ROBOTICS

Application manual

Tracking with optical sensors

Trace back information:
Workspace Main version a648
Checked in 2025-02-24
Skribenta version 5.6.018

Application manual
Tracking with optical sensors

Document ID: 3HAC095098-001
Revision: A

© Copyright 2004-2025 ABB. All rights reserved.
Specifications subject to change without notice.

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.

© Copyright 2004-2025 ABB. All rights reserved.
Specifications subject to change without notice.

Table of contents

© Copyright 2004-2025 ABB. All rights reserved.

This page is intentionally left blank

Overview of this manual

About this manual

This manual contains instructions for installing and configuring a RobotWare system using different types of optical sensors for tracking or searching.

Usage

This manual shall be used during:

- installation
- programming

> **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.

Who should read this manual?

This manual is intended for:

- installation personnel
- programmers

Prerequisites

A maintenance/repair/installation craftsman working with an ABB robot shall:

- be trained by ABB and have the required knowledge of mechanical and electrical installation/repair/maintenance work.
- be trained to respond to emergencies or abnormal situations.

References

Documentation referred to in the manual, is listed in the table below.

Document name	Document ID
Operating manual - OmniCore	*3HAC065036-001*
Application manual - Controller software OmniCore	*3HAC066554-001*
Technical reference manual - RAPID Overview	*3HAC065040-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC065038-001*
Technical reference manual - System parameters	*3HAC065041-001*
Operating manual - RobotStudio	*3HAC032104-001*
Application manual - RobotWare add-ins	*3HAC070207-001*
Application manual - ArcWare for OmniCore	*3HAC084370-001*
Application manual - MultiMove	*3HAC089689-001*

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

Continued

Revisions

Revision	Description
A	First edition.

© Copyright 2004-2025 ABB. All rights reserved.

1 Introduction

1.1 About use of optical sensors

Introduction

Optical sensors can be used for ABB robots to track along a welding seam. They can be an alternative when through-the-arc tracking is not available or possible.

Prerequisites

Hardware

- Complete robot with FlexPendant.
- Optical sensor that is supported by *ArcSeamTracking for OmniCore*, see *Limitations*.
- Calibration plate for sensor calibration.
- PC
- Ethernet cable to connect the PC to the OmniCore controller.

Software

- RobotWare 7.17 or later
- ArcWare for OmniCore1.1.3 or later
- ArcSeamTracking for OmniCore 1.1.0 or later
- License: *Arc Welding Premium+*
- Software to configure the Optical Sensor (provided by the sensor supplier)

For more information about RobotWare Add-Ins and options, see *References on page 7*.

Limitations

The following limitations apply:

Supported sensors

The hardware is acquired from optical tracking sensor suppliers, for example *ServoRobot*, *Scansonic*, *Meta*, etc.

The sensors have to support one of the protocols provided by the ABB controller software.

The only supported interface for optical sensors is TCP/IP on Ethernet.

Supported protocols

The following protocols are supported by OmniCore:

- LTAPPTCP: This is an ABB protocol that has to be implemented by sensor suppliers other than ServoRobot, so their sensors can be used together with an ABB robot.
- SOCKDEV: This is an implementation of the SERVO-ROBOT proprietary protocol (RoboCom Light version E04) used by that company’s sensors. Only functionality for tracking and data according to the constants listed in section *Constants on page 65* are implemented.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

High accuracy

For applications with a high demand of accuracy it is recommended to use robots with the option *Absolute Accuracy* together with the optical sensors.

MultiMove

Up to two robots using optical sensors for tracking simultaneously are supported in a MultiMove system.

> **Note**
>
> It may be possible, but not supported by ABB, to install and run more than two tracking robots in the same system. Such an application must be individually tested by the customer under the customer's own responsibility.

Glossary

Term	Explanation
LTAPPTCP	Laser Tracker Application Protocol for Ethernet
RW	RobotWare
TCP	Tool Center Point

© Copyright 2004-2025 ABB. All rights reserved.

1.2 Sensors

Introduction

Two different types of laser sensors are supported:

- 2.5 dimensional sensors: y, z, Rx
- 6 dimensional sensors: x, y, z, Rx, Ry, Rz

For tracking the sensor must be mounted on the robot in front of the tool/TCP in path/seam direction of the path/seam that is to be tracked. The sensor shall be mounted in such a way that it gets measurements of the seam and the tracked path is, on average, at the sensor's optimal distance.

Sensor look ahead limits

The sensor look ahead depends on the robot speed and the sensor frequency. A typical distance is 30 mm or more.

- Both high TCP speed and low sensor frequency demand longer look ahead. This limit is also influenced by internal delays in the controller. The delay is, as a rule of thumb, about 0.5 s, but it is necessary to adjust the sensor mounting in each specific installation to be sure the tracking works correctly. The following calculation gives a rough estimate for the lower limit of the sensor's look ahead:

  *speed * ((1 / sensor frequency) + controller delay)) < look ahead*
- With low TCP speed and/or high sensor frequency and/or large look ahead the robot controller’s internal measurement buffer (max. 200 measurements) gives an upper limit for the look ahead of the sensor:

  *look ahead < ((200 * speed) / sensor frequency)*

2.5D sensors

The 2.5D sensor is the most common type. These sensors use one laser stripe for measurement. They can measure y and z as well as the angle around x (path direction).

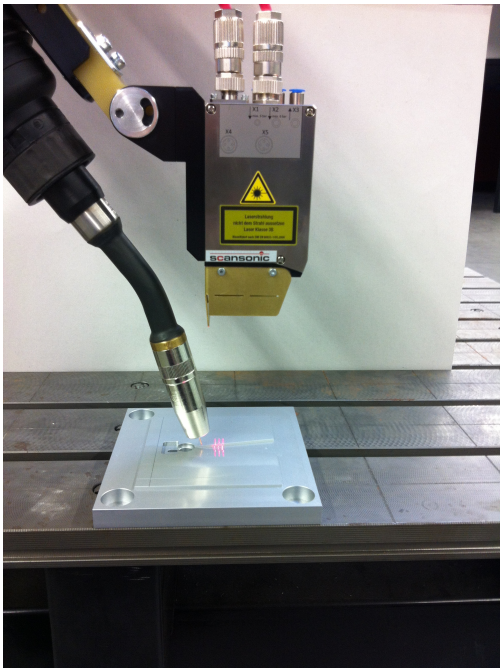
Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

xx1500001675

6D sensors

The 6D sensors can measure both position (x, y, z) and all three Euler angles.



xx1500001676

© Copyright 2004-2025 ABB. All rights reserved.

1.3 Product options

1.3.1 *Optical sensor based SeamTracking*

Purpose

The option *Optical sensor based SeamTracking* is used to integrate seam tracking with the welding instructions `ArcL`, `ArcC`, etc. If the optional argument `\Track` is used, these instructions handle both, the communication with the sensor and tracking.

What is included

The option *Optical sensor based SeamTracking* installs the following functionality:

- ABB supported sensor protocols.
- Instruction used to set up interrupt, based on input from the sensor: `IVarValue`.
- Instruction used to write to a sensor: `WriteVar`.
- Function for reading from a sensor: `ReadVar`.
- Calibration functionality for optical sensors.
- Switch to enable tracking in the welding instructions (`ArcL`, `ArcC`, etc.): `\Track`
- Data type to specify how the tracking should be performed: `trackdata`
- System parameters to configure sensor and sensor properties

  Process topic:
  - Sensor type
  - Physical sensor
  - Calibration Plate type
  - Optical Sensor
  - Optical Sensor Properties

  Communication topic:
  - Sensor Device

© Copyright 2004-2025 ABB. All rights reserved.

This page is intentionally left blank

2 Installation

2.1 Installing the sensor

Interfaces

The Ethernet networks used by OmniCore are distributed into the following segments:

Network segment	C30	C90XT V250XT Type A	V250XT Type B V400XT	E10	Usage
Private Network	I/O (Scalable I/O) ETHERNET SWITCH	I/O (Scalable I/O) ETHERNET SWITCH	DEV	DEVICE	Process equipment local to this specific robot.
	MGMT (Management)	MGMT (Management)	MGMT (Management)	MGMT (Management)	ABB service personnel.
	HMI (FlexPendant)	HMI (FlexPendant)	HMI (FlexPendant)	HMI (FlexPendant)	FlexPendant connection.
ABB Connect Network	ABB Connect	ABB Connect	ABB Connect	WAN 2	ABB Connected Services connection.
Public Network	WAN	WAN	WAN 1	WAN 1	Public/factory network.
			WAN 2		
I/O Network	LAN	LAN3	LAN	-	Secondary public/factory network. Isolated from WAN.

Note

For information regarding location of the Ethernet port connectors, see the Product manual for the respective OmniCore controller.

The optical sensor controller has to be connected to the controller's *Private Network*.

Ethernet communication

For more information about the basic configuration of the Ethernet ports, see section *Communication* in *Technical reference manual - System parameters*.

© Copyright 2004-2025 ABB. All rights reserved.

2.2 Software installation

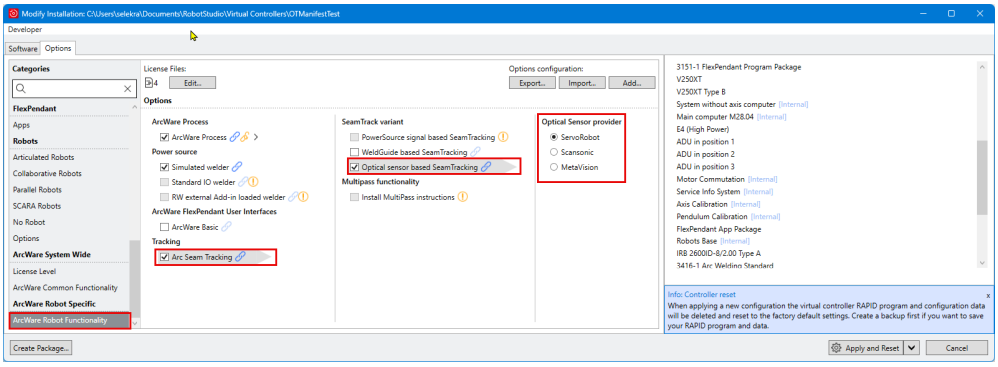
Installing *Optical sensor based SeamTracking*

Use *Modify Installation* in *RobotStudio* to configure a robot system. For more information see *Operating manual - RobotStudio*.

RobotWare 7.17 or later, *ArcWare for OmniCore* and *ArcSeamTracking for OmniCore* are required.

Make sure that the checkbox *Optical sensor based SeamTracking* is selected.

Select a sensor brand sub-option. When using a sensor brand that is not available, select the one that comes closest and modify the configuration when the robot is up and running.



xx2500000083

After installation, *Optical sensor based SeamTracking* needs to be configured, see *Configuration on page 17*.

© Copyright 2004-2025 ABB. All rights reserved.

3 Configuration

General

Some system parameters describe the properties of the optical sensor and the calibration plate:

- Sensor Type (Process), see *Configuring Sensor Type on page 20*.
- Calibration Plate type (Process), see *Configuring Calibration Plate type on page 27*.

These parameters are only used during calibration of the sensor.

Other parameters describe how the robot can access the sensor:

- Sensor Device (Communication), see *Configuring Sensor Device on page 18*.
- Physical Sensor (Process), see *Configuring Physical Sensor on page 24*.

To be able to use the sensor with the *ArcWare for OmniCore* option *Optical sensor based SeamTracking*, the properties of *Optical Sensor*, and *Optical Sensor Properties* have to be set, see *Configuring Optical sensor based SeamTracking on page 28*. The product installation provides default configurations that might require modifications to match the actual installation.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

3.1 Configuring Sensor Device

General

The *Sensor Device* holds all necessary information to enable the robot controller to access the optical sensor:

- *Type*: defines the communication protocol to be used to communicate with the sensor controller.
- *Remote Address*: holds the IP address of the sensor controllers robot communication port.
- *Remote Port Number*: holds the IP port on which the sensor controller accepts communication connections.

Optical sensor based SeamTracking supports a maximum of six sensors using either the protocol *SOCKDEV* or *LTAPPTCP*.

Optical sensor based SeamTracking installs default configurations of *Sensor Devices* for each robot in the system, for which the option is selected. The default values need to be reviewed and adapted to match the actual robot system.

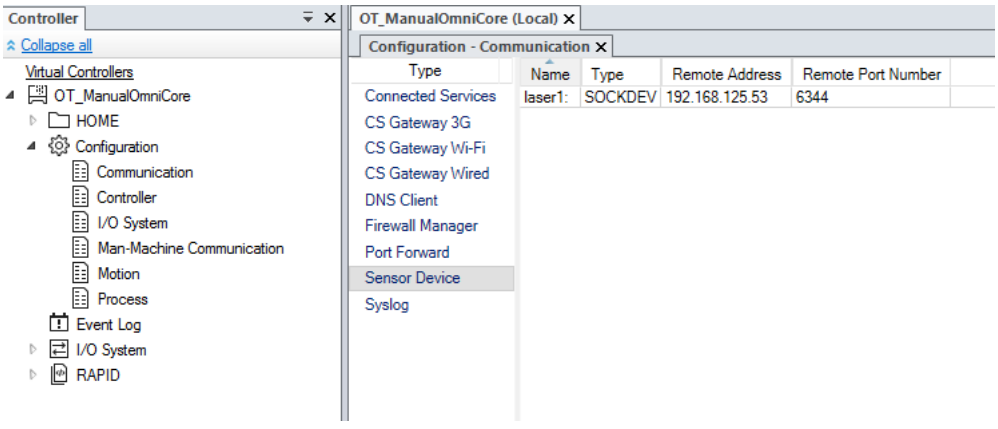
For *SOCKDEV* the installed default value for *Remote Port Number* is 6344 and for *LTAPPTCP* it is 5020.

The sensor acts as a server, the robot controller acts as a client, that is, the robot controller initiates the connection to the sensor.

For information about the supported sensors and protocols, see *Limitations on page 9*.

