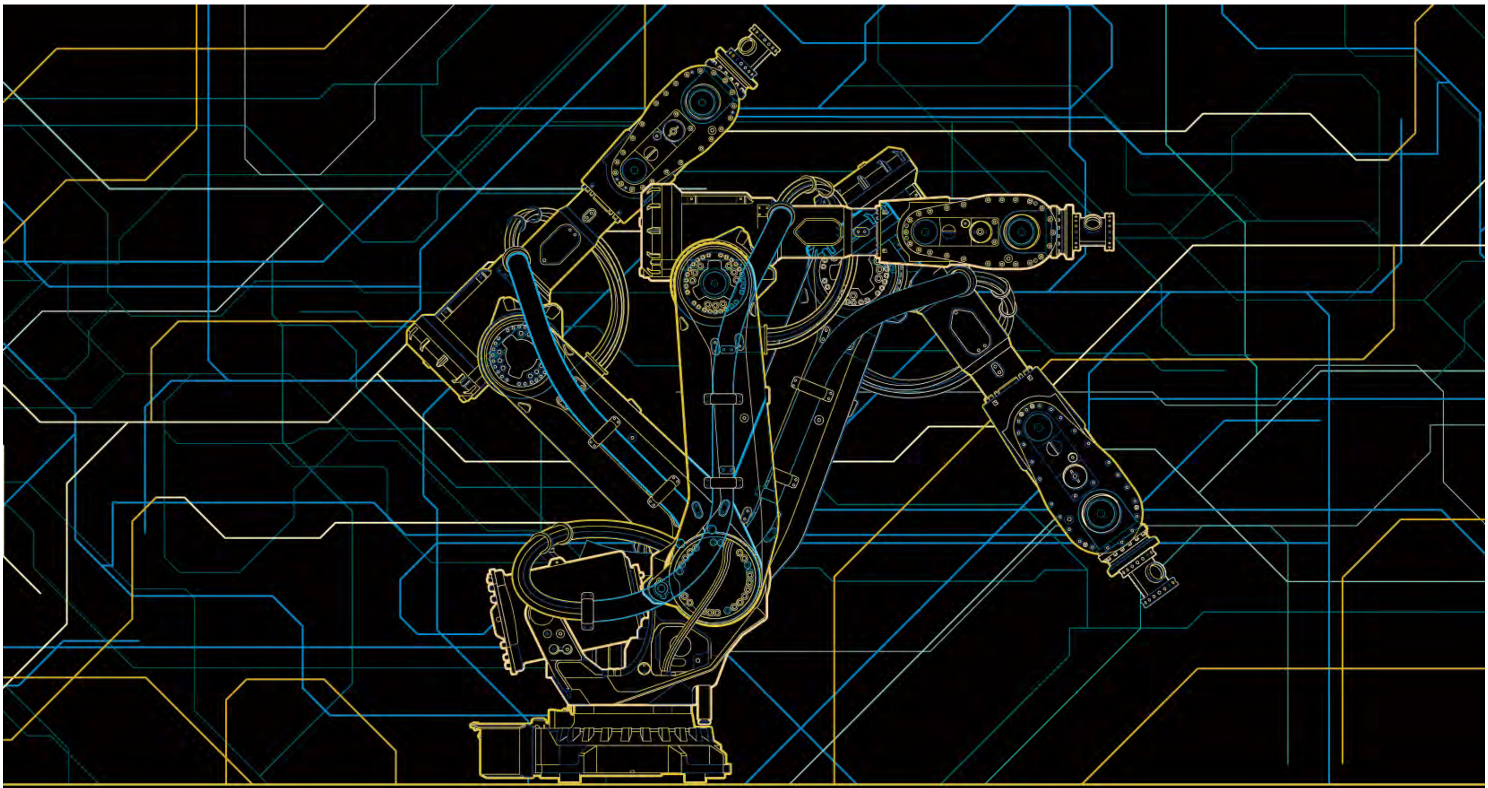


Operating manual
Seam tracking with Weldguide IV and MultiPass

Power and productivity
for a better world™

Trace back information:
Workspace Main version a117
Checked in 2015-08-31
Skribenta version 4.6.081

Operating manual

Seam tracking with Weldguide IV and MultiPass

RobotWare 6.02

Document ID: 3HAC054886-001

Revision: A

ABB AB
Robotics Products
Se-721 68 Västerås
Sweden

Table of contents

Overview of this manual

About this manual

This manual contains instructions for installing and configuring seam tracking with Weldguide IV and MultiPass.

Prerequisites

The installation/maintenance/repair personnel working with an ABB Robot must be trained by ABB and have the knowledge required for mechanical and electrical installation/maintenance/repair work.

References

References	Document ID
Operating manual - IRC5 with FlexPendant	*3HAC050941-001*
Operating manual - RobotStudio	*3HAC032104-001*
Application manual - Arc and Arc Sensor	*3HAC050988-001*
Application manual - Continuous Application Platform	*3HAC050990-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC050917-001*
Technical reference manual - RAPID overview	*3HAC050947-001*
Technical reference manual - System parameters	*3HAC050948-001*
Application manual - Controller software IRC5	*3HAC050798-001*
Operating Manual - ArcWelding PowerPac	*3HAC028931-001*
Service diagram - Weldguide IV	*3HAC054912-001*

Note

The document numbers that are listed for software documents are valid for RobotWare 6. Equivalent documents are available for RobotWare 5.

