ROBOTICS

Application manual

Force Control Standard

Trace back information:
Workspace 24D version a27
Checked in 2025-01-17
Skribenta version 5.6.018

Application manual
Force Control Standard

RobotWare 7.17

Document ID: 3HAC090251-001
Revision: B

Table of contents

Overview of this manual 9

1 Introduction 11

1.1 About Force Control 11
1.2 About FC basic control and assembly 13
1.3 About FC Pressure 14
1.4 About FC SpeedChange 15

2 Installation 17

2.1 Getting started with force control 17

3 Configuration 21

3.1 System parameters overview 21
3.2 Configuration example 25

4 Programming 29

4.1 RAPID components 29
4.1.1 Force controlled pressure applications 29
4.1.2 Force controlled speed change applications 31
4.1.3 Force controlled assembly applications 35
4.2 Code examples 40
4.2.1 Force controlled pressure applications 40
4.2.2 Force controlled speed change applications 41
4.2.3 Force controlled assembly applications 42

5 Execution behavior 45

5.1 Conflicting reference values 45
5.2 Damping and LP-filter 47
5.3 Overcome friction 50
5.4 Special cases 51
5.5 FC Press optimization 52
5.5.1 Use Spd FFW 52
5.6 FC SpeedChange control design 54
5.6.1 Controller scheme 54
5.6.2 Rule based logical control 56
5.6.3 Programming in path coordinates 58
5.6.4 Recovery routine 59

6 Troubleshooting 61

6.1 What to do when... 61

7 RAPID reference information 63

7.1 Instructions 63
7.1.1 FCAct 63
7.1.2 FCCalib 66
7.1.3 FCCondCombine 68
7.1.4 FCCondForce 69
7.1.5 FCCondOrient 72
7.1.6 FCCondPos 76
7.1.7 FCCondReoriSpeed 79
7.1.8 FCCondTCPSpeed 82
7.1.9 FCCondTorque 85
7.1.10 FCCondWaitWhile 88
7.1.11 FCDeact 90
7.1.12 FCPress1LStart 91

7.1.13 FCPressC 97
7.1.14 FCPressEnd 99
7.1.15 FCPressL 102
7.1.16 FCRefCircle 104
7.1.17 FCRefForce 106
7.1.18 FCRefLine 108
7.1.19 FCRefMoveFrame 110
7.1.20 FCRefRot 112
7.1.21 FCRefSpiral 114
7.1.22 FCRefSprForceCart 116
7.1.23 FCRefStart 119
7.1.24 FCRefStop 121
7.1.25 FCRefTorque 122
7.1.26 FCResetDampingTune 124
7.1.27 FCResetLPFilterTune 125
7.1.28 FCResetMaxForceChangeTune 126
7.1.29 FCSetDampingTune 127
7.1.30 FCSetLPFilterTune 129
7.1.31 FCSetMaxForceChangeTune 130
7.1.32 FCSpdChgAct 131
7.1.33 FCSpdChgDeact 134
7.1.34 FCSpdChgTunSet 135
7.1.35 FCSpdChgTunReset 137
7.1.36 FCSupvForce 139
7.1.37 FCSupvOrient 142
7.1.38 FCSupvPos 145
7.1.39 FCSupvReoriSpeed 147
7.1.40 FCSupvTCPSpeed 149
7.1.41 FCSupvTorque 151
7.2 Functions 154
7.2.1 FCGetForce 154
7.2.2 FCGetProcessData 156
7.2.3 FCIsForceMode 158
7.2.4 FCLoadID 159
7.3 Data types 163
7.3.1 fcboxvol 163
7.3.2 fccondstatus 165
7.3.3 fccylindervol 167
7.3.4 fcdamping 169
7.3.5 fcforcevector 171
7.3.6 fcframe 173
7.3.7 fclindir 174
7.3.8 fcprocessdata 175
7.3.9 fcplane 177
7.3.10 fcrotdir 178
7.3.11 fcspeedvector 179
7.3.12 fcspherevol 181
7.3.13 fcspdchgtunetype 182
7.3.14 fcxyznum 183

8 System parameters 185

8.1 Type *Robot* 185
8.1.1 Use FC Master 185
8.2 Type *FC Master* 186
8.2.1 The FC Master type 186
8.2.2 Name 187
8.2.3 Use FC Sensor 188
8.2.4 Use FC Kinematics 189
8.2.5 Use FC Application 190

8.2.6 Use FC Speed Change 191
8.3 Type *FC Sensor* 192
8.3.1 The FC Sensor type 192
8.3.2 Name 194
8.3.3 Force Sensor Mount Unit Name 195
8.3.4 Force Sensor Type 196
8.3.5 Force Sensor Frame x - z 197
8.3.6 Force Sensor Frame q1 - q4 198
8.3.7 Noise level 199
8.3.8 Force Sensor Scaling Factor fx, fy, fz, tx, ty, tz 200
8.4 Type *FC Kinematics* 201
8.4.1 The FC Kinematics type 201
8.4.2 Name 202
8.4.3 Damping in Force x Direction - Damping in Force z Direction 203
8.4.4 Damping in Torque x Direction - Damping in Torque z Direction 204
8.4.5 Bandwidth of force frame filter 205
8.4.6 Bandwidth of force loop filter 206
8.5 Type *FC Application* 207
8.5.1 The FC Application type 207
8.5.2 Name 208
8.5.3 Max Ref Force 209
8.5.4 Max Ref Force Change 210
8.5.5 Max Ref Torque 211
8.5.6 Max Ref Torque Change 212
8.5.7 Max Ref TCP Speed 213
8.5.8 Max Ref Rot Speed 214
8.5.9 Max Ref TCP Acc 215
8.5.10 Max Ref Rot Acc 216
8.5.11 Speed superv override 217
8.5.12 Largest measured contact force 218
8.5.13 Lowest measured contact force 219
8.5.14 Largest measured contact torque 220
8.5.15 Lowest measured contact torque 221
8.5.16 Keep contact force at stop 222
8.5.17 Max Press TCP Speed 223
8.5.18 Max Press Rot Speed 224
8.6 Type *FC Speed Change* 225
8.6.1 The FC Speed Change Type 225
8.6.2 Name 226
8.6.3 Speed ratio min 227
8.6.4 No of speed levels 228
8.6.5 Speed ratio delta 229
8.6.6 Speed max update period 230
8.6.7 Feedback type 231
8.6.8 Use Fdb LP filter 232
8.6.9 Fdb LP filter bandwidth 233
8.6.10 Maximum TCP speed 234
8.6.11 Recover rule fdb ratio 235
8.6.12 Decrease rule safety fdb ratio 236
8.6.13 Decrease rule safety fdb time 237
8.6.14 Fdb trend step size 238
8.6.15 Decrease rule 1 fdb ratio 239
8.6.16 Decrease rule 1 fdb trend 240
8.6.17 Decrease rule 2 fdb ratio 241
8.6.18 Decrease rule 2 fdb trend 242
8.6.19 Increase rule 1 fdb ratio 243
8.6.20 Increase rule 1 fdb trend 244
8.6.21 Increase rule 2 fdb ratio 245
8.6.22 Increase rule 2 fdb trend 246

9 Further references 247

9.1 TuneMaster 247
9.2 The coordinate systems 249
9.3 Force Sensor interface 252
9.3.1 About the Force Sensor interface 252
9.3.2 Sensors with less than six degrees of freedom 253
9.3.3 Room fixed sensor 254

Index 255

Overview of this manual

About this manual

This manual contains information about the RobotWare option Force Control Standard, for all robots using external force sensors. For GoFa robots, see *Application manual - Force control Standard for GoFa*.

Usage

This manual can be used to find out what Force control is and how to use it. It provides information about system parameters and RAPID components related to Force control, 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.
- be familiar with the RAPID programming language.
- be familiar with system parameters and how to configure them.

References

Reference	Document ID
Product manual - OmniCore C30	*3HAC060860-001*
Product manual - OmniCore C90XT	*3HAC073706-001*
Product manual - OmniCore V250XT Type B	*3HAC087112-001*
Product manual - OmniCore V400XT	*3HAC081697-001*
Product specification - OmniCore C line	*3HAC065034-001*
Product specification - OmniCore V line	*3HAC074671-001*
Application manual - TuneMaster	*3HAC063590-001*
Operating manual - OmniCore	*3HAC065036-001*
Operating manual - RobotStudio	*3HAC032104-001*
Technical reference manual - RAPID Overview	*3HAC065040-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC065038-001*
Technical reference manual - System parameters	*3HAC065041-001*

Revisions

Revision	Description
A	Released with RobotWare 7.15.
B	Released with RobotWare 7.17. • Removed information about MultiMove, as it cannot be combined with Force Control Standard.

This page is intentionally left blank

1 Introduction

1.1 About Force Control

Purpose

The purpose of Force Control is to make the robot sensitive to contact forces. The result is that the robot can "feel" its surroundings. It can apply a constant force on a surface, even if the exact position of the surface is not known.

Here are some examples from powertrain assembly applications where force control is useful:

- Piston assembly
- Forward clutch hub assembly
- Torque converter assembly

Force Control can also be used for different kinds of material removal processes, surface finishing and surface processing. Where the robot can hold the tool and work on a fixed part or hold the part and work on a fixed tool.

Here are some examples of machining applications where force control is useful:

- Grinding
- Milling
- Polishing
- Deburring
- Deflashing
- etc.

What is included

The option Force Control Standard gives access to:

- FC basic control and assembly, for more information see *About FC basic control and assembly on page 13*.
- FC Pressure, for more information see *About FC Pressure on page 14*.
- FC SpeedChange, for more information see *About FC SpeedChange on page 15*.

Limitations

- The total load, that is the sum of gravitational forces and external contact forces, must not exceed limits as specified in the load diagrams for a specific robot.

When the robot is force controlled, the following functionality is *not* accessible:

- *Arc*
- *Collision Detection*
- *Conveyor tracking*
- *Independent axes*
- Joint soft servo (instruction `SoftAct`)

Continues on next page

- *Path Offset*
- *PickMaster*
- RAPID instructions such as `FCAct`, `FCDeact`, `FCConditionWaitWhile`, and `FCRefStop` can only be called from normal level in a motion task.
- *Sensor* or *Analog synchronization*
- *Sensor interface*
- *SoftMove*
- Tracking functionality
- Force controlled pressure applications (FCPressL etc) and Force controlled speed change applications (FCSpdChgAct etc) cannot be combined with EGM instructions.
- *World Zones*

1.2 About FC basic control and assembly

Purpose

The purpose of FC basic control and assembly is to make the robot sensitive to contact forces. The robot can "feel" its surroundings, react, and obtain a certain pressure against an object. This means that the robot will change its position in order to fulfil the commanded force instruction. This is useful in testing application, and all kinds of insertion applications.

What is included

FC basic control and assembly gives access to:

- instruction to set up gravity compensation and sensor offset calibration.
- instructions for activation and deactivation of Force Control.
- instructions for defining reference values (desired force, torque or movement).
- instructions for end conditions.
- instructions for supervision.
- functions returning information about load, detected forces, or process status.
- data types supporting the instructions and functions.

Basic approach

A RAPID program using Force Control basically follows these steps. For a more detailed example of how this is done, see *Code examples on page 40*.

1. Identify the load and calibrate the system
2. Set up desired force and movement pattern.
3. Set up end condition.
4. Activate force control.
5. Activate force and movement pattern.
6. Wait for end condition to occur.
7. Deactivate force and movement patterns.
8. Deactivate force control.

1.3 About FC Pressure

Purpose

The purpose of FC Pressure is to make the robot sensitive to contact forces. The robot can "feel" its surroundings and follow the surface of the processed part to obtain a certain pressure against an object. This means that the robot will change its position in order to apply a constant force/pressure on a surface, even if the exact position of the surface is not known.

Since pressure is obtained by moving the robot path, this function is suited for polishing, grinding, and cleaning, where a surface should be made even and smooth. The material that is removed and the changes of the surface topology/dimensions depends on the process parameters like tooling, applied pressure, robot speed etc.

Here are some examples from foundry and metal fabrication where FC Pressure is useful:

- Grinding of faucets
- Polishing of kitchen sinks
- Deflashing, grinding and cleaning of castings
- Deburring of castings
- etc

For the best quality results it is recommended to use the option deflection compensation. Since the robot arm is deflecting under high contact forces there is the option to compensate for that deflection and follow the programmed path more optimally.

What is included

The function FC Pressure gives you access to:

- Instructions for programming FC Pressure start, movements and stop.
- Option to use deflection compensation in connection with force applications.

What is needed

FC Pressure requires a sensor input from the measured process forces to adjust the behavior of the robot. Depending of the application and required flexibility different force/torque sensors can be used.

For applications with the function FC Pressure, use

- 6 DOF, for more flexible solutions.

Basic approach

1. Identify the load and calibrate the system.
2. Move to a point close to contact.
3. Set up desired force and start movement towards the surface.
4. Move linear or circular performing the process with contact.
5. Leave surface and deactivate force control.

1.4 About FC SpeedChange

Purpose

In processes where path accuracy is important and where the finished result shall comply with specific dimensions, FC SpeedChange is recommended. This function will be useful combined with force sensor or other input indicating excessive process forces, which can deteriorate the finished result. When FC SpeedChange is active and if machining forces exceed a certain value, then the path speed will automatically be reduced, thus decreasing forces, minimizing changed dimensions due to deflections of the robot arm and most probably avoid damaging the part/tool due to stress and heat. This will guarantee path accuracy even if much material shall be removed.

See below some examples from foundry and metal fabrication where FC SpeedChange is useful:

- Grinding unevenly distributed material on casted surfaces
- Milling along the edge of a work piece
- Deburring along contour of a work piece
- Deflashing unevenly distributed burr along a part line on castings
- Deburring of castings
- etc.

What is included

The function FC SpeedChange gives you access to

- Instructions for programming FC SpeedChange.
- Instructions for defining a recovery function for FC SpeedChange.

What is needed

FC SpeedChange requires a sensor input from the measured process forces to adjust the behavior of the robot. Depending of the application and required flexibility different sensors can be used.

For applications with the function FC SpeedChange, use:

- 6 DOF force/torque, for more flexible solutions.

Basic approach

1. Configure the parameters for FC SpeedChange controller.
2. Identify the load and calibrate the sensor
3. Define recovery function.
4. Activate FC SpeedChange.
5. Execute the machining task; move along the path performing the process.
6. Deactivate FC SpeedChange.

2 Installation

2.1 Getting started with force control

Overview

This chapter describes the basic steps to get started with Force Control, from mounting the sensor to writing the first program. This manual only describes what is specific for a Force Control installation. For more information about installation and commissioning of the controller, see the product manual for the controller.

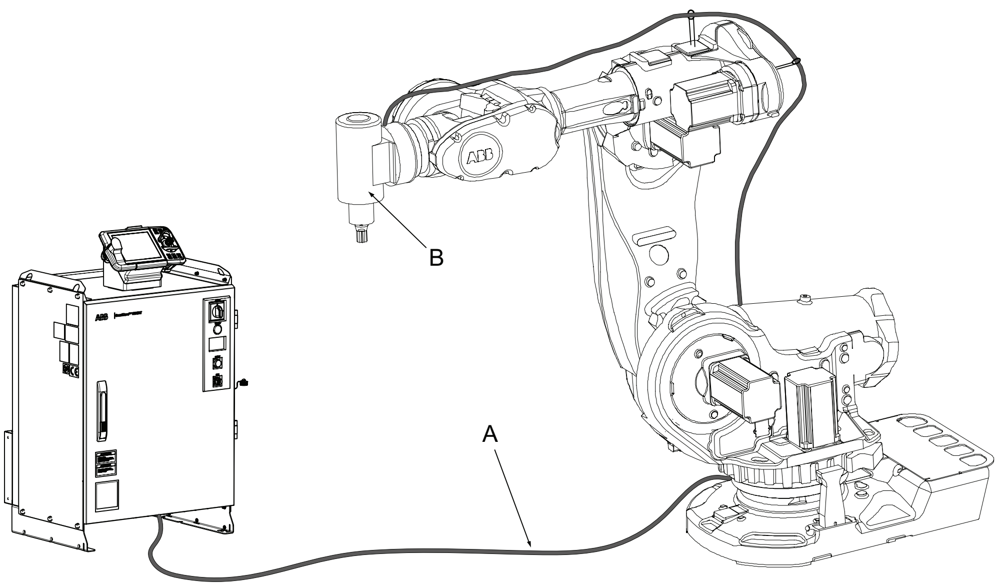
Hardware using a force sensor

The following hardware items are needed for Force Control:

A Cable between force sensor and the robot controller

B Robot mounted or room fixed force sensor

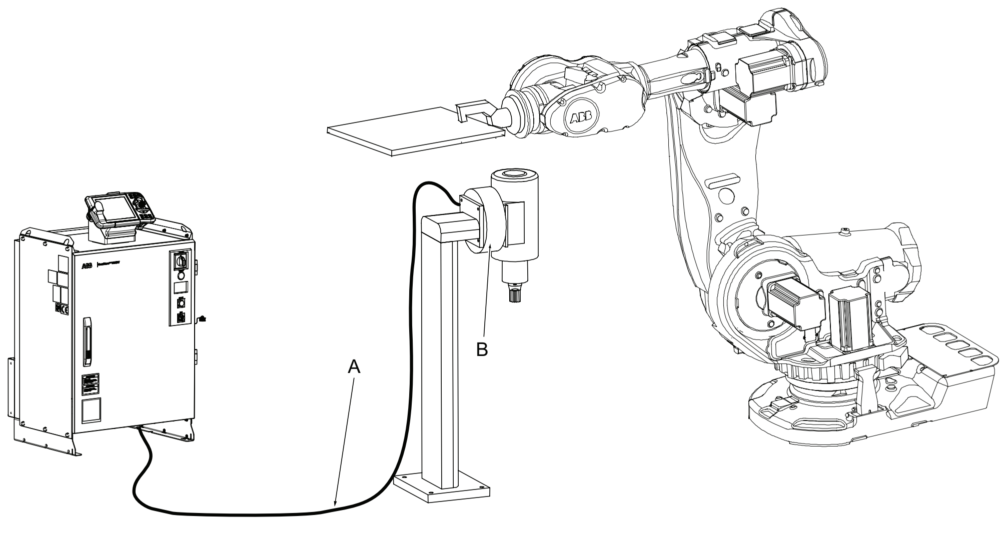
Robot mounted sensor



xx2300001761

Continues on next page

Room fixed sensor



xx2300001762

Force sensor and cable

ABB has integrated support for using the ABB force sensor, or force sensors from ATI Industrial Automation, which includes adapter plate and calibration parameter file for easy integration, but other sensor supplier can also be used. For further details see *About the Force Sensor interface on page 252*.

Note

For recommended 6 DOF force sensor see the product specification for the robot controller.

UL approved

For an UL approved solution, it is required that:

- the force sensor must be UL marked

Robot mounted ABB sensor

Use this procedure when mounting an ABB sensor on the robot. Mounting instructions are delivered with the sensor.

	Action	Note/illustration
1	Jog the robot to the zero position.	
2	Insert the position pin and ring in the robot flange.	Make sure the adapter plate and the sensor are not rotated by mistake. The orientation of the sensor is the same as tool0.
3	Mount the adapter on the flange and fix it with screws.	Make sure that the adapter is tightly fixed.
4	Insert the position pin for the sensor.	
5	Mount the force sensor on the adapter and fix it with screws. Rotate axis 6 when needed.	Make sure that the sensor is tightly fixed.

Continues on next page

Basic setup

	Action	Note/illustration
1	Mount the force sensor fixed in the room or on the robot's mounting flange. Sometimes adapter plates are needed between robot and sensor.	For non-ABB sensors, see the manual from the sensor supplier. **Note** Make sure that the adapter and the sensor are tightly fixed, and that the orientation is correct.
2	Connect the cable from the sensor to the robot controller.	See the product manual for the robot controller.
3	Connect the etherCAT cable to the robot controller.	• Make sure to position the cable on the robot so that it is not damaged by the movements of the robot.
4	Connect a pc with RobotStudio and RobotWare to the robot controller.	See *Operating manual - RobotStudio*.
5	Modify the system in RobotStudio. Force Control parameters are set in the configuration topic *Motion*.	Some of the Force Control parameters are already set. However, make sure they work for your application.
6	For non-ABB sensor/signal configuration, sensor calibration data needs to be supplied by the manufacturer. Right click the *Configuration* node in RobotStudio, select **Load Parameters** and then **Load parameters and replace duplicates**.	See also *System parameters overview on page 21* and *Configuration example on page 25*.
7	Now that the system is configured the last step is to program the application. To get started more easily there are some basic code examples on how to use Force control. There is also a RAPID component overview for easy usage.	*RAPID components on page 29* *Code examples on page 40*

General guidelines for Force Control

These guidelines can be useful even if you are an advanced programmer in Force Control and it is recommended to go through these steps from time to time.

1. Force sensor calibration is required prior to any operation with force control enabled.
2. Jogging the robot in force control is possible but all teaching and jogging in force control mode should be done with extreme caution.
3. Avoid programming and jogging through singularities, see *Technical reference manual - RAPID Overview*, section *Singularities*.
4. Always change damping parameters carefully and in small steps.
5. Always change bandwidth of force filter carefully and in small steps.
6. If you want to be sure to limit reference forces and movement speed, change the maximum parameters in the configuration to a desired value to avoid mistakes in the program.
7. Use the supervision instructions to limit the movement of the robot.

Continues on next page

8. Avoid deactivating Force Control while in contact. Remember that force control is deactivated when the program pointer is moved.
9. In case the program is interrupted, always start the program from the beginning.

Programming guidelines for Force Control

These guidelines can be useful even if you are an advanced programmer in Force Control and it is recommended to go through these steps from time to time.

1. Even if Force Control will change the path to obtain the contact reference, it's always best to have an initial programming of the path as close to the correct surface as possible.
2. Always try your new Force Control instructions with small reference without contact to any object to verify the movements.
3. Make sure that the Force Control start and end positions are not in contact with the work piece, but as close as possible.
4. Always use a lower speed and a smaller reference than intended when trying out the new movements.
5. Log the test signal for the spindle current/torque at normal operation in order to identify the correct, desirable signal level to be use in the SpeedChange controller.

3 Configuration

3.1 System parameters overview

About this section

This is an overview of the system parameters used in Force Control. For more information, see the respective parameter in section *System parameters on page 185*.

Robot

These parameters belong to the type *Robot* in the topic *Motion*.

Parameter	Description
Use FC Master	Specifies which *FC Master* to use.

FC Master

These parameters belong to the type *FC Master* in the topic *Motion*.

Parameter	Description
Name	The name of the *FC Master*.
Use FC Sensor	Specifies which *FC Sensor* to use.
Use FC Kinematics	Specifies which *FC Kinematics* to use.
Use FC Application	Specifies which *FC Application* to use.
Use FC Speed Change	Specifies which *FC Speed Change* to use.

FC Sensor

These parameters belong to the type *FC Sensor* in the topic *Motion*.

Parameter	Description
Name	The name of the *FC Sensor*.
Force Sensor Mount Unit Name	The name of the mechanical unit the sensor is mounted on. Most often ROB_1. If the sensor is room fixed the value should be empty.
Force Sensor Type	Specifies if the sensor should measure both force and torque or only force.
Noise level	The highest noise level at which a force sensor calibration is allowed. Used to check that the robot is standing still.
Force Sensor Frame x - z	The origin of the sensor coordinate system, expressed in the tool0 coordinate system when the sensor is mounted on the robot, or expressed in the world coordinate system when the sensor is room fixed. The unit is meter.
Force Sensor Frame q1 - q4	The orientation of the sensor coordinate system in relation to the tool0 coordinate system when the sensor is mounted on the robot, or to the world coordinate system when the sensor is room fixed.

Continues on next page

FC Kinematics

These parameters belong to the type *FC Kinematics* in the topic *Motion*.

Parameter	Description
Name	The name of the *FC Kinematics*.
Damping in Force x Direction Damping in Force y Direction Damping in Force z Direction	Specifies how the robot speed depends on the contact force. A higher value makes the robot less sensitive to contact forces. Different damping can be used for different directions. The unit is Ns/m.
Damping in Torque x Direction Damping in Torque y Direction Damping in Torque z Direction	Specifies how the tool reorientation speed depends on the torque induced by the environment. A higher value makes the robot less sensitive to contact torques. Different damping can be used for different directions. The unit is Nms/rad.
Bandwidth of force loop filter	Specifies the behavior of the force control loop. A higher value makes the robot more compliant but can cause instability. The unit is Hz.
Bandwidth of force frame filter	The force measured in "force frame" will be low pass filtered with this bandwidth. The unit is Hz.

FC Application

These parameters belong to the type *FC Application* in the topic *Motion*.

Parameter	Description
Name	The name of the *FC Application*.
Max Ref Force	Maximum allowed reference force. The unit is N.
Max Ref Force Change	Maximum allowed change in reference force. The unit is N/s.
Max Ref Torque	Maximum allowed reference torque. The unit is Nm.
Max Ref Torque Change	Maximum allowed change in reference torque. The unit is Nm/s.
Max Ref TCP Speed	Maximum allowed reference speed. The unit m/s.
Max Ref Rot Speed	Maximum allowed reference rotational speed. The unit is rad/s.
Max Ref TCP Acc	Maximum allowed reference acceleration. The unit is m/s^2
Max Ref Rot Acc	Maximum allowed reference rotational acceleration. The unit is rad/s^2.
Speed superv override	Overrides the speed supervision with a factor.
Largest measured contact force	If measured contact force is larger than this value it is set to this value.
Lowest measured contact force	A measured contact force lower than this value will be set to zero [dead band]. The unit is N.
Largest measured contact torque	If measured contact torque is larger than this value it is set to this value.
Lowest measured contact torque	A measured contact torque lower than this value will be set to zero [dead band]. The unit is Nm.
Max Press TCP Speed	Maximum linear speed in press movements. The unit is m/s.
Max Press Rot Speed	Maximum rotation speed in press movements. The unit is rad/s.
Keep contact force at stop	Defines whether the robot should be allowed to remain in contact when force control execution is stopped.

Continues on next page

Note

The values of the *Max Ref xxx* parameters define the ramping step of the reference movement. If the parameter value is set too high, the ramping will produce jerks and trig the speed supervision.

Note

If the parameters *Lowest measured contact force* and *Lowest measured contact torque* are set too low, there is a risk that the robot will drift when in force control mode.

FC Speed Change

These parameters belong to the type *FC Speed Change* in the topic *Motion*.

Parameter	Description
Name	Defines the name of the *FC Speed Change*.
Speed ratio min	Specifies the minimum allowed speed ratio.
No of speed levels	Defines the number of speed levels.
Speed ratio delta	Limits acceleration/deceleration due to the SpeedChange functionality. A low value will give slower but smoother speed changes. Too high value of Speed_ratio_delta will result in jerky behavior.
Speed max update period	Specifies the minimum time in seconds between speed changes.
Feedback type	Defines the type of feedback to be used for speed change control.
Use Fdb LP filter	Defines whether feedback low pass filter should be active.
Fdb LP filter bandwidth	Defines the bandwidth of the feedback low pass filter (Hz).
Maximum TCP speed	Defines the maximum original TCP speed for speed change. The unit is m/s.
Recover rule fdb ratio	A feedback to reference ratio larger than this while having reduced speed to lowest level will trig recover behavior or stop robot.
Decrease rule safety fdb ratio	Defines the maximum feedback to reference ratio.
Decrease rule safety fdb time	Define the maximum time in seconds that the feedback to reference ratio can be continuously over *Decrease rule safety fdb ratio* before reducing robot speed.
Fdb trend step size	Defines the minimum difference between two consecutive feedback values to count as a change in feedback.
Decrease rule 1 fdb ratio	For ABB internal use only.
Decrease rule 1 fdb trend	For ABB internal use only.
Decrease rule 2 fdb ratio	For ABB internal use only.
Decrease rule 2 fdb trend	For ABB internal use only.
Increase rule 1 fdb ratio	For ABB internal use only.
Increase rule 1 fdb trend	For ABB internal use only.

Continues on next page

Parameter	Description
Increase rule 2 fdb ratio	For ABB internal use only.
Increase rule 2 fdb trend	For ABB internal use only.

3.2 Configuration example

Overview

This section shows a real configuration example, intended to facilitate the setup of the configuration parameters. Some parameters are sensor specific, but others can be copied, like maximum allowed values.

Robot

Parameter	Value	Unit/Note
Use FC Master	fc_master1	-

FC Master

Parameter	Value	Unit/Note
Name	fc_master1	-
Use FC Sensor	fc_sensor1	-
Use FC Kinematics	fc_kinematics1	-
Use FC Application	fc_application1	-
Use FC Speed Change	fc_speed_change1	-

FC Sensor

Parameter	Value	Unit/Note
Name	fc_sensor1	-
Force Sensor Type	Force and Torque	6DOF
Mount Unit Name	ROB_1	Sensor mounted on robot 1
Noise level	25	-
Force Sensor Frame x	0	m
Force Sensor Frame y	0	m
Force Sensor Frame z	0.05	m
Force Sensor Frame q1	1	-
Force Sensor Frame q2	0	-
Force Sensor Frame q3	0	-
Force Sensor Frame q4	0	-

FC Kinematics

Parameter	Value	Unit/Note
Name	fc_kinematics1	-
Bandwidth of force frame filter	25	Hz
Bandwidth of force loop filter	3	Hz
Damping in Force x Direction	3000	Ns/m

Continues on next page

Parameter	Value	Unit/Note
Damping in Force y Direction	3000	Ns/m
Damping in Force z Direction	3000	Ns/m
Damping in Torque x Direction	400	Nms/rad
Damping in Torque y Direction	400	Nms/rad
Damping in Torque z Direction	400	Nms/rad

FC Application

Parameter	Value	Unit/Note
Name	fc_application1	-
Max Ref Force	1000	N
Max Ref Force Change	1000	N/s
Max ref Torque	400	Nm
Max Ref Torque Change	200	Nm/s
Max Ref TCP Speed	5	m/s
Max Ref Rot Speed	5	rad/s
Max Ref TCP Acc	1	m/s^2
Max Ref Rot Acc	1	rad/s^2
Speed Superv Override	3	-
Largest measured contact force	1000	N
Lowest measured contact force	3	N
Largest measured contact torque	400	Nm
Lowest measured contact torque	1	Nm
Max Press TCP Speed	5	m/s
Max Press Rot speed	5	rad/s

FC Speed Change

Parameter	Value	Unit/Note
Name	fc_speed_change1	-
Speed ratio min	0.1	-
No of speed levels	2	-
Speed ratio delta	0.07	-
Speed max update period	0.08	s
Feedback type	Calib. Force Magn.	
Use Fdb LP filter	Yes	-
Fdb LP filter Bandwidth	30	Hz
Maximum TCP speed	0.3	m/s
Recover rule fdb ratio	1.3	-

Continues on next page

Parameter	Value	Unit/Note
Decrease rule safety fdb ratio	1.5	-
Decrease rule safety fdb time	0.1	s
Fdb trend step size	8	-
Decrease rule1 fdb ration	0.7	-
...	...	...
Increase rule 2 fdb trend	10	-

This page is intentionally left blank

4 Programming

4.1 RAPID components

4.1.1 Force controlled pressure applications

Overview

These are the instructions used to start stop and run force controlled pressure applications.

Between the start and end instructions any combination and number of `FCPressL` and `FCPressC` can be used. The speed, force and zone may be changed for a new instruction allowing the process to be changed along with the properties of the application.

Pressure instructions

Instruction	Description
`FCPress1LStart`	Activates Force Control, starts movement and defines the data needed for the process below such as: • Movement: `ToPoint`, `Speed`, zonedata `zone`, tool `WObj` • Force settings: direction of force, Force Threshold to start `movementForceFrameRef` i.e `Wobj`
`FCPressL`	Moves linear to robtarget with a force in the direction setup by `FCPress1LStart`. Magnitude of the force can be changed for every `FCPressL`.
`FCPressC`	Moves circular to robtarget with a force in the direction setup by `FCPress1LStart`. Magnitude of the force can be changed for every `FCPressC`.
`FCPressEnd`	Leaves surface and moves to robtarget.

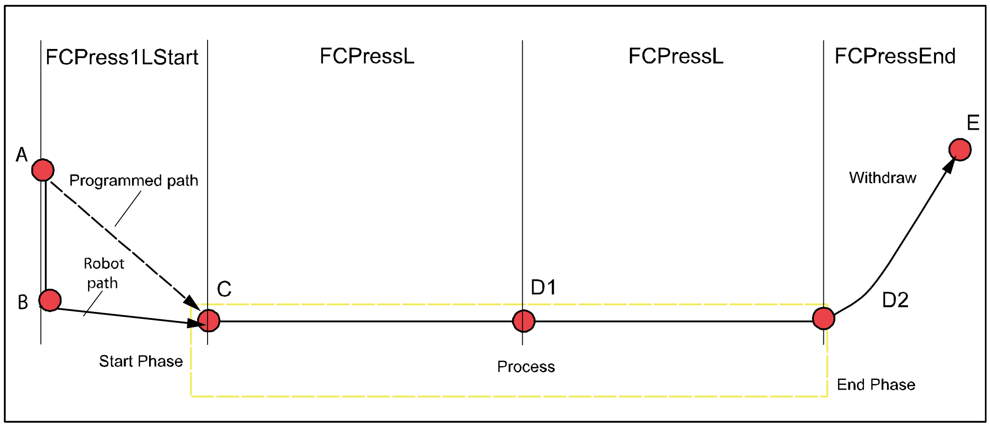
Calibrate the sensor

Instruction	Description
`FCLoadID`	Identifies the load measured by the force sensor. The identified load is used to calibrate the force sensor.
`FCCalib`	Calibrates the force sensor to remove sensor offset and compensate for gravity. Note that the calibration requires a precise definition of the load. Therefore, use the function `FCLoadID` before `FCCalib`.

Continues on next page

The phases

The figure below describes the phases for the FC pressure application.



xx0600003195

1 During the start phase the robot will switch to force control mode and move in the direction of the reference force in order to search contact with the work piece (B). Once contact is achieved the robot will start the movement towards the programmed position (C).
2 During the process any number and combination of `FCPressL` and `FCPressC` may be used in order to run the application process (D1) (D2)...(Dx).
3 After the last process movement the robot will retract from the work piece in the opposite direction of the reference force until zero contact force is measured. At this time the robot will switch to position control and move to the end position (E).

4.1.2 Force controlled speed change applications

Overview

The force controlled *SpeedChange* function will automatically slow down the robot speed based on the process force information (measured by force sensor, spindle motor current, etc.). The robot will slow down when the process force level raises above a defined threshold. After the process force reduces below a certain level, the robot will automatically regain its programmed speed. Between activation and deactivation any standard move instruction can be used.