Configuration on the robot controller side

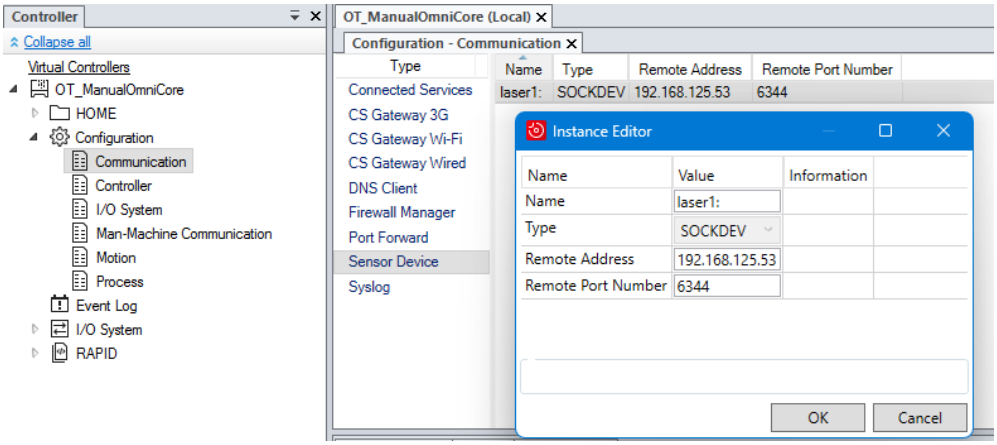
Start RobotStudio and select **Configuration** > **Process** > **Sensor Device** to edit an existing or add a new *Sensor Device*



xx2500000084

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.



xx2500000085

Configuration on the sensor controller side

For configuration of the sensor controller, see the sensor supplier’s installation and user manual.

© Copyright 2004-2025 ABB. All rights reserved.

3.2 Configuring Sensor Type

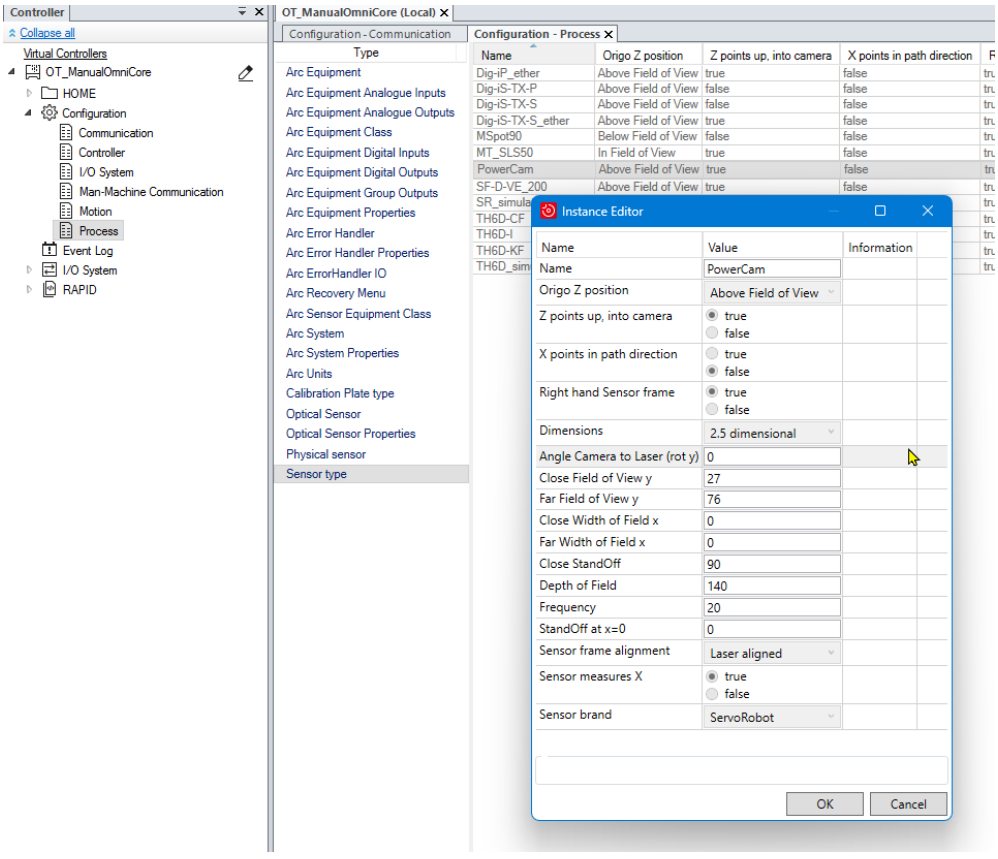
General

Each instance of the *Sensor Type* configuration describes the properties of a specific optical sensor model. Each *Physical Sensor* refers to a *Sensor Type*, and it is crucial for the calibration of the *Physical Sensor* that these data are correct.

Optical sensor based SeamTracking installs a number of *Sensor Type* instances. It is also possible to add new *Sensor Type* instances, which requires a data sheet for that specific type. The data sheet has to be provided by the sensor company. With RobotStudio the configuration data can be entered directly, see *RobotStudio on page 20*. The FlexPendant application **Sensor Setup** provides a wizard, see *FlexPendant on page 20*.

RobotStudio

In RobotStudio the cofiguration editor can be used to configure a new instance of the type *Sensor Type*. On the **Controller** tab, select **Configuration** > **Process** > **Sensor Type** and add a new *Sensor Type*.



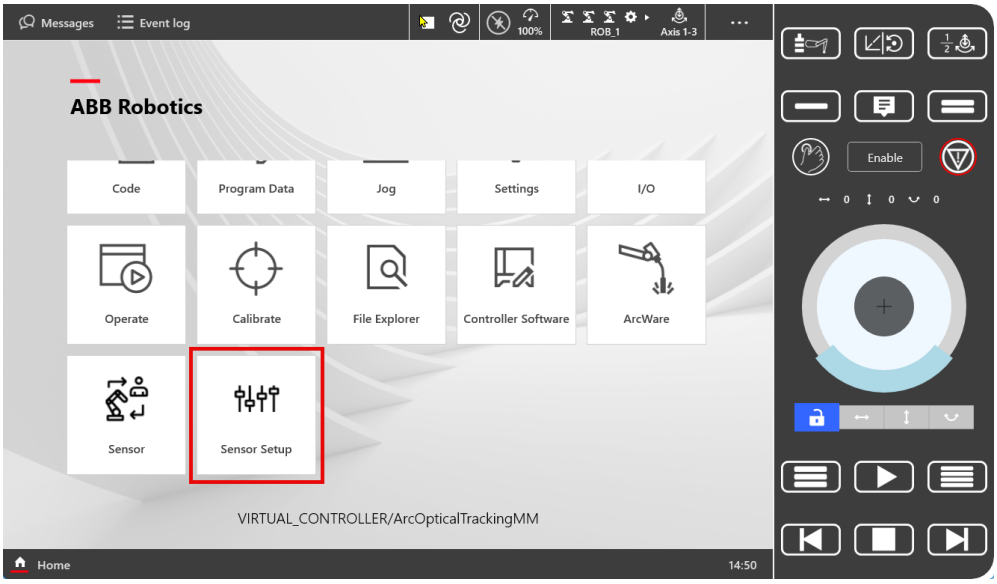
xx2500000090

FlexPendant

Use the data sheet for the new *Sensor Type* instance and start the FlexPendant application **Sensor Setup**.

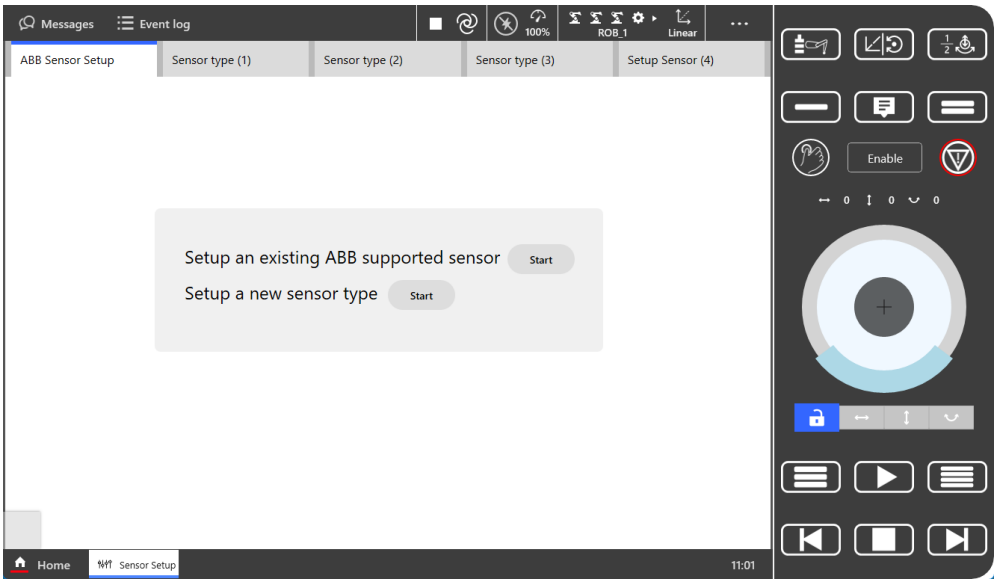
Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.



xx2500000103

Tap **Setup a new sensor type.**



xx2500000092

Select the alternative that fits the new sensor type:

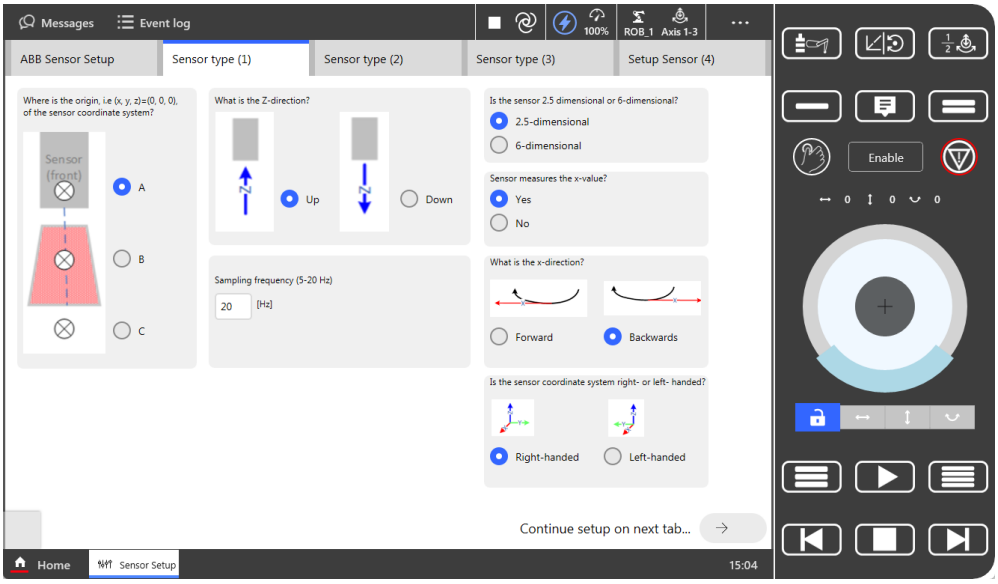
- Position of the sensor coordinate system
- Z-direction of the sensor coordinate system
- Sampling frequency - the default of 20 Hz will fit in most cases.
- Sensor measured dimensions (2.5 means Y, Z, and Rx)
- Is the X-component measured by the sensor?
- X-direction of the sensor coordinate system
- Right or left handed?

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

3.2 Configuring Sensor Type

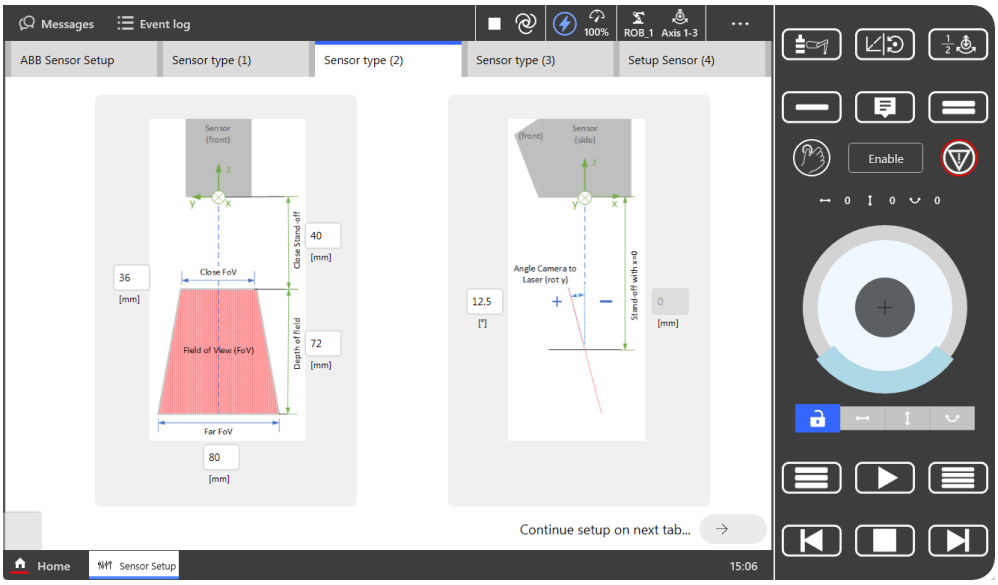
Continued



xx2500000094

Provide the data defining the Field-of-View:

- Close Stand-off
- Close Field-of-View
- Depth of field
- Far Field-of-View
- Angle between laser and Z-direction of the sensor coordinate system
- For sensors that do not measure the X-component: Stand-off where X = 0

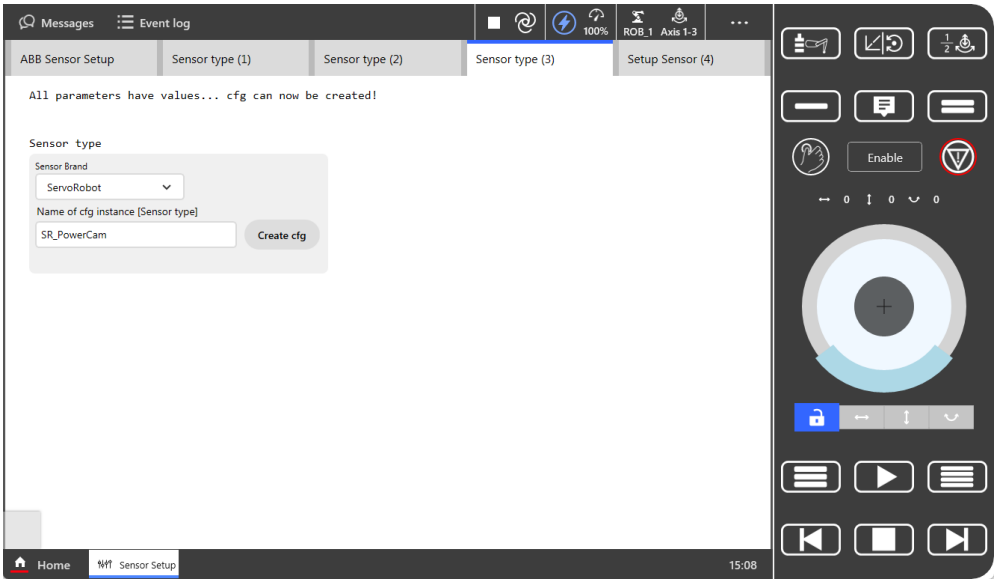


xx2500000095

Select the sensor brand and provide a name for the new sensor type. Tap **Validate Input**.