Revisions

Revision	Description
-	Released with RobotWare 6.01. First release.
A	Released with RobotWare 6.02. • Updated and restructured section *Communication on page 33*. • Added a description of the FlexPendant GUI, see *Illustration WGView on page 61*. • Added list of spare parts, see *Spare parts on page 117*. • Recommended values for weld_penetration changed to 1-10.

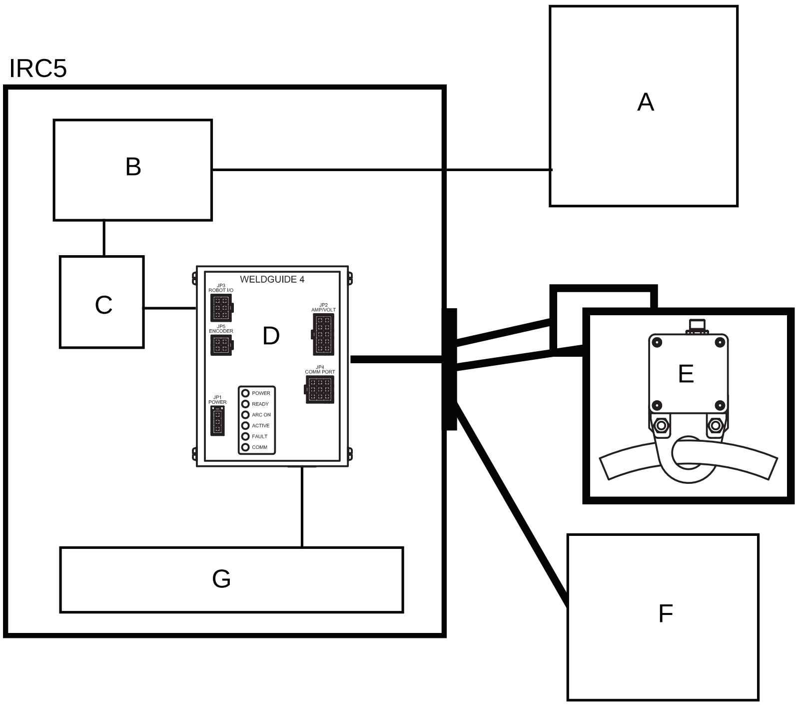
This page is intentionally left blank

1 About Weldguide and seam tracking

1.1 About Weldguide

Introduction to Weldguide

Weldguide is a Thru-Arc™ tracking sensor designed for a robotic welding system. Weldguide uses a microprocessor based weld sequence controller that is seamlessly integrated to the robot controller via Ethernet. The system gives a tracking functionality of the path, adjusting the robot to the actual path location. Weldguide measures current and voltage of the arc and sends path corrections to the robot. Measurements are made at the edge of the weave pattern. It is designed to track difficult welding joint variations due to cast components or other pre-process problems. It monitors and controls through-the-arc seam tracking.



xx1500000579

- A Power source
- B Welding interface
- C I/O
- D Weldguide board
- E Voltage and current sensors
- F PC
- G IRC5 main computer

Prerequisites

The following prerequisites apply:

- IRC5 controller

Continues on next page

- RobotWare 5.13 or higher with the options *Arc* and *Weldguide MultiPass* for the basic version (*Weldguide Basic*).
- RobotWare 5.13.02 or higher with the options *Arc* and *Weldguide MultiPass* for the advanced version (*Weldguide Advanced*).
- Users must be trained welders to fully understand the measured results and the robot's reactions based on the `trackdata`.

Limitations

The following limitations apply:

- MultiPass instructions can only be used with the first arc system.
- In a MultiMove system the MultiPass instructions can only be used in semi-coordinated mode. Synchronized coordinated motion is not yet supported.
- In a MultiMove system, Weldguide is supported to be used on two systems.
- Welddata tuning with the RobotWare Arc user interface on the FlexPendant is not supported for the `ArcRepL` instruction.

Weldguide with aluminium

Weldguide tracks work by using a weave while welding and as the edges of the weave are reached, there is a change in electrode extension created by the joint configuration. The change in electrode extension changes the electrical resistance in the wire which changes the current and voltage values. From this, we read the change in impedance at the sides of the weld and where we get the change early, we can make the offset.

Aluminium and its alloys have such low electrical resistance that it is very difficult to get a big enough change in impedance to measure. Some alloys may give better result than others, but each case would have to be well tested to make sure.

1.2 Basic and advanced Weldguide version

Versions

There are two versions of Weldguide, a basic (*Basic*) and an advanced (*Advanced*) version.

	Weldguide Basic	Weldguide Advanced
FlexPendant user interface	Yes	Yes
Height sensing[i]	Yes	Yes
Centerline tracking	Yes	Yes
Inverted centerline tracking	Yes	Yes
MultiPass[ii]	Yes	Yes
Adaptive fill[iii]	No	Yes
Single side tracking	No	Yes

i Torch to Work tracking, Z direction
ii MultiPass capability with variable replay of paths
iii Adaptive control of welding speed and weave width

Upgrading to advanced version

To upgrade from basic to advanced version, the Weldguide board must be unlocked to activate single side tracking and adaptive filling.