SpeedChange instructions

These are the instructions used to activate and deactivate force controlled SpeedChange:

Instruction	Description
`FCSpdChgAct`	Activates force controlled SpeedChange function with following parameters: • Reference (force, spindle motor current, etc.), reduce robot speed when the measured signal is greater than the reference. • Recover function name (RAPID routine with the specified name will be called when certain condition satisfies. • Different recover behaviors (e.g., *MultipleRecover*, *NonStopAllTime*).
`FCSpdChgDeact`	Deactivates force controlled SpeedChange function.

These are instructions used to tune online configuration parameters for speed change function.

Instruction	Description
`FCSpdChgTunSet`	Change configuration parameter to a new value, with following input arguments • Configuration parameter type • New valid value
`FCSpdChgTunReset`	Restore configuration parameter to its original value that stored in configuration file • Configuration parameter type

Calibrate the sensor

This instruction is used to calculate the gravity and center of gravity for the current load. `FCLoadId` and `FCCalib` are required before the sensor can be used for Force Control.

Instruction	Description
`FCCalib`	Calibrates the force sensor to remove sensor offset and compensate for gravity. Note that the calibration requires a precise definition of the load. Therefore, use the function `FCLoadID` before `FCCalib`.

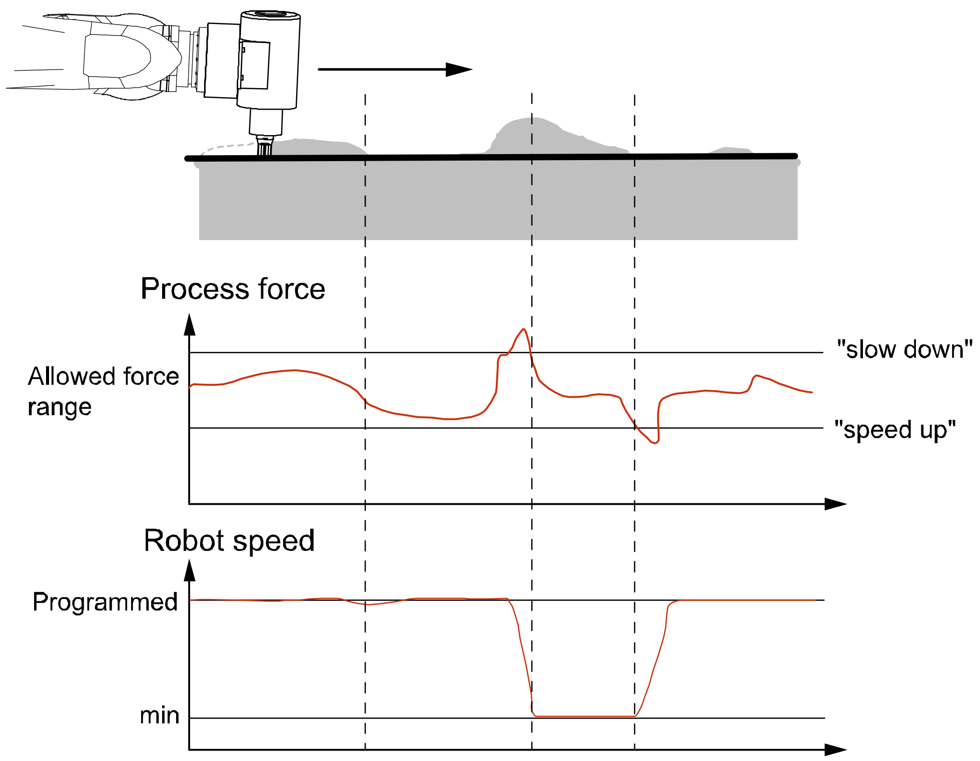
Note

`FCCalib` is only used for SpeedChange with force/torque sensor.

Continues on next page

Example

The figure below illustrates how the robot speed is adapting to keep the process force within allowed force range. For more information, see *How does it work? on page 33*.

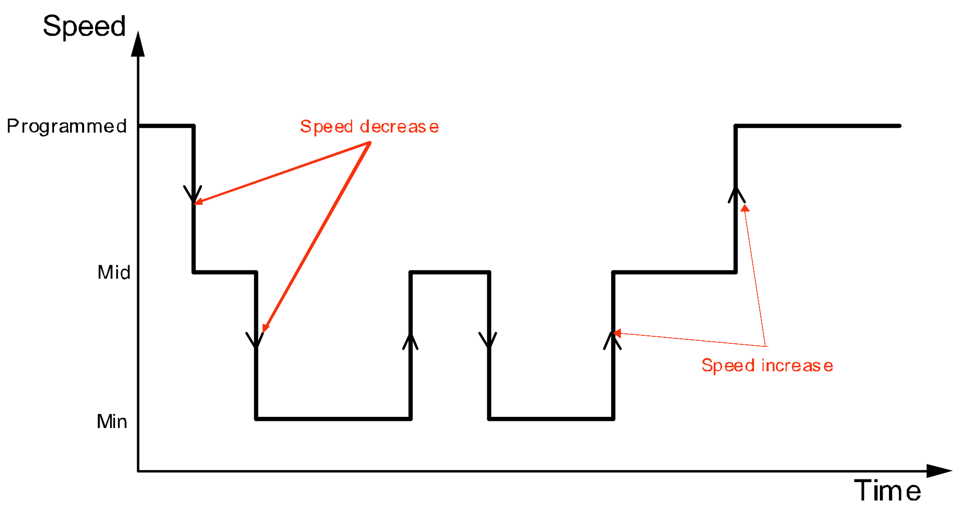


en0600003356

Continues on next page

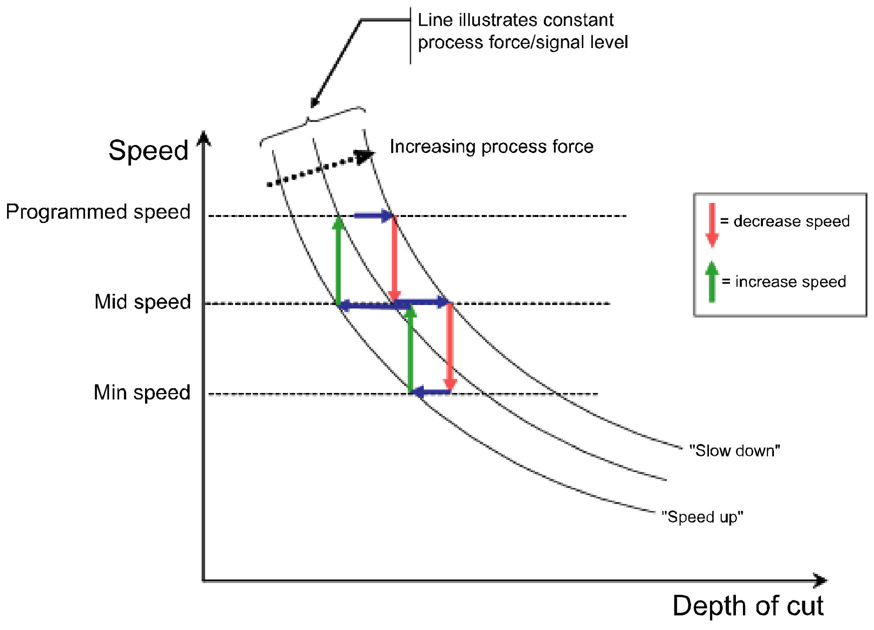
How does it work?

The control function for change the speed of the robot is "rule based" and include discrete speed levels in-between which the robot speed is changed. The number of speed levels can be defined using the parameter *No of speed levels*. Below is an example showing a process using 3 speed levels. When the process forces increase the speed is reduced and vice a verse:



en0600003275

The controller function is illustrated in the picture below:



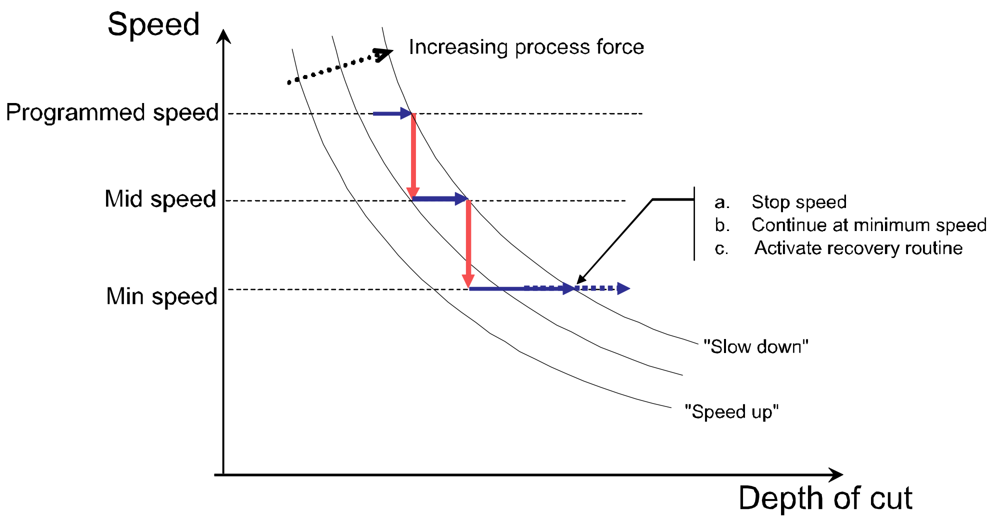
en0600003276

Continues on next page

If changes in process will appear suddenly and a short response time for SpeedChange is urgently required one shall consider to use a 2 speed level solution. This will result in the quickest speed reduction. Applications with slowly changing process forces will gain cycle time using a multiple speed level solution, but number of speed levels above 4 should be used with care.

For situations where not even the minimum speed will reduce the process forces below the "slow down" level there are 3 optional behaviors to choose between.

1. When reaching the "slow down" level simply stop the robot speed.
2. When reaching the "slow down" level continue with minimum speed.
3. When reaching the "slow down" level activate a recovery routine. Recovery routine to be defined by the program designer and shall include a back up procedure to eliminate the cause of the exceeded process forces. After executing the recovery routine the robot will continue on the original path. If the "slow down" level is reached during the recovery routine the robot will stop.



en0600003277

4.1.3 Force controlled assembly applications

About this section

This section presents an overview of the RAPID components, i.e. instructions, functions and data types, used in force controlled assembly applications. For more detailed information, such as the syntax, see *RAPID reference information on page 63*.

Note

The RAPID instructions for Force Control can be found in the **MotionAdv.** pick list when programming using the FlexPendant or RobotStudio.

Instructions for sensor calibration

`FCLoadId` is used to estimate the mass and center of gravity of the current load as measured by the force sensor. `FCCalib` is used to calibrate the system, and is required before the sensor can be used for force control.

Instruction	Description
`FCLoadID`	Identifies the load measured by the force sensor. The identified load is used to calibrate the force sensor.
`FCCalib`	Calibrates the force sensor to remove sensor offset and compensate for gravity. Note that the calibration requires a precise definition of the load. Therefore, `FCLoadID` should be used before `FCCalib`.

Instructions for activation/deactivation of Force Control

These are the instructions used to activate and deactivate Force Control:

Instruction	Description
`FCAct`	Activates Force Control. `FCAct` activates Force Control and defines the parameters below: • force control coordinate system (used e.g. for damping of Force Control). • damping (i.e. sensitivity to contact forces in different directions in the force control coordinate system).
`FCDeact`	Deactivates Force Control.

Instructions for force, torque and movement references

Force, torque and movement references are normally used to search for a fit.

When activating a force or a torque reference, the robot will attempt to maintain the specified force. When activating a movement reference, the robot will attempt to move according to the specified movement pattern.

Continues on next page

These are the instructions used to handle force, torque and movement references:

Instruction	Description
`FCRefMoveFrame`	Sets up a reference movement coordinate system, i.e. a coordinate system in which reference movement can be defined. The origin of this coordinate system is always the tool center point, but the directions can be defined. If no reference movement coordinate system is specified, the directions of the work object coordinate system will be used. If no work object is used either, the directions of the world coordinate system will be used.
`FCRefForce`	Specifies a reference force (size and direction) that the robot will try to maintain. The unit is N. The reference force is activated by `FCRefStart`.
`FCRefTorque`	Specifies a reference torque (size and direction) that the robot will try to maintain. The unit is Nm. The reference torque is activated by `FCRefStart`.
`FCRefSprForceCart`	Specifies a position dependent reference force. The size of the force reference will increase with the distance from a specified position.
`FCRefSpiral`	Specifies a reference movement. The robot TCP tries to make a spiral movement with larger and larger circles (and then smaller and smaller). If a contact force affects the robot, e.g. something is blocking the path, the movement will deviate from the intended path. The reference movement is activated by `FCRefStart`.
`FCRefCircle`	Specifies a reference movement. The robot TCP tries to make a circle movement. If a contact force affects the robot, e.g. something is blocking the path, the movement will deviate from the intended path. The reference movement is activated by `FCRefStart`.
`FCRefLine`	Specifies a reference movement. The robot tries to move back and forth along a line. If a contact force affects the robot, e.g. something is blocking the path, the movement will deviate from the intended path. The reference movement is activated by `FCRefStart`.
`FCRefRot`	Specifies a reference movement. The robot tries to rotate back and forth. If a contact force affects the robot, e.g. something is blocking the path, the movement will deviate from the intended path. The reference movement is activated by `FCRefStart`.
`FCRefStart`	Activates reference force, torque and movement. The robot starts to move in order to achieve the specified reference values.
`FCRefStop`	Deactivates reference force, torque and movement. The reference values can also be deactivated by conditions (see *Instructions for end conditions on page 37*).

Continues on next page

Instructions for end conditions

Force, torque and movement references are normally used to search for a fit. End conditions are used to determine when such a search has succeeded. The search will continue with the specified reference values as long as the condition is true. When the condition turns false, the end condition is triggered and the search stops. All conditions have a time-out that allows execution to continue after a specified period of time, even if the criterion for ending the search was never fulfilled.

The reference values are by default active after an end condition has been triggered, but can be deactivated by the optional argument `ZeroRefAtEnd` in the `FCCondWaitWhile` instruction.

These are the instructions used to handle end conditions:

Instruction	Description
`FCCondPos`	Sets up a TCP position condition. The execution will wait while the condition is true and let the reference values affect the robot. The condition is activated by `FCCondWaitWhile`.
`FCCondOrient`	Sets up a tool orientation condition. The execution will wait while the condition is true and let the reference values affect the robot. The condition is activated by `FCCondWaitWhile`.
`FCCondTCPSpeed`	Sets up a TCP speed condition. The execution will wait while the condition is true and let the reference values affect the robot. The condition is activated by `FCCondWaitWhile`.
`FCCondReoriSpeed`	Sets up a tool reorientation speed condition. The execution will wait while the condition is true and let the reference values affect the robot. The condition is activated by `FCCondWaitWhile`.
`FCCondForce`	Sets up a force condition. The execution will wait while the condition is true and let the reference values affect the robot. The condition is activated by `FCCondWaitWhile`.
`FCCondTorque`	Sets up a torque condition. The execution will wait while the condition is true and let the reference values affect the robot. The condition is activated by `FCCondWaitWhile`.
`FCCondWaitWhile`	Activates the specified condition. By default, the reference values continue to be active after a condition is met. By setting the argument `ZeroRefAtEnd`, the reference values are deactivated when a condition is met. Only the last specified condition is activated.

Instructions for supervision

Instructions for supervision can be used as safety measurements, limiting robot speed, their work area etc. All supervision conditions must be true, otherwise an emergency stop will occur.

Supervision conditions must be set up before `FCAct`, as this is the activating instruction.

Continues on next page

These are the instructions used to handle supervision:

Instruction	Description
`FCSupvPos`	Defines a volume that the TCP must stay within. The supervision is activated by `FCAct`. If no position supervision is set, a default position supervision is set up and activated by `FCAct`. The default supervision is a box stretching 500 mm in each direction from the point where the TCP is when `FCAct` is executed.
`FCSupvOrient`	Defines orientation limits that the tool orientation must stay within. The supervision is activated by `FCAct`.
`FCSupvTCPSpeed`	Defines speed limits that the TCP speed must stay within. The supervision is activated by `FCAct` and is by default 250 mm/s.
`FCSupvReoriSpeed`	Defines reorientation speed limits that the tool orientation must stay within. The supervision is activated by `FCAct` and is by default 50 leg/s.
`FCSupvForce`	Defines force limits that the contact force must stay within. The supervision is activated by `FCAct`.
`FCSupvTorque`	Defines torque limits that the torque must stay within. The supervision is activated by `FCAct`.

Functions

These are the functions used for Force Control:

Function	Description
`FCLoadID`	Identifies the present load using the force sensor. **Note** It is important to make a precise load identification and calibrate the sensor with this load before using Force Control. Always use a Move instruction with fine position before a `FCLoadID`.
`FCGetForce`	Retrieves the force sensor readings.
`FCGetProcessData`	Returns information about the process.
`FCIsForceMode`	Returns true if the robot is in force mode, else false.

Data types

These are the data types used in Force Control:

Data type	Description
`fcboxvol`	A box volume used by `FCCondPos` and `FCSupvPos` to define if the TCP should be inside or outside the box.
`fccylindervol`	A cylinder volume used by `FCCondPos` and `FCSupvPos` to define if the TCP should be inside or outside the cylinder.
`fcspherevol`	A sphere volume used by `FCCondPos` and `FCSupvPos` to define if the TCP should be inside or outside the sphere.
`fcprocessdata`	Used by `FCGetProcessData` to return information about the Force Control process.

Continues on next page

Data type	Description
`fcframe`	Used by `FCAct` and `FCRefMoveFrame` to define which coordinate system should be the reference for the force control coordinate system and the reference movement coordinate system.
`fcplane`	Used by `FCRefCircle` and `FCRefSpiral` to define in which plane the robot should move.
`fcforcevector`	Used by `FCGetForce` to return the detected forces in different directions.
`fcdamping`	Used by `FCAct` to define how fast the robot should move in a direction when it is exposed to a force or a torque in that direction.
`fclindir`	Used by `FCRefLine` to define in which direction the robot should move.
`fcrotdir`	Used by `FCRefRot` to define in which direction the robot should rotate.
`fccondstatus`	Part of the data type `fcprocessdata`. Used to show which conditions are fulfilled.
`fcspeedvector`	Defines3 linear and 3 rotational speed components.
`fcxyznum`	Defines a numerical value in each of the directions x, y and z.

4.2 Code examples

4.2.1 Force controlled pressure applications

Overview

This section provides examples on how to program the press instructions in Force Control for Machining. The basic approach for creating a RAPID program using force controlled press instruction is as follows:

1. Identify the load.
2. Move to a point close to contact but not in contact.
3. Calibrate.
4. Setup force control directions and start movement.
5. Move linear or circular with contact.
6. Leave surface.

Example

The following example uses force Z with a movement in x-direction.

```
PROC press1()
  PERS loaddata TestLoad:=[0.001,[0,0,0.001],[1,0,0,0],0,0,0];

  ! Identify the load using the sensor
  TestLoad:=FCLoadID();

  ! Move close to contact
  MoveJ offs(B,0,0,2) , v100, fine, tool0; !! start 2mm above
       contactpoint

  ! Calibrate the force sensor
  FCCalib TestLoad;

  ! Approach surface and start move to robtarget C at 50% of 60 N
       i.e 30 N
  FCPress1LStart C, v100, \Fz:=60, 50, z30, myTool;

  ! Move Linear from C to D1 with a Force of 50 N in the z-direction
  FCPressL D1,v100,50,z30,myTool;

  ! Move Linear from D1 to D2 with a Force of 70 N in the
       z-direction
  FCPressL D2,v100,70,z30,myTool;

  ! Leave surface and move to robtarget E, Force control is disabled
       after this instruction
  FCPressEnd E, v100,myTool;
ENDPROC
```

4.2.2 Force controlled speed change applications

Overview

This section provides examples on how to program the force controlled SpeedChange function. The basic approach for creating a RAPID program using force controlled SpeedChange is as follows:

1. Configure FC SpeedChange parameters such as, *Feedback type*, *LP filter*, etc.
2. Identify load and calibrate sensor.
3. Active FC SpeedChange with reference and desired recover behavior.
4. Perform machining task.
5. Deactivate FC SpeedChange.

Note

Do not enable force control (`FCAct` in Assembly FC), when using force controlled SpeedChange function.

Example

This example shows how to use Force Controlled SpeedChange function with force sensor. Before running the RAPID program, make sure to set the parameter *Feedback type* to *Calib Force Magn* for FC SpeedChange system.

```
PERS robtarget myHome := ...
VAR tooldata myTool := ...
VAR wobjdata myWobj := ...
PERS loaddata myLoad := ...
myLoad := FCLoadID();
FCCalib myLoad;

! move to home position
MoveL myHome, v200, fine, myTool\WObj:=myWobj;

! turn on spindle motor before machining
TurnOnMotor();

! activate SpeedChange with reference force = 200 (N)
FCSpdChgAct 200;

! conduct machining task along path
MoveL ...
...

! deactivate SpeedChange function
FCSpdChgDeact;

! turn off spindle motor after machining
TurnOffMotor();
```

4.2.3 Force controlled assembly applications

Overview

This section provides examples on how to program the robot. The basic approach for creating a RAPID program using Force Control Assembly is as follows:

1. Identify the load and calibrate the system.
2. Set up desired force and movement pattern.
3. Set up end condition.
4. Activate force control.
5. Activate force and movement pattern.
6. Wait for end condition to occur.
7. Deactivate force and movement patterns.
8. Deactivate force control.

Activating force control

This example shows the simplest way of achieving force control. Between the instructions `FCAct` and `FCDeact` the robot will be sensitive to all forces affecting the sensor. The robot will move away from any contact forces, trying to maintain zero contact force on the sensor.

```
VAR tooldata tool1:=[TRUE,[[97.4,0,223],[1,0,0,0]], [5,[23,0,75],
     [1,0,0,0],0,0,0]];
PERS loaddata my_load:=[0.001,[0,0,0.001],[1,0,0,0],0,0,0];

! Identify the load using the sensor
my_load := FCLoadID();
! Calibrate the force sensor
FCCalib my_load;
! Activate force control
FCAct tool1;
! Force control is active for 5 seconds
WaitTime 5;
! Deactivate force control
FCDeact;
```

Find object position

In this example, the robot will move in the z direction of the world frame. When it runs into an object, it will stop when the force is 10 N and wait there with a constant force. The position of the object can then be measured.

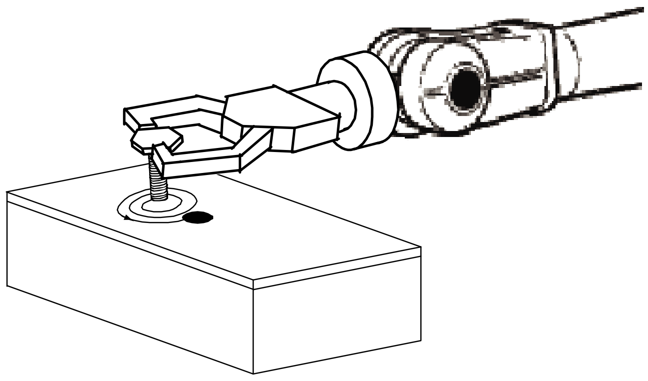
```
VAR tooldata tool1 := ...
PERS loaddata my_load:=[0.001,[0,0,0.001],[1,0,0,0],0,0,0];
VAR pos pos1;

my_load := FCLoadID();
FCCalib my_load;

! Setup the force reference with 10N in Z-direction of the world
     frame
```

Continues on next page

```
FCRefForce \Fz:=10;

! Activate Force Control
FCAct tool1;

! Start moving the robot to achieve the specified force
FCRefStart;

! Wait 10 sec, so the robot will reach the ordered force
WaitTime 10;

! Read robot position
pos1 := CPos(\Tool:=tool1);

! Stop the reference values
FCRefStop;

! Deactivate force control
FCDeact;
```

Position search

In this example, the robot holds a bolt that should be inserted in a hole. The bolt is pressed towards a surface with a force of 10 N. It is moved in spirals along the surface until the hole is found. When the bolt is above the hole, the force will press it into the hole. When the bolt enters the hole, the z value of the TCP will become less than 550, the position search is finished and the program execution continues.


xx0500001482

```
VAR tooldata tool1 := ...
PERS loaddata my_load:=[0.001,[0,0,0.001],[1,0,0,0],0,0,0];
VAR fcboxvol my_box:= [-9e9, 9e9, -9e9, 9e9, 550, 9e9];
VAR fcprocessdata process_data;

my_load := FCLoadID();
FCCalib my_load;

! Setup the force reference with 10N in negative z direction
FCRefForce \Fz:=-10;
```

Continues on next page

```
! Setup movement pattern
FCRefSpiral FCPlane_XY, 90, 50, 10;

! Setup end condition: while Z>550 and time<60
FCCondPos \Box:= my_box, 60;

! Activate Force Control
FCAct tool1;

! Activate force and movement pattern (the robot starts to move)
FCRefStart;

! Wait for end condition
FCCondWaitWhile;

! Check if the position condition or timeout trigged the condition
! Note that if a condition is FALSE it means it has trigged
process_data:=FcGetProcessData(\DataAtTrigTime);
TPWrite "timecond = " \BOOL := process_data.conditionstatus.time;
TPWrite "positioncond = " \BOOL :=
     process_data.conditionstatus.position;
FcRefStop
FCDeact;
```

Supervision example

In this example, supervision is used. A position supervision limits the x coordinate to be between -200 and 1000, the y coordinate to be between -500 and 500 and the z coordinate to be between 300 and 1200. A force supervision in the force coordinate system limits the force in positive z direction to 1000 N.

```
VAR tooldata tool1 := ...
PERS loaddata my_load:=[0.001,[0,0,0.001],[1,0,0,0],0,0,0];
VAR fcboxvol my_supv_box:= [-200, 1000, -500, 500, 300, 1200];

my_load := FCLoadID();
FCCalib my_load;

! Set up position supervision
FCSupvPos \Box:= my_supv_box;

! Set up force supervision
FCSupvForce \Zmax:=1000;

! Activate Force Control and supervision
FCAct tool1;
...
FCDeact;
```

5 Execution behavior

5.1 Conflicting reference values

Only one reference in each direction

For each of the directions x, y and z, there can only be one reference movement.

A new reference value overwrites the old value in the same direction.

Example 1

In this example, a force reference of 20 N will be activated by `FCRefStart`. The first `FCRefForce` will be overwritten, and therefore has no effect.

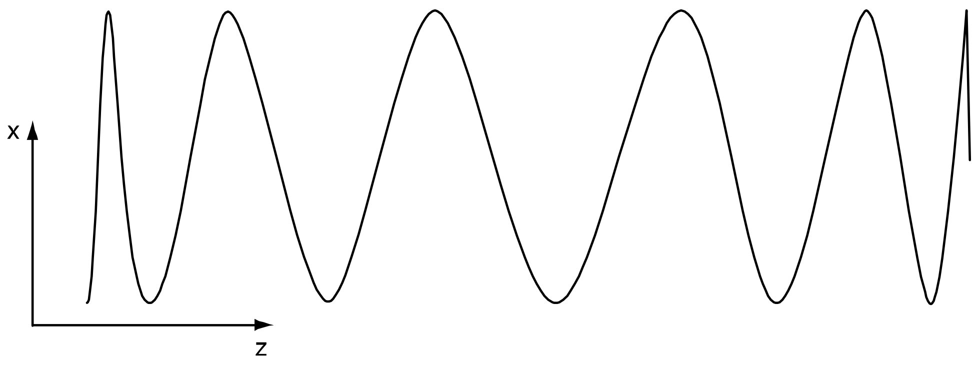
```
FCRefForce \Fx:=10;
FCRefForce \Fx:=20;
FCAct tool1;
FCRefStart;
```

Example 2

In this example, the reference movement in the x direction will be determined by the `FCRefCircle` instruction. The reference movement in the z direction is overwritten by the `FCRefLine` instruction.

```
FCRefCircle FCPLANE_XZ, 15, 300;
FCRefLine FC_LIN_Z, 300, 100;
FCAct tool1;
FCRefStart;
```

The resulting reference movement will be similar to the illustration below.



xx0500001576

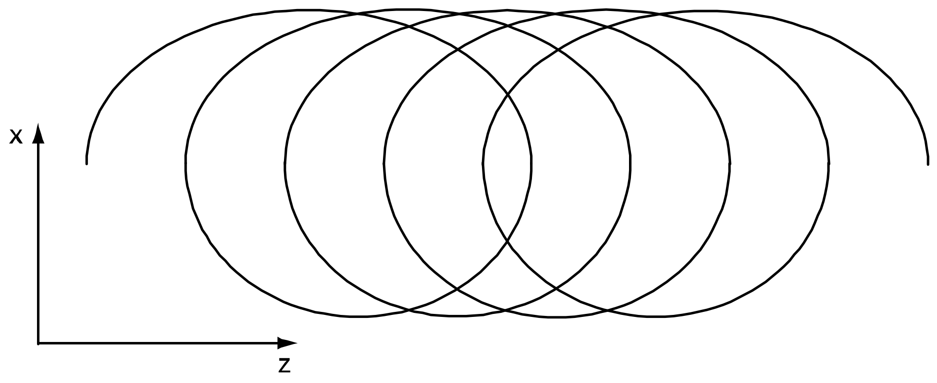
Example 3

If you want a movement in the shape of a circle that drifts in one direction, you cannot use `FCRefCircle` and `FCRefLine` (see example above). Instead you can use `FCRefCircle` and `FCRefForce`, where the reference force results in a linear movement.

```
FCRefCircle FCPlane_XZ, 180, 100;
FCRefForce \Fz:=10;
FCAct tool1;
```

Continues on next page

```
FCRefStart;
```



xx0500001577

5.2 Damping and LP-filter

Damping

Damping is a definition of how large force is required for the robot to move at a certain speed. The damping parameters define how many Newtons are required to make the robot move at 1 m/s. The higher the value, the less responsive the robot gets.

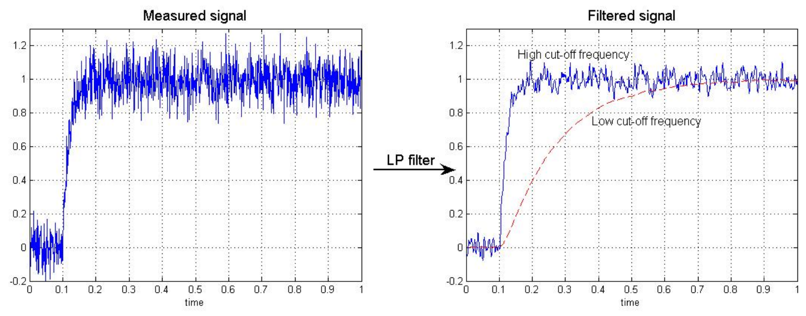
In Force Control, a contact force will make the TCP move with a speed proportional to the contact force. A contact torque will make the tool reorient with a speed proportional to the contact torque. The damping variable defines the proportions between a force and the resulting speed, and a torque and the resulting reorientation speed, in the direction x, y and z. The values are given as a percentage of the of the system parameter values defined in the type FC Kinematics, see *The FC Kinematics type on page 201*.

LP-filter

A Low-Pass filter lets the amplitude of low frequency signals pass through, and the amplitude of frequencies higher than the cut-off frequency are attenuated. If the signal is changing rapidly, a high cut-off frequency is needed. On the other hand, if the measured force is noisy, a low cut-off frequency may be required in order to remove the noise.

Illustration

The figure illustrates an LP-filter.



en0900000128

Force controller structure and tuning

The picture shows a simplified picture of the force control loop.

In a force controlled direction the measured forces are subtracted from the corresponding reference forces.

This difference is divided by D (=Damping). Damping is a force to speed factor and thus a speed reference is generated. This speed reference is low pass filtered with a cut off frequency that should be chosen depending on your robot model and process etc.

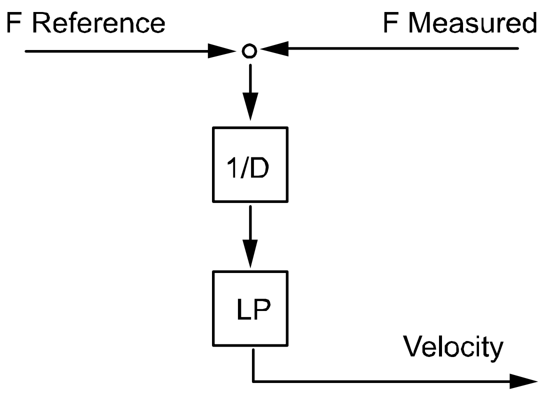
Default it is set to 3Hz which is a suitable value when the contact is really stiff (metal to metal).

Continues on next page

For the large robot it is not possible to increase this value much but for a small robot with some compliance in the tool a filter frequency up to 25Hz can be used.

Since tuning both damping and low pass filter depends on compliance of tool, robot model, robot configuration etc. there is a unique set of these parameters for each process.

Both the damping and the LP filter cut-off frequency strongly affect how quickly and accurately the system is able to control the pressure force. If the robot reacts slowly when the force changes, or loses contact with the workpiece for periods of one or several seconds also for accurately programmed paths and low speeds, it is often possible to improve performance by decreasing the damping and/or increasing the LP filter cut-off frequency. On the other hand, if the robot bounces or vibrates rapidly with constant or increasing amplitude when pressing against the surface, this indicates that the damping should be increased and/or the filter frequency should be decreased.



en0600003351

Note

Changing the parameters of the damping or Low pass filter might make the robot unstable.

Damping

The damping can easily be changed by an argument `in FCPress1LStart`. The argument represent a percentage of the configured damping value. (down to 50%, no upper limit)

For bigger changes the damping value needs to be re configured under, Motion->FC Kinematics-> Damping in Force Z direction.

Low pass filter

Low Pass filter can be configured by the parameters in *Type FC Kinematics on page 201* or set by the RAPID instruction *FCSetLPFilterTune on page 129*.

Default value for `FCSetLPFilterTune` is set by the system parameter *Bandwidth of force loop filter* in the type *FC Kinematics*.

Continues on next page

The value set by this method is valid until the instruction `FCResetFilterTune` is used, a new value is set, or the controller is restarted.

Note

The low pass filer cannot be changed on the GoFa robot.

5.3 Overcome friction

Overview

The friction generated between the tool and the work object depends on the force applied by the robot. If the speed parameter in the `FCRefLine` instruction is too low the friction will keep the robot from moving.