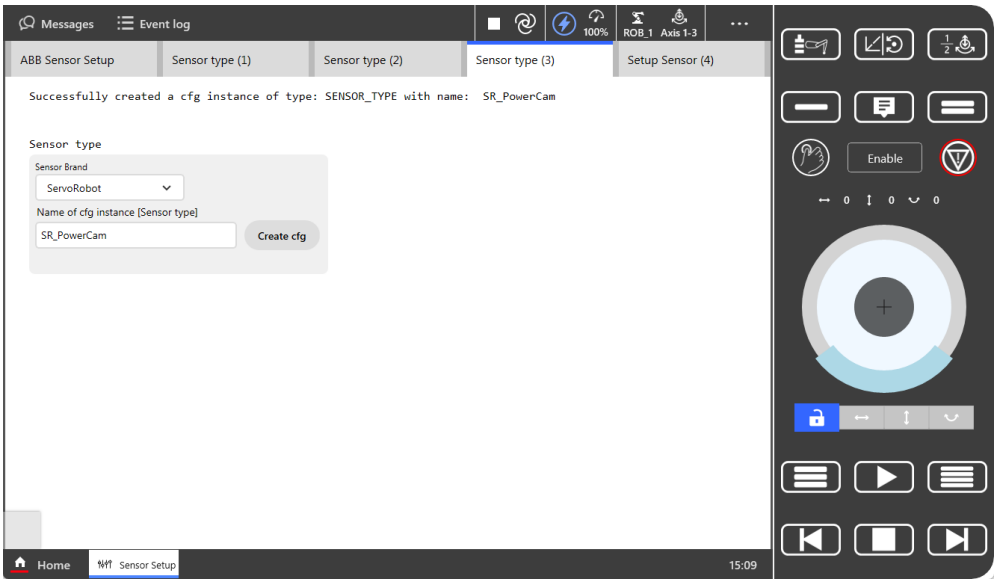
Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.



xx2500000105

If validation is OK, tap **Create cfg**.



xx2500000096

© Copyright 2004-2025 ABB. All rights reserved.

3.3 Configuring Physical Sensor

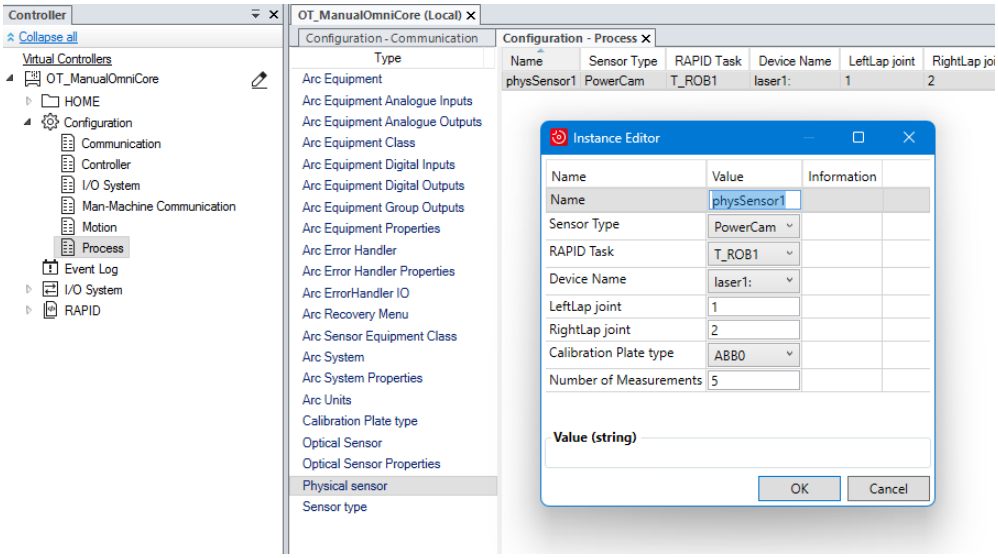
General

The configuration data of a *Physical Sensor* is used by the sensor calibration instructions, see *RAPID instructions on page 36*. This data contain all information that is necessary to perform a sensor calibration:

- *Sensor Type*: refers to an instance of type *Sensor Type*.
- *RAPID Task*: defines on which robot the sensor is mounted.
- *Device Name*: refers to the *Sensor Device* to which the sensor controller is connected.
- *LeftLap joint*: defines which tracking joint definition, defined in the sensor's controller, shall be used for measurements on the left edge of the calibration plate.
- *RightLap joint*: defines which tracking joint definition, defined in the sensor's controller, shall be used for measurements on the right edge of the calibration plate.
- *Calibration Plate type*: defines the calibration plate type that is used for the calibration of the sensor.
- *Number of Measurements*: defines how many measurements the sensor performs for each position that is measured during calibration. The value used in the calculations of the calibration is the mean value of these measurements.

RobotStudio

In RobotStudio the cofiguration editor can be used to configure a new instance of the type *Physical Sensor*. On the **Controller** tab, select **Configuration** > **Process** > **Physical Sensor** and add a new *Physical Sensor*.



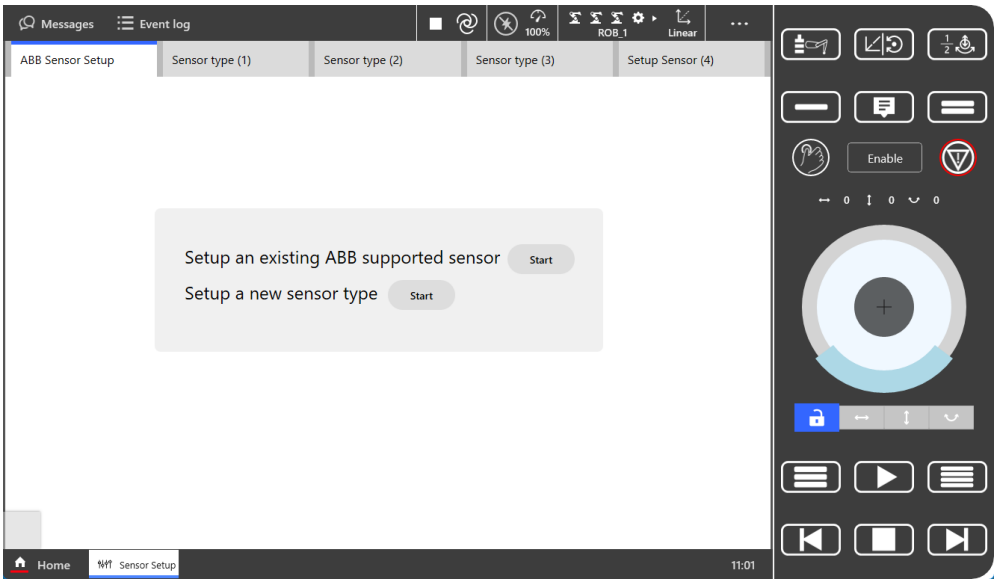
xx2500000089

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

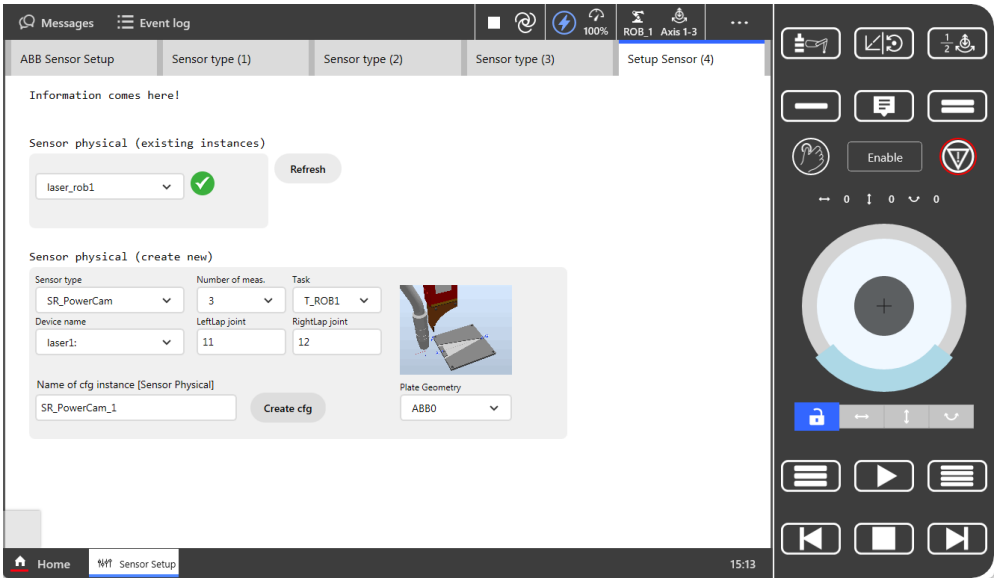
FlexPendant

Open the FlexPendant application **Sensor Setup** and select **Setup an existing ABB supported sensor**.



xx2500000092

Fill in all requested data and tap **Create cfg**.



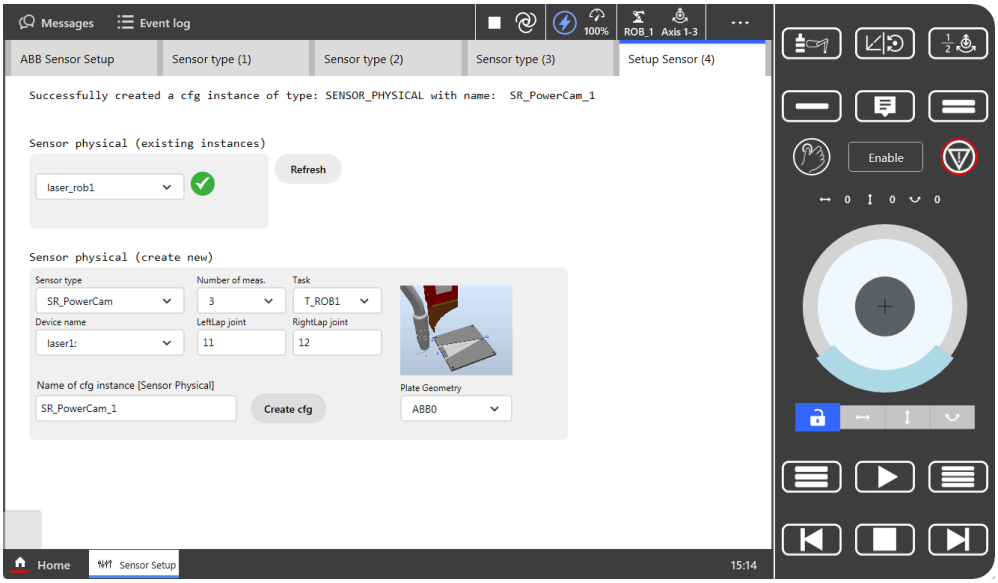
xx2500000097

A message is shown on top of the window stating if the creation of the *Physical Sensor* was successful or not.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

3.3 Configuring Physical Sensor
Continued



xx2500000098

© Copyright 2004-2025 ABB. All rights reserved.

3.4 Configuring Calibration Plate type

General

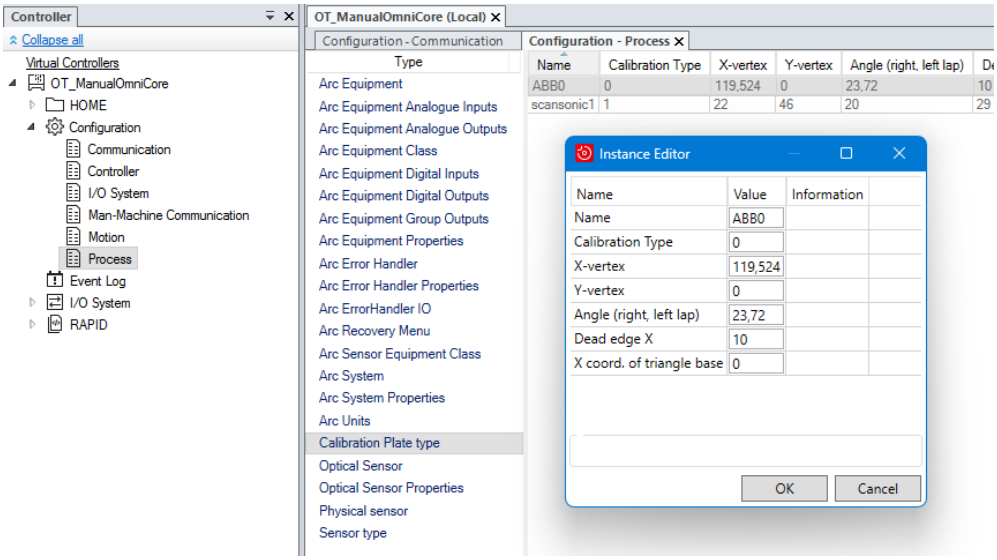
The configuration data for *Calibration Plate type* describes the geometry of a calibration plate. *Optical sensor based SeamTracking* installs two instances of the type *Calibration Plate types*:

- *ABB0*: This type of plate can be ordered from ABB.
- *scansonic1*: A plate of this type is delivered together with Scansonic sensors.