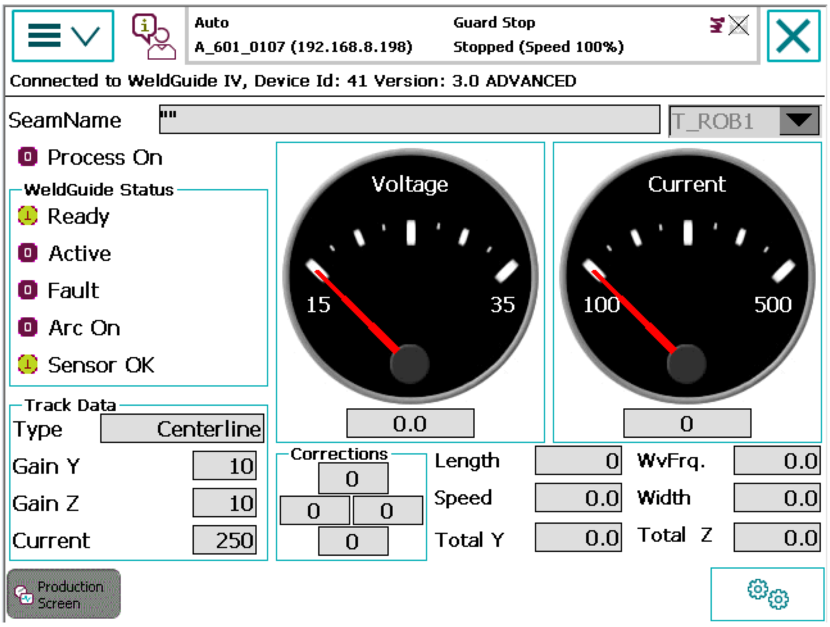
The Weldguide board has a unique serial number stored in a file in the robot controller. The file is stored in the folder */HOME/Arc/ConfigTemplates/Weldguide* and is named `WgSerialNum_x_x.txt`, where `_x_x` is programmatically replaced with the unique serial number.

Use this procedure to upgrade to advanced version.

1. Copy the `WgSerialNum_x_x.txt` file to your computer and send it to your ABB contact. You will receive a new file to place in the *HOME* folder on the controller.
2. Restart the controller to update the board. The file will be automatically removed from the *HOME* folder after the update.

Continues on next page

3 To verify the update, check the event log and the FlexPendant application. The Device Id shown in the FlexPendant application has changed from 40 to 41 and the text *ADVANCED* is displayed.



xx1500000544

1.3 Tracking methods

Introduction

A through-the-arc tracking system uses the arc as a sensor to adjust the robot path to the actual location of the part. Measuring the arc voltage and welding current, synchronized with the robot weave pattern, the stick-out length is calculated on both sides and in the middle of the weld. The stick-out length in the middle and the difference between the sides are converted in to robot vertical and horizontal corrections.

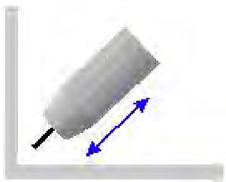
It is necessary to understand that there are several tracking modes as well as understanding their relationship within the tracking process.

The tracking methods described below are controlled by the `trackdata` component `track_type`. See *trackdata on page 105*.

Torch-to-work tracking (height, Z direction)

In torch-to-work mode, the same contact tip to work length is maintained. The contact tip to work distance is specified as voltage and current settings in the weld data. Weaving with almost zero width is required because the correction calculations are synchronized with the weave pattern.

> **Note**
>
> Use `track_type 5` in `trackdata`.



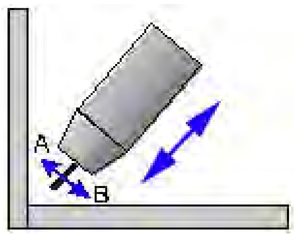
xx1300000875

Centerline tracking (center, Y and Z direction)

Centerline tracking is the most commonly used tracking method. While torch-to-work tracking is based on measurements made in the middle of the weave pattern, the centerline tracking is based on measurement made on the sides of the weave pattern. Corrections are calculated based on the difference in stick-out between the sides. The position of the weld can be adjusted side to side using the bias (`track_bias`) parameter.

> **Note**
>
> Use `track_type 0` in `trackdata`.

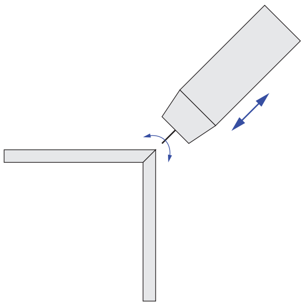
Continues on next page



xx1300000876

Inverted centerline tracking

For inverted centerline tracking, make sure to use v-shaped weaving and negative height. See `weavedata` and `trackdata` in *Application manual - Continuous Application Platform.*



xx1500000545

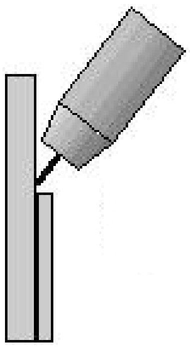
> **Note**
>
> Use `track_type 20` inverted centerline or `track_type 30` inverted centerline in `trackdata`. For `track_type 30`, both voltage and current are specified.

Single side tracking (right and left)

The difference between the centerline tracking method and the single side method is the way cross seam corrections are calculated. When using the single side method, data from one side of the weave is used. The length of the stick-out in the center of the weave is stored as a reference. The side of the grove is then detected as a difference in stick-out at one of the sides compared with the center. The difference in tick-out required for detecting the side is defined as a penetration level (`weld_penetration`). A higher penetration level makes the weld move further into the selected side. This method can be used when tracking a lap joint, were the arc might consume one of the side of the grove.

> **Note**
>
> Use `track_type 2` for right side tracking, and `track_type 3` for left side tracking.