Example

```
...
!Setup the force reference in the positive z direction
FCRefForce \Fz:=300;
FCRefLine FC_LIN_Y, 0.1, 100;
FCAct tool1;
FCRefStart;
...
```

In this example the force against the work object is high and the speed is very low, so the robot might not move from its position.

To get the robot moving try one of the following:

- Reduce the force.
- Increase the speed.
- Add oscillation to the force.

Note

The amount of friction also depends on the materials of the tool and work object.

5.4 Special cases

Jog the robot in force control

The user is allowed to jog the robot in force control, the setup by `FCAct` is valid during jogging. To jog the robot in normal mode, use `FCDeact` or PPmove (see below).

Use PPmove

Moving the program pointer in the RAPID code is normally no problem. But if the program pointer is moved when the robot is in Force mode, the robot automatically switches from force mode to position mode.

Move the robot

Normal Move instructions are not allowed, and will be ignored if used, in force mode. This also applies to regain movements.

Start, stop and stepwise execution in manual mode

If you push the stop button force control references stop, but the force control mode is still active. When you push the start button force control references will start again.

During stepwise execution any force control references will start and stop, just as if you were pushing the start and stop buttons.

5.5 FC Press optimization

5.5.1 Use Spd FFW

Overview

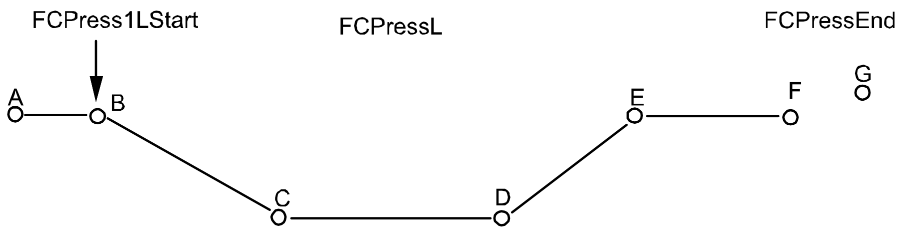
If the path is complex and the programmed path is accurate the performance is going to be enhanced by adding optional argument `UseSpdFFW` (use speed feed forward). `FCPressure` may be also used for temporary leaving the surface without deactivating.

Example 1

This example illustrates how to increase the performance.

```
FCPress1LStart;
```

The force in this example is directed down, see picture below. It would be possible to run directly from robtarget B to robtarget E but the performance will increase by adding robtarget C and D and using optional argument `\UseSpdFFW`.



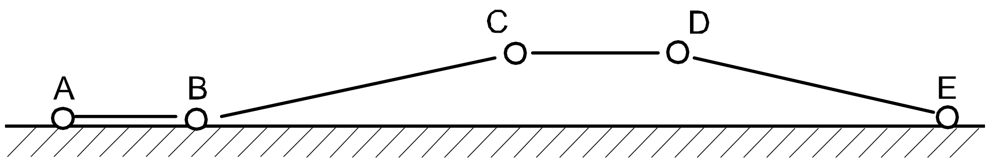
en0600003350

Example 2

This example illustrates how to leave the surface without deactivating.

```
FCPress1LStart B, v100, 70, z30 \UseSpdFFW, tool1;
FCPressL C, v100, 0, z30, tool1;
FCPressL D, v100, 0, z30, tool1;
FCPressL E, v100, 70, z30, tool1;
```

The reference force is temporarily switched off (set to 0) together with the optional argument `UseSpdFFW`. The robot will leave the surface and follow the path to C and D. Note that the robot is still force controlled and will not behave 100% like position controlled robot. The robot will not reach exactly to position C or D.



en0600003357

Continues on next page

Note

Leaving the surface works best for tool coordinate system or work object coordinate system, i.e. argument `ForceFrameRef` in instruction `FCPress1LStart` set to `FC_REFFRAME_TOOL` (default) or `FC_REFFRAME_WOBJ`. If path coordinate system (FC_REFFRAME_PATH) is used, a jerky motion may occur. See *Programming in path coordinates on page 58*.

5.6 FC SpeedChange control design

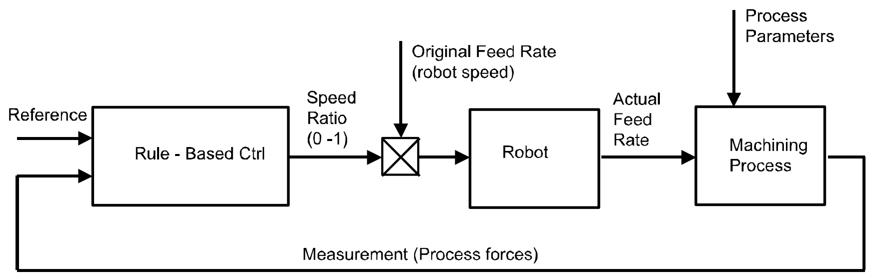
5.6.1 Controller scheme

Overview

The FC SpeedChange controller for reducing/increasing the robot speed is a rule-based logic controller. The design details are described in following sections.

Scheme

The force controlled speed control scheme is shown in the figure below. The maximum force is specified for the machining process, and the actual process force is monitored and controlled to be less than the maximum force by adjusting the machining feed rate (robot speed). The output of the rule-based logic control is the percentage (between 0% and 100%) of the original feed rate.

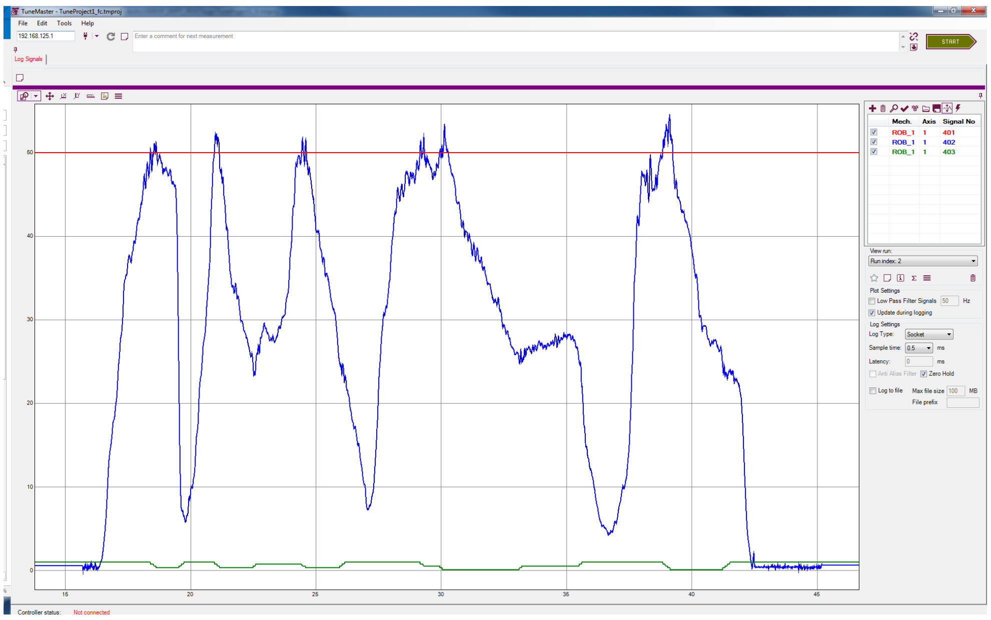


en0600003278

Continues on next page

Available test signals to tune the process

The following picture illustrates the three test signals that appear from a recording made with the TuneMaster. In this example the reference signal is set to value 50. When the difference between the reference signal (401) and the measurement signal (402) fulfills the criteria for speed decrease the speed ratio signal (403) drops, in this case to four levels.



	Mech.	Axis	Signal No
✓	ROB_1	1	401
✓	ROB_1	1	402
✓	ROB_1	1	403

View run: Run index: 2

Plot Settings
☐ Low Pass Filter Signals 50 Hz
☑ Update during logging

Log Settings
Log Type: Socket
Sample time: 0.5 ms
Latency: 0 ms
☐ Anti Alias Filter ☑ Zero Hold
☐ Log to file Max file size 100 MB
File prefix

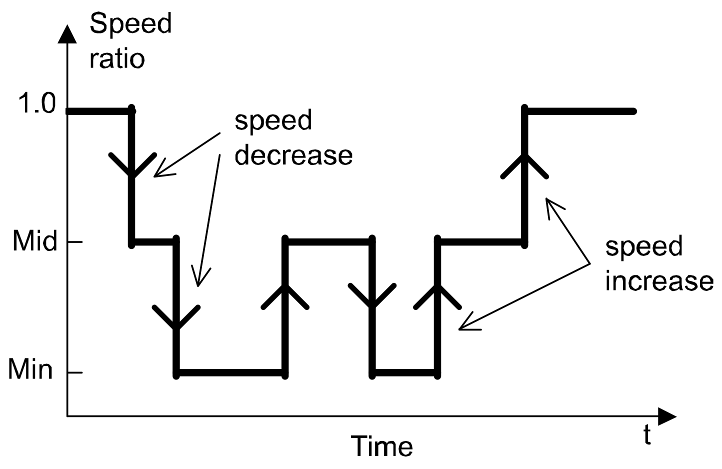
xx1800000946

Signal 401 (red color)	Reference test signal
Signal 402 (blue color)	Measurement (Process force) test signal
Signal 403 (green color)	Speed ratio signal

5.6.2 Rule based logical control

Increase/Decrease

The controller output, speed ratio, is generated by certain rules based on the measured process force information. A sample speed ratio output of 3-step rule-based logic controller is shown in the following figure.



en0600003279

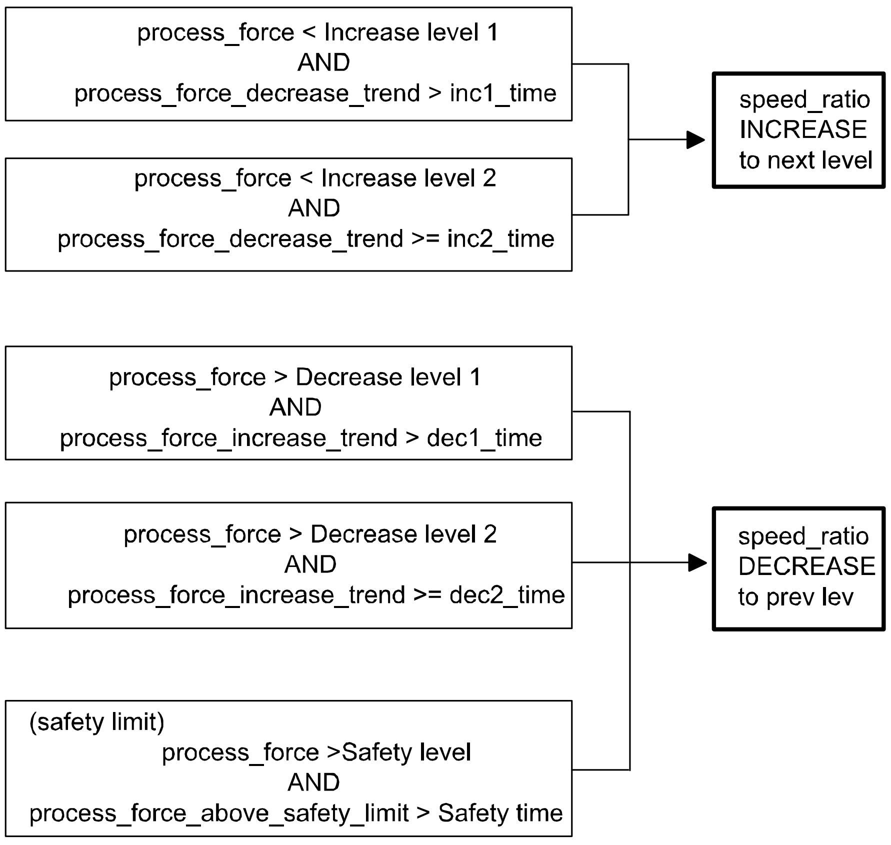
Instead of changing continuously as in normal PID control, speed ratio in rule-based logic control is divided into several discrete steps. The logic rules will decide when the speed will decrease or increase to the next stage or remain at current stage. The goal of SpeedChange is to keep process force below a pre-specified maximum force as fast as possible.

Although ideally more steps means more control accuracy, 3 steps would be enough for most applications. Too many steps will increase the response time when cutting large-size material and make speed reduction less responsive. 2-step speed control would be the most used control setup.

Continues on next page

Rules

The following rules decide when to increase or decrease the speed ratio. In all other conditions, controller maintains the previous speed ratio.


process_force < Increase level 1
AND
process_force_decrease_trend > inc1_time
process_force < Increase level 2
AND
process_force_decrease_trend >= inc2_time
speed_ratio INCREASE to next level
process_force > Decrease level 1
AND
process_force_increase_trend > dec1_time
process_force > Decrease level 2
AND
process_force_increase_trend >= dec2_time
(safety limit)
process_force >Safety level
AND
process_force_above_safety_limit > Safety time
speed_ratio DECREASE to prev lev

en0600003280

5.6.3 Programming in path coordinates

Overview

Defining the force control coordinate system relative to the path coordinate system gives the possibility to define the force control action relative to the programmed motion trajectories of the robot. This is particularly useful in situations where it is desired to apply a force against a surface with a varying normal direction, without reorienting the tool.

For a description of the path coordinate system see *Technical reference manual - RAPID Instructions, Functions and Data types*, instruction `CorrCon`.

Usage

It is important to note that all directions in path coordinates except the force control direction will be position-controlled. If the robot TCP position has drifted away from the programmed path and the path frame undergoes a quick rotation, this will cause the position reference to change quickly. This will introduce a very rapid corrective motion or "jerk" in the position-controlled directions. Built-in supervision is present in the system in order to stop the robot if the corrective motion becomes too fast. If this occurs, modify the program in one or several of the following ways:

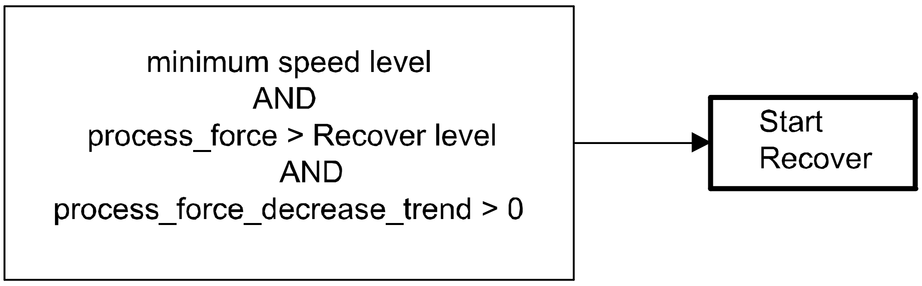
1. Program the path closer to the surface.
2. Decrease the path speed, especially near sharp corners in the path.
3. Reprogram the path to avoid sharp path corners, for example, by increasing the size of the corner zones.

A rule of thumb is that the force control path deviation, the distance from the programmed TCP position to the true TCP position, should be shorter than the effective radius of curvature of the programmed path.

5.6.4 Recovery routine

Overview

If process force is still higher than the reference force when the feed rate is already at its lowest possible speed (usually this condition happens if larger volume of material than expected is encountered during machining process), a recovery process will start in order to avoid tool damage. The following diagram is the rule to enter the recovery process.


minimum speed level
AND
process_force > Recover level
AND
process_force_decrease_trend > 0
Start Recover

en0600003282

The recovery process could simply stop the robot, or more a advanced solution would be to perform local cutting in layers to get rid of the large material, and then continue the original path after local cutting. The local cutting process needs to be implemented by user as a RAPID routine, no recovery routine is defined by default. A sample recovery routine is shown as below.

Recover example

```
PROC user_recover_routine()
VAR robtarget current_rbtrgt;
! get starting robot target
current_rbtrgt := CRobT(\Tool:=UserTool \WObj:=UserWobj);
! local cutting relative to current_rbtrgt
MoveL RelTool(current_rbtrgt,dx,dy,dz), v50, z0, UserTool
     \WObj:=UserWobj;
MoveL RelTool() …… ;
……
! move back to starting point
MoveL current_rbtrgt, v50, z0, UserTool\WObj:=UserWobj;
ENDPROC
```

The user-defined recover routine will be automatically called during recover process when specified.

6 Troubleshooting

6.1 What to do when...

Troubleshooting drifting robot

When the robot is in force control mode (after executing `FCAct`) and no external force except gravity are present, it should not move (as long as no reference force, torque or movement is applied). If the robot drifts away with a slow movement anyway, check the following:

Step	Action
1	The robot should be near its working position when calibrated with the instruction `FCCalib`.
2	Verify that the load is identified with the instruction `FCLoadID` and that the `LoadIdErr` is smaller than 0.1 for an optimal load identification. If the movement of axes 5 and 6 are too limited during this load identification, the result may be poor.
3	Verify that the system parameters are correctly defined. E.g. too low damping in the type *FC Kinematics* may cause the robot to drift.
4	Verify the orientation of gravity with respect to the base frame. If necessary update the system parameter *Gravity Alpha* and *Gravity Beta* in the type *Robot*. For more information, see *Technical reference manual - System parameters*.

Troubleshooting that the robot stops with an emergency error

When the force control is activated by `FCAct` the speed and position is by default supervised. If the supervisions conditions are triggered the robot stops with an emergency error. These default conditions can be changed by specifying new conditions in the program.

Example 1

```
FCSupvPos \PosSupvFrame:=[current_pos.trans, [1,0,0,0]]
     \Box:=[-1500,1500,-1500,1500,-1500,1500];
FCAct tool0 \ForceFrameRef:=FC_REFFRAME_WOBJ
     \ForceFrameOrient:=[1,0,0,0]
     \DampingTune:=[50,50,50,50,50,50];
```

Sets up a position supervision where the TCP must stay between -1500 mm and 1500 mm in all the three directions (i.e. x, y, z).

Example 2

```
FCSupvTCPSpeed \Speed \Xmin:=-1000 \Xmax:= 1000 \Ymin:=-1000 \Ymax:=
     1000 \Zmin:=-1000 \Zmax:= 1000;
FCAct tool0 \ForceFrameRef:=FC_REFFRAME_WOBJ
     \ForceFrameOrient:=[1,0,0,0]
     \DampingTune:=[50,50,50,50,50,50];
```

Sets up a linear speed supervision where the TCP speed must stay between -1000 mm/s and 1000 mm/s in all the three directions (i.e. x, y, z).

Example 3

```
FCSupvReoriSpeed \Speed \Xmin:=-100 \Xmax:= 100 \Ymin:=-100 \Ymax:=
     100 \Zmin:=-100 \Zmax:= 100;
```

Continues on next page

```
FCAct tool0 \ForceFrameRef:=FC_REFFRAME_WOBJ
     \ForceFrameOrient:=[1,0,0,0]
     \DampingTune:=[50,50,50,50,50,50];
```

Sets up a reorientation speed supervision where the reorientation speed must stay between -100 deg/s and 50 deg/s around all the three directions (i.e. x, y, z).

7 RAPID reference information

7.1 Instructions

7.1.1 FCAct

Usage

`FCAct` is used to activate Force Control. At the same time as Force Control is activated, `FCAct` is used to define the coordinate system for Force Control, and tune the force and torque damping. If a coordinate system is not specified in `FCAct` a default force control coordinate system is created with the same orientation as the work object coordinate system.

All Force Control supervision is activated by `FCAct`.

Basic example

```
VAR tooldata tool1:=[TRUE,[[97.4,0,223],[1,0,0,0]], [5,[23,0,75],
     [1,0,0,0],0,0,0]];
FCAct tool1;
```

Activates Force Control with tool `tool1`.

The force control coordinate system has the same orientation as the world coordinate system. All dampings are set to 100% of the damping values in the system parameters.

Since the reference were not set in this example the robot will move away from any contact, trying to keep zero contact force.

See also *More examples on page 64*.

Arguments

```
FCAct Tool [\WObj] [\ForceFrameRef] [\ForceFrameOrient]
     [\DampingTune]
```

`Tool`

Data type: `tooldata`

The tool used during Force Control. The center point of this tool is the center of the force control coordinate system. Note that the dimensions of the sensor and any interface plates need to be included in the tool definition.

`[\WObj]`

Work object

Data type: `wobjdata`

Many of the Force Control definitions are based on the work object coordinate system. For example the orientation of the force control coordinate system and all the corresponding definitions are often given in relation to the work object coordinate system. Search patterns and end conditions are usually also defined in this coordinate system. If no work object is defined the default work object, with coordinate system equal to the world coordinate system, is used.

Continues on next page

`[\ForceFrameRef]`

Data type: `fcframe`

`ForceFrameRef` here defines which coordinate system the force control coordinate system is related to. The parameter can be set to either the work object coordinate system or the tool coordinate system. The default value is the work object coordinate system.

`[\ForceFrameOrient]`

Data type: `orient`

This parameter specifies the orientation from the coordinate system selected in `ForceFrameRef`. The default value is [1, 0, 0, 0]. For information about how to calculate orientations, see the data type `orient` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

`[\DampingTune]`

Tuning of force and torque damping

Data type: `fcdamping`

The `DampingTune` values can be used to modify the relation between the sensed force and the generated velocity in each direction. By default the values are 100% (of system parameter values) in all directions, but they can be between 50% and infinity. Smaller values means that the robot is more sensitive to external forces.

Program execution

Execution behavior:

- `FCAct` activates Force Control, but does not activate reference values. Until `FCRefStart` is executed, the robot behavior is to move away from any sensed contact trying to keep zero contact force.
- All supervision conditions (e.g. `FCSupvPos`), set up prior to the `FCAct` instruction, are activated by `FCAct`. After activation, if any of these conditions are false, an emergency stop will occur.

More examples

Example 1

```
VAR tooldata tool1:=[TRUE,[[97.4,0,223],[1,0,0,0]], [5,[23,0,75],
     [1,0,0,0],0,0,0]];
FCAct tool1 \ForceFrameRef:=FC_REFFRAME_TOOL
     \ForceFrameOrient:=[0,0,1,0];
```

Activates Force Control and defines a force control coordinate system. Based on the tool coordinate system, but rotated 180 degrees around the tools y axis.

Example 2

```
VAR tooldata tool1:=...
VAR wobjdata my_wobj := [FALSE,TRUE,"",[[0,0,0],[,0,0,0]],
     [[0,0,0],[0.07071,0,0.7071,0]]];
FCAct tool1 \WObj:=my_wobj;
```

Activates Force Control with:

- tool `tool1`

Continues on next page

- force control coordinate system orientation equal to the orientation of the work object `my_wobj`.

Default supervision

When the force control is activated the speed and position is by default supervised. If the supervisions conditions are triggered, the robot stops with an emergency error.

These conditions are set as a default supervision:

- The position supervision is set so that the TCP must stay between -500mm and 500 mm in all directions (i.e. x, y, z) of the position supervision coordinate system.
- The linear speed supervision checks that the speed in all directions of the work object coordinate system is between -250 mm/s and 250 mm/s.
- The reorientation speed supervision checks that the reorientation speed in all directions of the work object coordinate system must be between -50 deg/s and 50 deg/s.

The default supervised conditions can be changed by specifying new conditions in the code, for more information see*Troubleshooting that the robot stops with an emergency error on page 61*.

Limitations

The Force Control will only behave correctly if the load is identified with `FCLoadID` and the sensor is calibrated with `FCCalib` before activating Force Control.

Syntax

```
FCAct
  [Tool':='] <persistent (PERS) of tooldata>
  ['\'Wobj':=' <persistent (PERS) of wobjdata>]
  ['\'ForceFrameRef':=' <expression (IN) of fcframe>]
  ['\'ForceFrameOrient':=' <expression (IN) of orient>]
  ['\'DampingTune':=' <expression (IN) of fcnumvector>]';'
```

Related information

For information about	See
The data type `fcframe`	*fcframe on page 173*
Load identification	*FCLoadID on page 159*
Sensor calibration	*FCCalib on page 66*
Deactivating Force Control	*FCDeact on page 90*

7.1.2 FCCalib

Usage

`FCCalib` is used to calibrate the force sensor. Before this instruction is executed it is not possible to switch to force control. It is necessary to specify the data for the used load as an argument to this instruction. Load data can easily be retrieved by first performing a load identification using the function `FCLoadID`. By using the load data the system can do an internal calibration to compensate for sensor offset and prepare gravity force compensation. It is important to understand that the function `FCCalib` needs to be run every time the load is changed.

Tip

It is also recommended to do the calibration close to the position where the robot will be doing most work.

Note

The Calibration should always be done when no contact forces are present. The only exception is when using the optional parameter `Recovery` which might be used for example after an emergency stop.

Basic example

```
PERS loaddata my_load:=[0.001,[0,0,0.001],[1,0,0,0],0,0,0];
my_load:= FCLoadID();
FCCalib my_load;
```

The example above shows how to use `FCCalib`. It is very important for force control performance to have a good load definition. It is therefore strongly recommended to identify the load using the function `FCLoadID`.

Note

If sensor is room fixed, any load can be used as argument.

Arguments

```
FCCalib Load [\Recovery][\IgnoreStandStillCheck]
```

`Load`

Data type: `loaddata`

The load used to calibrate the sensor. Only mass and centre of gravity is presently used, so inertia does not have to be specified. Note that this load is the load the sensor feels. It is normal that this load is *not* zero even if only the sensor itself is mounted on the robot. Use the function `FCLoadID` to identify the load.

`[\Recovery]`

Data type: `switch`

Continues on next page

Specifies whether to use the previous calibration offset, which was read the last time `FCCalib` was called without this argument. Makes it possible to activate force control when in contact. The argument can be needed for example after an emergency stop.

`[\IgnoreStandStillCheck]`

Data type: `switch`

Specifies if the process should check if vibrations from the previous movement should be ignored. If the process is stable and the robot is standing still, setting this argument can reduce the cycle time. This argument should not be set if the tool, such as the spindle, is vibrating.

Program execution

Before the sensor is calibrated with `FCCalib`, most other Force Control instructions are not allowed.

Syntax

```
FCCalib
  [Load':='] <expression (IN) of loaddata>
  ['\' Recovery]';'
  ['\' IgnoreStandStillCheck]';'
```

Related information

For information about	See
Load identification	*FCLoadID on page 159*.

7.1.3 FCCondCombine

Usage

`FCCondCombine` is used to combine more than one condition. The condition is later activated by calling the instruction `FCCondWaitWhile`, which will wait and hold the program execution while the specified condition is true.

Once activated with `FCCondWaitWhile`, the program execution will continue to wait while combined condition is within its specified limits.

Basic examples

The following example illustrates the instruction `FCCondCombine`.

```
FCCondCombine FC_POSITION_AND_FORCE_OR_TIME;
```

Enables a combination of a force and position condition.

Arguments

```
FCCondCombine fccombcond
```

`fccombcond`

Data type: `string`

The following values are allowed:

- FC_POSITION_OR_TIME
- FC_SPEED_OR_TIME
- FC_FORCE_OR_TIME
- FC_POSITION_OR_FORCE_OR_TIME
- FC_POSITION_AND_FORCE_AND_TIME
- FC_POSITION_AND_FORCE_OR_TIME

Limitations

The timeout used will be the one for the condition that was setup last.

Example if `FCCondPos` and `FCCondForce` is combined:

```
FCCondForce \Xmin:=-100 \XMax:=100, 60 ; ! 60s timeout
FCCondPos \Box:= my_box, 30; ! 30s timeout
FCCondCombine FC_POSITION_AND_FORCE_OR_TIME ;
```

Syntax

```
FCCondCombine < expression (IN) of string > ';'
```

Related information

For information about	See
`FCCondWaitWhile`	*FCCondWaitWhile on page 88*

7.1.4 FCCondForce

Usage

`FCCondForce` is used to set up an end condition based on measured force. The condition is later activated by calling the instruction `FCCondWaitWhile`, which will wait and hold the program execution while the specified condition is true. This allows the reference force, torque and movement to continue until the force is outside the specified limits.

A force condition is set up by defining minimum and maximum limits for the force in the directions of the force control coordinate system. Once activated with `FCCondWaitWhile`, the program execution will continue to wait while the measured force is within its specified limits.

It is possible to specify that the condition is fulfilled when the force is outside the specified limits instead. This is done by using the switch argument `Outside`.

The condition on force is specified in the force control coordinate system. This coordinate system is setup by the user in the instruction `FCAct`.

Basic example

```
FCCondForce \Xmin:=-100 \XMax:=100, 60;
```

Defines a force condition that is true when the force in the x direction of the force control coordinate system is between -100 N and 100 N. No restriction is put on the force in other directions.

When this condition is activated the program execution will wait until the measured force is outside its limits, or until 60 seconds has passed.

See also *More examples on page 70*.

Arguments

```
FCCondForce [\XMin] [\XMax] [\YMin] [\YMax] [\ZMin] [\ZMax]
     [\Outside] TimeOut
```

`[\XMin]`

Minimum force in x direction

Data type: `num`

Lower limit for force in the x direction of the force control coordinate system. A negative value limits the maximum force in the negative x direction.

The unit is Newton and the default value is negative infinity.

`[\XMax]`

Maximum force in x direction

Data type: `num`

Upper limit for force in the x direction of the force control coordinate system. A negative value limits the minimum force in negative x direction.

The unit is Newton and the default value is positive infinity.

`[\YMin]`

Minimum force in y direction

Continues on next page

Data type: `num`

Lower limit for force in the y direction of the force control coordinate system. A negative value limits the maximum force in the negative y direction.

The unit is Newton and the default value is negative infinity.

`[\YMax]`

Maximum force in y direction

Data type: `num`

Upper limit for force in the y direction of the force control coordinate system. A negative value limits the minimum force in negative y direction.

The unit is Newton and the default value is positive infinity.

`[\ZMin]`

Minimum force in z direction

Data type: `num`

Lower limit for force in the z direction of the force control coordinate system. A negative value limits the maximum force in the negative z direction.

The unit is Newton and the default value is negative infinity.

`[\ZMax]`

Maximum force in z direction

Data type: `num`

Upper limit for force in the z direction of the force control coordinate system. A negative value limits the minimum force in negative z direction.

The unit is Newton and the default value is positive infinity.

`[\Outside]`

Data type: `switch`

Specify that the condition is fulfilled when the force is outside the specified limits.

`TimeOut`

Data type: `num`

This is the maximum time the condition is valid, in seconds. If the force condition has not turned false before this time, the wait is interrupted and the next RAPID instruction is executed.

Program execution

Execution behavior:

- A time condition must be given (the argument `TimeOut`). The condition is considered true as long as the force condition AND the time condition is true.
- Use *FCGetProcessData on page 156*, too see if the condition was met or timed out.

More examples

Example 1

```
FCCondForce \XMin:=-100 \XMax:=100 \YMin:=-200 \YMax:=200, 60;
```

Continues on next page

Defines a force condition where the force in the x direction should be between -100 N and 100 N and in the y direction between -200 N and 200 N. The time condition is set to 60 seconds.

Example 2

```
FCCondForce \ZMin:=0 \ZMax:=100 \Outside, 60;
```

In this example the switch `Outside` is set, which means that the condition is fulfilled as long as the force is outside the specified limits. That is as long as the force in the Z direction is smaller than 0 N or larger than 100 N. The time-out is 60 seconds.

Example 3

```
FCCondForce \ZMax:=-10, 60;
```

This condition is true as long as the force in negative z direction is larger than 10 N. The time-out is 60 seconds.

Note

Sometimes the measured force is quite noisy. It is possible to filter the measured force by using the system parameter *Bandwidth of force frame filter*, under type *FC Kinematics*.

Limitations

The maximum reference force has different limit for different robot models, see the data sheet for Force Control.

Syntax

```
FCCondForce
  ['\'XMin':=' <expression (IN) of num>]
  ['\'XMax':=' <expression (IN) of num>]
  ['\'YMin':=' <expression (IN) of num>]
  ['\'YMax':=' <expression (IN) of num>]
  ['\' ZMin':=' <expression (IN) of num>]
  ['\' ZMax':=' <expression (IN) of num>]
  ['\'Outside]','
  [TimeOut':='] <expression (IN) of num>';'
```

Related information

For information about	See
Setting up TCP position condition	*FCCondPos on page 76*
Setting up reorientation condition	*FCCondReoriSpeed on page 79*
Setting up TCP speed condition	*FCCondTCPSpeed on page 82*
Setting up torque condition	*FCCondTorque on page 85*

7.1.5 FCCondOrient

Usage

FCCondOrient is used to set up an end condition for the tool orientation. The condition is later activated by calling the instruction FCCondWaitWhile, which will wait and hold the program execution while the specified condition is true. This allows the reference force, torque and movement to continue until the orientation is outside the specified limits.

An orientation condition is set up by defining a maximum angle and a maximum rotation from a reference orientation. The reference orientation is either defined by the current z direction of the tool, or by specifying an orientation in relation to the z direction of the work object.

Once activated, the tool orientation must be within the limits (or outside, if the argument Outside is used).

Basic example

```
FCCondOrient \MaxAngle:= 15, 60;
```

In this example, no orientation condition coordinate system is specified. This means that the condition coordinate system is the same as the tool coordinate system at the time of execution of this instruction. When this condition is activated the program execution will wait until the tool's z axis deviates more than 15 degrees from the z axis of the condition coordinate system, or until 60 seconds has passed.

See also *More examples on page 74*.

Arguments

```
FCCondOrient [\OrientCondFrame] [\MaxAngle] [\MaxRot] [\Outside]
     TimeOut
```

[\OrientCondFrame]

Orient condition coordinate system

Data type: orient

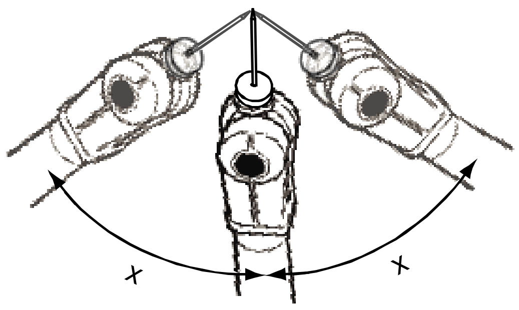
OrientCondFrame is used to set the coordinate system in which the tool orientation condition is defined. The coordinate system is set by an orient in relation to the work object coordinate system. If OrientCondFrame is omitted, the tool coordinate system at the time of execution is used as orientation condition coordinate system.

Continues on next page

[\MaxAngle]

Data type: num

The maximum allowed angle between the z direction of the tool and the z direction of the orientation condition coordinate system. The unit is degrees.



