



ROBOTICS

Application manual

Conveyor tracking



Trace back information:
Workspace 26-1 version a13
Checked in 2026-04-21
Skribenta version 5.6.019

Application manual

Conveyor tracking

RobotWare 8.0

Document ID: 3HAC066561-001

Revision: Q

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damage to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission.

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

Table of contents

Overview of this manual	9
Network security	12
Open source and 3rd party components	13
1 Introduction to conveyor tracking	15
1.1 Physical components	15
1.2 Features	17
1.3 Limitations	19
1.4 Principles of conveyor tracking	20
2 Conveyor tracking modules	25
2.1 Network based conveyor tracking module	25
3 Installation	39
3.1 Hardware connections	39
3.2 Installing the encoder for conveyor tracking	40
3.2.1 Encoder location	41
3.2.2 Encoder cables	42
3.3 Connecting the encoder to the CTM	43
3.4 Connecting a camera to the CTM	47
3.5 Connecting the sync switches to the CTM	48
4 Conveyor Tracking tab in RobotStudio	49
5 Configuration and calibration	57
5.1 About software installation and configuration	57
5.2 Configuring conveyor tracking module	58
5.3 Verifying the installation of encoders and sensors	61
5.4 Configuration of robot controllers	63
5.5 Calibrating CountsPerMeter	66
5.6 Calibrating the conveyor base frame	67
5.7 Defining conveyor start window and sync separation	71
5.8 Avoiding robot reach problems	72
5.9 Using a trigger output on the CTM	73
5.10 Defining the robot adjustment speed	74
5.11 Additional adjustments	75
5.12 Configuring a track motion to follow a conveyor	76
5.13 Installing conveyor tracking software	78
5.14 Installing additional conveyors work areas for conveyor tracking	79
6 Programming	81
6.1 Programming conveyor tracking	81
6.2 Working with the object queue	82
6.3 Activating the conveyor	83
6.4 Defining a conveyor coordinated work object	84
6.5 Waiting for a work object	85
6.6 Programming the conveyor coordinated motion	86
6.7 Dropping a work object	87
6.8 Entering and exiting conveyor tracking motion in corner zones	88
6.9 Information on FlexPendant	89
6.10 Programming considerations	90
6.11 Finepoint programming	91
6.12 Operating modes	92

7	System parameters	93
7.1	Introduction	93
7.2	Topic Motion	94
7.3	Topic Process	96
8	RAPID reference	99
8.1	Instructions	99
8.1.1	WaitWObj - Wait for work object on conveyor	99
8.1.2	DropWObj - Drop work object on conveyor	102
8.1.3	CnvPredictReach - High speed conveyors	103
9	Advanced queue tracking	109
9.1	Introduction to advanced queue tracking	109
9.2	Working with the object queue	111
10	Circular conveyor tracking	113
10.1	Introduction to circular conveyor tracking	113
10.2	Encoder type selection and location	115
10.3	Installation and configuration	116
10.4	Additional motion settings	117
10.5	Calibrating the conveyor base frame	118
10.6	Manually calibrating the conveyor base frame	119
10.7	TCP measurement method	120
11	Accelerating conveyors	123
11.1	Introduction to accelerating conveyors	123
11.2	System parameters	124
11.3	Predicting speed changes	125
11.4	UseAccProfile - Use acceleration profile	126
12	Indexing conveyors	129
12.1	Description of indexing conveyor options	129
12.2	Tracking indexing conveyors	130
12.2.1	Setting up tracking for an indexing conveyor	130
12.2.2	System parameters	131
12.2.3	Using indexing conveyor tracking from RAPID	132
12.2.4	RecordProfile	133
12.2.5	WaitAndRecProf	134
12.2.6	StoreProfile	136
12.2.7	LoadProfile	137
12.2.8	ActivateProfile	138
12.2.9	DeactProfile	139
12.2.10	CnvGenInstr	140
12.3	Indexing conveyor with servo control (Indexing Conveyor Control)	143
12.3.1	Introduction to indexing conveyors with servo control	143
12.3.2	Setting up a servo controlled indexing conveyor	145
12.3.3	System parameters and configuration files	146
12.3.4	Testing the indexing conveyor setup	151
12.3.5	Calibrating the base frame	152
12.3.6	indcnvdata	153
12.3.7	IndCnvInit	154
12.3.8	IndCnvEnable and IndCnvDisable	155
12.3.9	IndCnvReset	156
12.3.10	IndCnvAddObject	157
12.3.11	RAPID programming example	158
12.3.12	Minimizing trigger time	161

13 Conveyor tracking and MultiMove	163
13.1 About conveyor tracking and MultiMove	163
13.2 Configuration example for UnsyncCnv	165
13.3 Configuration example for SyncCnv	167
13.4 Tasks and programming techniques	169
13.5 Independent movements, example UnsyncCnv	171
13.6 Coordinated synchronized movements, example SyncCnv	173
13.7 Motion principles	176
13.8 Combining synchronized and un-synchronized mode	177
14 Troubleshooting	179
14.1 Overview	179
14.2 Event messages	180
14.3 System parameters	183
14.4 Robot path characteristics	184
14.5 Power failure	186
14.6 Collision detection	187
14.7 Technical support	188
Index	189

This page is intentionally left blank

Overview of this manual

About this manual

This manual describes the options *Conveyor Tracking*, *Indexing Conveyor Control*, and *Tracking Unit Interface* for OmniCore. Conveyor tracking, or line tracking, is when the robot follows a work object mounted on a moving conveyor.

The option *Indexing Conveyor Control* is used when the servo for an indexing conveyor is controlled by the robot controller.

The Tracking Unit Interface is used to establish communication with a remote Conveyor Tracking Module.

Usages

The manual describes how the options are installed, programmed, and operated.



Note

It is the responsibility of the integrator to conduct a risk assessment of the final application.

It is the responsibility of the integrator to provide safety and user guides for the robot system.

References

The manual contains references to the following information products:

RobotWare 7

Reference	Document ID
<i>Operating manual - OmniCore</i>	3HAC065036-001
<i>Operating manual - Integrator's guide OmniCore</i>	3HAC065037-001
<i>Operating manual - RobotStudio</i>	3HAC032104-001
<i>Application manual - Controller software OmniCore RobotWare 7</i>	3HAC066554-001
<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>	3HAC065038-001
<i>Technical reference manual - RAPID Overview</i>	3HAC065040-001
<i>Technical reference manual - RAPID kernel</i>	3HAC065039-001
<i>Technical reference manual - System parameters</i>	3HAC065041-001
<i>Application manual - Additional axes</i>	3HAC082287-001
<i>Application manual - MultiMove</i>	3HAC089689-001

RobotWare 8

Reference	Document ID
<i>Operating manual - OmniCore</i>	3HAC098383-001
<i>Operating manual - Integrator's guide OmniCore</i>	3HAC098381-001
<i>Operating manual - RobotStudio</i>	3HAC032104-001

Continues on next page

Reference	Document ID
<i>Application manual - Controller software OmniCore RobotWare 8</i>	3HAC098393-001
<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>	3HAC098387-001
<i>Technical reference manual - RAPID Overview</i>	3HAC098389-001
<i>Technical reference manual - RAPID kernel</i>	3HAC098388-001
<i>Application manual - Additional axes</i>	3HAC098394-001
<i>Technical reference manual - System parameters</i>	3HAC098390-001
<i>Application manual - MultiMove</i>	3HAC098578-001

Revisions

The following revisions of this manual have been released.

Revision	Description
A	Released with RobotWare 7.0.
B	Released with RobotWare 7.0.1. "Cyber security" replaced by "Cybersecurity" in entire manual.
C	Released with RobotWare 7.1. - Removed obsolete RAPID objects. UseReachableTargets, GetMaxUsageTime, ResetMaxUsageTime - New event message 50024: Corner path failure added in section Event messages on page 180. - Updated the section CnvPredictReach - High speed conveyors on page 103.
D	Released with RobotWare 7.2. - Note added in section Working with the object queue on page 111 saying that for CTM, the ERR_CNV_OBJ_LOST error is not supported. - Minor corrections.
E	Released with RobotWare 7.3. Information regarding the adjustment speed parameter updated in About software installation and configuration on page 57, Configuring conveyor tracking module on page 58, Defining the robot adjustment speed on page 74, Topic Process on page 96, Event messages on page 180, System parameters on page 183 and Robot path characteristics on page 184.
F	Released with RobotWare 7.4. - Reference to information about firewall settings in Integrator's Guide added in section Connecting the CTM to the robot controller on page 44. - CTM connection information updated in section Connecting the CTM to the robot controller on page 44.
G	Released with RobotWare 7.6. Added clarification about sharing of encoders between different work areas, see Configuring conveyor tracking module on page 58 and Configuration of robot controllers on page 63.
H	Released with RobotWare 7.7. Updated descriptions for DSQC2000 regarding advanced queue tracking.
J	Released with RobotWare 7.8. - Updated information about sync inputs and speed filter settings for DSQC2000. - Added type <i>Conveyor Internal</i> to section Topic Process on page 96.

Continues on next page

Revision	Description
K	Released with RobotWare 7.10. • Path to RobotWare installation folder updated in Installing additional conveyors work areas for conveyor tracking on page 79. • Information about firewall settings added in sections Configuring conveyor tracking module on page 58 and Configuration of robot controllers on page 63. • Updated information about discovery LED in section Network based conveyor tracking module on page 25. • Removed the information about the parameter <i>Use spline parameters</i> as this is obsolete.
L	Released with RobotWare 7.13. • Added the system parameter <i>Enable orientation correction</i>, see Type Robot on page 94.
M	Released with RobotWare 7.15. • Added a reference to firewall information in <i>Operating manual - RobotStudio</i>, see Conveyor Tracking tab in RobotStudio on page 49.
N	Released with RobotWare 7.19. • Updated the section Introduction to indexing conveyors with servo control on page 143. • Minor corrections.
P	Released with RobotWare 7.20. • Company name updated to reflect current legal entities.
Q	Released with RobotWare 8.0.  Note This user documentation is published before RobotWare 8 is released. • Company name updated to reflect current legal entities.

Network security

Network security

This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide, and continuously ensure, a secure connection between the product and to your network or any other network (as the case may be).

You shall establish and maintain any appropriate measures (such as, but not limited to, the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its entities are not liable for damage and/or loss related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

Open source and 3rd party components

Open source and 3rd party components

ABB products use software provided by third parties, including open source software. The following copyright statements and licenses apply to various components that are distributed inside the ABB software. Each ABB product does not necessarily use all of the listed third party software components. Licensee must fully agree and comply with these license terms or the user is not entitled to use the product. Start using the ABB software means accepting also referred license terms. The third party license terms apply only to the respective software to which the license pertains, and the third party license terms do not apply to ABB products. With regard to programs provided under the GNU general public license and the GNU lesser general public license licensor will provide licensee on demand, a machine-readable copy of the corresponding source code. This offer is valid for a period of three years after delivery of the product.

ABB software is licensed under the ABB end user license agreement, which is provided separately.

RobotWare

For RobotWare, there is license information in the folder \licenses in the RobotWare distribution package.

OpenSSL

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

CTM

For OleOS, the Linux based operating system used on the conveyor tracking module (CTM), a list of copyright statements and licenses is available in the file `/etc/licenses.txt` located on the CTM board and accessible via the console port or by downloading the file over SFTP.

For the CTM application, a list of copyright statements and licenses is available in the file `/opt/ABB.com/ctm/licenses.txt` located on the CTM board and accessible via the console port or by downloading the file over SFTP.

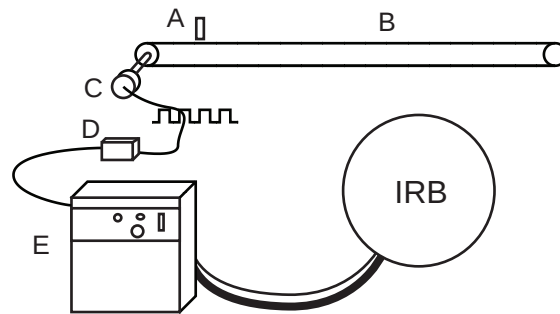
This page is intentionally left blank

1 Introduction to conveyor tracking

1.1 Physical components

Option Conveyor Tracking

A typical installation when using the option *Conveyor Tracking* includes the following components:



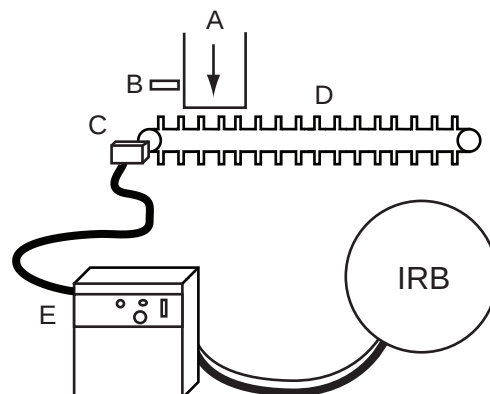
xx1200001074

A	Synchronization switch
B	Conveyor
C	Encoder, 24 V
D	Encoder interface
E	Robot controller
IRB	Robot

The encoder and synchronization switch are connected to the encoder interface.

Option Indexing Conveyor Control

A typical installation when using the option *Indexing Conveyor Control* includes the following components:



xx1200001075

A	Infeeder
B	Sensor
C	Motor unit

Continues on next page

1 Introduction to conveyor tracking

1.1 Physical components

Continued

D	Indexed conveyor
E	Robot controller
IRB	IRB robot

An extra drive unit and SMB needs to be installed, see *Application manual - Additional axes*.

1.2 Features

Accuracy

In automatic mode, at 150 mm/s constant conveyor speed, the tool center point (TCP) of the robot will stay within +/- 2 mm of the path. This distance is maintained whether the conveyor is in motion or not. This is valid as long as the robot is within its dynamic limits with the added conveyor motion. This figure depends on the calibration of the robot and conveyor and is applicable for linear conveyor tracking only.

Object queue

For the option *Conveyor Tracking*, a queue is maintained of up to 254 objects that have passed the synchronization switch. For the option *Indexing Conveyor Control*, the queue can contain up to 100 objects.

Start window

A program can choose to wait for an object that is within a window past the normal starting point, or wait for an object to pass a specific distance, or immediately take the first object in the object tracking queue. Objects that go beyond the start window are automatically skipped.

RAPID access to queue and conveyor data

A RAPID program has access to the number of objects in the object queue, and the current position and speed of the conveyor. A RAPID program can also remove the first object in the tracking queue or all objects in the queue.

Maximum distance

A maximum tracking distance can be specified to stop the robot from tracking outside of the working or safety area.

Track follows conveyor

If the robot is mounted on a linear track, then the system can be configured so that the track will follow the conveyor and maintain the relative position to the conveyor. The TCP speed relative the work object on the conveyor will still be the programmed speed.

Enter and exit conveyor tracking in corner zones

It is possible to enter and exit conveyor tracking via corner zones as well as via fine points. Use corner zones to achieve a minimum cycle time.

Exit and re-enter conveyor tracking to same object

It is possible to exit and re-enter to the same object on the conveyor unlimited times until the object moves outside the working area, reaches the maximum distance, or is explicitly dropped by the RAPID program.

Continues on next page

1 Introduction to conveyor tracking

1.2 Features

Continued

Multiple conveyors

Up to six conveyors are supported with the option *Conveyor Tracking*. Each encoder must be connected to an encoder interface.

Up to two indexed conveyors can be used with the option *Indexing Conveyor Control*.

Coordinated finepoint

A finepoint can be programmed while moving relative to the conveyor. This conveyor coordinated finepoint will ensure that the robot stops moving relative to the conveyor and will follow the conveyor while the RAPID program continues execution. The robot will hold the position within +/- 0.7 mm depending on calibration of the robot and conveyor.

Calibration of linear conveyors

A calibration method is provided for easy calibration of the position and direction of the conveyor motion in the robot work space. The linear conveyor may take any position and orientation.

1.3 Limitations

Small orientation error with SingArea\Wrist

There can be a small orientation error of the TCP while following the conveyor and make long motions with SingArea\Wrist. This error can be eliminated by using several short movements with SingArea\Wrist.

Robot mounted on track

If the robot is mounted on a track and the track is to be used to follow the conveyor motions, then the track and conveyor must be parallel.

The motion on the track and the motion on the conveyor must have the same direction of positive motion. See [Configuring a track motion to follow a conveyor on page 76](#).

Calibrating circular conveyors

There are no built-in methods for calibrating circular conveyors. This can be evaded by calculating a quaternion orientation manually or with other tools during base frame calibration.

Additional axes

For the option *Conveyor Tracking*, each conveyor is considered an additional axis. Thus the system limitation of 6 active additional axes must be reduced by the number of active and installed conveyors.

The first installed conveyor will use measurement node 6 and the second conveyor will use measurement node 5. These measurement nodes are not available for additional axes and no resolvers should be connected to these nodes on any additional axes measurement boards.

For the option *Indexing Conveyor Control*, each conveyor is considered as two additional axes.

Object queue lost on restart or power failure

If the system is restarted or if the power supply fails, then the object queue will be lost.

Minimum and maximum speed

The minimum conveyor speed to maintain smooth and accurate motions depends on the encoder selection. It can vary from 4 mm/s to 8 mm/s. See [Minimum and maximum counts per second on page 40](#).

There is no explicit maximum speed for the conveyor. Accuracy will degrade at speeds above the specification and with high speed robot motions or with very high conveyor speeds (> 500 mm/s) and the robot will no longer be able to follow the conveyor.

WaitWObj after DropWObj

If a WaitWObj instruction is used immediately after a DropWObj instruction, it may be necessary to add a WaitTime 0.1; after the DropWObj instruction.

1 Introduction to conveyor tracking

1.4 Principles of conveyor tracking

1.4 Principles of conveyor tracking

Description

In conveyor tracking, the robot Tool Center Point (TCP) will automatically follow a work object that is defined on the moving conveyor. While tracking the conveyor the robot controller will maintain the programmed TCP speed relative to the work object even if the conveyor runs at different speeds.

The options *Conveyor Tracking* and *Indexing Conveyor Control* are built on the coordinated work object, as with coordinated motion with additional axes. See *Application manual - Additional axes*.

Conveyor as a mechanical unit

In the option *Conveyor Tracking*, the conveyor is treated as a mechanical unit. It has all features of a mechanical unit except that it is not controlled by the robot controller. In the option *Indexing Conveyor Control*, the conveyor is treated as a mechanical unit and controlled by the robot controller.

As a mechanical unit, the conveyor can be activated and deactivated. The position of the conveyor is shown on the FlexPendant, and in the robtarget when a position is modified (ModPos).

Conveyor coordinated work object

The robot movements are coordinated to the movements of a user frame connected to the conveyor mechanical unit. For example a user frame is placed on the conveyor and connected to its movements.

A movable work object is used with the name of the conveyor mechanical unit. The conveyor tracking coordination will be active if the mechanical unit is active and the conveyor coordinated work object is active. When the conveyor coordinated work object is used, in jogging or in a move instruction, the data in the uframe component will be ignored and the location of the user coordinate system will only depend on the movements of the conveyor mechanical unit. However the oframe component will still work giving an object frame related to the user frame and also the displacement frame can be used.

Waiting for a work object on the conveyor

The difference between a conveyor coordinated work object and a work object that is coordinated to another type of mechanical unit is that there is no work object for coordination until an object appears on the conveyor. There must be a work object present on the conveyor before the robot can coordinate the TCP positions to the conveyor.

For the option *Conveyor Tracking*, work objects on the conveyor are detected by the synchronization switch that is connected to the encoder interface. The unit tracks all objects that have past the synchronization switch and are within specified distances in the work area.

For the option *Indexing Conveyor Control*, the work object is created when the defined number of objects (that is, number of indexes) have passed. Before starting coordinated motion with the conveyor, the RAPID program must first check if there

Continues on next page

is a work object available on the conveyor. If an object is available then execution continues and the motions can use the coordinated work object. If there is no object, then the RAPID program waits until a work object is available.

Connecting to a work object

The RAPID instruction `WaitWObj` is used to wait for a work object on the conveyor before starting conveyor coordinated motion. When the `WaitWObj` instruction is successful then the conveyor work object is said to be connected to the RAPID program. See [WaitWObj - Wait for work object on conveyor on page 99](#).

Once a RAPID program has connected to a work object on the conveyor then robot motion instructions and jogging commands can use this work object just as any other work object. When using the conveyor connected coordinated work object then all motions are relative to the work object on the conveyor.

The robot can switch between the conveyor coordinated work object and another coordinated work object in the system, but only one conveyor work object can be connected.

Disconnecting from a work object

The work object will remain connected until one of the following occurs:

- A `DropWObj` instruction is issued
- The maximum distance defined for the conveyor is reached
- Controller is restarted
- Power failure

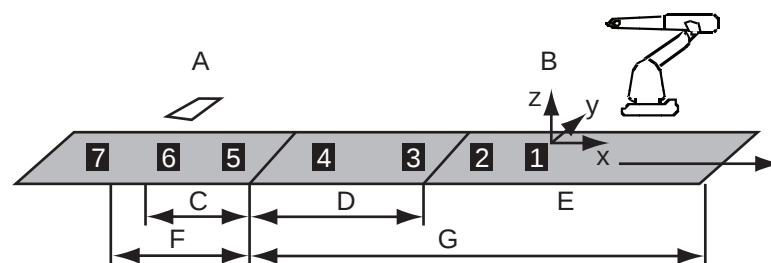
The connection to the work object will not be lost when deactivating the conveyor mechanical unit, and will return on re-activation.

The instruction `DropWObj` ends the connection before the maximum distance is reached.

After a `DropWObj` instruction is issued it is possible to immediately wait for and connect to the next work object in the conveyor object queue.

Start window and queue tracking distance

The object queue is based on a set of distances relative to the conveyor and synchronization switch. See the following figure.



xx1200001076

A	Synchronization switch
B	Work object user frame
C	Minimum distance (<i>minimum distance</i>)

Continues on next page

1 Introduction to conveyor tracking

1.4 Principles of conveyor tracking

Continued

D	Start window width (<i>StartWinWidth</i>)
E	Working area
F	Queue tracking distance (<i>QueueTrckDist</i>)
G	Maximum distance (<i>maximum distance</i>)
1-7	Objects on conveyor

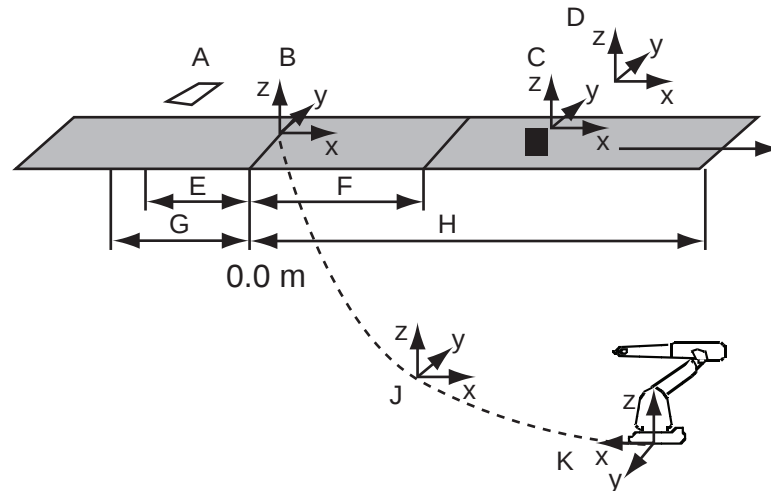
The conditions and states of objects 1 to 7 can be described as in the following table.

For descriptions of the system parameters *StartWinWidth* and *QueueTrckDist* see [Type Conveyor Ici \(DSQC2000\) on page 97](#). For descriptions of the system parameters minimum distance and and maximum distance see [Type Conveyor systems on page 96](#).

Objects	Description
1	This object is connected as indicated by the coordinate frame attached to the position of the object on the conveyor.
2	Object 2 is outside the start window and is no longer tracked. This object will be skipped and cannot be connected by a <code>WaitWObj</code> instruction. It is skipped because the conveyor speed is such that coordination with the object could not be completed before the object moves outside the maximum distance or work area of the robot.
3 and 4	These objects are within the start window and are tracked. If object 1 is and 4 dropped via a <code>DropWObj</code> instruction then object 3 is the next object to be connected when a <code>WaitWObj</code> instruction is issued. Because objects 3 and 4 were in the start window, the <code>WaitWObj</code> instruction will not wait but return immediately with object 3.
5 and 6	These objects lie within the queue tracking distance and are tracked. If objects and 6, 3, and 4 were not present, then a <code>WaitWObj</code> instruction would stop program execution until object 5 entered the start window.
7	This object has not yet passed the synchronization switch and has not yet been registered.

Continues on next page

Coordinate systems



xx1200001077

A	Synchronization switch
B	Base frame for conveyor
C	User frame
D	Object frame
E	Minimum distance
F	Start window width
G	Queue tracking distance
H	Maximum distance
J	World frame
K	Base frame for robot

The preceding figure shows the principle coordinate frames used in conveyor tracking. See the following table.

Coordinate system	Relative to
Base frame of robot	World frame
World frame	N.a.
Base frame of conveyor	World frame
User frame, coordinated to conveyor	World frame via base frame of conveyor
Object frame (not shown)	User frame

Key frames and positions

The two key frames in conveyor tracking are the base frame for the conveyor and the user frame in the conveyor coordinated work object. The position of user frame in the conveyor coordinated work object is determined from the position of the conveyor base frame and the linear position of the conveyor in meters.

The encoder interface provides the position of the conveyor relative the synchronization switch and the *Queue Tracking Distance*. When the conveyor position is 0.0 meters, the user frame for the conveyor coordinated work object

Continues on next page

1 Introduction to conveyor tracking

1.4 Principles of conveyor tracking

Continued

coincides with the base frame of the conveyor. As the conveyor moves, then the user frame in the conveyor coordinated work object moves along the x-axis of the conveyor base frame.

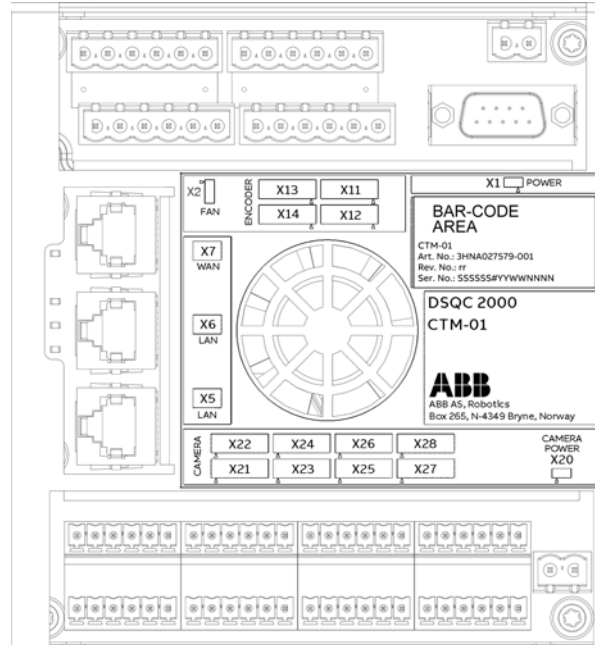
See *Operating manual - OmniCore* and *Technical reference manual - RAPID Overview*.

2 Conveyor tracking modules

2.1 Network based conveyor tracking module

Description

The CTM is a network based conveyor interface that provides connections for 4 encoders and 8 cameras.



xx1800000346

The CTM uses network communication to share conveyor speed and position data with one or more robot controllers. It contains a WAN port, which is used to connect to the robot controllers and two LAN ports that can be used for installation and service purposes. It can be located either inside the robot controller or inside a separate cabinet. To improve the quality of the signals keep the CTM close to the encoders and the sensors.

Each of the encoder inputs support one 2 phase encoder. Each of the camera connections consist of a digital sync input, a 24 V digital trigger output, and a camera power output. The camera connection can also be used for other sync input sources, for example photocells.

The installation is described in section [Installation on page 39](#).

Functions

Ethernet connections

The CTM includes WAN and LAN Ethernet ports. The WAN port of the CTM must be connected to the same Ethernet network as the robot controllers. The robot controllers should be connected to the same network with any of their public network ports (see *Operating manual - Integrator's guide OmniCore*). If the CTM is only used by one robot controller, for example an internally mounted CTM, the

Continues on next page

2 Conveyor tracking modules

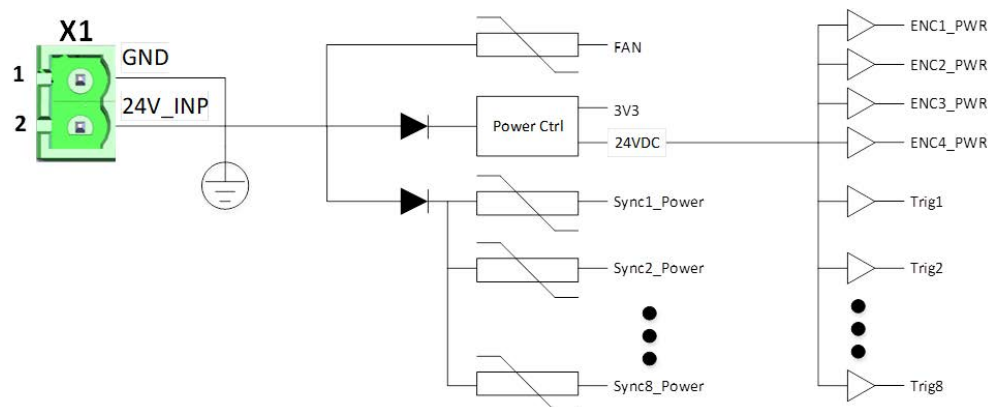
2.1 Network based conveyor tracking module

Continued

WAN port of the CTM should be connected to any of the robot controllers public network or private network ports (see *Operating manual - Integrator's guide OmniCore*).

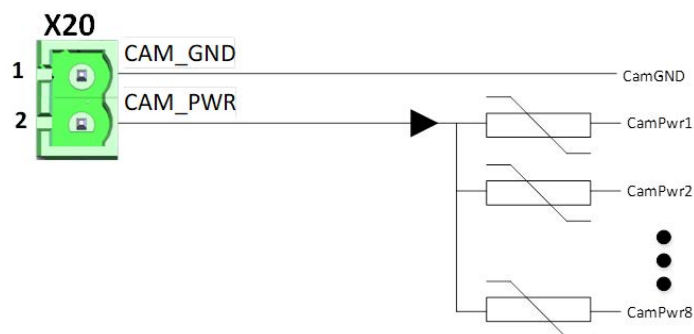
Power distributions

The CTM has two power inlets, power inlet (X1) and an optional camera power inlet (X20). The power inlet is 24 V, it supplies power to all main function on the CTM module, as shown in the following figure:



xx1800000347

The Camera power (X20) and its ground connection are separated from the rest of the CTM power. The camera power can only be routed from the power inlet through the CTM's current protection circuits to the individual outputs in the camera connectors (X21-X28), as shown in the following figure:



xx1800000348

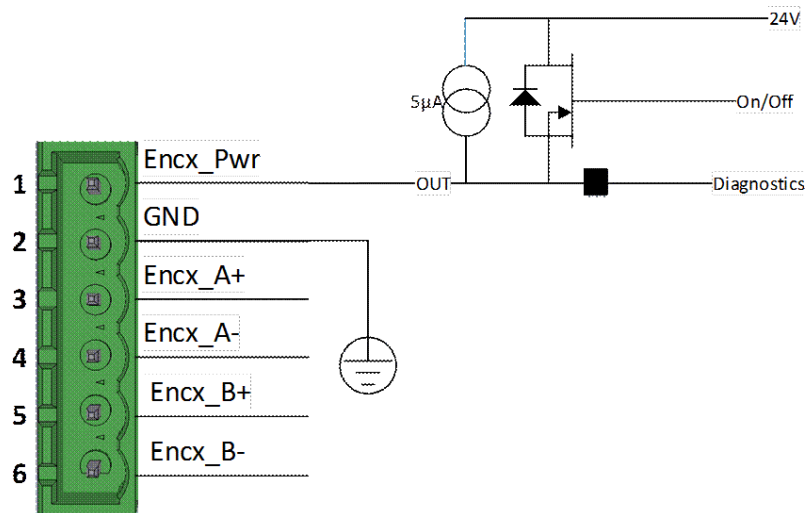
The 24V_INP, 24VDC, CAM_PWR, and 3V3 are covered by diagnostics functions and their levels are displayed in RobotStudio.