It is possible to add a new instance of the type *Calibration Plate types* in RobotStudio. The left and right lap of the plate type must be the sides of an isosceles triangle.

RobotStudio

To view or add a new instance of the type *Calibration Plate type*, start RobotStudio and select **Configuration** > **Process** > **Calibration Plate type**.



xx2500000088

© Copyright 2004-2025 ABB. All rights reserved.

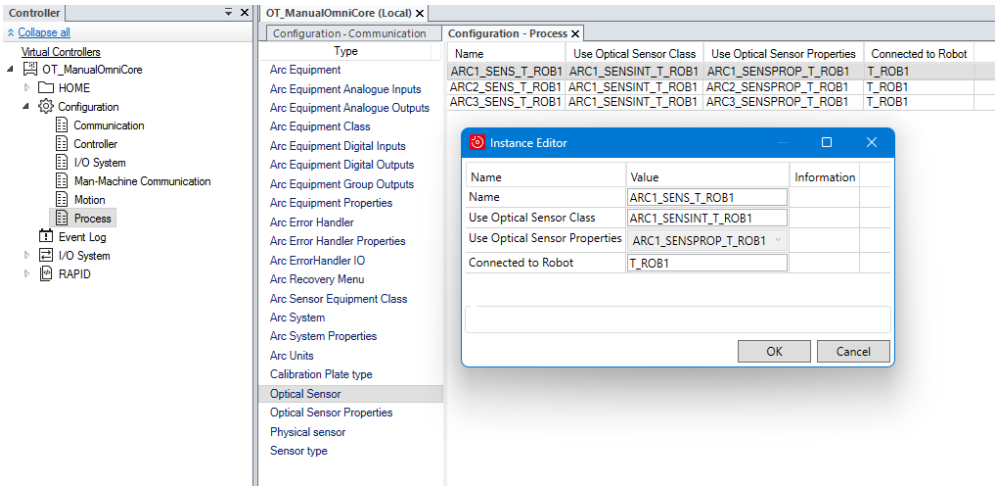
3.5 Configuring *Optical sensor based SeamTracking*

Introduction

In *Optical sensor based SeamTracking* the sensors are configured statically using the system parameter types *Optical Sensor* and *Optical Sensor Properties*. This means that the controller needs to be restarted to activate any changes.

The *Optical Sensor* data establishes the connection between the *Optical Sensor Properties*, the *Arc Sensor Equipment Class*, and the RAPID task of the robot that the sensor is to be used with.

In the example below *Optical Sensor* defines what *Optical Sensor Properties* are connected to robot 1 in the controller.

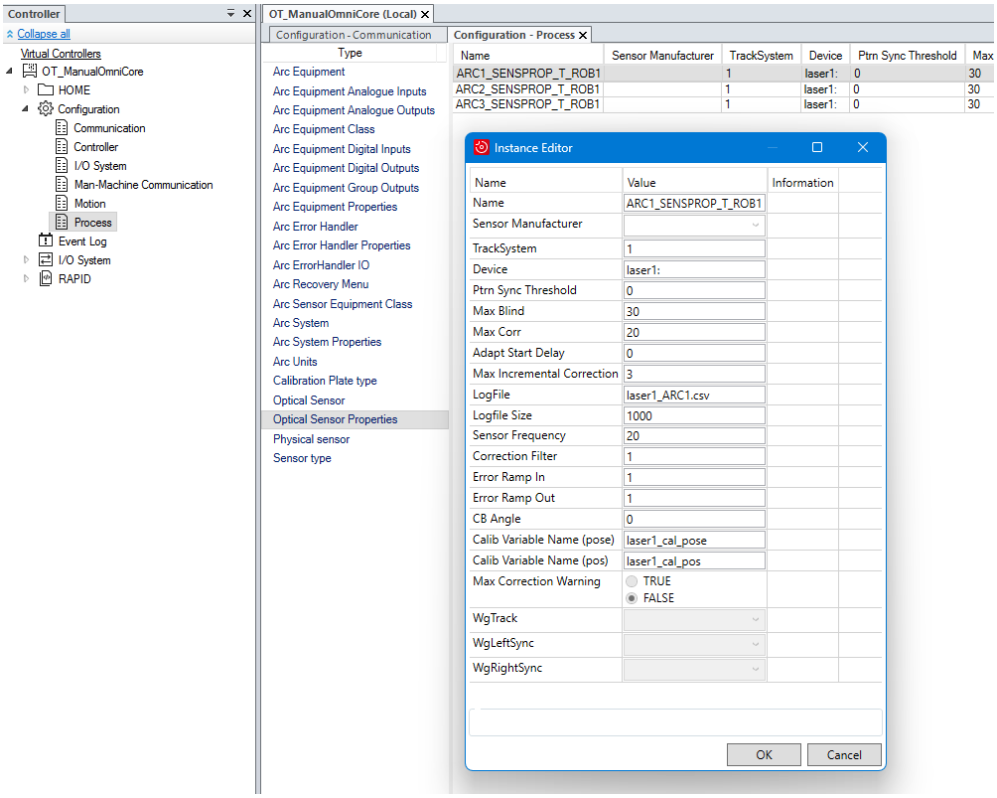


xx2500000086

The *Optical Sensor Properties* specifies a *Sensor Device*, where the sensor is connected. That device must have been set up before, see *Configuring Sensor Device on page 18*. In addition to that, it defines behavior and limits for the sensor.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.



xx2500000087

For a detailed description of the configuration parameters *Optical Sensor* and *Optical Sensor Properties*, see *Application manual - ArcWare for OmniCore*.

© Copyright 2004-2025 ABB. All rights reserved.

This page is intentionally left blank

4 Calibration

4.1 Principles of Laser Tracker Calibration

Principles

- *Calibration plate*: A plate with a well-known geometry referenced by process parameters defined in the configuration database (see *Configuration on page 17*) and a work object.
- *Sensor mounting*: The sensor has to be mounted on axis 6 of the robot. The calibration will define a tool frame for the sensor.
- *MultiMove*: It is possible to define up to 6 sensors in a robot.
- *Execution from RAPID*: Calibration is implemented as RAPID instructions that are executed in the RAPID motion task of the robot whose sensor to calibrate.

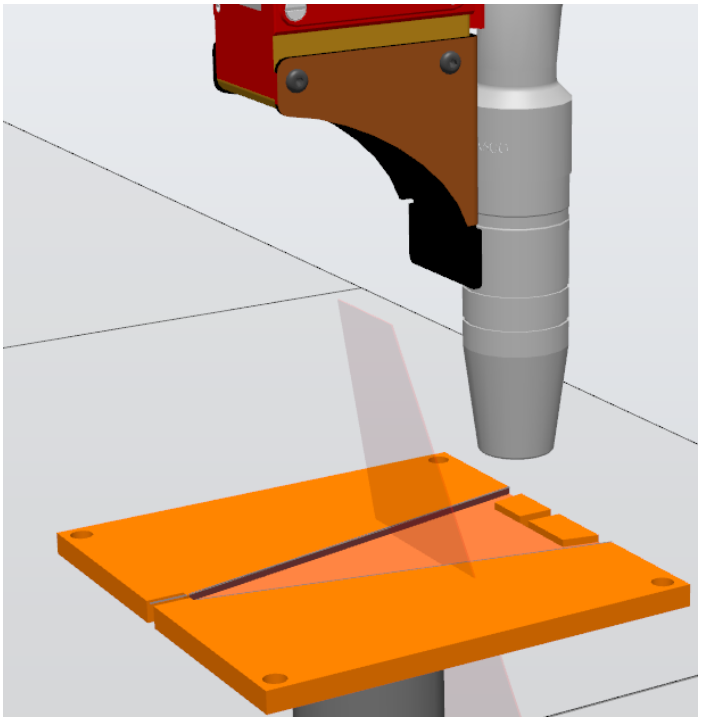
© Copyright 2004-2025 ABB. All rights reserved.

4.2 Prerequisites

Sensor prerequisites

- The sensor is physically connected to the controller and mounted on axis 6 of the robot.
- The sensor controller communication protocol is LTAPPTCP or SOCKDEV

> **Note**
>
> For best results the y-axis and z-axis of the sensor measurement frame should be aligned with that of the calibration plate in the start position.



xx2500000091

Plate prerequisites

- At least one calibration plate of type *ABB0* or *scansonic1* is mounted within the robot's work area. The plate can be mounted with any orientation including vertically or under a ceiling, but the robot work area around the plate shall be free of risk of collision. Possible errors may occur if the orientation of the plate does not allow reorientation of the robot tool during calibration and verification.
- The plate geometry is defined in the topic *Process*, in the type *Calibration Plate type*, see *Configuring Calibration Plate type on page 27*.
- For each calibration plate geometry, one left lap and one right lap joint have to be defined in the sensor controller.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

- The *tracking point* for these joints has to be located on the *upper edge* of the defined joints. It is recommended to define joints dedicated to the sensor calibration.

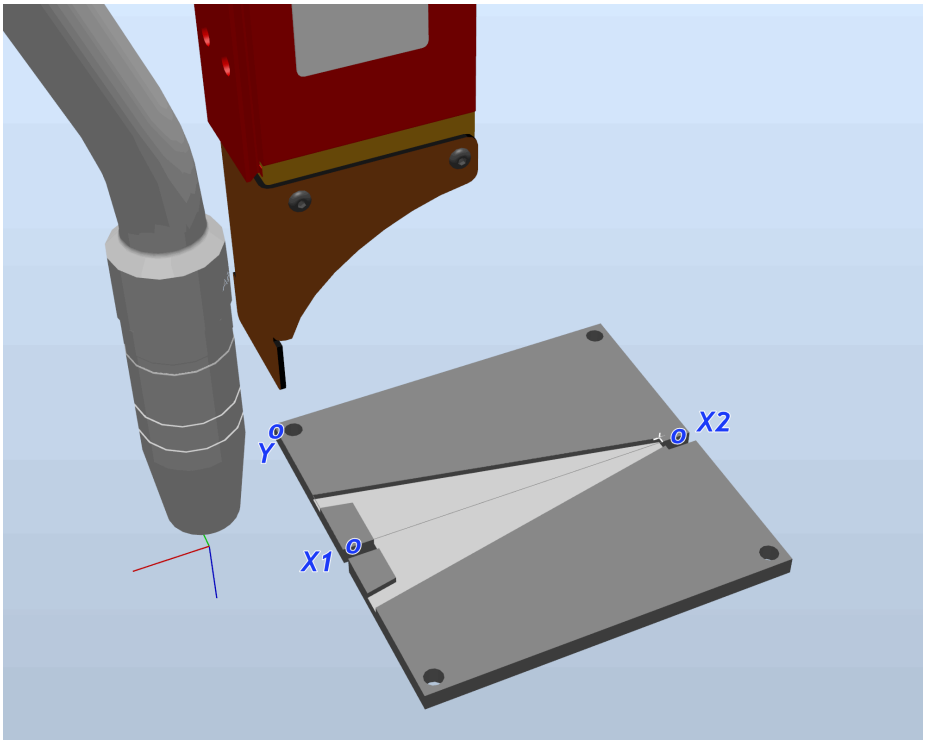
Defining a work object for the calibration plate

A work object has to be defined for the used calibration plate. Use the 3 point method to define the object frame of the work object according to the figures below. The work object is then used in the calibration instructions, see *RAPID instructions on page 36*.

> **Note**
>
> It is important to use a precisely calibrated tool, for example, using BullsEye, for the definition of the calibration plate work object to get best results in the sensor calibration, which gives best tracking results.

Placement of the 3 points for different *Calibration Plate types*

- *Calibration Plate type: ABB0*



xx2500000122

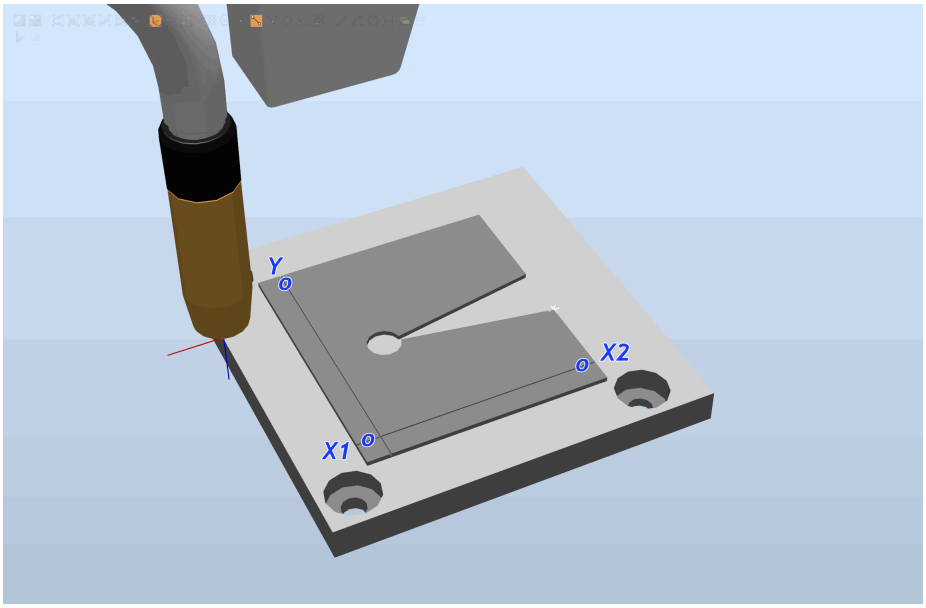
- *Calibration Plate type: scansonic1*

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

4.2 Prerequisites

Continued



xx2500000123

© Copyright 2004-2025 ABB. All rights reserved.

