Configuration example

The device which provides the input data for EGM, has to be configured as an UdpUc device in the following way:

Name	Type	Serial Port	Remote Address	Remote Port Number
UCdevice	UDPUC	N/A	192.168.10.20	6510
UCdevice2	UDPUC	N/A	192.168.10.20	6511
UCdevice3	UDPUC	N/A	192.168.10.21	6510

If EGM Position Stream or EGM Position Guidance is used in a MultiMove system for more than one RAPID task, a device has to be configured for each RAPID task. The device names and either the remote address or the remote port or both have to be different.

After this configuration change, the controller has to be restarted. Now the device can be used by EGM to guide a robot. For more information, see *Using EGM Position Guidance with an UdpUc device on page 102*.

4 System parameters

4.1 Type *External Motion Interface Data*

4.1.1 The External Motion Interface Data type

Overview

This section describes the type *External Motion Interface Data*, which belongs to the topic *Motion*. Each parameter of the type is described in a separate information topic in this section.

Type description

The *External Motion Interface Data* type contains a number of parameters that defines the characteristics for an *External Motion Interface Data*.

4.1.2 Name

Parent

Name belongs to the type *External Motion Interface Data*, in the topic *Motion*.

Description

The name of the *External Motion Interface Data*.

Usage

This is the public identity of the *External Motion Interface Data*.

The parameter does not require a restart of the controller when modified.

Allowed values

A string with maximum 32 characters.

4.1.3 Level

Parent

Level belongs to the type *External Motion Interface Data*, in the topic *Motion*.

Description

External Motion Interface Level determines the system level at which the corrections are applied.

Usage

Level can have the following values:

Value:	Name	Description:
0	Raw	Corresponds to raw corrections, added just before the servo controllers
1	Filtering	Applies extra filtering on the correction, but also introduces some extra delays and latency
2	Path	Applies path corrections.

The parameter does not require a restart of the controller when modified.

Limitation

When using *Level 0*, low-pass filtering is necessary to avoid vibrations in the robot.

Allowed values

Allowed values are level *0*, *1* or *2*

The default value is 1.

4.1.4 Do Not Restart after Motors Off

Parent

Do Not Restart after Motors Off belongs to the type *External Motion Interface Data*, in the topic *Motion*.

Description

Do Not Restart after Motors Off determines if the *External Motion Interface* execution should automatically restart after the controller has been in motors off state, for example after an emergency stop.

Usage

If *False* (default), execution of the corrections will continue in the same state as when the system entered the Motors Off state.

If *True*, execution will continue with all corrections in the STANDBY state.

Allowed values

True or False.

4.1.5 Return to Program Position when Stopped

Parent

Return to Program Position when Stopped belongs to the type *External Motion Interface Data*, in the topic *Motion*.

Description

Return to Program Position when Stopped determines if axes currently running *External Motion Interface* should return to the programmed position, when program execution is stopped.

Usage

If *False* (default), axes will stop in their current position.

If *True*, axes will move to the programmed start position.

Limitation

The motion returning the axes to the programmed position will be defined in joint space. If the axes are far from the programmed position when *Return to Programmed Position when Stopped* is defined as *False*, unexpected trajectories may result. Therefore, it is recommended only to set this value to *False*, if the distance from the programmed position to the corrected position is known to be small.

Allowed values

True or False.

4.1.6 Default Ramp Time

Parent

Default Ramp Time belongs to the type *External Motion Interface Data*, in the topic *Motion*.

Description

Default Ramp Time defines the default total time for stopping *External Motion Interface* movements when *External Motion Interface* execution is stopped.

Usage

The value will be used to determine how fast the speed contribution from *External Motion Interface* should be ramped to zero when program execution is stopped, and how fast axes return to the programmed position if the *Return to Programmed Position when Stopped* is *True*.

This value can normally be lower than 1. The value should be tuned and checked during application. A big robot with heavy payload that runs at high speed will need a higher value, while a small robot with small payload that runs at low speed can have a low value to stop quickly.

> **ℹ Note**
>
> Since movement during ramping will be a joint movement, the robot will deviate from its current position and guidance position during the stop.

Limitation

The value only affects the part of the motion that is generated by the *External Motion Interface* execution. It does not affect any simultaneous movements that have, for instance, been programmed in RAPID.

Allowed values

A value between 0.005 and 10.0 seconds.

The default value is 0.5 seconds.

4.1.7 Default Proportional Position Gain

Parent

Default Proportional Position Gain belongs to the type *External Motion Interface Data*, in the topic *Motion*.

Description

Default Proportional Position Gain defines the default proportional gain of the *External Motion Interface* position feedback control.

Allowed values

A value between 0.0 and 20.0.

The default value is 5.0.

4.1.8 Default Low Pass Filter Bandwidth

Parent

Default Low Pass Filter Bandwidth belongs to the type *External Motion Interface Data*, in the topic *Motion*.

Description

Default Low Pass Filter Bandwidth Time defines the default bandwidth of the low-pass filter used to filter the speed contribution from the *External Motion Interface* execution.

Allowed values

A value between 0.0 and 100.0 Hz.

The default value is 20.0 Hz.

> ℹ **Note**
>
> Too low or too high values can result in unpredictable movement.

5 RAPID reference information

5.1 Instructions

5.1.1 EGMActJoint - Prepare an EGM movement for a joint target

Usage

EGMActJoint activates a specific EGM process and defines static data for the sensor guided movement to a joint target, that is, data that is not changed frequently between different EGM movements.

Basic examples

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0], [1,0,0,0]];
CONST egm_minmax egm_minmax1:=[-1,1];
VAR signaldo RobotDO 1;
PERS dnum RobotOut{10}:= [3.0, 6.0, 9.0, 12.0, 15.0, 18.0, 21.0,
    24.0, 27.0, 30];
VAR signaldi RobotDI 0;
PERS dnum RobotIn{10}:= [33.0, 0, 0, 0, 0, 0, 0, 0, 0, 0];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
    \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
    \aiR6rz:=ai_06;
EGMActJoint egmID1 \DOtoSensor:=RobotDO \DataToSensor:=RobotOut
    \DIfromSensor:=RobotDI \DataFromSensor:=RobotIn
    \J1:=egm_minmax1 \J3:=egm_minmax1 \J4:=egm_minmax1;
```

Arguments

```
EGMActJoint EGMid [\StreamStart] [\Tool] [\WObj] [\TLoad]
    [\DOtoSensor] [\DataToSensor] [\DIfromSensor]
    [\DataFromSensor] [\J1] [\J2] [\J3] [\J4] [\J5] [\J6] [\J7]
    [\LpFilter] [\SampleRate] [\MaxPosDeviation]
    [\MaxSpeedDeviation]
```

EGMid

Data type: egmident

EGM identity.

[\StreamStart]

Data type: switch

StreamStart **starts streaming position data to external equipment. The data is sent at the cyclicity defined with** \SampleRate **and the content is according to the protocol specification in the Google Protobuf definition file** *egm.proto*.

StreamStart **is only available if EGM is set up using** EGMSetupUC, **i.e. the protocol UdpUc is used for communication with the external equipment.**

Continues on next page

5 RAPID reference information

`[\Tool]`

> **Data type:** `tooldata`
>
> The tool in use for movements performed with the instruction `EGMRunJoint`.
>
> The argument `[\Tool]` is optional. The default value when the argument is omitted is `tool0`.

`[\Wobj]`

> The work object in use for movements performed with the instruction `EGMRunJoint`.
>
> *Work Object*
>
> **Data type:** `wobjdata`
>
> The work object (object coordinate system) to which the robot position in the instruction is related.
>
> This argument can be omitted and if it is then the position is related to the world coordinate system. If, on the other hand, a stationary TCP or coordinated external axes are used this argument must be specified in order for a circle relative to the work object to be executed.

`[\TLoad]`

> The load in use for movements performed with the instruction `EGMRunJoint`.
>
> *Total load*
>
> **Data type:** `loaddata`
>
> The `\TLoad` argument describes the total load used in the movement. The total load is the tool load together with the payload that the tool is carrying. If the `\TLoad` argument is used, then the `loaddata` in the current `tooldata` is not considered.
>
> If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead.
>
> To be able to use the `\TLoad` argument it is necessary to set the value of the system parameter `ModalPayLoadMode` to 0. If `ModalPayLoadMode` is set to 0, it is no longer possible to use the instruction `GripLoad`.
>
> The total load can be identified with the service routine LoadIdentify. If the system parameter `ModalPayLoadMode` is set to 0, the operator has the possibility to copy the `loaddata` from the tool to an existing or new `loaddata` persistent variable when running the service routine.
>
> It is possible to test run the program without any payload by using a digital input signal connected to the system input `SimMode` (Simulated Mode). If the digital input signal is set to 1, the `loaddata` in the optional argument `\TLoad` is not considered, and the `loaddata` in the current `tooldata` is used instead.

> ℹ️ **Note**
>
> The default functionality to handle payload is to use the instruction `GripLoad`. Therefore the default value of the system parameter `ModalPayLoadMode` is 1.

`[\DOToSensor]`

> **Data type:** `signaldo`

Continues on next page

Application manual - Externally Guided Motion
3HAC073319-001 Revision: H

I/O signal sent from the robot controller to an external interface such as a sensor. The data is sent at the cyclicity defined with `\SampleRate` and the content is according to the protocol specification in the Google Protobuf definition file egm.proto. The option is only available if EGM is set up using *EGMSetupUC*, that is, the protocol UdpUc is used for communication with the external equipment.

[`\DataToSensor`]

Data type: `dnum`

An array of up to 40 `dnum` values sent from the robot controller to an external interface such as a sensor. The data is sent at the cyclicity defined with `\SampleRate` and the content is according to the protocol specification in the Google Protobuf definition file egm.proto. The option is only available if EGM is set up using *EGMSetupUC*, that is, the protocol UdpUc is used for communication with the external equipment.

[`\DIFromSensor`]

Data type: `signaldi`

I/O signal sent from an external interface such as a sensor to the robot controller. The data is sent at the cyclicity defined with `\SampleRate` and the content is according to the protocol specification in the Google Protobuf definition file egm.proto. The option is only available if EGM is set up using *EGMSetupUC*, that is, the protocol UdpUc is used for communication with the external equipment.

[`\DataFromSensor`]

Data type: `dnum`

An array of up to 40 `dnum` values sent from an external interface such as a sensor to the robot controller. The data is sent at the cyclicity defined with `\SampleRate` and the content is according to the protocol specification in the Google Protobuf definition file egm.proto. The option is only available if EGM is set up using *EGMSetupUC*, that is, the protocol UdpUc is used for communication with the external equipment.

[`\J1`] [`\J2`] [`\J3`] [`\J4`] [`\J5`] [`\J6`] [`\J7`]

Data type: `egm_minmax`

Convergence criteria for joint 1 to 6 in degrees for 6-axis robots, and joint 1 to 7 in degrees for 7-axis robots. The default value is ±0.5 degrees.

The convergence criteria data is used to decide if the robot has reached the ordered joint positions. If the difference between the ordered joint position and the actual joint position is within the range of `egm_minmax.min` and `egm_minmax.max`, the joint is regarded to have reached its ordered position. If no convergence criteria is specified for a joint, that was selected in `EGMRunJoint`, the default value is used.

As soon as all joints that were specified in `EGMRunJoint` have reached their ordered positions, the robot itself has reached its ordered position and RAPID execution continues with the next RAPID instruction.

[`\LpFilter`]

Data type: `num`

5 RAPID reference information

Low pass filter bandwidth, in Hertz (Hz), used to filter sensor noise.

`[\SampleRate]`

Data type: `num`

Input data reading sample rate in multiples of 4 milliseconds. Valid values are 4, 8, 12, 16, etc.

The default value is 4 milliseconds.

`[\MaxPosDeviation]`

Data type: `num`

Maximum joint deviation from the programmed position in degrees, i.e. the fine point the EGM movement started at. The same value is used for all joints.

The default value is 1000 degrees.

`[\MaxSpeedDeviation]`

Data type: `num`

Maximum admitted joint speed change in degrees/second. All joints will be reduced with the same ratio if one joint is limited.

The default value is 1.0 degrees/second.

Limitations

- If `EGMActJoint` is executed several times with the same `EGMid`, the latest activation data is used for `EGMRunJoint` instructions that follow until a new `EGMActJoint` is run.

- `EGMActJoint` can only be used in RAPID motion tasks.

Syntax

```
EGMActJoint
  [EGMid ':='] <variable (VAR) of egmident>
  ['\'Tool ':=' <persistent (PERS) of tooldata>]
  ['\'Wobj ':=' <persistent (PERS) of wobjdata>
  ['\'TLoad ':=' <persistent (PERS) of loaddata>]
  ['\'DOToSensor ':=' <variable> (VAR) of signaldo]
  ['\'DataToSensor ':=' <persistent> (PERS) of dnum]
  ['\'DIFromSensor ':=' <variable> (VAR) of signaldi]
  ['\'DataFromSensor ':=' <persistent> (PERS) of dnum]
  ['\'J1 ':=' <expression (IN) of egm_minmax>]
  ['\'J2 ':=' <expression (IN) of egm_minmax>]
  ['\'J3 ':=' <expression (IN) of egm_minmax>]
  ['\'J4 ':=' <expression (IN) of egm_minmax>]
  ['\'J5 ':=' <expression (IN) of egm_minmax>]
  ['\'J6 ':=' <expression (IN) of egm_minmax>]
  ['\'J7 ':=' <expression (IN) of egm_minmax>]
  ['\'LpFilter ':=' <expression (IN) of num>]
  ['\'SampleRate ':=' <expression (IN) of num>]
  ['\'MaxPosDeviation ':=' <expression (IN) of num>]
  ['\'MaxSpeedDeviation ':=' <expression (IN) of num>] ';'
```

Related information

For information about	See
Instruction `EGMRunJoint`	*EGMRunJoint - Perform an EGM movement with a joint target on page 65*
Instruction `EGMStreamStart`	*EGMStreamStart - start EGM position streaming on page 87*
Data type `egm_minmax`	*egm_minmax - Convergence criteria for EGM on page 95*
`MoveJ`	*Technical reference manual - RAPID Instructions, Functions and Data types*

5.1.2 EGMActMove - Prepare an EGM movement with path correction

Usage

`EGMActMove` is used to activate a specific EGM process and defines static data for the movement with path correction, i.e. data that is not changed frequently between different EGM path correction movements.

Basic examples

The following example illustrates the instruction `EGMActMove`.

Example 1