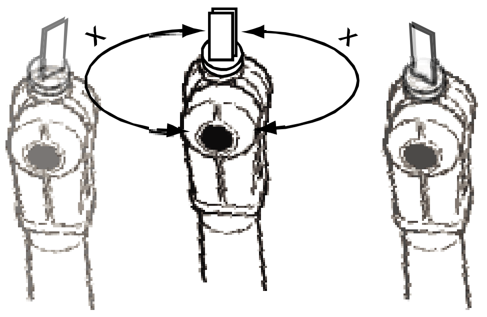
xx0500001913

x	MaxAngle

[\MaxRot]

Data type: num

The maximum tool rotation around the z axis, compared to the orientation condition coordinate system. The unit is in degrees.



xx0500001912

x	MaxRot

[\Outside]

Data type: switch

Makes the condition true when the tool orientation is outside the specified angles.

TimeOut

Data type: num

This is the maximum time the condition is valid, in seconds. If the orientation condition has not turned false before this time, the wait is interrupted and the next RAPID instruction is executed.

Continues on next page

Program execution

Execution behavior:

- A time condition must be given (the argument `TimeOut`). The condition is considered true as long as the orientation condition AND the time condition is true.
- Use *FCGetProcessData on page 156*, too see if the condition was met or timed out

More examples

Example 1

```
FCCondOrient \MaxRot:= 45, 60;
```

In this example, the orientation condition coordinate system is set to the same as the tool coordinate system at the time of execution of this instruction. When this condition is activated the program execution will wait until the tool's rotation around the z axis deviates more than 45 degrees from the condition coordinate system, or until 60 seconds has passed.

Example 2

```
VAR orient my_orient:=[0,0,1,0];
FCCondOrient \OrientCondFrame:=my_orient \MaxAngle:= 30, 60;
```

In this example, the z direction of the orientation condition coordinate system is in negative z direction of the work object coordinate system. When this condition is activated the program execution will wait until the tool's z direction deviates more than 30 degrees from the z direction of the condition coordinate system. If this does not happen within 60 seconds there is a time-out.

Example 3

```
VAR orient my_orient:=[0,0,1,0];
FCCondOrient \OrientCondFrame:=my_orient \MaxAngle:=15 \MaxRot:=45,
     60;
```

In this example, the z direction of the orientation condition coordinate system is in negative z direction of the work object coordinate system.

When this condition is activated the program execution will wait until the first of the following occurs:

- The tool's z direction deviates more than 30 degrees from the z direction of the orientation condition coordinate system.
- The tool's rotation around the z axis deviates more than 45 degrees from the orientation condition coordinate system.
- 60 seconds has passed.

Syntax

```
FCCondOrient
  ['\'OrientCondFrame':=' <expression (IN) of orient>]
  ['\'MaxAngle':=' <expression (IN) of num>]
  ['\'MaxRot':=' <expression (IN) of num>]
  ['\'Outside]','
  [TimeOut':='] <expression (IN) of num>';'
```

Continues on next page

Related information

For information about	See
Setting up position condition	*FCCondPos on page 76*
Setting up TCP speed condition	*FCCondTCPSpeed on page 82*
Setting up reorientation speed condition	*FCCondReoriSpeed on page 79*
Setting up force condition	*FCCondForce on page 69*
Setting up torque condition	*FCCondTorque on page 85*
Activating previous conditions	*FCCondWaitWhile on page 88*

7.1.6 FCCondPos

Usage

`FCCondPos` is used to set up an end condition for the TCP position. The condition is later activated by calling the instruction `FCCondWaitWhile`, which will wait and hold the program execution while the specified condition is true. This allows the reference force, torque and movement to continue until the specified position is outside the specified limits.

A position condition is set up by defining a volume in space for the TCP position. Once activated the measured TCP position has to be within the specified volume (or outside, if the argument `Outside` is used).

Basic example

```
VAR fcboxvol my_box:= [-100, 100, -200, 200, -300, 300];
FCCondPos \Box:= my_box, 60;
```

When this condition is activated the program execution will wait until the robot TCP is outside the defined box or until 60 seconds has passed.

See also *More examples on page 77*.

Arguments

```
FCCondPos [\PosCondFrame] [\Box] | [\Cylinder] | [\Sphere]
     [\Outside] TimeOut
```

`[\PosCondFrame]`

Position condition coordinate system

Data type: `pose`

`PosCondFrame` is used to set the coordinate system in which the TCP position condition is defined. The coordinate system is set by a pose in relation to the work object coordinate system. The default value is [(0,0,0),(1,0,0,0)], meaning that if the parameter is omitted the position TCP condition is defined in the work object coordinate system.

`[\Box]`

Data type: `fcboxvol`

Defines a box-shaped volume. The position condition is by default true when the TCP is inside the box. If the argument Outside is set, the condition is true when the TCP is outside the box.

One, and only one, of the arguments `Box`, `Cylinder` and `Sphere` must be used.

`[\Cylinder]`

Data type: `fccylindervol`

Defines a cylinder-shaped volume. The position condition is by default true when the TCP is inside the cylinder. If the argument Outside is set, the condition is true when the TCP is outside the cylinder.

One, and only one, of the arguments `Box`, `Cylinder` and `Sphere` must be used.

Continues on next page

`[\Sphere]`

Data type: `fcspherevol`

Defines a sphere-shaped volume. The position condition is by default true when the TCP is inside the sphere. If the argument Outside is set, the condition is true when the TCP is outside the sphere.

One, and only one, of the arguments `Box`, `Cylinder` and `Sphere` must be used.

`[\Outside]`

Data type: `switch`

Makes the condition true when the TCP is outside the specified volume.

`TimeOut`

Data type: `num`

This is the maximum time the condition is valid, in seconds. If the position condition has not turned false before this time, the wait is interrupted and the next RAPID instruction is executed.

Program execution

Execution behavior:

- A time condition must be given (the argument `TimeOut`). The condition is considered true as long as the position condition AND the time condition is true.
- Use *FCGetProcessData on page 156*, too see if the condition was met or timed out

More examples

```
VAR fccylindervol my_cyl:= [300, 0, -200, 500, 250];
VAR pose my_cs := [[0,0,600],[0.7071,0,0.7071,0]];
FCCondPos \PosCondFrame := my_cs \Cylinder:=my_cyl, 60;
```

In this example the cylinder is not directly specified in the work object coordinate system but in a new coordinate system defined in relation to the work object coordinate system.

Syntax

```
FCCondPos
  ['\'PosCondFrame':=' <expression (IN) of datatype pose>]
  ['\'Box':=' <expression (IN) of datatype fcboxvol>]
  ['\'Cylinder':=' <expression (IN) of datatype fccylindervol>]
  ['\'Spehere':=' <expression (IN) of datatype fcspherevol>]
  ['\'Outside':=']','
  [TimeOut':='] <expression (IN) of num>';'
```

Related information

For information about	See
The data type `fcboxvol`	*fcboxvol on page 163*
The data type `fccylindervol`	*fccylindervol on page 167*

Continues on next page

For information about	See
The data type `fcspherevol`	*fcspherevol on page 181*
Setting up orientation condition	*FCCondOrient on page 72*
Setting up TCP speed condition	*FCCondTCPSpeed on page 82*
Setting up force condition	*FCCondForce on page 69*
Setting up reorientation speed condition	*FCCondReoriSpeed on page 79*
Activating previous conditions	*FCCondWaitWhile on page 88*

7.1.7 FCCondReoriSpeed

Usage

`FCCondReoriSpeed` is used to setup an end condition for the reorientation speed. The condition is later activated by calling the instruction `FCCondWaitWhile`, which will wait and hold the program execution while the specified condition is true. This allows the reference force, torque and movement to continue until the reorientation speed is outside the specified limits.

A reorientation speed condition is setup up by defining minimum and maximum limits for the TCP reorientation speed in all directions of the work object. Once activated with `FCCondWaitWhile`, the program execution will wait while the measured reorientation speed is within its specified limits. If the argument `Outside` is set, the execution will wait while the reorientation speed is outside the limits.

The condition on the reorientation speed is specified in the work object coordinate system.

Basic example

```
FCCondReoriSpeed \XMin:=-50 \XMax:=50, 60;
```

Defines a reorientation speed limit condition that is fulfilled if the reorientation speed around work object's x direction is between -50 degrees per seconds and 50 degrees per second. No restriction is put on the reorientation speed in other directions.

When this condition is activated the program execution will wait until the measured speed is outside its specified limits or until 60 seconds has passed.

Arguments

```
FCCondReoriSpeed [\XMin] [\XMax] [\YMin] [\YMax] [\ZMin] [\ZMax]
     [\Outside] TimeOut
```

`[\XMin]`

Minimum reorientation speed around the x direction

Data type: `num`

Lower reorientation speed limit around the work object's x direction. A negative value limits the maximum reorientation speed in the negative x direction.The unit is degrees per second and the default value is negative infinity.

`[\XMax]`

Maximum reorientation speed around the x direction

Data type: `num`

Upper reorientation speed limit around the work object's x direction. A negative value limits the minimum reorientation speed in the negative x direction.The unit is degrees per second and the default value is positive infinity.

`[\YMin]`

Minimum reorientation speed around the y direction

Data type: `num`

Continues on next page

Lower reorientation speed limit around the work object’s y direction. A negative value limits the maximum reorientation speed in the negative y direction.The unit is degrees per second and the default value is negative infinity.

`[\YMax]`

Maximum reorientation speed around the y direction

Data type: `num`

Upper reorientation speed limit around the work object’s y direction. A negative value limits the minimum reorientation speed in the negative y direction.The unit is degrees per second and the default value is positive infinity.

`[\ZMin]`

Minimum reorientation speed around the z direction

Data type: `num`

Lower reorientation speed limit around the work object’s z direction. A negative value limits the maximum reorientation speed in the negative z direction.The unit is degrees per second and the default value is negative infinity.

`[\ZMax]`

Maximum reorientation speed around the z direction

Data type: `num`

Upper reorientation speed limit around the work object’s z direction. A negative value limits the minimum reorientation speed in the negative z direction.The unit is degrees per second and the default value is positive infinity.

`[\Outside]`

Data type: `switch`

Specify that the condition is fulfilled when the speed is outside the specified limits.

`TimeOut`

Data type: `num`

This is the maximum time the condition is valid, in seconds. If the force condition has not turned false before this time, the wait is interrupted and the next RAPID instruction is executed.

Program execution

Execution behavior:

- A time condition must be given (the argument TimeOut). The condition is considered true as long as the force condition AND the time condition is true.
- Use *FCGetProcessData on page 156*, too see if the condition was met or timed out

Syntax

```
FCCondReoriSpeed
  ['\'XMin':=' <expression (IN) of num>]
  ['\'XMax':=' <expression (IN) of num>]
  ['\'YMin':=' <expression (IN) of num>]
  ['\'YMax':=' <expression (IN) of num>]
```

Continues on next page

```
['\'ZMin':=' <expression (IN) of num>]
['\'ZMax':=' <expression (IN) of num>]
['\'Outside]','
[TimeOut':='] <expression (IN) of num>';'
```

Related information

For information about	See
Setting up orientation condition	*FCCondOrient on page 72*
Setting up TCP speed condition	*FCCondTCPSpeed on page 82*
Setting up force condition	*FCCondForce on page 69*
Setting up torque condition	*FCCondTorque on page 85*
Activating previous condition	*FCCondWaitWhile on page 88*

7.1.8 FCCondTCPSpeed

Usage

`FCCondTCPSpeed` is used to setup an end condition for the TCP speed. The condition is later activated by calling the instruction `FCCondWaitWhile`, which will wait and hold the program execution while the specified condition is true. This allows the reference force, torque and movement to continue until the speed is outside the specified limits.

A TCP speed condition is setup up by defining minimum and maximum limits for the TCP speed in all directions of the work object. Once activated with `FCCondWaitWhile`, the program execution will continue to wait while the measured speed is within its specified limits.

It is possible to specify that the condition is fulfilled when the speed is outside the specified limits instead. This is the done by using the switch argument `Outside`.

The condition on TCP speed is specified in the work object coordinate system.

Basic example

```
FCCondTCPSpeed \Xmin:=100 \Xmax:=100, 60;
```

Defines a speed limit condition that is true if the speed in work object’s x direction is between -100 and 100 mm/s. No restriction is put on the speed in other directions.

When this condition is activated the program execution will wait until the measured speed in the x direction is outside its specified limits, or until 60 seconds has passed.

See also *More examples on page 84*.

Arguments

```
FCCondTCPSpeed [\XMin] [\XMax] [\YMin] [\YMax] [\ZMin] [\ZMax]
     [\Outside] TimeOut
```

`[\XMin]`

Minimum speed in the x direction

Data type: `num`

Lower limit for TCP speed in the x direction of the work object coordinate system. A negative value limits the maximum speed in the negative x direction.

The unit is mm/s and the default value is negative infinity.

`[\XMax]`

Maximum speed in the x direction

Data type: `num`

Upper limit for TCP speed in the x direction of the work object coordinate system. A negative value limits the minimum speed in negative x direction.

The unit is Newton and the default value is positive infinity.

`[\YMin]`

Minimum speed in the y direction

Data type: `num`

Continues on next page

Lower limit for TCP speed in the y direction of the work object coordinate system. A negative value limits the maximum speed in the negative y direction.

The unit is mm/s and the default value is negative infinity.

`[\YMax]`

Maximum speed in the y direction

Data type: `num`

Upper limit for TCP speed in the y direction of the work object coordinate system. A negative value limits the minimum speed in negative y direction.

The unit is Newton and the default value is positive infinity.

`[\ZMin]`

Minimum speed in the z direction

Data type: `num`

Lower limit for TCP speed in the z direction of the work object coordinate system. A negative value limits the maximum speed in the negative z direction.

The unit is mm/s and the default value is negative infinity.

`[\ZMax]`

Maximum speed in the z direction

Data type: `num`

Upper limit for TCP speed in the z direction of the work object coordinate system. A negative value limits the minimum speed in negative z direction.

The unit is Newton and the default value is positive infinity.

`[\Outside]`

Data type: `switch`

Specify that the condition is fulfilled when the speed is outside the specified limits.

`TimeOut`

Data type: `num`

This is the maximum time the condition is valid, in seconds. If the speed condition has not turned false before this time, the wait is interrupted and the next RAPID instruction is executed.

Program execution

Execution behavior:

- A time condition must be given (the argument `TimeOut`). The condition is considered true as long as the TCP speed condition AND the time condition is true.
- Use *FCGetProcessData on page 156*, too see if the condition was met or timed out

Continues on next page

More examples

Example 1

```
FCCondTCPSpeed \Xmin:=-50 \Xmax:=50 \Ymin:=-50 \Ymax:=50 \Zmin:=-50
     \Zmax:=50 \Outside, 60;
```

In this example limits for the speed are specified, but by setting the switch `Outside` the condition is fulfilled when the speed is outside the specified limits. That is, the speed needs to be larger than 50 mm/s or smaller than -50 mm/s for one of the specified directions.

When this condition is activated the program execution will wait until the measured speed is between -50 and 50 mm/s in all directions, or until 60 seconds has passed.

Syntax

```
FCCondTCPSpeed
  [ '\' XMin ':=' < expression (IN) of num > ]
  [ '\' XMax ':=' < expression (IN) of num > ]
  [ '\' YMin ':=' < expression (IN) of num > ]
  [ '\' YMax ':=' < expression (IN) of num > ]
  [ '\' ZMin ':=' < expression (IN) of num > ]
  [ '\' ZMax ':=' < expression (IN) of num > ]
  [ '\' Outside ] ','
  [ TimeOut ':=' ] < expression (IN) of num > ';'
```

Related information

For information about	See
Setting up position condition	*FCCondPos on page 76*
Setting up orientation condition	*FCCondOrient on page 72*
Setting up reorientation speed condition	*FCCondReoriSpeed on page 79*
Setting up force condition	*FCCondForce on page 69*
Setting up torque condition	*FCCondTorque on page 85*
Activating previous set condition	*FCCondWaitWhile on page 88*

7.1.9 FCCondTorque

Usage

`FCCondTorque` is used to set up an end condition for torque. The condition is later activated by calling the instruction `FCCondWaitWhile`, which will wait and hold the program execution while the specified condition is true. This allows the reference force, torque and movement to continue until the torque is outside the specified limits.

A torque condition is setup up by defining minimum and maximum limits for the torque in the directions of the force control coordinate system. Once activated with `FCCondWaitWhile`, the program execution will continue to wait while the measured torque is within its specified limits.

It is possible to instead specify that the condition is fulfilled when the torque is outside the specified limits. This is done by using the switch argument `Outside`.

The condition on torque is specified in the force control coordinate system. This coordinate system is setup by the user in the instruction `FCAct`.

Basic example

```
FCCondTorque \XMin:=-100 \XMax:=100, 60;
```

Defines a torque condition that is true when the torque in the x direction of the force control coordinate system is between -100 Nm and 100 Nm. No restriction is put on the torque in other directions.

When this condition is activated the program execution will wait until the measured torque is outside its limits, or until 60 seconds has passed.

See also *More examples on page 87*.

Arguments

```
FCCondTorque [\XMin] [\XMax] [\YMin] [\YMax] [\ZMin] [\ZMax]
     [\Outside] TimeOut
```

`[\XMin]`

Minimum torque in x direction

Data type: `num`

Lower limit for torque in the x direction of the force control coordinate system. A negative value limits the maximum torque in the negative x direction.

The unit is Nm and the default value is negative infinity.

`[\XMax]`

Maximum torque in x direction

Data type: `num`

Upper limit for torque in the x direction of the force control coordinate system. A negative value limits the minimum torque in negative x direction.

The unit is Nm and the default value is positive infinity.

`[\YMin]`

Minimum torque in y direction

Continues on next page

Data type: `num`

Lower limit for torque in the y direction of the force control coordinate system. A negative value limits the maximum torque in the negative y direction.

The unit is Nm and the default value is negative infinity.

`[\YMax]`

Maximum torque in y direction

Data type: `num`

Upper limit for torque in the y direction of the force control coordinate system. A negative value limits the minimum torque in negative y direction.

The unit is Nm and the default value is positive infinity.

`[\ZMin]`

Minimum torque in z direction

Data type: `num`

Lower limit for torque in the z direction of the force control coordinate system. A negative value limits the maximum torque in the negative z direction.

The unit is Nm and the default value is negative infinity.

`[\ZMax]`

Maximum torque in z direction

Data type: `num`

Upper limit for torque in the z direction of the force control coordinate system. A negative value limits the minimum torque in negative z direction.

The unit is Nm and the default value is positive infinity.

`[\Outside]`

Data type: `switch`

Specify that the condition is fulfilled when the torque is outside the specified limits.

`TimeOut`

Data type: `num`

This is the maximum time the condition is valid, in seconds. If the torque condition has not turned false before this time, the wait is interrupted and the next RAPID instruction is executed.

Program execution

Execution behavior:

- A time condition must be given (the argument `TimeOut`). The condition is considered true as long as the torque condition AND the time condition is true.
- Use *FCGetProcessData on page 156*, too see if the condition was met or timed out

Continues on next page

More examples

Example 1

```
FCCondTorque \XMin:=-10 \XMax:=10 \YMin:=-10 \YMax:=10 \ZMin:=-10
     \ZMax:=10 \Outside, 60;
```

In this example, torque limits in all directions of the force control coordinate system are specified. The switch `Outside` set, which means that the condition is fulfilled as long as the torque is outside the specified limits. That is, the torque needs to be larger than 10 Nm or smaller than -10 Nm for one of the specified directions.

When this condition is activated the program execution will wait until the measured torque is between -10 and 10 Nm in all directions, or until 60 seconds has passed.

Syntax

```
FCCondTorque
  [ '\' XMin ':=' < expression (IN) of num > ]
  [ '\' XMax ':=' < expression (IN) of num > ]
  [ '\' YMin ':=' < expression (IN) of num > ]
  [ '\' YMax ':=' < expression (IN) of num > ]
  [ '\' ZMin ':=' < expression (IN) of num > ]
  [ '\' ZMax ':=' < expression (IN) of num > ]
  [ '\' Outside ] ','
  [ TimeOut ':=' ] < expression (IN) of num > ';'
```

Related information

For information about	See
Setting up position condition	*FCCondPos on page 76*
Setting up orientation condition	*FCCondOrient on page 72*
Setting up TCP speed condition	*FCCondTCPSpeed on page 82*
Setting up reorientation speed condition	*FCCondReoriSpeed on page 79*
Setting up force condition	*FCCondForce on page 69*
Activating previous set condition	*FCCondWaitWhile on page 88*

7.1.10 FCCondWaitWhile

Usage

`FCCondWaitWhile` is used to activate previously set up conditions.This instruction will wait and hold the program execution while the specified condition is true. This allows the reference force, torque and movement to continue until a specified condition limit is reached.

Basic example

```
FCRefLine FC_LIN_X, 500, 100;
FCCondForce \Zmax:=10, 60;
FCAct;
FCRefStart;
FCCondWaitWhile;
```

In this example, `FCCondWaitWhile` activates the force condition. The execution will wait and the reference movement will continue until the force in positive z direction is above 10 N, or until 60 seconds has passed.

See also *More examples on page 88*.

Arguments

```
FCCondWaitWhile [\ZeroRefAtEnd]
```

`[\ZeroRefAtEnd]`

Zero reference at end

Data type: `switch`

If this argument is used all references will be switched off once the condition turns false and the wait is over.

More examples

Example 1

```
FCRefLine FC_LIN_X, 500, 100;
FCCondForce \Zmin:=10, 60;
FCAct;
FCRefStart;
FCCondWaitWhile \ZeroRefAtEnd;
```

Same as the basic example except here the system will immediately turn off the reference movement once the wait is over.

Syntax

```
FCCondWaitWhile
  [ '\' ZeroRefAtEnd ] ';'
```

Related information

For information about	See
Setting up position condition	*FCCondPos on page 76*
Setting up orientation condition	*FCCondOrient on page 72*

Continues on next page

For information about	See
Setting up TCP speed condition	*FCCondTCPSpeed on page 82*
Setting up reorientation speed condition	*FCCondReoriSpeed on page 79*
Setting up force condition	*FCCondForce on page 69*
Setting up torque condition	*FCCondTorque on page 85*

7.1.11 FCDeact

Usage

`FCDeact` is used to deactivate Force Control. After a successful deactivation the robot is back in position control.

Basic example

```
FCDeact;
```

Deactivates Force Control.

Arguments

```
FCDeact
```

There are no arguments to the instruction `FCDeact`.

Program execution

When running the instruction `FCDeact` the robot goes from being force controlled to being position controlled.

Limitations

The force control can only be deactivated if:

- The reference values are deactivated by `FCRefStop` , or if the cfg parameter *Keep contact force at stop* in FC Application is set.
- There are no robot movements.

Syntax

```
FCDeact ';'
```

Related information

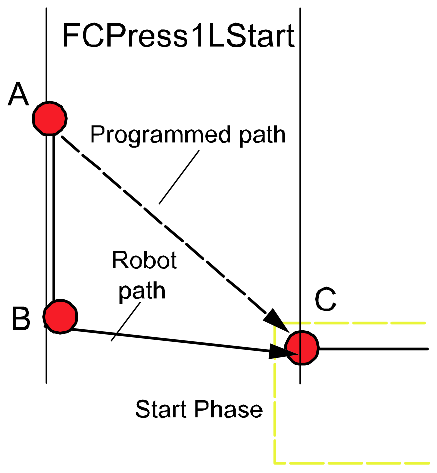
For information about	See
Activating force control	*FCAct on page 63*

7.1.12 FCPress1LStart

Usage

`FCPress1LStart` is used to make contact to a surface and move the tool centre point (TCP) linearly to a first given destination. The following contact movement should be done with the `FCPressL` instruction. Instruction starts a sequence for regulation in one direction on force. If you want to follow a corner or otherwise follow in more than one dimension this instruction should not be used.

Move close to the contact point (NOT IN CONTACT) The point `ToPoint` is where the first move in contact from the contact point will go. ForceThreshold is the parameter that will have to be tuned for best result.



xx0600003285

(A) is the point close to contact. The force reference will move us to point (B) which is a point when we have contact. The movement to point C (ToPoint) will start when the force has reached a certain level defined by the parameter ForceThreshold (in % of ordered force)

Instruction sets up some data that is true for a sequence. The force is set in each coordinate direction, this will result in one force in one direction calculated from these settings. This direction of force definition is true for all following FCPress-move instructions and will be until `FCPressEnd`. All parameters in `FCPress1LStart` except `ToPoint`, `Speed`, `Force` and `Zone` is true until `FCPressEnd`.

Additionally, there is an option to compensate for the robots deflection under the impact of the force. This option requires the setup of a force sensor. Deflection compensation will be most accurate for robots with the Absolute Accuracy option, even though it can also run without. The user should not change the tool from the last move instruction to the activation of deflection compensation.

Note

The distance B to C must be more than 100 ms, or else C will become a fine point.

Continues on next page

Basic example

Basic example of the instruction `FCPress1LStart` is illustrated below.

See also *FCPress1LStart on page 91*.

Example

```
FCPress1LStart p10, v100 \Fz:=200, 57, z30, tool1
```

Move in positive force direction (= z) until 57% (= 114N) of force is reached and then start toward p10 while force builds up to 100% (=200N).

Arguments

```
FCPress1LStart ToPoint [\ToNextPoint] Speed [\Fx] [\Fy] [\Fz]
     ForceThreshold [\ForceFrameRef] [\ForceChange] [\DampingTune]
     [\TimeOut] [\UseSpdFFW] [\DeflectionCompGain] [\PosSupvDist]
     Zone Tool [\WObj]
```

`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).

`[\ToNextPoint]`

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). This should only be used if the robtarget in the first `FCPressL` is to close and therefore becomes a corner path failure.

`Speed`

Data type: `speeddata`

The speed data that applies to movements. Speed data defines the velocity for the tool centre point, the tool reorientation and external axes.

`[\Fx]`

Reference force in x direction

Data type: `num`

Defines the constant reference force in the x direction of the force control coordinate system.If this argument is omitted there will be zero contact force in the x direction.

`[\Fy]`

Reference force in y direction

Data type: `num`

Defines the constant reference force in the y direction of the force control coordinate system.If this argument is omitted there will be zero contact force in the y direction.

`[\Fz]`

Reference force in z direction

Data type: `num`

Continues on next page

Defines the constant reference force in the z direction of the force control coordinate system.If this argument is omitted there will be zero contact force in the z direction.

`ForceThreshold`

Data type: `num`

Percentage of the contact force that should be reached before move toward the robtarget starts. TCP moves in force direction until this percentage is reached. When percentage of force is reached, the movement toward target starts.

`[\ForceFrameRef]`

Data type: `fcframe`

`ForceFrameRef` here defines which coordinate system the force control coordinate system is related to. The parameter can be set to either the work object coordinate system, the tool coordinate system or the path coordinate system described in section *Programming in path coordinates on page 58*.

FC_REFFRAME_WOBJ, FC_REFFRAME_TOOL or FC_REFFRAME_PATH.

The default value is the tool coordinate system.

`[\ForceChange]`

Data type: `num`

Tuning parameter to ramp up force. Unit is [N/s]. This argument overrides configured value.

`[\DampingTune]`

Data type: `num`

`DampingTune` is the relation value between the measured force and the applied resulting force. By default the value are 100% (of system parameter values), but it can be between 50% and infinity. Smaller values than 100% means that the robot is more sensitive to external force

`[\TimeOut]`

Data type: `num`

If force hasn't build up before this time is reached then continue with next instruction. Unit is [s].

`[\UseSpdFFW]`

Data type: `switch`

If this argument is used then feed forward regulation is used. If argument isn't used then regulation in force direction is done only with force control without help from programmed path. Use this argument if path is complex and programmed path is close to actual path.

`[\DeflectionCompGain]`

Data type: `num`

The `DeflectionCompGain` is scaling the correction of the robot arm's deflection under an applied force. Default value is 1. If it is not set or set to 0, the deflection

Continues on next page

compensation is not activated. Higher gains will make the correction more aggressive and should be used with care.

Note

Deflection compensation will be active until `PressEnd`.

`[\PosSupvDist]`

Data type: `num`

The robot will stop if it has moved more than the distance `PosSupvDist` away from the programmed path. Default value is 20mm. Unit is [mm].

`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 used during Force Control. It is the center point of this tool that is used for all calculations. Note that the dimensions of the sensor and any interface plates need to be included in the tool definition. To change tool force control has to be deactivated.

`[\WObj]`

Data type: `wobjdata`

The work object (coordinate system) to which the robot position in the instruction is related. This argument can be omitted, and if it is, 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. To change work object force control has to be deactivated.

Note

The coordinate systems mentioned in the arguments list are described in *The coordinate systems on page 249*.

Program execution

Execution behavior:

- `FCPress1LStart` activates Force Control.

More examples

More examples of how to use the instruction `FCPress1LStart` are illustrated below.

Example 1

```
VAR tooldata tool1:=[TRUE,[[97.4,0,223],[1,0,0,0]],
      [5,[23,0,75],[1,0,0,0],0,0,0]];
```

Continues on next page

```
FCPress1LStart p10, v100 \Fy:=-100, 57
     \ForceFrameRef:=FC_REFFRAME_TOOL, z30, tool1;
```

Activates Force Control and defines a force control coordinate system based on the tool1 coordinate system with force on negative y axis.

Example 2

```
VAR tooldata tool1:=...
VAR wobjdata my_wobj :=
     [FALSE,TRUE,"",[[0,0,0],[,0,0,0]],[[0,0,0],[0.07071,0,0.7071,0]]];
FCPress1LStart p10, v100 \Fz:=200, 57,
     \ForceFrameRef:=FC_REFFRAME_WOBJ \ForceChange:=200, z30, tool1
     \WObj:=my_wobj;
```

Activates Force Control with:

- tool tool1
- force control coordinate system orientation equal to the orientation of the work object my_wobj.
- maximum force change to 200 Newton per second

Example 3

```
VAR tooldata tool1:=...

VAR wobjdata my_wobj :=
[FALSE,TRUE,"",[[0,0,0],[,0,0,0]],[[0,0,0],[0.07071,0,0.707 1,0]]];

FCPress1LStart p10, v100 \Fz:=200, 57,
     \ForceFrameRef:=FC_REFFRAME_WOBJ \ForceChange:=200
     \PosSupvDist:=100 , z30, tool1 \WObj:=my_wobj;
```

This example will behave like example 2 unless the robot would deviate from the programmed path. In example 2 the robot will stop if it deviates more then 20 mm, and in this example the robot may drift 100 mm before it stops.

Example 4

```
VAR wobjdata my_wobj :=
[FALSE,TRUE,"", [[0,0,0],[,0,0,0]], [[0,0,0],[0.07071,0,0.707
     1,0]]];

MoveAbsJ jpos10 \NoEOffs,v50,fine, tool1\WObj:=my_wobj;

FCPress1LStart p10, v50\Fz:=100, 20 \ForceFrameRef:=FC_REFFRAME_TOOL
     \DampingTune:=100 \DeflectionCompGain:=1 \PosSupvDist:=50,
     fine, tool1;
```

This example will activate deflection compensation which will make the robot follow the programmed path more accurately.

Limitations

The Force Control will only behave correctly if the load is identified with `FCLoadID` and the sensor is calibrated with `FCCalib` before activating Force Control with `FCPress1LStart` FCPress1-move instructions can only be used between `FCPress1LStart` and `FCPressEnd`.

Continues on next page

Syntax

```
FCPress1LStart
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ToNextPoint ':=' < expression (IN) of robtarget > ] ','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' Fx ':=' < expression (IN) of num > ]
  [ '\' Fy ':=' < expression (IN) of num > ]
  [ '\' Fz ':=' < expression (IN) of num > ] ','
  [ ForceThreshold ':=' ] < expression (IN) of num >
  [ '\' ForceFrameRef ':=' < expression (IN) of fcframe > ]
  [ '\' ForceChange ':=' < expression (IN) of num > ]
  [ '\' DampingTune ':=' < expression (IN) of num > ]
  [ '\' TimeOut ':=' < expression (IN) of num > ]
  [ '\' UseSpdFFW ] ','
  [ '\' DeflectionCompGain '=' < expression (IN) of num > ]
  [ '\' PosSupvDist ':=' < expression (IN) of num > ]
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ';'
```

Related information

For information about	See
Linear one dimensional press instruction	*FCPressL on page 102*
Circular one dimensional press instruction	*FCPressC on page 97*
End the press instruction	*FCPressEnd on page 99*

7.1.13 FCPressC

Usage

`FCPressC` is used to move the tool center point (TCP) circular to a given destination and during this movement a contact force can be maintained to a surface

Basic examples

Example

```
VAR num Force=60;
FCPressC p10, p20, v100, Force, z30, tool0;
```

Move circularly to point `p20` with speed `v100` and a contact force of 60 N in the direction decided by the `FCPress1LStart` instruction. The Circle is defined by the start position, the circle point `p10` and the destination point `p20`.

Arguments

```
FCPressC CirPoint ToPoint Speed Force Zone Tool [\Wobj]
```

CirPoint

Circle point

Data type: `robtarget`

The circle 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).

`ToPoint`

Destination point

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`

The speed of the TCP

Data type: `speeddata`

The speed data that applies to movements. Speed data defines the velocity for the tool centre point, the tool reorientation and external axes.

`Force`

Data type: `num`

The force size in the direction defined in the coordinate system chosen in `FCPress1LStart`.

`Zone`

Data type: `zonedata`

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

`Tool`

Data type: `tooldata`

Continues on next page

The tool is used during Force Control. Note that this must be the same tool that is used in the `FCPress1LStart` instruction.

This argument is only present due to offline programming purposes.

`[\Wobj]`

Data type: `wobjdata`

The work object (coordinate system) to which the robot position in the instruction is related to. Note that this must be the same work object that is used in the `FCPress1LStart` instruction.

This argument is only present due to offline programming purposes.

Program execution

Execution behavior:

- `FCPressC` moves toward target in contact with surface at the specified force. Movement will follow the surface and as a result the path will not be completely circular.

Limitations

Instruction can only be used between a `FCPress1LStart` and a `FCPressEnd`. Tool and WObj cannot be changed while force control is active. Tool and work object are set in `FCPress1LStart` and cannot be changed until after `FCPressEnd` and a new `FCPress1LStart`.