4.3 Using RW Arc

Required options

The following options must be included in the license for the robot:

Description	Option number
ArcWare Premium+	3416-3

The names specified in the system parameters *Pose* and *Pos* under *Process > Optical Sensor Properties* (see *Application manual - ArcWare for OmniCore*) have to hold the result of the optical sensor calibration. The easiest is to create task persistent data for both and use them in the arguments of the calibration instruction, see *RAPID instructions on page 36*.

© Copyright 2004-2025 ABB. All rights reserved.

4.4 RAPID instructions

4.4.1 MoveOptCalibL - Optical Sensor Calibration with linear approach path

Usage

`MoveOptCalibL` is used to calibrate an optical sensor and has a linear approach path. The result of the calibration is automatically stored in the persistent *pose* and *pos* variables, whose names are found in *Process > Optical Sensor Properties > Calib Variable Name (pos(e))*.

Example

```
! Result of the sensor calibration is necessary for tracking with
     ArcWare
! - the names have to match the configured names in the Optical
     Sensor Properties
TASK PERS pose laser1_cal_pose := [[0,0,0],[1,0,0,0]];
TASK PERS pos laser1_cal_pos := [0,0,0];

! Work object for the calibration plate
! - Needs to be calibrated before sensor calibration
TASK PERS wobjdata
     CalibPlate_laser1:=[FALSE,TRUE,"",[[0,0,0],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];

! robtargets need to be updated before sensor calibration
CONST robtarget p_calibApproach :=
     [[0,0,0],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p_calibStart :=
     [[0,0,0],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

! Name of the physical sensor defined in the process configuration
! - Needs to be set to the actual name before sensor calibration
CONST string physSensor :="physicalSensor1";

PROC sensorCalib(PERS tooldata tWelder)
MoveJ p_calibApproach, v1000, z50, tWelder\WObj:=CalibPlate_laser1;
MoveOptCalibL physSensor, p_calibStart, v500, fine,
     tWelder\WObj:=CalibPlate_laser1\CreateLogFile;
MoveJ p_calibApproach, v1000, fine, tWelder\WObj:=CalibPlate_laser1;
ENDPROC
```

Arguments

```
MoveOptCalibL SensorName ToPoint Speed Zone RobotTool [\WObj]
     [\TLoad] [\AlignY] [MaxMoveDistX] [\MaxMoveDistZ]
     [\CreateLogFile]
```

`SensorName`

Data type: `string`

The sensor name is the name of the sensor that is calibrated. It has to be defined as *Physical Sensor*, see *Configuring Physical Sensor on page 24*.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

4.4.1 MoveOptCalibL - Optical Sensor Calibration with linear approach path

ArcWare

Continued

`ToPoint`

Data type: `robtarget`

The destination position of the robot and additional axes. This is either defined as a named position or stored directly in the instruction (indicated by an * in the instruction).

`Speed`

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed`.

Speed data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

`Zone`

Data type: `zonedata`

Zone data defines how close the axes must be to the programmed position before they can start moving towards the next position.

In the case of a fly-by point, a corner path is generated past that position. In the case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point.

A stop point (fine) is always generated automatically at the start position of a weld and at a controlled weld end position. Fly-by points, such as z10, should be used for all other weld positions.

Weld data changes over to the next arc welding instruction at the center point of the corner path.

`RobotTool`

Data type: `tooldata`

The tool used for the calibration movement. The TCP of the tool is the point moved to the specified destination position. The z-axis of the tool should be parallel with the torch.

`[\WObj]`

Data type: `wobjdata`

The work object (coordinate system) defines position and orientation of the calibration plate. The instruction's robot position is referenced to this work object. When this argument is omitted, the robot position is referenced to the world coordinate system. It must, however, be specified if a stationary TCP or coordinated additional axes are used.

Note

The work object argument `\Wobj` has to be used as it passes the calibration plate position to the calibration. Without this work object, it is not possible to perform a calibration.

`[\TLoad]`

Data type: `loaddata`

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

4.4.1 MoveOptCalibL - Optical Sensor Calibration with linear approach path

ArcWare

Continued

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

If the \TLoad argument is set to load0, then the \TLoad argument is not considered and the loaddata in the current tooldata is used instead. For a complete description of the \TLoad argument, see *MoveL - Moves the robot linearly*.

[\AlignY]

Data type: switch

If this argument is present, the sensor is aligned to an angle of 0° around the Y-axis.

[\MaxMoveDistX]

Data type: num

If this argument is present, the movement of the calibration in work object X-direction is limited to *MaxMoveDistX* mm.

[\MaxMoveDistY]

Data type: num

If this argument is present, the movement of the calibration in work object Y-direction is limited to *MaxMoveDistX* mm.

[\CreateLogFile]

Data type: switch

If this argument is present, a log file is saved, logging the whole calibration process. The log file has a predefined name (*SensorName*_calib.log) and is placed in the data directory of the ArcSeamTracking Add-in:

ADDINDATA\ABB.ROBOTICS.APPLICATIONS.ARCSEAMTRACKING\DATA

Program execution

Motion

Robot and additional axes are moved to the destination position, which is the start position for the calibration of the optical sensor. During the calibration process the TCP of the robot is moved and reoriented above the calibration plate, but the movements that calibration demands might move outside the borders of the calibration plate in both X- and Y-direction.

Syntax

```
MoveOptCalibL
  [SensorName ':='] <expression (IN) of string>
  [ToPoint ':='] <expression (IN) of robtarget>
  [Speed ':='] <expression (IN) of speeddata>
  [Zone ':='] <expression (IN) of zonedata>
  [RobotTool ':='] <persistent (PERS) of tooldata>
  ['\' WObj ':=' <persistent (PERS) of wobjdata>]
  ['\' TLoad ':='] <persistent (PERS) of loaddata>]
  ['\' WObj ':=' <persistent (PERS) of wobjdata>]
  ['\' AlignY]
```

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

4.4.1 MoveOptCalibL - Optical Sensor Calibration with linear approach path

ArcWare

Continued

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


© Copyright 2004-2025 ABB. All rights reserved.

4.4.2 MoveOptCalibJ - Optical Sensor Calibration with joint movement approach path

Usage

`MoveOptCalibJ` is used to calibrate an optical sensor and has a joint movement approach path. The result of the calibration is automatically stored in the persistent *pose* and *pos* variables, whose names are found in *Process > Optical Sensor Properties > Calib Variable Name (pos(e))*.

Example

```
! Result of the sensor calibration is necessary for tracking with
     ArcWare
! - the names have to match the configured names in the Optical
     Sensor Properties
TASK PERS pose laser1_cal_pose := [[0,0,0],[1,0,0,0]];
TASK PERS pos laser1_cal_pos := [0,0,0];

! Work object for the calibration plate
! - Needs to be calibrated before sensor calibration
TASK PERS wobjdata
     CalibPlate_laser1:=[FALSE,TRUE,"",[[0,0,0],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];

! robtargets need to be updated before sensor calibration
CONST robtarget p_calibApproach :=
     [[0,0,0],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p_calibStart :=
     [[0,0,0],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

! Name of the physical sensor defined in the process configuration
! - Needs to be set to the actual name before sensor calibration
CONST string physSensor :="physicalSensor1";

PROC sensorCalib(PERS tooldata tWelder)
MoveJ p_calibApproach, v1000, z50, tWelder\WObj:=CalibPlate_laser1;
MoveOptCalibJ physSensor, p_calibStart, v500, fine,
     tWelder\WObj:=CalibPlate_laser1\CreateLogFile;
MoveJ p_calibApproach, v1000, fine, tWelder\WObj:=CalibPlate_laser1;
ENDPROC
```

Arguments

```
MoveOptCalibJ SensorName ToPoint Speed Zone RobotTool [\WObj]
     [\TLoad] [\AlignY] [MaxMoveDistX] [\MaxMoveDistZ]
     [\CreateLogFile]
```

`SensorName`

Data type: `string`

The sensor name is the name of the sensor that is calibrated. It has to be defined as *Physical Sensor*, see *Configuring Physical Sensor on page 24*.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

`ToPoint`

Data type: `robtarget`

The destination position of the robot and additional axes. This is either defined as a named position or stored directly in the instruction (indicated by an * in the instruction).

`Speed`

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed`.

Speed data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

`Zone`

Data type: `zonedata`

Zone data defines how close the axes must be to the programmed position before they can start moving towards the next position.

In the case of a fly-by point, a corner path is generated past that position. In the case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point.

A stop point (fine) is always generated automatically at the start position of a weld and at a controlled weld end position. Fly-by points, such as z10, should be used for all other weld positions.

Weld data changes over to the next arc welding instruction at the center point of the corner path.

`RobotTool`

Data type: `tooldata`

The tool used for the calibration movement. The TCP of the tool is the point moved to the specified destination position. The z-axis of the tool should be parallel with the torch.

`[\WObj]`

Data type: `wobjdata`

The work object (coordinate system) defines position and orientation of the calibration plate. The instruction's robot position is referenced to this work object. When this argument is omitted, the robot position is referenced to the world coordinate system. It must, however, be specified if a stationary TCP or coordinated additional axes are used.

Note

The work object argument `\Wobj` has to be used as it passes the calibration plate position to the calibration. Without this work object, it is not possible to perform a calibration.

`[\TLoad]`

Data type: `loaddata`

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

4.4.2 MoveOptCalibJ - Optical Sensor Calibration with joint movement approach path

ArcWare

Continued

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

If the \TLoad argument is set to load0, then the \TLoad argument is not considered and the loaddata in the current tooldata is used instead. For a complete description of the \TLoad argument, see *MoveL - Moves the robot linearly*.

[\AlignY]

Data type: switch

If this argument is present, the sensor is aligned to an angle of 0° around the Y-axis.

[\MaxMoveDistX]

Data type: num

If this argument is present, the movement of the calibration in work object X-direction is limited to *MaxMoveDistX* mm.

[\MaxMoveDistY]

Data type: num

If this argument is present, the movement of the calibration in work object Y-direction is limited to *MaxMoveDistX* mm.

[\CreateLogFile]

Data type: switch

If this argument is present, a log file is saved, logging the whole calibration process. The log file has a predefined name (*SensorName*_calib.log) and is placed in the data directory of the ArcSeamTracking Add-in:

ADDINDATA\ABB.ROBOTICS.APPLICATIONS.ARCSEAMTRACKING\DATA

Program execution

Motion

Robot and additional axes are moved to the destination position, which is the start position for the calibration of the optical sensor. During the calibration process the TCP of the robot is moved and reoriented above the calibration plate, but the movements that calibration demands might move outside the borders of the calibration plate in both X- and Y-direction.

Syntax

```
MoveOptCalibJ
  [SensorName ':='] <expression (IN) of string>
  [ToPoint ':='] <expression (IN) of robtarget>
  [Speed ':='] <expression (IN) of speeddata>
  [Zone ':='] <expression (IN) of zonedata>
  [RobotTool ':='] <persistent (PERS) of tooldata>
  ['\' WObj ':=' <persistent (PERS) of wobjdata>]
  ['\' TLoad ':='] <persistent (PERS) of loaddata>]
  ['\' WObj ':=' <persistent (PERS) of wobjdata>]
  ['\' AlignY]
```

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

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


© Copyright 2004-2025 ABB. All rights reserved.

This page is intentionally left blank

5 Programming

Accessing an optical sensor from RAPID

The communication with optical sensors is based on the concept of sensor variables. Each sensor supports a number of variables that give access to different functionality in the sensors. Not all predefined variables might be supported by all sensor suppliers.

For the list of sensor variables see *Constants on page 65*.

The RAPID interface to the sensor is independent of the actual sensor protocol, that is, it is the same for all sensors. But if there is a need to add new variables to get access to special functionality in a sensor, the sensors with the protocol SOCKDEV differ from all the other ones:

- For a sensor using the protocol LTAPPTCP it is sufficient that the sensor supplier implements the new functionality and makes sure that its variable number is unique.
- For a sensor using the protocol SOCKDEV, ABB has to implement the new functionality together with the sensor supplier. That is, both have to do work.

It is possible to connect to a sensor, control the sensor, and read sensor data using the following RAPID instructions, see *RAPID components for optical tracking on page 65*.

© Copyright 2004-2025 ABB. All rights reserved.

This page is intentionally left blank

6 Tracking

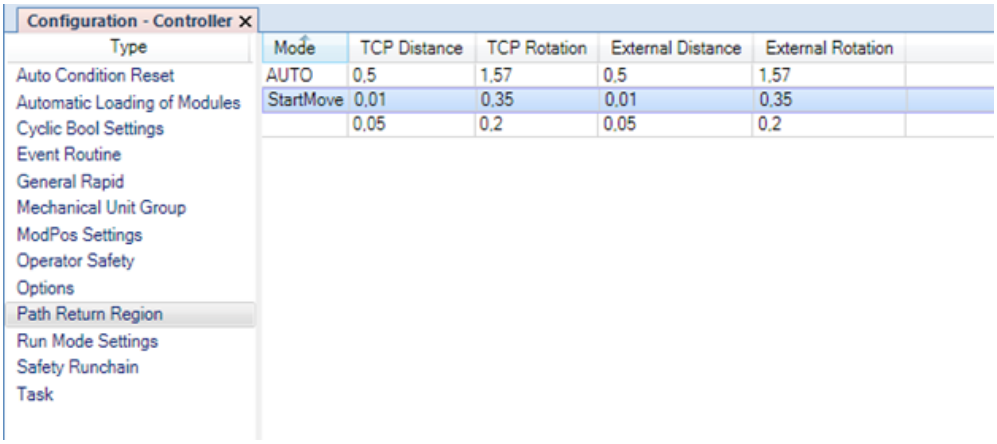
6.1 Adapting the path regain distance

Path Return Region

It might be necessary to adapt the limit of the StartMove regain distance for the TCP distance to the tracking conditions. The StartMove regain distance defines a limit above which no automatic regain will be performed. The default value is 10 mm, see the type *Path Return Region* in *Technical reference manual - System parameters*.

If the value of the path correction during tracking in some point on the tracked path is bigger than the StartMove regain distance for the TCP distance, you have to increase the configured value. You might not encounter problems as long as no process errors occur, but if such an error occurs at a point of the path, where the correction is bigger than the StartMove regain distance for the TCP distance, the movement cannot be restarted from the error handler.