Continues on next page

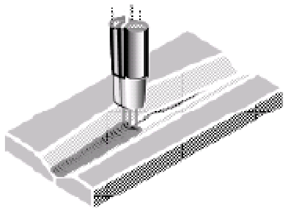


xx1300000877

Adaptive fill

Adaptive fill allows the robot to identify and adjust for variations in joint tolerances. If the joint changes in width, the robot's weave will increase or decrease and the travel speed will be adjusted accordingly.

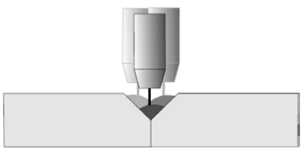
> **Note**
>
> Use `track_type 1` in `trackdata`.



xx1300000878

MultiPass

MultiPass welds are sometimes required due to the required weld size and thickness of the material being joined. Weldguide makes this easy by tracking the first pass and storing the actual tracked path so it can offset for subsequent passes.



xx1300000879

This page is intentionally left blank

2 Installation

2.1 Weldguide hardware

System overview

The Weldguide system consists of the following major components:

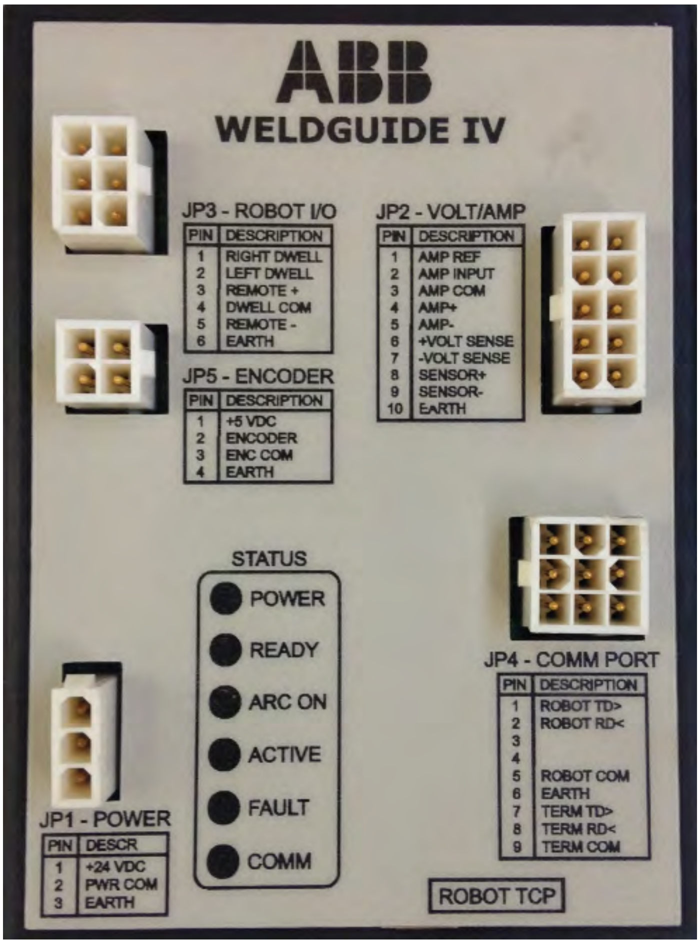
- DIN rail mountable embedded microprocessor assembly (the Weldguide board).
- Integrated Volt/Ampere sensor assembly.
- Installation wiring harness and cable assembly.
- Weldguide connector panel.

The embedded microprocessor assembly provides:

- Six isolated 24 VDC inputs.
- Four isolated solid state relay outputs.
- EtherNet (default).
- One RS-232 robot serial port (only when Weldguide IV is used as replacement for Weldguide III).
- One isolated RS-232 offline programming serial port.
- A remote analog sensor interface.

Continues on next page

Embedded microprocessor



xx1500000546

Dimension	70 mm x 89 mm x 118 mm (H x W x L) **Note** Allow an additional clearance of 38 mm above the PCB for connector clearance. Module mounts on 35 mm DIN rail.
Weight	0.13 kg
Power input	12-32 VDC @ 0.2 A (nominal 24 VDC input)
Logic inputs	Optically isolated 12-24 VDC @ 10 mA

Continues on next page

Solid core sensor

xx1500000547

Dimension	53 mm x 64 mm x 143 mm (H x W x L) **Note** Allow an additional clearance of 38 mm below the sensor for connector clearance. Max welding cable size 25 mm.
Diameter	27 mm
Weight	0.198 kg
Sensor current:	0-800 ADC. Accuracy: ±1.5% full scale ±2 digits 0-1000 ADC. Accuracy: ±2.5% full scale ±3 digits
Sensor voltage	0-50 VDC. Accuracy: ±1% full scale ±2 digits 0-100 VDC. Accuracy ±2.0% full scale ±2 digits

Continues on next page

Split core sensor

xx1500000548

Dimension	41 mm x 136 mm x 149 mm (H x W x L) **Note** Allow an additional clearance of 38 mm below the sensor for connector clearance. Max welding cable size 25 mm.
Diameter	57 mm
Weight	1.27 kg
Sensor current:	0-1000 ADC. Accuracy: ±2.0% full scale ±3 digits
Sensor voltage	0-50 VDC. Accuracy: ±1% full scale ±2 digits 0-100 VDC. Accuracy ±2.0% full scale ±2 digits

2.2 Overview of interface configuration

Input and output signals

The Weldguide controller requires a minimum of two output signals from the robot controller, two dwell bit signals that indicate the left and right most extreme position of the weave pattern. In addition the Weldguide provides four outputs to the robot that can be used to indicate the operational status of the embedded controller. The dwell input bits share a single common and are configured for a 24 VDC sourcing output. The INP4-INP8 inputs also share a single common input and are configured for a 24 VDC sourcing output. The Weldguide outputs share a common output and can be configured for sourcing or sinking outputs.