Syntax

```
FCPressC
  [ CirPoint':=' ] < expression (IN) of robtarget> ','
  [ ToPoint':=' ] < expression (IN) of robtarget> ','
  [ Speed ':=' ] < expression (IN) of speeddata > ','
  [ Force':=' ] < expression (IN) of num > ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':='] < persistent (PERS) of tooldata >
  ['\' Wobj ':=' < persistent (PERS) of wobjdata > ] ';'
```

Related information

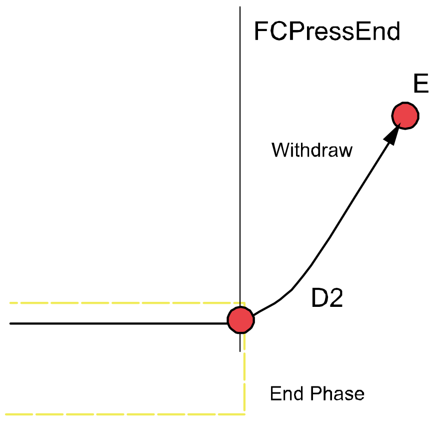
For information about	See
Start press instruction.	*FCPress1LStart on page 91*
Linear one dimensional press instruction.	*FCPressL on page 102*
End press instruction.	*FCPressEnd on page 99*

7.1.14 FCPressEnd

Usage

`FCPressEnd` is used to release the contact from the `FCPress1LStart` and `FCPressL`

When calling this function the position is D2 which is a point where a contact force is present. The user specifies a point E which should be close to contact but NOT IN contact



xx0600003286

The same force reference as was used in `FCPress1LStart` (but with different sign) will start to reduce the contact force giving an upward movement. At the same time the move instruction to point E will give a horizontal movement

When the horizontal movement is finished, force control is switch off and position control will move to point E.

Note

The horizontal (orthogonal to force direction) distance between point (D2) and (E) should be short.

Basic examples

Basic example of the instruction `FCPressEnd` is illustrated below.

See also *More examples on page 100*.

Example

```
FCPressEnd p10, v100, tool0;
```

Move to p10 with 100mm/s and on the way when force is zero turn off force control.

Arguments

```
FCPressEnd ToPoint Speed [\ForceChange] [\ZeroContactValue |
     DeactOnly] Tool [\Wobj]
```

`ToPoint`

Data type: `robtarget`

Continues on next page

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 for the tool centre point, the tool reorientation and external axes.

`[\ForceChange]`

Data type: `num`

Tuning parameter to ramp up force. Value in argument is given in [N/s].

`[\ZeroContactValue | DeactOnly]`

Data type: `num | switch`

Two options are available to deactivate force control. The first is to specify a force limit and if the force is less than the argument `ZeroContactValue`, then force control is deactivated.

The second option, `DeactOnly`, deactivates force control when the reference force is less than zero force. This option can be useful when experiencing a lot of disturbances from a vibrating tool when measuring the force. This option can also be used when having problems due to roughly calibrated work objects.

If the arguments are not used, a default value defined in the system parameters will be used.

`Tool`

Data type: `tooldata`

The tool is used during Force Control. Note that this must be the same tool that is used in the `FCPress1LStart` instruction.

This argument is only present due to offline programming purposes.

`[\Wobj]`

Data type: `wobjdata`

The work object (coordinate system) to which the robot position in the instruction is related to. Note that this must be the same work object that is used in the `FCPress1LStart` instruction.

This argument is only present due to offline programming purposes.

Program execution

Execution behavior:

- `FCPressEnd` deactivates force control and switch to position control when force becomes less then `ZeroContactValue`.

More examples

More examples of how to use the instruction `FCPressEnd` are illustrated below.

Example 1

```
FCPressEnd p10, v100 \ForceChange:=100;
```

Continues on next page

Set max release speed of force to 100N/s and then deactivate Force Control.

Example 2

```
FCPressEnd p10, v100 \ZeroContactValue:=2.5;
```

When force is less than 2.5N then deactivate force control and continue move to p10.

Example 3

```
FCPressEnd p10, v100 \DeactOnly;
```

When the reference force is deactivated, the force control is also deactivated and the tool will leave the surface.

Limitations

Tool and work object are set in `FCPress1LStart` and cannot be changed until after `FCPressEnd` in a new `FCPress1LStart`.

Syntax

```
FCPressEnd
  [ToPoint ':='] <expression (IN) of robtarget>','
  [Speed ':='] <expression (IN) of speeddata>
  ['\' ForceChange ':='] <expression (IN) of num>
  ['\' ZeroContactValue ':='] <expression (IN) of num>','
  |['\' DeactOnly ':='] <expression (IN) of num>','
  [Tool ':='] <persistent (PERS) of tooldata>
  ['\' Wobj ':=' <persistent (PERS) of wobjdata>]';'
```

Related information

For information about	See
Start the press instruction	*FCPress1LStart on page 91*

7.1.15 FCPressL

Usage

`FCPressL` is used to move the tool centre point (TCP) linearly to a given destination and during this movement a contact force can be maintained to a surface.

Basic examples

Basic examples of the instruction `FCPressL` are illustrated below.

Example

```
VAR num Force=60;
FCPressL p10, v100, Force, z30, tool0;
```

Move to point `p10` with speed `v100` and a contact force of 60 N in the direction decided by the `FCPress1LStart` instruction.

Arguments

```
FCPressL ToPoint Speed Force Zone Tool [\Wobj]
```

`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 for the tool centre point, the tool reorientation and external axes.

`Force`

Data type: `num`

The force size in the direction defined in the coordinate system chosen in `FCPress1LStart`.

`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 is used during Force Control. Note that this must be the same tool that is used in the `FCPress1LStart` instruction.

This argument is only present due to offline programming purposes.

`[\Wobj]`

Data type: `wobjdata`

Continues on next page

The work object (coordinate system) to which the robot position in the instruction is related to. Note that this must be the same work object that is used in the `FCPress1LStart` instruction.

This argument is only present due to offline programming purposes.

Program execution

Execution behavior:

- `FCPressL` moves toward target in contact with surface at the specified force. Movement will follow the surface and as a result the path will not be completely linear

Limitations

Instruction can only be used between a `FCPress1LStart` and a `FCPressEnd`.

Tool and WObj cannot be changed while force control is active. Tool and work object are set in `FCPress1LStart` and cannot be changed until after `FCPressEnd` and a new `FCPress1LStart`.

Syntax

```
FCPressL
  [ ToPoint':=' ] < expression (IN) of robtarget> ','
  [ Speed ':=' ] < expression (IN) of speeddata > ','
  [ Force ':=' ] < expression (IN) of num > ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':='] < persistent (PERS) of tooldata >
  ['\' Wobj ':=' < persistent (PERS) of wobjdata > ] ';'
```

Related information

For information about	See
Start press instruction	*FCPress1LStart on page 91*
Circular one dimensional press instruction.	*FCPressC on page 97*
End press instruction.	*FCPressEnd on page 99*

7.1.16 FCRefCircle

Usage

FCRefCircle is used to specify a reference movement for Force Control. The purpose of a reference movement is usually to search through an area to try to find a fit.The instruction is only used to set up the reference movement, not to activate it. Activation is done with the instruction FCRefStart. Once activated, the robot will try to move according to the references (i.e. in a circle). This reference will not be enforced; if a contact force affects the robot the movement pattern will be hard to predict.

The circle is specified in the reference movement coordinate system. The coordinate system origin is the tool center point. Its orientation is normally the same as that of the work object coordinate system but can be changed by using the instruction FCRefMoveFrame.

Basic example

```
FCRefCircle FCPLANE_XY, 30, 100;
```

Sets up, but does not activate, a circular shaped reference movement in the XY plane. The speed is 30 degrees per second and the radius 100 mm.

Arguments

```
FCRefCircle Plane Speed Radius
```

Plane

Data type: fcplane

Specifies which plane the circle is defined in (FCPLANE_XY, FCPLANE_XZ or FCPLANE_YZ)

Speed

Data type: num

The speed of the circle movement. The unit is degrees per second

Radius

Data type: num

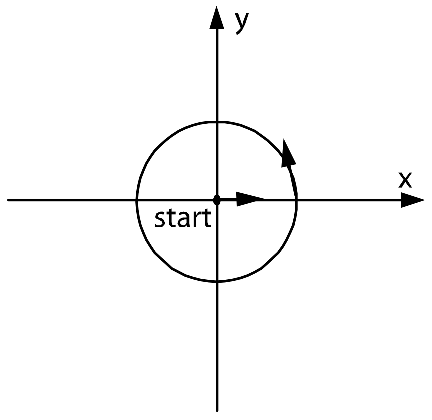
The radius of the circle. The unit is in mm.

Program execution

Execution behavior:

- The reference movement must be set up before activating the references with FCRefStart.
- The circular path starts in the middle of the circle, moves to the circle boundary and then moves counter clockwise.

Continues on next page



en0500002368

Syntax

```
FCRefCircle
  [ Plane ':=' ] < expression (IN) of fcplane > ','
  [ Speed ':=' ] < expression (IN) of num > ','
  [ Radius ':=' ] < expression (IN) of num > ';'
```

Related information

For information about	See
The data type plane	*fcplane on page 177*
Setting up linear reference movement	*FCRefLine on page 108*
Setting up spiral reference movement	*FCRefSpiral on page 114*
Activating reference values	*FCRefStart on page 119*
Deactivating reference values	*FCRefStop on page 121*

7.1.17 FCRefForce

Usage

`FCRefForce` is used to specify a reference force for Force Control. This instruction is only used to set up the reference force, not to activate it. Activation is done with the instruction `FCRefStart`.

Once activated the robot will start to move in order to achieve the reference force. The reference force is usually set up by using a constant force, but it is possible to use an oscillating reference force.

Basic example

```
FCRefForce \Fz:=10;
```

Sets up a constant reference force of 10 N in the positive z direction of the force control coordinate system.

See also *More examples on page 107*.

Arguments

```
FCRefForce [\Fx] [\Fy] [\Fz] [\Amp] [\Period]
```

`[\Fx]`

Reference force in x direction

Data type: `num`

Defines the constant reference force in the x direction of the force control coordinate system.

If this argument is omitted there will be zero contact force in the x direction

`[\Fy]`

Reference force in y direction

Data type: `num`

Defines the constant reference force in the y direction of the force control coordinate system.

If this argument is omitted there will be zero contact force in the y direction

`[\Fz]`

Reference force in z direction

Data type: `num`

Defines the constant reference force in the z direction of the force control coordinate system.

If this argument is omitted there will be zero contact force in the z direction

`[\Amp]`

Amplitude of force oscillation

Data type: `fcxyznum`

The magnitude of the optional oscillating part of the force reference, in the unit Newtons.

If `Amp` is used, `Period` should also be used.

Continues on next page

`[\Period]`

Period of force oscillation

Data type: `fcxyznum`

The period time for the optional oscillating part of the reference torque, in the unit seconds. If `Period` is used, `Amp` should also be used.

Program execution

Execution behavior:

- The reference force is specified in the force control coordinate system.
- The reference force must be set up before activating the references with `FCRefStart`.

More examples

Example 1

```
FCRefForce \Fy:=20 \Fz:=10;
```

Sets up a constant references force that has a 20 N component in the x direction and a 10N component in the z direction.

Example 2

```
VAR fcxyznum myAmp:=[0,0,10];
VAR fcxyznum myPeriod:=[0,0,1];
FCRefForce \Fz:=10 \Amp:=myAmp \Period:=myPeriod;
```

Sets up an oscillating reference force between 0 and 20 Newton in the positive z direction of the force control system.

Syntax

```
FCRefForce
  [ '\' Fx ':=' < expression (IN) of num > ]
  [ '\' Fy ':=' < expression (IN) of num > ]
  [ '\' Fz ':=' < expression (IN) of num > ]
  [ '\' Amp ':=' < expression (IN) of fcxyznum > ]
  [ '\' Period ':=' < expression (IN) of fcxyznum > ] ';'
```

Related information

For information about	See
Setting up torque reference	*FCRefTorque on page 122*
Activating reference values	*FCRefStart on page 119*
Deactivating reference values	*FCRefStop on page 121*
The data type `fcxyznum`	*fcxyznum on page 183*

7.1.18 FCRefLine

Usage

`FCRefLine` is used to specify a reference movement for Force Control. This instruction is only used to set up the reference movement, not to activate it. Activation is done with the instruction `FCRefStart`.

Once activated with `FCRefStart`, the robot will try to move according to the references (i.e. back and forth along a linear path). This reference will not be enforced, if a contact force affects a robot the movement pattern will be hard to predict.

The purpose of a reference movement is usually to search through an area to try to find a fit.

The line is specified in the reference movement coordinate system. This coordinate system's origin is the tool center point. Its orientation is normally the same as the orientation of the work object coordinate system but can be changed by using the instruction `FCRefMoveFrame`.

Basic example

```
FCRefLine FC_LIN_X, 500, 100;
```

Sets up, but does not activate, a linear shaped reference movement in the x-direction. The maximum speed is 500 mm/s and the distance peak to peak is 100 mm [amplitude +/-50 mm].

Arguments

```
FCRefLine Direction MaxSpeed Distance [\OneSideOfStartPos]
```

`Direction`

Data type: `fclindir`

Specifies which direction the reference is set in (FC_LIN_X, FC_LIN_Y, FC_LIN_Z).

`MaxSpeed`

Data type: `num`

The maximum speed of the linear movement. The unit is millimeters per second.

`Distance`

Data type: `num`

The amplitude of the movement. The TCP is oscillating between positive and negative value of the parameter `Distance` / 2. The unit is millimeters.

`[\OneSideOfStartPos]`

Data type: `switch`

This argument limits the movement to only one side of the start position. The side depends on the sign of the `MaxSpeed` argument.

Continues on next page

Program execution

Execution behavior:

- The reference movement must be set up before activating the references with `FCRefStart`
- A line in any of the linear directions is oscillating between the positive and the negative value of the parameter `Distance / 2` (i.e. the movement from one turning point to the other is `Distance`).

Syntax

```
FCRefLine
  [ Direction ':=' ] < expression (IN) of fclindir > ','
  [ Speed ':=' ] < expression (IN) of num > ','
  [ Distance ':=' ] < expression (IN) of num > ','
  ['\'OneSideOfStartPos] ';'
```

Related information

For information about	See
Setting up a reference movement coordinate system	*FCRefMoveFrame on page 110*
Setting up circular reference movement	*FCRefCircle on page 104*
Setting up rotational reference movement	*FCRefRot on page 112*
Activating reference values	*FCRefStart on page 119*
Deactivating reference values	*FCRefStop on page 121*

7.1.19 FCRefMoveFrame

Usage

`FCRefMoveFrame` is used to set up a coordinate system, in which reference movements can be defined. It is called the reference movement coordinate system.

The origin of this coordinate system is always the tool center point, but the user can specify orientation by using `FCRefMoveFrame`. The orientation is specified in relation to the orientation of the work object coordinate system or the tool coordinate system.

If no coordinate system is defined (i.e. `FCRefMoveFrame` is not used) the reference movement coordinate system has the same orientation as the work object coordinate system.

Basic example

```
VAR orient myOrient:= [0.924,0,0,0.383];
FCRefMoveFrame myOrient;
FCRefLine FC_LIN_X, 500, 100;
```

Without the coordinate system definition there would be a linear movement in the x direction of the work object. With the definition shown in this example there will be a linear movement in the xy direction of the work object.The x and y axes are moved clockwise 45 degrees around the z axis.

Arguments

```
FCRefMoveFrame [\RefMoveFrameRef][\RefMoveFrameOri]
```

`[\RefMoveFrameRef]`

Data type: `fcframe`

`RefMoveFrameRef` defines which coordinate system the reference coordinate system is related to. The parameter can be set to either the work object coordinate system or the tool coordinate system. The default value is the work object coordinate system.

`[\RefMoveFrameOri]`

Data type: `orient`

This parameter specifies the orientation from the coordinate system selected in `RefMoveFrameRef`. The default value is [1,0,0,0]. For information about how to calculate orientations, see the data type `orient` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

More examples

These scenarios illustrate a reference movement coordinate system related to the tool frame versus the work object frame.

Example 1 - Tool coord

```
FCRefMoveFrame \RefMoveFrameRef:=FC_REFFRAME_TOOL;
```

Example 2 - Work object coord

```
FCRefMoveFrame \RefMoveFrameRef:=FC_REFFRAME_WOBJ;
```

Continues on next page

Scenario 1

The tool frame and work object frame share the same orientation, the z -axis pointing upwards. If a rotation around the z-axis is started the result will be the same for Example 1and Example 2.

Scenario 2

If the orientation of the tool is changed, however, the result will no longer be the same. In Example 1 the tool will still rotate around the z axis of the tool. In Example 2, however, the tool will rotate in a cone shaped pattern.

Syntax

```
FCRefMoveFrame
  [ '\' RefMoveFrameRef ':=' < expression (IN) of fcframe > ]
  [ '\' RefMoveFrameOri ':=' < expression (IN) of orient > ] ';'
```

Related information

For information about	See
The data type `fcframe`	*fcframe on page 173*.
Setting up spiral reference movement	*FCRefSpiral on page 114*.
Setting up circular reference movement	*FCRefCircle on page 104*.
Setting up linear reference movement	*FCRefLine on page 108*.
Activating reference values	*FCRefStart on page 119*.
Deactivating reference values	*FCRefStop on page 121*.

7.1.20 FCRefRot

Usage

`FCRefRot` is used to specify a reference movement for Force Control. This instruction is only used to set up the reference movement, not to activate it. Activation is done with the instruction `FCRefStart`.

Once activated with `FCRefStart`, the robot will try to move according to the references (i.e. rotate around a chosen axis in a coordinate system). This reference will not be enforced, if a contact force affects a robot the movement pattern will be hard to predict.

The purpose of a reference movement is usually to search through an area to try to find a fit.

The rotation is specified in the reference movement coordinate system. The coordinate system origin is the tool center point. Its orientation is normally the same as the orientation of the work object coordinate system but can be changed by using the instruction `FCRefMoveFrame`.

Basic example

```
FCRefRot FC_ROT_Z, 5, 10;
```

Sets up a rotation around the work object z direction. When activated the TCP will rotate back and forth around the z-axis with a distance (peak to peak) of 10 degrees [amplitude +/- 5 degrees].The maximum speed will be 5 degrees per second.

Arguments

```
FCRefRot Direction MaxSpeed Distance [\OneSideOfStartPos]
```

`Direction`

Data type: `fcrotdir`

Specifies the direction of the rotation (`FC_ROT_X`,`FC_ROT_Y`,`FC_ROT_Z`).

`MaxSpeed`

Data type: `num`

The maximum speed of the rotational movement. The unit is degrees per second.

`Distance`

Data type: `num`

The amplitude of the movement. The TCP is oscillating between positive and negative value of the parameter `Distance /` 2. The unit is in degrees.

`[\OneSideOfStartPos]`

Data type: `switch`

This argument limits the movement to only one side of the start position. The side depends on the sign of the `MaxSpeed` argument.

Continues on next page

Program execution

Execution behavior:

- The reference movement must be set up before activating the references with `FCRefStart`.
- The rotation angle describes a sine function of the time with an amplitude `Distance` / 2 (i.e. the movement from one turning point to the other is `Distance`).

Syntax

```
FCRefRot
  [ Direction ':=' ] < expression (IN) of fcrotdir > ','
  [ Speed ':=' ] < expression (IN) of num > ','
  [ Distance ':=' ] < expression (IN) of num > ] ','
  ['\'OneSideOfStartPos] ';'
```

Related information

For information about	See
The data type `fcrotdir`	*fcrotdir on page 178*
Setting up spiral reference movement	*FCRefSpiral on page 114*
Setting up circular reference movement	*FCRefCircle on page 104*
Setting up linear reference movement	*FCRefLine on page 108*
Activating reference values	*FCRefStart on page 119*
Deactivating reference values	*FCRefStop on page 121*

7.1.21 FCRefSpiral

Usage

`FCRefSpiral` is used to specify a reference movement for Force Control. This instruction is only used to set up the reference movement, not to activate it. Activation is done with the instruction `FCRefStart`.

Once activated with `FCRefStart`, the robot will try to move according to the references (i.e. in a spiral). This reference will not be enforced, if a contact force affects a robot the movement pattern will be hard to predict.

The purpose of a reference movement is usually to search through an area to try to find a fit.

The spiral is specified in the reference movement coordinate system. This coordinate system's origin is the tool center point. Its orientation is normally the same as the orientation of the work object coordinate system but can be changed by using the instruction `FCRefMoveFrame`.

Basic example

```
FCRefSpiral FCPLANE_XY, 50, 100, 10
```

Sets up, but does not activate, a spiral shaped reference movement in the XY plane. The speed is 50 degrees per second and the largest radius 100 mm. After expanding the radius for 10 turns the radius will decrease for another 10 turns. After this the movement will be repeated in opposite direction.

Arguments

```
FCRefSpiral Plane Speed Radius Turns
```

`Plane`

Data type: `fcplane`

Specifies which plane the spiral is defined in (`FCPLANE_XY, FCPLANE_XZ or FCPLANE_YZ`).

`Speed`

Data type: `num`

The speed of the spiral movement. The unit is degrees per second.

`Radius`

Data type: `num`

The radius of the spiral. The unit is in mm.

`Turns`

Data type: `num`

The number of turns expanding the spiral.

Program execution

Execution behavior:

- The reference movement must be set up before activating the references with `FCRefStart`

Continues on next page

Syntax

```
FCRefSpiral
  [ Plane ':=' ] < expression (IN) of fcplane > ','
  [ Speed ':=' ] < expression (IN) of num > ','
  [ Radius ':=' ] < expression (IN) of num > ','
  [ Turns ':=' ] < expression (IN) of num > ';'
```

Related information

For information about	See
The data type `fcplane`	*fcplane on page 177*
Setting up linear reference movement	*FCRefLine on page 108*
Setting up circular reference movement	*FCRefCircle on page 104*
Setting up rotational reference movements	*FCRefMoveFrame on page 110*
Activating reference values	*FCRefStart on page 119*
Deactivating reference values	*FCRefStop on page 121*

7.1.22 FCRefSprForceCart

Usage

`FCRefSprForceCart` is used to set up a position dependent force reference. This force reference works like a virtual mechanical spring. The further away the robot TCP is from a defined attractor point, the larger the reference force trying to pull the robot towards the attractor.The attractor point is defined in the work object coordinate system. It is possible to define different stiffness in different directions.

Basic example

```
FCRefSprForceCort \stiffnessX:=100 \MAxForceX:=1000;
```

Sets up a reference spring force, with a stiffness of 100 N/m and a maximum force of 1000 N. The attractor position is not set and therefore implicitly set to the current TCP position. The spring is only active in the x direction of the work object.

See also *More examples on page 117*.

Arguments

```
FCRefSprForceCart [\StiffnessX] [\StiffnessY] [\StiffnessZ]
     [MaxForceX] [MaxForceY] [MaxForceZ][\PosAttractor]
```

`[\StiffnessX]`

Data type: `num`

This argument defines the spring stiffness in the x direction of the work object. This factor multiplied with the distance between TCP and attractor point gives the force reference in the x direction.

`[\StiffnessY]`

Data type: `num`

This argument defines the spring stiffness in the y direction of the work object. This factor multiplied with the distance between TCP and attractor point gives the force reference in the y direction

`[\StiffnessZ]`

Data type: `num`

This argument defines the spring stiffness in the z direction of the work object. This factor multiplied with the distance between TCP and attractor point gives the force reference in the z direction

`[\MaxForceX]`

Data type: `num`

This argument defines the maximum allowed force in x direction of the work object when the robot is in spring mode. Even if the distance between TCP and attractor point keeps increasing the force in x-direction never gets larger than `MaxForce`X.

`[\MaxForceY]`

Data type: `num`

Continues on next page

This argument defines the maximum allowed force in y direction of the work object when the robot is in spring mode. Even if the distance between TCP and attractor point keeps increasing the force in y-direction never gets larger than `MaxForceY`.

`[\MaxForceZ]`

Data type: `num`

This argument defines the maximum allowed force in z direction of the work object when the robot is in spring mode. Even if the distance between TCP and attractor point keeps increasing the force in z-direction never gets larger than `MaxForceZ`.

`[\PosAttractor]`

Attractor position

Data type: `pos`

The attractor position is the position the robot TCP tries to reach. This position is set in the work object coordinate system. If omitted, the TCP at the time of execution of this instruction is used.

Program execution

Execution behavior:

- The reference spring force must be set up before activating the references with `FCRefStart`.

More examples

Example 1

```
VAR pos myPos := [100,200,300];
FCRefSprForceCart \stiffnessY:=50 \MaxForceY:=200 \PosAttractor:=
     myPos;
```

Sets up a reference spring force, with a stiffness of 50 N/m and a maximum force of 200 N in the y-direction of the work object. The attractor position is specifically set.

Syntax

```
FCRefSprForceCart
  [ '\' StiffnessX ':=' < expression (IN) of num > ]
  [ '\' StiffnessY ':=' < expression (IN) of num > ]
  [ '\' StiffnessZ ':=' < expression (IN) of num > ]
  [ '\' MaxForceX ':=' < expression (IN) of num > ]
  [ '\' MaxForceY ':=' < expression (IN) of num > ]
  [ '\' MaxForceZ ':=' < expression (IN) of num > ]
  [ '\' PosAttractor ':=' < expression (IN) of pos > ] ';'
```

Related information

For information about	See
Setting up force reference	*FCCondForce on page 69*
Setting up torque reference	*FCCondTorque on page 85*
Activating reference values	*FCAct on page 63*

Continues on next page

For information about	See
Deactivating reference values	*FCDeact on page 90*

7.1.23 FCRefStart

Usage

FCRefStart is used to activate previously set up force, torque or movement references.

Basic example

```
FCRefForce \Fx:=l0;
FCRefTorque \Tx:=l0;
FCAct tool1;
FCRefStart;
```

After execution of this code, both the force and the torque references will be active.

See also *More examples on page 119*.

Arguments

```
FCRefStart
```

Program execution

Execution behavior:

- FCRefStart activates any references set up since the last FCRefStart instruction. All reference values activated by the first FCRefStart will be deactivated by a new FCRefStart.
- Several reference instructions can be activated by a FCRefStart instruction. However, not all instructions will be valid, see *Conflicting reference values on page 45*.

More examples

Example 1

```
FCRefForce \Fx:=l0;
FCRefSpiral FCPLANE_XY, 50, 100, 10;
FCAct tool1;
FCRefStart;
WaitTime 10;
FCRefForce \Fx:=l0;
FCRefStart;
```

At first, both reference force and reference movement are used. After 10 seconds the reference movement is stopped without ever releasing the reference force.

Example 2

```
FCRefCircle FCPLANE_XY, 60, 100;
FCRefLine LinX, 200, 50;
FCAct tool1;
FCRefStart;
```

In this example two instruction set up the reference movement in x direction. The value of the last instruction is used in this case. The movement in y direction will be according to the circle setup but the movement in the x direction will be according to the line setup.

Continues on next page

Syntax

```
FCRefStart
```

Related information

For information about	See
Deactivating reference values	*FCRefStop on page 121*

7.1.24 FCRefStop

Usage

`FCRefStop` is used to deactivate reference values. References can be either force, torque or movement references.

The same start and stop instruction is used for all references. The `FCRefStop` instruction stops all started references.

Basic example

```
FCRefForce \Fy:=l0;
FCRefTorque \Ty:=l0;
FCRefAct tool1;
FCRefStart;
WaitTime 10;
FCRefStop;
```

The reference force and torque are deactivated after 10 seconds.

Arguments

```
FCRefStop
```

Limitations

`FCRefStop` cannot stop only some of the active references, e.g. stop the reference movement but maintain the reference force. However, this can be done by a new setup instruction followed by a new start instruction. See *FCRefStart on page 119*.

Syntax

```
FCRefStop
```

Related information

For information about	See
Activating reference values	*FCRefStart on page 119*

7.1.25 FCRefTorque

Usage

`FCRefTorque` is used to specify a reference torque for Force Control. This instruction is only used to set up the reference torque, not to activate it. Activation is done with instruction `FCRefStart`.

Once activated the robot will start to move in order to achieve the reference torque. The reference torque is usually set up by using a constant torque, but it is possible to use an oscillating reference torque.

Basic example

```
FCRefTorque Ty:=10;
```

Setup a constant reference torque of 10 Nm around the positive y direction of the force control coordinate system.

See also *More examples on page 123*.

Arguments

```
FCRefTorque [\Tx] [\Ty] [\Tz] [\Amp] [\Period]
```

`[\Tx]`

Reference torque around x direction

Data type: `num`

Defines the constant torque reference around the x direction of the force control coordinate system. If this argument is omitted there will be zero reference torque in this direction.

`[\Ty]`

Reference torque around y direction

Data type: `num`

Defines the constant torque reference around the y direction of the force control coordinate system. If this argument is omitted there will be zero reference torque in this direction.

`[\Tz]`

Reference torque around z direction

Data type: `num`

Defines the constant torque reference around the z direction of the force control coordinate system. If this argument is omitted there will be zero reference torque in this direction.

`[\Amp]`

Amplitude

Data type: `fcxyznum`

The magnitude of the optional oscillating part of the torque reference, in the unit Nm. If `Amp` is used, `Period` should also be used.

Continues on next page

`[\Period]`

Data type: `fcxyznum`

The period time for the optional oscillating part of the reference torque, in the unit seconds. If `Period` is used, `Amp` should also be used.

Program execution

Execution behavior:

- The reference force is specified in the force control coordinate system.
- The reference torque must be set up before activating the references with `FCRefStart`.

More examples

Example 1

```
FCRefTorque \Ty:=20 \Tz:=10;
```

Setup a constant reference torque of 20 Nm around the positive y direction and 10Nm around the posistive z direction of the force control coordinate system.

Example 2

```
VAR fcxyznum myAmp:=[0,0,10];
VAR fcxyznum myPeriod := [0,0,1];
FCRefTorque \Tz:=10 \Amp:=myAmp \Period:=myPeriod;
```

Sets up an oscillating reference torque between 0 and 20 Nm with a period of 1 second.

Syntax

```
FCRefTorque
  [ '\' Tx ':=' < expression (IN) of num > ]
  [ '\' Ty ':=' < expression (IN) of num > ]
  [ '\' Tz ':=' < expression (IN) of num > ]
  [ '\' Amp ':=' < expression (IN) of fcxyznum > ]
  [ '\' Period ':=' < expression (IN) of fcxyznum > ] ';'
```

Related information

For information about	See
Setting up force reference	*FCRefForce on page 106*
Activating reference values	*FCRefStart on page 119*
Deactivating reference values	*FCRefStop on page 121*
The data type `fcxyznum`	*fcxyznum on page 183*

7.1.26 FCResetDampingTune

Usage

`FCResetDampingTune` is used to reset the damping in force directions, previously set up by `FCSetDampingTune`. `FCResetDampingTune` resets to the actual value set by the instruction `FCAct`, not to the value in the configuration file.

Basic example

Example

```
FCResetDampingTune;
```

Resets the damping in force direction.

Arguments

```
FCResetDampingTune
```

Program execution

Execution behavior:

- Reset damping value.

Syntax

```
FCResetDampingTune ';'
```

Related information

For information about	See
Set the damping in force direction	*FCSetDampingTune on page 127*
Configuration parameters for damping.	*Damping in Force x Direction - Damping in Force z Direction on page 203*

7.1.27 FCResetLPFilterTune

Usage

`FCResetLPFilterTune` is used to reset the low pass filter cut off frequency to the configured value. This will change the response of force loop according to description in *Damping and LP-filter on page 47*.

Basic examples

Example

```
FCResetLPFilterTune
```

Resets the low pass filter to configured value.

Arguments

```
FCResetLPFilterTune
```

Program execution

Execution behavior:

- Resets the force loop to the configured cut off frequency value.

Limitations

This instruction cannot be used by the GoFa robot.

Syntax

```
FCResetLPFilterTune;
```

Related information

For information about	See
Setting the parameter for the low pass filter.	*Bandwidth of force loop filter on page 206*
Instruction how to set low pass filter	*FCSetLPFilterTune on page 129*

7.1.28 FCResetMaxForceChangeTune

Usage

`FCResetMaxForceChangeTune` is used to reset the force ramp to the value that is specified in system parameter *Max Ref Force Change on page 210*.

Basic example

Example

```
FCResetMaxForceChangeTune;
```

Reset the force change ramp to the value defined in system parameter *Max Ref Force Change on page 210*.

Arguments

```
FCResetMaxForceChangeTune;
```

Program execution

Execution behavior:

- `FCResetMaxForceChangeTune` needs to be run if `FCSetMaxForceChangeTune` has been used to change the force ramp.

Syntax

```
FCResetMaxForceChangeTune
```

Related information

For information about	See
Defining maximum force ramping.	*Max Ref Force Change on page 210*
Defining a temporary force ramping value.	*FCSetMaxForceChangeTune on page 130*

7.1.29 FCSetDampingTune

Usage

`FCSetDampingTune` is used to tune the damping in the force control coordinate systems. The parameters tuned are those described in *Damping in Torque x Direction - Damping in Torque z Direction on page 204* and *Damping in Force x Direction - Damping in Force z Direction on page 203*.

Damping can be set in the configuration file or by the instruction `FCAct`. The difference is that this instruction can be used when force control is active. `FCSetDampingTune` tunes the actual values set by the instruction `FCAct`, not the value in the configuration file.

Basic example

```
VAR num xdamp:=100;
VAR num ydamp:=200;
VAR num zdamp:=200;
VAR num rxdamp:=100;
VAR num rydamp:=100;
VAR num rzdamp:=100
FCSetDampingTune xdamp, ydamp, zdamp, rxdamp, rydamp, rzdamp;
```

In this example the dampings are increased in the linear y and z directions, which makes the robot less compliant in these directions.

Arguments

```
FCSetDampingTune xdamp, ydamp, zdamp, rxdamp, rydamp, rzdamp
```

`xdamp`

Data type: `num`

A percentage value on how much the damping should change in the linear x direction.

`ydamp`

Data type: `num`

A percentage value on how much the damping should change in the linear y direction.

`zdamp`

Data type: `num`

A percentage value on how much the damping should change in the linear z direction.

`rxdamp`

Data type: `num`

A percentage value on how much the damping should change in the rotational x direction.

`rydamp`

Data type: `num`

Continues on next page

A percentage value on how much the damping should change in the rotational y direction.

`[\RampTime]`

Data type: `num`

How fast the damping should change. Default value is 0.15.

`rzdamp`

Data type: `num`

A percentage value on how much the damping should change in the rotational z direction.

Program execution

The instruction can be used to change damping while force control is active.