Set the StartMove regain distance for the TCP distance to a value, that covers all path correction values in your system.



xx1700001353

© Copyright 2004-2025 ABB. All rights reserved.

6.2 Sensor functionality

Description

Path tracking is fully integrated into *Optical Tracking with ArcWare*.

The sensors that are used for tracking, have to provide the following functionality:

LTAPPTCP

LTAPPTCP variable	Unit	Description
LTAPP__AGE	ms	Measurement age
LTAPP__GAP	mm	Gap
LTAPP__X	mm	X-value
LTAPP__Y	mm	Y-value
LTAPP__Z	mm	Z-value
LTAPP__SUSPEND	-	Laser suspend
LTAPP__JOINT	-	Activation of sensor joint
LTAPP__UNIT	-	Resolution selection

SOCKDEV

Robo-Com command	Robo-ComLight command
AcknowledgeAll	ack
BeginJointFinding	bjf
EndMode	emo
GetCorrection	gco1
GetError	ger
GetParameter	gpa
GetStatus	gst
GetVision (no filter)	gvi1
SensorDisable	sdi

© Copyright 2004-2025 ABB. All rights reserved.

6.3 Tracking with ArcWare

Description

This functionality requires the license *Arc Welding Premium+*.

To use tracking with ArcWare welding instructions, the sensor must first be configured in ArcWare.

Tracking with ArcWare is activated by adding the optional argument `\Track` to the arc welding RAPID instructions `ArcLStart`, `ArcL`, `ArcLEnd`, etc. The `trackdata` specified with that argument defines the tracking behavior.

For more information about arc welding instructions see *Application manual - ArcWare for OmniCore*.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

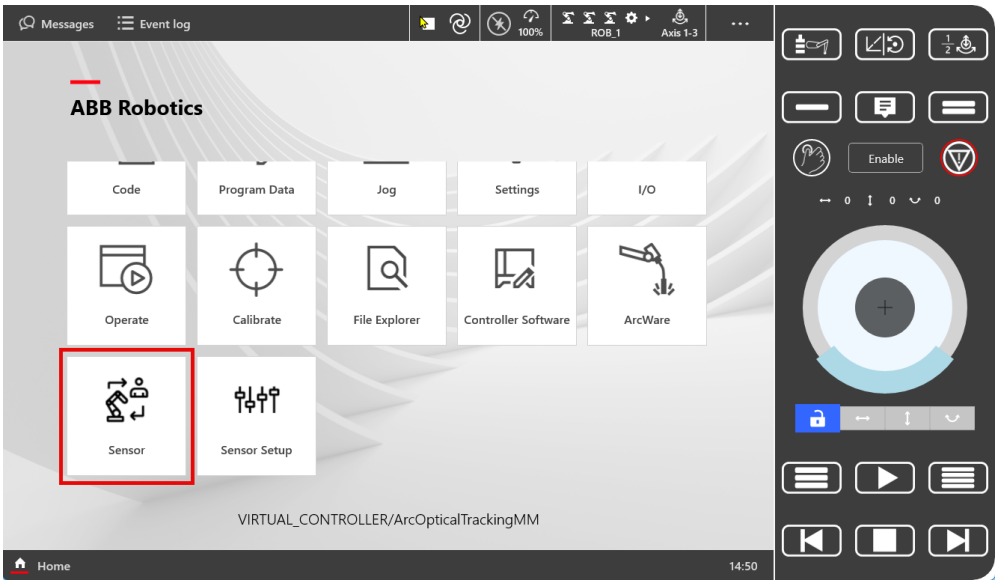
6.3.1 FlexPendant support

FlexPendant application

Start the Flexpendant application **Sensor**.

The application has support for, each function has its own window:

- Manual functions
- Data editing
- Tracklog evaluation
- Calibration status



xx2500000139

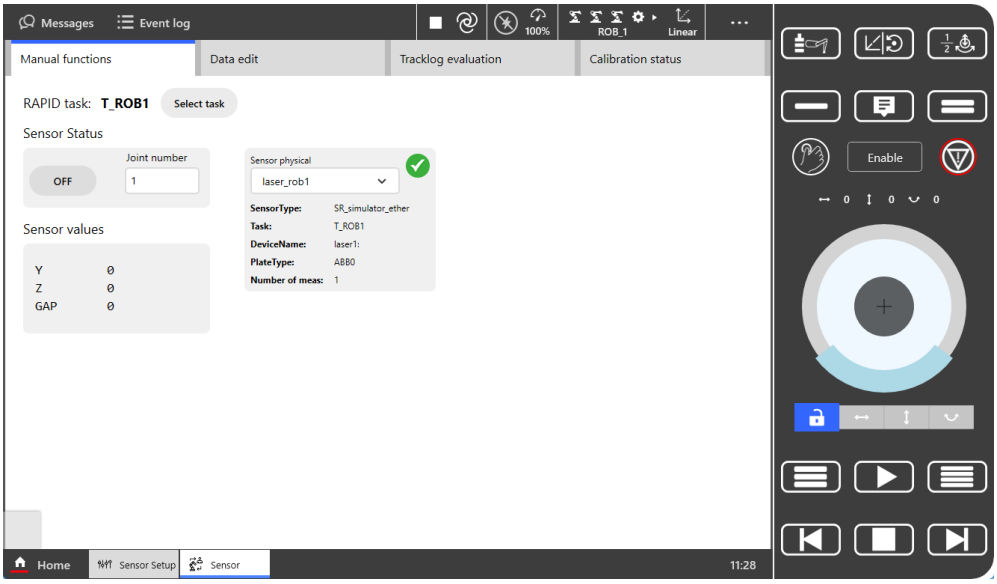
Manual functions

In this window it is possible to

- Switch on and off the sensors
- Change the active joint
- Monitor the sensor readings

Continues on next page

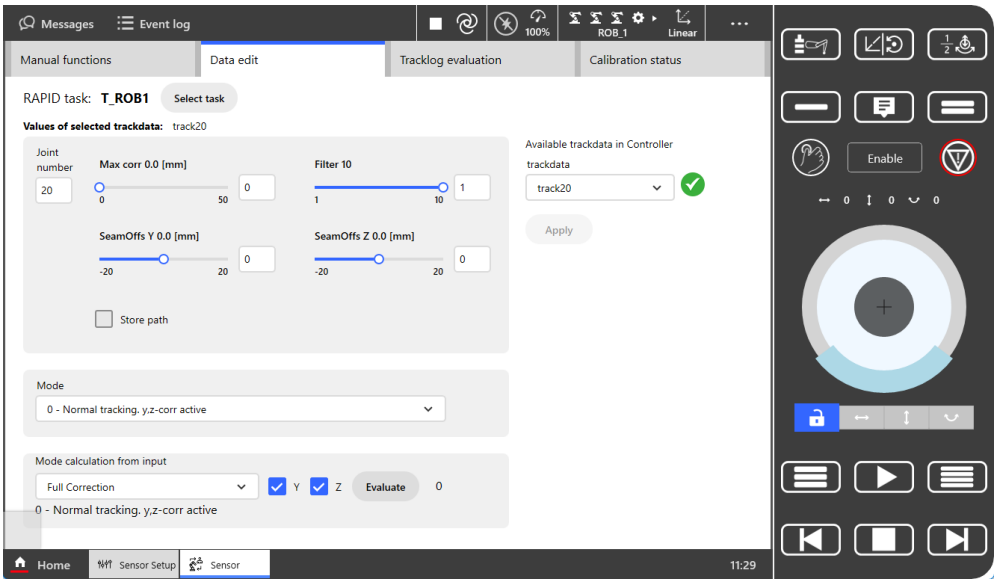
© Copyright 2004-2025 ABB. All rights reserved.



xx2500000099

Data edit

This interface makes it easier to program *trackdata*, especially the *track mode*.



xx2500000100

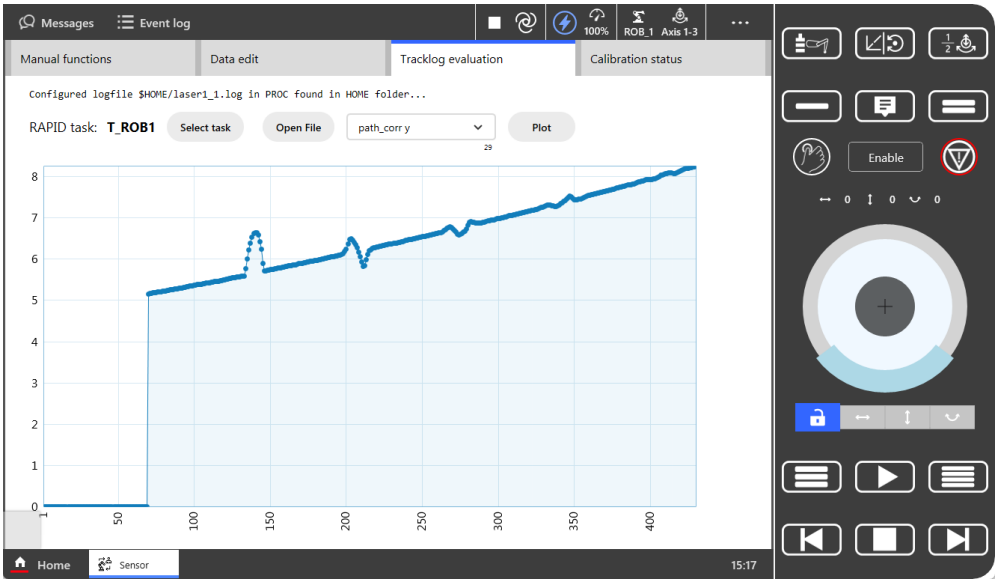
Tracklog evaluation

The tracklog viewer can help to analyze the performance of the sensor with respect to seam visibility. This can help in a decision if the joint definition has to be improved or the sensor angle, etc.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

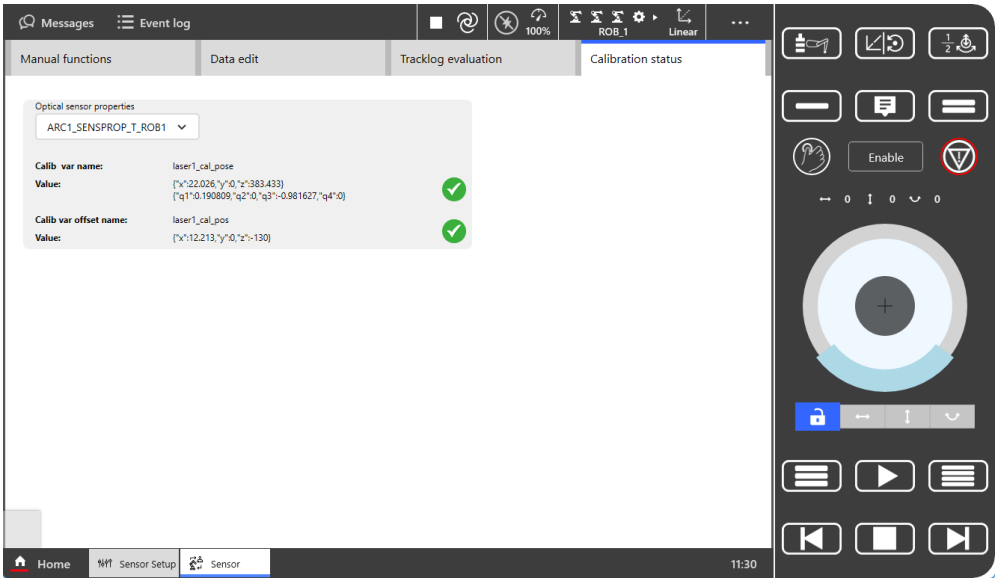
6.3.1 FlexPendant support
Continued



xx2500000101

Calibration status

In this window the calibration result of the latest calibration of any tracking sensor can be inspected.



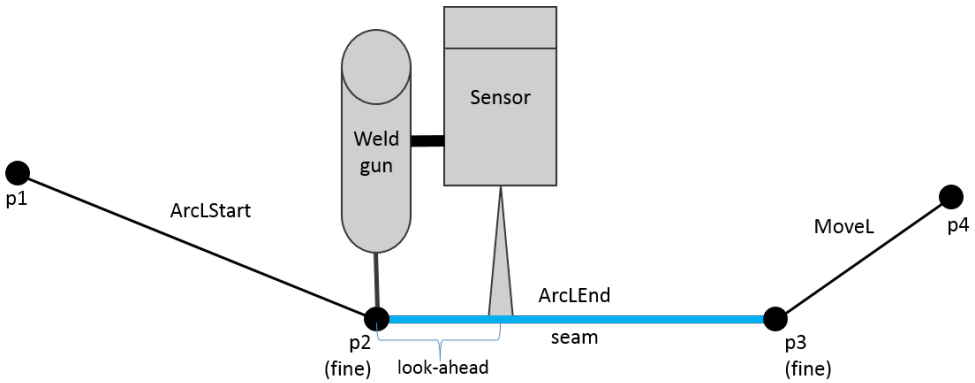
xx2500000102

© Copyright 2004-2025 ABB. All rights reserved.

6.4 Pre process tracking

Description

Optical seam tracking sensors are look-ahead trackers, which means that the sensor cannot start to track at the seam start because the process is started when the weld gun (TCP) is at the seam start. The consequence is that the sensor is a distance of length *look-ahead* into the seam.



xx1700000361

This fact may, in the worst case, make it impossible to use tracking with an optical sensor. One example is if the seam length is of the same magnitude as the look-ahead distance. Another example is if the start position for the seam is not the same for all work pieces, and it is not possible to search it.

With ArcWare it is possible to avoid this problem by using *Pre process tracking*.