Encoder power outputs

Each encoder connection (X11 - X14) supports an encoder power output. The encoder power output is 24 V with diagnostics functions, which is controlled by a high side driver circuit. In case of an overload or short circuit on the CTM, it is turned off. This output also contains a Discovery function, which can detect if an encoder is connected to the power. A constant test current of approximately 5 uA checks the output of the Discovery function, if the output is off it is activated. This test current will cause an unconnected trigger output pin to float at 10 V, even

Continues on next page

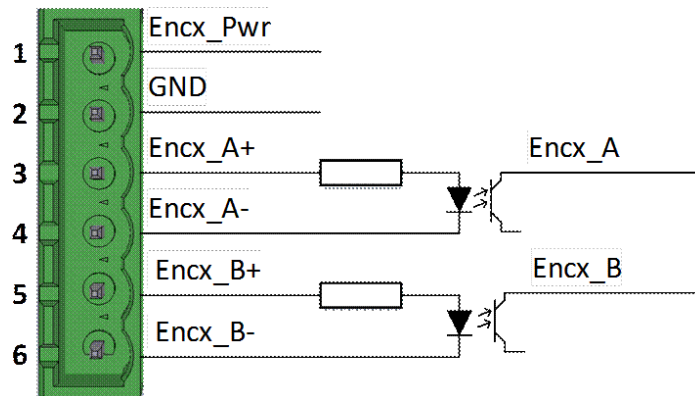
when the output signal is turned off. This power output is always on, except during startup, as shown in the following figure:



xx1800000349

Encoder inputs

Each encoder connection (X11-X14) supports a two phase encoder. The inputs are isolated and open ended to support NPN, PNP, and Push-Pull type encoders. The encoder can be powered from the CTM or by an external power supply, as shown in the following figure:



xx1800000350

Sync inputs

Each camera connector (X21 - X28) has a sync input which is isolated through opto couplers. These couplers are single ended, share a common GND with the CTM board, and support PNP and push-pull sensors. It is recommended to use

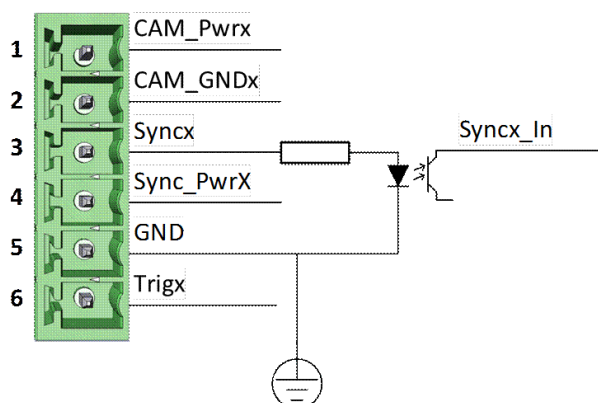
Continues on next page

2 Conveyor tracking modules

2.1 Network based conveyor tracking module

Continued

the sync power output to power the sync sensor signal, as shown in the following figure:



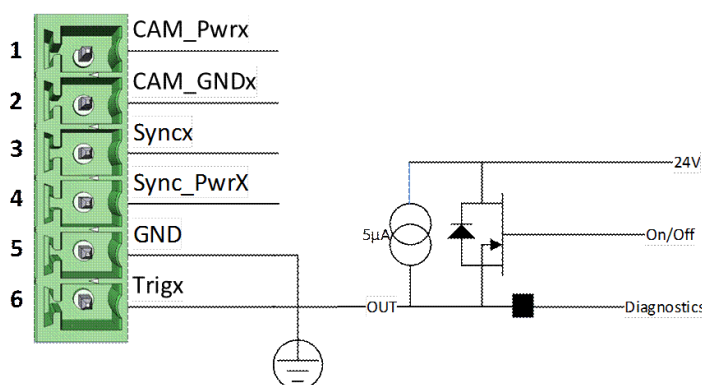
xx1800000351

The minimum time that the sync input must stay high on the CTM is 10 μ s. Additionally the conveyor position must have changed since last sync signal (only 1 sync signal can be signaled in each unique encoder-position).

Trigger outputs

Each camera connector (X21 - X28) has a trigger output. This output is a 24 V digital output with diagnostics functions, which is controlled by a high side driver circuit. In case of an overload or short circuit on the CTM, it is turned off. This output also contains a Discovery function, which can detect if an encoder is connected to the power. A constant test current of approximately 5 μ A checks the output of the Discovery function, if the output is off it is activated. This test current will cause an unconnected trigger output pin to float at 10 V, even when the output signal is turned off.

The following figure shows a drawing of the trigger output.



xx1800000352

Technical specification DSQC2000, CTM

General	
Power supply	24 VDC (-15/+20%), typically 200 mA (Current not including power outputs)
Operating temperature	+5 °C - +65 °C

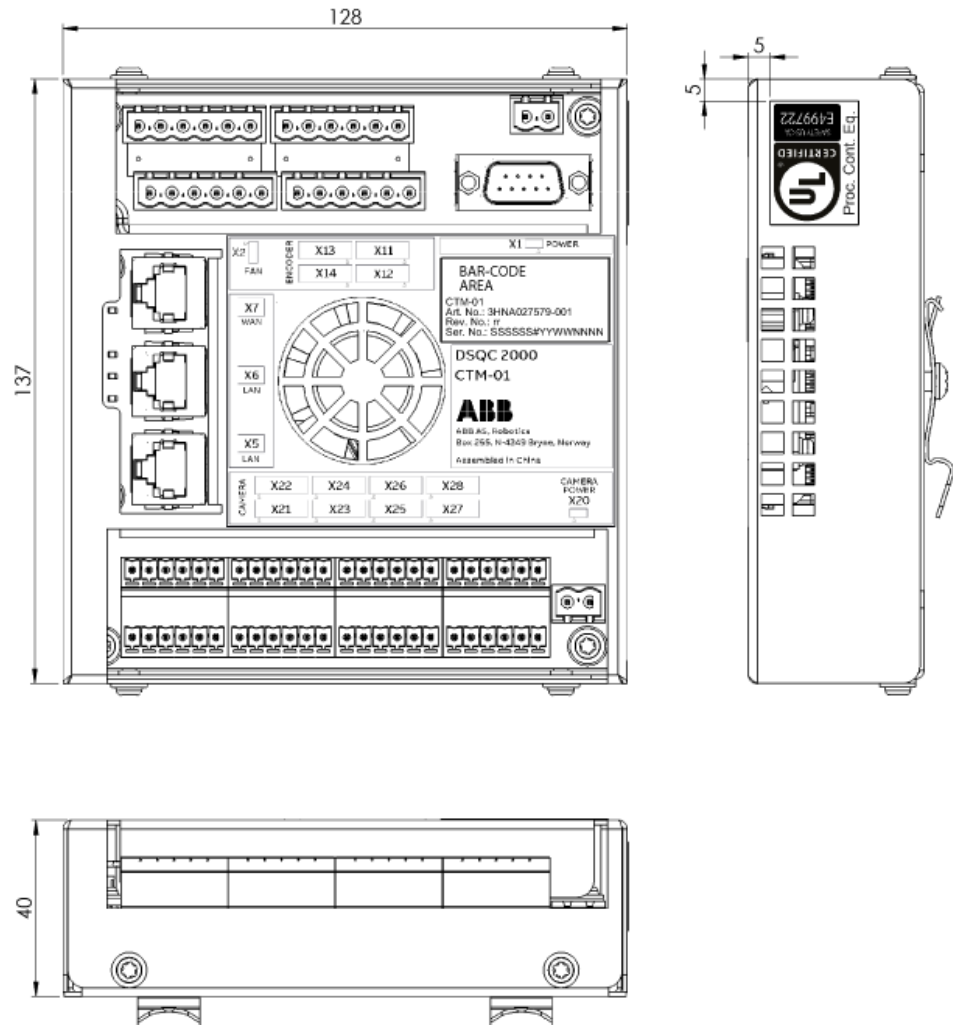
Continues on next page

2 Conveyor tracking modules

2.1 Network based conveyor tracking module

Continued

General	
Ethernet LAN	2 switched LAN ports, 100 Mbit (Only for installation and service purposes)
Ethernet WAN	1 WAN port, 100 Mbit
Dimensions	137 x 128 x 40 mm



xx2000002184

Encoders	4 pcs (X11-X14)
Power output	24 VDC, max 120 mA With connection discovery and overload protection/diagnostic
Frequency	0 - 20 kHz
Input current	5.2 mA at 24 VDC
Voltage levels	15 VDC < '1' < 30 VDC, -3 VDC < '0' < 5 VDC
Supported encoder types	PNP, NPN Push-Pull (HTL)

Continues on next page

2 Conveyor tracking modules

2.1 Network based conveyor tracking module

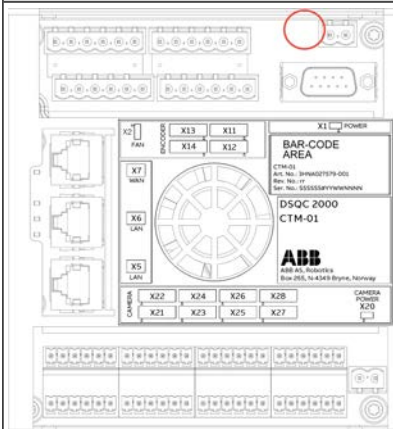
Continued

Cameras	8 pcs (X21-X28)
Camera power output	Supplied from X20 camera power inlet (normally 24 VDC, maximum 250 mA) With overload protection
Sync input signal	
Power output	24 VDC, 120 mA With connection discovery and overload protection/diagnostic
Input Current	5.2 mA at 24 VDC
Voltage levels	15 VDC < '1' < 30 VDC, -3 VDC < '0' < 5 VDC
Trigger output	
Digital output	24 VDC, Max 120 mA With connection discovery and overload protection/diagnostic
Minimum load	0.1 mA (Floating pins will drift towards voltage rails)

LED indicators

The CTM module has the following LED diagnostics and information:

- Module Status LED – BiColor LED displays the module status.

Status LED	Description
 xx1800000353	During startup Red (steady): Default at power up
	OS loading Red (blinking): Startup completed OK
	Run-time Green (steady): Up and running, OS loading completed OK

- Module Discovery LED – Indicates the connection status of the CTM module with a RobotStudio client.

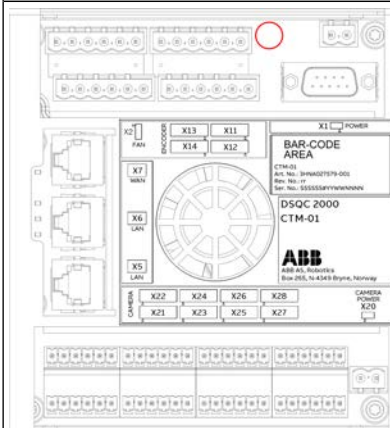
The discovery LED can be only green, or bicolor green + red.

Continues on next page

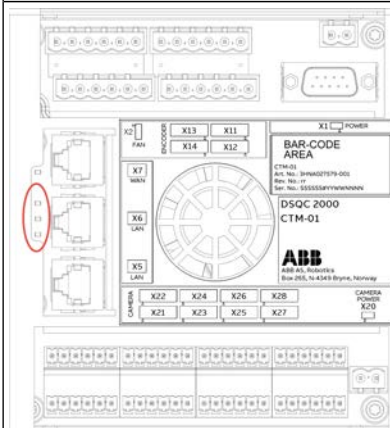
2 Conveyor tracking modules

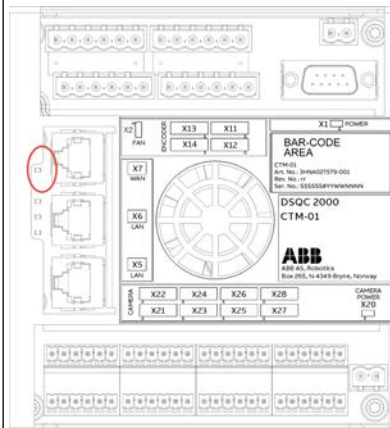
2.1 Network based conveyor tracking module

Continued

Discovery LED	Description
 xx1800001604	During startup LED off
	OS loading LED off
	Run-time Green + green (blinking), or, green + red (blinking): Connected and selected by a RobotStudio client.

- On the CPU board, there are three Ethernet connection status LEDs and a LED displaying the power status.

Ethernet Link Status	Description
 xx1800000354	LED 1 - Ethernet LAN 1 LED - Green LED: Ethernet LAN 1, 100 Mbit link enabled - Ethernet port 1, Connector X5
	LED 2 - Ethernet LAN 2 LED - Green LED: Ethernet LAN 2, 100 Mbit link enabled - Ethernet port 2, Connector X6
	LED 3 - Ethernet WAN port LED - Green LED: Ethernet WAN, 100M bit link enabled - Ethernet port 3, Connector X7

CPU board power status	Description
 xx1800000355	LED 4 - Power Status Green LED: 3.3 VDC power is OK

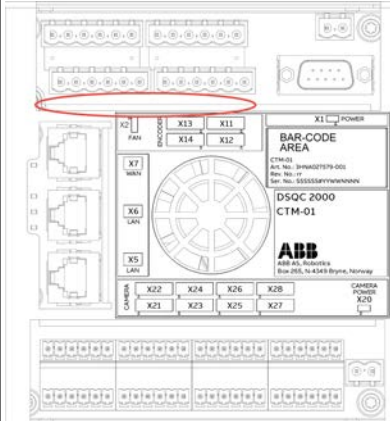
- Each encoder input has two status LEDs, one for each input pair.

Continues on next page

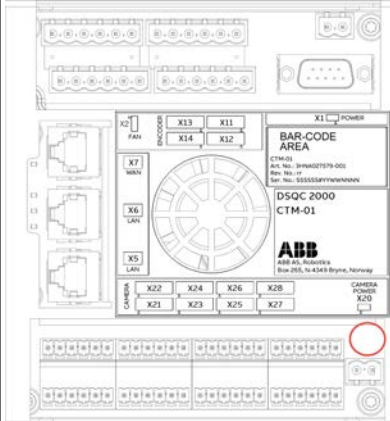
2 Conveyor tracking modules

2.1 Network based conveyor tracking module

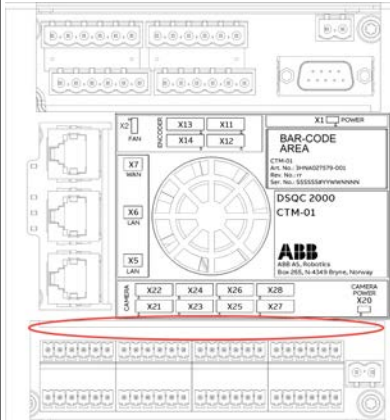
Continued

CPU board power status	Description
 xx1800000356	LED_ENC (1 - 4) A & B LEDs For each encoder connection (X11 - X14) there is a LED displaying the input level of each encoder pair.

- A LED displaying the status of the camera power inlet.

Camera power status LED	Description
 xx1800000357	LED_CAM_PWR Displays the power status of the optional camera power inlet.

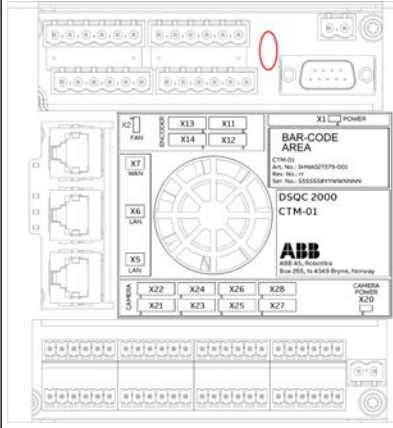
- Each camera connection has two status LEDs, one for the sync signal input and one for the trigger signal output.

Camera status LED	Description
 xx1800000358	LED_SYNC (1 - 8) LEDs For each camera connection (X21 - X28) there is a LED displaying the input stage of the sync input signal. LED_Trig (1 - 8) LEDs For each camera connection (X21 - X28) there is a LED displaying the output stage of the trigger output signal.

Continues on next page

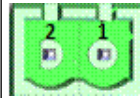
Buttons

The CTM module has the following two user interactive buttons:

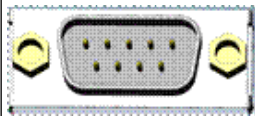
Buttons	Description
 xx1800001605	White SW1 button - Short press: Devices connected to a RobotStudio client is highlighted. - Press and hold while the CTM is re-booting: CTM resets to default factory settings.  WARNING Resetting the CTM to default factory settings will reinstall the current version of the firmware and reset all the passwords to default. It will reset configured updates, for example, Sync Separation settings for encoders will be reset to 0. Configured network settings will not be affected.
	Black SW2 button - Short press: CTM is restarted (warm start). - Long press: CTM is rebooted.

Connectors

X1 – CTM power inlet connector

X1	Pin number	Name	Function
 xx1800000359	1	GND	CTM power inlet
	2	24V_INP	

X3 – Console port for debugging

X3	Pin number	Name	Function
 xx1800000360	1	(Not connected)	
	2	SCI_RX	Console receive
	3	SCI_TX	Console transmit
	4	(Not connected)	
	5	GND	
	6	(Not connected)	
	7	(Not connected)	
	8	(Not connected)	
	9	(Not connected)	

Continues on next page

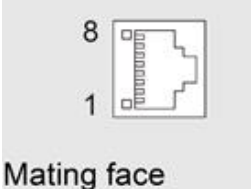
2 Conveyor tracking modules

2.1 Network based conveyor tracking module

Continued

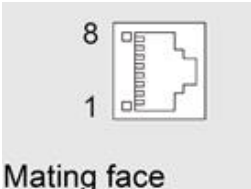
X5-X6 – Ethernet connections

X5 – LAN port 1 connection

X5	Pin number	Name	Function
 Mating face xx1800000361	1	TX1+	Transmit pair
	2	TX1-	Transmit pair
	3	RX1+	Receive pair
	4	i	
	5	i	
	6	RX1-	Receive pair
	7	i	
	8	i	

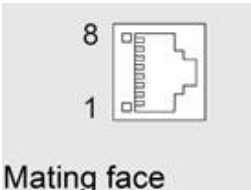
i Connected to a resistor network (provides a common mode termination)

X6 – LAN port 2 connection

X6	Pin number	Name	Function
 Mating face xx1800000361	1	TX2+	Transmit pair
	2	TX2-	Transmit pair
	3	RX2+	Receive pair
	4	i	
	5	i	
	6	RX2-	Receive pair
	7	i	
	8	i	

i Connected to a resistor network (provides a common mode termination)

X7 – WAN port connection

X7	Pin number	Name	Function
 Mating face xx1800000361	1	TX3+	Transmit pair
	2	TX3-	Transmit pair
	3	RX3+	Receive pair
	4	i	
	5	i	
	6	RX3-	Receive pair
	7	i	
	8	i	

i Connected to a resistor network (provides a common mode termination)

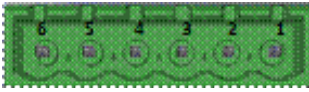
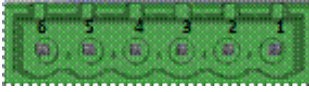
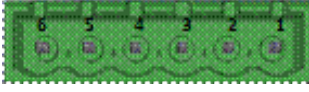
Continues on next page

2 Conveyor tracking modules

2.1 Network based conveyor tracking module

Continued

X11-X14 – Encoder connections

X11	Pin number	Name	Function
 xx1800000362	1	ENC1PWR	Encoder 1 power output
	2	GND	
	3	ENC1A+	Encoder input 1, Pair A
	4	ENC1A-	
	5	ENC1B+	Encoder input 1, Pair B
	6	ENC1B-	
X12	Pin number	Name	Function
 xx1800000362	1	ENC2PWR	Encoder 2 power output
	2	GND	
	3	ENC2A+	Encoder input 2, Pair A
	4	ENC2A-	
	5	ENC2B+	Encoder input 2, Pair B
	6	ENC2B-	
X13	Pin number	Name	Function
 xx1800000362	1	ENC3PWR	Encoder 3 power output
	2	GND	
	3	ENC3A+	Encoder input 3, Pair A
	4	ENC3A-	
	5	ENC3B+	Encoder input 3, Pair B
	6	ENC3B-	
X14	Pin number	Name	Function
 xx1800000362	1	ENC4PWR	Encoder 4 power output
	2	GND	
	3	ENC4A+	Encoder input 4, Pair A
	4	ENC4A-	
	5	ENC4B+	Encoder input 4, Pair B
	6	ENC4B-	

X20 – Camera power inlet

X20	Pin number	Name	Function
 xx1800000363	1	CAM_GND	Camera power inlet
	2	CAM_PWR	

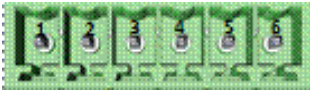
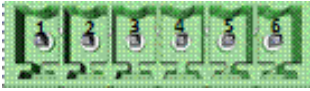
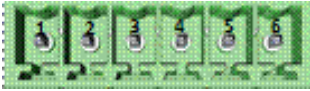
Continues on next page

2 Conveyor tracking modules

2.1 Network based conveyor tracking module

Continued

X21-X28 – Camera connections

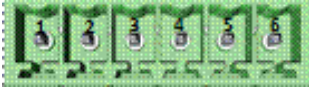
X21	Pin number	Name	Function
 xx1800000364	1	CAMPWR1	Camera 1 power output
	2	CAMPWRGND	
	3	SYNC1	Sync 1 input signal
	4	SYNCPWR1	Power output for sync signal
	5	GND	GND for trigger and sync power
	6	TRIG1	Trigger 1 output
X22	Pin number	Name	Function
 xx1800000364	1	CAMPWR2	Camera 2 power output
	2	CAMPWRGND	
	3	SYNC2	Sync 2 input signal
	4	SYNCPWR2	Power output for sync signal
	5	GND	GND for trigger and sync power
	6	TRIG2	Trigger 2 output
X23	Pin number	Name	Function
 xx1800000364	1	CAMPWR3	Camera 3 power output
	2	CAMPWRGND	
	3	SYNC3	Sync 3 input signal
	4	SYNCPWR3	Power output for sync signal
	5	GND	GND for trigger and sync power
	6	TRIG3	Trigger 3 output
X24	Pin number	Name	Function
 xx1800000364	1	CAMPWR4	Camera 4 power output
	2	CAMPWRGND	
	3	SYNC4	Sync 4 input signal
	4	SYNCPWR4	Power output for sync signal
	5	GND	GND for trigger and sync power
	6	TRIG4	Trigger 4 output

Continues on next page

2 Conveyor tracking modules

2.1 Network based conveyor tracking module

Continued

X25	Pin number	Name	Function
 xx1800000364	1	CAMPWR5	Camera 5 power output
	2	CAMPWRGND	
	3	SYNC5	Sync 5 input signal
	4	SYNCPWR5	Power output for sync signal
	5	GND	GND for trigger and sync power
	6	TRIG5	Trigger 5 output
X26	Pin number	Name	Function
 xx1800000364	1	CAMPWR6	Camera 6 power output
	2	CAMPWRGND	
	3	SYNC6	Sync 6 input signal
	4	SYNCPWR6	Power output for sync signal
	5	GND	GND for trigger and sync power
	6	TRIG6	Trigger 6 output
X27	Pin number	Name	Function
 xx1800000364	1	CAMPWR7	Camera 7 power output
	2	CAMPWRGND	
	3	SYNC7	Sync 7 input signal
	4	SYNCPWR7	Power output for sync signal
	5	GND	GND for trigger and sync power
	6	TRIG7	Trigger 7 output
X28	Pin number	Name	Function
 xx1800000364	1	CAMPWR8	Camera 8 power output
	2	CAMPWRGND	
	3	SYNC8	Sync 8 input signal
	4	SYNCPWR8	Power output for sync signal
	5	GND	GND for trigger and sync power
	6	TRIG8	Trigger 8 output

This page is intentionally left blank

3 Installation

3.1 Hardware connections

Introduction to hardware installation

For the option *Conveyor Tracking* or PickMaster 3, the conveyor is connected to the robot controller using an encoder and the conveyor tracking module (CTM-01, DSQC2000). See [Installing the encoder for conveyor tracking on page 40](#).

For the option *Indexing Conveyor Control*, the conveyor is controlled by a motor unit (ABB or other). See [Installing the additional axis for servo control on page 145](#).

3 Installation

3.2 Installing the encoder for conveyor tracking

3.2 Installing the encoder for conveyor tracking

Selecting encoder type

The encoder provides a series of pulses indicating the motion of the conveyor. This is used to synchronize the motions of the robot to the motion of the conveyor. The encoder has two pulse channels, A and B, that differ in phase by 90°. Each channel will send a fixed number of pulses per revolution depending on the construction of the encoder.

The number of pulses per revolution for the encoder must be selected in relation to the gearing between the conveyor and the encoder. The pulse ratio from the encoder should be in the range of 1250-2500 pulses per meter of conveyor motion. The pulses from channels A and B are used in quadrature to multiply the pulse ratio by 4 to get counts. This means that the control software will measure 5000 to 10000 counts per meter for an encoder with the pulse ratio given above. Reducing the number of measured counts below 5000 will reduce the accuracy of the robot tracking. Increasing the number of measured counts above 10000 will have no significant effect as inaccuracies in robot and cell calibration will be the dominating factors for accuracy.

The encoder must be of 2 phase type for quadrature pulses, to register reverse conveyor motion, and to avoid false counts due to vibration etc. when the conveyor is not moving.

Output signal	Typical spec, Open collector PNP output encoder
Voltage	10-30 V (normally supplied by 24 VDC from DSQC2000, CTM-01)
Current	50-100 mA
Phase	2 phase with 90 degree phase shift
Duty cycle	50%

An example encoder is the *Lenord & Bauer GEL 262*.

See also the encoder information in section [Network based conveyor tracking module on page 25](#).

Minimum and maximum counts per second

Minimum speed	The lower limit on the number of counts per second before the encoder signals zero speed is 40 counts per second. If the speed of the conveyor is lower, zero speed will be indicated. At 10,000 counts per meter, the minimum conveyor speed is 4 mm/s.
Maximum speed	The upper limit on the number of counts per second before the encoder can no longer keep track of the counts along the conveyor is 20,000 counts per second. At 10,000 counts per meter, the maximum conveyor speed is 2,000 mm/s.

Continues on next page

3.2.1 Encoder location

Description

The encoder is normally installed on the conveyor drive unit. The encoder can be connected to an output shaft on the drive unit, directly or via a gear belt arrangement.

Prerequisites

If the encoder is connected directly to a drive unit shaft, a flexible coupling must be used to prevent applying mechanical forces to the encoder rotor. Do not use a coupling using a plastic or rubber hose. If the drive unit includes a clutch arrangement, the encoder must be connected on the conveyor side of the clutch.

If the conveyor drive unit is located a long distance away from the robot then the conveyor itself can be a source of inaccuracy as the conveyor will stretch or flex over the distance from the drive unit to the robot cell. In such a case it may be better to mount the encoder closer to the robot cell with a different coupling arrangement.

3 Installation

3.2.2 Encoder cables

3.2.2 Encoder cables

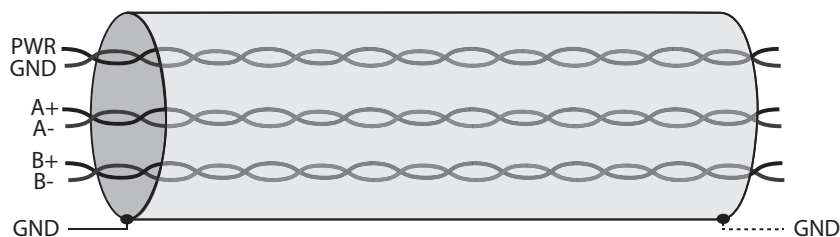
Description

The encoder should be connected to the robot by a screened cable to reduce noise. The noise immunity will be better with screened twisted pair cable. Cable length, pulse frequency and environmental conditions must be taken in account, case by case.

The cable screen can be grounded in one of the following ways:

- Grounded in the controller end, open in the encoder end.
- Grounded in both ends.
- High frequency connection with a capacitor between screen and ground in the encoder end.

The method for screen connection depends on the environment and can be different from case to case. Parameter like cable length, encoder frequency and other equipment with power electronics (motor-drive) and so on influence on the electromagnetic environment.



xx1800001539

PWR	Power
GND	Ground
A+, A-	Encoder signal A
B+, B-	Encoder signal B

3.3 Connecting the encoder to the CTM

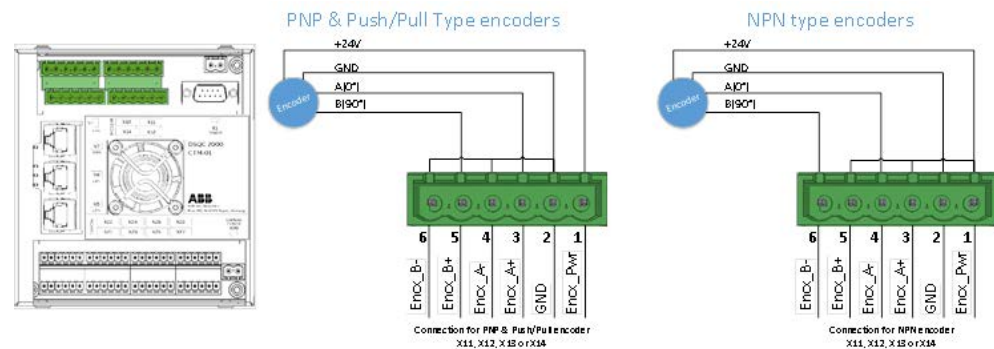
Description

An encoder is connected to a single encoder interface in the CTM. If multiple robots are following the same conveyor, its speed and position are supplied by the CTM to all the robot controllers through a network communication.

The conveyor tracking module CTM (DSQC2000) can be located either inside the robot controller or a cabinet close to the conveyor encoder (to reduce the cable length between the CTM and the encoder).

Encoder connections

The CTM has four independent encoder connections supporting PNP, Push-Pull, and NPN encoders.



xx1800000365

The configurations support the Connection Discovery functions and overload protection/diagnostic of the encoder power output.

If the encoder is externally powered, then the CTM power pins must not be connected. This configuration removes the connection discovery and overload protection and diagnostic functions of the encoder power output.

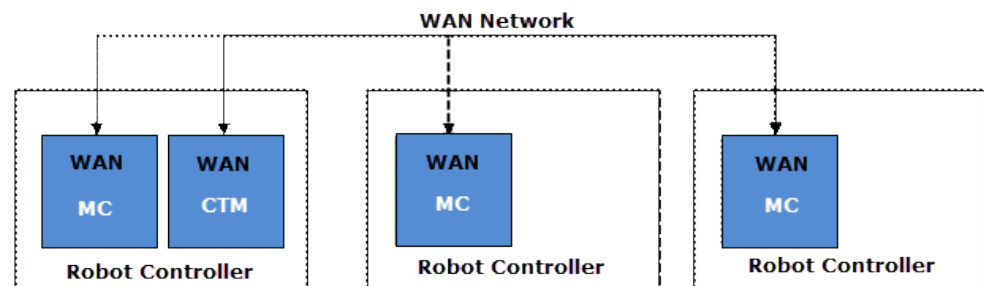
Continues on next page

**Note**

When connecting the CTM to the WAN port on the robot controller, you first have to enable the firewall for CTM by setting **Enable on Public Network** to YES for the Network Service RobICI. See *Operating manual - Integrator's guide OmniCore*, section "Firewall settings", for more information.