Syntax

```
FCSetDampTune
  [ xdamp ':=' ] < expression (IN) of num >' ,'
  [ ydamp ':=' ] < expression (IN) of num > ','
  [ zdamp ':=' ] < expression (IN) of num > ','
  [ rxdamp ':=' ] < expression (IN) of num > ','
  [ rydamp ':=' ] < expression (IN) of num > ','
  [ rzdamp ':=' ] < expression (IN) of num >
  [ '\' RampTime' :=' < expression (IN) of num > ] ';'
```

Related information

For information about	See
Reset damping in force direction.	*FCResetDampingTune on page 124*
Damping kinematics.	*Damping in Force x Direction - Damping in Force z Direction on page 203*

7.1.30 FCSetLPFilterTune

Usage

`FCSetLPFilterTune` is used change the response of force loop according to description in *Damping and LP-filter on page 47*.

Basic examples

```
FCSetLPFilterTune 2;
```

Set the force loop cut off frequency to 2 Hz. A low value will make the force control less compliant but more stable.

Arguments

```
FCSetLPFilterTune CutOffFreq;
```

`CutOffFreq`

Cut off frequency

Data type: `num`

Cut off frequency

Program execution

Execution behavior:

- Set cut off frequency.

Limitations

Instruction cannot be executed when force control is active.

This instruction cannot be used by the GoFa robot.

Syntax

```
FCSetLPFilterTune
  [CutOffFreq ':=' ] < expression (IN) of num> ';'
```

Related information

For information about	See
Setting the parameter for low pass filter.	*Bandwidth of force frame filter on page 205*
Instruction how to reset the low pass filter.	*FCResetLPFilterTune on page 125*

CAUTION

The cut off frequency effects the force loop stability.

7.1.31 FCSetMaxForceChangeTune

Usage

`FCSetMaxForceChangeTune` defines a temporary ramp value to be used instead of that specified in system parameter *Max Ref Force Change on page 210*.

This is useful if the ramping needs to be at low value when ramping up, but can be faster when force references are stopped.

It is also useful if the process consists of several steps, where the initial contact needs to be slower than changes later in the process.

Basic example

Example

```
FCSetMaxForceChangeTune 1000;
```

This temporarily changes the force ramp to 1000 N/s.

Example 2

```
FCRefForce \Fx:=l00;
FCAct tool1;
FCRefStart; ! This will now ramp with value from configuration
WaitTime 10;
FCSetMaxForceChangeTune 1000; ! This will now ramp with 1000 N/s
FCRefForce \Fx:=l0;
FCRefStart;
```

Arguments

```
FCSetMaxForceChangeTune ForceChange
```

`ForceChange`

Data type: `num`

Specifies the temporary force ramp in N/s.

A value between 0 and 10,000 N/s.

Program execution

Execution behavior:

- `FCSetMaxForceChangeTune` changes the force ramp (force change) for the next `FCRefStart` or `FCRefStop` instruction.

Syntax

```
FCSetMaxForceChangeTune
  [ForceChange':='] <expression (IN) of num>';'
```

Related information

For information about	See
Defining maximum force ramping.	*Max Ref Force Change on page 210*
Resetting the force ramp to the configured value.	*FCResetMaxForceChangeTune on page 126*

7.1.32 FCSpdChgAct

Usage

The `FCSpdChgAct` is used to activate FC SpeedChange function with desired reference and recover behavior. When FC SpeedChange function is active, the robot speed will be reduced/increased in order to keep the measured signal close to the reference.

Basic examples

Basic example of the instruction `FCSpdChgAct` is illustrated below.

See also *FCSpdChgAct on page 131*.

Example

```
FCSpdChgAct 200 /RecoverFunName:="local_grind";
```

Activate FC SpeedChange with user-specified recover routine `local_grind`. The measured process signal will be controlled to be 200 by slowing down TCP speed when required.

Arguments

```
FCSpdChgAct Reference [\RecoverFunName] [\NonStopAllTime]
     [\MultipleRecover]
```

`Reference`

Data type: `num`

The reference value for the process force. (Process force defined by input, such as Force sensor, current, torque etc.) The measurement will be controlled below this reference value. The value of the reference must be identified in tests during normal conditions.

`[\RecoverFunName]`

Data type: `string`

This parameter specifies the name of user-defined recovery routine. The recovery routine will be executed, if the process force is still is too large after the TCP speed already is reduced to the minimum speed. The recovery routine needs to be implemented by the user in order to have desired recover behavior. If no recover routine is specified, the robot will stop immediately when the above recover condition met.

`[\NonStopAllTime]`

Data type: `switch`

This option can only be used when `RecoverFunName` argument is NOT used. The robot will at most slow down to minimum feed rate (speed), which means that the robot will not stop for any overload occurring at minimum speed. USE THIS OPTION WITH CAUTION.

`[\MultipleRecover]`

Data type: `switch`

Continues on next page

This option can only be used when `RecoverFunName` argument is used with this option, the user-specified recover procedure will be called multiple times along the path whenever overload happens at minimum feed speed. If this option is not specified, the user-specified recover procedure will be called the first time when recover condition met. If the recover condition is met again along the path, the robot will stop immediately.

Program execution

Execution behavior:

- Configure *FC Speed Change* system parameters before calling `FCSpdChgAct`.
- If 6DOF force sensor is used for feedback, `FCCalib` must be called before `FCSpdChgAct`.
- The RobotWare option *Path Recovery* must be installed in order to use `FcSpdChgAct` instruction with recover function. The only exception is to use the `FcSpdChgAct` instruction with `NonStopAllTime`.
- User-specified recovery routine will not be called recursively. Which means, if the recovery condition met when controller is executing user-specified recovery routine, the robot will stop immediately instead of calling user-specified recovery routine from itself.
- If the RAPID program pointer is moved manually, FC SpeedChange function will be deactivated automatically.
- If the RAPID program stops, jogs away from current position, then restarts without regain the path, FC SpeedChange function will be deactivated automatically.

Limitations

- Do NOT change tool and work object frame in RAPID program between `FCSpdChgAct` and `FCSpdChgDeact`.

More examples

More examples of how to use the instruction `FCSpdChgAct` are illustrated below.

Example 1

```
FCSpdChgAct 200;
```

Activate FC SpeedChange function with reference 200. No user-specified recover behavior is defined. The robot will stop immediately when recover condition met.

Example 2

```
FCSpdChgAct 200 \RecoverFunName:="local_grind";
```

Activate FC SpeedChange function with reference 200 and user-specified recover routine `local_grind`. Local_grind will be executed when recover condition met, but will be called only once.

Syntax

```
FCSpdChgAct
  [ Reference ':=' ] < expression (IN) of num > ','
  [ RecoverFunName ':=' ] < expression (IN) of string > ','
```

Continues on next page

```
[ '\' NonStopAllTime ] ','
[ '\' MultipelRecover ] ';'
```

Related information

For information about	See
Deactivate SpeedChange	*FCSpdChgDeact on page 134*

7.1.33 FCSpdChgDeact

Usage

Deactivate FC SpeedChange function.

Basic examples

Basic example of the instruction `FCSpdChgDeact` is illustrated below.

Example

```
FCSpdChgDeact;
```

Deactivates SpeedChange function.

Arguments

```
FCSpdChgDeact
```

There are no arguments to the instruction.

Syntax

```
FCSpdChgDeact
```

Related information

For information about	See
Activate SpeedChange.	*FCSpdChgAct on page 131*

7.1.34 FCSpdChgTunSet

Usage

FCSpdChgTunSet is used to set FC SpeedChange system parameters to a new value.

Basic examples

Basic example of the instruction FCSpdChgTunSet is illustrated below.

See also *FCSpdChgTunSet on page 135*.

Example

```
FCSpdChgTunSet 0.2, FC_SPEED_RATIO_MIN;
```

Set FC SpeedChange system parameter *Speed ratio min* to 0.2.

Arguments

```
FCSpdChgTunSet value type;
```

value

Data type: num

Value to be set for the FC SpeedChange system parameter.

type

Data type: fcspdchgtunetype

The FC SpeedChange system parameter whose value is to be set (FC_SPEED_RATIO_MIN, FC_NO_OF_SPEED_LEVELS). Only two FC SpeedChange system parameters can be tuned by this instruction, as shown in the following table:

Parameter	Type
Speed ratio min	FC_SPEED_RATIO_MIN
No of speed levels	FC_NO_OF_SPEED_LEVELS

Program execution

Execution behavior:

- Set new values to tunable FC SpeedChange system parameters.

More examples

More examples of how to use the instruction FCSpdChgTunSet are illustrated below.

Example 1

```
FCSpdChgTunSet 3, FC_NO_OF_SPEED_LEVELS;
```

Set FC SpeedChange system parameter *No of speed levels* to 3.

Continues on next page

Limitations

`FCSpdChgTunSet` will not set system parameter to the new value if called inside `FCSpdChgAct - FCSpdChgDeact` instruction block. It must be called before activating FC Speed Change. The valid value for the system parameters are shown in the following table:

Parameter	Type
Speed ratio min	0.02 - 1.0
No of speed levels	2 - 10

Syntax

```
FCSpdChgTunSet
  [ value ':=' ] < expression (IN) of num> ','
  [ type ':=' ] < expression (IN) of fcspdchgtunetype> ';'
```

Related information

For information about	See
Set tune parameters to original.	*FCSpdChgTunReset on page 137*

7.1.35 FCSpdChgTunReset

Usage

`FCSpdChgTunReset` reset tuneable FC SpeedChange system parameters to original value stored in configuration.

Basic examples

Basic examples of the instruction `FCSpdChgTunReset` are illustrated below.

See also *FCSpdChgTunReset on page 137*.

Example

```
FCSpdChgTunReset FC_SPEED_RATIO_MIN
```

Reset FC SpeedChange system parameter *Speed ratio min* to its original value.

Arguments

```
FCSpdChgTunReset type;
```

`type`

Data type: `fcspdchgtunetype`

The FC SpeedChange system parameter whose value is to be reset (FC_SPEED_RATIO_MIN, FC_NO_OF_SPEED_LEVELS). Only two FC SpeedChange system parameters can be reset by this instruction, as shown in the following table:

Parameter	Type
Speed ratio min	FC_SPEED_RATIO_MIN
No of speed levels	FC_NO_OF_SPEED_LEVELS

Program execution

Execution behavior:

- Reset tunable FC SpeedChange system parameters.

More examples

More examples of how to use the instruction `FCSpdChgTunReset` are illustrated below.

Example

```
FCSpdChgTunReset FC_NO_OF_SPEED_LEVELS;
```

Reset FC SpeedChange system parameter *No of speed levels*.

Limitations

`FCSpdChgTunReset` will not reset system parameter if called inside `FCSpdChgAct` - `FCSpdChgDeact` instruction block. It must be called outside the `FCSpdChgAct` - `FCSpdChgDeact` instruction block.

Syntax

```
FCSpdChgTunReset
  [ type ':=' ] < expression (IN) of fcspdchgtunetype >';'
```

Continues on next page

Related information

For information about	See
Set tune parameters.	*FCSpdChgTunSet on page 135*

7.1.36 FCSupvForce

Usage

`FCSupvForce` is used to set up force supervision in Force Control. The supervision is activated when Force Control is activated with the instruction `FCAct`.

The force supervision is set up by defining minimum and maximum limits for the force in the directions of the force control coordinate system. Once activated, the supervision will stop the execution if the force is outside the allowed values.

The force supervision is specified in the force control coordinate system. This coordinate system is setup by the user with the instruction `FCAct`.

Basic example

```
FCSupvForce \Xmin:=-200 \Xmax:=200;
```

Defines force supervision that checks that the force in the x direction of the force control coordinate system is between -200 N and 200 N. This means that the force magnitude must be smaller than 200 N in both positive and negative x direction. No restrictions are used for the force in other directions.

See also *More examples on page 140*.

Arguments

```
FCSupvForce [\XMin] [\XMax] [\YMin] [\YMax] [\ZMin] [\ZMax]
```

`[\XMin]`

Minimum force in x direction

Data type: `num`

Lower limit for force in the x direction of the force control coordinate system. A negative value limits the maximum force in the negative x direction.

The unit is Newton and the default value is negative infinity.

`[\XMax]`

Maximum force in x direction

Data type: `num`

Upper limit for force in the x direction of the force control coordinate system. A negative value limits the minimum force in negative x direction.

The unit is Newton and the default value is positive infinity.

`[\YMin]`

Minimum force in y direction

Data type: `num`

Lower limit for force in the y direction of the force control coordinate system. A negative value limits the maximum force in the negative y direction.

The unit is Newton and the default value is negative infinity.

`[\YMax]`

Maximum force in y direction

Data type: `num`

Continues on next page

Upper limit for force in the y direction of the force control coordinate system. A negative value limits the minimum force in negative y direction.

The unit is Newton and the default value is positive infinity.

`[\ZMin]`

Minimum force in z direction

Data type: `num`

Lower limit for force in the z direction of the force control coordinate system. A negative value limits the maximum force in the negative z direction.

The unit is Newton and the default value is negative infinity.

`[\ZMax]`

Maximum force in z direction

Data type: num

Upper limit for force in the z direction of the force control coordinate system. A negative value limits the minimum force in negative z direction.

The unit is Newton and the default value is positive infinity.

Program execution

Execution behavior:

- If the supervision conditions are exceeded, execution stops with an emergency error.

More examples

Example 1

```
FCSupvForce \Xmin:=-100 \Xmax:=100 \Ymin:=-100 \Ymax:=100
     \Zmin:=-100 \ZMax:=100;
```

This supervision does not allow the force to be larger than 100 N in any of the directions (positive or negative x, y and z). Note that the force may be 100 N in both x, y and z direction, resulting in a total force magnitude of 173 N.

Syntax

```
FCSupvForce
  [ '\' XMin ':=' < expression (IN) of num > ]
  [ '\' XMax ':=' < expression (IN) of num > ]
  [ '\' YMin ':=' < expression (IN) of num > ]
  [ '\' YMax ':=' < expression (IN) of num > ]
  [ '\' ZMin ':=' < expression (IN) of num > ]
  [ '\' ZMax ':=' < expression (IN) of num > ] ';'
```

Related information

For information about	See
Setting up torque supervision.	*FCSupvTorque on page 151*
Setting up position supervision.	*FCSupvPos on page 145*
Setting up tool orientation supervision.	*FCSupvOrient on page 142*

Continues on next page

For information about	See
Setting up TCP speed supervision.	*FCSupvTCPSpeed on page 149*
Setting up reorientation speed supervision.	*FCSupvReoriSpeed on page 147*
Activate supervision.	*FCAct on page 63*

7.1.37 FCSupvOrient

Usage

`FCSupvOrient` is used to set up an supervision for the tool orientation. The supervision is activated when Force Control is activated with the instruction `FCAct`.

An orientation supervision is set up by defining a maximum angle and a maximum rotation from a reference orientation. The reference orientation is either defined by the current z direction of the tool, or by specifying an orientation in relation to the z direction of the work object.

Once activated, the tool orientation must be within the limits otherwise the supervision will stop the execution.

Basic example

```
FCSupvOrient \MaxAngle:= 30;
```

In this example, no supervision coordinate system is specified. This means that the supervision coordinate system is the same as the tool coordinate system at the time of execution of this instruction. When this supervision is activated it will stop the execution if the tool’s z axis deviates more than 30 degrees from the z axis of the supervision coordinate system.

See also *More examples on page 143*.

Arguments

```
FCSupvOrient [\OrientSupvFrame] [\MaxAngle] [\MaxRot] [\Outside]
```

`[\OrientSupvFrame]`

Orient supervision coordinate system

Data type: `orient`

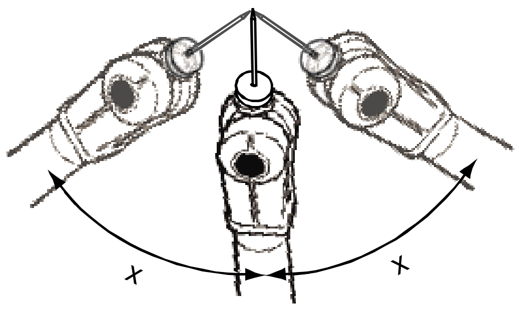
`OrientSupvFrame` is used to set the supervision coordinate system in which the tool orientation supervision is defined. The coordinate system is set by an orient in relation to the work object coordinate system. If `OrientSupvFrame` is omitted, the tool coordinate system at the time of execution is used as supervision coordinate system.

Continues on next page

`[\MaxAngle]`

Data type: `num`

The maximum angle between the z direction of the tool and the z direction of the supervision coordinate system. The unit is degrees.



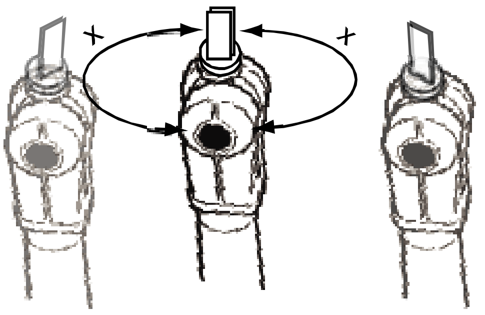
xx0500001913

x	`MaxAngle`

`[\MaxRot]`

Data type: `num`

The maximum rotation angle around the z axis of the tool, compared to the z direction of the supervision coordinate system. The unit is degrees.



xx0500001912

x	`MaxRot`

Program execution

Execution behavior:

- If the supervision conditions are exceeded, execution stops with an emergency error.

More examples

Example 1

```
FCSupvOrient \MaxRot:= 90;
```

In this example, the supervision coordinate system is set to the same as the tool coordinate system at the time of execution. If the rotation is larger than 90 degrees around the z axis the supervision will stop the execution.

Example 2

```
VAR orient my_orient:=[0,0,1,0];
```

Continues on next page

```
FCSupvOrient \OrientSupvFrame:=my_orient \MaxAngle:=30 \MaxRot:=45;
```

In this example the z direction of the supervision coordinate system is in negative z direction of the work object coordinate system.

The supervision will stop the execution if:

- The tool's z direction deviates more than 30 degrees from the z direction of the supervision coordinate system.
- The tool's rotation around the z axis deviates more than 45 degrees from the supervision coordinate system.

Syntax

```
FCSupvOrient
  [ '\' OrientSupvFrame ':=' < expression (IN) of pose > ]
  [ '\' MaxAngle ':=' < expression (IN) of num > ]
  [ '\' MaxRot ':=' < expression (IN) of num > ] ';'
```

Related information

Form information about	See
Setting up position supervision	*FCSupvPos on page 145*
Setting up force supervision	*FCSupvForce on page 139*
Setting up torque supervision	*FCSupvTorque on page 151*
Setting up TCP speed supervision	*FCSupvTCPSpeed on page 149*
Setting up reorientation speed supervision	*FCSupvReoriSpeed on page 147*
Activating supervision	*FCAct on page 63*

7.1.38 FCSupvPos

Usage

`FCSupvPos` is used to set up position supervision in Force Control. Supervision is activated when Force Control is activated with the instruction `FCAct`.

Position supervision is set up by defining a volume in space for the TCP. Once activated, the supervision will stop the execution if the TCP is outside this volume.

Basic example

```
VAR fcboxvol my_box:= [-500, 500, -500, 500, -500, 500];
FCSupvPos \Box:= my_box;
```

Sets up a position supervision where the TCP must stay between -500 mm and 500 mm in all directions of the work object coordinate system.

See also *More examples on page 146*.

Arguments

```
FCSupvPos [\PosSupvFrame] [\Box] | [\Cylinder] | [\Sphere]
```

`[\PosSupvFrame]`

Position supervision coordinate system

Data type: `pose`

This parameter sets the coordinate system in which the TCP position supervision is defined. The coordinate system is set by a pose in relation to the work object coordinate system. The default value is the unity pose, meaning that if the parameter is omitted the position TCP supervision is defined in the work object coordinate system.

`[\Box]`

Data type: `fcboxvol`

Defines a box-shaped volume that the TCP must be inside.

One, and only one, of the arguments `Box`, `Cylinder` and `Sphere` must be used.

`[\Cylinder]`

Data type: `fccylindervol`

Defines a cylinder-shaped volume that the TCP must be inside.

One, and only one, of the arguments `Box`, `Cylinder` and `Sphere` must be used.

`[\Sphere]`

Data type: `fcspherevol`

Defines a sphere-shaped volume that the TCP must be inside.

One, and only one, of the arguments `Box`, `Cylinder` and `Sphere` must be used.

Program execution

Execution behavior:

- If the supervision conditions are exceeded, execution stops with an emergency error.

Continues on next page

More examples

Example 1

```
VAR fcboxvol my_box:= [-9e9, 9e9, -9e9, 9e9, 300, 800];
FCSupvPos \Box:= my_box;
```

Sets up a position supervision where the TCP must stay between 300 mm and 800 mm in the z direction of the work object coordinate system. No limits are set for the x and the y directions.

Example 2

```
VAR fccylindervol my_cyl:= [300, 0, -200, 500, 250];
VAR pose my_cs := [[0,0,600],[0.7071,0,0.7071,0]];
FCSupvPos \PosSupvframe:=my_cs \Cylinder:=my_cyl;
```

In this example the cylinder is not directly specified in the work object coordinate system but in a new coordinate system defined in relation to the work object coordinate system.

Syntax

```
FCSupvPos
  [ '\' PosSupvFrame ':=' < expression (IN) of pose > ]
  [ '\' Box ':=' < expression (IN) of fcboxvol > ]
  | [ '\' Cylinder ':=' < expression (IN) of fccylindervol > ]
  | [ '\' Sphere ':=' < expression (IN) of fcspherevol > ] ';'
```

Related information

For information about	See
The data type `fcboxvol`	*fcboxvol on page 163*
The data type `fcylindervol`	*fccylindervol on page 167*
The data type `fcspherevol`	*fcspherevol on page 181*
Setting up force supervision	*FCSupvForce on page 139*
Setting up torque supervision	*FCSupvTorque on page 151*
Setting up tool orientation supervision	*FCSupvOrient on page 142*
Setting up TCP speed supervision	*FCSupvTCPSpeed on page 149*
Setting up reorientation speed supervision	*FCSupvReoriSpeed on page 147*
Activate supervision	*FCAct on page 63*

7.1.39 FCSupvReoriSpeed

Usage

`FCSupvReoriSpeed` is used to set up reorientation speed supervision in Force Control. The supervision is activated when Force Control is activated with the instruction `FCAct`.

The reorientation speed supervision is set up by defining minimum and maximum limits for the reorientation speed around the axis of the work object coordinate system. Once activated, the supervision will stop the execution if the values of the reorientation speed are to high.

There are two speed supervisions: `FCSupvReoriSpeed` and `FCSupvTCPSpeed`, which is described in section *FCSupvTCPSpeed on page 149*.

Both supervisions may be required because:

- A robot axis can rotate with high speed while the TCP is stationary.
- The TCP can be far from the rotating axis and a small axis rotation may result in a high speed movement of the TCP.

Basic example

```
FCSupvReoriSpeed \XMin:=-100 \XMax:=100;
```

Defines torque supervision that checks that the torque speed in the x direction the work object coordinate system is between -100 deg/s and 100 deg/s. No restriction is put on the reorientation speed in other directions.

Arguments

```
FCSupvReoriSpeed [\XMin] [\XMax] [\YMin] [\YMax] [\ZMin] [\ZMax]
```

`[\XMin]`

Minimum speed around the x direction

Data type: `num`

Lower limit for reorientation speed around the x direction of the work object coordinate system. A negative value limits the maximum speed in the negative x direction.

The unit is degrees/s and the default value is -50.

`[\XMax]`

Maximum speed around the x-direction

Data type: `num`

Upper limit for reorientation speed around the x direction of the work object coordinate system. A negative value limits the minimum speed around negative x direction.

The unit is degrees/s and the default value is 50.

`[\YMin]`

Minimum speed around the y direction

Data type: `num`

Continues on next page

Lower limit for reorientation speed around the y direction of the work object coordinate system. A negative value limits the maximum speed around the negative y direction.

The unit is degrees/s and the default value is -50.

`[\YMax]`

Maximum speed around the y direction

Data type: `num`

Upper limit for TCP speed around the y direction of the work object coordinate system. A negative value limits the minimum speed around negative y direction.

The unit is degrees/s and the default value is 50.

`[\ZMin]`

Minimum speed around the z direction

Data type: `num`

Lower limit for TCP speed around the z direction of the work object coordinate system. A negative value limits the maximum speed around the negative z direction.

The unit is degrees/s and the default value is -50.

`[\ZMax]`

Maximum speed around the z direction

Data type: `num`

Upper limit for TCP speed around the z direction of the work object coordinate system. A negative value limits the minimum speed around negative z direction.

The unit is degrees/s and the default value is 50.

Syntax

```
FCSupvReoriSpeed
  [ '\' XMin ':=' < expression (IN) of num > ]
  [ '\' XMax ':=' < expression (IN) of num > ]
  [ '\' YMin ':=' < expression (IN) of num > ]
  [ '\' YMax ':=' < expression (IN) of num > ]
  [ '\' ZMin ':=' < expression (IN) of num > ]
  [ '\' ZMax ':=' < expression (IN) of num > ] ';'
```

Related information

For information about	See
Setting up force supervision	*FCSupvForce on page 139*
Setting up torque supervision	*FCSupvTorque on page 151*
Setting up position supervision	*FCSupvPos on page 145*
Setting up orientation supervision	*FCSupvOrient on page 142*
Setting up TCP speed supervision	*FCSupvTCPSpeed on page 149*
Activating the supervision	*FCAct on page 63*

7.1.40 FCSupvTCPSpeed

Usage

`FCSupvTCPSpeed` is used to set up TCP speed supervision in Force Control. The supervision is activated when Force Control is activated with the instruction `FCAct`.

The TCP speed supervision is set up by defining minimum and maximum limits for the TCP speed in the directions of the work object coordinate system. Once activated, the supervision will stop the execution if too high TCP speed values are detected.

There are two speed supervisions: `FCSupvTCPSpeed` and `FCSupvReorispeed`, which is described in section *FCSupvReoriSpeed on page 147*.

Both supervisions may be required because:

- A robot axis can rotate with high speed while the TCP is stationary.
- The TCP can be far from the rotating axis and a small axis rotation may result in a high speed movement of the TCP.

Basic example

```
FCSupvTCPSpeed \XMin:=-100 \XMax:=100;
```

Defines TCP speed supervision that checks that the TCP speed in the x direction the work object coordinate system is between -100mm/s and 100mm/s. No restriction is put on the TCP speed in other directions.

Arguments

```
FCSupvTCPSpeed [\XMin] [\XMax] [\YMin] [\YMax] [\ZMin] [\ZMax]
```

`[\XMin]`

Minimum speed in the x direction

Data type: `num`

Lower limit for TCP speed in the x direction of the work object coordinate system. A negative value limits the maximum speed in the negative x direction.

The unit is mm/s and the default value is -250.

`[\XMax]`

Maximum speed in the x direction

Data type: `num`

Upper limit for TCP speed in the x direction of the work object coordinate system. A negative value limits the minimum speed in negative x direction.

The unit is mm/s and the default value is 250.

`[\YMin]`

Minimum speed in the y direction

Data type: `num`

Lower limit for TCP speed in the y direction of the work object coordinate system. A negative value limits the maximum speed in the negative y direction.

The unit is mm/s and the default value is -250.

Continues on next page

`[\YMax]`

Maximum speed in the y direction

Data type: `num`

Upper limit for TCP speed in the y direction of the work object coordinate system. A negative value limits the minimum speed in negative y direction.

The unit is mm/s and the default value is 250.

`[\ZMin]`

Minimum speed in the z direction

Data type: `num`

Lower limit for TCP speed in the z direction of the work object coordinate system. A negative value limits the maximum speed in the negative z direction.

The unit is mm/s and the default value is -250.

`[\ZMax]`

Maximum speed in the x direction

Data type: `num`

Upper limit for TCP speed in the z direction of the work object coordinate system. A negative value limits the minimum speed in negative z direction.

The unit is mm/s and the default value is 250.

Program execution

Execution behavior:

- If the supervision conditions are exceeded, execution stops with an emergency error.

Syntax

```
FCSupvTCPSpeed
  [ '\' XMin ':=' < expression (IN) of num > ]
  [ '\' XMax ':=' < expression (IN) of num > ]
  [ '\' YMin ':=' < expression (IN) of num > ]
  ['\' YMax ':=' < expression (IN) of num > ]
  ['\' ZMin ':=' < expression (IN) of num > ]
  ['\' ZMax ':=' < expression (IN) of num > ] ';'
```

Related information

For information about	See
Setting up force supervision	*FCSupvForce on page 139*
Setting up torque supervision	*FCSupvTorque on page 151*
Setting up position supervision	*FCSupvPos on page 145*
Setting up orientation supervision	*FCSupvOrient on page 142*
Setting up reorientation speed supervision	*FCSupvReoriSpeed on page 147*
Activating the supervision	*FCAct on page 63*

7.1.41 FCSupvTorque

Usage

`FCSupvTorque` is used to set up torque supervision in Force Control. The supervision is activated when Force Control is activated with the instruction `FCAct`.

The torque supervision is set up by defining minimum and maximum limits for the torque in the directions of the force control coordinate system. Once activated, the supervision will stop the execution if the torque is outside the allowed values.

Basic example

```
FCSupvTorque \Xmin:=-100 \Xmax:=100;.
```

Defines torque supervision that checks that the torque around the x axis is between -100 Nm and 100 Nm. This means that the torque magnitude must be smaller than 100 Nm in both clockwise and counterclockwise around the x axis. No restrictions are used for the torque in other directions

See also *More examples on page 152*.

Arguments

```
FCSupvTorque [\XMin] [\XMax] [\YMin] [\YMax] [\ZMin] [\ZMax]
```

`[\XMin]`

Minimum torque around the x axis

Data type: `num`

Lower limit for torque around the x axis of the force control coordinate system. A negative value limits the maximum torque in the opposite direction around the x axis.

The unit is Nm and the default value is negative infinity.

`[\XMax]`

Maximum torque around the x axis

Data type: `num`

Upper limit for torque around the x axis of the force control coordinate system. A negative value limits the minimum torque around negative x axis.

The unit is Nm and the default value is positive infinity.

`[\YMin]`

Minimum torque around the y axis

Data type: `num`

Lower limit for torque around the y axis of the force control coordinate system. A negative value limits the maximum torque in the opposite direction around they axis.

The unit is Nm and the default value is negative infinity.

`[\YMax]`

Maximum torque around the y axis

Data type: `num`

Continues on next page

Upper limit for torque around the y axis of the force control coordinate system. A negative value limits the minimum torque around negative y axis.

The unit is Nm and the default value is positive infinity.

`[\ZMin]`

Minimum torque around the z axis

Data type: `num`

Lower limit for torque around the z axis of the force control coordinate system. A negative value limits the maximum torque in the opposite direction around the z axis.

The unit is Nm and the default value is negative infinity.

`[\ZMax]`

Maximum torque around the z axis

Data type: `num`

Upper limit for torque around the z axis of the force control coordinate system. A negative value limits the minimum torque around negative z axis

The unit is Nm and the default value is positive infinity.

Program execution

Execution behavior:

- If the supervision conditions are exceeded, execution stops with an emergency error.

More examples

Example 1

```
FCSupvTorque \Xmin:=-100 \Xmax:=100 \Ymin:=-100 \Ymax:=100;.
```

Defines torque supervision that checks that the torque around the x and z axes are between -100 Nm and 100 Nm. This means that the torque magnitude must be smaller than 100 Nm in both clockwise and counter-clockwise around the x and y axes. No restrictions are used for the torque around the z axis.

Syntax

```
FCSupvTorque
  [ '\' XMin ':=' < expression (IN) of num > ]
  [ '\' XMax ':=' < expression (IN) of num > ]
  [ '\' YMin ':=' < expression (IN) of num > ]
  [ '\' YMax ':=' < expression (IN) of num > ]
  [ '\' ZMin ':=' < expression (IN) of num > ]
  [ '\' ZMax ':=' < expression (IN) of num > ] ';'
```

Related information

For information about	See
Setting up force supervision	*FCSupvForce on page 139*
Setting up position supervision	*FCSupvPos on page 145*
Setting up TCP speed supervision	*FCSupvTCPSpeed on page 149*

Continues on next page

For information about	See
Setting up reorientation speed supervision	*FCSupvReoriSpeed on page 147*
Activating the supervision	*FCAct on page 63*

7.2 Functions

7.2.1 FCGetForce

Usage

The function `FCGetForce` is used to retrieve the force sensor readings. The measured force and torque is returned in a force vector. It is possible to transform the measured force and torque from the force sensor coordinate system to either the tool coordinate system or the work object coordinate system. If the system has been calibrated, i.e. the instruction `FCCalib` has been executed, it is possible to return the force and torque without any offset. In a calibrated system it is also possible to remove the force and torque due to gravity from the sensor readings and only show contact force

Basic example

```
VAR fcforcevector myForceVector;
myForceVector:= FCGetForce();
```

In this example `FCGetForce` gets the values from the sensor and saves it in the variable `myForceVector`. If the system has not been calibrated, using the instruction `FCCalib`, raw measurement data will be returned. That means the sensor offset will be included in the result. If the system has been calibrated, only the force and torque corresponding to the gravity and contact forces will be shown.

See also *More examples on page 155*.

Return value

Data type: `fcforcevector`

The function returns a value of the data type `fcforcevector`, whose components are force and torque in three dimensions (x, y, z).

Arguments

```
FCGetForce ([\Tool] [\WObj] | [\ContactForce])
```

`[\Tool]`

Data type: `tooldata`

If a tool is specified the returned force will be transformed to the coordinate system of this tool.

`[\WObj]`

Data type: `wobjdata`

If a work object is specified the returned force will be transformed to the coordinate system

`[\ContactForce]`

Data type: `switch`

This option will remove the present gravity force from the result, displaying only contact forces. Note that this option is only allowed if the system has been calibrated before using the function `FCGetForce`.

Continues on next page

Program execution

Execution behavior:

- If the sensor has not been calibrated the returned force is the same as if the sensor had not yet been mounted on the robot, and depends on the sensor manufacturer. Some sensors are offset compensated at startup, which means that the result will be the same as if the sensor had been calibrated, but some are not.
- The resulting torque is in the origin of the new coordinate system. When a transformation is done it is assumed that the contact is in the TCP.

More examples

Example 1

```
VAR fcforcevector myForceVector;
myForceVector:=FCGetForce(\ContactForce);
```

In this example the force and torque due to gravity is removed, meaning that what we see is only contact forces.

Example 2

```
VAR fcforcevector myForceVector;
myForceVector:=FCGetForce(\WObj:=wobj2);
```

In this example the force readings are transformed to the work object coordinate system before returned.

Example 3

```
VAR fcforcevector myForceVector;
myForceVector:=FCGetForce(\Tool:=tool2);
```

In this example the force readings are transformed to the tool coordinate system before returned. It is necessary that the sensor is calibrated or else the function will return an error.