Ports

Weldguide communicates with the robot through an EtherNet port. The robot has full access to the Weldguide variables and configuration parameters through this port.

Weldguide can also communicate with the robot through an a RS-232 serial port, which supports the ABB sensor protocol version 1.4. This port is only used when using the Weldguide IV as replacement for Weldguide III.

Weldguide has a second RS-232 terminal port that supports an ASCII serial protocol. This port is isolated from the Weldguide power supply. It can be used to configure and monitor the embedded controller.

Continues on next page

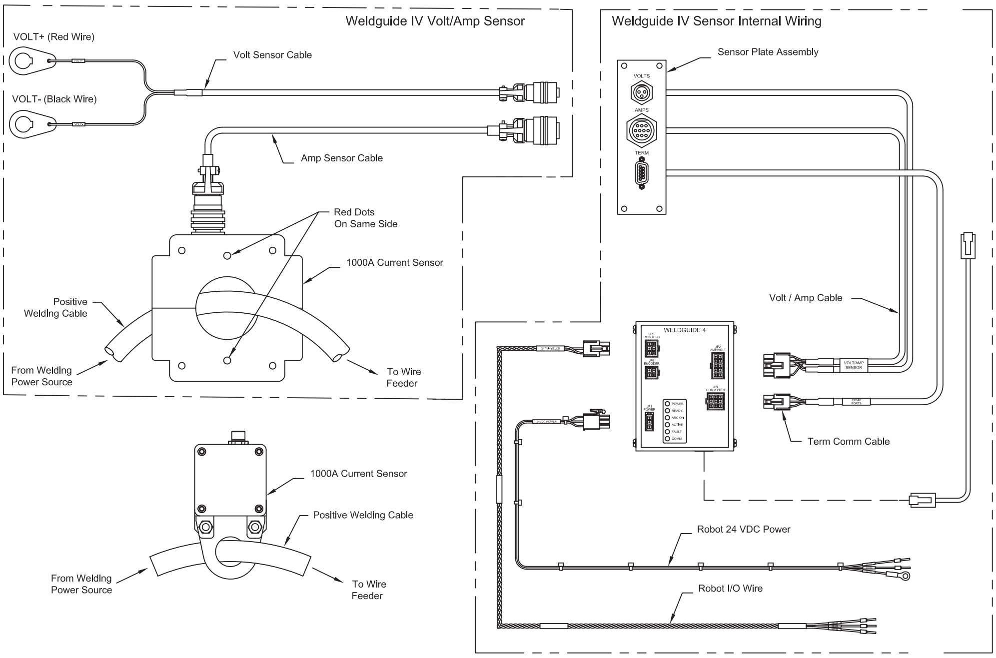
Connector and plates

A system bulkhead plate is provided to assist in the installation of the terminal RS-232 and remote sensor cable. The I/O connectors have one meter long cables and are terminated to the associated plugs that are installed on the embedded control module.

xx1500000549

2.3 Mounting and connecting the board and connector panel

Schematic overview



xx1500000554

Mounting

The Weldguide board can be mounted on any DIN rail system. It can be, for example, mounted on the IRC5 cabinet door.

The Weldguide connector panel is mounted on the connection interface on the IRC5 cabinet (XS10, XS11, or XS12), see *Connections on the controller on page 25*.

If the application is not functioning properly after installation, see section *Trouble shooting on page 115* for a possible solution.

Continues on next page

Connectors

The following connectors are used to interface the Weldguide controller to the robot and external sensor.

xx1500000550

JP1	A 3-pin connector to supply 24 VDC power.
JP2	A 10-pin connector to connect the external Volt/Ampere sensor.
JP3	A 6-pin connector for the robot I/O connections which includes dwell bit signals and discrete I/O signals.
JP4	A 9-pin connector for serial communication (robot RS-232 and terminal RS-232). (only when Weldguide IV is used as replacement for Weldguide III)
JP5	(Not used)
ROBOT TCP	An EtherNet connector for communicating with the robot controller.

Status LEDs

The following green status LEDs are available on the Weldguide controller.

LED	Description
POWER	Indicates that 24VDC power has been applied.
READY	Indicates that Weldguide is operational.
ARC ON	Indicates when the welding arc is established.
ACTIVE	Indicates when the controller is generating correction vectors.
FAULT	Indicates when a dwell bit fault condition has been detected.
COMM	Shows active communication to the robot controller.

Continues on next page

Mounting and connecting the board

Use this procedure to mount and connect the Weldguide board and connector panel.

1. Mount the Weldguide board on a DIN rail. Allow access to the terminal block panel.
2. Route the Weldguide control cables (sensor connection and EtherNet connection) from the connector panel to the Weldguide board. Keep the Weldguide control cables away from any power control lines or cables. If possible, place the cables in a wiring duct independent of other high voltage control solenoid or motor wiring.
3. Connect the V/A sensor connection cable to the Weldguide board (JP2 -VOLT-AMP).
4. Connect the EtherNet connection cable to the Weldguide board (ROBOT TCP).
5. Connect the robot communication cable (dwell bits) to the Weldguide board (JP3 - ROBOT I/O) and to an I/O board. Only the two dwell bits are needed for tracking. The other signals can be used to monitor the Weldguide board.