Mounted in robot controller, sharing conveyor info with other controllers

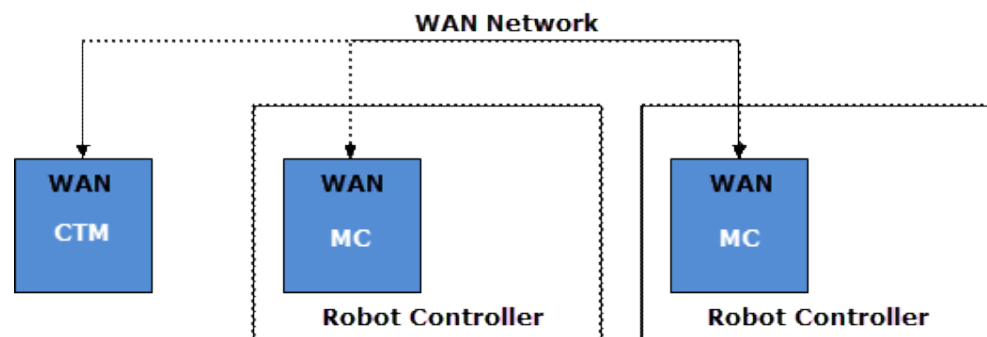
When the CTM module is connected to the same network as the robot controllers, the conveyor information may be shared from the CTM module to all the robot controllers.



xx1800000368

Mounted outside robot controller, sharing conveyor info with other controllers

The CTM module can be mounted externally, that is outside the robot controller.



xx1800000369

Continues on next page

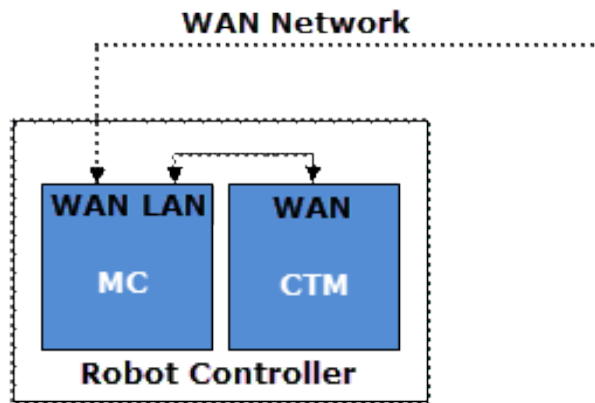
3 Installation

3.3 Connecting the encoder to the CTM

Continued

Mounted inside robot controller, no sharing of conveyor info

If the conveyor is not shared with any other robot controller, the CTM may be installed inside the controller using the Private network. This will reduce the number of external WAN ports required on the robot controller.



xx2100002143



Note

The WAN port is used on the CTM and LAN 2 is used on the main computer.

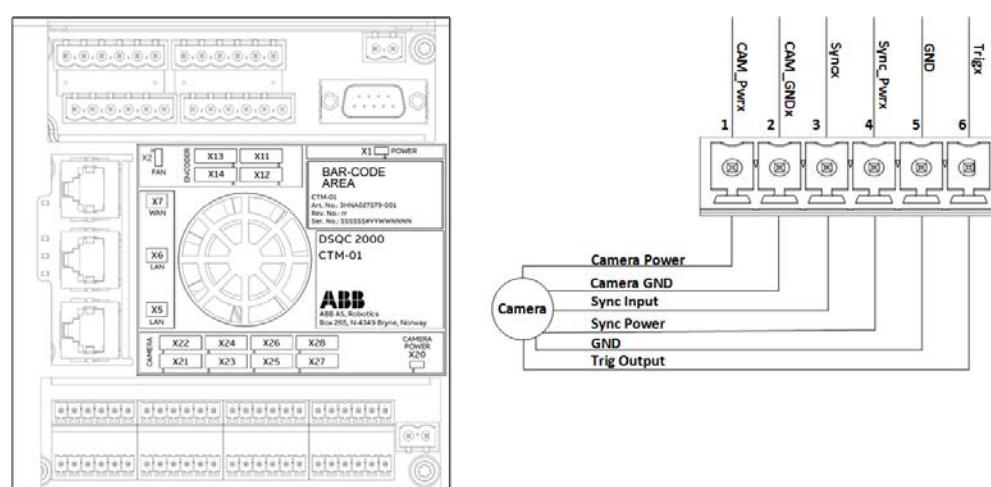
3.4 Connecting a camera to the CTM

Description

The CTM contains 8 camera connections to connect cameras or sync signals. The cameras can be powered separately or through the CTM.

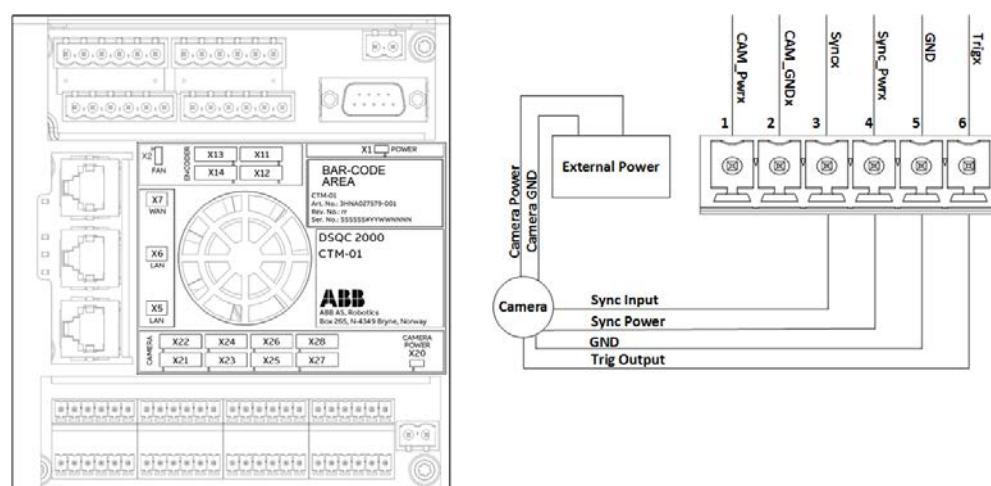
The sync signal input supports PNP and Push-Pull type sensors and can be powered by the Sync_Pwr output or externally. The trigger output is a 24 V digital output signal.

The configuration shown in the following figure, supports overload protection on CameraPower, Sync_Power, and Trig_output. The Connection Discovery function and the overload protection/diagnostics exist on Sync_Power and Trig_output.



xx1800000371

The configuration shown in the following figure, does not supports overload protection on CameraPower. The Connection Discovery function and the overload protection/diagnostics exist on Sync_Power and Trig_output.



xx1800000372

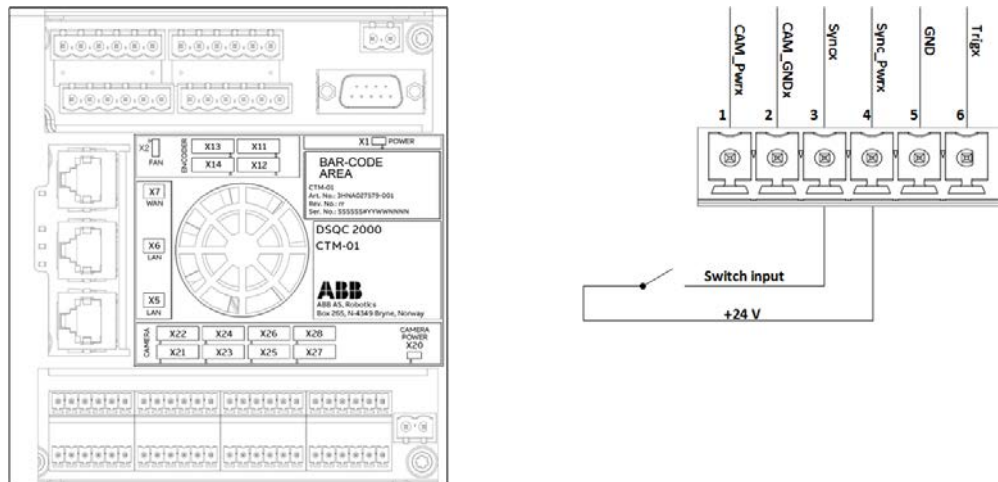
3 Installation

3.5 Connecting the sync switches to the CTM

3.5 Connecting the sync switches to the CTM

Description

The sync input signal in each of the camera connectors can also be used as a sync switch. It can be powered by Sync_Pwr output or externally and supports PNP and Push-Pull type sensors, as shown in the following figure:



xx1800000375

4 Conveyor Tracking tab in RobotStudio

Introduction

The Conveyor Tracking tab contains a tree view browser displaying the connected DSQC2000 units and other related objects. It is used as a user interface to configure settings and monitor live signals of the DSQC2000 tracking units.



Note

While in the tree view browser or any of the related object dialogues, pressing F1 will open the *Application manual - Conveyor tracking*.

To open the Conveyor Tracking tab:

	Action
1	Click Conveyor Tracking from the Configuration group, in the Controller tab. The Conveyor Tracking tab opens.



Note

For detailed information about firewall settings for RobotStudio, see section "System requirements\Firewall settings" in *Operating manual - RobotStudio*.

Connecting and adding a CTM

Connecting a CTM

One of the following methods can be used to connect the CTM to the computer:

1 Connecting the CTM through a WAN port

This method is recommended only after a fixed IP address is configured for the CTM - WAN interface, see [Network settings on page 50](#).

Connect the computer with a fixed IP address to the same WAN network, through a network switch or a direct connection. Configure the computer's IP address with the CTM's subnet.

Example:

CTM - WAN interface	192.168.8.xx
Computer network interface	192.168.8.246

The default factory setting for WAN is **Obtain an IP address automatically** (DHCP client). To connect a WAN with this setting a DHCP server is required on the network.

2 Connecting through a LAN port

This method is recommended for the initial setup of a CTM with default factory settings. It can also be used for debugging, when there is a connection problem with the CTM.

Configure the fixed IP address of the computer with the subnet of the CTM – LAN interface. The LAN interface has IP address 192.168.126.200.

Continues on next page

4 Conveyor Tracking tab in RobotStudio

Continued

Example:

CTM - LAN interface	192.168.126.200
Computer network interface	192.168.126.246

Adding a CTM

To add a CTM, a connection must be established between the computer and the CTM:

	Action
1	Click Add CTM . The Add CTM dialog box displays all the detected CTMs.
2	Select the CTM from the list and click OK . The CTM is added to the tree view object browser.

Identifying a CTM in RobotStudio

If RobotStudio is connected to a network with multiple CTMs, it may become difficult to identify a particular CTM. This can be avoided by a direct connection via LAN port, for more information, see [Connecting a CTM on page 49](#), section *Connecting through a LAN port*.

To simplify identification, it is recommended to assign a unique and descriptive name for every CTM, for more information, see [Rename on page 52](#).

One of the following methods can also be used to identify a CTM:

- Select the device in the tree view browser, its corresponding green discovery LED will start blinking on the CTM module.
- Press the white SW1 button on the CTM module once, a green dot will start blinking on the corresponding CTM symbol in the tree view browser.



WARNING

Do not press the SW1 button while the CTM is restarting, this will reset the CTM to default factory settings and any updates or upgrades will be lost.

Network settings

The WAN port of the CTM is connected to the robot controllers through an Ethernet network. To establish communication between the robot controllers and the CTM, a fixed IP address must be assigned to the CTM. The IP address should be located on the same subnet as the network interface of the connected robot controllers.

To change the network settings:

	Action
1	Right-click the CTM in the tree view browser, select Network settings , and then click WAN interface .
2	Enter the IP address , the Subnet mask , and the Default gateway (optional) and then click OK .

Continues on next page

Authenticate

A CTM has the following two predefined users:

User	Default password	Description
abbadmin	abbadmin	It is an advanced user and is used for advanced maintenance and troubleshooting.
ctmuser	ctmuser	It is a normal user and is used for everyday maintenance and troubleshooting.

**Note**

Login as an advanced user to upgrade the firmware.

To use some features of a CTM, a login is required. If you are using the default credentials, login is automatic to the Conveyor Tracking tab. Otherwise, authentication is requested whenever required.

**Note**

It is recommended to change the password to improve security.

Appropriate credentials are required to make any modifications to the CTM configuration and firmware.

**WARNING**

You cannot recover a lost password, so ensure the password is not lost or forgotten.

To login as a user:

	Action
1	Right-click the CTM and click Authenticate > Login as a Different User .
2	Enter the required credentials and click Login .

To log off the current user:

	Action
1	Right-click the CTM and click Authenticate > Log Off .

To change the password of the current user:

	Action
1	Right-click the CTM and click Authenticate > Change password .
2	Change the password and click OK .

Continues on next page

Rename

You can rename the CTM, the encoders, and the sensors.

To rename any object:

	Action
1	Right-click the object in the tree view browser and then click Rename . The Rename dialog box opens.
2	Enter a new name for the object and then click OK . The new names are stored in the CTM and will be displayed in RobotStudio, once it is restarted.

Configuring an encoder

You can configure some settings of an encoder, for example, Speed filter.

To configure an encoder:

	Action
1	Right-click the encoder in the tree view browser and then click Configure encoder . The Configure encoder tab opens.
2	Modify the settings and click Apply .

An encoder can be configured as used or not used. This change in configuration will affect how the encoder is displayed in the tree view browser. To tune the encoder speed filtering, you can define the **Low-pass filter cut off frequency** in the **Speed filter** section.

For DSQC2000: The low-pass filter is to smoothen the speed-calculation generated by CTM based on the encoder-frequency. By default, it is configured as a 2nd order IIR-filter with a cut-off frequency of 10 Hz, so signal-noise is reduced for disturbances < ~100 ms.

Configuring a sensor

You can configure some settings of a sensor, for example, Sync Separation.

To configure a sensor:

	Action
1	Right-click the sensor in the tree view browser and then click Configure sensor . The Configure sensor dialog box opens.
2	Modify the settings and click Apply .

Sync separation is used to filter an unstable signal in the sync input signal. It defines the minimum encoder distance (counts) between two sync pulses from the sensor. When the actual distance is shorter than this value, then the second sync pulse is ignored.

The sensor type can be configured as **I/O Sensor**, **Camera**, or **Not used**. This change in configuration will affect how the sensor is displayed in the tree view browser.



Note

It is recommended to set the same sync separation value for all the encoders.

Continues on next page

The camera pulse width defines the pulse length (ms) used with the camera trigger signal.

**Note**

For high speed conveyors, the pulse length must be shorter than the time between consecutive camera images, so that:

Camera pulse width in ms < 1000 * (Trigger distance in mm / Max conveyor speed in mm/s)

Restart**Overview**

A restart or reboot of a CTM can be performed using the tree view browser.

Restart

A restart is required after modifying the settings of the CTM.

To restart the CTM:

	Action
1	Right-click the desired CTM in the tree view browser and then click Restart .

Reboot

A reboot is slower than a restart, it corresponds to the restart after cycling the power.

To reboot the CTM:

	Action
1	Right-click the desired CTM in the tree view browser and then click Reboot .

Signals

There are 3 types of live signals: encoder, sensor, and other. You can sort and filter the signals by various properties:

Signal property	Description
Name	Name of the signal
Type	Signal value type, for example, float
Value	Signal live value
Unit	Signal value unit, for example, Hz
Category	Signal category, for example, Public or Internal
Sensor	Sensor number (1-8)
Encoder	Encoder number (1-4)
Function	Signal name used in the robot controller, for example, <i>TrigVis</i>
Label	Connector on the CTM, for example, X11
Description	Information on what the signal represents

Continues on next page

4 Conveyor Tracking tab in RobotStudio

Continued

List all signals

To list all the signals:

	Action
1	Right-click the desired CTM in the tree view browser and then click Signals .

List all encoder signals

To list all the encoder signals:

	Action
1	Right-click the desired encoder in the tree view browser and then click Signals .

List all sensor signals

To list all the sensor signals:

	Action
1	Right-click the desired sensor in the tree view browser and then click Signals .

Backup and Restore

You can save a backup of the CTM settings. The backup will contain:

- Data about the CTM configurations, including encoder and sensor parameters and new names. While restoring the backup, these settings are applied to the target CTM.
- Network settings
- Firmware version information

Creating a backup of the CTM

Keeping a backup is recommended to simplify the task of replacing the CTM with a new unit.

To create a backup of the CTM:

	Action
1	Right-click the CTM in the tree view browser and then select Create Backup... . The Create Backup from CTM dialog box opens.
2	Enter a Backup Name .
3	Enter or select a save Location and click OK . A backup of the CTM is created in the entered location.

Restoring a backup

To restore a backup of the CTM:

	Action
1	Right-click the CTM in the tree view browser and then select Restore Backup... . The Restore from Backup to CTM dialog box opens.
2	Enter or browse to the required Location , select backup folder and click Open . The backup of the CTM saved in the folder is displayed in the list of the Available backups section.
3	Select the required backup from the list and click OK . The CTM's settings will be restored according to the backup, after a restart of the CTM.

Continues on next page

Firmware upgrade

A firmware upgrade is normally provided by ABB, as a .cab file.

**WARNING**

It is recommended to backup the CTM configurations, as a firmware upgrade may result in the factory default network settings.

To upgrade the firmware of the CTM:

	Action
1	Login as an advanced user to upgrade the firmware, see Authenticate on page 51 .
2	Right-click the CTM and then click Firmware Upgrade . The Firmware Upgrade dialog box opens. It displays the current version of the firmware.
3	Click Browse , navigate to the .cab file and then click Open . The Verify Software dialog box opens. It displays the publisher of the file.
4	If the publisher is trusted, click Yes . The Firmware Upgrade dialog box opens. It displays the new firmware version.
5	Verify it and click Upgrade , to download it. The CTM will restart after the download and installation.

After a firmware upgrade, you may have to restore the network settings of the CTM.

Restore a backup (see [Restoring a backup on page 54](#)) or manually enter the correct network settings.

This page is intentionally left blank

5 Configuration and calibration

5.1 About software installation and configuration

Introduction

This section is a step-by-step instruction on how to install, connect, configure and calibrate conveyor work areas for conveyor tracking. This includes robot controller, conveyor tracking module, encoders and sensors for object detection.

The encoder and conveyor tracking module descriptions in this chapter are not valid for the option *Indexing Conveyor Control*.

Installed software on delivery

The conveyor tracking functions and the RAPID instructions are specified in the license. The conveyor software is already installed on delivery.

The Conveyor Tracking option installs one conveyor work area to the robot controller configuration (that is, in the system parameters). To add more conveyor work areas, see [Installing additional conveyor work areas for conveyor tracking on page 79](#).

The PickMaster 3 option has a Conveyor Configuration suboption, that can be used to specify how many conveyor work areas should be installed (0 – 6).

5.2 Configuring conveyor tracking module

Introduction

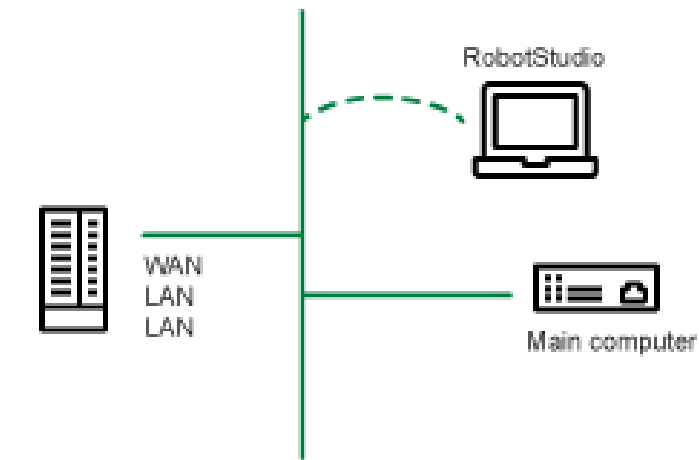
The conveyor tracking module (CTM, DSQC2000) provides an interface to monitor up to 4 encoders and 8 object detecting sensors. Encoders and sensors can be shared for usage with different conveyor work areas. One CTM unit can be used simultaneously by up to 40 robot controllers. To be able to use a CTM, the robot controller needs the software option *Tracking Unit Interface*.

Cybersecurity for installations with CTM

For information on addressing security threats in a network installation with CTM and ABB robots, see *Operating manual - Integrator's guide OmniCore*.

Connecting a robot controller to the CTM

The robot controller is connected to the CTM through an Ethernet network. The CTM has 3 network interfaces: WAN and 2 LAN. The WAN interface is used to communicate with robot controllers. If the CTM is shared by more than one robot controller, it should be connected to the public network port on the robot controller. When used with a single robot controller, additionally it is possible to connect a CTM on the private network, for example, LAN 2 of the robot main computer.



xx1800000382

**Note**

To establish a communication between the robot controller and the CTM, the IP address of the robot controller must be on the same subnet as the CTM.

To set up and verify a communication between the CTM and the robot controller:

	Action
1	Configure the IP address of the CTM, see Network settings on page 50 .

Continues on next page

	Action
2	Configure a RobICI device. For example, CTM1, in the robot controller, see Configuring CTM in the robot controller on page 63 .
3	To configure the IP address of the RobICI device, use the IP address of the CTM. In RobotStudio, right-click the RobICI device in the RobICI Device Type under the I/O System tree and select Edit RobICI Device(s) . Assign the IP address to the Server IP address field.
4	If CTM is connected to the public network of the robot controller, configure the IP address of the robot controller.
5	If CTM is connected to the public network of the robot controller, configure the network firewall on the controller. Configuration is done in RobotStudio under Configuration\Communication\Firewall Manager. Set the parameter <i>Enable on Public Network</i> to YES.
6	Restart the robot controller.
7	Check the state of the I/O devices. The communication is established, if the CTM1 is in Running state.

Examples

Example 1

If the CTM is connected to the public network of the robot controller:

CTM WAN interface	192.168.8.48
Robot controller, RobICI device IP address	192.168.8.48
Robot controller, IP address	192.168.8.50

To view or change the IP address of the robot controller on the public network, edit the network settings in RobotStudio. See *Operating manual - Integrator's guide OmniCore*.

Example 2

If the CTM is connected to the private network of the robot controller:

CTM WAN interface	192.168.125.20
Robot controller, RobICI device IP address	192.168.125.20
Robot controller, Private Network IP address	192.168.125.1

It is not possible to change the IP address of the private network of the robot controller.

Configuring the CTM with RobotStudio

The CTM unit is configured using RobotStudio. You can connect a computer with RobotStudio to the WAN or LAN port of the CTM.

In the **Controller** tab click on **Conveyor Tracking** from the **Configuration** group. A **Conveyor Tracking** tab is displayed.

You can search for any conveyor tracking units on the network, see [Connecting and adding a CTM on page 49](#).

Continues on next page

5 Configuration and calibration

5.2 Configuring conveyor tracking module

Continued

Minimum configuration

The setup of The IP address of the WAN interface is the minimum configuration required, see [Network settings on page 50](#). A communication between the CTM and a robot controller can be established only after the IP address is defined.

You can change the following parameters:

- Name of the CTM and name of individual encoders and sensors, see [Rename on page 52](#).
- Encoder parameters, see [Configuring an encoder on page 52](#).
- Sensor parameters, see [Configuring a sensor on page 52](#).

5.3 Verifying the installation of encoders and sensors

Description

The CTM provides functionality that simplifies the verifications of installed encoders and sensors. These verifications can be done by connecting RobotStudio to the CTM. The verifications are based on the monitoring of certain CTM signals and are independent of robot controllers.

Verifying if the encoder is electrically connected to the power

This test is only applicable if the encoder is powered by the encoder interface.

To verify if the encoder is electrically connected to the power:

	Action
1	Open the signals for the encoder, see List all encoder signals on page 54 .
2	Restart the CTM.
3	Verify the value of <i>TwoPhaseEncX:PowerWired</i> . If the value is 1, it indicates that the encoder is connected to the power supply. If the value is 0, it indicates that the encoder is not connected to the power supply.

Verifying if the trigger signal is electrically connected to a camera

To verify if the trigger signal is electrically connected to a camera:

	Action
1	Open the signals for the sensor, see List all sensor signals on page 54 .
2	Verify if the value of <i>DigoutX:Wired</i> is 1, else check the connection of the trigger signal and the GND with the camera.

Verifying the encoder function

To verify the encoder function:

	Action
1	Open the signals for the encoder, see List all encoder signals on page 54 . The encoder function can be checked by monitoring the signal <i>EncX:position</i> that displays the encoder count for encoder X.
2	Move the conveyor and verify if the <i>EncX:position</i> changes accordingly.
3	If the value does not change or is unstable, verify if the encoder is properly installed, electrically and mechanically.

Verifying the sensor function

To verify the sensor function:

	Action
1	Open the signals for the sensor, see List all sensor signals on page 54 . The sensor function can be checked by monitoring the signal <i>syncXencY:Value</i> , it displays the encoder count for encoder Y at the latest sync input event on sensor X.
2	Move the conveyor Y a short distance.
3	Generate a sync input from the sensor, for example by detecting a object with an I/O sensor or by triggering an image with a camera sensor.

Continues on next page

5 Configuration and calibration

5.3 Verifying the installation of encoders and sensors

Continued

	Action
4	Verify if the <i>syncXencY:Value</i> is same as the value of the <i>EncY:position</i> .
5	If the value is not the same, verify if the encoder is properly installed, electrically and mechanically.

Verifying the direction of positive motion

To verify the direction of positive motion:

	Action
1	Open the signals for the encoder, see List all encoder signals on page 54 . The <i>EncX:position</i> signal displays the encoder count for encoder X.
2	Move the conveyor towards the robot and verify if the signal value increases or decreases. The <i>EncX:position</i> signal displays the encoder count for encoder X.
3	If the signal value increases, the direction of positive motion is correct. If the signal value decreases, reverse the A and B inputs from the encoder to the encoder interface.

Identifying the gear ratio

Open the signals for the encoder, see [List all encoder signals on page 54](#).

The gear ratio defines the motion distance of the conveyor as a function of measured encoder counts.

The gear ratio can be used to configure the system parameter *CountsPerMeter* for the conveyor work area. It can be identified with the following procedure:

	Action
1	Record the first encoder count C1 as <i>EncX:position</i> .
2	Move the conveyor forward for at least 1 meter. Measure the actual movement distance D. For a linear conveyor, D is measured in meter. For a rotational conveyor, D is measured in radians.
3	Record a second encoder count C2 as <i>EncX:position</i> .
4	Calculate the gear ratio = $(C2 - C1) / D$.

5.4 Configuration of robot controllers

Configuring CTM in the robot controller

In the robot controller configuration, the CTM is represented as a RobICI I/O device on the RobICI network. The RobICI network is an internal ABB communication protocol, see *Operating manual - Integrator's guide OmniCore*.

The option *Conveyor Tracking* installs one RobICI device to the RobICI network.

The option *PickMaster 3* installs one RobICI device if 1-6 conveyor work areas have been selected in the sub-option *Conveyor Configuration*.

Additional RobICI devices can be added using RobotStudio. There are templates available to setup *Conveyor Tracking Device 2*, *Conveyor Tracking Device 3*, and *Conveyor Tracking Device 4*. However, one RobICI device (with 4 encoder inputs) is sufficient in most scenarios. For example, in a scenario with 5-6 conveyor work areas, some of them are typically located on the same conveyor belt and can share the same encoder.

To add a device in RobotStudio, in the controller tab, select **Configuration -> I/O System -> RobICI Device**. Right-click and select **New RobICI Device**. Click **OK** to save the new device.

The following needs to be configured for a RobICI device:

- The configured IP address of the CTM.
- The network firewall needs to be configured on the controller. Configuration is done in RobotStudio under Configuration\Communication\Firewall Manager. Set the parameter *Enable on Public Network* to YES.

Continues on next page

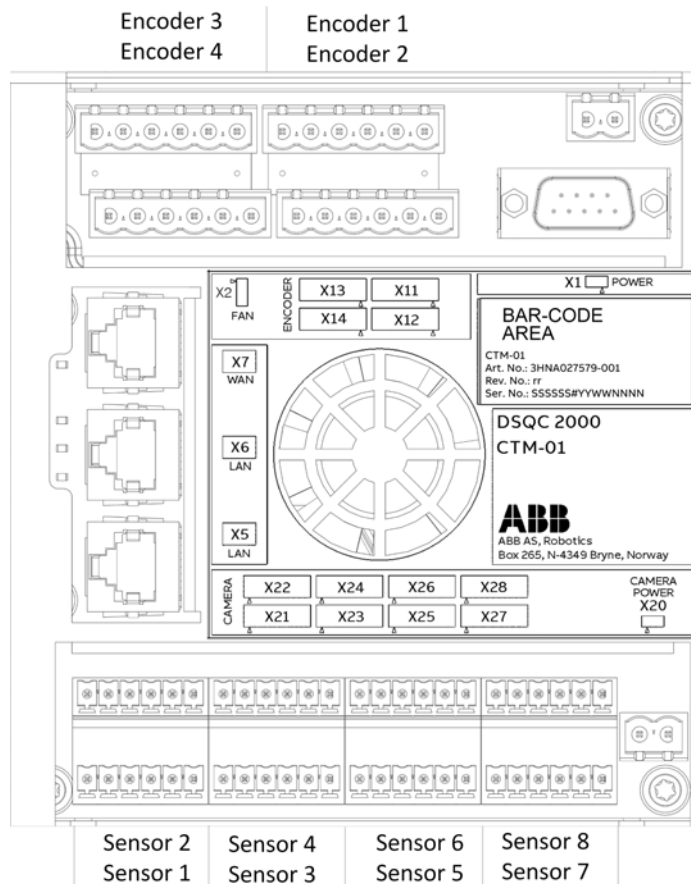
5 Configuration and calibration

5.4 Configuration of robot controllers

Continued

Configuring a conveyor work area using CTM

For each conveyor work area, a selection needs to be made on which encoder and which sensor that shall be used for conveyor tracking.



xx1800000384

The selection is made by modifying the DeviceMap of nine I/O signals that belongs to the conveyor work area. The nine signals are mapped to the configured CTM device. The value in the DeviceMap column represent encoder and sensor numbers.



Note

One encoder may be used in parallel by multiple conveyor work areas on the same robot controller, or on different robot controllers. One sensor may also be used by multiple conveyor work areas, however, the sensor trigger function should not be operated simultaneously by more than one work area.

Example 1

In this example, conveyor work area 1 is used with encoder 1 and sensor 1 on CTM1. The DeviceMap is adjusted accordingly:

Name	Device	DeviceMap	DeviceMap digits represent
c1Counts	CTM1	Enc1:position	Encoder 1
c1CountsPerSec	CTM1	Enc1:velocity	Encoder 1

Continues on next page

Name	Device	DeviceMap	DeviceMap digits represent
c1SpeedBandWidth	CTM1	e1_SpeedBand-Width:SetPoint	Encoder 1
c1TrigVis	CTM1	Digout1:Setpoint	Sensor 1
c1TrigAutoMode	CTM1	Cam1:Connect	Sensor 1
c1TrigAutoDist	CTM1	Cam1:Interval	Sensor 1
c1TrigAutoEncNo	CTM1	Cam1Enc:DeviceNo	Sensor 1
c1SoftSync	CTM1	Sync1Enc1:SyncTrig	Sensor 1, encoder 1
c1CntFromEnc	CTM1	Sync1Enc1:Value	Sensor 1, encoder 1

Example 2

In this example, conveyor work area 2 is used with encoder 2 and sensor 8 on CTM1. The DeviceMap is adjusted accordingly:

Name	Device	DeviceMap	DeviceMap digits represent
c2Counts	CTM1	Enc2:position	Encoder 2
c2CountsPerSec	CTM1	Enc2:velocity	Encoder 2
c2SpeedBandWidth	CTM1	e2_SpeedBand-Width:SetPoint	Encoder 2
c2TrigVis	CTM1	Digout8:Setpoint	Sensor 8
c2TrigAutoMode	CTM1	Cam8:Connect	Sensor 8
c2TrigAutoDist	CTM1	Cam8:Interval	Sensor 8
c2TrigAutoEncNo	CTM1	Cam8Enc:DeviceNo	Sensor 8
c2SoftSync	CTM1	Sync8Enc2:SyncTrig	Sensor 8, encoder 2
c2CntFromEnc	CTM1	Sync8Enc2:Value	Sensor 8, encoder 2

Example 3

In this example, conveyor work area 3 is used with encoder 4 and sensor 5 on CTM1. The DeviceMap is adjusted accordingly:

Name	Device	DeviceMap	DeviceMap digits represent
c3Counts	CTM1	Enc4:position	Encoder 4
c3CountsPerSec	CTM1	Enc4:velocity	Encoder 4
c3SpeedBandWidth	CTM1	e4_SpeedBand-Width:SetPoint	Encoder 4
c3TrigVis	CTM1	Digout5:Setpoint	Sensor 5
c3TrigAutoMode	CTM1	Cam5:Connect	Sensor 5
c3TrigAutoDist	CTM1	Cam5:Interval	Sensor 5
c3TrigAutoEncNo	CTM1	Cam5Enc:DeviceNo	Sensor 5
c3SoftSync	CTM1	Sync5Enc4:SyncTrig	Sensor 5, encoder 4
c3CntFromEnc	CTM1	Sync5Enc4:Value	Sensor 5, encoder 4

5.5 Calibrating CountsPerMeter

Description

If the exact gear ratio between the encoder and the conveyor is unknown (typically the case) then the system parameter *CountsPerMeter* must be calibrated using either a tape-measure or the robot TCP as a measuring device.