```
VAR egmident EGMid1;
PERS tooldata tLaser := [TRUE, [[148,50,326],
    [0.3902618,-0.589657,-0.589656,0.3902630]],
    [1,[-0.92,0,-0.39], [1,0,0,0], 0,0,0]];
EGMGetId EGMid1;
EGMSetupLTAPP ROB_1, EGMid1, "pathCorr", "OptSim", 1\LATR;
EGMActMove EGMid1, tLaser.tframe\SampleRate:=48;
```

This program registers an EGM process, and sets up a sensor that uses the communication protocol LTAPP and is of the type *look-ahead* as data source (sensor). The sensor shall use the joint type definition number 1 for the tracking. The rate at which the controller will access the device and the sensor frame of the device are also setup.

Arguments

```
EGMActMove EGMid, SensorFrame [\SampleRate]
```

EGMid

Data type: `egmident`

EGM identity.

SensorFrame

Data type: `pose`

Sensor frame.

[\SampleRate]

Data type: `num`

Input data reading sample rate in multiples of 24 ms. Valid values: 24, 48, 72, etcetera.

Program execution

The sensor frame and the sensor sampling rate are connected to an EGM identity until they are either reset with `EGMReset` or changed by another `EGMActMove` instruction.

Syntax

```
EGMActMove
  [EGMid ':='] <variable (VAR) of egmident> ','
  [SensorFrame ':='] < expression (IN) of pose>
```

Continues on next page

```
['\'SampleRate ':=' <expression (IN) of num>] ';'
```

Related information

For information about	See
EGMReset	*EGMReset - Reset an EGM process on page 64*

5.1.3 EGMActPose - Prepare an EGM movement for a pose target

Usage

`EGMActPose` activates a specific EGM process and defines static data for the sensor guided movement to a pose target, that is, data that is not changed frequently between different EGM movements.

Basic examples

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0], [1,0,0,0]];
CONST egm_minmax egm_minmax_lin:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot:=[-0.1,0.2];
CONST pose posecor:=[[1200,400,900], [0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
        [0.903899,-0.00320735,0.427666,0.00765917]];
VAR signaldo RobotDO 1;
PERS dnum RobotOut{10}:= [3.0, 6.0, 9.0, 12.0, 15.0, 18.0, 21.0,
        24.0, 27.0, 30];
VAR signaldi RobotDI 0;
PERS dnum RobotIn{10}:= [33.0, 0, 0, 0, 0, 0, 0, 0, 0, 0];


EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
        \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
        \aiR6rz:=ai_06;
EGMActPose egmID1 \Tool:=tool0 \WObj:=wobj0, \DOtoSensor:=RobotDO
        \DataToSensor:=RobotOut \DIfromSensor:=RobotDI
        \DataFromSensor:=RobotInposecor, EGM_FRAME_WOBJ, posesens,
        EGM_FRAME_TOOL \x:=egm_minmax_lin \y:=egm_minmax_lin
        \z:=egm_minmax_lin \rx:=egm_minmax_rot \ry:=egm_minmax_rot
        \rz:=egm_minmax_rot \LpFilter:=20;
```

Arguments

```
EGMActPose EGMid [\StreamStart] [\Tool] [\WObj] [\TLoad]
        [\DOtoSensor] [\DataToSensor] [\DIfromSensor]
        [\DataFromSensor] CorrFrame CorrFrType SensorFrame
        SensorFrType [\x] [\y] [\z] [\rx] [\ry] [\rz] [\LpFilter]
        [\SampleRate] [\MaxPosDeviation] [\MaxSpeedDeviation]
```

`EGMid`

Data type: `egmident`

EGM identity.

`[\StreamStart]`

Data type: `switch`

`StreamStart` **starts streaming position data to external equipment. The data is sent at the cyclicity defined with** `\SampleRate` **and the content is according to the protocol specification in the Google Protobuf definition file** *egm.proto*.

`StreamStart` **is only available if EGM is set up using** `EGMSetupUC`, **i.e. the protocol UdpUc is used for communication with the external equipment.**

Continues on next page

Application manual - Externally Guided Motion
3HAC073319-001 Revision: H

[\Tool]

Data type: tooldata

The tool in use for movements performed with the instruction EGMRunPose.

The argument [\Tool] is optional. The default value when the argument is omitted is tool0.

[\Wobj]

The work object in use for movements performed with the instruction EGMRunPose.

Work Object

Data type: wobjdata

The work object (object coordinate system) to which the robot position in the instruction is related.

This argument can be omitted and if it is then the position is related to the world coordinate system. If, on the other hand, a stationary TCP or coordinated external axes are used this argument must be specified in order for a circle relative to the work object to be executed.

[\TLoad]

The load in use for movements performed with the instruction EGMRunPose.

Total load

Data type: loaddata

The \TLoad argument describes the total load used in the movement. The total load is the tool load together with the payload that the tool is carrying. If the \TLoad argument is used, then the loaddata in the current tooldata is not considered.

If the \TLoad argument is set to load0, then the \TLoad argument is not considered and the loaddata in the current tooldata is used instead.

To be able to use the \TLoad argument it is necessary to set the value of the system parameter ModalPayLoadMode to 0. If ModalPayLoadMode is set to 0, it is no longer possible to use the instruction GripLoad.

The total load can be identified with the service routine LoadIdentify. If the system parameter ModalPayLoadMode is set to 0, the operator has the possibility to copy the loaddata from the tool to an existing or new loaddata persistent variable when running the service routine.

It is possible to test run the program without any payload by using a digital input signal connected to the system input SimMode (Simulated Mode). If the digital input signal is set to 1, the loaddata in the optional argument \TLoad is not considered, and the loaddata in the current tooldata is used instead.

ℹ️ Note

The default functionality to handle payload is to use the instruction GripLoad. Therefore the default value of the system parameter ModalPayLoadMode is 1.

[\DOToSensor]

Data type: signaldo

Continues on next page

5.1.3 EGMActPose - Prepare an EGM movement for a pose target
Externally Guided Motion
Continued

I/O signal sent from the robot controller to an external interface such as a sensor. The data is sent at the cyclicity defined with `\SampleRate` and the content is according to the protocol specification in the Google Protobuf definition file egm.proto. The option is only available if EGM is set up using *EGMSetupUC*, that is, the protocol UdpUc is used for communication with the external equipment.

`[\DataToSensor]`

Data type: `dnum`

An array of up to 40 `dnum` values sent from the robot controller to an external interface such as a sensor. The data is sent at the cyclicity defined with `\SampleRate` and the content is according to the protocol specification in the Google Protobuf definition file egm.proto. The option is only available if EGM is set up using *EGMSetupUC*, that is, the protocol UdpUc is used for communication with the external equipment.

`[\DIFromSensor]`

Data type: `signaldi`

I/O signal sent from an external interface such as a sensor to the robot controller. The data is sent at the cyclicity defined with `\SampleRate` and the content is according to the protocol specification in the Google Protobuf definition file egm.proto. The option is only available if EGM is set up using *EGMSetupUC*, that is, the protocol UdpUc is used for communication with the external equipment.

`[\DataFromSensor]`

Data type: `dnum`

An array of up to 40 `dnum` values sent from an external interface such as a sensor to the robot controller. The data is sent at the cyclicity defined with `\SampleRate` and the content is according to the protocol specification in the Google Protobuf definition file egm.proto. The option is only available if EGM is set up using *EGMSetupUC*, that is, the protocol UdpUc is used for communication with the external equipment.

`CorrFrame`

Data type: `pose`
Correction frame.

`CorrFrType`

Data type: `egmframetype`
Frame type of the correction frame.

`SensorFrame`

Data type: `pose`
Sensor frame.

`SensFrType`

Data type: `egmframetype`
Frame type of the sensor frame.

Continues on next page

5.1.3 EGMActPose - Prepare an EGM movement for a pose target
Externally Guided Motion
Continued

`[\x] [\y] [\z]`

    Data type: `egm_minmax`

    Convergence criteria for x, y, and z in millimeters. The default value is ±1.0 millimeters.

    The convergence criteria data is used to decide if the robot has reached the ordered position in the specified axis direction. If the difference between the ordered position and the actual position is within the range of `egm_minmax.min` and `egm_minmax.max`, the robot is regarded to have reached its ordered position. If no convergence criteria is specified for an axis direction, that was selected in `EGMRunPose`, the default value is used.

    As soon as all axes that were specified in `EGMRunPose` have reached their ordered positions, the robot itself has reached its ordered position and RAPID execution continues with the next RAPID instruction.

`[\rx] [\ry] [\rz]`

    Data type: `egm_minmax`

    Convergence criteria for rotation x, y, and z in degrees. The default value is ±0.5 degrees.

    The convergence criteria data is used to decide if the robot has reached the ordered orientation along the specified axis. If the difference between the ordered orientation and the actual orientation is within the range of `egm_minmax.min` and `egm_minmax.max`, the robot is regarded to have reached its ordered orientation. If no convergence criteria is specified for an axis orientation, that was selected in `EGMRunPose`, the default value is used.

    As soon as all axes orientations that were specified in `EGMRunPose` have reached their ordered orientation, the robot itself has reached its ordered position and RAPID execution continues with the next RAPID instruction.

`[\LpFilter]`

    Data type: `num`

    Low pass filter bandwidth, in Hertz (Hz), used to filter sensor noise.

    The default value is taken from the configuration of the `EGMSetupXX` instruction.

`[\SampleRate]`

    Data type: `num`

    Input data reading sample rate in multiples of 4 milliseconds. Valid values are 4, 8, 12, 16, etc.

    The default value is 4 milliseconds.

`[\MaxPosDeviation]`

    Data type: `num`

    Maximum joint deviation from the programmed position in degrees, i.e. the fine point the EGM movement started at. The same value is used for all joints.

    The default value is 1000 degrees.

5.1.3 EGMActPose - Prepare an EGM movement for a pose target
Externally Guided Motion
Continued

`[\MaxSpeedDeviation]`

> **Data type:** `num`
>
> Maximum admitted joint speed change in degrees/second. All joints will be reduced with the same ratio if one joint is limited.
>
> The default value is 1.0 degrees/second.

Limitations

- If `EGMActPose` is executed several times with the same `EGMid`, the latest activation data is used for `EGMRunPose` instructions that follow until a new `EGMActPose` is run.
- `EGMActPose` can only be used in RAPID motion tasks.

Syntax

```
EGMActPose
  [EGMid ':='] <variable (VAR) of egmident>
  ['\'Tool ':=' <persistent (PERS) of tooldata>]
  ['\'Wobj ':=' <persistent (PERS) of wobjdata>]
  ['\'TLoad ':=' <persistent (PERS) of loaddata>] ','
  ['\'DOToSensor ':=' <variable> (VAR) of signaldo]
  ['\'DataToSensor ':=' <persistent> (PERS) of dnum]
  ['\'DIFromSensor ':=' <variable> (VAR) of signaldi]
  ['\'DataFromSensor ':=' <persistent> (PERS) of dnum]
  [CorrFrame ':='] < expression (IN) of pose> ','
  [CorrFrType ':='] < expression (IN) of egmframetype> ','
  [SensorFrame ':='] < expression (IN) of pose> ','
  [SensorFrType ':='] < expression (IN) of egmframetype>
  ['\'x ':=' <expression (IN) of egm_minmax>]
  ['\'y ':=' <expression (IN) of egm_minmax>]
  ['\'z ':=' <expression (IN) of egm_minmax>]
  ['\'rx ':=' <expression (IN) of egm_minmax>]
  ['\'ry ':=' <expression (IN) of egm_minmax>]
  ['\'rz ':=' <expression (IN) of egm_minmax>]
  ['\'LpFilter ':=' <expression (IN) of num>]
  ['\'SampleRate ':=' <expression (IN) of num>]
  ['\'MaxPosDeviation ':=' <expression (IN) of num>]
  ['\'MaxSpeedDeviation ':=' <expression (IN) of num>] ';'
```

Related information

For information about	See
Instruction `EGMRunPose`	*EGMRunPose - Perform an EGM movement with a pose target on page 68*
Instruction `EGMStreamStart`	*EGMStreamStart - start EGM position streaming on page 87*
Data type `egm_minmax`	*egm_minmax - Convergence criteria for EGM on page 95*

5.1.4 EGMGetId - Gets an EGM identity

Usage

EGMGetId is used to reserve an EGM identity (EGMid). That identity is then used in all other EGM RAPID instructions and functions to identify a certain EGM process connected to the RAPID motion task from which it is used.

An egmident is identified by its name, that is, a second or third call of EGMGetId with the same egmident will neither reserve a new EGM process nor change its content.

To release an egmident for use by other EGM processes, the RAPID instruction EGMReset has to be used.

It is possible to use maximum 4 different EGM identities at the same time.

Basic examples

```
VAR egmident egmID1;
EGMGetId egmID1;
```

Arguments

```
EGMGetId EGMid
```

EGMid

Data type: egmident

EGM identity.

Limitations

- EGMGetId can only be used in RAPID motion tasks.

Syntax

```
EGMGetId
  [EGMid ':='] <variable (VAR) of egmident> ';'
```

Related information

For information about	See
EGMReset	*EGMReset - Reset an EGM process on page 64*

5.1.5 EGMMoveC - Circular EGM movement with path correction
Externally Guided Motion

5.1.5 EGMMoveC - Circular EGM movement with path correction

Usage

`EGMMoveC` is used to move the tool center point (TCP) circularly to a given destination with path correction. During the movement the orientation normally remains unchanged relative to the circle.