Syntax

```
FCGetForce '('
  ['\' Tool ':=' ] < persistent (PERS) of tooldata >
  | [ '\' WObj ':=' ] < persistent (PERS) of wobjdata >
  [ '\' ContactForce ] ')'
```

A function with a return value of the data type `fcforcevector`.

Related information

For information about	See
The data type `fcforcevector`	*fcforcevector on page 171*
Identifying the load	*FCLoadID on page 159*

7.2.2 FCGetProcessData

Usage

The function FCGetProcessData is used to retrieve six different variables gathered in a data type called fcprocessdata. If no arguments are used the fcprocessdata returned will be from the moment when the function was executed.

Basic example

```
VAR fcprocessdata mydata;
mydata := FCGetProcessData()
```

In this example FCGetProcessData retrieves the values from the system and saves it in a variable called mydata.

See *More examples on page 156*.

Return value

Data type: fcprocessdata

The function returns a variable of type fcprocessdata and its components are

- Condition Status
- Time
- Measured position in Reference movement coordinate system
- Measured speed in the work object frame.
- Measured Force in the force sensor coordinate system
- Measured force in Force Control coordinate system

Arguments

```
FCGetProcessData (\DataAtTrigTime)
```

[\DataAtTrigTime]

Data type: switch

If this argument is used the function will return fcprocessdata from the moment when the user defined condition was fulfilled.

More examples

Example 1

```
VAR fcprocessdata mydata;
mydata := FCGetProcessData(\DataAtTrigTime);
```

In this example FCGetProcessData get the values from when the user defined condition was true and saves it in a variable mydata.

Example 2

```
!Sets up a force condition.
FCCondForce \Xmin:=-40,TimeOut:=20;
!Defines a horisontal force in positive x-direction.
FCRefForce \Fx:= 20;

!Start the force references
```

Continues on next page

```
FCRefStart;
!Wait until conditions are met or timeout
FCCondWaitWhile;

!Saves the condition data at trig time
assemblydata:= FCGetProcessData(\dataattrigtime);

!Check if the force condition or TimeOut trigged the condition
IF (assemblydata.conditionstatus.force = FALSE) THEN
  !The Force conditions are met
ELSE
  !No conditions are met and program has done a timeout.
ENDIF
```

In this example the force in base frame must be larger than -40N. If the condition is not met the program will do a timeout after 20 seconds.

Syntax

```
FCGetProcessData '('
  [ '\' DataAtTrigTime ':=' < expression (IN) of datatype > ] ')'
```

A function with a return value of the data type `fcprocessdata`.

Related information

For information about	See
`fcprocessdata`	*fcprocessdata on page 175*

7.2.3 FCIsForceMode

Usage

The function `FCIsForceMode` is used to retrieve information whether or not the system is in force control mode.

Basic example

```
VAR bool fc_mode;
fc_mode := FCIsForceMode();
```

In this example `FCIsForceMode` returns TRUE if force mode is activated.

Return value

Data type: `bool`

The function returns TRUE when force control is activated, FALSE when it is deactivated.

Syntax

```
FCIsForceMode
```

A function with no arguments and a return value of the data type `bool`.

7.2.4 FCLoadID

Usage

The function FCLoadID is used to identify the load the sensor feels. Robot axes 5 and 6 move the load while the force sensor is used to detect the mass and center of gravity of the load.

The identified load is returned in a loaddata variable. At the present only the mass and center of gravity is identified. Inertias are set to zero. The identified load is used to calibrate the force sensor.

Note

It is important to get an accurate definition of the load in order to get a correct calibration of the sensor. If loadidErr is higher than 0.1, the load identification is not optimal and it is recommended to check the sensor coordinate system, see *FC Sensor on page 21*

See also *Troubleshooting drifting robot on page 61*.

Note

FCLoadID shall not be used when the sensor is room fixed.

Basic example

```
VAR loaddata my_load;
my_load := FCLoadID();
```

In this example the load in the sensor coordinate system is identified and the value is saved in the variable my_load.

See also *More examples on page 161*.

Return value

Data type: loaddata

The function returns a variable of type loaddata with the identified mass and center of gravity expressed in the force sensor coordinate system.

Arguments

```
FCLoadID ([\MaxMoveAx5] [\MaxMoveAx6] [\ReadingsPerPoint]
      [\PointsPerAxis] [\loadidErr] [\WarningsOff])
```

[\MaxMoveAx5]

Maximum movement of axis 5

Data type: num

This parameter decides the maximum movement of robot axis 5 during the load identification procedure. Based on the present position of the robot axis 5, it will move at the most MaxMoveAx5 degrees in both directions.

The unit is degrees. The default value is 180 degrees.

Continues on next page

`[\MaxMoveAx6]`

Maximum movement of axis 6

Data type: `num`

This parameter decides the maximum movement of robot axis 6 during the load identification procedure. Based on the present position of the robot axis 6, it will move at the most `MaxMoveAx6` degrees in both directions.

The unit is degrees. The default value is 180 degrees.

`[\ReadingsPerPoint]`

Data type: `num`

The number of readings in every point on the axis. The default value is 6 and an average of the readings is calculated.

`[\PointsPerAxis]`

Data type: `num`

The number of points on each axis to make the readings on. The default value is 6. A larger value slows down the identification but may improve the result.

`[\LoadidErr]`

Data type: `num`

`LoadidErr` is an INOUT parameter that returns a value between 0 and 1 depending on the result of the load identification. A value higher than 0.1 indicates that the identification is not optimal.

`[\WarningsOff]`

Data type: `switch`

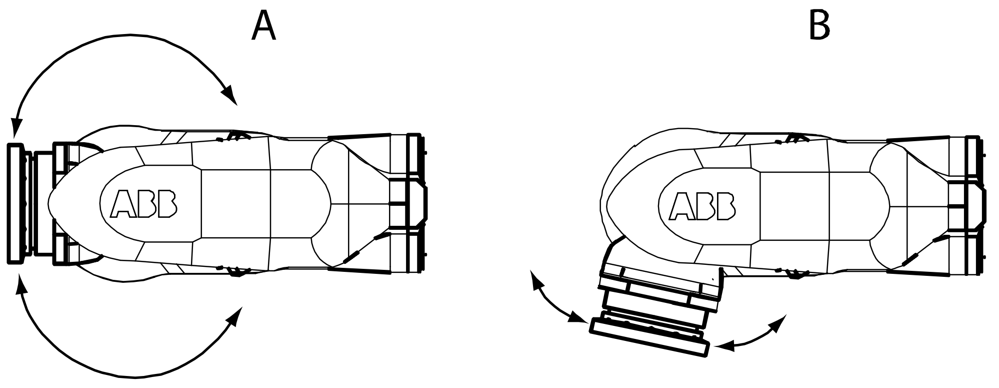
`WarningsOff` is used to disable displaying a message that axis 5 and axis 6 will move outside the specified maximum degrees. The warning message may be useful when running in manual mode, but in automatic mode it is better switched off.

Continues on next page

Program execution

Axis 5 is moved the same angle in both directions from the current position. The same is the case for axis 6.

Even if the movement range is not limited by `MaxMoveAx5` the movement is limited by the robot itself. If the axis is near its end position, the movement will be small and affect the accuracy of the load identification.



xx0500001793

A	The position of axis 5 allows large movements which result in a better load identification.
B	The position of axis 5 allows only a small movement in one direction.The movement in the other direction will be equally small. This results in a poor load identification.

More examples

Example 1

```
VAR num my_status;
VAR loaddata my_sensor_load;
my_sensor_load:= FCLoadID(\MaxMoveAx5:=30 \MaxMoveAx6:=90
     \ReadingsPerPoint:= 6 \PointsPerAxis:= 5 \LoadIDErr:=
     my_status);
```

The identified load is returned defined in the sensor coordinate system. The robot will move a maximum of 30 degrees in the direction of axis 5 and 90 degrees in direction of axis 6 based on the present robot position. The identification will be based on 6 readings in every point and 5 points on each axis. The variable `my_status` will show the status of the accomplished load identification.

Syntax

```
FCLoadID '('
  [ '\' MaxMoveAx5 ':=' < expression (IN) of num > ]
  [ '\' MaxMoveAx6 ':=' < expression (IN) of num > ]
  [ '\' ReadingsPerPoint ':=' < expression (IN) of num > ]
  [ '\' PointsPerAxis ':=' < expression (IN) of num > ]
  [ '\' LoadidErr ':=' < expression (INOUT) of num> ]
  ['\' WarningsOff ] ')'
```

A function with a return value of the data type `loaddata`.

Continues on next page

Related information

For information about	See
Calibrating the force sensor	*FCCalib on page 66*

7.3 Data types

7.3.1 fcboxvol

Usage

`fcboxvol` is used by the Force Control instruction `FCCondPos` and `FCSupvPos`. It defines a cubic volume that is used to verify if the TCP is inside or outside the volume.

Description

`fcboxvol` consists of minimum and maximum values in the directions x, y and z. The numbers refer to the unit mm. By default, the directions refer to the work object coordinate system, but can be changed with arguments in `FCCondPos` and `FCSupvPos`.

Components

`xmin`

Data type: `num`

Minimum value in the x direction. The unit is mm.

`xmax`

Data type: `num`

Maximum value in the x direction. The unit is mm.

`ymin`

Data type: `num`

Minimum value in the y direction. The unit is mm.

`ymax`

Data type: `num`

Maximum value in the y direction. The unit is mm.

`zmin`

Data type: `num`

Minimum value in the z direction. The unit is mm.

`zmax`

Data type: `num`

Maximum value in the z direction. The unit is mm.

Examples

Example 1

```
VAR fcboxvol my_box:= [-100, 100, -200, 200, -300, 300];
FCCondPos \Box:= my_box, 60;
```

In this example, a condition is set up to be true while the TCP is inside a box.

Continues on next page

Structure

```
< dataobject of fcboxvol >
  < xmin of num >
  < xmax of num >
  < ymin of num >
  < ymax of num >
  < zmin of num >
  < zmax of num >
```

Related information

For information about	See
Setting up position condition.	*FCCondPos on page 76*
Setting up position supervision.	*FCSupvPos on page 145*

7.3.2 fccondstatus

Usage

`fccondstatus` is used to define which of the conditions position, speed, force and time that are fulfilled.

Note

By default all components are set to true, if a condition is triggered the component defining that condition is set to false.

Components

`position`

Data type: `bool`

Defines if the `position` condition is triggered.

`speed`

Data type: `bool`

Defines if the `speed` condition is triggered.

`force`

Data type: `bool`

Defines if the `force` condition is triggered.

`time`

Data type: `bool`

Defines if the `time` condition is triggered.

Examples

Example 1

```
!Sets up a force condition.
FCCondForce \Xmin:=-40,TimeOut:=20;

!Defines a horisontal force in positive x-direction.
FCRefForce \Fx:= 20;

!Start the force references
FCRefStart;
!Wait until conditions are triggered or timeout
FCCondWaitWhile;

!Saves the condition data at trig time
assemblydata:= FCGetProcessData(\dataattrigtime);

!Check if the force condition or TimeOut trigged the condition
IF (assemblydata.conditionstatus.force = FALSE) THEN
  !The Force conditions are triggered
```

Continues on next page

```
ELSE
  !No conditions are triggered and program has done a timeout.
```

In this example the force in base frame must be larger than -40N. If the condition is not triggered the program will do a timeout after 20 seconds.

Structure

```
< dataobject of fccondstatus >
  < position of bool >
  < speed of bool >
  < force of bool >
  < time of bool >
```

Related information

For information about	See
The data type `fcprocessdata`	*fcprocessdata on page 175*

7.3.3 fccylindervol

Usage

`fccylindervol` is used by the Force Control instruction `FCCondPos` and `FCSupvPos`. It defines a cylinder volume that is used to verify if the TCP is inside or outside the volume.

Description

`fccylindervol` consists x, y and z values for the center of the cylinder bottom, and values for height and radius. The numbers refer to mm and height is always in the z direction. By default, the directions refer to the work object coordinate system, but can be changed with arguments in `FCCondPos` and `FCSupvPos`.

Components

`xcbottom`

Data type: `num`

X value for the center of the bottom of the cylinder. The unit is mm.

`ycbottom`

Data type: `num`

Y value for the center of the bottom of the cylinder. The unit is mm.

`zcbottom`

Data type: `num`

Z value for the center of the bottom of the cylinder. The unit is mm.

`height`

Data type: `num`

The height of the cylinder. The unit is mm.

`radius`

Data type: `num`

The radius of the cylinder. The unit is mm.

Examples

Example 1

```
VAR fccylindervol my_cylinder:= [300, 0, -200, 500, 250];
FCCondPos \Cylinder:= my_cylinder, 60;
```

In this example, a condition is set up to be true while the TCP is inside a cylinder.

Structure

```
< dataobject of fccylindervol >
  < xcbottom of num >
  < ycbottom of num >
  < zcbottom of num >
  < height of num >
  < radius of num >
```

Continues on next page

Related information

For information about	See
Setting up position condition	*FCCondPos on page 76*
Setting up position supervision	*FCSupvPos on page 145*

7.3.4 fcdamping

Usage

`fcdamping` is used by the instruction `FCAct` to specify the correlation between a contact force and the resulting speed.

Description

In Force Control, a contact force will make the TCP move with a speed proportional to the contact force. A contact torque will make the tool reorient with a speed proportional to the contact torque. A `fcdamping` variable defines the proportions between a force and the resulting speed, and a torque and the resulting reorientation speed, in the directions x, y and z.

The values are given as a percentage of the system parameter values defined in the type *FC Kinematics*.

Components

`xdamp`

Damping in the x direction

Data type: `num`

Force damping (relation between force and TCP speed) in the x direction. A smaller value means that the robot is more sensitive. By default the value is 100% (of the system parameter value), and allowed values are between 50% and infinity.

`ydamp`

Damping in the y direction

Data type: `num`

Force damping (relation between force and TCP speed) in the y direction. A smaller value means that the robot is more sensitive. By default the value is 100% (of the system parameter value), and allowed values are between 50% and infinity.

`zdamp`

Damping in the z direction

Data type: `num`

Force damping (relation between force and TCP speed) in the z direction. A smaller value means that the robot is more sensitive. By default the value is 100% (of the system parameter value), and allowed values are between 50% and infinity.

`rxdamp`

Damping in the rotational x direction

Data type: `num`

Torque damping (relation between torque and tool reorientation speed) around the x direction. A smaller value means that the robot is more sensitive. By default the value is 100% (of the system parameter value), and allowed values are between 50% and infinity.

Continues on next page

`rydamp`

Damping in the rotational y direction

Data type: `num`

Torque damping (relation between torque and tool reorientation speed) around the y direction. A smaller value means that the robot is more sensitive. By default the value is 100% (of the system parameter value), and allowed values are between 50% and infinity.

`rzdamp`

Damping in the rotational z direction

Data type: `num`

Torque damping (relation between torque and tool reorientation speed) around the z direction. A smaller value means that the robot is more sensitive. By default the value is 100% (of the system parameter value), and allowed values are between 50% and infinity.

Examples

Example 1

```
VAR fcdamping my_damping:=[50,100,100,300,300,300];
FCAct tool1 \DampingTune:=my_damping;
```

In this example, Force Control is activated with a more sensitive force damping in the x direction while the torque damping is stiffer in all directions.

Structure

```
< dataobject of fcdamping >
  < xdamp of num >
  < ydamp of num >
  < zdamp of num >
  < rxdamp of num >
  < rydamp of num >
  < rzdamp of num >
```

Related information

For information about	See
Activating Force Control.	*FCAct on page 63*
System parameters of the type *FC Kinematics*	*The FC Kinematics type on page 201*

7.3.5 fcforcevector

Usage

`fcforcevector` is used by the instruction `FCGetForce` to return force and torque in three dimensions (x, y, z).

Components

`xforce`

Data type: `num`

The force in x direction. The unit is N.

`yforce`

Data type: `num`

The force in y direction. The unit is N.

`zforce`

Data type: `num`

The force in z direction. The unit is N.

`xtorque`

Data type: `num`

The torque in x direction. The unit is Nm.

`ytorque`

Data type: `num`

The torque in y direction. The unit is Nm.

`ztorque`

Data type: `num`

The torque in z direction. The unit is Nm.

Example

In this example `FCGetForce()` get the values from the sensor and saves it in a variable `myForceVector`.

```
VAR fcforcevector myForceVector;
myforceVector := FCGetForce();
```

Structure

```
< dataobject of fcforcevector >
  < xforce of num >
  < yforce of num >
  < zforce of num >
  < xtorque of num >
  < ytorque of num >
  < ztorque of num >
```

Continues on next page

Related information

For information about	See
`FCGetForce`	*FCGetForce on page 154*

7.3.6 fcframe

Usage

- work object coordinate system
- tool coordinate system
- path coordinate system

Example

```
VAR tooldata tool1:=[TRUE,[[97.4,0,223],[1,0,0,0]], [5,[23,0,75],
     [1,0,0,0],0,0,0]];
VAR fcframe refcoordsys:=FC_REFFRAME_TOOL;
```

Predefined data

Constant	Comment
`FC_REFFRAME_TOOL`	Tool coordinate system
`FC_REFFRAME_WOBJ`	Work object coordinate system

Characteristics

`fcframe` is an alias data type for `num` and consequently inherits its characteristics.

7.3.7 fclindir

Usage

`fclindir` is used by the instruction `FCRefLine` and specifies the the reference direction.

Example

```
    FCRefLine FC_LIN_X, 500, 100;
```

Sets up, but does not activate, a linear shaped reference movement in the x-direction. The maximum speed is 500 mm/s and the amplitude is 100 mm.

Predefined data

Constant	Comment
`FC_LIN_X`	Reference in the x direction
`FC_LIN_Y`	Reference in the y direction
`FC_LIN_Z`	Reference in the z direction

Characteristics

`fclindir` is an alias data type for `num` and consequently inherits its characteristics.

Related information

For information about	See
`FCRefLine`	*FCRefLine on page 108*

7.3.8 fcprocessdata

Usage

`fcprocessdata` is used by the function `FCGetProcessdata` to retrieve data.

Description

`fcprocessdata` contains components, which are useful for supervising force control for assembly.

Components

`conditionstatus`

Data type: `fccondstatus`

Shows which of the conditions that are fulfilled.

`time`

Data type: `num`

Gives the time since the `FCCondWaitWhile` was executed.

`poseinrefcs`

Data type: `pose`

Gives the measured position and orientation in the reference movement coordinate system.

`speedinwobjframe`

Data type: `fcspeedvector`

Gives the linear and reorientation speed in the work object.

`sensorforce`

Data type: `fcforcevector`

Gives the measured force and torque in the force sensor coordinate system.

`forceframeforce`

Data type: `fcforcevector`

The measured force and torque in the force control coordinate system.

`force_control_active`

Data type: `bool`

Shows if force control is active.

Examples

In the examples below the variable `mydata` of the data type `fcprocessdata` is created. `FCGetProcessData` returns the data immediately or retrieves it when the wait while condition is triggered, and it is saved in `mydata`.

Example 1

```
VAR fcprocessdata mydata;
mydata := FCGetProcessData();
mydata.conditionstatus.position;
```

Continues on next page

```
TPWrite "If false, the position condition was triggered: "
     \Bool:=mydata.conditionstatus.position;
```

This example shows if the position condition was triggered.

Example 2

```
VAR fcprocessdata mydata;
mydata := FCGetProcessData();
TPWrite "Force in z direction, from sensor: "
     \Num:=mydata.sensorforce.zforce;
```

This example shows the force in the force sensor z direction. The unit is Newton.

Example 3

```
VAR fcprocessdata mydata;
mydata := FCGetProcessData();
TPWrite "Speed in x direction of the work object: "
     \Num:mydata.speedinwobjframe.vx;
```

This example shows the speed in the x direction of the work object. The unit is mm/s.

Structure

```
< dataobject of fcprocessdata >
  < conditionstatus of fccondstatus >
  < time of num >
  < poseinrefcs of pose >
  < speedinwobjframe of fcspeedvector >
  < sensorforce of fcforcevector >
  < forceframeforce of forcevector >
  < force_control_active of bool >
```

Related information

For information about	See
fccondstatus	*fccondstatus on page 165*
fcforcevector	*fcforcevector on page 171*
fcgetprocessdata	*FCGetProcessData on page 156*
fcspeedvector	*fcspeedvector on page 179*

7.3.9 fcplane

Usage

fcplane is used by the instructions FCRefCircle and FCRefSpiral. It defines in which plane the movement should be carried out.

Example

```
VAR fcplane myplane:=FCPLANE_XY;
FCRefSpiral myplane, 50, 100, 10;
```

Sets up a spiral shaped reference movement in the xy plane.

Limitations

fcplane can only define the three basic planes of the reference movement coordinate system. To define another plane, use the instruction FCRefMoveFrame to reorient the reference movement coordinate system.

Predefined data

Constant	Comment
FCPLANE_XY	the xy plane
FCPLANE_XZ	the xz plane
FCPlLANE_YZ	the yz plane

Characteristics

fcplane is an alias data type for num and consequently inherits its characteristics.

Related information

For information about:	Also see
Setting up circular reference movement.	*FCRefCircle on page 104*
Setting up spiral reference movement	*FCRefSpiral on page 114*
Defining another plane.	*FCRefMoveFrame on page 110*

7.3.10 fcrotdir

Usage

`fcrotdir` is used by the instruction `FCRefRot` and specifies a rotation around a chosen axis.

Example

```
    FCRefRot FC_ROT_Z, 5, 10;
```

Sets up a rotation around the work object z direction. When activated the TCP will rotate back and forth around the z-axis with an amplitude of 10 degrees The maximum speed will be 5 degrees per second.

Predefined data

Constant	Comment
FC_ROT_X	Rotation around the x axis
FC_ROT_Y	Rotation around the y axis
FC_ROT_Z	Rotation around the z axis

Characteristics

`fcrotdir` is an alias data type for `num` and consequently inherits its characteristics.

Related information

For information about.	Also see
FCRefRot	*FCRefRot on page 112*

7.3.11 fcspeedvector

Usage

`fcspeedvector` is used by the instruction `FCGetProcessdata` to return speed in and around three dimensions (x, y, z).

Components

`vx`

Data type: `num`

The speed in x direction. The unit is mm/s

`vy`

Data type: `num`

The speed in y direction. The unit is mm/s

`vz`

Data type: `num`

The speed in z direction. The unit is mm/s.

`wx`

Data type: `num`

The speed around the x axis. The unit is deg/s.

`wy`

Data type: `num`

The speed around the y axis. The unit is deg/s.

`wz`

Data type: `num`

The speed around the z axis. The unit is deg/s.

Examples

Example

```
VAR fcprocessdata mydata;
mydata := FCGetProcessData();
mydata.speedinwobjframe.vx;
```

In this examples a variable mydata is created and the data type is `fcprocessdata`. `FCGetProcessData` returns the data either from when the function is called or when the user defined condition were set true and saves it in mydata. The datatype of `speedinwobjframe` is `fcspeedvector`, so by adding `vx` you get the speed in the x direction.The unit is mm/s.

Structure

```
< dataobject of fcspeedvector >
  < vx of num >
  < vy of num >
  < vz of num >
```

Continues on next page

```
< wx of num >
< wy of num >
< wz of num >
```

Related information

For information about	See
fcprocessdata	*fcprocessdata on page 175*
FCGetProcesData	*FCGetProcessData on page 156*

7.3.12 fcspherevol

Usage

`fcspherevol` is used by the Force Control instruction `FCCondPos` and `FCSupvPos`. It defines a spherical volume that is used to verify if the TCP is inside or outside the volume.

Description

`fcspherevol` consists x, y and z values for the center of the sphere, and a radius value . By default, the directions refer to the work object coordinate system, but can be changed with arguments in `FCCondPos` and `FCSupvPos`.

Components

`xc`

Data type: `num`

X value for the center of the sphere. The unit is mm.

`yc`

Data type: `num`

Y value for the center of the sphere. The unit is mm.

`zc`

Data type: `num`

Z value for the center of the sphere. The unit is mm.

`radius`

Data type: `num`

The radius of the sphere. The unit is mm.

Examples

Example 1

```
VAR fcspherevol my_sphere:= [-100, 100, -200, 20];
FCCondPos \Sphere:= my_sphere, 60;
```

In this example, a condition is set up to be true while the TCP is inside a sphere.

Structure

```
< dataobject of fcspherevol >
  < xc of num >
  < yc of num >
  < zc of num >
  < radius of num >
```

Related information

For information about	See
Setting up position condition	*FCCondPos on page 76*
Setting up position supervision	*FCSupvPos on page 145*

7.3.13 fcspdchgtunetype

Usage

`fcspdchgtunetype` is used by the instructions `FCSpdChgTunSet` and `FCSpdChgTunReset` to select which system parameter should be changed.

The alternatives are:

- Speed ratio min
- No of speed levels

Examples

In this example the FC SpeedChange system parameter *Speed ratio min* is set to to 0.2.

```
FCSpdChgTunSet 0.2, FC_SPEED_RATIO_MIN;
```

Predefined data

Constant	Comment
`FC_SPEED_RATIO_MIN`	Speed ratio min
`FC_NO_OF_SPEED_LEVELS`	No of speed levels

Characteristics

`fcspdchgtunetype` is an alias data type for `num` and consequently inherits its characteristics.

Related information

For information about	See
Setting FC SpeedChange system parameter to a new value	*FCSpdChgTunSet on page 135*
Reset FC SpeedChange system parameter to its original value	*FCSpdChgTunReset on page 137*

7.3.14 fcxyznum

Usage

fcxyznum is used by the Force Control instruction FCRefForce and FCRefTorque. It is used to define amplitude and period that are specified in three directions.

Description

fcxyznum consist of one value in each direction x, y and z.

Components

x

Data type: num

Value in the x direction.

y

Data type: num

Value in the y direction.

z

Data type: num

Value in the z direction.

Examples

Example 1

```
VAR fcxyznum my_amp:=[10,0,0];
VAR fcxyznum my_period:=[1,0,0];
FCRefForce \Amp:=my_amp \Period:=my_period;
```

In this example, an oscillating force reference with amplitude10 N and period of 1 second in the x direction of the force control coordinate system.

Predefined data

Constant	Value	Comment
pi	3.1415926	
EOF_BIN	-1	End of file (binary)
EOF_NUM	9.998E36	End of file (decimal numerical)

Structure

```
< dataobject of fcxyznum >
  < x of num >
  < y of num >
  < z of num >
```

Related information

For information about	See
Force reference	*FCRefForce on page 106*

Continues on next page

For information about	See
Torque reference	*FCRefTorque on page 122*

8 System parameters

8.1 Type *Robot*

8.1.1 Use FC Master

Parent

Use FC Master belongs to the type *Robot*, in the topic *Motion*.

Cfg name

use_fc_master

Usage

Use FC Master is given the same value as the parameter *Name* of the *FC Master* to use.

Prerequisite

An *FC Master* must be defined.

Allowed values

A string with maximum 32 characters.

Related information

The FC Master type on page 186.

8.2 Type *FC Master*

8.2.1 The FC Master type

Overview

This section describes the type *FC Master* which belongs to the topic *Motion*. Each parameter of this type is described in a separate information topic in this section.

Cfg name

FC_MASTER

Type description

The type *FC Master* specifies which *FC Sensor*, *FC Kinematics,* and *FC Application* to use for Force Control.

8.2.2 Name

Parent

Name belongs to the type *FC Master*, in the topic *Motion*.

Cfg name

name

Usage

Defines the name of the *FC Master*.

Allowed values

A string with maximum 32 characters

8.2.3 Use FC Sensor

Parent

Use FC Sensor belongs to the type *FC Master*, in the topic *Motion*.

Cfg name

use_fc_sensor

Usage

Use FC Sensor is given the same value as the parameter *Name* of the *FC Sensor* to use.

Prerequisites

An FC sensor must be defined.

Allowed values

A string with maximum 32 characters.

Related information

The FC Sensor type on page 192.

8.2.4 Use FC Kinematics

Parent

Use FC Kinematics belongs to the type *FC Master*, in the topic *Motion*.

Cfg name

use_fc_kinematics

Usage

Use FC Kinematics is given the same value as the parameter *Name* of the *FC Kinematics* to use.

Allowed values

A string with maximum 32 characters.

Related information

The FC Kinematics type on page 201

8.2.5 Use FC Application

Parent

Use FC Application belongs to the type *FC Master*, in the topic *Motion*.

Cfg name

use_fc_application

Usage

Use FC Application is given the same value as the parameter *Name* in the *FC Application* to use.

Allowed values

A string with maximum 32 characters.

Related information

The FC Application type on page 207

8.2.6 Use FC Speed Change

Parent

Use FC Speed Change belongs to the type *FC Master*, in the topic *Motion*.

Cfg name

use_fc_speed_change

Usage

Use FC Speed Change is given the same value as the parameter *Name* in the *FC Speed Change* to use.

Allowed values

A string with maximum 32 characters.

Related information

The FC Application type on page 207.

8.3 Type *FC Sensor*

8.3.1 The FC Sensor type

Overview

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

Cfg name

FC_SENSOR

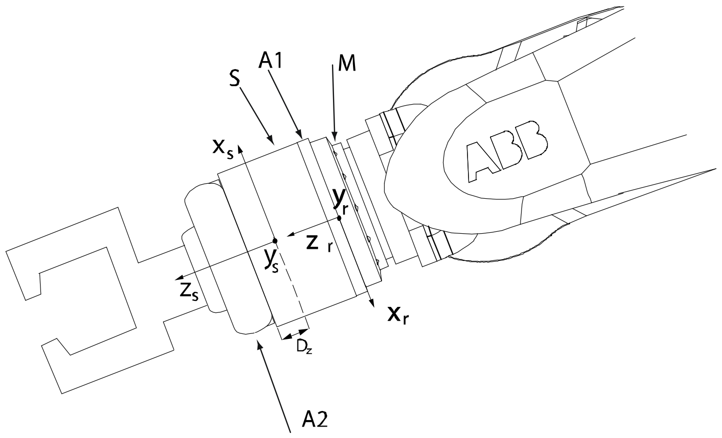
Type description

The type *FC Sensor* is used to define the Force Control sensor. The sensor can be either fixed in the room or mounted on a robot, as specified by the parameter *Force Sensor Mount Unit Name*. It can be a full-fledged sensor measuring both force and torque (6 DOF) or a sensor measuring only force, which is specified by the parameter *Force Sensor Type*.

The sensor has a built in coordinate system measuring forces in x, y and z directions. In order to translate the measured values to other coordinate systems, the sensor coordinate system must be defined. *Force Sensor Frame x - z* defines the position of the sensor coordinate system relative the robot's tool0 coordinate system (robot mounted sensor) or the world coordinate system (room fixed sensor). *Force Sensor Frame q1 - q4* defines the orientation of the sensor coordinate system, relative the robot's tool0 coordinate system (robot mounted sensor) or the world coordinate system (room fixed sensor).

Continues on next page

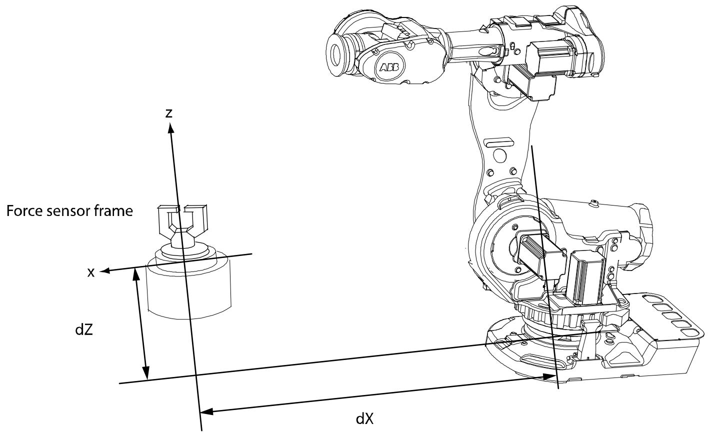
Illustration - robot mounted sensor



xx0500001554

x	Coordinate system for the robot's tool0.
x	Coordinate system for the sensor.
M	Robot tool flange.
A1	Adapter plate on the inside (not always used).
A2	Adapter plate on the outside (not always used).
D	*Basic transform Dz* is the distance between the sensor front and the coordinate system of the sensor.
S	Force sensor.

Illustration - room fixed sensor



xx0600003077

```
Force Sensor Frame = (dX,0,dZ), (1, 0, 0, 0)
```

8.3.2 Name

Parent

Name belongs to the type *FC Sensor*, in the topic *Motion*.

Cfg name

name

Usage

Defines the name of the *FC Sensor*.

Allowed values

A string with maximum 32 characters.

8.3.3 Force Sensor Mount Unit Name

Parent

Force Sensor Mount Unit Name belongs to the type *FC Sensor*, in the topic *Motion*.

Cfg name

force_sensor_mount_unit_name

Description

Defines on which mechanical unit the force sensor is mounted.

The value should be ROB_1, ROB_2, ROB_3 or ROB_4 when the sensor is mounted on a robot. When the sensor is room fixed the value should be left empty.

Allowed values

ROB_1, ROB_2, ROB_3, ROB_4 or empty.

8.3.4 Force Sensor Type

Parent	*Force Sensor Type* belongs to the type *FC Sensor*, in the topic *Motion*.
Cfg name	force_sensor_type
Description	Defines the type of sensor. If it is a 6 degree of freedom sensor measuring both force and torque the value should be *Force and Torque.* If it is a pure force sensor the value should be *Only Force.*
Allowed values	*Force and Torque* *Only Force*

8.3.5 Force Sensor Frame x - z

Parent

Force Sensor Frame x - z belongs to the type *FC Sensor*, in the topic *Motion*.

Cfg name

force_sensor_frame_pos_x

force_sensor_frame_pos_y

force_sensor_frame_pos_z

Description

Defines the position of the force sensor frame in relation to tool0 (robot mounted sensor) or the world frame (room fixed sensor).

Usage

If the sensor is mounted on a robot, the sensor frame is specified with regard to the robot's tool0 coordinate system. *Force Sensor Frame x - z* defines the distance from the center of the robot's mounting plate to the center of the sensor's coordinate system.

Normally *Force Sensor Frame x* and *Force Sensor Frame y* are set to zero. *Force Sensor Frame z* specifies the thickness of the sensor including the adaptor between robot and sensor (if any).

Consult the sensor supplier for detailed data if the measurement origin is not at the surface of the sensor.

If the sensor is fixed in the room the sensor frame is defined in relation to the world frame (robot base in normal cases). Assuming that the sensor z direction is facing the robot 2 meters away from the robot base in the x-direction at 1.5 m height, *Force Sensor Frame x* should be set to 2, Force Sensor Frame y to 0 and Force Sensor Frame to 1.5.