If the robot TCP is used as the measuring device then an accurately defined tool must be used.

If the gear ratio is identified for the encoder, see [Identifying the gear ratio on page 62](#), its value can be used for *CountsPerMeter*. Alternatively, the method mentioned in the [Calculating and calibrating CountsPerMeter on page 66](#) section can be used.

Calculating and calibrating CountsPerMeter

The counts from the encoder can be found by reading a predefined I/O signal. For example CNV1 the signal name is c1counts. Note the counts for the current conveyor position (named counts_1 in the formula below) and then move away the conveyor at least 1 meter. Read the counts again (named counts_2) and measure the distance the conveyor is moved (named measured_meters). The accuracy will be best if this distance is large as possible within the work space. Use a tape-measure (or differences in robot tool position) to find the exact distance. Use this formula to calculate *CountsPerMeter*:

$$CountsPerMeter = \frac{(counts_2 - counts_1)}{(measured_meters)}$$

Use this procedure to modify *CountsPerMeter*.

- 1 Open the **Configuration Editor** and select **Topics and Process**.
- 2 Select the type **Conveyor Ici**.
- 3 Select **CountsPerMeter** and change the value.

5.6 Calibrating the conveyor base frame

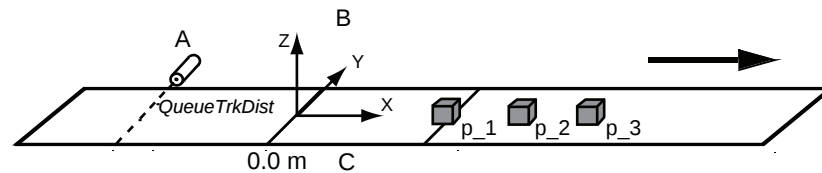
Introduction to the conveyor base frame

The accuracy of the conveyor tracking depends on the accuracy of the conveyor base frame calibration. For linear conveyors, use a method that uses the robot TCP to measure the position and orientation of the conveyor in the work space.

Prerequisites

Before calibrating the base frame of the conveyor the values for the *CountsPerMeter* and *QueueTrkDist* must be correct. See [Calibrating CountsPerMeter on page 66](#).

The conveyor base frame calibration method will use the measurement of 4 positions of the same object on the conveyor to determine the conveyor base frame (B), as shown in the following figure.



xx1200001098

A	Synchronization switch
B	Base frame
C	Conveyor distance

Before defining the 4 positions, an object must be defined on the conveyor.

Creating the work object (option Conveyor Tracking)

Use this procedure to create a work object. To create a work object for a servo controlled indexing conveyor, see [Creating the work object on page 152](#).

- 1 Step forward through a RAPID program containing the two instructions:

```
ActUnit CNV1;
WaitWObj wobjcnv1;
```

Define the conveyor coordinated work object, see [Defining a conveyor coordinated work object on page 84](#).

- 2 Run the conveyor until an object passes the sync switch and the 0.0 m point. The WaitWObj instruction will end execution.
- 3 Stop the conveyor.
- 4 Calibrate the base frame, see [Calibrating the base frame on page 68](#).



Note

Once an object is on the conveyor and beyond the 0.0 m point, it is possible to use the base frame calibration method to define the conveyor position and orientation in the work space.

Continues on next page

5 Configuration and calibration

5.6 Calibrating the conveyor base frame

Continued

Creating the work object when the conveyor is not moving (option Conveyor Tracking)

The controller uses the signal *doCxxNewObjStrobe* to create an object using the current encoder counter and will generate a pulse on the signal *doCxxCntToEncStr*. Use these RAPID instructions as an example to create a work object when the conveyor is not moving.

```
ActUnit CnvMecUnit;
WaitTime 0.5;
DropWObj WorkObject;
WaitTime 0.5;
!
CONNECT NewObj WITH ObjTrap;
ISignalDO cxxCntToEncStr,1,NewObj;
PulseDO cxxNewObjStrobe;
WaitUntil NewObjReported=TRUE;

TRAP ObjTrap
  WaitTime 0.2;
  NewObjReported:=TRUE;
  RETURN;
ENDTRAP
```

Limitations

The conveyor speed must be zero

All other objects in the queue will be removed

Creating the work object (option PickMaster)

Load and run the RAPID program *PrepareCalib.prg*. It is located in the HOME folder.

Select the conveyor work area, CNVX, which needs calibration and let the program execution complete.

Calibrating the base frame

Use this procedure to calibrate the conveyor base frame (applicable for both *Conveyor Tracking* and *Indexing Conveyor Control*).

If the PickMaster option is used, see *Application manual - PickMaster* @3.

- 1 On the FlexPendant, open the **Calibration** window and select the conveyor.
- 2 Tap **Base Frame**.
- 3 Tap **4 Point**.
- 4 Select the first point, **Point 1**. This point will be the origin for the user frame in the conveyor coordinated work object.
- 5 Point out Point 1 on the object on the conveyor with the robot TCP.
- 6 Modify the position by tapping **ModPos**.
- 7 Move the conveyor in the positive direction and repeat the above for the points 2, 3, and 4.
- 8 Tap **OK** to calculate the base frame for the selected conveyor mechanical unit.

Continues on next page

A dialog with the calibration result is shown. The calculation log shows the conveyor base frame expressed in the world coordinate system, see [Calculation result for the base frame on page 69](#).

- 9 To save the calculation result in a separate file for later use in a PC:
 - a Tap **File**.
 - b Specify a name and a location where to save the result.
 - c Tap **OK**.
- 10 If the estimated error is:
 - Acceptable, tap **OK** to confirm the new user frame.
 - Not acceptable, tap **Cancel** and redefine.
- 11 Restart the controller and verify the results of the calibration. See [Verifying the base frame calibration on page 69](#).

Calculation result for the base frame

Field	Description
Unit	The name of the mechanical unit for which the definition of base frame has been done.
List contents	Description
Method	Displays the selected calibration method.
Mean error	The accuracy of the robot positioning against the reference point.
Max error	The maximum error for one positioning.
Cartesian X	The x coordinate for the base frame.
Cartesian Y	The y coordinate for the base frame.
Cartesian Z	The z coordinate for the base frame.
Quaternion 1-4	Orientation components for the base frame.

Verifying the base frame calibration

If the PickMaster 3 option is used, see *Application manual - PickMaster @3*, section *Verifying the calibrations*.

After restarting the controller, use this procedure to verify the conveyor base frame calibration.

- 1 Create a work object, see:
 - [Creating the work object \(option Conveyor Tracking\) on page 67](#), for *Conveyor Tracking*.
 - [Calibrating the base frame on page 152](#), for *Indexing Conveyor Control*.
- 2 Move the robot tool center point back to the previously chosen point 1 on the work object.
- 3 In the **Jogging** window, read the X, Y, Z position of the tool center point. Make sure to use the correct tool and wobcnv1.
- 4 The robot TCP x, y, and z position in the work object coordinates should be 0.0 mm (or very close to that).

Continues on next page

5 Configuration and calibration

5.6 Calibrating the conveyor base frame

Continued

- 5 In the **Jogging** window, select the work object wobcnv1 and coordinate system WObj and jog the robot in the x, y, and z directions of the conveyor. Verify that the x-direction is in the direction of positive motion of the conveyor.

5.7 Defining conveyor start window and sync separation

Start window

The start window is the length along the conveyor in which objects are tracked and are ready for connection. When a `WaitWObj` instruction is issued the system will connect to the first object inside the start window or wait otherwise.

If an object goes beyond the start window then it is no longer tracked and it is not available for connection. Such objects are automatically skipped. The purpose of the start window is to provide a buffer of objects for speed variations of the conveyor. If an object is connected within the start window then it should be certain that the motion coordinated to the object can be completed before the working area limit or maximum distance is reached.



Note

It is recommended to configure a non zero start window, otherwise all objects will be skipped and no object will be available for connection (default value is 10 m).

Sync separation

The parameter *Sync Separation* is used to filter out unwanted sync signals from a synchronization switch. This parameter establishes a minimum distance that the conveyor must move after one sync signal before a new sync signal is accepted as a valid object.

Defining start window and sync separation

Sync separation is configured for each encoder directly on the tracking unit, see [Configuring the CTM with RobotStudio on page 59](#).

The start window is configured on the robot controller for the conveyor work areas. Use this procedure to define the start window.

- 1 Open the **Configuration Editor** and select the topic **Process**.
- 2 Select the type **Conveyor Ici**.
- 3 Select the parameter *StartWinWidth* and change the value.

5.8 Avoiding robot reach problems

Description

A strategy is required to avoid coordination of motion beyond the reach of the robot.

With the Conveyor Tracking option, it is possible to setup maximum and minimum distances of the conveyor, to simplify this task.

With the PickMaster 3 option, coordination of motion beyond the reach of the robot can be prevented with the instruction `UseReachableTargets`.

Maximum and minimum distances

It is possible to monitor the position of the conveyor and automatically drop any connected objects that move outside the maximum or minimum specified distance.

The purpose is to prevent coordination of motion beyond the work area of the robot for both forward and backward operation of the conveyor.

It is recommended to set a negative value of the parameter *minimum distance*, otherwise all objects will be dropped as soon as they are connected. To have a positive minimum value to enter the work area it is recommended to use the optional argument `\RelDist` with the `WaitWObj` instruction.

Defining maximum and minimum distances

Use this procedure to define the distances.

- 1 Open the **Configuration Editor** and select the topic **Process**.
- 2 Select the type **Conveyor systems**.
- 3 Select **CNV1**.
- 4 Select the parameters *maximum distance* and *minimum distance* and change the values.
- 5 Select the type **Conveyor Ici**.
- 6 Select **ICI1**.
- 7 Select the parameter *Supervise max_dist Off* and set the value to **No**.

See [Topic Process on page 96](#).

5.9 Using a trigger output on the CTM

Description

The trigger output signals in the CTM are primarily intended to control image acquisition of cameras.

A trigger output can be operated in two ways, manual mode or auto mode. Auto mode is used when a fixed distance is required between consecutive images. Manual mode is used when image acquisition should be controlled by a signal, for example from a RAPID program.

The DO signal, *cXTrigAutoMode*, is used to switch modes, where the value 1 activates auto mode and 0 activates manual mode.

Manual mode

In manual mode, the DO signal, *cXTrigVis*, can be used to set or reset the trigger output, for example from a RAPID program.

Auto mode

In auto mode, the trigger output will be pulsed by the CTM at a selected frequency according to the encoder distance.

The supervised encoder (1, 2, 3, or 4) is selected by the GO signal, *cXTrigAutoEncNo*.

The distance interval (number of counts) is selected by the GO signal, *cXTrigAutoDist*.

The pulse length of the signal can be modified by configuring the CTM through RobotStudio, see [Configuring a sensor on page 52](#).

PickMaster 3 Option

With the PickMaster 3 option, select *cXTrigVis* as the trigger signal in the PickMaster application, see *Application manual - PickMaster @3*. Auto mode selection, encoder selection, and distance interval is fully managed by PickMaster. The pulse length can be modified by configuring the CTM through RobotStudio, see [Configuring a sensor on page 52](#).

5.10 Defining the robot adjustment speed

Adjusting the speed

When entering conveyor tracking, the robot must adjust its speed to the speed of the conveyor. The speed with which the robot catches up to the conveyor for the first motion is controlled by the parameter adjustment speed.

See [Type Conveyor systems on page 96](#) on how to derive a proper value for this parameter.

Defining the robot adjustment speed

Use this procedure to define the robot adjustment speed.

- 1 Open the **Configuration Editor** and select the topic **Process**.
- 2 Select the type **Conveyor systems** and select **CNV1**.
- 3 Select the parameter *adjustment speed* and change the values.

See [Type Conveyor systems on page 96](#).

5.11 Additional adjustments

Description

There are more parameters that can be adjusted in the motion system.

Regulating CPU load and accuracy

The parameter *Path Resolution* specifies the period of the path planner in planning steps along the path (no units). Step calculations require lots of CPU time and if steps cannot be calculated in time to keep the robot on the path then error **50082 Deceleration Limit** may occur. As conveyor tracking increases the general CPU load then the parameter *Path Resolution* can be increased if this error occurs. See [Type Motion Planner on page 94](#).

Defining the mechanical unit parameters

The mechanical unit parameters define the name used in RAPID, and the conditions for activation and deactivation.

These parameters can be changed to ensure activation of the conveyor. See [Type Mechanical Unit on page 94](#).

Defining additional robot parameters

Some parameters in the type *Robot* (topic *Motion*) can need adjustments. See [Type Robot on page 94](#).

5 Configuration and calibration

5.12 Configuring a track motion to follow a conveyor

5.12 Configuring a track motion to follow a conveyor

Track following a conveyor

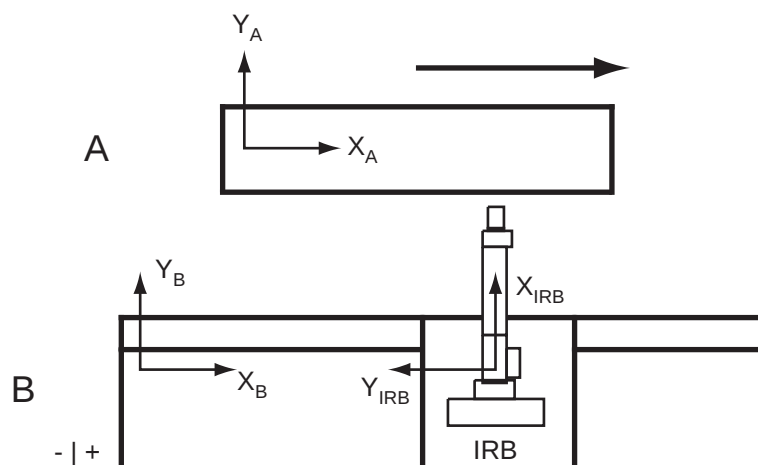
If the robot is mounted on a track, and the track is parallel to the conveyor, then the motion can be configured so that the track follows the conveyor. The robot and the track must be configured for *coordinated track motion*. See *Operating manual - OmniCore* for information on configuring coordinated motion.

Once the robot and track are configured for coordinated motion, then conveyor tracking will automatically use the track to follow the conveyor. The track will maintain the same position relative to the object as the object moves on the conveyor as it was during programming.

The track and robot base frame must be defined so that positive motion of the track is in the same direction as the conveyor. In some installations this may require a re-definition of the track's direction of positive motion and calibration position.

Example configuration of track and conveyor directions

The following figure shows an example configuration.



xx1200001099

A	Conveyor The arrow shows the direction of movement for the conveyor
B	Track motion The - + shows the direction of movement for the track
Conveyor quaternion	1, 0, 0, 0
Robot base quaternion	0.7071, 0, 0, 0.7071
Track base quaternion	1, 0, 0, 0

Recommendations for programming

Avoid moving the track when programming the conveyor coordinated instructions in the RAPID program. All motions of the track relative to the conveyor are saved and played back during conveyor tracking. Tracks typically have an acceleration ability that is far below that of the robot joints. If the track must move relative to

Continues on next page

the object then this will require an acceleration that will cause a reduction of the robot's TCP speed along the path in order to maintain coordination.

Tracking the conveyor with a robot instead of a track

If the robot base is not coordinated with the track axis, then the robot will do conveyor tracking without using the track. If the robot base frame is coordinated with the track, and conveyor tracking with the robot (instead of the track) is wanted, then change the parameter *Track Conveyor With Robot* in the type *Robot* (topic *Motion*). See [Type Robot on page 94](#).

5 Configuration and calibration

5.13 Installing conveyor tracking software

5.13 Installing conveyor tracking software

Reloading saved *Motion* parameters

If the CNV1 mechanical unit does not appear on the FlexPendant then the *Motion* parameters must be reloaded manually in RobotStudio, from the robot controller <system name>\...\cnv\cnv1_moc.cfg.

5.14 Installing additional conveyors work areas for conveyor tracking

Installing additional conveyor work areas for conveyor tracking

The option *Conveyor Tracking* installs one conveyor work area in the configuration (that is, in the system parameters). The PickMaster 3 option installs 0-6 conveyor work areas depending on the selected Conveyor Configuration. If additional conveyors work areas should be tracked with the same robot controller then the parameters for the additional conveyor work areas must be loaded manually. Up to 6 conveyor work areas can be installed on one controller.

For *Indexing Conveyor Control*, see [Installing the additional axis for servo control on page 145](#).

Installing additional conveyors work areas

Use this procedure to install additional conveyor work areas for conveyor tracking, for DSQC2000:

- 1 If the encoder is not connected to a preconfigured RobICI device, add a new RobICI device via RobotStudio, see [Configuring CTM in the robot controller on page 63](#).



Note

Several conveyor work areas can share the same encoder interface on a DSQC2000, for example two robots picking objects from the same conveyor.

- 2 In RobotStudio, load the files `cnvici2_eio.cfg`, `cnvici2_prc.cfg`, and `cnv2_moc.cfg`. (Repeat for additional conveyors.)
These files are found in the installation folder of Robotware, for example, `C:\...\ABB\RobotWare\RobotControl_version\options\cnv`.
- 3 Restart the system.
- 4 For each conveyor work area, define the correct DeviceMap for the I/O signals, see [Configuring a conveyor work area using CTM on page 64](#).
- 5 Restart the system.

This page is intentionally left blank

6 Programming

6.1 Programming conveyor tracking

Prerequisites

To create a program that uses conveyor tracking and a conveyor coordinated work object, a work object must be present in the start window. An object must be moved into the start window. If there are several objects already on the conveyor, then it can be necessary to first clear the object queue and then move the conveyor.

6.2 Working with the object queue

I/O signals and the object queue

There are several I/O signals that allow a user or a RAPID program to monitor and control the object queue. The following table shows the I/O signals that impact the object queue.

I/O signal	Description
c1ObjectsInQ	Group input showing the number of objects in the object queue.
c1Rem1PObj	Remove first pending object from the object queue. Setting this signal will cause the first pending object to be dropped from the object queue. Pending objects are objects that are in the queue but are not connected to a work object.  Note It takes some time for the signal to apply. After setting the signal, it is therefore advisable to wait for 0.15 s before a new work object is connected.
c1RemAllPObj	Remove all pending objects. Setting this signal will empty all objects from the object queue. If an object is connected, then it is not removed.  Note It takes some time for the signal to apply. After setting the signal, it is therefore advisable to wait for 0.15 s before a new work object is connected.
c1DropWObj	Setting this signal will drop the tracked object and disconnect that object. The object is removed from the queue. This should not be set from RAPID, use a DropWobj instruction instead.

6.3 Activating the conveyor

Activation

As an additional axis and mechanical unit the conveyor must be activated before it can be used for work object coordination. The usual `ActUnit` instruction is used to activate the conveyor and `DeactUnit` can be used to deactivate the conveyor.

As an additional axis and mechanical unit the conveyor must be activated before it can be used for work object coordination. The usual `ActUnit` instruction is used to activate the conveyor and `DeactUnit` can be used to deactivate the conveyor.

By default, the conveyor is installed non-active on start. The conveyor can be configured to always be active at start. See [Type Mechanical Unit on page 94](#).

Handling of the objects queue after activation (DSQC2000 only)

The object queue is handled by the controller only after the conveyor activation. It is not recommended to access any conveyor related IO signal or RAPID instruction before activating the conveyor.

6 Programming

6.4 Defining a conveyor coordinated work object

6.4 Defining a conveyor coordinated work object

Settings for wobjcnv1

Use these settings to define a new conveyor coordinated work object, wobjcnv1 (data type wobjdata).

Data	Data type	Description
ufprog (user frame programmed)	bool	Defines if a fixed user coordinate system is used. Set to FALSE for a movable user coordinate system, that is, coordinated to conveyor.
ufmec (user frame mechanical unit)	string	The conveyor mechanical unit with which the robot movements are coordinated. Specified with the name that is defined in the system parameters, for example CNV1.

For more information about data types, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

For more information about defining work objects, see *Operating manual - OmniCore*.

6.5 Waiting for a work object

Waiting for a work object

Motions that should be coordinated with the conveyor cannot be programmed until an object on the conveyor has been connected with a `WaitWObj` instruction.

If the object on the conveyor is already connected from a previous `WaitWObj` or if connection was established during activation, then execution of a second `WaitWObj` instruction will cause an error. This error can be handled in an error handler.

See [WaitWObj - Wait for work object on conveyor on page 99](#).

6.6 Programming the conveyor coordinated motion

Programming the conveyor

- 1 Create a program with the following instructions:

```
ActUnit CNV1;  
ConfL\Off;  
MoveL waitp, v1000, fine, tool1;  
WaitWObj wobjcnv1;
```
- 2 Single-step the program past the `WaitWObj` instruction.
If there is an object already in the start window on the conveyor then the instruction will return, else execution will stop while waiting for an object on the conveyor.
- 3 Run the conveyor until a work object is created.
The program exits the `WaitWObj` and is now connected to the object. Stop the conveyor with the object in an accessible position on the workspace.
- 4 Program the `MoveL`, `MoveC`, `PaintL`, or `PaintC` instructions using the *wobjcnv1* conveyor coordinated work object.
- 5 End the coordinated motion with a fixed-frame work object (not coordinated to the conveyor). Note that this is the only way to end the use of the work object.
- 6 Add a `DropWObj wobjcnv1;` instruction.
- 7 If this is the end of the program, or the conveyor is no longer needed, then add a `DeactUnit CNV1;` instruction.



Note

In the case of CTM (DSQC2000 only) it is recommended to leave the conveyor active to avoid missing objects.

Example

The following program shows an example for a conveyor tracking program.

```
ConfL\Off;  
MoveJ p0, vmax, fine, tool1;  
ActUnit CNV1;  
WaitWObj wobjcnv1;  
MoveL p10, v1000, z1, tool1\Wobj:=wobjcnv1;  
MoveL p20, v1000, z1, tool1\Wobj:=wobjcnv1;  
MoveL p30, v500, z20, tool1\Wobj:=wobjcnv1;  
MoveL p40, v500, fine, tool1;  
DropWObj wobjcnv1;  
MoveL p0, v500, fine;  
DeactUnit CNV1;
```

6.7 Dropping a work object

Introduction

A connected work object can be dropped once conveyor coordinated motion has ended. Make sure that the robot motion is no longer using the conveyor positions when the work object is dropped. If motion still requires the positions then a stop will occur when the object is dropped.

It is not necessary to be connected in order to execute a DropWObj instruction. No error will be returned if there was no connected object.

Finepoints

Conveyor coordinated motion does not end in a finepoint. As long as the work object is coordinated to the conveyor, the robot motion will be coupled to the conveyor even in a finepoint. A fixed-frame or non-conveyor work object must be used in a motion instruction before dropping the conveyor work object.

Corner zones

Take care when ending coordination in a corner zone and executing the DropWObj instruction as the work object will be dropped before the robot has left the corner zone and the motion still requires the conveyor coordinated work object.

6.8 Entering and exiting conveyor tracking motion in corner zones

Enter and exit coordinated motion

Once a WaitWObj instruction has connected to a work object on the conveyor then it is possible to enter and exit coordinated motion with the conveyor via corner zones.

Example

```
MoveL p10, v1000, fine, tool1;
WaitWObj wobjcnv1
! enter coordination in zone
MoveL p20, v1000, z50, tool1;
MoveL p30, v500, z1, tool1\Wobj:=wobjcnv1;
MoveL p40, v500, z1, tool1\Wobj:=wobjcnv1;
MoveL p50, v500, z20, tool1\Wobj:=wobjcnv1;
MoveL p60, v1000, z50, tool1;
! exit coordination in zone
MoveL p70, v500, fine, tool1;
DropWObj wobjcnv1;
MoveL p10, v500, fine, tool1;
```

The move instruction ending coordination must be a fixed work object, for example ufprog is TRUE.

Take care when exiting coordination in a corner zone and executing a DropWObj instruction.

The following example shows an incorrect method for ending coordination in corner zones:

```
...
MoveL p50, v500, z20, tool1\Wobj:=wobjcnv1;
MoveL p60, v1000, z50, tool1;
! exit coordination in zone
DropWObj wobjcnv1;
```

This will cause an error, because the work object is dropped while robot is still in corner zone.

6.9 Information on FlexPendant

Available information on the FlexPendant

Information	Available in app
The conveyor position and speed The conveyor position is displayed in meters, and the conveyor speed in m/s. The speed will be 0 m/s until an object has passed the synchronization switch.	Jog
The position in millimeters of the conveyor object This value will be negative if a Queue Tracking Distance is defined. When an object passes the synchronization switch then the position will be automatically updated.	Jog
All signals that are defined for conveyor tracking	I/O

6 Programming

6.10 Programming considerations

6.10 Programming considerations

Performance limits

Conveyor tracking will be lost if joint speed limits are reached, particularly in singularities. Ensure that the path during tracking does not exceed the acceleration, speed, and motion capabilities of the robot.

Motion commands

Only linear and circular motion instructions are allowed for conveyor tracking, that is MoveL and MoveC.

Do not start tracking with a MoveC instruction as the resulting circle diameter will depend on the conveyor position. Always start tracking with a MoveL.

Finepoints

Finepoints are allowed during conveyor tracking. The robot will hold the TCP still relative to the conveyor and RAPID execution will continue, see [Finepoint programming on page 91](#).

ConfL

The RAPID instruction ConfL\Off must be executed before coordination with the conveyor. The purpose is to avoid configuration errors that would otherwise occur as the robot changes configuration during conveyor tracking.

Other RAPID limitations

The instructions StorePath and RestoPath do not function during conveyor tracking.

A Search instruction will stop the robot when hit or if the search fails. Make the search in the same direction as the conveyor moves and after the search stops continue with a move to a safe position. Use an error handler to move to a safe position if the search fails.

EoffsSet, EoffsOn, and EoffsOff have no effect for the conveyor additional axis, but can affect conveyor tracking with coordinated track. They also have effect on the sensor taught position.

Power fail restart is not possible with the synchronization option.

It is possible to use worldzone in conveyor tracking but during tracking the robot may move outside worldzone borders due to the position adjustment at servo level. Hence, taking margin is recommended when defining the worldzone.

6.11 Finepoint programming

Example with SetDO

While tracking the conveyor it is possible to use a finepoint. The following program example shows how motion may be stopped with respect to the conveyor so that an I/O signal may be set:

```
WaitWObj wobjcnv1
MoveL p1, v500, z20, tool1\Wobj:=wobjcnv1;
MoveL p2, v500, fine, tool1\Wobj:=wobjcnv1;
SetDO release_gripper;
WaitTime 0.1;
MoveL p3, v500, z20, tool1\Wobj:=wobjcnv1;
MoveL p4, v500, fine, tool1;
DropWObj wobjcnv1;
```

In the above example the SetDO will be executed after the robot arrives at p2. The robot will then track the conveyor for 0.1 seconds while the WaitTime instruction is executed. It will then move to p3 and on to p4 via a corner zone before ending coordination with a fixed work object (wobj0 in this case).

Example with stoppointdata

Finepoints can also be programmed with stoppointdata. To make the robot follow the conveyor during 0.1 second, the following program can be used.

```
VAR stoppointdata followtime:=[3,FALSE,[0,0,0,0],0,0.1, "",0,0];
WaitWObj wobjcnv1MoveL p1, v500, z20, tool1\Wobj:=wobjcnv1;
MoveL p2,v500,z1\Inpos:=followtime,tool1\Wobj:=wobjcnv1;
SetDO release_gripper;
MoveL p3, v500, z20, tool1\Wobj:=wobjcnv1;
MoveL p4, v500, fine, tool1;
DropWObj wobjcnv1;
```

For coordinated movements the \Inpos event will not occur for instructions Break and WaitTime. Instead, use the data type stoppointdata.

6.12 Operating modes

Operation in manual reduced speed mode (<250 mm/s)

When the conveyor is not moving, then the forward and backward hard buttons on the FlexPendant can be used to step through the program. New instructions can be added and programmed positions can be modified (ModPos).

To simplify programming, the conveyor may be moved to new positions between instructions. The robot will return to the correct position when forward or backward button is pressed.

The robot will recover as normal if the enabling device is released during motion.

The robot cannot perform coordinated motions to the conveyor while in manual reduced speed mode and the conveyor is moving.

Operation in automatic mode

Once a WaitWObj instruction has been executed, then it is no longer possible to step through the program with forward and backward hard buttons while the conveyor is moving.

Action	Description
Start and stop	The robot will stop and no longer track the conveyor if the stop button is pressed, or a Break or StopMove instruction is executed, between the WaitWObj and DropWObj instructions. The conveyor coordinated work object will not be lost but if the conveyor is moving then the object will quickly move out of the working area. Restart from the current instruction is not possible and the program must be restarted from the Main routine or with a WaitWObj instruction.
Emergency stop	When the emergency stop is pressed the robot will stop immediately. If the program was stopped and restarted after a WaitWObj then the coordinated work object will not be lost, but if the conveyor is moving then the object will quickly move out of the working area. It is not possible to restart from the current instruction and the program must be restarted either from the Main routine or with a WaitWObj instruction.

Operation in manual full speed mode (100%)

Operation in manual mode (100%) is similar to operation in automatic mode. The program can be run by pressing and holding the start button, but once a WaitWObj instruction has been executed then it is no longer possible to step through the program with the forward and backward buttons while the conveyor is moving.

Action	Description
Stop and restart	When the start button is released, or emergency stop is pressed, the robot will stop immediately. If the program was stopped after a WaitWObj then the coordinated work object will not be lost but if the conveyor is moving then the object will quickly move out of the working area. It is not possible to restart from the current instruction and the program must be restarted either from the Main routine or with a WaitWObj instruction.

7 System parameters

7.1 Introduction

About system parameters

This chapter describes system parameters used for all conveyor tracking options. Some parameters that are specific for only one option are presented in procedure context if they need to be changed.

All basic system parameters and the system parameter principles are listed in *Technical reference manual - System parameters*.

System parameters are modified using the **Configuration Editor** in RobotStudio.

7 System parameters

7.2 Topic Motion

7.2 Topic Motion

Type Mechanical Unit

The instance is named *CNV1*. See [Defining the mechanical unit parameters on page 75](#), and [Activating the conveyor on page 83](#).

Parameter	Description
Name	The name of the unit, usually <i>CNV1</i> , <i>CNV2</i> etc. This name is used in the Jogging window and from the program, for example when a unit should be activated.
Activate at Start-up	Defines if the conveyor should be activated automatically at start.  Note This parameter should not be used when combining synchronized and un-synchronized mode, see Combining synchronized and un-synchronized mode on page 177 .
Deactivation Forbidden	Defines if the conveyor is allowed to be deactivated.

Type Motion Planner

The instance is named *motion_planner*. See [Regulating CPU load and accuracy on page 75](#).

Parameter	Description
Path Resolution	The parameter corresponds in some sense to the distance between two points in the path. Increasing path resolution means increasing the distance, which leads to a decrease in the resolution of the path.
Process Update Time	Determines how often the process path information is calculated.

Type Robot

The instance is named *ROB_1*, *ROB_2* etc. See [Additional adjustments on page 75](#), and [Configuring a track motion to follow a conveyor on page 76](#).