Basic examples

The following example illustrates the instruction `EGMMoveC`.

Example 1

```
VAR egmident EGMid1;
PERS tooldata tReg := [TRUE, [[148,0,326],
    [0.8339007,0,0.551914,0]], [1,[0,0,100], [1,0,0,0], 0,0,0]];
PERS tooldata tLaser := [TRUE, [[148,50,326],
    [0.3902618,-0.589657,-0.589656,0.3902630]],
    [1,[-0.92,0,-0.39], [1,0,0,0], 0,0,0]];
EGMGetId EGMid1;
EGMSetupLTAPP ROB_1, EGMid1, "pathCorr", "OptSim", 1\LATR;
EGMActMove EGMid1, tLaser.tframe\SampleRate:=48;
MoveL p6, v10, fine, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p12, v10, z5, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p7, v10, z5, tReg\WObj:=wobj0;
EGMMoveC EGMid1, p13, p14, v10, z5, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p15, v10, fine, tReg\WObj:=wobj0;
MoveL p8, v1000, z10, tReg\WObj:=wobj0;
EGMReset EGMid1;
```

This program registers an EGM process, and sets up a sensor that uses the communication protocol LTAPP and is of the type *look-ahead* as data source (sensor). The sensor shall use the joint type definition number 1 for the tracking. The rate at which the controller will access the device and the sensor frame of the device are also setup.

The robot is moved to the start point of the tracking path with a `MoveL` instruction. The `EGMMove` instructions perform the robot movement with corrections from the sensor.

Finally the robot is moved to a departure position, and the EGM identity is released.

Arguments

```
EGMMoveC EGMid, CirPoint, ToPoint, Speed, Zone, Tool, [\Wobj]
    [\TLoad] [\NoCorr]
```

`EGMid`

Data type: `egmident`

EGM identity.

`CirPoint`

Data type: `robtarget`

The circle point of the robot. The circle point is a position on the circle between the start point and the destination point. To obtain the best accuracy it should be

Continues on next page

placed about halfway between the start and destination points. If it is placed too close to the start or destination point, the robot may give a warning. The circle point is defined as a named position or stored directly in the instruction (marked with an * in the instruction). The position of the external axes are not used.

ToPoint

Data type:robtarget

The destination point of the robot and external axes. It is defined as a named position or stored directly in the instruction (marked with an * in the instruction).

Speed

Data type: speeddata

The speed data that applies to movements. Speed data defines the velocity of the TCP, the tool reorientation, and external axes.

Zone

Data type:zonedata

Zone data for the movement. Zone data describes the size of the generated corner path.

Tool

Data type: tooldata

The tool in use when the robot moves. The tool center point is the point that is moved to the specified destination point.

[\WObj]

Work Object

Data type: wobjdata

The work object (object coordinate system) to which the robot position in the instruction is related.

This argument can be omitted and if it is then the position is related to the world coordinate system. If, on the other hand, a stationary TCP or coordinated external axes are used this argument must be specified in order for a circle relative to the work object to be executed.

[\TLoad]

Total load

Data type: loaddata

The \TLoad argument describes the total load used in the movement. The total load is the tool load together with the payload that the tool is carrying. If the \TLoad argument is used, then the loaddata in the current tooldata is not considered.

If the \TLoad argument is set to load0, then the \TLoad argument is not considered and the loaddata in the current tooldata is used instead.

To be able to use the \TLoad argument it is necessary to set the value of the system parameter ModalPayLoadMode to 0. If ModalPayLoadMode is set to 0, it is no longer possible to use the instruction GripLoad.

The total load can be identified with the service routine LoadIdentify. If the system parameter `ModalPayLoadMode` is set to 0, the operator has the possibility to copy the `loaddata` from the tool to an existing or new `loaddata` persistent variable when running the service routine.

It is possible to test run the program without any payload by using a digital input signal connected to the system input `SimMode` (**Simulated Mode**). If the digital input signal is set to 1, the `loaddata` in the optional argument `\TLoad` is not considered, and the `loaddata` in the current `tooldata` is used instead.

> **Note**
>
> The default functionality to handle payload is to use the instruction `GripLoad`. Therefore the default value of the system parameter `ModalPayLoadMode` is 1.

[\NoCorr]

Data type: `switch`

Path correction is switched off.

Program execution

`EGMMoveC` moves the robot along a programmed circular path with superimposed corrections from a sensor. During the movement the instruction requests correction data from the sensor at the rate set up with `EGMActMove`. If the optional argument `\NoCorr` is present, no correction is added to the programmed path.

Error handling

The following recoverable errors are generated and can be handled in an error handler. The system variable `ERRNO` will be set to:

Name	Cause of error
ERR_UDPUC_COMM	An error occurred in the communication with the UdpUc device.
	In synchronous mode ERR_UDPUC_COMM is always a recoverable error, which can be handled by an error handler.
	In asynchronous mode (EGMRunX\NoWait) ERR_UDPUC_COMM is always raised as fatal error if EGM is executed.

Limitations

- `EGMMoveC` can only be used in RAPID motion tasks.
- `EGMMoveC` cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

Syntax

```
EGMMoveC
  [GMid ':='] <variable (VAR) of egmident> ','
  [CirPoint ':='] < expression (IN) of robtarget> ','
  [ToPoint ':='] < expression (IN) of robtarget> ','
  [Speed ':='] < expression (IN) of speeddata> ','
```

```
[Zone ':=']  < expression (IN) of zonedata> ','
[Tool ':=']  < persistent (PERS) of tooldata>
['\'WObj ':=' < persistent (PERS) of wobjdata>]
['\'TLoad ':=' < persistent (PERS) of loaddata>]
['\'NoCorr] ';'
```

Related information

For information about	See
MoveC	*Technical reference manual - RAPID Instructions, Functions and Data types*

5.1.6 EGMMoveL - Linear EGM movement with path correction

Usage

`EGMMoveL` is used to move the tool center point (TCP) linearly to a given destination with path correction. When the TCP is to remain stationary then this instruction can also be used to reorient the tool.

Basic examples

The following example illustrates the instruction `EGMMoveL`.

Example 1

```
VAR egmident EGMid1;
PERS tooldata tReg := [TRUE, [[148,0,326],
     [0.8339007,0,0.551914,0]], [1,[0,0,100], [1,0,0,0], 0,0,0]];
PERS tooldata tLaser := [TRUE, [[148,50,326],
     [0.3902618,-0.589657,-0.589656,0.3902630]],
     [1,[-0.92,0,-0.39], [1,0,0,0], 0,0,0]];
EGMGetId EGMid1;
EGMSetupLTAPP ROB_1, EGMid1, "pathCorr", "OptSim", 1\LATR;
EGMActMove EGMid1, tLaser.tframe\SampleRate:=48;
MoveL p6, v10, fine, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p12, v10, z5, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p7, v10, z5, tReg\WObj:=wobj0;
EGMMoveC EGMid1, p13, p14, v10, z5, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p15, v10, fine, tReg\WObj:=wobj0;
MoveL p8, v1000, z10, tReg\WObj:=wobj0;
EGMReset EGMid1;
```

This program registers an EGM process, and sets up a sensor that uses the communication protocol LTAPP and is of the type *look-ahead* as data source (sensor). The sensor shall use the joint type definition number 1 for the tracking. The rate at which the controller will access the device and the sensor frame of the device are also setup.

The robot is moved to the start point of the tracking path with a `MoveL` instruction. The `EGMMove` instructions perform the robot movement with corrections from the sensor.

Finally the robot is moved to a departure position, and the EGM identity is released.

Arguments

```
EGMMoveL EGMid, ToPoint, Speed, Zone, Tool, [\Wobj] [\TLoad]
     [\NoCorr]
```

`EGMid`

Data type: `egmident`

EGM identity.

`ToPoint`

Data type: `robtarget`

The destination point of the robot and external axes. It is defined as a named position or stored directly in the instruction (marked with an * in the instruction).

Continues on next page

Speed

Data type: `speeddata`

The speed data that applies to movements. Speed data defines the velocity of the TCP, the tool reorientation, and external axes.

Zone

Data type:`zonedata`

Zone data for the movement. Zone data describes the size of the generated corner path.

Tool

Data type: `tooldata`

The tool in use when the robot moves. The tool center point is the point that is moved to the specified destination point.

[\WObj]

Work Object

Data type: `wobjdata`

The work object (object coordinate system) to which the robot position in the instruction is related.

This argument can be omitted and if it is then the position is related to the world coordinate system. If, on the other hand, a stationary TCP or coordinated external axes are used this argument must be specified in order for a circle relative to the work object to be executed.

[\TLoad]

Total load

Data type: `loaddata`

The \TLoad argument describes the total load used in the movement. The total load is the tool load together with the payload that the tool is carrying. If the \TLoad argument is used, then the `loaddata` in the current `tooldata` is not considered.

If the \TLoad argument is set to `load0`, then the \TLoad argument is not considered and the `loaddata` in the current `tooldata` is used instead.

To be able to use the \TLoad argument it is necessary to set the value of the system parameter `ModalPayLoadMode` to 0. If `ModalPayLoadMode` is set to 0, it is no longer possible to use the instruction `GripLoad`.

The total load can be identified with the service routine LoadIdentify. If the system parameter `ModalPayLoadMode` is set to 0, the operator has the possibility to copy the `loaddata` from the tool to an existing or new `loaddata` persistent variable when running the service routine.

It is possible to test run the program without any payload by using a digital input signal connected to the system input `SimMode` (Simulated Mode). If the digital

input signal is set to 1, the `loaddata` in the optional argument `\TLoad` is not considered, and the `loaddata` in the current `tooldata` is used instead.

> **ℹ Note**
>
> The default functionality to handle payload is to use the instruction `GripLoad`. Therefore the default value of the system parameter `ModalPayLoadMode` is 1.

[\NoCorr]

Data type: `switch`

Path correction is switched off.

Program execution

`EGMMoveL` moves the robot along a programmed linear path with superimposed corrections from a sensor. During the movement the instruction requests correction data from the sensor at the rate set up with `EGMActMove`. If the optional argument `\NoCorr` is present, no correction is added to the programmed path.

Error handling

The following recoverable errors are generated and can be handled in an error handler. The system variable `ERRNO` will be set to:

Name	Cause of error
ERR_UDPUC_COMM	An error occurred in the communication with the UdpUc device. In synchronous mode `ERR_UDPUC_COMM` is always a recoverable error, which can be handled by an error handler. In asynchronous mode (`EGMRunX\NoWait`) `ERR_UDPUC_COMM` is always raised as fatal error if EGM is executed.

Limitations

- `EGMMoveL` can only be used in RAPID motion tasks.
- `EGMMoveL` cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

Syntax

```
EGMMoveL
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ToPoint ':='] < expression (IN) of robtarget> ','
  [Speed ':='] < expression (IN) of speeddata> ','
  [Zone ':='] < expression (IN) of zonedata> ','
  [Tool ':='] < persistent (PERS) of tooldata>
  ['\'WObj ':=' < persistent (PERS) of wobjdata>]
  ['\'TLoad ':=' < persistent (PERS) of loaddata>]
  ['\'NoCorr] ';'
```

Related information

For information about	See
MoveL	*Technical reference manual - RAPID Instructions, Functions and Data types*

5.1.7 EGMReset - Reset an EGM process

Usage

EGMReset **resets a specific EGM process** (EGMid), **that is, the reservation is canceled.**

Basic examples

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0], [1,0,0,0]];
CONST egm_minmax egm_minmax_lin:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot:=[-0.1,0.2];
CONST pose posecor:=[[1200,400,900], [0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
      [0.903899,-0.00320735,0.427666,0.00765917]];


EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
      \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
      \aiR6rz:=ai_06;
EGMActPose egmID1 \Tool:=tool0 \WObj:=wobj0, posecor,
      EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
      \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
      \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
      \RampInTime:=0.05;
EGMReset egmID1;
```

Arguments

```
EGMReset EGMid
```

EGMid

    Data type: egmident

    EGM identity.

Syntax

```
EGMReset
  [EGMid ':='] <variable (VAR) of egmident>';'
```

5.1.8 EGMRunJoint - Perform an EGM movement with a joint target

Usage

`EGMRunJoint` performs a sensor guided movement to a joint target from a fine point for a specific EGM process (`EGMid`) and defines which joints will be moved.

Basic examples

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0],[1,0,0,0]];
CONST egm_minmax egm_minmax1:=[-1,1];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Joint \aiR1x:=ai_01
      \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
      \aiR6rz:=ai_06;
EGMActJoint egmID1, \J1:=egm_minmax1 \J3:=egm_minmax1
      \J4:=egm_minmax1;
EGMRunJoint egmID1, EGM_STOP_HOLD \J1 \J3 \RampInTime:=0.05;
```

Arguments

```
EGMRunJoint EGMid, Mode [\NoWaitCond] [\J1] [\J2] [\J3] [\J4] [\J5]
      [\J6] [\J7] [\CondTime] [\RampInTime] [\RampOutTime]
      [\PosCorrGain]
```

`EGMid`

Data type: `egmident`

EGM identity.

`Mode`

Data type: `egmstopmode`

Defines how the movement is ended (`EGM_STOP_HOLD`, `EGM_STOP_RAMP_DOWN`)

`[\NoWaitCond]`

Data type: `switch`

If this optional argument is used, `EGMRunJoint` will release the RAPID program pointer before the movement is completed. It is then mandatory to use the RAPID instruction `EGMWaitCond` to complete the EGM Position Guidance movement. Between `EGMRunJoint` and `EGMWaitCond` no other movement instruction is allowed.

`[\J1] [\J2] [\J3] [\J4] [\J5] [\J6] [\J7]`

Data type: `switch`

Move joint 1 to 6 for 6-axis robots, and joint 1 to 7 for 7-axis robots.

`[\CondTime]`

Data type: `num`

The time in seconds that the convergence criteria defined in `EGMActJoint` has to be fulfilled before the target point is considered to be reached and `EGMRunJoint` releases RAPID execution to continue to the next instruction.