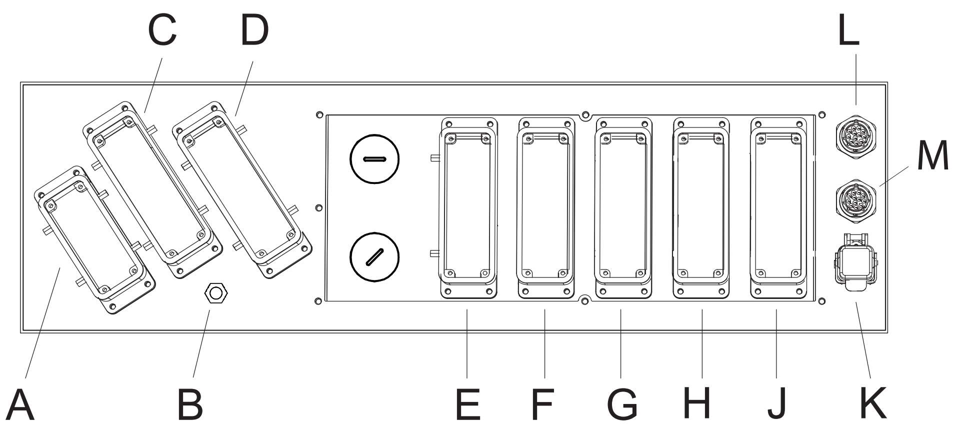
   Note

   Note that the I/O signals must also be configured in the system parameters using the correct names, see *Configuring I/O on page 36*.

6. Connect the power cable to the Weldguide board (JP1 - POWER). Only supply 24 Volts!
7. Continue with the sensor, see *Installing the sensor on page 27*.

For more information, see *Service diagram - Weldguide IV*.

Connections on the controller



xx0500001852

	Description
A	XP.0 Mains connection
B	Earth connection point
C	XS.1 Robot power connection

Continues on next page

	Description
D	XS.7 Additional axes power connection
E	XS.13/XS.5 Customer power/signals external connection
F	XS.10 Customer options
G	XS.11 Customer options
H	XS.12 Customer options
J	X3 Customer safety signals
K	XS.28 Network connection
L	XS.41 Additional axes SMB connection
M	XS.2 Robot SMB connection

2.4 Installing the sensor

Mounting

Install the sensor as closely as possible to the wire feed drive motor assembly, preferably at the attachment point of the weld cable to the wire drive assembly. Try to keep the sensor at least 1 meter from the power source terminals. The important thing is to keep the voltage sensing as close as possible to the wire feeder. This will reduce the effect of voltage drop due to weld cable lengths.

The V/A sensor uses a through-hole linear current sensor and a terminal block for the voltage connection. The sensor must be installed around the welding cable.

Connecting the sensor

Use this procedure to install the sensor.

Note

Note that the sensor must also be calibrated, see *Calibrating the sensor on page 29*.

1 For a solid core sensor disconnect the welding cable from the wire feed drive and insert the cable through the sensor. For a split core sensor this is not necessary.

Observe the current direction markings on the sensor for proper operation.

xx1500000547

xx1500000548

Note

You may need to create a pig-tail adapter to connect the welding cable through the solid core sensor to the wire feeder.

Continues on next page

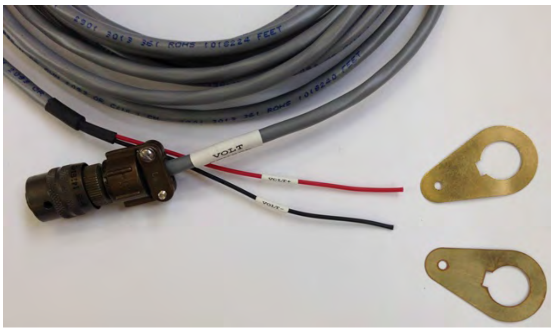
2 Connect the voltage cable to the connector panel on the IRC5 cabinet.

xx1500000549

3 Connect the other end of the voltage cable, the plus and minus wires. The wires need to be extended, with 18 AWG (1 mm) or equivalent wire.

Connect the plus wire to the wire feed drive motor welding power cable connection, and connect the minus wire to a suitable location as close as possible to the welding fixture. See *Example of voltage cable connections on page 30*.

> **Note**
>
> Do not connect the cable to the plus and minus of the power source.



xx1500000551

> **Note**
>
> If the sensor cannot be mounted at the wire drive assembly we recommend connecting a 1.0 A in-line fuse to the positive sense lead at the motor drive. This will prevent excessive current be drawn through the wire if the sense lead becomes damaged.

Continues on next page

4 Connect the current (Ampere) sensor cable to the sensor.

xx1300001956

Calibrating the sensor

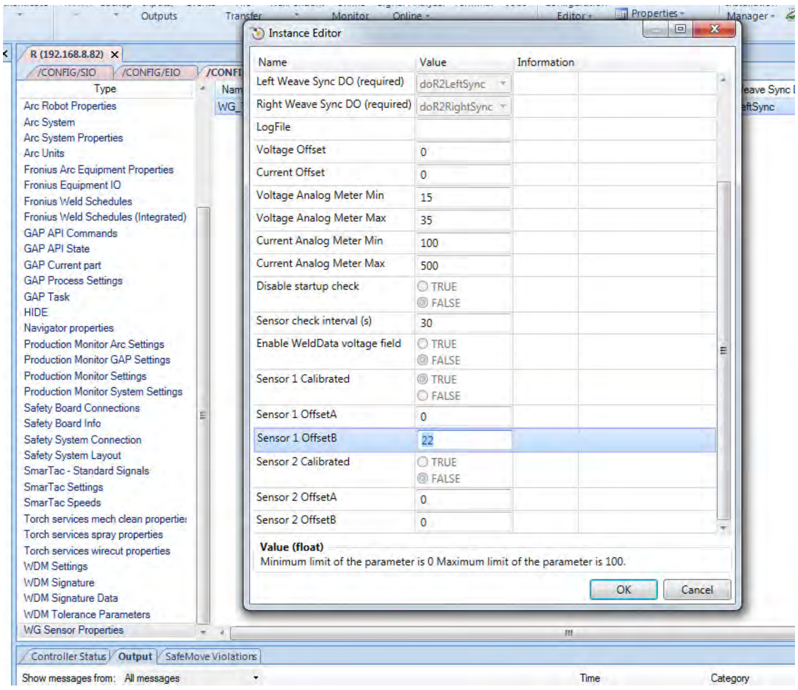
Use this procedure to calibrate the sensor, using RobotStudio or the FlexPendant.