Parameter	Description
Corvec correction level	Defines how often corrections of robot positions are done. Default is 1. Increasing the value gives corrections more often. Should be set to 2 or 3 to get good accuracy during conveyor acceleration. For IRB 360 with high payloads (6-8 kg), for YuMi robots, and for big robots like IRB 6700, it should be set to 1.
Track Conveyor With Robot	If Yes, then the robot will track the conveyor without using the track axis even if robot is coordinated with the track. Default value is <i>No</i> .

Continues on next page

Parameter	Description
Max accel for conveyor tracking	This parameter should only be used when conveyor speed is higher than 600 mm/s, when the robot has to pick a part on the conveyor, and when increase of max_external_pos_adjustment and Stop_ramp is not an option. Defines the max acceleration allowed for the robot during conveyor tracking. The aim of this parameter is to reduce adjustment in servo task as big adjustment values can cause speed supervision errors or max_external_pos adjustment error (50163). The drawback of too low acceleration is an increase of cycle time. Default value is 1 (limit not used), max value is 100 (m/s²).
Max External Pos Adjustment	Defines the maximum position adjustment allowed in the servo task. Can be increased for big robots with heavy load and high conveyor speed if error 50163 occurs. First verify that <i>Adjustment speed</i> and <i>Adjustment accel</i> are correctly defined (see Type Conveyor systems on page 96). Default value 0.2, maximum 0.8, minimum 0.1. Defined in meters. If increased, <i>Start ramp</i> and <i>Stop ramp</i> should also be increased to 20 or 30 (see Type Conveyor systems on page 96).
Enable orientation correction	The parameter <i>Enable orientation correction</i> is used to correct the orientation of a 5-axis delta robot, to make it possible to reach positions that are hard to program, for example, inside boxes or bins. Setting the value to Yes will allow the robot to use internal correction so that it can reach a pose, even if the programmed orientation cannot be reached with a 5-axis robot.

Type Single

The instance is named *CNV1*. See [Calibrating the conveyor base frame on page 67](#).

Parameter	Description
Base frame x Base frame y Base frame z	Defines the x, y, and z-direction of the base frame position in Base frame y relation to the world frame (in meters).
Base frame q1-q4	Defines the quaternions of the base frame orientation in relation to the world frame. That is, the orientation of the conveyor base coordinate system.

Type Single Type

The instance is named *CNV1*.

Parameter	Description
Mechanics	Defines what type of mechanics the single type uses.

Type Transmission

The instance is named *CNV1*.

Parameter	Description
Rotating Move	Defines if the conveyor is rotating (Yes) or linear (No).

7 System parameters

7.3 Topic Process

7.3 Topic Process

Type Conveyor systems

The instance is named *CNV1*. See [Avoiding robot reach problems on page 72](#), and [Defining the robot adjustment speed on page 74](#).

Parameter	Description
Adjustment speed	The speed (in mm/s) at which the robot should catch up to the conveyor. The general recommended value is 130 % of the conveyor speed. As minimum, the value should be more than 100 % with some margin. If the robots speed is very fast compared to the conveyor speed, a further increase of the value is often necessary. If the value is set too low, robot movements may become jerky or the conveyor tracking accuracy may become reduced. On the other hand, if the value is set too high, the drive system may become overloaded, causing motion supervision errors. Generally, the maximum recommended value is 200 %. For IRB360 in applications with high robot speed, the maximum recommended value is 500 %.
minimum distance	The minimum distance (in millimeters) that a connected object can have before being automatically dropped. If an object is dropped during coordinated motion, then the motion is stopped and an error is produced. Note that if the minimum distance is set to a positive value, then all objects are dropped as soon as they are connected.
maximum distance	The maximum distance (in millimeters) that a connected object can have before being automatically dropped. If an object is dropped during coordinated motion, then the motion is stopped and an error is generated.
Stop ramp	The number of samples used to ramp down the correction when tracking is stopped. The default value of this parameter is 5, and this is a suitable setting for an IRB 360 with payload in the range 0-1 kg. For an IRB 360 with payload 1-8 kg, the value of this parameter should be increased to avoid vibrations and overload of the mechanical structure. A value of at least 10 should be chosen for a robot with 8 kg payload. When tracking with a track axis this parameter should be increased to 30. For a robot switching between conveyors increasing the stop ramp will increase the distance needed to reach accurate tracking on next conveyor. For example, for an IRB 360 10 steps of 12 ms at 5 m/s this means 0.6 meters.
Start ramp	The number of samples used to ramp up the correction when tracking is started. The default value of this parameter is 5, and this is a suitable setting for an IRB 360 with payload in the range 0-1 kg. For an IRB 360 with payload 1-8 kg, this parameter should be increased to avoid vibrations and overload of the mechanical structure. A value of at least 10 should be chosen for a robot with 8 kg payload. When tracking with a high speed conveyor this parameter can be increased. During the ramping the accuracy of tracking is not reached so be careful when increasing this parameter. For an IRB 360 the best choice is to use the default value (5) if the load is not higher than 1 kg. When switching between 2 conveyors with short distance between pick and place the use of time interpolation is the optimal solution. For example, put the robot in a wait position just above the pick conveyor with a very short distance to pick position. In this case a low speed or time interpolation should also be used.

Continues on next page

Parameter	Description
Adjustment accel	The maximum acceleration (in mm/s²) at which robot should catch up to the conveyor. By default no limitation. For big robots and heavy load or limited robot acceleration (like use of AccSet or PathAccLim) it can be necessary to set <i>Adjustment accel</i> according to the robot performances. With robot on track or trolley tracking with the track the performance of the track is automatically used. This parameter must be adjusted when the robot cannot continue its path but remains tracking the same position on the conveyor. For big robots like IRB 6600 <i>Adjustment accel</i> should be set around 1000 if conveyor speed is higher than 150 mm/s.
Speed filter length	The number of samples used for average filter of conveyor speed. Maximum value is 50. Default value 1, which equals no filter. Should be used only in case of high level of noise on conveyor speed and speed reduction on robot due to this noise.
Acc dependent filter	Specifies the acceleration dependent filter. Default value is 1 m/s ² . To value get good accuracy during acceleration this value should be set equal to the maximum acceleration of the conveyor. A low value means harder filtering. If there is a problem with noise this parameter should be reduced. If an IRB 360 is used for fast picking this parameter should be set to 0, this will turn off the filtering to improve the response times.
syncfilter ratio	Defines how fast the robot should adjust the speed to the conveyor speed. Default value is 0.8. For IRB 360 this can be reduced to 0.5 to improve the accuracy in fast pick and place applications with low payload (0-1 kg). A too low value might result in jerky movements.

Type Conveyor Ici (DSQC2000)

Parameter	Description
CountsPerMeter	Gives the number of quadrature pulses per meter of motion of the conveyor. Should be in the range of 5000-10000 for linear conveyors.
QueueTrckDist	The queue tracking distance (meters) defines the placement of the 0.0 meter point relative to the synchronization switch on the conveyor. All objects in this distance are tracked. The position returned for the object will be negative, relative to the 0.0 m point. All objects in this distance are tracked, but connection is not allowed until an object has passed 0.0 meters.
StartWinWidth	This distance defines the start window (meters). The start window defines the area that if a program starts using an object within the window, then all program coordination can end before the maximum distance or work area is reached. All objects within this window are tracked and are eligible for use in a coordinated work object. A WaitWObj instruction will connect to the first object in the window.
PosUpdateTime	Defines how often the system reads the speed and position of the conveyor from the I/O system.

Type Conveyor Internal

The instance is named *INTERNAL1*.

Parameter	Description
Eio unit name	Name of the simulated I/O unit.
Connected signal	Name of the digital input signal for connection.
Position signal	Name of the analog input signal for conveyor position.

Continues on next page

7 System parameters

7.3 Topic Process

Continued

Parameter	Description
Velocity signal	Name of the analog input signal for conveyor speed.
Null_speed signal	Name of the digital input signal indicating zero speed on the conveyor.
DropWObj signal	Name of the digital output signal to drop a connected object.
ObjLost signal	Name of the digital input signal to indicate that an object has gone past the start window without being connected.
RemAllPObj sig	Name of the digital input signal to remove all Pobj.
Rem1PObj sig	Name of the digital input signal to remove one Pobj.
Pos Update time	Defines how often position and velocity output signals are updated.
Supervise max_dist Off	Boolean to remove supervision of maximum and minimum distance. Default value is YES.
New object strobe	Name of digital output signal showing a new object in queue.
Objects in queue	Name of group output signal showing the number of objects in queue.
Count1 from encoder	Name of group output for new object position low word.
Count2 from encoder	Name of group output for new object position high word.
Single to track	Name of the single that is moving the indexing conveyor. (<i>Indexing Conveyor Control.</i>)

8 RAPID reference

8.1 Instructions

8.1.1 WaitWObj - Wait for work object on conveyor

Description

WaitWObj (*Wait Work Object*) connects to a work object in the start window on the conveyor mechanical unit.

Example

```
WaitWObj wobj_on_cnv1;
```

The program connects to the first object in the object queue that is within the start window on the conveyor. If there is no object in the start window then execution stops and waits for an object.

Arguments

```
WaitWObj WObj [\RelDist][\MaxTime][\TimeFlag]
```

WObj

Work Object

Data type: wobjdata

The moving work object (coordinate system) to which the robot position in the instruction is related. The mechanical unit conveyor is to be specified by the ufmec in the work object.

[\RelDist]

Relative Distance

Data type: num

Waits for an object to enter the start window and go beyond the distance specified by the argument. If the work object is already connected, then execution stops until the object passes the distance. If the object has already gone past the relative distance then execution continues.

[\MaxTime]

Maximum Time

Data type: num

The maximum period of waiting time permitted, expressed in seconds. If this time runs out before the sensor connection or relative distance is reached, the error handler will be called, if there is one, with the error code ERR_WAIT_MAXTIME. If there is no error handler, the execution will be stopped.

[\TimeFlag]

Timeout Flag

Data type: bool

Continues on next page

8 RAPID reference

8.1.1 WaitWObj - Wait for work object on conveyor

Continued

The output parameter that contains the value TRUE if the maximum permitted waiting time runs out before the sensor connection or relative distance is reached. If this parameter is included in the instruction, it is not considered to be an error if the maximum time runs out. This argument is ignored if the MaxTime argument is not included in the instruction.

Program execution

If there is no object in the start window then program execution stops. If an object is present, then the work object is connected to the conveyor and execution continues.

If a second WaitWObj instruction is issued while connected then an error is returned unless the \RelDist optional argument is used.

More examples

Example 1

```
WaitWObj wobj_on_cnv1\RelDist:=500.0;
```

If not connected, then wait for the object to enter the start window and then wait for the object to pass the 500 mm point on the conveyor. If already connected to the object, then wait for the object to pass 500 mm.

Example 2

```
WaitWObj wobj_on_cnv1\RelDist:=0.0;
```

If not connected, then wait for an object in the start window. If already connected, then continue execution as the object has already gone past 0.0 mm.

Example 3

```
WaitWObj wobj_on_cnv1;  
WaitWObj wobj_on_cnv1\RelDist:=0.0;
```

The first WaitWObj connects to the object in the start window. The second WaitWObj will return immediately if the object is still connected, but will wait for the next object if the previous object had moved past the maximum distance or was dropped.

Example 4

```
WaitWObj wobj_on_cnv1\RelDist:=500.0\MaxTime:=0.1\Timeflag:=flag1;
```

The WaitWObj will return immediately if the object has passed 500 mm but otherwise will wait 0.1 seconds for an object. If no object passes 500 mm during this 0.1 seconds the instruction will return with flag1=TRUE.

Limitations

20 is required to connect to the first object in the start window. Once connected, a second WaitWObj instruction with \RelDist optional argument will take only normal RAPID instruction execution time.

Error handling

If following errors occur during execution of the WaitWObj instruction, the system variable ERRNO will be set. These errors can then be handled in the error handler.

ERR_CNV_NOT_ACT	The conveyor is not activated.
-----------------	--------------------------------

Continues on next page

8.1.1 WaitWObj - Wait for work object on conveyor

Continued

ERR_CNV_CONNECT	The WaitWObj instruction is already connected.
ERR_CNV_DROPPED	The object that the instruction WaitWObj was waiting for has been dropped by another task.
ERR_WAIT_MAXTIME	The object did not come in time and there is no Timeflag.
ERR_CNV_OBJ_LOST	The object has passed the StartwindowWidth without being connected.

Syntax

```
WaitWObj
[ WObj ':=' ] < persistent (PERS) of wobjdata > ';'
[ '\ ' RelDist ':=' < expression (IN) of num > ]
[ '\ ' MaxTime ':=' < expression (IN) of num > ]
[ '\ ' TimeFlag ':=' < variable (VAR) of bool > ] ';'

```

Related information

For information about	See
DropWObj	DropWObj - Drop work object on conveyor on page 102
The data types wobjdata, num, and bool	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>

8 RAPID reference

8.1.2 DropWObj - Drop work object on conveyor

8.1.2 DropWObj - Drop work object on conveyor

Description

DropWObj (*Drop Work Object*) is used to disconnect from the current object and the program is ready for the next.

Example

```
MoveL *, v1000, z10, tool, \WObj:=wobj_on_cnv1;  
MoveL *, v1000, fine, tool, \WObj:=wobj0;  
DropWObj wobj_on_cnv1;  
MoveL *, v1000, z10, tool, \WObj:=wobj0;
```

Arguments

DropWObj WObj

WObj

Work Object

Data type: wobjdata

The moving work object (coordinate system) to which the robot position in the instruction is related. The mechanical unit conveyor is to be specified by the ufmec in the work object.

Program execution

Dropping the work object means that the object is not longer tracked. The object is removed from the object queue and cannot be recovered.

Limitations

If the instruction is issued while the robot is actively using the conveyor coordinated work object then the motion stops.

The instruction can be issued only after a fixed work object has been used in the preceding motion instructions with either a fine point or several (>1) corner zones.

Syntax

```
DropWObj  
[ WObj ':='] < persistent (PERS) of wobjdata> ';' ;
```

Related information

For information about	See
WaitWObj	WaitWObj - Wait for work object on conveyor on page 99
The data type wobjdata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>

8.1.3 CnvPredictReach - High speed conveyors

Description

This section describes the module *CnvPredictReach* that is used to predict if the picking and placing positions are within the work area of the delta robot, when using high speed conveyors (speeds higher than 800 mm/s). This module can only be used for simple and repeatable picking and placing programs, it cannot be used with buffer or stop and go conveyors.

Example: Initialization of data structure

```

RECORD cnvpickeroptimdata
  bool Tpumessage;
  num firststeppredictionratio;
  num maxfirststeppredictiontime;
  num secondsteppredictionratio;
  num maxsecondsteppredictiontime;
  num maxfirstcyclepredictiontime;
  bool wait_end_of_pp_movement;
  string pickconveyor;
  string placeconveyor;
ENDRECORD

```

Default value

```

TASK PERS cnvpickeroptimdata defaultcnvpickeroptimdata
:= [FALSE, 0.35, 0.6, 0.3, 0.75, 0.6, TRUE, "CNV1", "CNV2"];

```

Example: Picking and placing

Initialization phase

```

InitCnvPredictReach
  Safepos, resetttimearea, 1, defaultcnvpickeroptimdata;

```

In the picking program example from Pickmaster you will need to add a call to **checkpick** and **trigg** to **pickenter** and **pickexit**.

```

PROC Pick(num Index)
  IF Index > 0 THEN
    WObjPick:=ItmSrcData{Index}.Wobj;
    GetItmTgt ItmSrcData{Index}.ItemSource, PickTarget;
    CheckPick PickTarget.RobTgt, PickAct1, WObjPick;
    TriggL\Conc, RelTool(PickTarget.RobTgt, 0, 0, -ItmSrcData{Index}.OffsZ)
    , MaxSpeed, ItmSrcData{Index}.VacuumAct1\T2:=pickenter, z20, PickAct1\WObj:=WObjPick;
    TriggL\Conc, PickTarget.RobTgt, LowSpeed, ItmSrcData{Index}.SimAttach1, z5\
    Inpos:=ItmSrcData{Index}.TrackPoint, PickAct1\WObj:=WObjPick;
    GripLoad ItemLoad;
    TriggL
      RelTool(PickTarget.RobTgt, 0, 0, -ItmSrcData{Index}.OffsZ), LowSpeed,
      ItmSrcData{Index}.Ack\T2:=pickexit, z20, PickAct1\WObj:=WObjPick;
    AccSet 100, 100;
  ELSE
    ErrWrite "Missing item distribution", "Cannot pick because no item
      distribution contains current work area." \RL2:="Please check
      configuration";
  ENDIF
ENDPROC

```

Continues on next page

8 RAPID reference

8.1.3 CnvPredictReach - High speed conveyors

Continued

```
SafeStop;
ENDIF
ERROR
AckItmTgt ItmSrcData[Index].ItemSource,PickTarget,FALSE\Skip:=TRUE;
GetItmTgt ItmSrcData[Index].ItemSource,PickTarget;
RETRY;
ENDPROC
```

Example: Double picking or single placing

Initialization phase

```
InitCnvPredictReach
  Safepos, resetttimearea,1,defaultcnvpickeroptimdata;
```

For example when using the Pickmaster template, for double picking and single placing.

```
LOCAL CONST string aiCnv1PosSigName:="c1Position";
LOCAL CONST string aiCnv2PosSigName:="c2Position";
LOCAL CONST string aiCnv3PosSigName:="c3Position";
LOCAL CONST string aiCnv4PosSigName:="c4Position";
LOCAL CONST string aiCnv5PosSigName:="c5Position";
LOCAL CONST string aiCnv6PosSigName:="c6Position";
LOCAL CONST string aiCnv1SpeedSigName:="c1Speed";
LOCAL CONST string aiCnv2SpeedSigName:="c2Speed";
LOCAL CONST string aiCnv3SpeedSigName:="c3Speed";
LOCAL CONST string aiCnv4SpeedSigName:="c4Speed";
LOCAL CONST string aiCnv5SpeedSigName:="c5Speed";
LOCAL CONST string aiCnv6SpeedSigName:="c6Speed";
```

```
LOCAL VAR signalai aiCnv1Position;
LOCAL VAR signalai aiCnv2Position;
LOCAL VAR signalai aiCnv3Position;
LOCAL VAR signalai aiCnv4Position;
LOCAL VAR signalai aiCnv5Position;
LOCAL VAR signalai aiCnv6Position;
LOCAL VAR signalai aiCnv1Speed;
LOCAL VAR signalai aiCnv2Speed;
LOCAL VAR signalai aiCnv3Speed;
LOCAL VAR signalai aiCnv4Speed;
LOCAL VAR signalai aiCnv5Speed;
LOCAL VAR signalai aiCnv6Speed;
```

```
PROC EnumerateWorkAreas()
VAR num PickNumber:=1;
VAR num PlaceNumber:=1;
VAR num OtherNumber:=1;
```

```
InitCnvPredictReach
  Safepos, resetttimearea,1,defaultcnvpickeroptimdata;
```

```
FOR i FROM 1 TO MaxNoSources DO
IF (ItmSrcData[i].Used) THEN
```

Continues on next page

8.1.3 CnvPredictReach - High speed conveyors

Continued

```

IF (ItmSrcData{i}.SourceType = PICK_TYPE) THEN
PickWorkArea{PickNumber}:=i;
WObjPick:=ItmSrcData{i}.Wobj;
IF use_3areas THEN
InitPickArea 1,3,PickAct1 ,WObjPick, SafePos, FALSE;
ELSE
InitPickArea 1,2,PickAct1 ,WObjPick, SafePos, TRUE;
ENDIF
Incr PickNumber;
ELSEIF (ItmSrcData{i}.SourceType = PLACE_TYPE) THEN
PlaceWorkArea{PlaceNumber}:=i;
WObjPlace:=ItmSrcData{i}.Wobj;
InitPickArea 2,1,PickAct1 ,WObjPlace, SafePos, FALSE;
IF use_3areas THEN
InitPickArea 3,2,PickAct1 ,WObjPlace, SafePos, FALSE;
ENDIF
Incr PlaceNumber;
ELSE
OtherWorkArea{OtherNumber}:=i;
Incr OtherNumber;
ENDIF
InitIoaliasfromitmsrc i;
ENDIF
ENDFOR
ENDPROC

```

InitIoaliasfromitmsrc is simple alias definition for QuickAck function

```

PROC InitIoaliasfromitmsrc(num Index)
IF ItmSrcData{Index}.Wobj.ufmec = "CNV1" THEN
AliasIO aiCnv1PosSigName, aiCnv1Position;
AliasIO aiCnv1SpeedSigName, aiCnv1Speed;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV2" THEN
AliasIO aiCnv2PosSigName, aiCnv2Position;
AliasIO aiCnv2SpeedSigName, aiCnv2Speed;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV3" THEN
AliasIO aiCnv3PosSigName, aiCnv3Position;
AliasIO aiCnv3SpeedSigName, aiCnv3Speed;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV4" THEN
AliasIO aiCnv4PosSigName, aiCnv4Position;
AliasIO aiCnv4SpeedSigName, aiCnv4Speed;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV5" THEN
AliasIO aiCnv5PosSigName, aiCnv5Position;
AliasIO aiCnv5SpeedSigName, aiCnv5Speed;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV6" THEN
AliasIO aiCnv6PosSigName, aiCnv6Position;
AliasIO aiCnv6SpeedSigName, aiCnv6Speed;
ENDIF
ENDPROC

```

! *****

Continues on next page

8 RAPID reference

8.1.3 CnvPredictReach - High speed conveyors

Continued

```
!
! Function IsQuickAckOk
!
! Checks if it's OK to make a fast acknowledge.
!
! *****
FUNC bool IsQuickAckOk(num Index)
VAR num CnvPos:=0;
VAR num DropDist:=0;
VAR num RealPos:=0;
VAR bool QuickAckOk:=FALSE;
VAR num SafeDist;

IF ItmSrcData{Index}.Wobj.ufmec = "" THEN
QuickAckOk:=TRUE;
ELSE
SafeDist:=200;
IF ItmSrcData{Index}.Wobj.ufmec = "CNV1" THEN
CnvPos:=aiCnv1Position*1000;
DropDist:=Cnv1DropDist;
SafeDist:=aiCnv1Speed*1000*0.35;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV2" THEN
CnvPos:=aiCnv2Position*1000;
DropDist:=Cnv2DropDist;
SafeDist:=aiCnv2Speed*1000*0.35;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV3" THEN
CnvPos:=aiCnv3Position*1000;
DropDist:=Cnv3DropDist;
SafeDist:=aiCnv3Speed*1000*0.35;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV4" THEN
CnvPos:=aiCnv4Position*1000;
DropDist:=Cnv4DropDist;
SafeDist:=aiCnv4Speed*1000*0.35;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV5" THEN
CnvPos:=aiCnv5Position*1000;
DropDist:=Cnv5DropDist;
ELSEIF ItmSrcData{Index}.Wobj.ufmec = "CNV6" THEN
CnvPos:=aiCnv6Position*1000;
DropDist:=Cnv6DropDist;
ENDIF
IF CnvPos < DropDist-SafeDist THEN
QuickAckOk:=TRUE;
ELSE
QuickAckOk:=FALSE;
ENDIF
ENDIF
RETURN QuickAckOk;
ENDFUNC

! *****
!
```

Continues on next page

```

! Function QuickAck
!
! Makes a quick acknowledge. Returns FALSE if is
! wasn't possible.
!
! Example:
! PROC Place(num Index)
! Coordinated;
! WObjPlace:=ItmSrcData{Index}.Wobj;
! GetItmTgt ItmSrcData{Index}.ItemSource,PlaceTarget;
! MoveL\Conc,RelTool(PlaceTarget.RobTgt,0,0,-ItmSrcData{Index}
.OffsZ),MaxSpeed,z20,Gripper\WObj:=WObjPlace;
!
      TriggL\Conc,PlaceTarget.RobTgt,LowSpeed,ItmSrcData{Index}.VacuumRev1\T2:=
ItmSrcData{Index}.VacuumOff1,z5\Inpos:=ItmSrcData{Index}.TrackPoint,Gripper\WObj:=WObjPlace;
! GripLoad load0;
! IF QuickAck(Index, PlaceTarget) THEN
! MoveL
      RelTool(PlaceTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),LowSpeed,
z20,Gripper\WObj:=WObjPlace;
! ELSE
! TriggL RelTool(PlaceTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ)
,LowSpeed,ItmSrcData{Index}.Ack,z20,Gripper\WObj:=WObjPlace;
! ENDIF
! UnCoordinated;
! ENDPROC
!
!
! *****
FUNC bool QuickAck(num Index, itmTgt ItemTarget \switch Nack)
VAR bool QuickAckOk:=FALSE;
VAR bool AckNack:=TRUE;

IF Present(Nack) THEN
AckNack:=FALSE;
ENDIF
IF IsQuickAckOk(Index) THEN
AckItmTgt ItmSrcData{Index}.ItemSource,ItemTarget,AckNack;
QuickAckOk:=TRUE;
ENDIF
RETURN QuickAckOk;
ENDFUNC

```

**Note**

For CTM (DSQC2000 only), the signal definitions `signalai` must be replaced by `signalao` since the signals need to be configured as analog output.

This page is intentionally left blank

9 Advanced queue tracking

9.1 Introduction to advanced queue tracking

Introduction

When queue tracking mode is enabled in the conveyor I/O configuration, the job queue is external, which means it can be handled by the RAPID code. If the queue tracking mode is disabled, then the queue is handled internally.

System parameters

The following signals are used for advanced queue tracking and must be defined in the system parameters, topic *I/O*, type *Signal*.

I/O signal	Description
c1ObjectsInQ	Group input showing the number of objects in the object queue. These objects have passed the synchronization switch but have not gone outside the start window.
c1Rem1PObj	Remove first pending object from the object queue. Setting this signal will cause the first pending object to be dropped from the object queue. Pending objects are objects that are in the queue but are not connected to a work object.  Note It takes some time for the signal to apply. After setting the signal, it is therefore advisable to wait for 0.15 s before a new work object is connected.
c1RemAllPObj	Remove all pending objects. Setting this signal will empty all objects from the object queue. If an object is connected, then it is not removed.  Note It takes some time for the signal to apply. After setting the signal, it is therefore advisable to wait for 0.15 s before a new work object is connected.
c1DropWObj	Setting this signal will drop the tracked object and disconnect that object. The object is removed from the queue. This should not be set from RAPID, use the DropWobj instruction instead.
c1NewObjStrobe	This DO signal is pulsed when a new object is detected.
c1CntFromEnc	This 32-bit GI signal indicates the location (counter value) of the latest detected object. The signal is updated at every object detection. The signal can only detect new values when the conveyor is moving. To create a new object when the conveyor is not moving it is required to pulse <i>cxNewObjStrobe</i>. A new object will be created at the current conveyor position.  Note The 32-bit signal is now default, but the old 16-bit signals are still supported: c1CntFromEnc1 (low word) and c1CntFromEnc2 (high word).

Continues on next page

9 Advanced queue tracking

9.1 Introduction to advanced queue tracking

Continued

I/O signal	Description
c1CntToEncStr	Used in queue tracking mode. When this DO signal is pulsed, the object location that is specified by <i>c1CntToEnc</i> will be moved to the first position of the object queue. As a result, the next work object to be tracked will target this object location.
c1CntToEnc	Used in queue tracking mode. This 32-bit GO signal specifies a desired object location (counter value) that shall be tracked. For DSQC2000, the use of <i>cxCntToEnc</i> is not meant to add objects to the queue but only to select objects to be picked from the queue. Therefore, <i>cxObjectsInQ</i> is not incremented after a pulse of <i>cxCntToEncStr</i>. If the counter value set using <i>cxCntToEnc</i> does not match any existing object in the queue, the controller will connect to the first available object issuing the internal error <i>Invalid count_to_enc_strobe on cnv x</i>. Therefore, creating new objects at arbitrary counter positions is not supported. On the other hand, if the counter value selected using <i>cxCntToEnc</i> is valid and available for connection, the controller will automatically drop all the objects from the queue which are older than the selected object. To avoid automatic dropping, it is recommended to select the oldest object for connection and perform the pick/place operations using the relative offset to the connected object.
c1PosInJobQ	For DSQC2000, this signal has no effect.
c1EncSelec	• 0 = Encoder A selected • 1 = Encoder B selected (DSQC377 only)
c1SoftSync	DO signal that can be used to simulate the detection of a new virtual object. When the signal is pulsed, a new object with the current location (counter value) will enter the queue.
c1PassStw	DO indicating that an object has passed out of the start window without being connected (object lost). If the RAPID program is waiting in a <i>WaitWobj</i> instruction, the program pointer will be moved to the nearest error handler.

RAPID

If two RAPID instructions *WaitWobj* and *DropWobj* needs to be executed subsequently (without any other instruction in between), it is recommended to use *WaitTime* of at least 10 ms before *DropWobj*, to allow the correct internal state change of the object queue inside the controller.

9.2 Working with the object queue

Signal values

The option *Conveyor Tracking* provides several I/O signals that allow a user or RAPID program to monitor and control the object queue. The table in [System parameters on page 109](#), shows the I/O signals that impact the object queue. The counter values have to do with the queue tracking function. Positions detected on the encoder node are sent to the main computer to be stored in the job queue handled by the robot controller. Values are returned to the encoder when object is ready to be tracked.

Handling the object queue in RAPID

To handle the object queue in RAPID the program must store the counter value for each new object on the conveyor and write it to the conveyor board when the user wants to track this object.

The RAPID program needs the following elements:

```
SetDO c1PosInJobQ, 1;

! Connect a trap routine to the detection of new objects on the
  conveyor
CONNECT NewObj WITH NewObjOnConvey;
ISignalGI c1CntFromEnc, NewObj;

TRAP NewObjOnConvey
  ! A new object is detected; Read its position from input group
  signal
  ObjectPosition := GInputDnum(c1CntFromEnc);
  RETURN;
ENDTRAP

TRAP TrackNewObj
  ! To track a selected object, write its reference to output group
  signal
  SetGO c1CntToEnc, ObjectPosition;
  WaitTime 0.02;
  ! Activate the written reference
  PulseDO c1CntToEncStr;
  RETURN;
ENDTRAP
```

Passed start window signal

In some applications it is important to know if an object has gone through the start window without being connected. The encoder interface allows the robot controller software to detect when an object has passed the start window without being connected and is thus lost. The detection of the lost object is done on the next `WaitWObj` instruction. The next `WaitWObj` instruction, following after an object has moved outside the start window, will return with the error `ERR_CNV_OBJ_LOST`. This error can be handled in the RAPID error handler.

Continues on next page

9 Advanced queue tracking

9.2 Working with the object queue

Continued

A DO signal, *c1PassStw*, can be selected to indicate if an unconnected object is passing the start window. Go to system parameters, *Process* -> *Conveyor Ici* -> *ObjLost* signal and set it to *c1PassStw*. This signal will go high when an object has left the start window without being connected.



Note

For CTM (DSQC2000) the ERR_CNV_OBJ_LOST error is not supported. The object that has moved outside the start window without connection will be dropped and *waitwObj* will connect to the next object in the queue. If enabled, the signal *ObjLost* will be pulsed.

Simulation mode

The simulated encoder starts when the simulation signal is set. The simulation encoder counts is set to the real encoder counts when the simulation signal is set. If the simulation signal is reset the encoder value returns to real encoder position. The simulation speed is defined with the parameter *SimulationVel* in the type *Signal*, in the topic *I/O*.