Continues on next page

5.1.8 EGMRunJoint - Perform an EGM movement with a joint target
Externally Guided Motion
Continued

The default value is 1 s.

[\RampInTime]

Data type: `num`

Defines in seconds how fast EGM ramps in the movement after start, from the initial position to full tracking of the input position and speed. Defines in seconds how fast the movement is started.

[\RampOutTime]

Data type: `num`

Defines in seconds how fast a ramp down of EGM to the EGM start position will be performed in case parameter `Mode` is set to `EGM_STOP_RAMP_DOWN`.

This parameter has no meaning if the parameter `Mode` is set to `EGM_STOP_HOLD`.

The default ramp time is if stop is set from other inputs than RAPID, for example, program stop or similar.

[\PosCorrGain]

Data type: `num`

Position correction gain. A value between 0 and 1, default 1.

Error handling

The following recoverable errors are generated and can be handled in an error handler. The system variable `ERRNO` will be set to:

Name	Cause of error
ERR_UDPUC_COMM	An error occurred in the communication with the UdpUc device.
	In synchronous mode `ERR_UDPUC_COMM` is always a recoverable error, which can be handled by an error handler.
	In asynchronous mode (`EGMRunX\NoWait`) `ERR_UDPUC_COMM` is always raised as fatal error if EGM is executed.

Limitations

- Before the first use of `EGMRunJoint` the robot must have been moved since the controller was started by the execution of a `Move` instruction from RAPID.
- The starting point for an `EGMRunJoint` movement has to be a fine point.
- `EGMRunJoint` can only be used in RAPID motion tasks.
- If the instruction `EGMActPose` was executed instead of `EGMActJoint`, the following fatal error will occur: *41826 EGM mode mismatch*.
- If none of the switches `\J1` to `\J7` are specified, no movement is performed and RAPID execution continues to the next RAPID instruction.

Syntax

```
EGMRunJoint
  [EGMid ':='] <variable (VAR) of egmident> ','
  [Mode ':='] < expression (IN) of egmstopmode>
  ['\'NoWaitCond]
```

```
['\'J1]
['\'J2]
['\'J3]
['\'J4]
['\'J5]
['\'J6]
['\'J7]
['\'CondTime ':=' <expression (IN) of num>]
['\'RampInTime ':=' <expression (IN) of num>]
['\'RampOutTime ':=' <expression (IN) of num>]
['\'PosCorrGain ':=' <expression (IN) of num>] ';'
```

Related information

For information about	See
egmstopmode	*egmstopmode - Defines stop modes for EGM on page 97*
MoveJ	*Technical reference manual - RAPID Instructions, Functions and Data types*

5.1.9 EGMRunPose - Perform an EGM movement with a pose target
Externally Guided Motion

5.1.9 EGMRunPose - Perform an EGM movement with a pose target

Usage

`EGMRunPose` performs a sensor guided movement to a pose target from a fine point for a specific EGM process (`EGMid`) and defines which directions and orientations may be changed.

Basic examples

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0],[1,0,0,0]];
CONST egm_minmax egm_minmax_lin:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot:=[-0.1,0.2];
CONST pose posecor:=[[1200,400,900],[0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
      [0.903899,-0.00320735,0.427666,0.00765917]];


EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
      \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
      \aiR6rz:=ai_06;
EGMActPose egmID1 \Tool:=tool0 \WObj:=wobj0, posecor,
      EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
      \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
      \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
      \RampInTime:=0.05;
```

Arguments

```
EGMRunPose EGMid, Mode [\NoWaitCond] [\x] [\y] [\z] [\rx] [\ry]
      [\rz] [\CondTime] [\RampInTime] [\RampOutTime] [\Offset]
      [\PosCorrGain]
```

`EGMid`

Data type: `egmident`

EGM identity.

`Mode`

Data type: `egmstopmode`

Defines how the movement is ended (`EGM_STOP_HOLD`, `EGM_STOP_RAMP_DOWN`)

`[\NoWaitCond]`

Data type: `switch`

If this optional argument is used, `EGMRunPose` will release the RAPID program pointer before the movement is completed. It is then mandatory to use the RAPID instruction `EGMWaitCond` to complete the EGM Position Guidance movement. Between `EGMRunPose` and `EGMWaitCond` no other movement instruction is allowed.

`[\x] [\y] [\z]`

Data type: `switch`

Movement in x, y, and z direction.

Continues on next page

Application manual - Externally Guided Motion
3HAC073319-001 Revision: H

[\rx] [\ry] [\rz]

> **Data type:** switch
>
> Reorientation around x, y, and z axes.

[\CondTime]

> **Data type:** num
>
> The time in seconds that the convergence criteria defined in EGMActPose has to be fulfilled before the target point is considered to be reached and EGMRunPose releases RAPID execution to continue to the next instruction.
>
> The default value is 1 s.

[\RampInTime]

> **Data type:** num
>
> Defines in seconds how fast EGM ramps in the movement after start, from the initial position to full tracking of the input position and speed. Defines in seconds how fast the movement is started.

[\RampOutTime]

> **Data type:** num
>
> Defines in seconds how fast a ramp down of EGM to the EGM start position will be performed in case parameter Mode is set to EGM_STOP_RAMP_DOWN.
>
> This parameter has no meaning if the parameter Mode is set to EGM_STOP_HOLD.
>
> The default ramp time is if stop is set from other inputs than RAPID, for example, program stop or similar.

[\Offset]

> **Data type:** pose
>
> Possibility to define a static offset on top of the value given by the sensor.

[\PosCorrGain]

> **Data type:** num
>
> Position correction gain. A value between 0 and 1, default 1.

Error handling

> The following recoverable errors are generated and can be handled in an error handler. The system variable ERRNO will be set to:

Name	Cause of error
ERR_UDPUC_COMM	An error occurred in the communication with the UdpUc device.
	In synchronous mode ERR_UDPUC_COMM is always a recoverable error, which can be handled by an error handler.
	In asynchronous mode (EGMRunX\NoWait) ERR_UDPUC_COMM is always raised as fatal error if EGM is executed.

5.1.9 EGMRunPose - Perform an EGM movement with a pose target
Externally Guided Motion
Continued

Limitations

- Before the first use of `EGMRunPose` the robot must have been moved since the controller was started by the execution of a `Move` instruction from RAPID.
- The starting point for an `EGMRunPose` movement has to be a fine point.
- `EGMRunPose` can only be used in RAPID motion tasks.
- If the instruction `EGMActJoint` was executed instead of `EGMActPose`, the following fatal error will occur: *41826 EGM mode mismatch*.
- If none of the switches `\x` to `\rz` are specified, no movement is performed and RAPID execution continues to the next RAPID instruction.

Syntax

```
EGMRunPose
  [EGMid ':='] <variable (VAR) of egmident>','
  [Mode ':='] < expression (IN) of egmstopmode>
  ['\'NoWaitCond]
  ['\'x]
  ['\'y]
  ['\'z]
  ['\'rx]
  ['\'ry]
  ['\'rz]
  ['\'CondTime ':=' <expression (IN) of num>]
  ['\'RampInTime ':=' <expression (IN) of num>]
  ['\'RampOutTime ':=' <expression (IN) of num>]
  ['\'Offset ':=' <expression (IN) of pose>]
  ['\'PosCorrGain ':=' <expression (IN) of num>] ';'
```

Related information

For information about	See
`egmstopmode`	*egmstopmode - Defines stop modes for EGM on page 97*

5.1.10 EGMSetupAI - Setup analog input signals for EGM

Usage

EGMSetupAI is used to set up analog input signals for a specific EGM process (EGMid), as the source for position destination values to which the robot, and up to 6 additional axis, is to be guided.

EGM joint mode is the only EGM mode that supports 7-axis robots. For 7-axis robots, the first additional axis input provides the position for the additional robot axis.

Basic examples

```
VAR egmident egmID1;

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
    \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
    \aiR6rz:=ai_06;
```

Arguments

```
EGMSetupAI MecUnit, EGMid, ExtConfigName [\Joint] | [\Pose] |
    [\PathCorr] [\APTR] [\aiR1x] [\aiR2y] [\aiR3z] [\aiR4rx]
    [\aiR5ry] [\aiR6rz] [\aiE1] [\aiE2] [\aiE3] [\aiE4] [\aiE5]
    [\aiE6]
```

MecUnit

  Data type: mecunit

  Mechanical unit name.

EGMid

  Data type: egmident

  EGM identity.

ExtConfigName

  Data type: string

  The name of the external motion interface data as defined in the system parameters.

  For more information see *Technical reference manual - System parameters*, type *External Motion Interface Data*, topic *Motion*.

[\Joint]

  Data type: switch

  Selects joint movement for position guidance.

  At least one of the switches \Joint, \Pose, or \PathCorr has to be present.

[\Pose]

  Data type: switch

  Selects pose movement for position guidance.

  At least one of the switches \Joint, \Pose, or \PathCorr has to be present.

Continues on next page

5 RAPID reference information

[\PathCorr]

        Data type: `switch`

        Selects path correction.

        At least one of the switches \Joint, \Pose, **or** \PathCorr **has to be present.**

[\APTR]

        Data type: `switch`

        Setup an at-point-tracker type of sensor for path correction. For example WeldGuide or AWC.

        \APTR **has to be present if** \PathCorr **is used.**

[\aiR1x] [\aiR2y] [\aiR3z]

        Data type: `signalai`

        Specifies the signal that provides the x, y, and z value in millimeters for pose movement.

        Specifies the signal that provides the robot joint 1 to 3 angle in degrees for joint movement.

[\aiR4rx] [\aiR5ry] [\aiR6rz]

        Data type: `signalai`

        Specifies the signal that provides the rotation x, y, and z value of the robot in degrees for pose movement.

        Specifies the signal that provides the robot joint 4 to 6 angle in degrees for joint movement.

[\aiE1] [\aiE2] [\aiE3] [\aiE4] [\aiE5] [\aiE6]

        Data type: `signalai`

        Specifies the signal that provides the position of the additional axis joint 1 to 6.

        When using EGM joint mode with a 7-axis robot, then \aiE1 **provides the position for the additional robot axis.**

Error handling

        The following recoverable errors are generated and can be handled in an error handler. The system variable ERRNO will be set to:

Name	Cause of error
ERR_NO_ALIASIO_DEF	The signal variable is a variable declared in RAPID. It has not been connected to an I/O signal defined in the I/O configuration with instruction `AliasIO`.
ERR_NORUNUNIT	There is no contact with the I/O device.
ERR_SIG_NOT_VALID	The I/O signal cannot be accessed (only valid for ICI field bus).

Limitations

- EGMSetupAI **can only be used in RAPID motion tasks.**
- **The mechanical unit has to be a TCP robot.**

Continues on next page

- At least one signal has to be specified, otherwise an error is sent and RAPID execution is stopped.

Syntax

```
EGMSetupAI
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] <expression (IN) of string>
  [['\'Joint] | ['\'Pose] | ['\'PathCorr ]]
  ['\'APTR]
  ['\'aiR1x ':=' <variable (VAR) of signalai>]
  ['\'aiR2y ':=' <variable (VAR) of signalai>]
  ['\'aiR3z ':=' <variable (VAR) of signalai>]
  ['\'aiR4rx ':=' <variable (VAR) of signalai>]
  ['\'aiR5ry ':=' <variable (VAR) of signalai>]
  ['\'aiR6rz ':=' <variable (VAR) of signalai>]
  ['\'aiE1 ':=' <variable (VAR) of signalai>]
  ['\'aiE2 ':=' <variable (VAR) of signalai>]
  ['\'aiE3 ':=' <variable (VAR) of signalai>]
  ['\'aiE4 ':=' <variable (VAR) of signalai>]
  ['\'aiE5 ':=' <variable (VAR) of signalai>]
  ['\'aiE6 ':=' <variable (VAR) of signalai>] ';'
```

Related information

For information about	See
System parameters	*System parameters on page 35*

5.1.11 EGMSetupAO - Setup analog output signals for EGM

Usage

`EGMSetupAO` is used to set up AO signals for a specific EGM process (`EGMid`) as the source for position destination values to which the robot, and up to 6 additional axis, is to be guided.

EGM joint mode is the only EGM mode that supports 7-axis robots. For 7-axis robots, the first additional axis input provides the position for the additional robot axis.

Basic examples

```
VAR egmident egmID1;


EGMGetId egmID1;
EGMSetupAO ROB_1, egmID1, "default" \Pose \aoR1x:=ao_01
        \aoR2y:=ao_02 \aoR3z:=ao_03 \aoR4rx:=ao_04 \aoR5ry:=ao_05
        \aoR6rz:=ao_06;
```

Arguments