Allowed values

A value between -10 and 10 meters.

8.3.6 Force Sensor Frame q1 - q4

Parent

Force Sensor Frame q1 - q4 belong to the type *FC Sensor*, in the topic *Motion*.

Cfg name

force_sensor_frame_orient_u0

force_sensor_frame_orient_u1

force_sensor_frame_orient_u2

force_sensor_frame_orient_u3

Description

Defines the orientation of the force sensor coordinate system with respect to the robot’s tool0 coordinate system (robot mounted sensor) or the world coordinate system (room fixed sensor). The orientation is specified as four quaternion values.

Allowed values

A value between -1 and 1.

Related information

For more information on how to calculate quaternions, see the section about the data type `orient` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

8.3.7 Noise level

Parent	*Noise level* belongs to the type *FC Sensor*, in the topic *Motion*.
Cfg name	force_sensor_noise
Description	Defines the highest noise level at which a force sensor calibration should be allowed. Used to check that the robot is standing still at a force sensor calibration for example.
Usage	If the process is noisy and `FCCalib` or `FCLoadId` fails the value can be increased.
Allowed values	A value between 1 and 1000.

8.3.8 Force Sensor Scaling Factor fx, fy, fz, tx, ty, tz

Parent

Force Sensor Scaling Factor belongs to the type *FC Sensor*, in the topic *Motion*.

Description

The parameters *Force Sensor Scaling Factor fx, fy, fz* defines the scaling of the force in the X, Y, and Z axis from raw sensor data.

The parameters *Force Sensor Scaling Factor tx, ty, tz* defines the scaling of the torque in the X, Y, and Z axis from raw sensor data.

Usage

The SRI sensors have no scaling factors.

The ATI sensor has as scaling factor of 0.000001 for all axes.

Allowed values

A value between -100 and 100.

Default value is 1 (no scaling is done).

8.4 Type *FC Kinematics*

8.4.1 The FC Kinematics type

Overview

This section describes the type *FC Kinematics* which belongs to the topic *Motion*. Each parameter of this type is described in a separate information topic in this section.

Cfg name

FC_KINEMATICS

Type overview

The type *FC Kinematics* is used to define Force Control damping. Damping is a definition of how large contact force is required for the robot to move at a certain speed.

8.4.2 Name

Parent

Name belongs to the type *FC Kinematics*, in the topic *Motion*.

Cfg name

name

Description

Name defines the name of the *FC Kinematics*.

Usage

Name is used to reference the *FC Kinematics* from the parameter *Use FC Kinematics* in the type *FC Master*.

Allowed values

A string with maximum 32 characters.

8.4.3 Damping in Force x Direction - Damping in Force z Direction

Parent

Damping in Force x Direction - *Damping in Force z Direction* belong to the type *FC Kinematics*, in the topic *Motion*.

Cfg name

damping_fx

damping_fy

damping_fz

Description

Damping in Force ... Direction defines the damping of forces in kinematics in the x, y, or z direction.

Usage

Defines how many Newtons are required to make the robot move 1 m/s. The higher the value, the less responsive the robot gets.

The damping can be different in different directions. That is why there is one parameter for x, one for y and one for z.

The damping can be tuned (as a percentage of the system parameter values) by the RAPID instruction `FCAct`.

WARNING

A too low damping value can make the robot unstable.

Make sure the damping is not too low, even if the tuning level in the `FCAct` instruction is 50%. If the robot drifts away by itself, or if it vibrates, increase the damping value.

Allowed values

A value between min and 10,000,000 Ns/m.

Note

For each robot type there exists minimum allowed values of the damping. It is not possible to set the damping lower than these values.

8.4.4 Damping in Torque x Direction - Damping in Torque z Direction

Parent

Damping in Torque x Direction - *Damping in Torque z Direction* belong to the type *FC Kinematics*, in the topic *Motion*.

Cfg name

damping_tx

damping_ty

damping_tz

Description

Damping in Torque ... Direction defines the damping of torque in kinematics in the x, y, or z direction.

Usage

Defines how many Nm are required to make the robot move 1 rad/s. The higher the value, the less responsive the robot gets.

The damping can be different in different directions. That is why there is one parameter for x, one for y and one for z.

The damping can be tuned (as a percentage of the system parameter values) by the RAPID instruction `FCAct`.

WARNING

Too low a damping value can make the robot unstable.

Make sure the damping is not too low, even if the tuning level in the `FCAct` instruction is 50%. If the robot slowly rotates by itself, or if it vibrates, increase the damping value.

Allowed values

A value between minimum and 10,000,000 Nms/rad.

Note

For each robot type there is a minimum allowed value for damping. It is not possible to set the damping value lower than the minimum value.

8.4.5 Bandwidth of force frame filter

Parent	*Bandwidth of force frame filter* belongs to the type *FC Kinematics*, in the topic *Motion*.
Cfg name	force_frame_ filter_bandwidth
Description	*Bandwidth of force frame filter* defines the bandwidth in Hz of a low pass filter used to filter measured forces, used for example in force conditions.
Usage	In applications where the measured force/torque are too noisy this parameter can be used to filter the signals in order to eliminate false triggering and errors.
Allowed values	A value between 0 and 125. A value larger than 100 will switch the filter off.

8.4.6 Bandwidth of force loop filter

Parent

Bandwidth of force loop filter belongs to the type *FC Kinematics*, in the topic *Motion*.

Cfg name

force_loop_filter_bandwidth

Description

Bandwidth of force loop filter defines the bandwidth in Hz of a low pass filter used in the force control loop.

Usage

If the robot reacts too slowly for changes in contact force an increase of this parameter will make the robot more adaptable. Too high value will cause instability.

Allowed values

A value between 0.1 and 250. Default value is 3 Hz. A higher value will make the robot move compliant but may cause instability.

Limitations

This parameter cannot be used by the GoFa robot.

8.5 Type *FC Application*

8.5.1 The FC Application type

Overview

This section describes the type *FC Application* which belongs to the topic *Motion*. Each parameter of this type is described in a separate information topic in this section.

Cfg name

FC_APPLICATION

Type description

The type *FC Application* defines a number of limits for the reference values used in Force Control.

8.5.2 Name

Parent	*Name* belongs to the type *FC Application*, in the topic *Motion*.
Cfg name	name
Usage	*Name* is used to reference the *FC Application* from the parameter *Use FC Application* in the type *FC Master*.
Allowed values	A string with maximum 32 characters.

8.5.3 Max Ref Force

Parent

Max Ref Force belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

max_force

Usage

Max Ref Force defines the maximum allowed reference force, for the force specified by the instruction `FCRefForce`. If a larger value is used in `FCRefForce` the execution will continue with a reference force equal to *Max Ref Force* and a warning is shown in the event log.

Allowed values

A value between 0 and max.

Note

There is a maximum allowed reference that depends on the default load of the robot.

This maximum force is equal to the robot type's default load, times the gravity constant.

For a 60kg default load the highest allowed force is 60*9,81=589N.

If a higher value is set in *Max Ref Force* than the maximum allowed the higher value will be ignored.

8.5.4 Max Ref Force Change

Parent

Max Ref Force Change belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

max_force_change

Description

Max Ref Force Change defines the maximum allowed change in the reference force.

Usage

When the instruction `FCRefStart` is executed, the force is ramped up to the desired reference force specified in `FCRefForce`. This ramping is determined by *Max Ref Force Change*.

If a very large oscillating reference force is specified in `FCRefForce`, the oscillations are limited so that the change in force never exceeds *Max Ref Force Change*.

Allowed values

A value between 0 and 10,000 N/s.

8.5.5 Max Ref Torque

Parent

Max Ref Torque belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

max_torque

Description

Max Ref Torque defines the maximum allowed reference torque.

Usage

The reference torque, specified by the instruction `FCRefTorque`, cannot be larger than *Max Ref Torque*. If a larger value is used in `FCRefTorque` the execution will continue with a reference torque equal to *Max Ref Torque* and a warning is shown in the event log.

Allowed values

A value between 0 and max Nm.

Note

There is a maximum allowed reference that depends on the robots default load.

8.5.6 Max Ref Torque Change

Parent

Max Ref Torque Change belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

max_torque_change

Description

Max Ref Torque Change defines the maximum allowed reference torque change.

Usage

When the instruction `FCRefStart` is executed, the torque is ramped up to the desired reference torque specified in `FCRefTorque`. This ramping is determined by *Max Ref Torque Change*.

If a very large oscillating reference torque is specified in `FCRefTorque`, the oscillations are limited so that the change in torque never exceeds *Max Ref Torque Change*.

Allowed values

A value between 0 and 100,000 Nm/s.

8.5.7 Max Ref TCP Speed

Parent

Max Ref TCP Speed belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

max_lin_speed

Description

Max Ref TCP Speed defines the maximum allowed linear speed for the reference movement of Force Control.

Usage

The reference movement, specified by the instructions `FCRefSpiral`, `FCRefCircle`, or `FCRefLine`, cannot generate a speed larger than *Max Ref TCP Speed*. If the specified reference movement would result in a larger TCP speed, the speed will be limited to *Max Ref TCP Speed*.

Allowed values

A value between 0 and 10 m/s.

8.5.8 Max Ref Rot Speed

Parent

Max Ref Rot Speed belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

max_rot_speed

Description

Max Ref Rot Speed defines the maximum allowed rotational speed for the reference movement of Force Control.

Usage

The reference movement cannot generate a rotational speed larger than *Max Ref Rot Speed*. If the specified reference movement would result in a larger rotational speed, the speed will be limited to *Max Ref Rot Speed*.

Allowed values

A value between 0 and 10 rad/s.

8.5.9 Max Ref TCP Acc

Parent

Max Ref TCP Acc belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

max_lin_acc

Description

Max Ref TCP Acc defines the maximum allowed linear acceleration for the reference movement of Force Control.

Usage

When the instruction `FCRefStart` is executed, the TCP speed is ramped up to the desired reference movement specified in `FCRefSpiral`, `FCRefCircle`, or `FCRefLine`. This ramping is determined by *Max Ref TCP Acc*.

Allowed values

A value between 0 and 100 m/s^2.

Note

If the value is set too high the reference movement will result in a speed supervision error.

8.5.10 Max Ref Rot Acc

Parent

Max Ref Rot Acc belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

max_rot_acc

Description

Max Ref Rot Acc defines the maximum allowed rotational acceleration for the reference movement of Force Control.

Usage

When the instruction `FCRefStart` is executed, the rotational speed is ramped up to the desired reference movement. This ramping is determined by *Max Ref Rot Acc*.

Allowed values

A value between 0 and 100 rad/s^2.

Note

If the value is set to high the reference movement will lead to a speed supervision error.

8.5.11 Speed superv override

Parent

Speed superv override belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

spd_superv_override_factor

Description

Speed superv override is a parameter used to modify the default speed supervision. This parameter modifies the speed supervision in force control mode by a factor from 1 to 20.

Usage

When the robot is in force control mode the speed supervision might trig even during normal usage. If this happens it can be adjusted by defining a higher value of the parameter *Speed superv override*.

Allowed values

A value between 1 and 20.

8.5.12 Largest measured contact force

Parent

Largest measured contact force belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

axc_force_max

Description

If the measured contact force is larger than this value it is set to this value. The unit is N. The default value is 100 000 meaning this functionality is not active.

Usage

The parameter *Largest measured contact force* defines a truncation of measured force. If a measured force is larger than this value it is still assumed to be equal to this value. This can be useful in certain lead-tech applications but should otherwise not be used.

Allowed values

A value between 0 and 100000.

8.5.13 Lowest measured contact force

Parent

Lowest measured contact force belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

axc_force_min

Description

If the measured contact force is less than this value, it is set to zero. The unit is N.

Usage

Lowest measured contact force defines a threshold for the force, which needs to be exceeded in order to effect the robot position/speed. Too low a value might cause the robot to drift.

Allowed values

A value between 0 and 1000.

8.5.14 Largest measured contact torque

Parent	*Largest measured contact torque* belongs to the type *FC Application*, in the topic *Motion*.
Cfg name	axc_torque_max
Description	If a measured contact torque is larger than this value, it is set to this value. The unit is Nm. The default value is 100 000, meaning this function is inactive.
Usage	The parameter *Largest measured contact torque* defines a function of measured torque, if measured torque is larger than this value it is still assumed to be equal to this value. This can be useful in certain lead-tech applications but should otherwise not be used.
Allowed values	A value between 0 and 100000.

8.5.15 Lowest measured contact torque

Parent	*Lowest measured contact torque* belongs to the type *FC Application*, in the topic *Motion*.
Cfg name	axc_torque_min
Description	If a measured contact torque is less than this value, it is set to zero. The unit is Nm.
Usage	*Lowest measured contact torque* defines a threshold for the torque, which needs to be exceeded in order to effect the robot position/speed. Too low a value might cause the robot to drift.
Allowed values	A value between 0 and 1000.

8.5.16 Keep contact force at stop

Parent

Keep contact force at stop belongs to the type *FC Application*, in the topic *Motion*.

Cfg name

keep_contact_when_deactivating_fc

Description

Defines whether the robot should be allowed to remain in contact when force control execution is stopped.

Usage

If set to TRUE, the robot will keep the contact when the force control execution is stopped because of an emergency stop.

The currently active references will remain active during the stop, and when the force control execution is restarted. Additionally, it will be allowed to call `FCDeact` without a preceding call to `FCRefStop`, so that the robot remains in contact when switching to position control. After the switch, the active references are stopped.

Allowed values

TRUE/FALSE

8.5.17 Max Press TCP Speed

Parent	*Max Press TCP Speed* belongs to the type *FC Application*, in the topic *Motion*.
Cfg name	max_lin_speed_press
Description	Defines the maximum allowed TCP speed in FC Press instructions.
Usage	This parameter defines the highest TCP speed that can be used in FC Press instructions.
Allowed values	A value between 0 and 10 (m/s).

8.5.18 Max Press Rot Speed

Parent	*Max Press Rot Speed* belongs to the type *FC Application*, in the topic *Motion*.
Cfg name	max_rot_speed_press
Description	Defines the maximum allowed reorient speed in FC Press instructions.
Usage	This parameter defines the highest reorient speed that can be used in FC Press instructions.
Allowed values	A value between 0.01 and 1 (rad/s).

8.6 Type *FC Speed Change*

8.6.1 The FC Speed Change Type

Overview

This section describes the type *FC Speed Change*, which belongs to the topic *Motion*. Each parameter of the type is described in a separate information topic in this section.

Cfg name

FC_SPEED_CHANGE

Type description

The type *FC Speed Change* has a number of parameters used for the SpeedChange functionality available with the RobotWare option Machining FC.

8.6.2 Name

Parent

Name belongs to the type *FC Speed Change*, in the topic *Motion*.

Cfg name

name

Description

Defines the name of the *FC Speed Change*.

Allowed values

A string with maximum 32 characters.

8.6.3 Speed ratio min

Parent	*Speed ratio min* belongs to the type *FC Speed Change* in topic *Motion*.
Cfg name	speed_ratio_min
Description	Defines the minimum allowed speed ratio.
Usage	This parameter defines the lowest robot speed to be used (speed_ratio_min * programmed_speed).
Allowed values	A value between 0.02 and 1.

8.6.4 No of speed levels

Parent

No of speed levels belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

no_of_speed_levels

Description

Defines the number of discrete speed levels.

Allowed values

A value between 2 and 10.

8.6.5 Speed ratio delta

Parent

Speed ratio delta belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

speed_ratio_delta

Usage

The parameter limits acceleration/deceleration due to the SpeedChange functionality. A low value will give slower but smoother speed changes.

Allowed values

A value between 0.0001 and 1.

8.6.6 Speed max update period

Parent	*Speed max update period* belongs to the type *FC Speed Change* in topic *Motion*.
Cfg name	speed_max_update_period
Usage	This parameter defines the period of time after a speed change, during which the sensor signal will be disregarded. A short time will give faster reactions to overload but may cause the speed to vary too frequently.
Allowed values	A value between 0 and 1.

8.6.7 Feedback type

Parent	*Feedback type* belongs to the type *FC Speed Change* in topic *Motion*.
Cfg name	fdb_type
Usage	This parameter is used to decide which type of feedback should control the speed, that is, which sensor input is to be used for speed change control. The parameter *Disable check of saturation* can be used if it is likely that the power output will reach saturation level.
Allowed values	*Calib. Force Magn.* *Uncalib. Force Magn.*

8.6.8 Use Fdb LP filter

Parent

Use Fdb LP filter belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

fdb_lp_active

Description

Defines whether feedback low pass filter should be active.

Usage

If set to TRUE, the feedback values are low pass filtered before used. May be used to filter noisy signals.

Allowed values

TRUE/FALSE

8.6.9 Fdb LP filter bandwidth

Parent

Fdb LP filter bandwidth belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

fdb_lp_bandwidth

Usage

This parameter is used to filter the feedback values used in the speed change control. Setting it lower will slow down the reaction time for the speed change control.

Allowed values

A value between 1 and 250.

8.6.10 Maximum TCP speed

Parent	*Maximum TCP speed* belongs to the type *FC Speed Change* in topic *Motion*.
Cfg name	maximum_tcp_speed
Description	Defines the maximum original TCP speed for speed change (m/s).
Usage	If the user programs a speed above this value, the system will stop.
Allowed values	A value between 0.01 and 10.

8.6.11 Recover rule fdb ratio

Parent	*Recover rule fbd ratio* belongs to the type *FC Speed Change* in topic *Motion*.
Cfg name	recover_rule_fdb ratio
Description	Defines the maximum allowed feedback (fdb) to reference ratio when at lowest possible speed.
Usage	A feedback to reference ratio larger than this while having reduced speed to lowest level will trig recover behavior or stop robot. The recover function will be activated when the feedback signal is still too high when running at the lowest speed.
Allowed values	A value between 0.01 and 1000.

8.6.12 Decrease rule safety fdb ratio

Parent	*Decrease rule safety fdb ratio* belongs to the type *FC Speed Change* in topic *Motion*.
Cfg name	dec_ rule_safety_fdb_to_ref_ratio
Description	Defines the maximum feedback to reference ratio.
Usage	Speed will be reduced if the feedback to reference ratio is above this value for *Decrease rule safety fdb time* regardless of trends and changes of the feedback.
Allowed values	A value between 0.001 and 1000.

8.6.13 Decrease rule safety fdb time

Parent

Decrease rule safety fdb time belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

dec_ rule_safety_fdb_time

Description

Define the maximum time in seconds that the feedback to reference ratio can be continuously over *Decrease rule safety fdb ratio* before reducing robot speed.

Usage

Speed will be reduced if the feedback to reference ratio is above *Decrease rule safety fdb ratio* for this time regardless the trend of the feedback

Allowed values

A value between 0.001 and 1000.

8.6.14 Fdb trend step size

Parent

Fdb trend step size belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

fdb_trend_step_size

Description

Defines the minimum difference between consecutive fdb values to count as a change in feedback.

Usage

Used for deciding trends in the feedback that is needed for the SpeedChange rules. This parameter is used to compensate the effects of measurement noise on the trend calculation. Usually the value can be set 2 times the noise level.

Allowed values

A value between 0 and 1000.

8.6.15 Decrease rule 1 fdb ratio

Parent	*Decrease rule 1 fdb ratio* belongs to the type *FC Speed Change* in topic *Motion*.
Cfg name	dec_rule1_fdb_to_ref_ratio
Description	Part of condition 1 to decrease speed.
Usage	For ABB internal use only.
Allowed values	A value between 0.001 and 1000.

8.6.16 Decrease rule 1 fdb trend

Parent	*Decrease rule 1 fdb trend* belongs to the type *FC Speed Change* in topic *Motion*.
Cfg name	dec_rule1_fdb_trend
Description	Part of condition 1 to decrease speed.
Usage	For ABB internal use only.
Allowed values	A value between -10 and 10.

8.6.17 Decrease rule 2 fdb ratio

Parent

Decrease rule 2 fdb ratio belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

dec_rule2_fdb_to_ref_ratio

Description

Part of condition 2 to decrease speed.

Usage

For ABB internal use only.

Allowed values

A value between 0.001 and 1000.

8.6.18 Decrease rule 2 fdb trend

Parent	*Decrease rule 2 fdb trend* belongs to the type *FC Speed Change* in topic *Motion*.
Cfg name	dec_rule2_fdb_trend
Description	Part of condition 2 to decrease speed.
Usage	For ABB internal use only.
Allowed values	A value between -10 and 10.

8.6.19 Increase rule 1 fdb ratio

Parent

Increase rule 1 fdb ratio belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

inc_rule1_fdb_to_ref_ratio

Description

Part of condition 1 to increase speed.

Usage

For ABB internal use only.

Allowed values

A value between 0.001 and 1000.

8.6.20 Increase rule 1 fdb trend

Parent

Increase rule 1 fdb trend belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

inc_rule1_fdb_trend

Description

Part of condition 1 to increase speed.

Usage

For ABB internal use only.

Allowed values

A value between -10 and 10.

8.6.21 Increase rule 2 fdb ratio

Parent

Increase rule 2 fdb ratio belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

inc_rule2_fdb_to_ref_ratio

Description

Part of condition 2 to increase speed.

Usage

For ABB internal use only.

Allowed values

A value between 0.001 and 1000.

8.6.22 Increase rule 2 fdb trend

Parent

Increase rule 2 fdb trend belongs to the type *FC Speed Change* in topic *Motion*.

Cfg name

inc_rule2_fdb_trend

Description

Part of condition 2 to increase speed.

Usage

For ABB internal use only.

Allowed values

A value between -10 and 10.

9 Further references

9.1 TuneMaster

Overview

TuneMaster can be used to monitor forces in Force Control. Forces and torques in both force control coordinate system and sensor coordinate system can be viewed at the same time monitoring 6 signals each (for example x, y, z, wx, wy, wz).

Installation

This section only describes what is specific for setting up the TuneMaster for Force control. For more information see *Application manual - TuneMaster*.

Procedure

Follow these step to get started with the program.

Step	Action	Reference
1	Install TuneMaster.	*Application manual - TuneMaster*
2	Start TuneMaster.	
3	Define test signals.	Specified in section *Test signal number on page 247*.

Test signal number

To view the forces, a specific number needs to be entered in *Signal ident. man* of every channel in the test signal window.

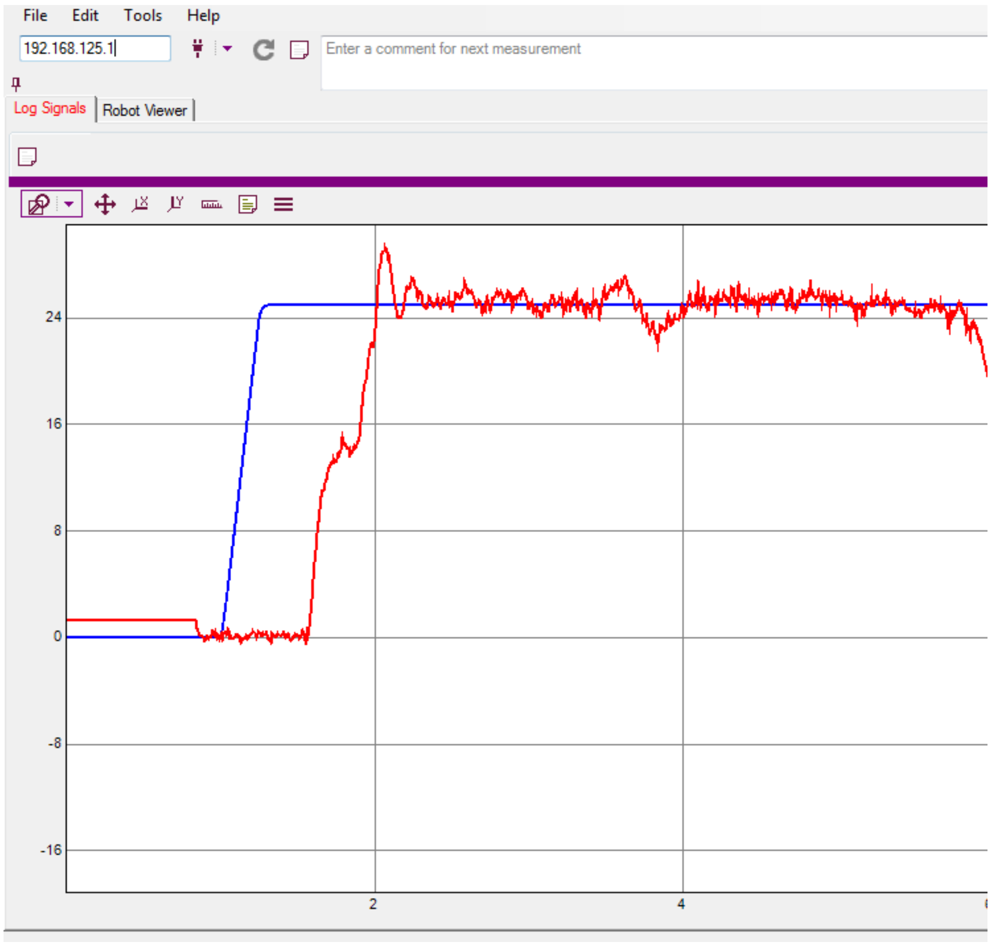
If the signals are not mapped to a specific axis always use axis 1.

It is important to note that in FC Pressure, the force control coordinate system is automatically rotated in such way that the z-axis of the force control coordinate system is always aligned with the pressure direction specified by the arguments `\Fx`, `\Fy` and `\Fz` to `FCPress1LStart`. This means that the test signal 209 should always be used for monitoring of the pressure force.

Signal number	Content (force component)
201	Sensor frame, x-direction.
202	Sensor frame, y-direction.
203	Sensor frame, z-direction.
204	Sensor frame, wx-direction.
205	Sensor frame, wy-direction.
206	Sensor frame, wz-direction.
207	Force frame, x-direction.
208	Force frame, y-direction.
209	Force frame, z-direction.
210	Force frame, wx-direction.

Continues on next page

Signal number	Content (force component)
211	Force frame, wy-direction.
212	Force frame, wz-direction.



xx1800000413

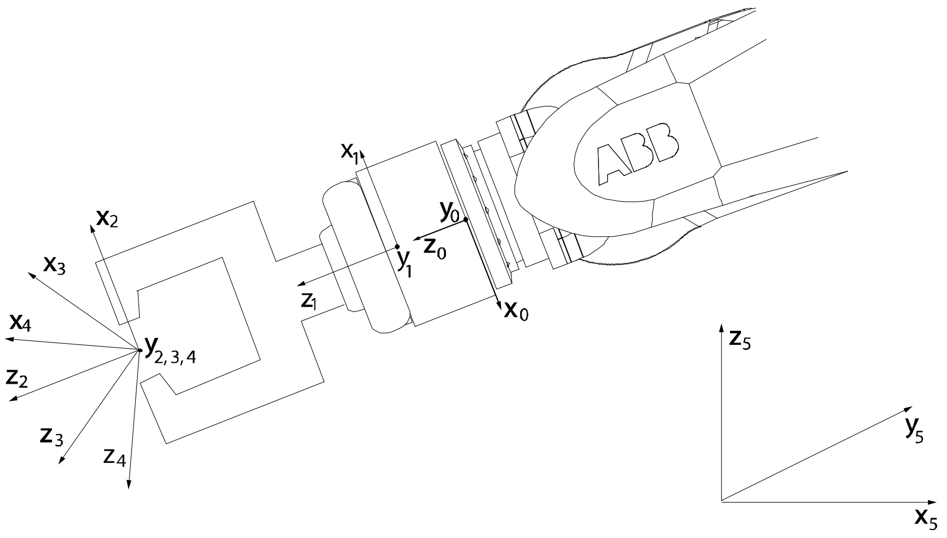
This table specifies the test signals for FCSpeedChange tuning.

Signal number	Content
401	Reference
402	Measurement (process force)
403	Speed ratio signal

9.2 The coordinate systems

About the coordinate systems

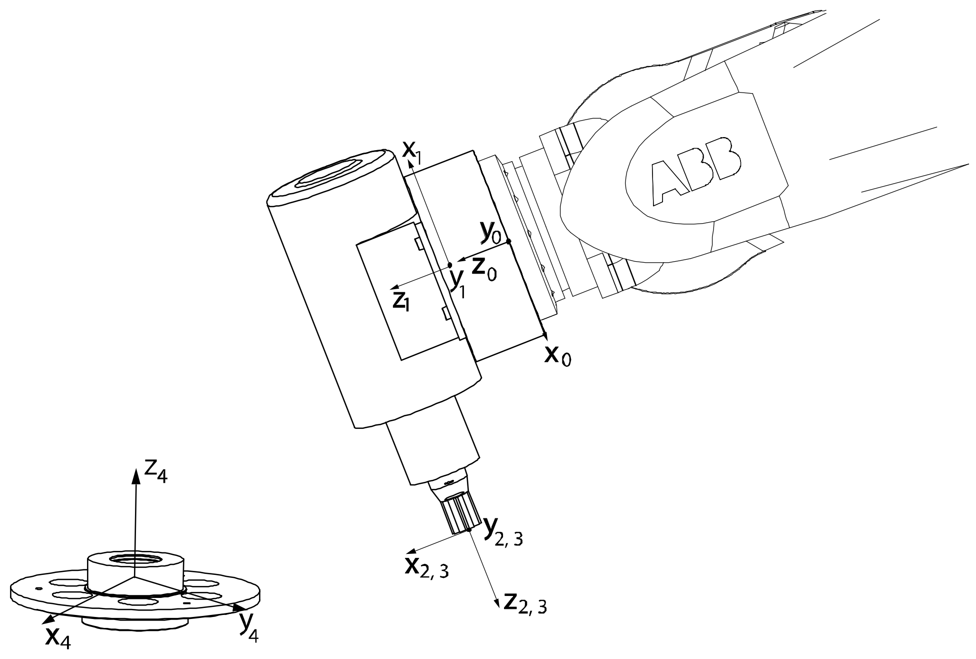
This is an overview over the new coordinate systems created for Force Control. For more information about the basic coordination system see related information.



xx0500002050

X0, Y0, Z0	Tool 0 coordinate system
X1, Y1, Z1	Sensor coordinate system
X2, Y2, Z2	Tool coordinate system
X3, Y3, Z3	Force control coordinate system
X4, Y4, Z4	Reference movement coordinate system
X5, Y5, Z5	Work object coordinate system

Continues on next page



xx0600003309

X0, Y0, Z0	Tool 0 coordinate system
X1, Y1, Z1	Sensor coordinate system
X2, Y2, Z2	Tool coordinate system
X3, Y3, Z3	Force control coordinate system
X4, Y4, Z4	Work object coordinate system

Force control coordinate system

The origin of the force control coordinate system is in the tool center point (TCP). The orientation is defined by the user in relation to the tool coordinate system, the work object coordinate system, or the path coordinate system.

Orient condition coordinate system

Orientation condition are defined in this coordinate system. The origin is the same as the work object coordinate system and the orientation is defined by an orient in relation to the work object coordinate system.

Orient supervision coordinate system

Orientation supervision are defined in this coordinate system. The origin is the same as the work object coordinate system and the orientation is defined by an orient in relation to the work object coordinate system.

Position condition coordinate system

Positions condition are defined in this coordinate system. It is defined by a pose in relation to the work object coordinate system.

Position supervision coordinate system

Positions supervision are defined in this coordinate system. It is defined by a pose in relation to the work object coordinate system.

Continues on next page

Reference movement coordinate system

The origin of the reference movement coordinate system is in the tool center point (TCP). The orientation is defined by the user in relation to the tool coordinate system or the work object coordinate system.

Sensor coordinate system

The origin and orientation of the sensor coordinate system depends on the manufacture and how it is mounted.

Tool 0 coordinate system

Tool 0 or the wrist coordinate system cannot be changed and is always the same as the mounting flange of the robot.

Tool coordinate system

The tool coordinate system is defined by the user.

Related information

For information about	See
Coordinate systems	*Technical reference manual - RAPID Overview*

9.3 Force Sensor interface

9.3.1 About the Force Sensor interface

Overview

The force control software has been designed with a generic interface for a 6 degree of freedom (6DOF) force / torque sensor.

ATI Industrial Automation have prepared sensors with correct settings, adapter plates and calibration files in order to meet the ABB interface requirements.

This section describes the interface as such and how to adapt a sensor to the requirements.

9.3.2 Sensors with less than six degrees of freedom

About this section

This section gives information on how to configure a sensor with less than six degrees of freedom.

Pure force sensor

If the system should use a pure force sensor it is important to define that the sensor only measures force. This is done under type *FC Sensor* in the topic *Motion*, by setting the parameter *Force Sensor Type* to *Only Force*.

9.3.3 Room fixed sensor

Overview

The normal case is to have the sensor mounted on a robot, but there is an option to have the sensor mounted at a stationary position in the work cell. The sensor must then be mounted on a tool or fixture in such a way that the contact force between the robot and the tool is registered.

Configuration

If the sensor is room fixed this must be defined under type *FC Sensor* in the topic *Motion*, by leaving the parameter *Force Sensor Mount Unit Name* empty.

The origin and the direction of the force sensor coordinate system are specified in relation to the world coordinate system.

RAPID

The RAPID function `FCLoadID` should not be used with a room fixed sensor.

`Loaddata` is not relevant to the RAPID instruction `FCCalib`, which is only used in order to find the offset for the sensor.

Note

The calibration convention is identifying the sensed force given from the sensor, which should be equal to the force with which the surrounding effects the robot. This means that there has to be different signs (+/-) in the calibration configuration if the sensor is room fixed or mounted on a robot.

Index

A
activate force control, 63
activation example, 42

C
calibrate force control, 66
circular movements, 104
coordinate systems, 249

D
deactivate force control, 90
drifting robot, 61

E
exceeding condition, 61

F
FC Application, type, 207
FCCondCombine, 68
FC Kinematics, type, 201
FC Master, type, 186
FC Pressure, 14
FC Sensor, type, 192
FC SpeedChange, 15
find object example, 42

I
installation, 17
installation example, 25

J
jog the robot, 51

M
monitor forces, 247
move the robot, 51

O
orientation condition, 72, 78, 81, 84, 87–88

P
position condition, 71, 75–76, 84, 87–88, 164, 168, 181
position search example, 43
ppmove, 51

R
reference movements, 45
reorientation speed condition, 71, 75, 78–79, 84, 87, 89

S
set up example, 25
set up force control, 17
supervision example, 44
system parameters, 21

T
tcp speed condition, 82
torque condition, 85
TuneMaster, 247

W
wait while condition, 88

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

3HAC090251-001, Rev B, en