10 Circular conveyor tracking

10.1 Introduction to circular conveyor tracking

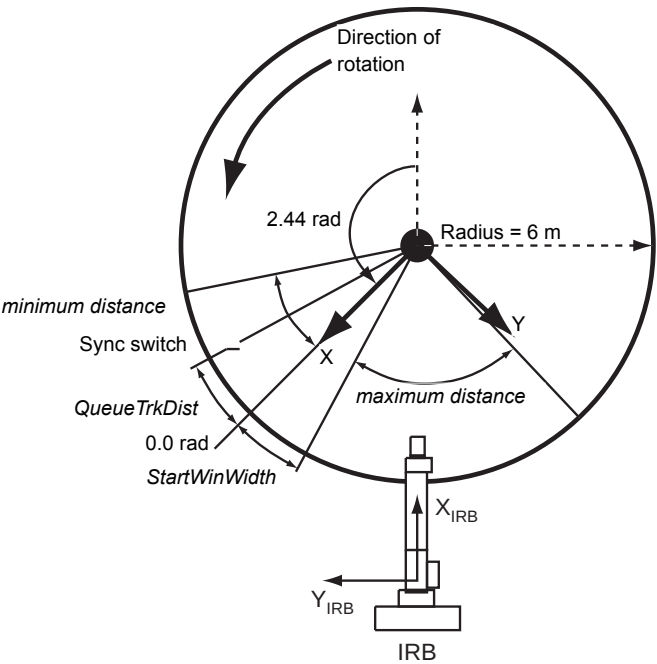
Introduction

Circular conveyors can be tracked with the option *Conveyor Tracking*. The principle for configuring circular conveyor tracking is to define values in radians instead of meters. Then configure as described in [Installation on page 39](#), and [Configuration and calibration on page 57](#).

This chapter will where there are differences in the configuration compared to linear conveyors.

Example

The figure below shows an example of circular conveyor tracking with example units and distances.



xx1200001101

CountsPerMeter	40000 counts per radian At 6 m radius, one count = 0.15 mm
minimum distance	-100 milliradians At 6 m radius, = -600 mm
Conveyor base frame	Base frame x = 8.0 m Base frame y = 0.0 m Base frame z = 0.0 m

Continues on next page

10 Circular conveyor tracking

10.1 Introduction to circular conveyor tracking

Continued

The x-axis is rotated 2.44 rad from the world X (X_{IRB})	Base frame q1 = 0.3420 Base frame q2 = 0.0000 Base frame q3 = 0.0000 Base frame q4 = 0.9397
<i>SyncSeparation</i>	0.005 rad At 6 m radius = 30 mm
<i>QueueTrkDist</i>	0.017 rad At 6 m radius = 100 mm
<i>maximum distance</i>	420 milliradians At 6 m radius = 2520 mm
<i>StartWinWidth</i>	0.017 rad At 6 m radius = 100 mm

10.2 Encoder type selection and location

Prerequisites

The goal in selecting an encoder for circular conveyor tracking is to have 0.1 mm to 0.2 mm resolution per count at the maximum radius of conveyor tracking.

Example

In the [Example on page 113](#), following at a 6 meter radius in order to have 0.15 mm per count, we must have 40,000 counts per radian at the center of the table. The counts are quadrature encoded (four counts per pulse), thus the encoder must give 10,000 pulses per radian of circular conveyor movement. For a full revolution there are 2π radians per revolution, giving a requirement for $10000 \times 2\pi = 62831.85$ pulses per revolution of the circular conveyor.

Selecting gear ratio

If an encoder with 1000 pulses per revolution is selected, then we require a gear ratio of 1 to 62.83185 between the circular conveyor and the encoder shaft.



Note

The maximum value for *CountsPerMeter* in the encoder software is 50000. This should be considered when selecting gearing and encoder.

10 Circular conveyor tracking

10.3 Installation and configuration

10.3 Installation and configuration

Software installation

The conveyor work area and conveyor tracking software are connected and installed in the same way as for linear conveyors.

Defining CountsPerMeter

The value of the parameter *CountsPerMeter* should be known from the selection of the encoder and the gear ratio between the circular conveyor and the encoder shaft. If the value is not known, then it is possible to measure the value following the same steps as outlined for a linear conveyor with extra equipment for measuring the change in angle of the conveyor between *position_1*, and *position_2*.

Defining the queue tracking distance

Before proceeding with conveyor setup and calibration it is necessary to define the desired queue tracking distance (*QueueTrckDist*). The queue tracking distance establishes the distance between the synchronization switch and the 0.0 rad point on the circular conveyor. The conveyor work area will keep track of all objects that have passed the synchronization switch but have not yet passed the 0.0 rad point.

Related information

[Calibrating CountsPerMeter on page 66](#)

10.4 Additional motion settings

Conveyor start window and sync separation

For circular conveyor tracking these distances are defined in radians.

Conveyor maximum and minimum distances



Note

For circular conveyor tracking these distances are defined in milliradians.

Conveyor adjustment speed

The same as for linear conveyors.

Motion System parameters

The same as for linear conveyors.

Mechanical Unit parameters

The same as for linear conveyors.

Transmission and Single Type parameters

The motion configuration of the conveyor must be adjusted to account for a circular motion of the conveyor. There are two parameters that must be adjusted.

Parameter	Type	Description
Rotating Move	Transmission	Defines if the conveyor is rotating (<i>Yes</i>) or linear (<i>No</i>).
Mechanics	Single Type	Defines the mechanical structure of the conveyor. Select <i>EXT_ROT</i> .

10.5 Calibrating the conveyor base frame

Calibration options

The accuracy of the circular conveyor tracking depends on the accuracy in specifying the conveyor base frame. There are two methods to calibrate the base frame for a circular conveyor:

- 1 Enter the orientation and position of the base frame based on drawings of the robot installation and simple TCP measurements.
- 2 Use the robot TCP as a measuring tool and measure several points along the conveyor with some trigonometric calculations to calculate the conveyor base frame position and quaternion.

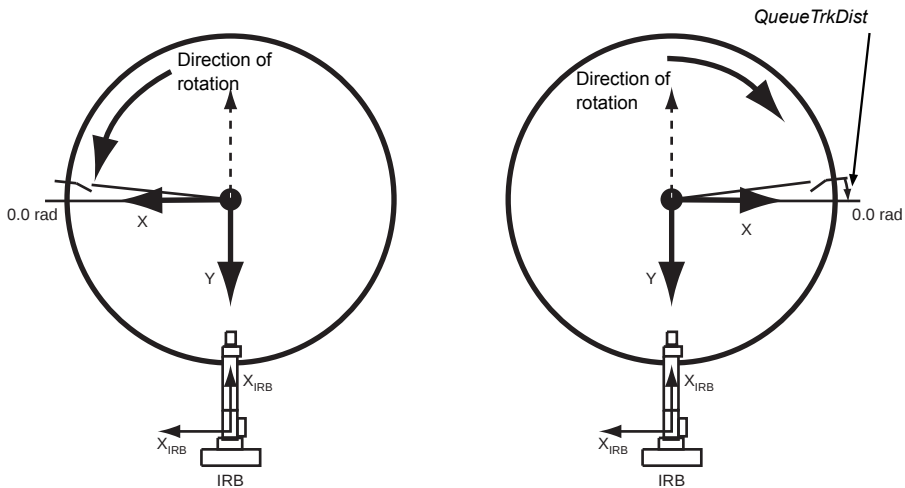
We recommend using the second method, with TCP as measuring tool. See [TCP measurement method on page 120](#).

10.6 Manually calibrating the conveyor base frame

Orientation

The definition of the quaternion for conveyor orientation will also define the location of the 0.0 radian point on the circular conveyor. The direction of the x-axis will define the 0.0 radian point while the direction of the z-axis will define the direction of positive rotation using the right-hand-rule.

The graphic below shows two installations, one with clockwise rotation and the other with counterclockwise rotation and the corresponding quaternions. In cases where the 0.0 rad point is not an even multiple of 90° from the world frame, calculation of the conveyor orientation quaternion must be done using manual calculations of the quaternion. The TCP can be used to help make measurements, see [TCP measurement method on page 120](#).



xx1200001102

Quaternion 0.7071, 0, 0, 0.7071	Quaternion 0, 0.7071, -0.7071, 0
---------------------------------	----------------------------------

Base frame position and start window start calibration

The conveyor base frame x, y, and z position must be specified relative to the world frame. This position must be calculated from the installation drawings or by using the robot as a measuring tool. Using the robot, one point can be marked on the edge of the circular conveyor and the TCP position is recorded for several points and the center point of the circle can be found. This is described in detail in the following section.

10.7 TCP measurement method

Recommendation

This section describes how to use TCP measurements and RAPID programs to calculate the conveyor base frame position and quaternion for a circular conveyor. This method uses three measured points on the circular conveyor to calculate the center of rotation. The three points should be spaced as far apart as possible around the periphery.

Calculating the x and y positions for the base frame

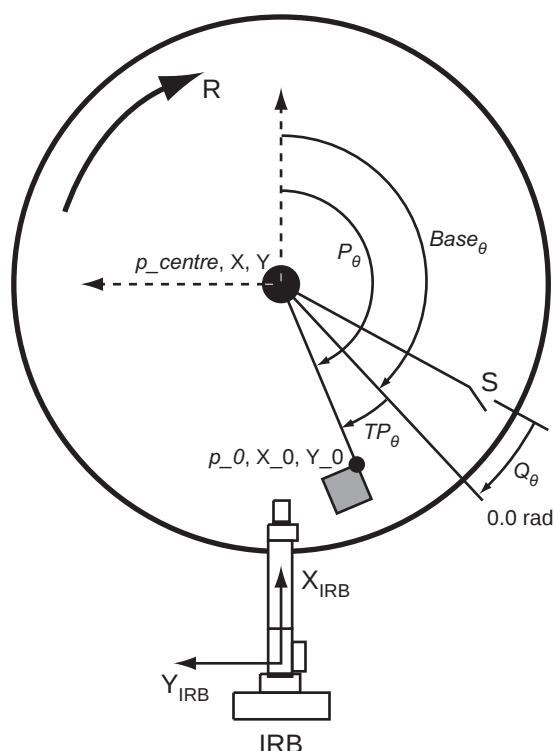
Use this procedure to calculate the x and y positions for the base frame.

- 1 Use Wobj0 on the FlexPendant. Pick out a reference point on the circular conveyor, jog the TCP to this point and record p_0 .
- 2 Run the conveyor to another position. Jog the TCP to the reference point and record p_1 .
- 3 Run the conveyor to a third position, jog the TCP to the reference point and record p_2 .
- 4 Use the function CNVUTL_cirCntr with the points p_0 , p_1 , and p_2 , to calculate the center of the circle, p_centre .
The system module cnv_utl can be found in Robotware.
- 5 Take the x and y values from p_centre and enter them into the base frame values for the conveyor, converting to meters, see [Topic Motion on page 94](#). These are shown in [Orientation on page 119](#). The z value will be entered later, once the work object zero position has been chosen.

Defining the base frame orientation and start window start calibration

The base frame quaternion defines where the 0.0 rad point is for the robot motion. The following figure shows an example of the angles that are used when defining the base frame orientation for the circular conveyor.

Continues on next page



xx1200001103

R	Direction of rotation
S	Synchronization switch
Q_θ	Queue tracking distance angle
TP_θ	Angle shown on FlexPendant
P_θ	Angle calculated from p_0 position
$Base_\theta$	Base frame angle to be converted to a quaternion

Calculating the quaternion

Use this procedure to calculate the quaternion for the base frame orientation.

- 1 Define a temporary conveyor base frame quaternion as 1, 0, 0, 0.
- 2 Define a conveyor coordinated work object, *wobjcnv1*.
- 3 Step forward through a RAPID program containing the two instructions:

```
ActUnit CNV1;
WaitWObj wobjcnv1;
```
- 4 Run the conveyor until an object passes through the sync switch and beyond the queue tracking distance. The *WaitWObj* instruction will end execution. Stop the conveyor.
- 5 Using *wobjcnv1*, move the robot TCP to the desired zero position on the work object and record this point, p_0 . Write down the X_0 , Y_0 , and Z_0 coordinates of the point p_0 as shown on the FlexPendant (*wobjcnv1* must be selected as work object).

Continues on next page

10 Circular conveyor tracking

10.7 TCP measurement method

Continued

- 6 Write down the angle shown in the **Jogging** window for the *CNV1* conveyor.

This is angle TP_θ , see example measurement points in [Defining the base frame orientation and start window start calibration on page 120](#).

- 7 Calculate P_θ from the X_0 and Y_0 coordinates of p_0 and the atan function. X_0 and Y_0 should both be positive when using the atan function. Check the value, it may be necessary to add 90 degrees:

$$P_\theta = \text{atan}\left(\frac{|Y_0|}{|X_0|}\right)$$

- 8 Calculate the value of Base.

$$Base_\theta = P_\theta - TP_\theta$$

- 9 Calculate the quaternion for the base frame taking into account the direction of rotation:

Counter clockwise rotation:

$$q1 = \cos(Base_\theta / 2)$$

$$q2 = 0.0$$

$$q3 = 0.0$$

$$q4 = \sin(Base_\theta / 2)$$

Clockwise rotation:

$$q1 = 0.0$$

$$q2 = \cos(Base_\theta / 2)$$

$$q3 = -\sin(Base_\theta / 2)$$

$$q4 = 0.0$$

- 10 Enter the value for z (in meters) from p_0 , and the values for the quaternions, $q1$, $q2$, $q3$, and $q4$, into the base frame for the conveyor, see [Topic Motion on page 94](#).

11 Accelerating conveyors

11.1 Introduction to accelerating conveyors

Description

This section describes how to optimize the tracking performance of accelerating and decelerating conveyors for the option *Conveyor Tracking*. This might be needed for example if good accuracy is needed during start and stop of the conveyor. To get good accuracy during tracking of accelerating conveyors it is important that all system parameters in the system are defined correctly. This chapter describes the parameters that are important for accelerating and decelerating conveyors. See [System parameters on page 124](#).

To further improve the accuracy it is possible to predict the speed change of the conveyor. This is done using a special RAPID function together with an I/O signal that is set just before the acceleration starts. See [Predicting speed changes on page 125](#).

For indexing conveyors, see [Indexing conveyors on page 129](#).

Prerequisites

The conveyor and encoder must be set up and calibrated correctly, see [Configuration and calibration on page 57](#).

11 Accelerating conveyors

11.2 System parameters

11.2 System parameters

Defining the system parameters

Use this procedure to define parameters that are important for an accelerating conveyor.

- 1 Set the speed filters.
See [Speed filters on page 124](#).
- 2 Change update rate of robot positions. See [Update rate of robot path on page 124](#).
- 3 Verify the performance of the system.

Speed filters

The filter parameters need to be changed. For both the filters there is a trade off between noise reduction and accuracy during acceleration. To get good accuracy during acceleration the filter values should be set according to the recommendations below. If there is too much noise in the system this might lead to disturbances in the robot movement and then the filter parameters should be decreased until this disturbances disappear.

Parameter	Description
Acc dependent filter Type <i>Conveyor Systems</i>	Specifies the setting of the acceleration dependent filter. Default value is 1 m/s ² . To get good accuracy during acceleration set it equal to the maximum acceleration of the conveyor. A low value gives harder filtering. If there is a problem with noise the value should be kept low. If IRB 360 is used for fast picking with low payload, this parameter should be set to 0. This will turn off the filtering to improve the response times.

The encoder speed filter should be set, see [Configuring an encoder on page 52](#).

Update rate of robot path

It is possible to define how often the robot path should be updated due to new conveyor position.

Parameter	Description
Corvec correction level Type <i>Robot</i>	Defines how often corrections of robot path shall be done. Default is 1. Should be set to 2 or 3 in order to get good accuracy during acceleration. Set to 1 if the prediction of speed changes functionality is used, see Predicting speed changes on page 125 . For IRB 360 with high payloads (6-8 kg), and for big robots like IRB 6600, it should be set to 1. Increasing it for big robots can lead to jerky movements.

11.3 Predicting speed changes

Introduction

It is possible to predict the speed change of a conveyor and use this prediction to improve the accuracy during tracking of an accelerating conveyor. The prediction is based on constant acceleration.

The prediction is setup from the RAPID instruction *UseAccProfile* and activated from an I/O signal. It is possible to have two independent profiles defined at the same time connected to two separate I/O signals. One could be used for starting and one for stopping the conveyor.

To access this RAPID instruction load the system module *Indexing_cnv* from the conveyor tracking option directory.

Setting up the signals

One or two I/O signals must be defined (one for each profile). These signals activates the prediction and should be set a predefined time before the speed change occurs.

First define the digital input I/O signals, see *Technical reference manual - System parameters*.

The names of the signals are used in the parameters *Sensor start signal* and *Sensor stop signal*. These parameters belong to the type *Conveyor systems* in the topic *Process*.

Parameter	Description
Sensor start signal	Name of the digital input signal to synchronize the prediction and the speed change. The signal must be set a predefined time before the speed change of the conveyor. How far ahead the signal should be set is configured in the RAPID instruction <i>UseAccProfile</i> .
Sensor stop signal	Name of the digital input signal to synchronize the prediction and the speed change. The signal must be set a predefined time before the speed change of the conveyor. How far ahead the signal should be set is configured in the RAPID instruction <i>UseAccProfile</i> .

The system parameters that affect the accuracy during acceleration are described in [System parameters on page 124](#).

11 Accelerating conveyors

11.4 UseAccProfile - Use acceleration profile

11.4 UseAccProfile - Use acceleration profile

Description

UseAccProfile is used to predict conveyor movement with constant acceleration or deceleration.

The profile uses either the acceleration for the conveyor or the time that it takes for the conveyor to accelerate or decelerate. If two profiles are defined, acceleration value needs to be used instead of a time value.

The prediction of the conveyor acceleration is started by setting the I/O signal configured in *Sensor start signal* or *Sensor stop signal*. For best result this signal must be set at least 150 ms before the conveyor is starting to accelerate or decelerate.

The settings for the acceleration can be changed during program execution.

Example

```
VAR intnum intno1;
VAR triggdata trigg1;
...
CONNECT intno1 With Acc_Dec;
TriggInt trigg1, 0.5\Time, intno1;

Resetset sensor_start_signal_D0;
UseACCProfile CNV1, 0.4, 0, 1\acc, \stop_sig;
SetDO STARTSTOP_CNV, 1;
TriggL p0, v20, trigg1, z10, tool1\Wobj:=wobjconv;
MoveJ p_start, v1000, fine, tool1;

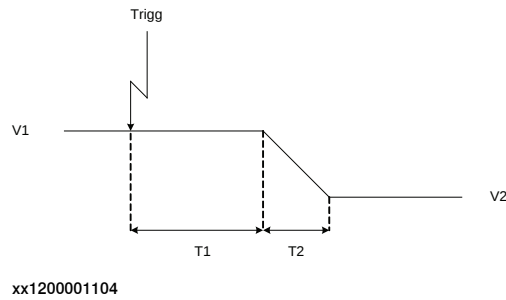
TRAP Acc_Dec PulseD0 \HIGH, sensor_start_signal_D0;
WaitTime 0.35;
SetDO STARTSTOP_CNV, 0;
ENDTRAP
```

In this example the start and stop of the conveyor is controlled by the I/O signal *STARTSTOP_CNV*. The deceleration profile is setup with a trigger_time of 0.4 s, end velocity of 0 m/s and the deceleration is 0.2 m/s². This means that the conveyor will decelerate from the current speed down to zero speed with a deceleration of 0.2 m/s² and that the sensor stop signal is going to be set 0.4 s before the conveyor is starting to decelerate.

The stop is triggered from a TriggL instruction.

As seen in the trap routine the *sensor_start_signal* is set 0.35 s before the stop order to the conveyor. However in the setup of the profile it is said that this signal is coming 0.4 s before the stop. In this case it might be that there is a delay in the communication with the conveyor controller of 0.05 s and this is compensated in this way.

Continues on next page



In the preceding figure, the profile from the example is shown. *V1* is the speed before the deceleration and can in this case be for example 0.2 m/s. *V2* is the speed after deceleration, in this case it is 0 m/s. *T1* represents the time between the *Trigg* is coming and the conveyor is starting to decelerate, in this example 0.4 s. *T2* is the length, in time, of the deceleration.

Arguments

```
UseAccProfile MechUnit, Trigger_time, V_end, Acc_time[\acc | time],
[\start_sig | stop_sig];
```

MechUnit

Mechanical Unit

Data type: mechunit

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

Trigger_time

Data type: num

The time between the *start_sensor_signal* is set and the time when the conveyor is starting to accelerate or decelerate. The time should not be smaller than 0.15 s. The value is given in seconds. In case of too small *Trigger_time* the profile might not be used.

V_end

Data type: num

Velocity to be reached at the end of acceleration or deceleration. In case of a stop this should be 0 m/s. The value is given in m/s.

Acc_time

Data type: num

Time from the start of the acceleration until the conveyor reaches the final speed (*V_end*). If *[\acc]* is set then this value is considered to be an acceleration value in m/s^2 describing the acceleration of the conveyor.

[\acc | time]

Data type: switch

Set to *acc* to use acceleration.

Set to *time* to use acceleration time.

Continues on next page

11 Accelerating conveyors

11.4 UseAccProfile - Use acceleration profile

Continued

[start_sig | stop_sig]

Data type: switch

Set to start_sig to use the signal configured as *Sensor start signal* to trigger the profile. Set to stop_sig to use the signal configured as *Sensor stop signal* to trigger the profile.

Program execution

To get the best possible accuracy during acceleration or deceleration it is important that the Trigger_time is the same as the time between setting the *Sensor start signal* and the time when the conveyor starts to accelerate or decelerate. The larger the difference is between these two times the poorer accuracy will be achieved.

If two profiles are configured in the system at the same time it is very important that the \acc option is used. This is to secure a good behavior when for example there is a mix between a start and a stop profile. This could happen when the conveyor for example is stopping and a start order is given so that the stop ramp never is finished.

Limitations

Before UseAccProfile is executed, the *Sensor start signal* and *Sensor stop signal* must be reset.

12 Indexing conveyors

12.1 Description of indexing conveyor options

Indexing conveyors

An indexing conveyor advances in steps instead of running continuously. One step is one index, and one or several indexes creates the work object. The conveyor belt can have pockets or magazines for the products, which makes them perfectly aligned for picking.

There are two ways to track indexing conveyors. If the conveyor is not controlled by the ABB robot controller then recorded profiles must be used, see [Tracking indexing conveyors on page 130](#). If the conveyor is controlled by the ABB robot controller the option *Internal Conveyor Control* should be used to get the best possible accuracy. See [Indexing conveyor with servo control \(Indexing Conveyor Control\) on page 143](#).

12 Indexing conveyors

12.2.1 Setting up tracking for an indexing conveyor

12.2 Tracking indexing conveyors

12.2.1 Setting up tracking for an indexing conveyor

Introduction

This section describes how to track an indexing conveyor using the option *Conveyor Tracking*. As the conveyor is not controlled by the robot controller, it is not possible to know exactly how the conveyor speed is changing. Instead, this method uses predicted speed changes. The repeatability is very important for the accuracy. Therefore, this method cannot be used if the conveyor movements are not repeatable. Then the option *Indexing Conveyor Control* with the conveyor controlled by the robot controller should be used.

To get good accuracy for indexing conveyors it must be possible to predict how the speed of the conveyor is changing. The prediction is based on a recorded profile of the conveyor during acceleration.

A new I/O signal must be defined and connected. RAPID instructions are used to handle prediction of conveyor position during speed changes.

Setting up tracking for an indexing conveyor

Use this procedure to set up tracking for an indexing conveyor.

- 1 Define a new I/O signal, topic *I/O*. See *Technical reference manual - System parameters*.
- 2 Connect the I/O signal to the conveyor system, topic *Process*. See [System parameters on page 131](#).
- 3 Define speed filter parameter. See [System parameters on page 131](#).
- 4 Record the profile. See [RecordProfile on page 133](#).
- 5 Store the profile. See [StoreProfile on page 136](#).
- 6 Load and/or activate the profile for production. See [LoadProfile on page 137](#), and [ActivateProfile on page 138](#).

12.2.2 System parameters

Topic Process

This parameter belongs to the type *Conveyor systems* in the topic *Process*.

Parameter	Description
Sensor_start_signal	Name of the digital input signal to synchronize recorded profile and new index movement. The signal must be set before start of conveyor movement. For example when a cam to move the conveyor the sensor can be placed to be triggered 100 ms before conveyor moves.

12 Indexing conveyors

12.2.3 Using indexing conveyor tracking from RAPID

12.2.3 Using indexing conveyor tracking from RAPID

Introduction

There are two ways to use the indexing conveyor tracking functionality from RAPID. One is to use the instruction `CnvGenInstr`, the other to use the predefined RAPID functions located in the RAPID module named *Indexing_cnv*. The RAPID instructions in *Indexing_cnv* encapsulates the functionality in `CnvGenInstr` to make it easier to use. To access these RAPID instructions, load the module *Indexing_cnv* from the conveyor tracking option directory.

12.2.4 RecordProfile

Description

RecordProfile resets all profile data and records a new profile of the conveyor movement as soon as the *sensor_start_signal* is set.

To be able to make a recording it is important that a connection to a work object is made before the recording is started. This means that a WaitWobj instruction has to be executed before the recording starts.

Example

```
ActUnit CNV1;
WaitWobj wobj_on_cnv1;
RecordProfile CNV1, 1, "index_profile";
WaitTime 0.2;
PulseD0 \HIGH sensor_start_signal_D0;
SetD0 STARTSTOP_CNV 1;
```

A profile of the conveyor is recorded as soon as the *sensor_start_signal* is set. In this example, the signal *STARTSTOP_CNV* starts the conveyor movement.

Arguments

RecordProfile MechUnit, Record_duration, Profile_type

MechUnit

Data type: mechunit

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

Record_duration

Duration of speed

Data type: num

Specifies the duration of record in seconds. Must be between 0.1 and $pos_update_time * 300$.

Profile_type

Type of profile

Data type: string

Value	Description
index_profile	Recording is started by <i>sensor_start_signal</i> .
start_stop_profile	A start and stop movement can be recorded. <i>sensor_start_signal</i> is used to record start movement and <i>sensor_stop_signal</i> is used to record <i>stop_movement</i> .
stop_start_profile	Same as for start_stop_profile but the <i>sensor_stop_signal</i> is used first.
stop_move_profile	The recording is started with <i>sensor_stop_signal</i> .

12 Indexing conveyors

12.2.5 WaitAndRecProf

12.2.5 WaitAndRecProf

Description

WaitAndRecProf resets all profile data and records a new profile of the conveyor movement as soon as the *sensor_start_signal* is set.

This instruction does the same as the instruction RecordProfile but it also handles the connection to a work object on the conveyor. This instruction is intended for use in PickMaster where the instruction WaitWobj is not available.

Example

```
WaitAndRecProf CNV1, 1, "index_profile";
```

A profile of the conveyor is recorded as soon as the *sensor_start_signal* is set.

Program execution

Use this procedure to execute the instruction WaitAndRecProf.

Arguments

```
WaitAndRecProf MechUnit, Record_duration, Profile_type
```

MechUnit

Data type: mechunit

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

Record_duration

Duration of speed

Data type: num

Specifies the duration of record in seconds. Must be between 0.1 and $pos_update_time * 300$.

Profile_type

Type of profile

Data type: string

Value	Description
index_profile	Recording is started by <i>sensor_start_signal</i> .
start_stop_profile	A start and stop movement can be recorded. <i>sensor_start_signal</i> is used to record start movement and <i>sensor_stop_signal</i> is used to record stop movement.
stop_start_profile	Same as for start_and_stop_profile but the <i>sensor_stop_signal</i> is used first.
stop_move_profile	The recording is started with <i>sensor_stop_signal</i> .

Continues on next page



Note

It is possible for the recording to fail and due to this no object gets created. As a result, the routine WaitAndRecProf is blocked in waitwobj with the message Waiting for sync signal. To simulate an object, use RAPID command `PulseD0 \PLength:=0.1,c1SoftSyncSig;`. After recording the profile, it is recommended to store the file and verify it. The first column is time, the second column is position, and the third column is speed.

12 Indexing conveyors

12.2.6 StoreProfile

12.2.6 StoreProfile

Description

StoreProfile activates and saves a recorded profile in a file.

Example

```
ActUnit CNV1;  
WaitWobj wobj_on_cnv1;  
RecordProfile CNV1, 1, "index_profile";  
WaitTime 0.2;  
PulseDO \HIGH sensor_start_signal_D0;  
SetDO STARTSTOP_CNV 1;  
WaitTime 2;  
SetDO STARTSTOP_CNV 0;  
StoreProfile CNV1, 0, "Profile.log";
```

A profile of the conveyor movement is recorded as soon as the *sensor_start_signal* is set and is stored in file profile.log.

Arguments

StoreProfile MechUnit, Delay, Filename

MechUnit

Data type: mechunit

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

Delay

Data type: num

The delay in seconds can be used to shift the record in time. It must be between 0.01 and 0.1. If the value is 0 (zero) no delay is added. The delay is not saved in the profile, it is only used for the activation. If the delay should be used together with a saved profile the delay has to be specified again in the instruction LoadProfile.

Filename

Data type: string

Name of the file where the profile is stored.

12.2.7 LoadProfile

Description

LoadProfile loads a recorded profile from a file.

Example

```
LoadProfile CNV1, 0, "profile.log";
WaitTime 0.2;
PulseD0 \HIGH sensor_start_signal_D0;
SetD0 STARTSTOP_CNV 1;
!
! Work against the conveyor
!
SetD0 STARTSTOP_CNV 0;
```

A saved profile of the conveyor movement is loaded and used for prediction of conveyor movement as soon as sensor_start_signal is set. Error warning SYS_ERR_MOC_CNV_REC_FILE_UNKNOWN if the file is not found.

Arguments

LoadProfile MechUnit, Delay, Filename

MechUnit

Data type: mechunit

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

Delay

Data type: num

The delay in seconds can be used to shift the record in time. It must be between 0.01 and 0.1. If the value is 0 (zero) no delay is added. The delay is not saved in the profile, it is only used for the activation.

Filename

Data type: string

Name of the file where the profile is stored.

12 Indexing conveyors

12.2.8 ActivateProfile

12.2.8 ActivateProfile

Description

ActivateProfile activates a profile that was just recorded to use it without having to save it before.

If the system is restarted, all unsaved records are lost. Therefore, use LoadProfile after restarts.

Do not use ActivateProfile after LoadProfile.

Example

```
ActivateProfile CNV1, 0;  
WaitTime 0.2;  
PulseDO \HIGH sensor_start_signal_D0;  
SetDO STARTSTOP_CNV 1;  
!  
! Work against the conveyor  
!  
SetDO STARTSTOP_CNV 0;
```

A profile of the conveyor is activated and used for prediction of conveyor movement as soon as the *sensor_start_signal* is set. Error warning SYS_ERR_MOC_CNV_REC_NOT_READY if record not finished.

Arguments

ActivateProfile MechUnit, Delay

MechUnit

Data type: mechunit

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

Delay

Data type: num

The delay in seconds can be used to shift the record in time. It must be between 0.01 and 0.1. If the value is 0 (zero) no delay is added.

12.2.9 DeactProfile

Description

DeactProfile deactivates a profile.

Example

```
DeactProfile CNV1;
```

A profile of the conveyor movement is deactivated and no longer used for prediction of conveyor movement.

Arguments

DeactProfile MechUnit

MechUnit

Data type: mechunit

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

12 Indexing conveyors

12.2.10 CnvGenInstr

12.2.10 CnvGenInstr

Description

`CnvGenInstr` sends a command to the conveyor process attached to the conveyor mechanical unit.

Example

```
CnvGenInstr CNV1, CNV_ACTIV_REC, mycnvdata;  
The controller will activate the record.
```

Arguments

```
CnvGenInstr MechUnit, cnvcmd, Data
```

MechUnit

Data type: `mechunit`

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

cnvcmd

Command

Data type: `num`

List of possible commands:

- `CNV_START_REC`
- `CNV_STOP_REC`
- `CNV_ACTIV_REC`
- `CNV_USE_FREQ`
- `CNV_RESET_ALPROF`
- `CNV_DEACT_PROF`
- `CNV_STORE_PROF`

Data

Data

Data type: `cnvgendata`

This structure is used to send `num` or `string` as parameters for different commands.

Program execution

All commands must be sent at least 0.2 seconds before start of conveyor movement.

More examples

Example1

```
VAR cnvgendata mycnvdata:=[0,0,0,0,"",""];  
CnvGenInstr CNV2,CNV_START_REC,mycnvdata;  
mycnvdata.value1:=1;
```

In this example, `data.value1` specifies the duration of recording in seconds. This value must be between 0.1 and `pos_update_time` * 300.

Continues on next page

Example 2

```
CnvGenInstr CNV2,CNV_STOP_REC,mycnvdata;
```

This example can be used if CNV_START_REC has been sent with duration 0.