```
EGMSetupAO MecUnit, EGMid, ExtConfigName [\Joint] | [\Pose] |
        [\PathCorr] [\APTR] [\aoR1x] [\aoR2y] [\aoR3Z] [\aoR4rx]
        [\aoR5ry] [\aoR6rz] [\aoE1] [\aoE2] [\aoE3] [\aoE4] [\aoE5]
        [\aoE6]
```

`MecUnit`

Data type: `mecunit`

Mechanical unit name.

`EGMid`

Data type: `egmident`

EGM identity.

`ExtConfigName`

Data type: `string`

The name of the external motion interface data as defined in the system parameters.

For more information see *Technical reference manual - System parameters* , type *External Motion Interface Data*, topic *Motion*.

`[\Joint]`

Data type: `switch`

Selects joint movement.

At least one of the switches \Joint **or** \Pose **has to be present.**

`[\Pose]`

Data type: `switch`

Selects pose movement.

At least one of the switches \Joint **or** \Pose **has to be present.**

Continues on next page

Application manual - Externally Guided Motion
3HAC073319-001 Revision: H

[\PathCorr]

Data type: `switch`

Selects path correction.

At least one of the switches `\Joint`, `\Pose`, or `\PathCorr` has to be present.

[\APTR]

Data type: `switch`

Setup an at-point-tracker type of sensor for path correction. For example WeldGuide or AWC.

`\APTR` has to be present if `\PathCorr` is used.

[\aoR1x] [\aoR2y] [\aoR3z]

Data type: `signalao`

Specifies the signal that provides the x, y, and z value in millimeters for pose movement.

Specifies the signal that provides the robot joint 1 to 3 angle in degrees for joint movement.

[\aoR4rx] [\aoR5ry] [\aoR6rz]

Data type: `signalao`

Specifies the signal that provides the rotation x, y, and z value of the robot in degrees for pose movement.

Specifies the signal that provides the robot joint 4 to 6 angle in degrees for joint movement.

[\aoE1] [\aoE2] [\aoE3] [\aoE4] [\aoE5] [\aoE6]

Data type: `signalao`

Specifies the signal that provides the position of the additional axis joint 1 to 6.

When using EGM joint mode with a 7-axis robot, then `\aoE1` provides the position for the additional robot axis.

Error handling

The following recoverable errors are generated and can be handled in an error handler. The system variable `ERRNO` will be set to:

Name	Cause of error
ERR_NO_ALIASIO_DEF	The signal variable is a variable declared in RAPID. It has not been connected to an I/O signal defined in the I/O configuration with instruction `AliasIO`.
ERR_NORUNUNIT	There is no contact with the I/O device.
ERR_SIG_NOT_VALID	The I/O signal cannot be accessed (only valid for ICI field bus).

Limitations

- `EGMSetupAO` can only be used in RAPID motion tasks.
- The mechanical unit has to be a TCP robot.

- At least one signal has to be specified, otherwise an error is sent and RAPID execution is stopped.

Syntax

```
EGMSetupAO
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] <expression (IN) of string>
  [['\'Joint] | ['\'Pose] | ['\'PathCorr]]
  ['\'APTR]
  ['\'aoR1x ':=' <variable (VAR) of signalao>]
  ['\'aoR2y ':='] <variable (VAR) of signalao>]
  ['\'aoR3z ':='] <variable (VAR) of signalao>]
  ['\'aoR4rx ':='] <variable (VAR) of signalao>]
  ['\'aoR5ry ':='] <variable (VAR) of signalao>]
  ['\'aoR6rz ':='] <variable (VAR) of signalao>]
  ['\'aoE1 ':='] <variable (VAR) of signalao>]
  ['\'aoE2 ':='] <variable (VAR) of signalao>]
  ['\'aoE3 ':='] <variable (VAR) of signalao>]
  ['\'aoE4 ':='] <variable (VAR) of signalao>]
  ['\'aoE5 ':='] <variable (VAR) of signalao>]
  ['\'aoE6 ':='] <variable (VAR) of signalao>] ';'
```

Related information

For information about	See
System parameters	*System parameters on page 35*

5.1.12 EGMSetupGI - Setup group input signals for EGM

Usage

EGMSetupGI is used to set up group input signals for a specific EGM process (EGMid) as the source for position destination values to which the robot, and up to 6 additional axis, is to be guided.

EGM joint mode is the only EGM mode that supports 7-axis robots. For 7-axis robots, the first additional axis input provides the position for the additional robot axis.

Basic examples

```
VAR egmident egmID1;

EGMGetId egmID1;
EGMSetupGI ROB_1, egmID1, "default" \Pose \giR1x:=gi_01
    \giR2y:=gi_02 \giR3z:=gi_03 \giR4rx:=gi_04 \giR5ry:=gi_05
    \giR6rz:=gi_06;
```

Arguments

```
EGMSetupGI MecUnit, EGMid, ExtConfigName [\Joint] | [\Pose] |
    [\PathCorr] [\APTR] [\giR1x] [\giR2y] [\giR3Z] [\giR4rx]
    [\giR5ry] [\giR6rz] [\giE1] [\giE2] [\giE3] [\giE4] [\giE5]
    [\giE6]
```

MecUnit

Data type: mecunit

Mechanical unit name.

EGMid

Data type: egmident

EGM identity.

ExtConfigName

Data type: string

The name of the external motion interface data as defined in the system parameters.

For more information see *Technical reference manual - System parameters*, type *External Motion Interface Data*, topic *Motion*.

[\Joint]

Data type: switch

Selects joint movement.

At least one of the switches \Joint or \Pose has to be present.

[\Pose]

Data type: switch

Selects pose movement.

At least one of the switches \Joint or \Pose has to be present.

Continues on next page

5 RAPID reference information

[\PathCorr]

> **Data type:** switch
>
> Selects path correction.
>
> **At least one of the switches** \Joint, \Pose, **or** \PathCorr **has to be present.**

[\APTR]

> **Data type:** switch
>
> Setup an at-point-tracker type of sensor for path correction. For example WeldGuide or AWC.
>
> \APTR **has to be present if** \PathCorr **is used.**

[\giR1x] [\giR2y] [\giR3z]

> **Data type:** signalgi
>
> Specifies the signal that provides the x, y, and z value in millimeters for pose movement.
>
> Specifies the signal that provides the robot joint 1 to 3 angle in degrees for joint movement.

[\giR4rx] [\giR5ry] [\giR6rz]

> **Data type:** signalgi
>
> Specifies the signal that provides the rotation x, y, and z value of the robot in degrees for pose movement.
>
> Specifies the signal that provides the robot joint 4 to 6 angle in degrees for joint movement.

[\giE1] [\giE2] [\giE3] [\giE4] [\giE5] [\giE6]

> **Data type:** signalgi
>
> Specifies the signal that provides the position of the additional axis joint 1 to 6.
>
> When using EGM joint mode with a 7-axis robot, then \giE1 provides the position for the additional robot axis.

Error handling

> The following recoverable errors are generated and can be handled in an error handler. The system variable ERRNO will be set to:

Name	Cause of error
ERR_NO_ALIASIO_DEF	The signal variable is a variable declared in RAPID. It has not been connected to an I/O signal defined in the I/O configuration with instruction AliasIO.
ERR_NORUNUNIT	There is no contact with the I/O device.
ERR_SIG_NOT_VALID	The I/O signal cannot be accessed (only valid for ICI field bus).

Limitations

- EGMSetupGI can only be used in RAPID motion tasks.
- The mechanical unit has to be a TCP robot.

Continues on next page

- Group signals can only handle positive values. Therefore their use in EGM is limited.
- At least one signal has to be specified, otherwise an error is sent and RAPID execution is stopped.

Syntax

```
EGMSetupGI
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] <expression (IN) of string>
  [['\'Joint] | ['\'Pose] | ['\'PathCorr]]
  ['\'APTR]
  ['\'giR1x ':=' <variable (VAR) of signalgi>]
  ['\'giR2y ':=' <variable (VAR) of signalgi>]
  ['\'giR3z ':=' <variable (VAR) of signalgi>]
  ['\'giR4rx ':=' <variable (VAR) of signalgi>]
  ['\'giR5ry ':=' <variable (VAR) of signalgi>]
  ['\'giR6rz ':=' <variable (VAR) of signalgi>]
  ['\'giE1 ':=' <variable (VAR) of signalgi>]
  ['\'giE2 ':=' <variable (VAR) of signalgi>]
  ['\'giE3 ':=' <variable (VAR) of signalgi>]
  ['\'giE4 ':=' <variable (VAR) of signalgi>]
  ['\'giE5 ':=' <variable (VAR) of signalgi>]
  ['\'giE6 ':=' <variable (VAR) of signalgi>] ';'
```

Related information

For information about	See
System parameters	*System parameters on page 35*

5.1.13 EGMSetupLTAPP - Setup the LTAPP protocol for EGM

Usage

EGMSetupLTAPP is used to set up an *LTAPP* protocol for a specific EGM process (EGMid) as the source for path corrections.

Basic examples

The following example illustrates the instruction EGMSetupLTAPP.

Example 1

```
VAR egmident EGMid1;
EGMGetId EGMid1;
EGMSetupLTAPP ROB_1, EGMid1, "pathCorr", "OptSim", 1\LATR;
```

This program registers an EGM process, and sets up the sensor OptSim that uses the communication protocol LTAPP and is of the type *look-ahead* as data source (sensor). The sensor shall use the joint type definition number 1 for the tracking.

Arguments

```
EGMActMove MecUnit, EGMid, ExtConfigName, Device, JointType [\APTR]
        | [\LATR]
```

MecUnit

Data type: mecunit

Mechanical unit name.

EGMid

Data type: egmident

EGM identity.

ExtConfigName

Data type: string

The name of the external motion interface data as defined in the system parameters.

For more information see *Technical reference manual - System parameters*, topic *Motion*, type *External Motion Interface Data*.

Device

Data type: string

LTAPP device name.

JointType

Data type: num

Defines the joint type, expressed as a number, that the sensor equipment shall use during path correction.

[\APTR]

Data type: switch

Setup an at-point-tracker type of sensor for path correction. For example WeldGuide or AWC.

Continues on next page

Either \APTR **or** \LATR **has to be present.**

[\LATR]

Data type: `switch`

Setup an Look-ahead-tracker type of sensor for path correction. For example Laser Tracker.

Either \APTR **or** \LATR **has to be present.**

Program execution

`EGMSetupLTAPP` connects the characteristic data of the sensor that is used to an EGM identity. That EGM identity can then be used in different `EGMActMove` and `EGMMove` instructions.

Syntax

```
EGMSetupLTAPP
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] < expression (IN) of string> ','
  [Device ':='] < expression (IN) of string> ','
  [JointType ':='] < expression (IN) of num>
  [['\'APTR] | ['\'LATR]] ';'
```

5.1.14 EGMSetupUC - Setup the UdpUc protocol for EGM

Usage

EGMSetupUC is used to set up a UdpUc protocol for a specific EGM process (EGMid) as the source for position destination values to which the robot, and up to 6 additional axis, is to be guided.

EGM joint mode is the only EGM mode that supports 7-axis robots. For 7-axis robots, the first additional axis input provides the position for the additional robot axis.

Basic examples

```
VAR egmident egmID1;
VAR string egmSensor:="egmSensor:";
EGMGetId egmID1;
EGMSetupUC ROB_1, egmID1, "default", egmSensor\Pose;
```

Arguments

```
EGMSetupUC MecUnit, EGMid, ExtConfigName, UCDevice [\Joint] |
        [\Pose] | [\PathCorr] [\APTR] | [\LATR] [\CommTimeout]
```

MecUnit

    Data type: mecunit

    Mechanical unit name.

EGMid

    Data type: egmident

    EGM identity.

ExtConfigName

    Data type: string

    The name of the external motion interface data as defined in the system parameters.

    For more information see *Technical reference manual - System parameters* , type *External Motion Interface Data*, topic *Motion*.

UCDevice

    Data type: string

    UdpUc device name.

[\Joint]

    Data type: switch

    Selects joint movement for position guidance.

    At least one of the switches \Joint, \Pose, **or** \PathCorr **has to be present.**

[\Pose]

    Data type: switch

    Selects pose movement for position guidance.

    At least one of the switches \Joint, \Pose, **or** \PathCorr **has to be present.**

Continues on next page

[\PathCorr]

Data type: switch

Selects path correction.

At least one of the switches \Joint, \Pose, or \PathCorr has to be present.

[\APTR]

Data type: switch

Setup an at-point-tracker type of sensor for path correction. For example WeldGuide or AWC.

Either \APTR or \LATR has to be present.

[\LATR]

Data type: switch

Setup an Look-ahead-tracker type of sensor for path correction. For example Laser Tracker.

Either \APTR or \LATR has to be present.

[\CommTimeout]

Data type: num

Time-out for communication with the external UdpUC device in seconds.

Error handling

The following recoverable errors are generated and can be handled in an error handler. The system variable ERRNO will be set to:

Name	Cause of error
ERR_UDPUC_COMM	An error occurred in the communication with the UdpUc device.
	In synchronous mode ERR_UDPUC_COMM is always a recoverable error, which can be handled by an error handler.
	In asynchronous mode (EGMRunX\NoWait) ERR_UDPUC_COMM is always raised as fatal error if EGM is executed.

Limitations

- EGMSetupUC can only be used in RAPID motion tasks.
- The mechanical unit has to be a TCP robot.

Syntax

```
EGMSetupUC
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] <expression (IN) of string> ','
  [UCDevice ':='] <expression (IN) of string>
  [['\'Joint] | ['\'Pose] | ['\'PathCorr]]
  [['\'APTR] | ['\'LATR]]
  ['\'CommTimeout ':=' <expression (IN) of num>] ';'
```

Related information

For information about	See
System parameters	*System parameters on page 35*

5.1.15 EGMStop - Stop an EGM movement

Usage

EGMStop **stops a specific EGM process (**EGMid**).**

Basic examples

In the RAPID motion task:
```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0], [1,0,0,0]];
CONST egm_minmax egm_minmax_lin:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot:=[-0.1,0.2];
CONST pose posecor:=[[1200,400,900], [0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
    [0.903899,-0.00320735,0.427666,0.00765917]];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1 \Pose \aiR1x:=ai_01 \aiR2y:=ai_02
    \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05 \aiR6rz:=ai_06;
EGMActPose egmID1 \Tool:=tool0 \WObj:=wobj0, posecor,
    EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
    \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
    \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
    \RampInTime:=0.05;
```

In a TRAP routine:
```
EGMStop egmID1, EGM_STOP_HOLD;
```

Arguments

    EGMStop EGMid, Mode [\RampOutTime]

EGMid

Data type: egmident

EGM identity.

Mode

Data type: egmstopmode

Defines how the movement is ended (EGM_STOP_HOLD, EGM_STOP_RAMP_DOWN**)**

[\RampOutTime]

Data type: num

Defines in seconds how fast a ramp down of EGM will be performed.

This parameter has no meaning if the parameter Mode **is set to** EGM_STOP_HOLD**.**

Limitations

- EGMStop **can only be used in RAPID motion tasks.**

5.1.15 EGMStop - Stop an EGM movement
Externally Guided Motion
Continued

Syntax