Pre process tracking with Arc

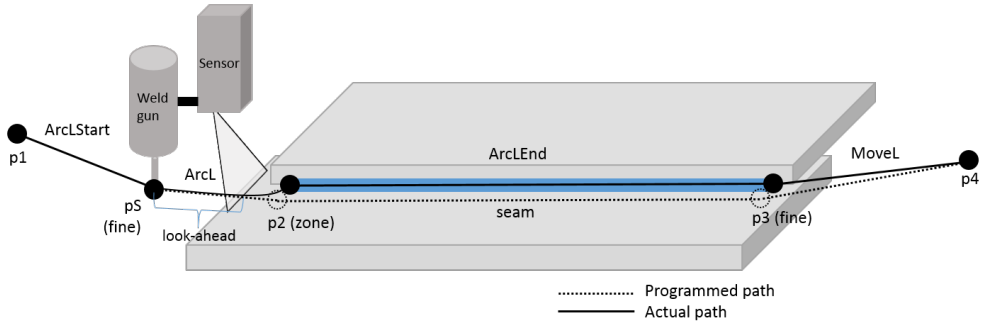
Compared to programming tracking without *Pre process tracking*, one must add one extra position (`pS`) that is used in the `ArcXStart` instruction and one extra `ArcX` instruction before the position where the process should start (`p2`). The position `pS` should be chosen in a way that the sensor measurements start before `p2` on the programmed path. The reason is that the robot should stop at the corrected `p2` position for ignition.

Place `p2`, the welding start, at a position where the sensor has the possibility to get valid measurements before `p2`. If the weld start cannot be placed as described above, one has to search the weld start position and use the searched position instead of `p2`. Otherwise the ignition will be done at the programmed `p2` position which might be off the seam.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

6.4 Pre process tracking
Continued



xx1700000362

To activate *Pre process tracking* you have to specify the optional argument `\PreProcessTracking` for the RAPID instruction `ArcXStart`. Below is a RAPID code example for optical seam tracking with *Pre process tracking*.

```
PROC LA_Arc()
  MoveJ p1, v1000, z50, tFroniusMTB400i_WR;
  ArcLStart pS, v1000, mySeam, weld10, fine,
       tFroniusMTB400i_WR\Track:=track4\PreProcessTracking;
  ArcL p2, v1000, mySeam, weld10, z50,
       tFroniusMTB400i_WR\Track:=track4;
  ArcLEnd p3, v1000, mySeam, myWeld, fine,
       tFroniusMTB400i_WR\Track:=track4;
  MoveL p4, v1000, z50, tFroniusMTB400i_WR;
ENDPROC
```


© Copyright 2004-2025 ABB. All rights reserved.

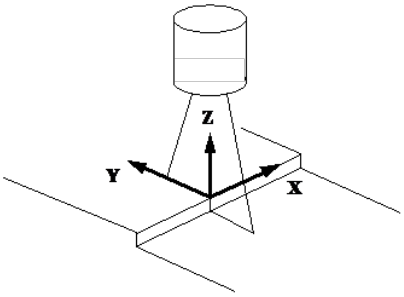
7 Reference information

7.1 Relationship between coordinate systems

Description

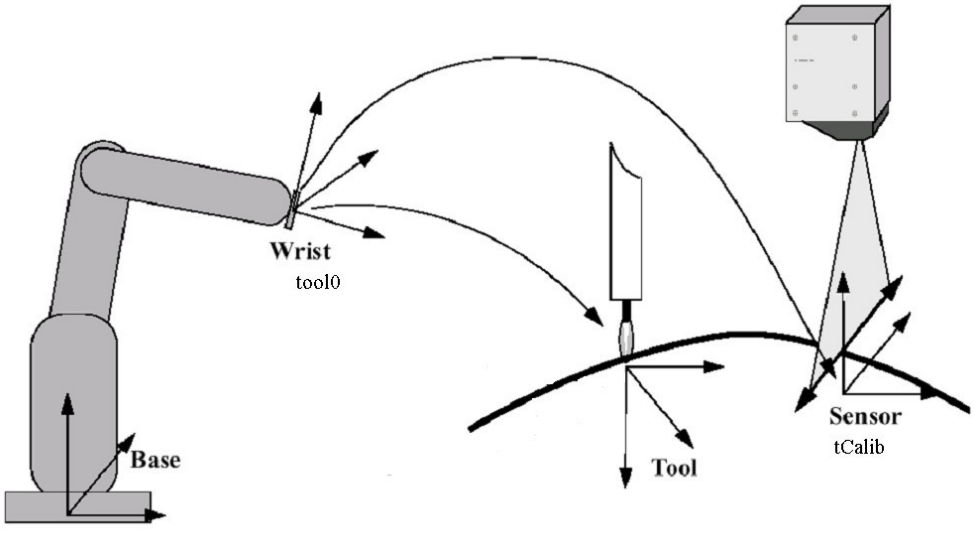
The sensor system shall provide information given in a rectangular right-hand coordinate system defined as follows:

- x-axis - in the direction of movement
- y-axis - horizontal direction
- z-axis - vertical direction "into" the sensor



xx1600001575

The sensor coordinate system has the following relation to the robot base coordinate system:



xx1600001576

© Copyright 2004-2025 ABB. All rights reserved.

7.2 Protocols

7.2.1 Introduction

Description

The communication is divided into two layers, the link protocol and the application protocol.

The task of the link protocol is to safely transfer messages between robot controller and sensor. The contents of the messages is of no concern for the link protocol. It should transfer any type of messages.

The task of the application protocol is to interpret the contents of the messages. Depending on the contents various actions shall be executed.

The link protocol is specified as a server/client protocol, where robot controller is the server and the sensor is the client. This makes the protocol very efficient. There is no need for initial negotiating to set up the link. A complete communication session with a request message down to the client, and a response message back to the server, will be transferred in only two telegrams with a minimum of overhead.

The protocols listed here are application protocols.

© Copyright 2004-2025 ABB. All rights reserved.

7.2.2 LTAPPTCP

Description

The LTAPPTCP protocol consists of a header a body and a footer. The complete message is sent in network order.

	Description
Header:	4 bytes Byte length of body
Body:	Body, see *Message Body on page 58* below.
Footer:	2 bytes 0xabb

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

7.2.2.1 Message Body

Message structure

Structure of a message:

```
<message type> <header><data>
```

Content	Description
<message type>	Indicates the type of message, including if the message is a request or a response. <message type> is always present in a message.
<header>	Contains a user number 0-15 to make it possible to communicate with several sensors on the same serial link. Head specifies if a response is required and if more data will be sent. Head has also bits for future use.
<data>	Additional data depending on the type of message. The number of characters could be 0..128.

Note

Note the difference between message and telegram. The telegram is the container for the message during the transmission over the link. The message is the real information sent from robot controller to sensor.

Requests and responses

A request is a message from robot controller to sensor, demanding an action to take place. Requests are indicated with <message type> = 1..126.

A response is a message from sensor to robot controller, indicating the result of a request. Responds are indicated with <message type> = 129..254.

The respond codes are calculated as corresponding request code added to hex 80. If a request has <message type> = 3, the corresponding response has <message type> = hex 83.

Data types

Data in messages are always of a specific data type. The following data types are used:

Name	Bytes	Type of data	Range
unsigned byte	1	integers	0..255
signed byte	1	integers	-128..+127
unsigned word	2	integers	0..65535
signed word	2	integers	-32768..+32767

Where data are of a type using more than one byte, the most significant part is transmitted first, the least significant part transmitted last.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

Status

In a response message there is always a <status> byte included. This indicates the overall status of the action requested:

Value	Description
+64..+127	OK with an additional specific indication.
+1..+63	OK with an additional generic indication.
0	OK
-1..-63	Not OK, generic error codes.
-64..-128	Not OK, specific error codes.

If <status> is = 0 the task has been performed as requested.

If <status> is < 0 some error has occurred. The value will indicate what type of error. Generic error codes are defined below. Specific error codes may be defined specially for each implementation.

If <status> is > 0 no error has occurred, but there is need to return some additional information. For example calibration has started but is not yet fulfilled.

Generic status codes

The following generic status codes can be returned in <status> in a response message:

Error code	Description
+2	No measurement.
+1	Not ready yet.
0	OK
-1	General error has occurred.
-2	Busy, try later.
-3	Unknown command.
-4	Illegal variable or block.
-5	External alarm.
-6	Camera alarm.
-7	Temperature alarm.
-8	Value out of range.
-9	CAMCHECK failed.

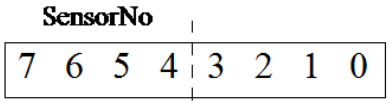
If <status> is = -1, a general error has occurred which is not specifically connected to the requested action. Read the block *Error log* if this function is available to get complete information.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

7.2.2.1 Message Body
Continued

Header



xx1600001577

Bits	Request messages	Response messages
7-4	Not used.	Not used.
3	0: No more data to send 1: More data to send.	Not used.
2-0	Not used.	Not used.

Functions

The following chapter describes all application protocol messages regarding:

- Function
- Syntax of request message
- Syntax of response message
- Parameters in request and response
- Description

Read variable

	Description
Function:	To read one or more not consecutive variables in the sensor.
Message type:	1
Request:	01<header><number><variable1><variable2>...<variable$_n$>
Response:	81<header><status><data1><data2>...<data$_n$>
Parameters:	<header>, see *Header on page 60*. <number>, unsigned word: number of variables to read. <status>, signed byte: status of the response. <variable$_j$>, unsigned word: indicates which variable to read, 0, 1, 2.., 65535. A list of sensor variables can be found in *Application manual - Controller software OmniCore*, section *Sensor Interface*. <data$_j$>, unsigned words: data for the variable *j* to read.
Description:	A number of variables may be defined in the sensor. By this function it is possible for robot controller to read these variables. Each variable has the data type `unsigned word`.

Write variable

	Description
Function:	To read one or more not consecutive variables in the sensor.
Message type:	2
Request:	02<header><number><variable1><data1>... <variable$_n$><data$_n$>
Response:	81<header><status>

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

	Description
Parameters:	<header>, see *Header on page 60*. <number>, unsigned word: number of variables to write. <variable$_j$>, unsigned word: indicates which variable to write, 0, 1, 2.., 65535. A list of sensor variables can be found in *Application manual - Controller software OmniCore*, section *Sensor Interface*. <data$_j$>, unsigned words: data for the variable *j* to write. <status>, signed byte: status of the response.
Description:	A number of variables may be defined in the sensor. By this function it is possible for robot controller to write these variables. Each variable has the data type unsigned word.

Telegram examples

All values are hexadecimal.

<header>=00 in all examples.

<status>=00 if Vision could execute the request (OK).

Acronym	Value
ETX	0x03
DLE	0x10
STX0	0x02
STX1	0x82
ENQ	0x05
NAK	0x15
BCC	Check sum CRC16, see *CRC16 algorithm on page 62*.
T	0x00

A list of sensor variables can be found in *Application manual - Controller software OmniCore*, section *Sensor Interface*.

Selection of joint number 3:

Robot: DLE STXn T 02 <head> 01 10 03 DLE ETX BCC

Vision: DLE STXn T 82 <head>< status> DLE ETX BCC

Turn on the camera with no laser power:

Robot: DLE STXn T 02 <head> 02 06 01 07 01 DLE ETX BCC

Vision: DLE STXn T 82 <head>< status> DLE ETX BCC

Activate the camera laser power (measure on):

Robot: DLE STXn T 02 <head> 01 07 00 DLE ETX BCC

Vision: DLE STXn T 82 <head><status> DLE ETX BCC

Request data and data not available:

Robot: DLE STXn T 01 <head> 03 09 0A 0B DLE ETX BCC

Vision: DLE STXn T 81 <head><status> 00 00 00 DLE ETX BCC

<status> = 02 (No measurement)

Request data and data is available:

Robot: DLE STXn T 01<head> 03 09 0A 0B DLE ETX BCC

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

7.2.2.1 Message Body
Continued

Vision: DLE STXn T 81<head><status><data1><data2><data3>DLE ETX BCC

Turn the laser power down (measure off):

Robot: DLE STXn T 02 <head> 01 07 01 DLE ETX BCC

Vision: DLE STXn T 82 <head><status> DLE ETX BCC

Turn off the camera:

Robot: DLE STXn T 02 <head> 01 06 00 DLE ETX BCC

Vision: DLE STXn T 82 <head><status> DLE ETX BCC

CRC16 algorithm

```
int crc16 (char *bufp, int count)
{
  unsigned int crc = 0;
  while (count--)
  {
    crc = (crc >> 8) ^ xtab [ (unsigned char) (crc ^ *bufp++) ];
  }
  return (crc);
}
static unsigned short xtab[256] = {
0x0000, 0xc0c1, 0xc181, 0x0140, 0xc301, 0x03c0, 0x0280, 0xc241,
0xc601, 0x06c0, 0x0780, 0xc741, 0x0500, 0xc5c1, 0xc481, 0x0440,
0xcc01, 0x0cc0, 0x0d80, 0xcd41, 0x0f00, 0xcfc1, 0xce81, 0x0e40,
0x0a00, 0xcac1, 0xcb81, 0x0b40, 0xc901, 0x09c0, 0x0880, 0xc841,
0xd801, 0x18c0, 0x1980, 0xd941, 0x1b00, 0xdbc1, 0xda81, 0x1a40,
0x1e00, 0xdec1, 0xdf81, 0x1f40, 0xdd01, 0x1dc0, 0x1c80, 0xdc41,
0x1400, 0xd4c1, 0xd581, 0x1540, 0xd701, 0x17c0, 0x1680, 0xd641,
0xd201, 0x12c0, 0x1380, 0xd341, 0x1100, 0xd1c1, 0xd081, 0x1040,
0xf001, 0x30c0, 0x3180, 0xf141, 0x3300, 0xf3c1, 0xf281, 0x3240,
0x3600, 0xf6c1, 0xf781, 0x3740, 0xf501, 0x35c0, 0x3480, 0xf441,
0x3c00, 0xfcc1, 0xfd81, 0x3d40, 0xff01, 0x3fc0, 0x3e80, 0xfe41,
0xfa01, 0x3ac0, 0x3b80, 0xfb41, 0x3900, 0xf9c1, 0xf881, 0x3840,
0x2800, 0xe8c1, 0xe981, 0x2940, 0xeb01, 0x2bc0, 0x2a80, 0xea41,
0xee01, 0x2ec0, 0x2f80, 0xef41, 0x2d00, 0xedc1, 0xec81, 0x2c40,
0xe401, 0x24c0, 0x2580, 0xe541, 0x2700, 0xe7c1, 0xe681, 0x2640,
0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0, 0x2080, 0xe041,
0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281, 0x6240,
0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441,
0x6c00, 0xacc1, 0xad81, 0x6d40, 0xaf01, 0x6fc0, 0x6e80, 0xae41,
0xaa01, 0x6ac0, 0x6b80, 0xab41, 0x6900, 0xa9c1, 0xa881, 0x6840,
0x7800, 0xb8c1, 0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41,
0xbe01, 0x7ec0, 0x7f80, 0xbf41, 0x7d00, 0xbdc1, 0xbc81, 0x7c40,
0xb401, 0x74c0, 0x7580, 0xb541, 0x7700, 0xb7c1, 0xb681, 0x7640,
0x7200, 0xb2c1, 0xb381, 0x7340, 0xb101, 0x71c0, 0x7080, 0xb041,
0x5000, 0x90c1, 0x9181, 0x5140, 0x9301, 0x53c0, 0x5280, 0x9241,
0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x9481, 0x5440,
0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5f00, 0x9fc1, 0x9e81, 0x5e40,
0x5a00, 0x9ac1, 0x9b81, 0x5b40, 0x9901, 0x59c0, 0x5880, 0x9841,
0x8801, 0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x8a81, 0x4a40,
```

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

```
0x4e00, 0x8ec1, 0x8f81, 0x4f40, 0x8d01, 0x4dc0, 0x4c80, 0x8c41,
0x4400, 0x84c1, 0x8581, 0x4540, 0x8701, 0x47c0, 0x4680, 0x8641,
0x8201, 0x42c0, 0x4380, 0x8341, 0x4100, 0x81c1, 0x8081, 0x4040};
```


© Copyright 2004-2025 ABB. All rights reserved.