Example 3

```
CnvGenInstr CNV2,CNV_ACTIV_REC,mycnvdata;
mycnvdata.value1:=0;
```

In this example, data.value1 specifies a delay added to record in seconds. This value must be between 0.01 and 0.1 seconds.

If value1=0 default value: then signal delay is used. Ready for use of profile on next index movement. Error warning SYS_ERR_MOC_CNV_REC_NOT_READY if record not finished.

Example 4

```
CnvGenInstr CNV2,CNV_USE_FREC,mycnvdata;
```

mycnvdata.string1:="myprofile": string1 must contain the name of the file where to read the recorded profile.

The file must have been created by the command CNV_STORE_PROF. Ready for use of profile on next index movement.

Error warning SYS_ERR_MOC_CNV_REC_FILE_UNKNOWN if record file not found.

Example 5

```
CnvGenInstr CNV2,CNV_RESET_ALPROF,mycnvdata;
```

Reset all profile data, ready for a new START_REC.

Example 6

```
CnvGenInstr CNV2,CNV_DEACT_PROF,mycnvdata;
```

Stop using profile.

Example 7

```
CnvGenInstr CNV2,CNV_STORE_PROF,mycnvdata;
```

mycnvdata.string1:="myprofile";string1 must contain the name of the file to store the profile.

Limitations

As access to files can take a lot of time it is recommended not to use CNV_USE_FREC and CNV_STORE_PROF while robot is moving.

Repeatability error between record and real cycles must be less than 120 ms. A delay between *sensor_start_signal* and conveyor movement must not vary more than 120 ms.

Error handling

No error handling for this instruction. In case of emergency stop of robot or conveyor the command CNV_DEACT_PROF should be used before restarting the robot.

Syntax

```
CnvGenInstr
[ MechUnit '[:=']< var of mechanical unit > ';'
[ Command '[:=']< expression (IN) of num> ';'

```

Continues on next page

12 Indexing conveyors

12.2.10 CnvGenInstr

Continued

```
[ Data ':=']< var of cnvgendata> ';' 
```

12.3 Indexing conveyor with servo control (Indexing Conveyor Control)

12.3.1 Introduction to indexing conveyors with servo control

Description of *Indexing Conveyor Control*

The option *Indexing Conveyor Control* includes RAPID instructions and one RAPID data type. A typical installation includes an infeed conveyor transporting the products in a row at high speed to the indexing conveyor. A photo eye (sensor) mounted on the infeeder detects products and sends a trig signal to the robot controller. The controller starts the indexing movement after a specified time, that is, when the product has entered the pocket on the indexing conveyor. The FlexPicker picks the products from the indexing conveyor, even during indexing movement, and puts them in a box on the output conveyor.

The robot controller controls the FlexPicker and the indexing conveyor. The output conveyor and the infeeder have their own drive systems.

Conveyor tracking will be used both on the indexing conveyor and the output conveyor, but no encoder and encoder interface is needed on the indexing conveyor since the position and speed are known as a part of the path planning in the robot controller.



Note

The option *Indexing Conveyor Control* requires that the robot controller has additional drive units.

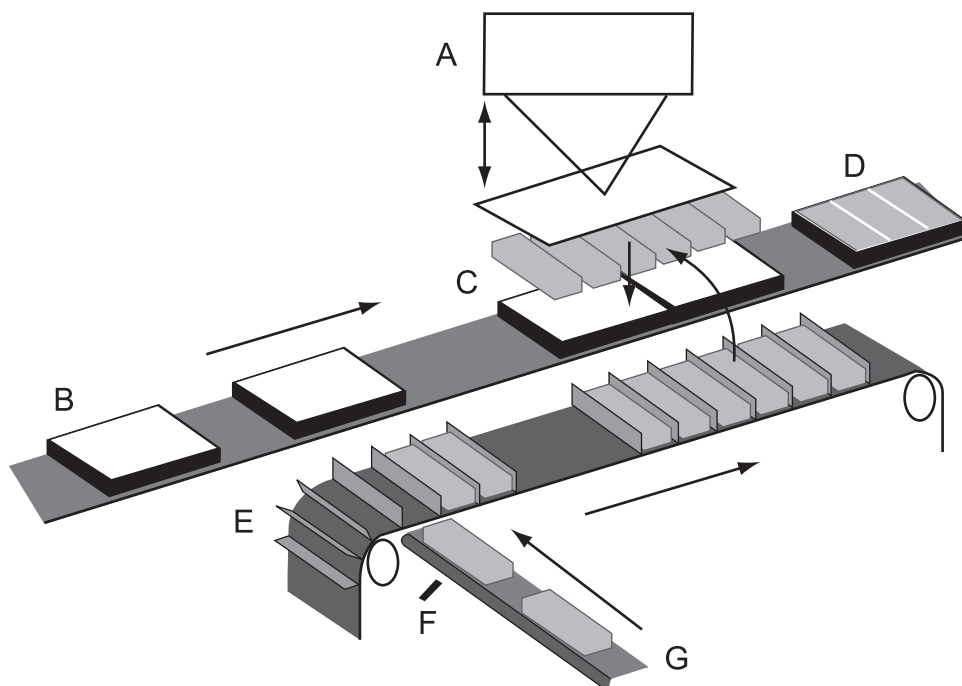
Continues on next page

12 Indexing conveyors

12.3.1 Introduction to indexing conveyors with servo control

Continued

Schematic overview



xx1000001425

A	Robot with gripper
B	Output conveyor with empty cartons, for example from carton erector
C	Robot picks products from indexing conveyor and places in cartons
D	Full cartons, for example to carton closer
E	Indexing conveyor with pockets
F	Photo eye
G	Product infeed

Terminology

In context of controlling and moving the conveyor this is referred to as *M7* in this document.

In context of tracking the conveyor, it is referred to as *CNV1*. Hence, the conveyor will be configured as two different mechanical units, *M7* and *CNV1*.

Indexing mode is when the system listens (or waits) for a signal from the sensor that a product is available.

In indexing mode it is not possible to jog or use Move instructions. To disable indexing mode, execute `IndCnvReset` or move the program pointer to the routine `Main (PP to Main)`.

Limitations

When using the option *Indexing Conveyor Control*, up to two indexing conveyors and two conventional (non-indexing) conveyors with encoder boards can be used per controller. Up to two IRB 360 robots can be configured in a MultiMove system.

12.3.2 Setting up a servo controlled indexing conveyor

Installing the additional axis for servo control

For the option *Indexing Conveyor Control*, the indexing conveyor should be running as an additional axis in the robot controller. See *Application manual - Additional axes*.

Installing the software

The conveyor tracking RAPID instructions, data types, and mechanical unit CNV1 are specified in the key string and do not need to be installed. If the system consists of more than one conveyor, three more files must be installed per conveyor. The files to install are stored on the controller.

The second conveyor to install is called CNV2 and the *Motion* configuration file is named `cnv2_moc.cfg`. The other two files to install for internally controlled conveyors are `cnvint2_prc.cfg` and `cnvint2_eio.cfg`.

12 Indexing conveyors

12.3.3 System parameters and configuration files

12.3.3 System parameters and configuration files

Topic I/O

Type Unit

Verify that the I/O unit *Qtrack1* is defined as *Virtual* in the type *Unit*.

Parameter	Value
Connected To Bus	Virtual1

Type Unit Type

Verify that the I/O unit type used for the photo eye is defined as *Change of State*.

Parameter	Value
Connection 1 type	Change Of State (COS)

Type Signal

In the type *Signal*, configure a digital input signal for the photo eye which will trigger an indexing movement of the conveyor.

Parameter	Value
Name	DI_Eye
Type of Signal	Digital Input

Topic Controller

When running a MultiMove system and the indexing conveyor is running in a separate motion task, the following must be added.

Type Mechanical Unit Group

In the type *Mechanical Unit Group*.

Parameter	Value
Mechanical Unit Group	M7
Mech Unit 1	M7
Use Motionplanner	motion_planner_3

Type Tasks

In the type *Task*.

Parameter	Value
Use Mechanical Unit Group	M7

Type Motion System

In the type *Motion System*.

Parameter	Value
Name	motion_planner_3
dyn_ipol_type	1

Continues on next page

**Note**

Verify that the *motion_planner_3* has *dyn_ipol_type* defined as 1. To edit this parameter, create a backup and edit *moc.cfg*. This cannot be changed using RobotStudio.

Topic Process**Type Conveyor systems**

In the type *Conveyor systems*, verify that the following parameters are set.

Parameter	Value
Syncfilter Ratio	0.0001
Acc Dependent Filter Value	0

Type Conveyor Internal

The instance is named *INTERNAL1*.

Parameter	Description
Eio unit name	Name of the simulated I/O unit.
Connected signal	Name of the digital input signal for connection.
Position signal	Name of the analog input signal for conveyor position.
Velocity signal	Name of the analog input signal for conveyor speed.
Null_speed signal	Name of the digital input signal indicating zero speed on the conveyor.
DropWObj signal	Name of the digital output signal to drop a connected object.
ObjLost signal	Name of the digital input signal to indicate that an object has gone past the start window without being connected.
RemAllPObj sig	Name of the digital input signal to remove all Pobj.
Rem1PObj sig	Name of the digital input signal to remove one Pobj.
Pos Update time	Defines how often position and velocity output signals are updated.
Supervise max_dist Off	Boolean to remove supervision of maximum and minimum distance. Default value is YES.
New object strobe	Name of digital output signal showing a new object in queue.
Objects in queue	Name of group output signal showing the number of objects in queue.
Count1 from encoder	Name of group output for new object position low word.
Count2 from encoder	Name of group output for new object position high word.
Single to track	Name of the single that is moving the indexing conveyor. (<i>Indexing Conveyor Control</i> .)

Continues on next page

12 Indexing conveyors

12.3.3 System parameters and configuration files

Continued

Topic Motion

Type Arm

In the type *Arm*, enable *Independent Joint* and add joint limits.

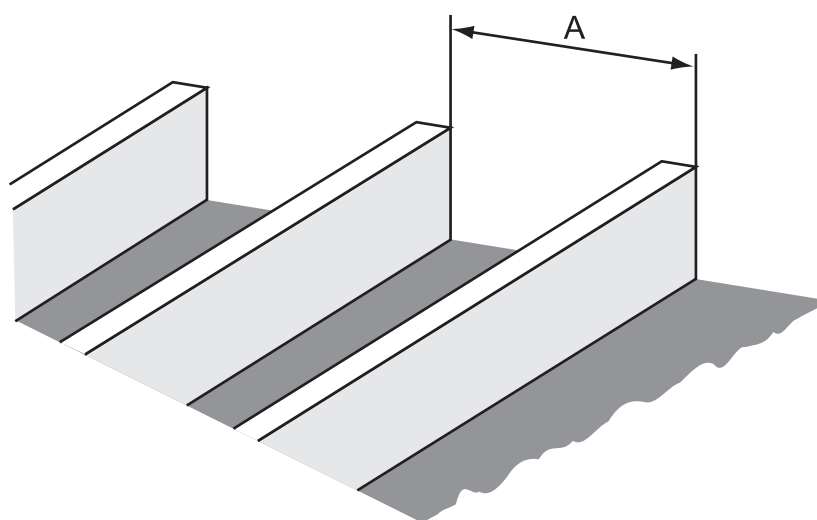
Parameter	Value
Independent Joint	On
Independent Upper Joint Bound	2E+07
Independent Lower Joint Bound	-2E+07

Type Single Type

In the type *Single Type*, the parameter *Pocket size* defines the distance the conveyor will move when triggered by the photo eye (the size of one pocket), see the following graphic. It is very important for conveyor tracking accuracy that the pocket size is given correctly. Use measuring tape and measure at least ten pockets to have a average for one pocket.

Time before indexing move defines the time from the photo eye is triggered until the conveyor movement starts. The recommended value is 0.3 (300 ms). Depending on the robot payload and deceleration distance, this value can be decreased (might be needed to increase for some applications). If it is important to use a value as small as possible, see [Minimizing trigger time on page 161](#).

Parameter	Value
Mechanics	FREE_ROT
Indexing move	Yes
Time before indexing move	0.3
Pocket size	0.05



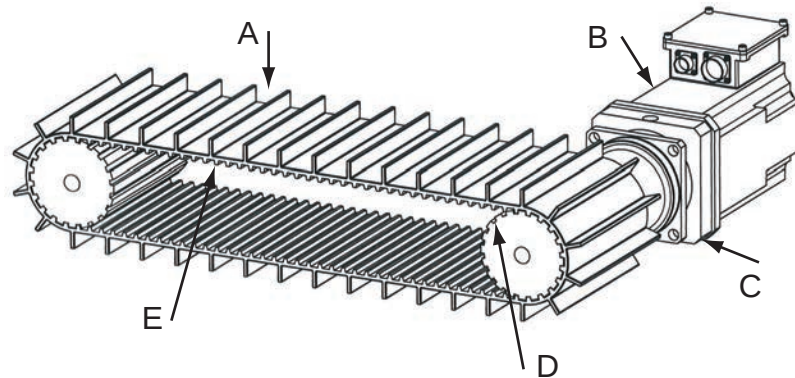
xx1200001105

A	Pocket size
---	-------------

Continues on next page

Type Transmission

The transmission must be represented by two integer parameters, *Transmission Gear High* and *Transmission Gear Low*. This is needed to avoid losing accuracy after a very big number of indexing movements. These parameters are computed in a way that makes it possible for the controller to move the indexing conveyor exactly one pocket instead of for example 50 mm which could be the measured pocket size.



xx1200001106

A	Pockets
B	Motor
C	Gearbox
D	Gear wheel teeth
E	Belt teeth

See the following example for the calculation of the parameters *Transmission Gear High*, *Transmission Gear Low*, and *Transmission Gear Ratio*. With this setup, the parameter *Rotating Move* must be set.

Calculation example:

D = Number of gear wheel teeth

A = Number of pockets

$$\text{Transmission Gear Low} = D * A = 20 * 36 = 720$$

G = Gear box ratio

E = Number of belt teeth

$$\text{Transmission Gear High} = G * E * 360 = 10 * 108 * 360 = 388800$$

$$\text{Transmission Gear Ratio} = \text{Transmission Gear High} / \text{Transmission Gear Low} = 388800 / 720 = 540 / 1 = 540$$

In this example we use the gear box ratio 10, this means 10 motor revolutions correspond to one gear wheel revolution. The *Transmission Gear High* / *Transmission Gear Low* ratio can be given as 388800 / 720 but 540 / 1 is easier to comprehend.

Parameter	Value
Rotating move	YES

Continues on next page

12 Indexing conveyors

12.3.3 System parameters and configuration files

Continued

Parameter	Value
Transmission Gear Ratio	540 (from example)
Transmission Gear High	540 (from example)
Transmission Gear Low	1 (from example)

To change the direction of movement of the axis, change sign of the transmission parameters *Transmission Gear Ratio* and *Transmission Gear High*.

Type Acceleration Data

For an indexing conveyor the acceleration data is given in m/s^2 even though the parameter *Rotating Move* is set. The indexing movement will be a symmetric triangular motion profile.

If different values are given for acceleration and deceleration, the smallest value will be used when creating the motion path.

To avoid vibrations and overload of the mechanical structure, the values for acceleration and deceleration should not be set higher than the robot capacity at the given payload.

Parameter	Value
Nominal Acceleration	25
Nominal Deceleration	25

12.3.4 Testing the indexing conveyor setup

Testing

Create a RAPID program using the code from the example below. Speed and acceleration are given in mm/s respectively mm/s^2 . To avoid vibrations and overload of the mechanical structure, the values for acceleration and deceleration should not be set higher than the robot capacity at the given payload.

In this example an indexing movement will be performed each time the digital input signal *DI_Eye* is triggered.

RAPID example

```
MODULE MainModule
  VAR indcnvdata indcnvdata1:=[0,0,0,0,0,0,0];
  PROC main()
    indcnvdata1.speed := 2000;
    indcnvdata1.acceleration := 25000;
    indcnvdata1.productsperpick := 6;
    indcnvdata1.productsperindex := 1;

    ActUnit M7;
    IndCnvInit M7, DI_Eye, indcnvdata1;
    IndCnvEnable M7;

    WHILE TRUE DO
      WaitTime 1;
    ENDWHILE
  ENDPROC
ENDMODULE
```

12.3.5 Calibrating the base frame

Creating the work object

To calibrate the base frame, first create the work object. Then calibrate and verify the calibration.

Use this procedure to calibrate the base frame for CNV1.

- 1 Jog the mechanical unit M7 to the correct position according to the infeed.
- 2 Perform a fine calibration of M7 in this position.
- 3 Activate the mechanical units M7 and CNV1.

```
ActUnit M7;  
ActUnit CNV1;
```

- 4 Initialize the indexing conveyor by executing the following RAPID instruction where `indcnvdata1` is setup as described in [Testing the indexing conveyor setup on page 151](#).

```
IndCnvInit M7, DI_Eye, indcnvdata1;
```

- 5 To make sure there are no objects in the object queue execute the RAPID instruction:

```
PulseDO \PLength:=0.1,c1RemAllP0bj;
```

- 6 Add a new object to the object queue by executing the RAPID instruction:

```
IndCnvAddObject M7;
```

- 7 To be able to jog M7 during calibration it is necessary to reset the indexing functionality by executing the RAPID instruction:

```
IndCnvReset M7;
```

Calibrating the base frame

Start the base frame calibration routine of CNV1 and move the conveyor to the calibration positions by jogging the mechanical unit M7. See [Calibrating the base frame on page 68](#).

Continue with verifying the calibration, see [Verifying the base frame calibration on page 69](#).

12.3.6 indcnvdata

Description

The data type `indcnvdata` contains information about the indexing movement and the number of pockets that the work object holds.

Example

```
VAR indcnvdata indcnvdata1:= [0,0,0,0,0,0,0];
indcnvdata1.speed:= 2000;
indcnvdata1.acceleration := 25000;
indcnvdata1.productsperpick := 6;
indcnvdata1.productsperindex := 1;
indcnvdata1.accuracytuning := 0;
```

Components

speed

Conveyor speed in mm/s. Normally this parameter is set to a high value to create a motion profile that is triangular.

acceleration

Conveyor acceleration in mm/s^2 . This value cannot be higher than what is configured in *Acceleration Data*, see [Type Acceleration Data on page 150](#).

To avoid vibrations and overload of the mechanical structure, the values for acceleration and deceleration should not be set higher than the robot capacity at the given payload.

productsperpick

The number of pockets that offsets each work object. If there is one product in each pocket this parameter defines how many products the robot will pick in each robot cycle.

productsperindex

Use this parameter to set how many products that should enter each pocket before executing an indexing movement.

accuracytuning

This parameter can be used to tune the synchronization between robot and indexing conveyor and requires a high speed camera. The tuning value is default 0 and can be adjusted $\pm 10\text{ms}$.

12 Indexing conveyors

12.3.7 IndCnvInit

12.3.7 IndCnvInit

Description

IndCnvInit used to set up the indexing conveyor functionality.

Example

```
IndCnvInit M7, DI_Eye, indcnvdata1;
```

Arguments

```
IndCnvInit MechUnit, Signal, indcnvdata1;
```

MechUnit

Mechanical Unit

Data type: mechunit

The name of the mechanical unit.

Signal

Signal

Data type: signaldi

The name of the digital input signal that triggers the indexing movement.

indcnvdata

Data type: indcnvdata

IndCnvData speed, acceleration, productsperpick, productsperindex, accuracytuning

Error handling

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

ERR_INT_NOTVAL	Not valid integer, decimal value
ERR_NO_ALIASIO_DEF	The signal variable is a variable declared in RAPID. It has not been connected to an I/O signal defined in the I/O configuration with instruction AliasIO.
ERR_NORUNUNIT	If there is no contact with the I/O unit

12.3.8 IndCnvEnable and IndCnvDisable

Description

IndCnvEnable is used to set the system in indexing mode. An indexing movement will be executed when the signal is triggered (indexing mode).

IndCnvDisable is used to stop listening to the digital input signal. No indexing movement will be performed even if the signal is triggered.



Note

It is not possible to jog or run Move instructions on the indexing conveyor until a IndCnvReset instruction has been executed.

Example

```
IndCnvEnable M7;  
IndCnvDisable M7;
```

Arguments

```
IndCnvEnable MechUnit;  
IndCnvDisable MechUnit;
```

MechUnit

Mechanical Unit

Data type: mechunit

The name of the mechanical unit.

Error handling

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

ERR_INDCNV_ORDER	An instruction requires execution of IndCnvInit before it is executed.
------------------	--

12 Indexing conveyors

12.3.9 IndCnvReset

12.3.9 IndCnvReset

Description

IndCnvReset ends indexing mode and sets the system to normal mode which makes it possible to jog and run Move instructions.

Example

```
IndCnvReset M7;
```

Arguments

```
IndCnvReset MechUnit;
```

MechUnit

Mechanical Unit

Data type: mechunit

The name of the mechanical unit.

12.3.10 IndCnvAddObject

Description

To manually add an object to the object queue, the RAPID instruction `IndCnvAddObject` can be executed.

Example

```
IndCnvAddObject M7;
```

Arguments

```
IndCnvAddObject MechUnit;
```

MechUnit

Mechanical Unit

Data type: mechunit

The name of the mechanical unit.

12 Indexing conveyors

12.3.11 RAPID programming example

12.3.11 RAPID programming example

Description

In this example an IRB 360 is picking a batch of products from an indexing conveyor (CNV1) and placing the products on the outfeed conveyor (CNV2). The outfeed conveyor is not controlled by the robot controller.

Example code

```
MODULE MainModule
TASK PERS wobjdata wobj1:= [FALSE,FALSE,"CNV1",[[0,0,0],
[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
TASK PERS wobjdata wobj2:= [FALSE,FALSE,"CNV2",[[0,0,0],
[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
CONST robtarget WaitPos:= [[-129.82,1.57,-
924.52],[0,0.999848,-0.0174063,0],
[0,0,0,0],[0,9E+09,9E+09,9E+09,0,0]];
CONST robtarget Cnv1_Above:= [[24.81,40.73,34.42],
[0.000115829,-0.753048,0.657965,-0.000112099],
[0,1,0,0],[171.015,9E+09,9E+09,9E+09,277.57,400]];
CONST robtarget Cnv1_Below:= [[24.81,40.73,-9.77],
[0.000115827,-0.75306,0.657952,-0.000112101],
[0,1,0,0],[171.015,9E+09,9E+09,9E+09,277.57,400]];
CONST robtarget Cnv2_Above:= [[-23.46,-1.68,201.85],
[1.72038E-05,0.997874,0.0651587,-0.000988565],
[0,0,0,0],[158.015,9E+09,9E+09,9E+09,277.57,0]];
CONST robtarget Cnv2_Below:= [[-23.46,-1.68,55.28],
[1.72335E-05,0.997876,0.0651288,-0.000988564],
[0,0,0,0],[158.015,9E+09,9E+09,9E+09,277.57,0]];
PERS tooldata Tool_2:= [TRUE,[[0,0,56.5],[1,0,0,0]],
[2,[0,0,26],[1,0,0,0],0,0,0.0016]];
VAR trigdata EaciPick;
VAR trigdata EaciPlace;
VAR speeddata speed_eaci;
VAR indcnvdata indcnvdata1:= [0,0,0,0,0,0,0];
VAR num prod_cnt;
CONST stoppointdata
  stoppoint_eaci:= [3,FALSE,[0,0,0,0],0,0.035,"",0,0];
PROC main()
  indcnvdata1.speed := 2000;
  indcnvdata1.acceleration := 25000;
  indcnvdata1.productsperpick := 6;
  indcnvdata1.productsperindex := 1;

  prod_cnt := 0;

  speed_eaci.v_tcp:= 5000;
  speed_eaci.v_ori:= 10000;

  ! Activate mechanical units
  ActUnit M7;
  ActUnit CNV1;
```

Continues on next page

```

ActUnit CNV2;

! Remove from queue and drop objects
PulseDO \PLength:=0.1,c1RemAllPObj;
PulseDO \PLength:=0.1,c2RemAllPObj;
DropWObj wobj1;
IF c2Connected = 1 THEN
    DropWObj wobj2;
ENDIF

! Set up pick and place trigg data
TriggEquip EaciPick, 2, 0.05 \DOp:=EaciSuck, 1;
TriggEquip EaciPlace, 2, 0.05 \DOp:=EaciSuck, 0;

! Set indexing conveyor CNV1 in indexing mode and start listening
to the photo eye signals
IndCnvInit M7, DI_Eye, indcnvdata1;IndCnvEnable M7;
! Go to init position
MoveL WaitPos, v1000, fine, Tool_2;

WHILE TRUE DO

    ! CNV1 Pick
    WaitWObj wobj1\RelDist:=300;
    MoveL Cnv1_Above, speed_eaci, z20, Tool_2\WObj:=wobj1;
    TriggL Cnv1_Below, speed_eaci, EaciPick,
        z1\Inpos:=stoppoint_eaci, Tool_2\WObj:=wobj1;
    MoveL Cnv1_Above, speed_eaci, z20, Tool_2\WObj:=wobj1;
    IF prod_cnt >= 1 THEN
        DropWObj wobj2;
    ENDIF
    ! CNV2 Place
    WaitWObj wobj2\RelDist:=50;
    MoveL Cnv2_Above, speed_eaci, z20, Tool_2\WObj:=wobj2;

    TriggL Cnv2_Below, speed_eaci, EaciPlace,
        z1\Inpos:=stoppoint_eaci, Tool_2\WObj:=wobj2;
    MoveL Cnv2_Above, speed_eaci, z20, Tool_2\WObj:=wobj2;

    prod_cnt := prod_cnt + 1;
    TPErase;
    TPWrite " Number of products: "\Num:=prod_cnt;

    DropWObj wobj1;
ENDWHILE

! Move program pointer to the instructions below to enable e.g.
jogging
! Stop listening to the photo eye signals
IndCnvDisable M7;

```

Continues on next page

12 Indexing conveyors

12.3.11 RAPID programming example

Continued

```
! Set indexing conveyor in normal mode
IndCnvReset M7;
ENDPROC
ENDMODULE
```

12.3.12 Minimizing trigger time

Trigger time

If the parameter *Time before indexing move* is set too short, the error **50423 IndCnv Time before indexing move too low** can occur.

To minimize the time between the trigger from the photo eye and start of the conveyor movement, follow the description below:

- 1 Set *Time before indexing move* to a high value, for example 0.4 s.
- 2 If trigger time is critical there is a chance to reduce the time by changing the parameters as suggested in [Possible solutions on page 161](#).
- 3 Run the full application with the robot coordinated to CNV1 and with full robot payload.
- 4 Run the service routine `IndCnvOptimalTimeBefore` to get a proposed optimal time before.
- 5 Update *Time before indexing move* to match the value proposed by the service routine.

There is a risk that the value suggested by the service routine `IndCnvOptimalTimeBefore` is too small so some extra 10 ms margin might be needed.

Possible solutions

Change the following parameters in the type *Motion Planner*.

Parameter	Value
Dynamic Resolution	0.3333
Path Resolution	0.3333
Queue Time	0.032256
Group Queue Time	0.016128

Possible limitations

If the error **50082 Deceleration limit** occurs, return to the original setting of *Queue Time* (which is 0.064512), in the type *Motion Planner* (topic *Motion*).

If the error **50226 Motor reference error** occurs, return to the original setting of *Group Queue Time* (which is 0.032256), in the type *Motion Planner* (topic *Motion*).

This page is intentionally left blank

13 Conveyor tracking and MultiMove

13.1 About conveyor tracking and MultiMove

Examples of use

Here are some examples of applications where conveyor tracking is combined with MultiMove:

- Several robots can work on the same object moving on a conveyor.
- Several robots can cooperate to pick objects on a conveyor.

Additional information

If the option *Indexing Conveyor Control* is used for a conveyor in a MultiMove system, then some additional parameters must be defined. See [System parameters and configuration files on page 146](#). See also [Indexing conveyor with servo control \(Indexing Conveyor Control\) on page 143](#).

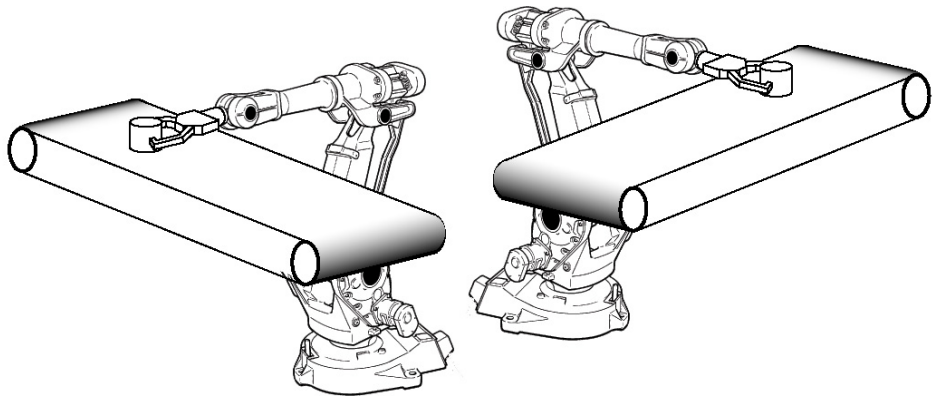
Application manual - MultiMove

Two application examples

This manual describes two examples of robot system setups to demonstrate how conveyor tracking can be combined with MultiMove. They are called *UnsyncCnv* and *SyncCnv*. See [Configuration example for UnsyncCnv on page 165](#), and [Configuration example for SyncCnv on page 167](#).

UnsyncCnv

In the example *UnsyncCnv*, two robots work independently on one work piece for each robot. They do not cooperate in any way and do not have to wait for each other. There is one conveyor mechanical unit for each robot.



xx1200001107

Continues on next page

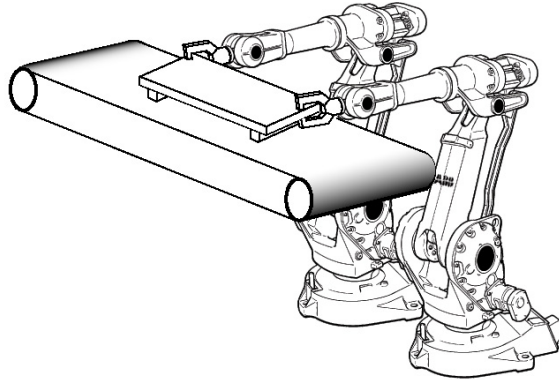
13 Conveyor tracking and MultiMove

13.1 About conveyor tracking and MultiMove

Continued

SyncCnv

In the example SyncCnv, two robots arc weld on the same work piece. The work object is moved by a conveyor. One conveyor mechanical unit is used in a separate motion planner.



xx1200001108

13.2 Configuration example for UnsyncCnv

About this example

This section describes how to configure the example UnsyncCnv, with two independent robots. The robots are handled by one task each.