```
EGMStop
  [EGMid ':='] <variable (VAR) of egmident>','
  [Mode ':='] < expression (IN) of egmstopmode>
  ['\'RampOutTime ':=' <expression (IN) of num>] ';'
```

5.1.16 EGMStreamStart - start EGM position streaming

Usage

`EGMStreamStart` starts streaming position data for a specific EGM process (`EGMid`).

Basic example

The following example illustrates the instruction `EGMStreamStart`.

Example 1

```
VAR egmident egmID1;

EGMGetId egmID1;
EGMSetupUC ROB_1, egmID1, "default", "UCdevice"\Joint;
EGMStreamStart egmID;
MoveAbsJ jpos20, v100, z20, Weldgun;
MoveAbsJ jpos10\NoEOffs, v1000, fine, Weldgun;
EGMStreamStop egmID1;
EGMReset egmID1;
```

Arguments

```
EGMStreamStart EGMid [\SampleRate];
```

`EGMid`

> **Data type:** `egmident`
>
> EGM identity.

`[\SampleRate]`

> **Data type:** `num`
>
> Input data reading sample rate in multiples of 4 milliseconds. Valid values are 4, 8, 12, 16, etc.
>
> The default value is 4 milliseconds.

Program execution

`EGMStreamStart` starts streaming position data to external equipment. The data is sent at the cyclicity defined with `\SampleRate` and the content is according to the protocol specification in the Google Protobuf definition file *egm.proto*.

Limitations

`EGMStreamStart` is only available if EGM is set up using `EGMSetupUC`, i.e. the protocol UdpUc is used for communication with the external equipment.

Syntax

```
EGMStreamStart
  [EGMid ':='] <variable (VAR) of egmident>';'
  ['\' SampleRate ':=' <expression (IN) of num>]','
```

5.1.17 EGMStreamStop - stop EGM position streaming

Usage

EGMStreamStop stops streaming position data for a specific EGM process (EGMid).

Basic example

The following example illustrates the instruction EGMStreamStop.

Example 1

```
VAR egmident egmID1;

EGMGetId egmID1;
EGMSetupUC ROB_1, egmID1, "default", "UCdevice"\Joint;
EGMStreamStart egmID;
MoveAbsJ jpos20, v100, z20, Weldgun;
MoveAbsJ jpos10\NoEOffs, v1000, fine, Weldgun;
EGMStreamStop egmID1;
EGMReset egmID1;
```

Arguments

```
EGMStreamStop EGMid;
```

EGMid

Data type: egmident

EGM identity.

Program execution

EGMStreamStop stops streaming position data to external equipment.

Limitations

EGMStreamStop is only available if EGM is set up using EGMSetupUC, i.e. the protocol UdpUc is used for communication with the external equipment.

Syntax

```
EGMStreamStop
  [EGMid ':='] <variable (VAR) of egmident>';'
```

5.1.18 EGMWaitCond - wait for EGM process

Usage

EGMWaitCond is used to wait for a specific EGM process (EGMid).

Basic example

The following example illustrates the instruction EGMWaitCond.

Example 1

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0],[1,0,0,0]];
CONST egm_minmax egm_minmax_lin:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot:=[-0.1,0.2];
CONST pose posecor:=[[1200,400,900],[0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
        [0.903899,-0.00320735,0.427666,0.00765917]];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
        \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
        \aiR6rz:=ai_06;
EGMActPose egmID1 \Tool:=tool0 \WObj:=wobj0, posecor,
        EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
        \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
        \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry
        \rz\RampInTime:=0.05;
SetDO doSignal1, 1;
…
EGMWaitCond
```

Arguments

```
            EGMWaitCond EGMid;
```

EGMid

Data type: egmident

EGM identity.

Program execution

EGMWaitCond will wait for an EGMRunJoint/Pose instruction to complete. If the movement has been completed before EGMWaitCond is run, program execution will continue with the next RAPID instruction at once.

Limitations

If EGMRunJoint or EGMRunPose are used with the optional argument \NoWaitCond, no movement instruction must be used before the EGM Position Guidance is completed by using EGMWaitCond.

5.1.18 EGMWaitCond - wait for EGM process
Externally Guided Motion
Continued

Syntax

```
EGMWaitCond
  [EGMid ':='] <variable (VAR) of egmident>';'
```

5.2 Functions

5.2.1 EGMGetState - Gets the current EGM state

Usage

EGMGetState **retrieves the state of an EGM process (**EGMid**).**

Basic examples

```
VAR egmident egmID1;
VAR egmstate egmState1:= EGM_STATE_DISCONNECTED;

EGMGetId egmID1;
egmState1 := EGMGetState(egmID1);
```

Return value

Data type: egmstate

The current state of the EGM process identified by the EGM identity specified in the argument.

Arguments

EGMGetState (EGMid)

EGMid

Data type: egmident

EGM identity.

Limitations

- EGMGetState **can only be used in RAPID motion tasks.**
- **The mechanical unit has to be a TCP robot.**

Syntax

```
EGMGetState '('
  [EGMid ':='] < variable (VAR) of egmident >')'
```

Related information

For information about	See
egmstate	*egmstate - Defines the state for EGM*

5.3 Data types

5.3.1 egmframetype - Defines frame types for EGM

Usage

egmframetype is used to define the frame types for corrections and sensor measurements in EGM.

Description

egmframetype is intended to be used in the instructions EGMActJ and EGMActPose.

Basic examples

```
CONST egm_minmax egm_minmax_lin1:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot1:=[-0.1,0.2];

EGMActPose egmID1\Tool:=tFroniusCMT\WObj:=wobj0, posecor,
    EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
    \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
    \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
```

Predefined values

Value	Description
EGM_FRAME_BASE	The frame is defined relative to the base frame (pose mode).
EGM_FRAME_TOOL	The frame is defined relative to the used tool (pose mode).
EGM_FRAME_WOBJ	The frame is defined relative to the used work object (pose mode).
EGM_FRAME_WORLD	The frame is defined relative to the world frame (pose mode).
EGM_FRAME_JOINT	The values are joint values (joint mode).

Characteristics

egmframetype is an alias data type for num.

Related information

For information about	See
EGMActJ	*EGMActJoint - Prepare an EGM movement for a joint target on page 43*
EGMActPose	*EGMActPose - Prepare an EGM movement for a pose target on page 50*

5.3.2 egmident - Identifies a specific EGM process

Usage

`egmident` identifies a specific EGM process.

Description

An `egmident` is reserved using the instruction `EGMGetId`. It is then used to identify and link together the instructions `EGMSetupXX`, `EGMActX`, `EGMRunX`, and `EGMReset` to the same EGM operation.

An `egmident` is identified by its name, i.e. a second or third call of `EGMGetId` with the same `egmident` will neither reserve a new process nor change its content. Only `EGMReset` releases an `egmident`.

Basic examples

```
VAR egmident egmID1;
VAR egmstate egmSt1;
TASK PERS wobjdata wobj_EGM1:=[FALSE, TRUE, "", [[500,700,900],
      [1,0,0,0]], [[0,0,0], [1,0,0,0]]];
CONST pose posecor:=[[1200,400,900], [0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
      [0.903899,-0.00320735,0.427666,0.00765917]];
CONST egm_minmax egm_minmax_lin1:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot1:=[-0.1,0.2];
CONST egm_minmax egm_minmax_joint1:=[-0.1,0.1];

PROC testAI()
  EGMReset egmID1;
  EGMGetId egmID1;
  mvHome;
  mvHome_EGMLinear;

  egmSt1:=EGMGetState(egmID1);
  TPWrite "EGM state 1: " \Num:=egmSt1;

  IF egmSt1<=EGM_STATE_CONNECTED THEN
    EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_MoveX
        \aiR2y:=ai_MoveY \aiR3z:=ai_MoveZ \aiR5ry:=ai_RotY
        \aiR6rz:=ai_RotZ;
  ENDIF

  EGMActPose egmID1 \Tool:=tFroniusCMT \WObj:=wobj0, posecor,
        EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin1
        \y:=egm_minmax_lin1 \z:=egm_minmax_lin1 \rx:=egm_minmax_rot1
        \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1 \LpFilter:=20;
  EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
        \RampInTime:=0.05;

  egmSt1:=EGMGetState(egmID1);
  IF egmSt1=EGM_STATE_CONNECTED THEN
    TPWrite "Reset lin 1";
```

Continues on next page

```
        EGMReset egmID1;
    ENDIF
ENDPROC
```

Limitations

There are up to 4 concurrent instances available for each RAPID task.

Characteristics

`egmident` is a non-value data type. It is set by calling `EGMGetId`.

Related information

For information about	See
`EGMGetId`	*EGMGetId - Gets an EGM identity on page 55*

5.3.3 egm_minmax - Convergence criteria for EGM

Usage

egm_minmax **is used to define the convergence criteria for EGM to finish.**

Description

egm_minmax **is intended to be used in the instructions** EGMActJ **and** EGMActPose.

Components

Min

> **Data type:** num
>
> *Minimum deviation*
>
> **Defines the minimum value of the position deviation. The default value is -0.5 degrees.**

Max

> **Data type:** num
>
> *Maximum deviation*
>
> **Defines the maximum value of the position deviation. The default value is 0.5 degrees.**

Basic examples

```
CONST egm_minmax egm_minmax_lin1:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot1:=[-0.1,0.2];

EGMActPose egmID1\Tool:=tFroniusCMT\WObj:=wobj0, posecor,
    EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin1
    \y:=egm_minmax_lin1 \z:=egm_minmax_lin1 \rx:=egm_minmax_rot1
    \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1 \LpFilter:=20;
```

Characteristics

Egm_minmax **has the following units:**

- **Millimeters for x, y and z in linear movement.**
- **Degrees for rx, ry, and rz in linear movement and for joint movement.**

Structure

```
< dataobject of egm_minmax >
  < min of num >
  < max of num >
```

Related information

For information about	See
EGMActJ	*EGMActJoint - Prepare an EGM movement for a joint target on page 43*
EGMActPose	*EGMActPose - Prepare an EGM movement for a pose target on page 50*

5.3.4 egmstate - Defines the state for EGM

Usage

`egmstate` is used to define the state for corrections and sensor measurements in EGM.

Description

`egmstate` is the return value of the function `EGMGetState`.

Basic examples

```
VAR egmstate egmSt1;
VAR egmident egmID1;

EGMReset egmID1;
EGMGetId egmID1;

egmSt1:=EGMGetState(egmID1);
TPWrite "EGM state: "\Num:=egmSt1;
```

Predefined values

Value	Description
EGM_STATE_DISCONNECTED	The EGM state of the specific process is undefined. No setup is active.
EGM_STATE_CONNECTED	The specified EGM process is not activated. Setup has been made, but no EGM movement is active.
EGM_STATE_RUNNING	The specified EGM process is running. The EGM movement is active, i.e. the robot is moved.

Characteristics

`egmstate` is an alias data type for `num`.

Related information

For information about	See
EGMGetState	*EGMGetState - Gets the current EGM state on page 91*

5.3.5 egmstopmode - Defines stop modes for EGM

Usage

egmstopmode **is used to define the stop modes for corrections and sensor measurements in EGM.**

Description

egmstopmode **is intended to be used in the instructions** EGMRunJoint, EGMRunPose **and** EGMStop.

Basic examples

From the RAPID motion task:

```
VAR egmstate egmSt1;
VAR egmident egmID1;

EGMReset egmID1;
EGMGetId egmID1;
CONST egm_minmax egm_minmax_lin1:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot1:=[-0.1,0.2];

EGMActPose egmID1 \Tool:=tFroniusCMT \WObj:=wobj0, posecor,
    EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
    \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
    \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
    \RampInTime:=0.05;
```

From a RAPID TRAP or background task:

```
EGMStop egmID1, EGM_STOP_RAMP_DOWN\RampOutTime:=5.0;
```

Predefined values

Value	Description
EGM_STOP_HOLD	Keeps the EGM end position.
EGM_STOP_RAMP_DOWN	Returns from the EGM end position to the start position.

Characteristics

egmstopmode **is an alias data type for num.**

Related information

For information about	See
EGMRunJoint	*EGMRunJoint - Perform an EGM movement with a joint target on page 65*
EGMRunPose	*EGMRunPose - Perform an EGM movement with a pose target on page 68*
EGMStop	*EGMStop - Stop an EGM movement on page 85*

5 RAPID reference information

5.3.6 EGM_MAX_RAPID_DNUM - Defines the maximal size of the array

Usage

`EGM_MAX_RAPID_DNUM` is used to define the maximal size of the arrays send between robot and sensor. It can be accessed in RAPID.

Description

`EGM_MAX_RAPID_DNUM` is intended to be used in the instructions `EGMActJoint` and `EGMActPose`.

Basic examples

The following example illustrates the data type `EGM_MAX_RAPID_DNUM`.

```
VAR EGM_MAX_RAPID_DNUM;
```

Predefined values

`EGM_MAX_RAPID_DNUM` has a value of 40.

Characteristics

`EGM_MAX_RAPID_DNUM` is an alias data type for `num`.

5.4 Code examples

5.4.1 Using EGM Position Stream

Description

The device which provides the input data for EGM, first has to be configured as an UdpUc device. See *How to configure UdpUc devices on page 34*.

Now the device can be used by EGM to stream positions of mechanical units to an external equipment. Simple examples are shown below.

It is possible to stream positions from several motion tasks, but you have to use one communication channel for each motion task.