1. In RobotStudio, open the **Configuration editor**.

   On the FlexPendant, tap the ABB menu, **Control panel**, and **Configuration**.
2. Select topic **Process** and **WG Sensor Properties**.
3. Set the value to **FALSE** for the parameter **Sensor 1 calibrated**.

   If you have a multimove system, do the same for parameter **Sensor 2 calibrated.**
4. Set the offset value to **0** for the parameter **Sensor 1 OffsetB**.

   If you have a multimove system, do the same for parameter **Sensor 2 OffsetB**.
5. Restart the controller.
6. Check the **WG Sensor Properties** that you now have an offset value, and that the **Sensor x calibrated** parameters has changed to **TRUE**.



xx1500000578

Continues on next page

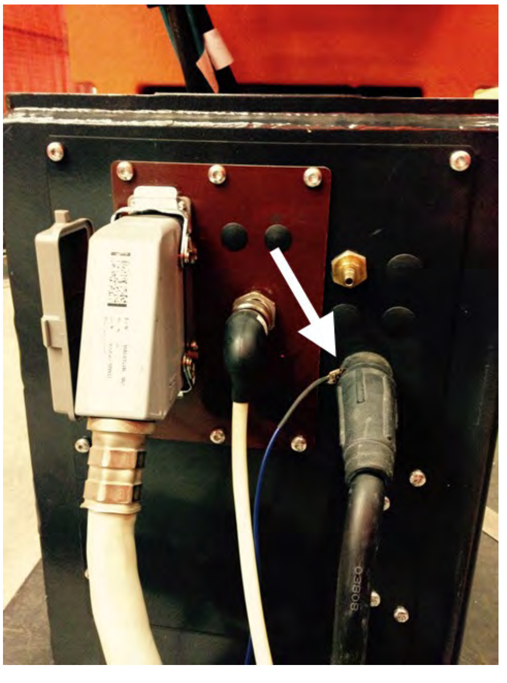
Example of voltage cable connections

The voltage cable is connected to the connector panel on the IRC5 cabinet, see *Connecting the sensor on page 27*.

The following figure shows that the plus cable is connected to the back of the wire feeder.

xx1500000552

The following figure shows that the minus cable is connected to the positioner. An alternative is to connect the minus cable close to the workpiece.



xx1500000553

2.5 Software installation

Installing RobotWare to the IRC5 controller

A RobotWare license with the option *Weldguide MultiPass* enabled is required to to run Weldguide.

Use RobotStudio to configure, build, and download a RobotWare system to the IRC5 controller.

For more information, see *Operating manual - RobotStudio*.

Note

Using multiple Weldguide (max 2) requires one of the MultiMove options *MultiMove Coordinated* or *MultiMove independent*.

3 Configuration

3.1 Communication

3.1.1 Introduction

General

By default the IRC5 controller is setup to communicate to the Weldguide board using serial communication. It is recommended to change the communication settings to use Ethernet communication instead, see *Using Ethernet communication on page 34*.

3.1.2 Using Ethernet communication

Changing communication settings

One of the following configuration files must be manually loaded to the IRC5 controller to change the communication settings for the Weldguide board.

The configuration files are available on the robot controller, in the folder *HOME\Arc\ConfigTemplates\Weldguide*.

Configuration	Description
`siETH_WG_T_ROB1.cfg`	Configuration file for Ethernet communication for robot 1.
`siETH_WG_T_ROB2.cfg`	Configuration file for Ethernet communication for robot 2.
`siAWC_T_ROB1.cfg`	Configuration file for serial communication. (default)

The Weldguide communication is established when the IRC5 controller is restarted.

The Weldguide board is normally connected to the LAN3 port of the IRC5 main computer, but when building a system with Weldguide, the configuration is modified so that any LAN port can be used.

For MultiMove the Weldguide boards are connected to the LAN3 port through a switch.

Note

Note that the default setting is automatically activated when using the restart mode **Reset system**.

Changing the IP-address of the Weldguide board

All Weldguide boards are preconfigured with the same IP-address when delivered.

Board	IP-address
Weldguide board 1	*192.168.125.50* (preconfigured)
Weldguide board 2 (MultiMove)	*192.168.125.51* (needs to be changed)

To change the IP-address of the Weldguide board a separate software tool is needed, the DeviceInstaller for XPort® Pro™. This tool can be downloaded from www.lantronix.com.

Use this procedure to change the IP-address of the Weldguide board.

1. Download and install the DeviceInstaller software to your PC.
2. Make sure that any installed firewalls on the PC allows communication with the Weldguide board, or are turned off.
3. Connect the Weldguide board to your PC.
4. Open DeviceInstaller.
5. Go to **Tools** and **Options** and select the correct network adapter to which the Weldguide board is connected.
6. Click **Assign IP**, and follow the steps of the **Assign IP Address** wizard.