7.2.3 SOCKDEV

Description

The SOCKDEV protocol is an implementation of the ServoRobot proprietary XML based protocol Robo-Com Light®. For more information contact Servo-Robot Inc.

© Copyright 2004-2025 ABB. All rights reserved.

8 RAPID reference

8.1 RAPID components for optical tracking

About the RAPID components

This is an overview of the instructions, functions, and data types that can be used for optical tracking.

For more information on the RAPID components, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

Instructions

Instructions	Description
SenDevice	`SenDevice` is used, to connect to a sensor device.
IVarValue	`IVarVal` (Interrupt Variable Value) is used to order and enable an interrupt when the value of a variable accessed via the serial sensor interface is changed.
WriteVar	`WriteVar` is used to write a variable to a sensor device.
WriteVarArr	`WriteVarArr` is used to write multiple variables to a sensor device.
ReadVarArr	`ReadVarArr` is used to read multiple variables from a sensor device.

Functions

Function	Description
ReadVar	`ReadVar` is used to read a variable from a sensor device.

Data types

Data Type	Description
sensorvardata	sensorvardata is used to setup the needed information for the different data points that is handled by the `ReadVarArr` and `WriteVarArr` commands.

Modules

The option includes a system module, *LTAPP__Variables*. This module contains the variable numbers defined in the protocol LTAPPTCP. It is automatically loaded as SHARED and makes the variables (CONST num) available in all RAPID tasks.

The option includes a second system module, *CalibData_laser*X with X = *robot number* of a robot with *Optical sensor based SeamTracking* selected. The module is also loaded automatically during installation.

Constants

Name	Number	Read/write (R/W)	Description
LTAPP__VERSION	1	R	A value that identifies the sensor software version.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

Name	Number	Read/write (R/W)	Description
LTAPP__RESET	3	W	Reset the sensor to the initial state, regardless of what state it is currently in.
LTAPP__PING	4	W	Sensor returns a response indicating its status.
LTAPP__CAMCHECK	5	W	Start camera check of the sensor. If this cannot be done within the time limit specified in the link protocol a *Not ready yet* status will be returned.
LTAPP__POWER_UP	6	RW	Turn power on (1) or off (0) for the sensor and initialize the filters. (Power on can take several seconds!)
LTAPP__LASER_OFF	7	RW	Switch the laser beam off (1) or on (0) and measure.
LTAPP__X	8	R	Measured X value, unsigned word. The units are determined by the variable *Unit*.
LTAPP__Y	9	R	Measured Y value, unsigned word. The units are determined by the variable *Unit*.
LTAPP__Z	10	R	Measured Z value, unsigned word. The units are determined by the variable *Unit*.
LTAPP__GAP	11	R	The gap between two sheets of metal. The units are determined by the variable *Unit*, -32768 if not valid.
LTAPP__MISMATCH	12	R	Mismatch, unsigned word. The units are determined by the variable *Unit*. -32768 if not valid.
LTAPP__AREA	13	R	Seam area, units in mm2, -32768 if not valid.
LTAPP__THICKNESS	14	RW	Plate thickness of sheet that the sensor should look for, LSB=0.1mm.
LTAPP__STEPDIR	15	RW	Step direction of the joint: Step on left (1) or right (0) side of path direction.
LTAPP__JOINT_NO	16	RW	Set or get active joint number.
LTAPP__AGE	17	R	Time since profile acquisition (ms), unsigned word.
LTAPP__ANGLE	18	R	Angle of the normal to the joint relative sensor coordinate system Z direction - in 0.1 degrees.
LTAPP__UNIT	19	RW	Units of X, Y, Z, gap, and mismatch. 0= 0.1mm, 1= 0.01mm.
-	20	-	Reserved for internal use.
LTAPP__APM_P1	31	R	*ServoRobot* only! Adaptive parameter 1
LTAPP__APM_P2	32	R	*ServoRobot* only! Adaptive parameter 2

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

Name	Number	Read/write (R/W)	Description
LTAPP__APM_P3	33	R	*ServoRobot* only! Adaptive parameter 3
LTAPP__APM_P4	34	R	*ServoRobot* only! Adaptive parameter 4
LTAPP__APM_P5	35	R	*ServoRobot* only! Adaptive parameter 5
LTAPP__APM_P6	36	R	*ServoRobot* only! Adaptive parameter 6
LTAPP__ROT_Y	51	R	Measured angle around sensor Y axis
LTAPP__ROT_Z	52	R	Measured angle around sensor Z axis A
LTAPP__X0	54	R	*Scansonic* sensors only. Measured X value line 1, unsigned word. The units are determined by the variable `Unit`.
LTAPP__Y0	55	R	*Scansonic* sensors only. Measured Y value line 1, unsigned word. The units are determined by the variable `Unit`.
LTAPP__Z0	56	R	*Scansonic* sensors only. Measured Z value line 1, unsigned word. The units are determined by the variable `Unit`.
LTAPP__X1	57	R	*Scansonic* sensors only. Measured X value line 2, unsigned word. The units are determined by the variable `Unit`.
LTAPP__Y1	58	R	*Scansonic* sensors only. Measured Y value line 2, unsigned word. The units are determined by the variable `Unit`.
LTAPP__Z1	59	R	*Scansonic* sensors only. Measured Z value line 2, unsigned word. The units are determined by the variable `Unit`.
LTAPP__X2	60	R	*Scansonic* sensors only. Measured X value line 3, unsigned word. The units are determined by the variable `Unit`.
LTAPP__Y2	61	R	*Scansonic* sensors only. Measured Y value line 3, unsigned word. The units are determined by the variable `Unit`.
LTAPP__Z2	62	R	*Scansonic* sensors only. Measured Z value line 3, unsigned word. The units are determined by the variable `Unit`.

© Copyright 2004-2025 ABB. All rights reserved.

8.2 RAPID components for optical tracking with Arc

About the RAPID components

This is an overview of the instructions, functions, and data types in *Arc* that can be used for optical tracking.

For more information on all instructions, functions, and data types in *Arc*, see *Application manual - ArcWare for OmniCore*.

Instructions

Instructions	Description
`ArcC`[i]	`ArcC` is used to weld along a circular path.
`ArcCEnd`[i]	`ArcCEnd` is used to weld along a circular path.
`ArcCStart`[i]	`ArcCStart` is used to weld along a circular path.
`ArcL`[i]	`ArcL` is used to weld along a straight seam.
`ArcLEnd`[i]	`ArcLEnd` is used to weld along a straight seam.
`ArcLStart`[i]	`ArcLStart` is used to weld along a straight seam.

i `trackdata` provides the `Arc` instructions with all data necessary for path correction with a Look-Ahead- or At-Point-Tracker. The data is passed to the `Arc` instructions with use of the optional argument `\Track`.

Functions

There are no functions for *Optical Tracking Arc*.

Data types

Data types	Description
`trackdata`	`trackdata` is used to control path corrections during the weld phase. `trackdata` provides the `Arc` instructions with all data necessary for path correction with a Look-Ahead- or At-Point-Tracker. The data is passed to the `Arc` instructions with use of the optional argument `\Track`.

© Copyright 2004-2025 ABB. All rights reserved.

9 System parameters

9.1 System parameters for Arc

About the system parameters

This is a description of the system parameters used by *Arc*. For information about other system parameters, see *Technical reference manual - System parameters*.

Type Optical Sensor

The type *Optical Sensor* holds parameters for the option *Optical Sensor* and belongs to the topic *Process*.

Parameter	Data type	Note
Name	string	The name of the optical sensor.
Use Optical Sensor Class	string	The *Optical Sensor Class* used by the *Arc Sensor Class*.
Use Optical Sensor Properties	string	The optical sensor properties used by the *Optical Sensor*. Two sensor properties are available: MSPOT90 and SCOUT.
Connected to Robot	string	The robot to which the sensor is connected.

Type Optical Sensor Properties

The type *Optical Sensor Properties* holds parameters for the optical sensor and belongs to the topic *Process*.

Parameter	Data type	Note
Name	string	The name of the *Arc Sensor Class*.
Sensor Manufacturer	string	The name of the sensor manufacturer.
Track System	num	Defines the TrackSystem type. TrackSystem number: Tracking system type 0: WeldGuide IV 1: Optical sensor based tracking 2: Signal based tracking
Device	string	The device name used for the tracker. Device must match the transmission protocol name configured as *Sensor Device* under the topic *Communication*. "Laser1:" for lasertracker.
Pattern Sync Threshold	num	The coordination position at the extents of the weaving pattern. It is specified as a percentage of the width on either side of the weaving center. When weaving is carried out beyond this point, a digital output signal is automatically set to one. This type of coordination is intended for seam tracking using Through-the-Arc Tracker.

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

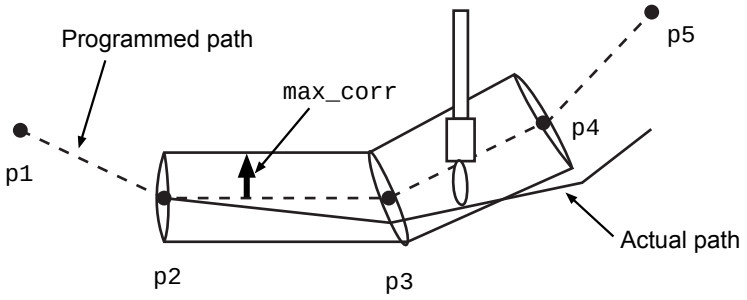
9.1 System parameters for Arc
Continued

Parameter	Data type	Note
Max Blind	num	The max_blind component defines the maximum distance in *mm* the robot is allowed to continue moving under the assumption that the last reported position error is still valid. The parameter should be tuned to match the maximum expected tack lengths used, or the length of other features. For example, clamps that may prevent the sensor from accurately detect the actual position and geometry of the seam. If the max_blind distance has been exceeded with no new position data from the sensor an error will be reported, and program execution is stopped.
Max Corr	num	Not used.
Adapt Start Delay	num	Not used.
Max Incremental Correction	num	Max incremental correction for the arc tracking system. If the incremental TCP correction is bigger than *Max Incremental Correction* and *Max Correction Warning* was set, the robot will continue its path, but the applied incremental correction will not exceed *Max Incremental Correction*. If *Max Correction Warning* was not set, a track error is reported and program execution is stopped. Default value is 3 mm.
LogFile	string	The name of the log file created during tracking.
Logfile size	num	The maximum number of log entries that can be stored in the log file. In case the whole track log cannot be saved in the file, the latest log entries are stored in the file. Default value is 1000
Sensor Frequency	num	Defines the sample frequency of the sensor used. (e.g. M-Spot-90 has 5Hz sampling frequency)
Correction Filter	num	Defines filtering of the correction calculated, using mean value over corr filter values. Use default value: 1
Error Ramp In	num	Defines during how many sensor readings ramp in is done after an error caused by sensor reading.
Error Ramp Out	num	Defines during how many sensor readings ramp out is done after an error caused by sensor reading.
CB Angle	num	Defines the angle between a 3D sensor beam and the sensor z-axis. Use default value: 0 for M-Spot-90 and 25 for SCOUT.
Calib Variable Name	num	The name of the calibration variable name found in the calibration programs. RAPID data type: pose
Calib Variable Offset Name	num	The name of the calibration variable offset name found in the calibration programs. RAPID data type: pos
Max Correction Warning	bool	If this parameter is enabled, program execution is not interrupted, when the limit for maximum correction, specified in the trackdata, is exceeded. Only a warning will be sent. Default value: FALSE

Continues on next page

© Copyright 2004-2025 ABB. All rights reserved.

Parameter	Data type	Note
WgTrack	string	Not used.
WgLeftSynch	string	Digital output signal for left syncpulse.
WgRightSynch	string	Digital output signal for right syncpulse.



xx1200000686

Max correction

© Copyright 2004-2025 ABB. All rights reserved.

This page is intentionally left blank

Index

© Copyright 2004-2025 ABB. All rights reserved.

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

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

ABB Engineering (Shanghai) Ltd.
Robotics & Discrete Automation
No. 4528 Kangxin Highway
PuDong New District
SHANGHAI 201315, China
Telephone: +86 21 6105 6666

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

abb.com/robotics

3HAC095098-001, Rev A, en

© Copyright 2004-2025 ABB. All rights reserved.
Specifications subject to change without notice.