Examples

Using EGMStreamStart and EGMStreamStop for one mechanical unit

This method is the easiest way to use EGM Position Stream, but it is not accurate for robots with Absolute Accuracy or heavy load.

```
VAR egmident egmID1;
EGMGetId egmID1;
! Set up the EGM data source: UdpUc server using device "UCdevice"
    and configuration "default"
EGMSetupUC ROB_1, egmID1, "default", "UCdevice"\Joint;
! Start the position stream for T_ROB1 including active external
    axis. Cycle time is 16 ms.
EGMStreamStart egmID1\SampleRate:=16;
! Run your program – streaming is active
MoveAbsJ jpos20, v100, z20, Weldgun;
…
…
MoveAbsJ jpos10\NoEOffs, v1000, fine, Weldgun;
! Stop the position stream – but it is not necessary if you want
    to stream until the controller shuts down
EGMStreamStop egmID1;
EGMReset egmID1;
```

Using EGMActXX \StreamStart for one mechanical unit

This method is preferred if you have a robot with Absolute Accuracy, because the RAPID instructions `EGMActPose` and `EGMActJoint` pass the data for tool and load to the controller.

```
VAR egmident egmID1;
! Used tool
TASK PERS tooldata Weldgun:=[TRUE,[[12.3313,-0.108707,416.142],
  [0.903899,-0.00320735,0.427666,0.00765917]],
  [2.6,[-111.1,24.6,386.6],[1,0,0,0],0,0,0.072]];
! limits for cartesian convergence: +-1 mm
CONST egm_minmax egm_minmax_lin1:=[-1,1];
! limits for orientation convergence: +-2 degrees
CONST egm_minmax egm_minmax_rot1:=[-2,2];
! Correction frame offset: none
```

Continues on next page

```
VAR pose corr_frame_offs:=[[0,0,0],[1,0,0,0]];
EGMGetId egmID1;
! Set up the EGM data source: UdpUc server using device "UCdevice"
        and configuration "default"
EGMSetupUC ROB_1, egmID1, "default", "UCdevice"\Joint;
! Correction frame is the World coordinate system and the sensor
        measurements are relative to the tool frame of the used tool
        (Weldgun). Start the position stream for T_ROB1 including
        active external axis. Cycle time is 16 ms.
EGMActPose egmID1\StreamStart\Tool:= Weldgun, corr_frame_offs,
EGM_FRAME_WORLD, Weldgun.tframe, EGM_FRAME_TOOL
\x:=egm_minmax_lin1\y:=egm_minmax_lin1\z:=egm_minmax_lin1
\rx:=egm_minmax_rot1\ry:=egm_minmax_rot1\rz:=egm_minmax_rot1
\LpFilter:=20;
! Run your program – streaming is active
MoveAbsJ jpos20, v100, z20, Weldgun;
…
…
MoveAbsJ jpos10\NoEOffs, v1000, fine, Weldgun;
! Stop the position stream – but this is not necessary if you want
        to stream until the controller shuts down
EGMStreamStop egmID1;
EGMReset egmID1;
```

Using EGMStreamStart and EGMStreamStop for multiple mechanical units

This example is for a MultiMove system with two robots, each mounted on a track motion.

RAPID task for robot 1:

```
VAR egmident egmID1;
! Activate the mechanical unit for the track motion
ActUnit TRACK1;
EGMReset egmID1;
EGMGetId egmID1;
! Set up the EGM streaming destination for ROB1, including active
        additional axis, using device "UCdevice1" and configuration
        "default"
EGMSetupUC ROB_1, egmID1, "default", "UCdevice1"\Joint;
EGMStreamStart egmID1;
! Start the position stream for ROB1 including active additional
        axis. Cycle time is 4 ms (default).
! Run your program – streaming is active
MoveJ p10, v1000, z50, Weldgun;
…
…
MoveAbsJ jpos10\NoEOffs, v1000, fine, Weldgun;
! Stop the position stream
EGMStreamStop egmID1;
! Deactivate the mechanical unit for the track motion
DeactUnit TRACK1;
```

RAPID task for robot 2:

```
VAR egmident egmID2;
```

Application manual - Externally Guided Motion

```
! Activate the mechanical unit for the track motion
ActUnit TRACK2;
EGMReset egmID2;
EGMGetId egmID2;
! Set up the EGM streaming destination for ROB2, including active
    additional axis, using device "UCdevice2" and configuration
    "default"
EGMSetupUC ROB_2, egmID2, "default", "UCdevice2"\Joint;
! Start the position stream for ROB2 including active additional
    axis. Cycle time is 4 ms (default).
EGMStreamStart egmID2;
! Run your program – streaming is active
MoveJ p10, v1000, z50, PKI_500;
…
…
MoveAbsJ jpos10\NoEOffs, v1000, fine, PKI_500;
! Stop the position stream
EGMStreamStop egmID2;
! Deactivate the mechanical unit for the track motion
DeactUnit TRACK2;
```

5.4.2 Using EGM Position Guidance with an UdpUc device

Description

The device which provides the input data for EGM, first has to be configured as an UdpUc device. See *How to configure UdpUc devices on page 34*.

Now the device can be used by EGM to guide a robot. A simple example is the following:

Example

```
MODULE EGM_test
  VAR egmident egmID1;
  VAR egmstate egmSt1;

  ! limits for cartesian convergence: +-1 mm
  CONST egm_minmax egm_minmax_lin1:=[-1,1];
  ! limits for orientation convergence: +-2 degrees
  CONST egm_minmax egm_minmax_rot1:=[-2,2];

  ! Start position
  CONST jointtarget
        jpos10:=[[0,0,0,0,40,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  ! Used tool
  TASK PERS tooldata
        tFroniusCMT:=[TRUE,[[12.3313,-0.108707,416.142],
        [0.903899,-0.00320735,0.427666,0.00765917]],
        [2.6,[-111.1,24.6,386.6],[1,0,0,0],0,0,0.072]];
  ! corr-frame: wobj, sens-frame: wobj
  TASK PERS wobjdata wobj_EGM1:=[FALSE,TRUE,"",
        [[150,1320,1140],[1,0,0,0]], [[0,0,0],[1,0,0,0]]];
  ! Correction frame offset: none
  VAR pose corr_frame_offs:=[[0,0,0],[1,0,0,0]];

    PROC main()
      ! Move to start position. Fine point is demanded.
      MoveAbsJ jpos10\NoEOffs, v1000, fine, tFroniusCMT;
      testuc;
    ENDPROC

    PROC testuc()
      EGMReset egmID1;
      EGMGetId egmID1;

      egmSt1:=EGMGetState(egmID1);
      TPWrite "EGM state: "\Num:=egmSt1;

      IF egmSt1 <= EGM_STATE_CONNECTED THEN
        ! Set up the EGM data source: UdpUc server using device
            "EGMsensor:" and configuration "default"
        EGMSetupUC ROB_1, egmID1, "default", "EGMsensor:"\pose;
      ENDIF
```

Continues on next page

```
            ! Correction frame is the World coordinate system and the
                sensor measurements are relative to the tool frame of
                the used tool (tFroniusCMT)
            EGMActPose egmID1\Tool:=tFroniusCMT, corr_frame_offs,
                EGM_FRAME_WORLD, tFroniusCMT.tframe, EGM_FRAME_TOOL
                \x:=egm_minmax_lin1 \y:=egm_minmax_lin1
                \z:=egm_minmax_lin1 \rx:=egm_minmax_rot1
                \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1 \LpFilter:=20;
            ! Run: the convergence condition has to be fulfilled during
                2 seconds before RAPID execution continues to the next
                instruction
            EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \CondTime:=2
                \RampInTime:=0.05;


            egmSt1:=EGMGetState(egmID1);
            IF egmSt1 = EGM_STATE_CONNECTED THEN
              TPWrite "Reset EGM instance egmID1";
              EGMReset egmID1;
            ENDIF
        ENDPROC
    ENDMODULE
```

5.4.3 Using EGM Position Guidance with signals as input

Description

All signals that are used together with EGM has to be defined in the I/O configuration of the system. I.e. the signals that are set up with `EGMSetupAI`, `EGMSetupAO`, or `EGMSetupGI`. After that, the signals can be used by EGM to guide a robot.

The following RAPID program example uses analog output signals as input. The main reason for analog output signals is, that they are easier to simulate than analog input signals. In a real application group input signals and analog input signals might be more common.

In the examples below we also set the analog output signals to a constant value before the `EGMRun` instruction just for simplicity. Normally an external device will update the signal values to give the desired robot positions.

The second example below illustrates how a 7-axis robot can be used with EGM joint mode.

Example 1

```
MODULE EGM_test
VAR egmident egmID1;
VAR egmident egmID2;

CONST egm_minmax egm_minmax_lin1:=[-1,1];
CONST egm_minmax egm_minmax_rot1:=[-2,2];
CONST egm_minmax egm_minmax_joint1:=[-0.1,0.1];

CONST robtarget p20:=[[150,1320,1140],
    [0.000494947,0.662278,-0.749217,-0.00783173], [0,0,-1,0],
    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p30:=[[114.50,1005.42,1410.38],
    [0.322151,-0.601023,0.672381,0.287914], [0,0,-1,0],
    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST jointtarget
    jpos10:=[[0,0,0,0,35,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST pose posecor:=[[1200,400,900],[1,0,0,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
    [0.903899,-0.00320735,0.427666,0.00765917]];

! corr-frame: world, sens-frame: world
VAR pose posecor0:=[[0,0,0],[1,0,0,0]];
VAR pose posesen0:=[[0,0,0],[1,0,0,0]];

TASK PERS tooldata tFroniusCMT:=[TRUE,[[12.3313,-0.108707,416.142],
    [0.903899,-0.00320735,0.427666,0.00765917]],
    [2.6,[-111.1,24.6,386.6],[1,0,0,0],0,0,0.072]];
TASK PERS loaddata load1:=[5,[0,1,0],[1,0,0,0],0,0,0];
! corr-frame: wobj, sens-frame: wobj
```

Continues on next page

Application manual - Externally Guided Motion

```
TASK PERS wobjdata
     wobj_EGM1:=[FALSE,TRUE,"",[[150,1320,1140],[1,0,0,0]],
     [[0,0,0],[1,0,0,0]]];
VAR pose posecor1:=[[0,0,0],[1,0,0,0]];
VAR pose posesen1:=[[0,0,0],[1,0,0,0]];
TASK PERS wobjdata
     wobj_EGM2:=[FALSE,TRUE,"",[[0,1000,1000],[1,0,0,0]],
     [[0,0,0],[1,0,0,0]]];
VAR pose posecor2:=[[150,320,0],[1,0,0,0]];
VAR pose posesen2:=[[150,320,0],[1,0,0,0]];

PROC main()
MoveAbsJ jpos10\NoEOffs, v1000, fine, tFroniusCMT;
testAO;
ENDPROC

PROC testAO()
! Get two different EGM identities. They will be used for two
     different eGM setups.
EGMGetId egmID1;
EGMGetId egmID2;

! Set up the EGM data source: Analog output signals and
     configuration "default"
! One guidance using Pose mode and one using Joint mode
EGMSetupAO ROB_1,egmID1,"default" \Pose \aoR1x:=ao_MoveX
     \aoR2y:=ao_MoveY \aoR3z:=ao_MoveZ \aoR5ry:=ao_RotY
     \aoR6rz:=ao_RotZ;
EGMSetupAO ROB_1,egmID2,"default" \Joint \aoR1x:=ao_MoveX
     \aoR2y:=ao_MoveY \aoR3z:=ao_MoveZ \aoR4rx:=ao_RotX
     \aoR5ry:=ao_RotY \aoR6rz:=ao_RotZ;

! Move to the starting point - fine point is needed.
MoveJ p30, v1000, fine, tool0;
! Set the signals
SetAO ao_MoveX, 150;
SetAO ao_MoveY, 1320;
SetAO ao_MoveZ, 900;
! Correction frame is the World coordinate system and the sensor
     measurements are also relative to the world frame
! No offset is defined (posecor0 and posesen0)
EGMActPose egmID1 \Tool:=tFroniusCMT \WObj:=wobj0 \TLoad:=load1,
     posecor0, EGM_FRAME_WORLD, posesen0, EGM_FRAME_WORLD
     \x:=egm_minmax_lin1 \y:=egm_minmax_lin1 \z:=egm_minmax_lin1
     \rx:=egm_minmax_rot1 \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1
     \LpFilter:=20 \SampleRate:=16 \MaxPosDeviation:=1000;
! Run: keep the end position without returning to the start position
EGMRunPose egmID1,
     EGM_STOP_HOLD\x\y\z\RampInTime:=0.05\PosCorrGain:=1;

! Move to the starting point - fine point is needed.
MoveJ p20, v1000, fine, tFroniusCMT;
```