Configuration

Task

Task	Type	MotionTask	Use Mechanical Unit Group
T_ROB1	NORMAL	Yes	rob1
T_ROB2	NORMAL	Yes	rob2

Mechanical Unit Group

Name	Robot	Mech Unit 1	Use Motion Planner
rob1	ROB_1	CNV1	motion_planner_1
rob2	ROB_2	CNV2	motion_planner_2

Motion Planner

Name
motion_planner_1
motion_planner_2

Mechanical Unit

Name	Allow Move of User Frame	Activate at Start-up	Deactivation Forbidden	Use Brake Relay
ROB_1	No	Yes	No	
ROB_2	No	Yes	No	
CNV1	Yes	No	No	rob1_brake
CNV2	Yes	No	No	rob2_brake

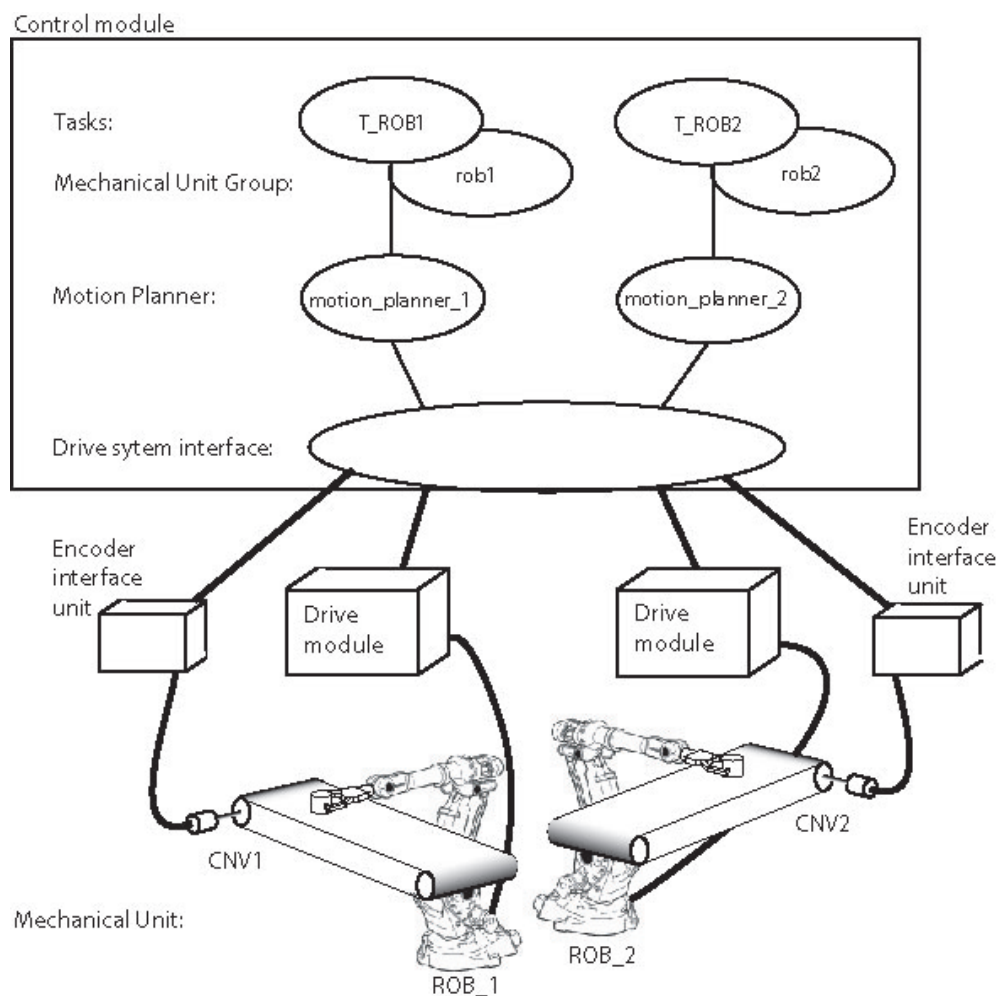
Continues on next page

13 Conveyor tracking and MultiMove

13.2 Configuration example for UnsyncCnv

Continued

Illustration



xx1200001109

13.3 Configuration example for SyncCnv

About this example

This section describes how to configure the example SyncCnv, with two robots and one positioner. Each mechanical unit is handled by a separate task.

Configuration

Task

Task	Type	MotionTask	Use Mechanical Unit Group
T_ROB1	NORMAL	Yes	rob1
T_ROB2	NORMAL	Yes	rob2
T_CONV3	NORMAL	Yes	conv3

Mechanical Unit Group

Name	Robot	Mech Unit 1	Use Motion Planner
rob1	ROB_1		motion_planner_1
rob2	ROB_2		motion_planner_2
conv3		CNV3	motion_planner_3

Motion Planner

Name
motion_planner_1
motion_planner_2
motion_planner_3

Mechanical Unit

Name	Allow Move of User Frame	Activate at Start-up	Deactivation Forbidden	Use Brake Relay
ROB_1	No	Yes	No	
ROB_2	No	Yes	No	
CNV3	Yes	Yes	No	rob1_brake

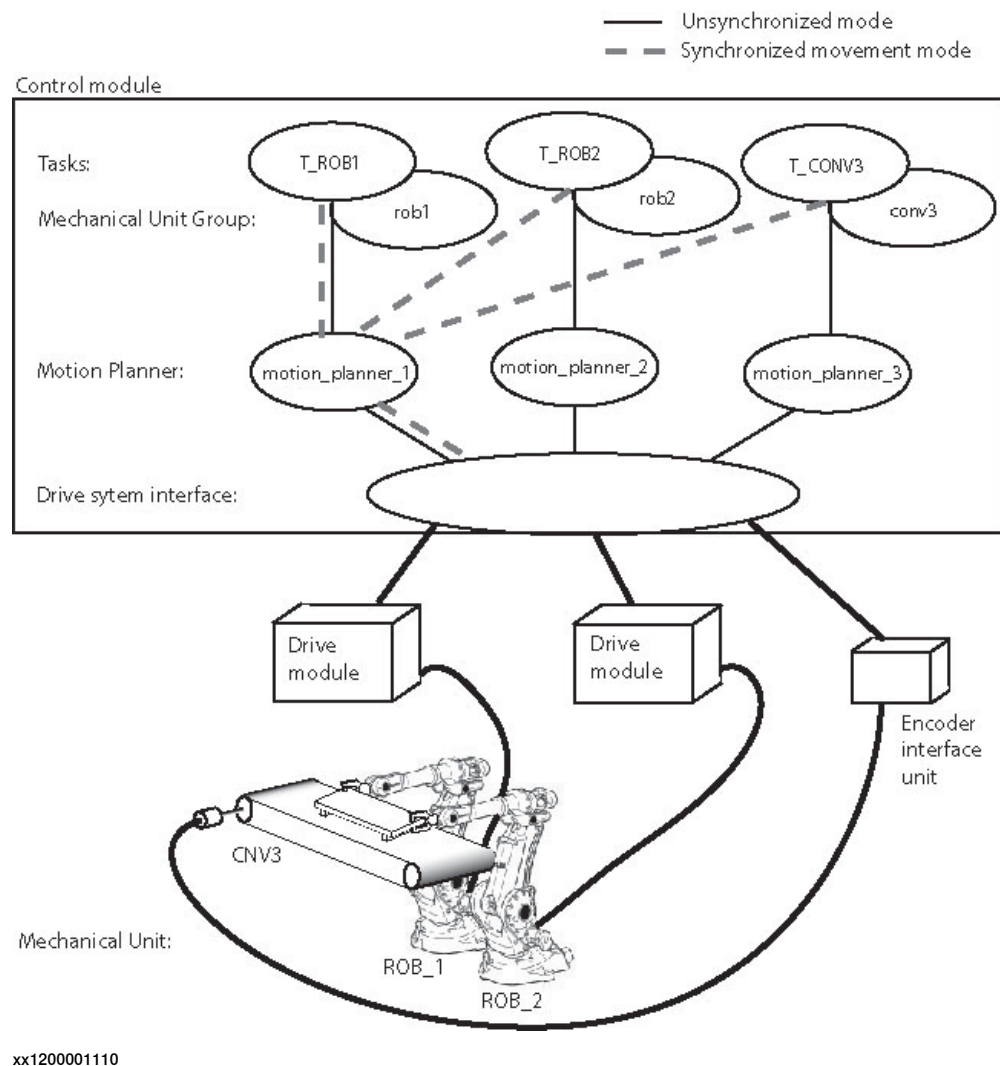
Continues on next page

13 Conveyor tracking and MultiMove

13.3 Configuration example for SyncCnv

Continued

Illustration



Calibration overview

For unsynchronized movements each conveyor must be calibrated with its motion group robot (after the base calibration of the robot):

- Cnv1 with Robot_1
- Cnv2 with Robot_2

For synchronized movements Cnv3 is calibrated with one robot only: Robot_1.

13.4 Tasks and programming techniques

Introduction to tasks

Each task program can handle the movements for one robot and up to 6 additional axes. Several tasks can be used, each containing a program quite similar to the program of the main task in a single robot application. For more information about tasks, see *Multitasking in Application manual - Controller software OmniCore RobotWare 8*.

One task program per robot

Each task program can only handle one TCP. This means that you must have one task for each robot.

Conveyor in separate tasks

Conveyors that move a work object can be handled by the same task program as one of the robots for un-synchronized movements. For synchronized movements where the conveyor should be able to move independently of the robots, it is best to have a separate task program for the conveyor.

Modifying positions

When modifying positions it is not possible to just stop the conveyor and modify the position. If this is done, there will be an offset in the position that corresponds to the length of the conveyor movement. This is since the conveyor and the robots belong to different mechanical unit groups, see [Configuration example for SyncCnv on page 167](#), and the position of the conveyor is not known to the robots at this point.

To be able to modify positions, the conveyor must belong to the same mechanical unit group as the robot. This is done by creating "fake" conveyors that are only used for modifying positions.

Example

Below there is a configuration example for topic *System*. Conveyors CNV1 and CNV2 are "fake" conveyors, and conveyor CNV3 is the real conveyor.

Task:

Task	Type	MotionTask	Use Mechanical Unit Group
T_ROB1	NORMAL	Yes	ROB1
T_ROB2	NORMAL	Yes	ROB2
T_CNV	NORMAL	Yes	GCVN

Mechanical Unit Group:

Name	Robot	Mech Unit 1	Use Motion Planner
ROB1	ROB_1	CNV1	motion_planner_1
ROB2	ROB_2	CNV2	motion_planner_2
GCVN		CNV3	motion_planner_3

Continues on next page

13 Conveyor tracking and MultiMove

13.4 Tasks and programming techniques

Continued

Mechanical Unit:

Name	Allow Move of User Frame	Activate at Start-up	Deactivation Forbidden	Use Brake Relay
CNV1	Yes	No	No	rob1_brake
CNV2	Yes	No	No	rob2_brake
CNV3	Yes	Yes	No	rob1_brake

In the configuration for topic *Process*, the "fake" conveyors CNV1 and CNV2 must be connected to the same encoder as CNV3, and must be configured using the same I/O signals as CNV3, see [Combining synchronized and un-synchronized mode on page 177](#).

The last step is to create unsynchronized RAPID procedures for each task with move instructions where the positions can be modified.

Action	
1	Create new work objects for the "fake" conveyors, for example wobj_CNV1 for ROB_1 and wobj_CNV2 for ROB_2. (In the synchronized production program both robots uses the same work object, for example wobj_CNV3.)
2	Copy the production procedures for robot 1, remove the synchronization, and replace the work objects. For example, the production procedure in synced task T_ROB1: MoveL p101\ID:=10, v300, fine, tool1\Wobj:=wobj_CNV3; MoveL p102\ID:=20, v300, fine, tool1\Wobj:=wobj_CNV3; MoveL p103\ID:=30, v300, fine, tool1\Wobj:=wobj_CNV3; is copied to a new routine and modified: MoveL p101, v300, fine, tool1\Wobj:=wobj_CNV1; MoveL p102, v300, fine, tool1\Wobj:=wobj_CNV1; MoveL p103, v300, fine, tool1\Wobj:=wobj_CNV1;
3	Copy the production procedures for robot 2, remove the synchronization, and replace the work objects. For example, the production procedure in synced task T_ROB2: MoveL p201\ID:=10, v300, fine, tool1\Wobj:=wobj_CNV3; MoveL p202\ID:=20, v300, fine, tool1\Wobj:=wobj_CNV3; MoveL p203\ID:=30, v300, fine, tool1\Wobj:=wobj_CNV3; is copied to a new routine and modified: MoveL p201, v300, fine, tool1\Wobj:=wobj_CNV2; MoveL p202, v300, fine, tool1\Wobj:=wobj_CNV2; MoveL p203, v300, fine, tool1\Wobj:=wobj_CNV2;
4	Modify the positions for each robot.

13.5 Independent movements, example UnsyncCnv

ROB1 task program

```
MODULE module1
  TASK PERS wobjdata wobj1 := [ FALSE, TRUE, "", [ [500, -200,
    1000], [1, 0, 0 ,0] ], [ [100, 200, 100], [1, 0, 0, 0] ]
    ];
  TASK PERS wobjdata wobjcnv1 := [ FALSE, FALSE, "CNV1", [ [0,0,
    0], [1, 0, 0 ,0] ], [ [0, 0, 0], [1, 0, 0, 0] ] ];
  TASK PERS tooldata tool1 := ...
  CONST robtarget p11 := ...
  ...
  CONST robtarget p14 := ...

  PROC main()
    ...
    IndependentMove;
    ...
  ENDPROC

  PROC IndependentMove()
    MoveL p11, v500, fine, tool1\WObj:=wobj1;
    WaitWObj wobjcnv1\RelDist:=10;
    MoveC p12, p13, v500, z10, tool1\WObj:=wobjcnv1;
    MoveC p14, p11, v500, fine, tool1\WObj:=wobj1;
  ENDPROC
ENDMODULE
```

ROB2 task program

```
MODULE module2
  TASK PERS wobjdata wobj2 := [ FALSE, TRUE, "", [ [500, -200,
    1000], [1, 0, 0 ,0] ], [ [100, 1200, 100], [1, 0, 0, 0] ]
    ];
  TASK PERS wobjdata wobjcnv2 := [ FALSE, FALSE, "CNV2", [ [0,0,
    0], [1, 0, 0 ,0] ], [ [0, 0, 0], [1, 0, 0, 0] ] ];
  TASK PERS tooldata tool2 := ...
  CONST robtarget p21 := ...
  ...
  CONST robtarget p24 := ...

  PROC main()
    ...
    IndependentMove;
    ...
  ENDPROC

  PROC IndependentMove()
    MoveL p21, v500, fine, tool2\WObj:=wobj2;
    WaitWObj wobjcnv2\RelDist:=10;
    MoveL p22, v500, z10, tool2\WObj:=wobjcnv2;
    MoveL p23, v500, z10, tool2\WObj:=wobjcnv2;
```

Continues on next page

13 Conveyor tracking and MultiMove

13.5 Independent movements, example UnsyncCnv

Continued

```
        MoveL p24, v500, z10, tool2\wobj:=wobjcnv2;  
        MoveL p21, v500, fine, tool2\wobj:=wobj2;  
    ENDPROC  
ENDMODULE
```

13.6 Coordinated synchronized movements, example SyncCnv

ROB1 task program

```

MODULE module1
  VAR syncident sync1;
  VAR syncident sync2;
  VAR syncident sync3;
  PERS tasks all_tasks{3} := [{"ROB1"}, {"ROB2"}, {"CONV3"}];
  PERS wobjdata wobjcnv3 := [ FALSE, FALSE, "CNV3", [ [0, 0, 0],
    [1, 0, 0, 0] ], [ [0, 0, 0], [1, 0, 0, 0] ] ];
  TASK PERS tooldata tool1 := ...
  CONST robtarget p100 := ...
  ...
  CONST robtarget p199 := ...

  PROC main()
    ...
    SyncMove;
    ...
  ENDPROC

  PROC SyncMove()
    MoveJ p100, v1000, z50, tool1;
    WaitSyncTask sync1, all_tasks;
    MoveL p101, v500, fine, tool1\WObj:=wobj1;
    SyncMoveOn sync2, all_tasks;
    MoveL p102\ID:=10, v300, fine, tool1\WObj:=wobjcnv3;
    MoveC p103, p104\ID:=20, v300, z10, tool1\WObj:=wobjcnv3;
    MoveL p105\ID:=30, v300, z10, tool1\WObj:=wobjcnv3;
    MoveC p106, p101\ID:=40, v300, fine, tool1\WObj:=wobj1;
    SyncMoveOff sync3;
    MoveL p199, v1000, fine, tool1;
    UNDO
    SyncMoveUndo;
  ENDPROC
ENDMODULE

```

ROB2 task program

```

MODULE module2
  VAR syncident sync1;
  VAR syncident sync2;
  VAR syncident sync3;
  PERS tasks all_tasks{3} := [{"ROB1"}, {"ROB2"}, {"CONV3"}];
  PERS wobjdata wobjcnv3 := [ FALSE, FALSE, "CNV3", [ [0, 0, 0],
    [1, 0, 0, 0] ], [ [0, 0, 0], [1, 0, 0, 0] ] ];

  TASK PERS tooldata tool2 := ...
  CONST robtarget p200 := ...
  ...
  CONST robtarget p299 := ...

```

Continues on next page

13 Conveyor tracking and MultiMove

13.6 Coordinated synchronized movements, example SyncCnv

Continued

```
PROC main()
...
SyncMove;
...
ENDPROC

PROC SyncMove()
MoveJ p200, v1000, z50, tool2;
WaitSyncTask sync1, all_tasks;
MoveL p201, v500, fine, tool2 \WObj:=wobj2;
SyncMoveOn sync2, all_tasks;
MoveL p202\ID:=10, v300, fine, tool2\WObj:=wobjcnv3;
MoveC p203, p204\ID:=20, v300, z10, tool2\WObj:=wobjcnv3;
MoveL p205\ID:=30, v300, z10, tool2\WObj:=wobjcnv3;
MoveC p206, p201\ID:=40, v300, fine, tool2\WObj:=wobj2;
SyncMoveOff sync3;
MoveL p299, v1000, fine, tool2;
UNDO
SyncMoveUndo;
ENDPROC
ENDMODULE
```

CONV3 task program

```
MODULE module3
VAR syncident sync1;
VAR syncident sync2;
VAR syncident sync3;
PERS tasks all_tasks{3} := [{"ROB1"}, {"ROB2"}, {"CONV3"}];
PERS wobjdata wobjcnv3 := [ FALSE, FALSE, "CNV3", [ [0, 0, 0],
[1, 0, 0, 0] ], [ [0, 0, 0], [1, 0, 0, 0] ] ];
CONST jointtarget angle_0 := [ [ 9E9, 9E9, 9E9, 9E9, 9E9, 9E9],
[ 0, 9E9, 9E9, 9E9, 9E9, 9E9] ];
...
CONST jointtarget angle_360 := [ [ 9E9, 9E9, 9E9, 9E9, 9E9, 9E9], [
360, 9E9, 9E9, 9E9, 9E9, 9E9] ];
PROC main()
...
SyncMove;
...
ENDPROC

PROC SyncMove()
MoveExtJ angle_neg20, vrot50, fine;
WaitSyncTask sync1, all_tasks;
! Wait for the robots
WaitWObj wobjcnv3;

SyncMoveOn sync2, all_tasks;
MoveExtJ angle_20\ID:=10, vrot100, fine;
```

Continues on next page

```
WaitWObj wobjcnv3\RelDist:=100;  
MoveExtJ angle_160\ID:=20, vrot100, z10;  
MoveExtJ angle_200\ID:=30, rot100, z10;  
MoveExtJ angle_340\ID:=40, rot100, fine;  
SyncMoveOff sync3;  
DropWObj wobjcnv3;  
UNDO  
SyncMoveUndo;  
ENDPROC  
ENDMODULE
```

13.7 Motion principles

Robot speeds

When the movements of several robots are synchronized, all robots adjust their speed to finish their movements simultaneously. This means that the robot movement that takes the longest time will determine the speed of the other robots.

13.8 Combining synchronized and un-synchronized mode

Introduction

For a combination of synchronized and un-synchronized mode is needed with a single conveyor there must be two mechanical units for un-synchronized mode. For example CNV1 and CNV2 can be connected to the same encoder. CNV3 can be configured using the same I/O signals as CNV1 in the topic *Process*. Replace default C2xx signals name with C1xx (that is, *position_signal c2position* becomes *position_signal c1position*).

Configuration

The configuration file Proc.cfg will look like this:

Conveyor

```
-name "CNV1" -sensor_type "CAN" -use_sensor "CAN1"\
-adjustment_speed 250 -min_dist -600 -max_dist 20000\
-correction_vector_ramp_length 10
#
-name "CNV2" -sensor_type "CAN" -use_sensor "CAN2"\
-adjustment_speed 250 -min_dist -600 -max_dist 20000\
-correction_vector_ramp_length 10
#
-name "CNV3" -sensor_type "CAN" -use_sensor "CAN3"\
-adjustment_speed 250 -min_dist -600 -max_dist 20000\
-correction_vector_ramp_length 10
#
```

Conveyor CAN sensor

```
-name "CAN1" -eio_unit_name "Qtrack1" -connected_signal
  "c1Connected"\
-position_signal "c1Position" -velocity_signal "c1Speed"\
-null_speed_signal "c1NullSpeed" -wait_wobj_signal "c1WaitWObj"\
-drop_wobj_signal "c1DropWObj" -data_timestamp "c1DTimestamp"\
-rem_all_pobj_signal "c1RemAllPObj"\
-rem_one_pobj_signal "c1Rem1PObj"
#
-name "CAN2" -eio_unit_name "Qtrack1" -connected_signal
  "c1Connected"\
-position_signal "c1Position" -velocity_signal "c1Speed"\
-null_speed_signal "c1NullSpeed" -wait_wobj_signal "c1WaitWObj"\
-drop_wobj_signal "c1DropWObj" -data_timestamp "c1DTimestamp"\
-rem_all_pobj_signal "c1RemAllPObj"\
-rem_one_pobj_signal "c1Rem1PObj"
#
-name "CAN3" -eio_unit_name "Qtrack1" -connected_signal
  "c1Connected"\
-position_signal "c1Position" -velocity_signal "c1Speed"\
-null_speed_signal "c1NullSpeed" -wait_wobj_signal "c1WaitWObj"\
-drop_wobj_signal "c1DropWObj" -data_timestamp "c1DTimestamp"\
-rem_all_pobj_signal "c1RemAllPObj"\
-rem_one_pobj_signal "c1Rem1PObj"
```

This page is intentionally left blank

14 Troubleshooting

14.1 Overview

Introduction

This chapter details the various problem scenarios in conveyor tracking and how to troubleshoot those scenarios.

In this chapter

This chapter contains the following topics:

Topics	For more information
Event messages	Event messages on page 180
System parameters	System parameters on page 183
Robot path characteristics	Robot path characteristics on page 184
Power failure	Power failure on page 186
Collision detection	Collision detection on page 187
Technical support	Technical support on page 188

14 Troubleshooting

14.2 Event messages

14.2 Event messages

Overview

This section details the different event messages and the information required to troubleshoot them.

Event messages

50082: Path calculation time exceeded

Description

The path calculation time for mechanical units running in motion planner `arg1` exceeds internal limit. The motion task did not execute within its time limit.

Cause

The CPU load is high. For example, due to frequent EIO communication.

Recommended actions

The following are the recommended actions:

- Set the system parameter `High Interpolation Priority` for the affected motion planner.
- Try to reduce the CPU load by one or more of the following actions:
 - Reduce speed.
 - Change `AccSet`.
 - Avoid singularity (`SingArea\Wrist`).
 - If the error occurs immediately after the start from finepoint, then increase the configuration parameter `Interpolation Buffer Startup Adjust` in the topic `Motion` and type `Motion Planner`.
- If using release 6.02.01 or 6.03.00 and the motion parameter `optimized_start_from_finepoint` is set to 1, then set the parameter to 0.

50163: Position adjustment

Description

External position adjustment too large. TCP speed, orientation speed, or external position speed exceed allowed robot performance.

Cause

The following are the probable causes:

- Dynamic limitations of the robot.
- Change in the robot's direction at high speed.

Recommended actions

The following are the recommended actions:

- Reduce the programmed TCP and orientation speeds.
- Modify the path.
- Ensure the program `WaitWObj` is closer to sync.
- Run in AUTO.

Continues on next page

- Tune adjustment_accel.
- Verify that the system parameter adjustment speed has a correct value, see [Type Conveyor systems on page 96](#). If the value is correct and the problem remains, increasing it carefully within the recommended boundaries may help.

**Note**

Start tracking in zone with change of robot's direction. Avoid starting the robot with a movement that is opposite in direction to the conveyor's movement.

50366: Reference error**Description**

An error has occurred in the reference calculation in motion planner arg1. Internal status arg2. As a consequence, the controller goes to Motors Off.

Cause

The following are the probable causes:

- The robot path corrections are high.
- The programmed TCP speed is high.
- The robot is closely tracking the conveyor to a singularity. This usually happens in six axis robots when axis 4 and 6 is in line. Often seen in applications with roof hanging robots.
- The performance is reduced by the use of AccSet or PathAccLim instructions.
- The corvec ramp is too short creating high acceleration.

Recommended actions

The following are the recommended actions:

- Check the error logs for previous errors that could be causing the present problem.
- Restart the program after moving the program pointer.
- Restart the controller.
- Set the robot speed slightly higher in RAPID.
- Start tracking after MoveJ with big zone. Add a short linear movement.

50376: Geometric interpolation failed**Description**

Task. arg1.

Failed to interpolate the desired geometry.

Program ref. arg2.

(Internal code: arg3).

Recommended actions

Increase the zone size, move the programmed point, change the tool orientation or change the interpolation method.

Continues on next page

14 Troubleshooting

14.2 Event messages

Continued

50496: Conveyor Tracking position error at pick

Description

Actual TCP position for robot arg1 is too far away from the ordered position on conveyor arg2 due to ramping. Position error: arg3. As a consequence, the robot may miss the picking or placing.

Cause

The ramping of correction is not finished when reaching pick position.

Recommended actions

The following are the recommended actions:

- Increase the distance between pick and place positions to ensure that the ramping is finished.
- Reduce programmed speed.
- Reduce ramping length. Set the configuration parameter `Start_ramp` and `Stop_ramp` in **Process > Conveyor systems > parameters**.
- Increase the **max allowed position error** at pick or place position by setting the configuration parameter `Max_tracking_error_at_pick_pos`.

50024: Corner path failure

Description

Corner path executed as stop point due to some of the following reasons:

- Time delay.
- Closely programmed points.
- System requires high CPU load.

Recommended actions

The following are the recommended actions:

- Edit `MOC.cfg` and add in *motion_planner*:
`-optimized_start_from_finepoint 0 -ipol_buffer_startup_adj 2 -ipol_margin_for_servo_queue_time 0.4.`
- Use programmed speed that is close to what the robot can do, for example do not use `vmax`.
- Use `corvec_correction_level 1` instead of 3. This should also help if the conveyor is running in constant speed.

14.3 System parameters

Overview

This section describes the troubleshooting steps for system parameters used in conveyor tracking.

Configuration process

As part of the configuration process (PROC.cfg), the following parameters should be set to their corresponding values:

- `adjustment_speed`: This parameter should be set to a value according to the description in [Type Conveyor systems on page 96](#).
- `adjustment_accel`: This parameter is not used by default, but can be tuned depending on the type of the robot. Suggested values for IRB340 is 50000 and for IRB660 is 7000.
- `Acc_dependent_filter_value` and `sync_filter_speed_ratio`:
The parameter `Acc_dependent_filter_value` is used by default (1). For accelerating or indexing conveyors, it is recommended to turn it off (0). For the same applications, `sync_filter_speed_ratio` should be set to 0.0001.

14 Troubleshooting

14.4 Robot path characteristics

14.4 Robot path characteristics

Overview

This section describes the troubleshooting steps for robot path characteristics used in conveyor tracking.

Changes to TCP speed

Avoid making major changes to the speed during conveyor tracking, especially in case of larger robots.



Note

The programmed path should be close to the actual performance of the robot, as a smooth path results in a path with better accuracy.

Changes to direction

- Avoid changing the direction of the robots at high speed.
- Avoid start tracking in zone with change of the robot direction.
- Avoid starting the robot's movement in opposite to the direction of the conveyor movement.

Changes to ramp length

The robot needs to ramp the conveyor tracking path corrections. During ramping, the robot is not in the correct position in relation to the conveyor. The ramp length can be tuned using *Start ramp* and *Stop ramp* in the process configuration.

- `correction_vector_ramp_length`: Increasing the value of this process parameter (`PROC.cfg`) smoothens the movement but will also increase the distance between the robot's position in relation to the moving conveyor. Use a long ramp to reduce the mechanical stress in robot, and ensure to have a longer path to the pick or place position.
- `correction_vector_stop_ramp`: Increasing the value of this process parameter (`PROC.cfg`) smoothens the movement at the end of tracking but can affect the start tracking movement on next conveyor when switching conveyor in zone.

Programming TCP speed

If the robot does not reach this speed, then avoid programing it to the maximum speed (`VelSet 100,10000;`). Setting a speed that is close to the actual speed of the robot results in a smoother path and a better accuracy. The actual robot TCP speed can be viewed in Robotstudio signal analyzer.

If Robotstudio is not used, then the recommendation for IRB360 1.13 models is `VelSet 100,7000` and for IRB360 1.6 and 1.8 models, it is `VelSet 100,6000`; For other models of robot, use `VelSet 100,5000` in case of heavy load.

If Pickmaster is used, then verify that the RAPID variable low speed is really low in `InitSpeed` routine `MaxSpeed.v_tcp:=Vtcp`; `LowSpeed.v_tcp:=Vtcp/3`;

Continues on next page

Ensure that low speed is set for vertical movements and between picks in case of multiple picks.

Jerky movements

Jerky or rough robot movements are usually due to major path corrections when ramping out or when switching conveyors or during stop tracking.

Check the program with high TCP speed on vertical movements and eventually limit acceleration by using the PathAccLim instruction.

Low value on configuration parameter adjustment_speed can cause jerky movements. Verify that the system parameter adjustment speed has a correct value, see [Type Conveyor systems on page 96](#). If the value is correct and the problem remains, increasing it carefully within the recommended boundaries may help.

I/O problems

During tracking, if the robot controller loses I/O connection to the encoder interface, then a sys_stop occurs. To jog the robot, you need to change the Wobj program as the current one will be on the conveyor. Instead of trying to regain a position on the conveyor, it is better to move the program pointer to the routine Main (PP to Main) to clear the current path.

14 Troubleshooting

14.5 Power failure

14.5 Power failure

Description

During tracking, if the power fails, then the conveyor loses the connection. After a restart, the previous path is not recalculated. Move the program pointer to the routine main (PP to Main).

14.6 Collision detection

Description

After a collision is detected, the movement is retracted using the joint interpolation method. The movement of the robot is related to the previous joint path and not the conveyor. It is possible that the robot might move backward and collide again. It is possible to avoid the retraction movement by using the switch `\NoBackoff` in the instruction `MotionSup`.

14 Troubleshooting

14.7 Technical support

14.7 Technical support

Overview

For questions or problems with conveyor tracking, contact your local ABB Robotics Service representative, see <http://www.abb.com/>.

Index

3

3rd party software, 13

A

accelerating conveyors, 123

ActivateProfile, 138

activating conveyor, 83

additional axes, 19

additional conveyors, 79

B

base frame calibration, 67

C

circular conveyor tracking, 113

CNV1

indexing conveyor, 144

CnvGenInstr, 140

conveyor base frame, 67

coordinate systems, 23

CountsPerMeter, 66

counts per meter, 66

D

DeactProfile, 139

distance

maximum, 72

minimum, 72

DropWObj, 102

DSQC2000, 39

E

encoder

prerequisites, 41

selecting type, 40

F

frames, 23

H

high speed conveyors, 103

I

IndCnvAddObject, 157

indcnvdata, 153

IndCnvDisable, 155

IndCnvEnable, 155

IndCnvInit, 154

IndCnvReset, 156

Indexing Conveyor Control

description, 143

indexing mode, 144

L

licenses, 13

limitations

Indexing Conveyor Control, 144

LoadProfile, 137

M

M7

indexing conveyor, 144

motor, 39

MultiMove, 163

independent, 163

synchronized, 163

N

network security, 12

O

object queue, 111

open source software, OSS, 13

P

principles of conveyor tracking, 20

Q

queue tracking, 109

queue tracking distance, 66

QueueTrckDist, 66

R

RecordProfile, 133

robot adjustment speed, 74

S

software licenses, 13

speed

robot, 74

start window, 71

StoreProfile, 136

Sync Separation, 71

T

track motion, 19, 76

U

UseAccProfile, 126

W

WaitAndRecProf, 134

WaitWObj, 99



ABB Robotics Sweden AB
S-721 71 VÄSTERÅS, Sweden
Telephone +46 10-732 50 00

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

ABB Robotics (Shanghai) Limited
No. 39, 99 Miaoqiao Road,
Pudong New Area,
Shanghai, 201315, China
Telephone: +86 21 6105 6666

ABB Robotics LLC
1250 Brown Road
Auburn Hills, MI 48326
USA
Telephone: +1 248 391 9000

abb.com/robotics