5.4.3 Using EGM Position Guidance with signals as input
Continued

```
                    ! Set the signals
                    SetAO ao_MoveX, 150;
                    SetAO ao_MoveY, 1320;
                    SetAO ao_MoveZ, 1100;
                    ! Run with the same frame definitions: ramp down to the start
                        position after having reached the EGM end position
                    EGMRunPose egmID1,
                        EGM_STOP_RAMP_DOWN\x\y\z\RampInTime:=0.05\PosCorrGain:=1;


                    ! Move to the starting point - fine point is needed.
                    MoveJ p30, v1000, fine, tool0;
                    ! Set the signals
                    SetAO ao_MoveX, 50;
                    SetAO ao_MoveY, -20;
                    SetAO ao_MoveZ, -20;
                    ! Correction frame is the Work object wobj_EGM1 and the sensor
                        measurements are also relative to the same work object. No
                        offset is defined (posecor1 and posesen1)
                    EGMActPose egmID1 \Tool:=tFroniusCMT \WObj:=wobj_EGM1 \TLoad:=load1,
                        posecor1, EGM_FRAME_WOBJ, posesen1, EGM_FRAME_WOBJ
                        \x:=egm_minmax_lin1 \y:=egm_minmax_lin1 \z:=egm_minmax_lin1
                        \rx:=egm_minmax_rot1 \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1
                        \LpFilter:=20;
                    ! Run: keep the end position without returning to the start position
                    EGMRunPose egmID1,
                        EGM_STOP_HOLD\x\y\z\RampInTime:=0.05\PosCorrGain:=1;


                    ! Move to the starting point - fine point is needed.
                    MoveJ p20, v1000, fine, tFroniusCMT;
                    ! Set the signals
                    SetAO ao_MoveX, 0;
                    SetAO ao_MoveY, 0;
                    SetAO ao_MoveZ, 0;
                    ! Correction frame is the Work object wobj_EGM2 and the sensor
                        measurements are also relative to the same work object. This
                        time an offset is defined for the correction frame (posecor2),
                        and for the sensor frame (posesen2)
                    EGMActPose egmID1 \Tool:=tFroniusCMT \WObj:=wobj_EGM2 \TLoad:=load1,
                        posecor2, EGM_FRAME_WOBJ, posesen2, EGM_FRAME_WOBJ
                        \x:=egm_minmax_lin1 \y:=egm_minmax_lin1 \z:=egm_minmax_lin1
                        \rx:=egm_minmax_rot1 \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1
                        \LpFilter:=20;
                    ! Run: keep the end position without returning to the start position
                    EGMRunPose egmID1,
                        EGM_STOP_HOLD\x\y\z\RampInTime:=0.05\PosCorrGain:=1;


                    ! Move to the starting point - fine point is needed.
                    MoveJ p20, v1000, fine, tFroniusCMT;
                    ! Set the signals
                    SetAO ao_MoveX, 0;
                    SetAO ao_MoveY, 0;
                    SetAO ao_MoveZ, 0;
```

```
! Correction frame is of tool type and the sensor measurements are
      relative to the work object wobj_EGM2. This time an offset
      is defined for the correction frame (posecor2), and for the
      sensor frame (posesen2)
EGMActPose egmID1 \Tool:=tFroniusCMT \WObj:=wobj_EGM2, posecor2,
      EGM_FRAME_TOOL, posesen2, EGM_FRAME_WOBJ \x:=egm_minmax_lin1
      \y:=egm_minmax_lin1 \z:=egm_minmax_lin1 \rx:=egm_minmax_rot1
      \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1 \LpFilter:=20;
EGMRunPose egmID1,
      EGM_STOP_HOLD\x\y\z\RampInTime:=0.05\PosCorrGain:=1;


! Move to the starting point - fine point is needed.
MoveJ p20, v1000, fine, tFroniusCMT\TLoad:=load1;
! Set the signals
SetAO ao_MoveX, 150;
SetAO ao_MoveY, 1320;
SetAO ao_MoveZ, 1100;
! Same as last, but with tool0 and wobj0
EGMActPose egmID1, posecor2, EGM_FRAME_TOOL, posesen2,
      EGM_FRAME_WOBJ \x:=egm_minmax_lin1 \y:=egm_minmax_lin1
      \z:=egm_minmax_lin1 \rx:=egm_minmax_rot1 \ry:=egm_minmax_rot1
      \rz:=egm_minmax_rot1 \LpFilter:=20;
! Run: keep the end position without returning to the start position
EGMRunPose egmID1,
      EGM_STOP_HOLD\x\y\z\RampInTime:=0.05\PosCorrGain:=1;


! Move to the starting point - fine point is needed.
MoveJ p20, v1000, fine, tFroniusCMT\TLoad:=load1;
! Set the signals
SetAO ao_MoveX, 70;
SetAO ao_MoveY, -5;
SetAO ao_MoveZ, 0;
SetAO ao_RotX, 0;
SetAO ao_RotY, 0;
SetAO ao_RotZ, 0;
! Joint guidance for joints 2-6
EGMActJoint egmID2 \J2:=egm_minmax_joint1 \J3:=egm_minmax_joint1
      \J4:=egm_minmax_joint1 \J5:=egm_minmax_joint1
      \J6:=egm_minmax_joint1 \LpFilter:=20;
! Run: keep the end position without returning to the start position
EGMRunJoint egmID2, EGM_STOP_HOLD \J2 \J3 \J4 \J5 \J6 \CondTime:=0.1
      \RampInTime:=0.05 \PosCorrGain:=1;


EGMReset egmID1;
EGMReset egmID2;
ENDPROC
ENDMODULE
```

Example 2

```
MODULE EGM_IRB14000_test
VAR egmident egmID;
CONST egm_minmax egm_minmax_joint1:=[-0.1,0.1];
! For handling if the test is used with left or right arm.
VAR jointtarget jpos10;
CONST jointtarget jpos10_L:=[[0,-130,30,0,40,0],
    [135,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST jointtarget jpos10_R:=[[0,-130,30,0,40,0],
    [-135,9E+09,9E+09,9E+09,9E+09,9E+09]];
PROC main()
  IF GetMecUnitName(ROB_ID) = "ROB_L" THEN
    jpos10 := jpos10_L;
    testAO;
  ELSEIF GetMecUnitName(ROB_ID) = "ROB_R" THEN
    jpos10 := jpos10_R;
    testAO;
  ENDIF
ENDPROC
PROC testAO()
  ! Get an EGM idenity.
  EGMGetId egmID;
  ! Set up the EGM data source:
  ! Analog output signals and configuration "default".
  ! Only the EGM Joint mode support IRB14000.
  ! Notice the joint mapping of the analog output signals.
  EGMSetupAO ROB_ID, egmID, "default" \Joint \aoR1x:=ao_J1
        \aoR2y:=ao_J2 \aoR3z:=ao_J4 \aoR4rx:=ao_J5 \aoR5ry:=ao_J6
        \aoR6rz:=ao_J7 \AoE1:=ao_J3;
  ! Move to the starting point - fine point is needed.
  MoveAbsJ jpos10\NoEOffs, v50, fine, tool0;
  ! Set the signals (using an incrementing offset from the initial
        position).
  ! Another set of analog signals should be created, if running
        this code for both arms at the same time.
  ! Notice the joint mapping from a jointtarget to the analog output
        signals.
  SetAO ao_J1, jpos10.robax.rax_1 + 1;
  SetAO ao_J2, jpos10.robax.rax_2 + 2;
  SetAO ao_J3, jpos10.extax.eax_a + 3;
  SetAO ao_J4, jpos10.robax.rax_3 + 4;
  SetAO ao_J5, jpos10.robax.rax_4 + 5;
  SetAO ao_J6, jpos10.robax.rax_5 + 6;
  SetAO ao_J7, jpos10.robax.rax_6 + 7;
  ! Joint guidance for joints 1-7.
  EGMActJoint egmID \J1:=egm_minmax_joint1 \J2:=egm_minmax_joint1
        \J3:=egm_minmax_joint1 \J4:=egm_minmax_joint1
        \J5:=egm_minmax_joint1 \J6:=egm_minmax_joint1
        \J7:=egm_minmax_joint1 \LpFilter:=20;
  ! Run: keep the end position without returning to the start
        position.
```

```
EGMRunJoint egmID, EGM_STOP_HOLD \J1 \J2 \J3 \J4 \J5 \J6 \J7
        \CondTime:=1 \RampInTime:=0.05 \PosCorrGain:=1;
    EGMReset egmID;
ENDPROC ENDMODULE
```

5.4.4 Using EGM Path Correction with different protocol types

Description

This example contains examples for different sensor and protocol types. The basic RAPID program structure is the same for all of them and they use the same external motion data configuration.

Example

```
MODULE EGM_PATHCORR
! Used tool
PERS tooldata tEGM:=[TRUE,[[148.62,0.25,326.31],
    [0.833900724,0,0.551914471,0]], [1,[0,0,100],
    [1,0,0,0],0,0,0]];
! Sensor tool, has to be calibrated
PERS tooldata
    tLaser:=[TRUE,[[148.619609537,50.250017146,326.310337954],
    [0.390261856,-0.58965743,-0.58965629,0.390263064]],
    [1,[-0.920483747,-0.000000536,-0.390780849],
    [1,0,0,0],0,0,0]];
! Displacement used
VAR pose PP:=[[0,-3,2],[1,0,0,0]];
VAR egmident egmId1;

! Protocol: LTAPP
! Example for a look ahead sensor, e.g. Laser Tracker
PROC Part_2_EGM_OT_Pth_1()
  EGMGetId egmId1;
  ! Set up the EGM data source: LTAPP server using device "Optsim",
      configuration "pathCorr", joint type 1 and look ahead
      sensor.
  EGMSetupLTAPP ROB_1, egmId1, "pathCorr", "OptSim", 1\LATR;
  ! Activate EGM and define the sensor frame. Correction frame is
      always the path frame.
  EGMActMove egmId1, tLaser.tframe\SampleRate:=48;
  ! Move to a suitable approach position.
  MoveJ p100,v1000,z10,tEGM\WObj:=wobj0;
  MoveL p110,v1000,z100,tEGM\WObj:=wobj0;
  MoveL p120,v1000,z100,tEGM\WObj:=wobj0;
  ! Activate displacement (not necessary but possible)
  PDispSet PP;
  ! Move to the start point. Fine point is demanded.
  MoveL p130, v10, fine, tEGM\WObj:=wobj0;
  ! movements with path corrections.
  EGMMoveL egmId1, p140, v10, z5, tEGM\WObj:=wobj0;
  EGMMoveL egmId1, p150, v10, z5, tEGM\WObj:=wobj0;
  EGMMoveC egmId1, p160, p165, v10, z5, tEGM\WObj:=wobj0;
  ! Last path correction movement has to end with a fine point.
  EGMMoveL egmId1, p170, v10, fine, tEGM\WObj:=wobj0;
  ! Move to a safe position after path correction.
  MoveL p180,v1000,z10,tEGM\WObj:=wobj0;
  ! Release the EGM identity for reuse.
```

Continues on next page

```
      EGMReset egmId1;
  ENDPROC


  ! Protocol: LTAPP
  ! Example for an at point sensor, e.g. Weldguide
  PROC Part_2_EGM_WG_Pth_1()
    EGMGetId egmId1;
    ! Set up the EGM data source: LTAPP server using device "wg1sim",
          configuration "pathCorr", joint type 1 and at point sensor.
    EGMSetupLTAPP ROB_1, egmId1, "pathCorr", "wg1sim", 1\APTR;
    ! Activate EGM and define the sensor frame, which is the tool
          frame for at point trackers.
    ! Correction frame is always the path frame.
    EGMActMove egmId1, tEGM.tframe\SampleRate:=48;
    ! Move to a suitable approach position.
    MoveJ p100,v1000,z10,tEGM\WObj:=wobj0;
    MoveL p110,v1000,z100,tEGM\WObj:=wobj0;
    MoveL p120,v1000,fine,tEGM\WObj:=wobj0;
    ! Activate displacement (not necessary but possible)
    PDispSet PP;
    ! Move to the start point. Fine point is demanded.
    MoveL p130, v10, fine, tEGM\WObj:=wobj0;
    ! movements with path corrections.
    EGMMoveL egmId1, p140, v10, z5, tEGM\WObj:=wobj0;
    EGMMoveL egmId1, p150, v10, z5, tEGM\WObj:=wobj0;
    EGMMoveC egmId1, p160, p165, v10, z5, tEGM\WObj:=wobj0;
    ! Last path correction movement has to end with a fine point.
    EGMMoveL egmId1, p170, v10, fine, tEGM\WObj:=wobj0;
    ! Move to a safe position after path correction.
    MoveL p180,v1000,z10,tEGM\WObj:=wobj0;
    ! Release the EGM identity for reuse.
    EGMReset egmId1;
  ENDPROC


  ! Protocol: UdpUc
  ! Example for an at point sensor, e.g. Weldguide
  PROC Part_2_EGM_UDPUC_Pth_1()
    EGMGetId egmId1;
    EGMSetupUC ROB_1, egmId1, "pathCorr", "UCdevice"\PathCorr\APTR;
    EGMActMove egmId1, tEGM.tframe\SampleRate:=48;
    ! Move to a suitable approach position.
    MoveJ p100,v1000,z10,tEGM\WObj:=wobj0;
    MoveL p110,v1000,z100,tEGM\WObj:=wobj0;
    MoveL p120,v1000,fine,tEGM\WObj:=wobj0;
    ! Activate displacement (not necessary but possible)
    PDispSet PP;
    ! Move to the start point. Fine point is demanded.
    MoveL p130, v10, fine, tEGM\WObj:=wobj0;
    ! movements with path corrections.
    EGMMoveL egmId1, p140, v10, z5, tEGM\WObj:=wobj0;
    EGMMoveL egmId1, p150, v10, z5, tEGM\WObj:=wobj0;
```

```
        EGMMoveC egmId1, p160, p165, v10, z5, tEGM\WObj:=wobj0;
        ! Last path correction movement has to end with a fine point.
        EGMMoveL egmId1, p170, v10, fine, tEGM\WObj:=wobj0;
        ! Move to a safe position after path correction.
        MoveL p180,v1000,z10,tEGM\WObj:=wobj0;
        ! Release the EGM identity for reuse.
        EGMReset egmId1;
    ENDPROC
    ENDMODULE
```

6 UdpUc code examples

File locations

The following code examples are available in the RobotWare distribution.

File	Description
egm-sensor.cs	Example using protobuf-csharp-port
egm-sensor.cpp	Example using Google protocol buffers C++
egm.proto	The *egm.proto* file defines the data contract between the robot and the sensor.

The files can be obtained from the PC or the robot controller.

- **In the RobotWare installation folder in RobotStudio:** *...\RobotPackages\ RobotWare_RPK_<version>\utility\Template\EGM*
- **On the IRC5 Controller:** *<SystemName>\PRODUCTS\<RobotWare_xx.xx.xxxx>\utility\Template\EGM*

ℹ Note

Navigate to the RobotWare installation folder from the RobotStudio **Add-Ins** tab, by right-clicking on the installed RobotWare version in the **Add-Ins** browser and selecting **Open Package Folder**.

This page is intentionally left blank

Index

ABB

ABB AB
Robotics & Discrete Automation
S-721 68 VÄSTERÅS, Sweden
Telephone +46 10-732 50 00

ABB AS
Robotics & Discrete Automation
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.
Robotics & Discrete Automation
No. 4528 Kangxin Highway
PuDong New District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

ABB Inc.
Robotics & Discrete Automation
1250 Brown Road
Auburn Hills, MI 48326
USA
Telephone: +1 248 391 9000

abb.com/robotics