Continues on next page

Assign the correct IP-address according to the above table.

7 Click **Finish** to complete the wizard.

For more information about DeviceInstaller, see the included user manual or www.lantronix.com.

Limitations

Note

The Weldguide board may only be used on a private network to the IRC5 controller. It is not allowed to connect the Weldguide board to a public network.

Note

The Weldguide board has a built-in web- and FTP-server, with a default user and password, which are not supported by the IRC5 controller.

For more information see the description for XPort® Pro™ on www.lantronix.com.

3.1.3 Configuring I/O

Configuring signals

The following I/O configuration must be added for a Weldguide system. Note that the signal names listed below must be used.

Signal	Description
doR1LeftSync	Left sync signal for robot 1
doR1RightSync	Right sync signal for robot 1
doR2LeftSync	Left sync signal for robot 2
doR2RightSync	Right sync signal for robot 2

Example configuration

Example EIO files containing the left and right sync signals are available on the robot controller, in the folder *HOME\Arc\ConfigTemplates\Weldguide*

Configuration	Description
`eWG_T_ROB1.cfg`	EIO configuration file for robot 1
`eWG_T_ROB2.cfg`	EIO configuration file for robot 2

Note

These example I/O signals are simulated and are intended as examples only. An EIO file configured for the EIO board that is connected to the Weldguide unit must be loaded in the system, to have proper Weldguide functionality.

Note that the signal names specified in the table above must be used.

3.1.4 Verifying communication

Verifying communication

A simple test can be used to check if the communication with the Weldguide board is set up correctly.

1. Create a simple weld program and activate tracking.
2. Block welding and run the program.

   If everything is working the green COMM led on the Weldguide board flashes on a high frequency, see *Status LEDs on page 24*.

Example program

```
PROC Weldguide()
  !
  MoveJ pApproachPos,z10,tWeldGun;
  ArcLStart p10, v1000, sm1, wd1\Weave:=wv1, fine,
        tWeldGun\Track:=tr1;
  ArcLEnd p20, v1000, sm1, wd1\Weave:=wv1, fine,
        tWeldGun\Track:=tr1;
  MoveJ pDepartPos, v1000, fine, tWeldGun;
  !
ENDPROC
```

3.1.5 Verifying configuration

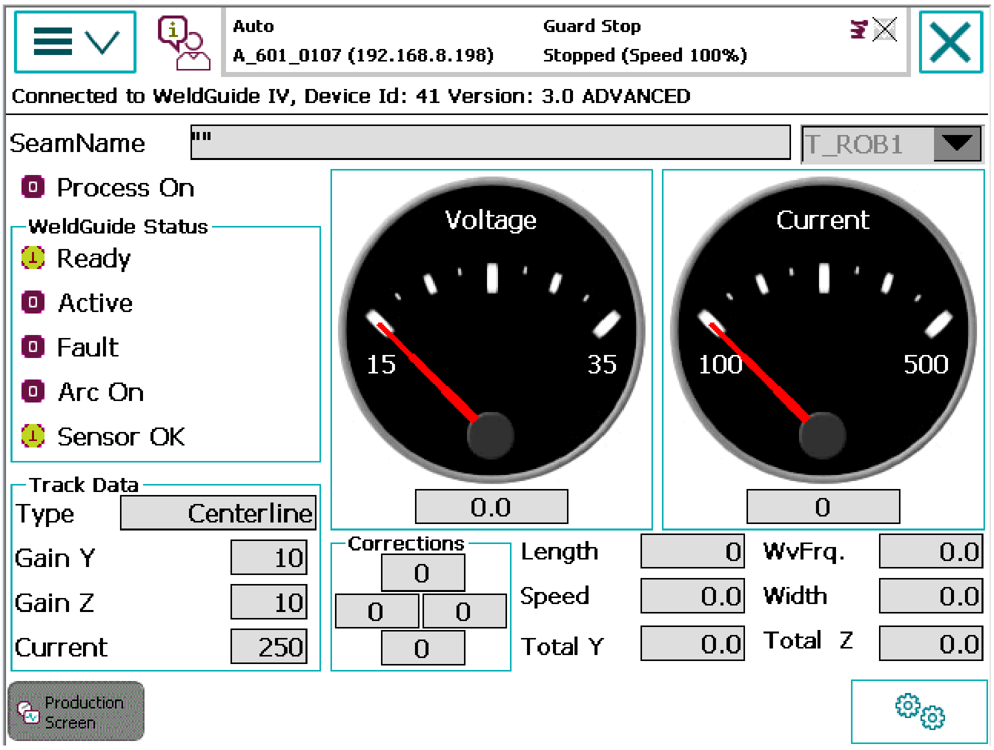
Startup check

During a warmstart, Weldguide checks the following:

1. Check if the Weldguide hardware is connected.
2. Check if the Volt/Ampere sensor is connected.
3. Check if the dwell bits (outputs) are configured and connected.

This might be useful to make sure that the equipment is configured correctly and that the sensor is connected correctly, see *Installation on page 17*. Any errors or warnings are written to the event log.

The Weldguide status is displayed on the FlexPendant.



xx1500000544

3.2 Configuring the system

Introduction

The first step in establishing Thru-Arc Tracking is to set up good stable welding parameters that prevent harsh arc conditions to include the arc start and end of weld.

The technology requires stable arc conditions to derive proper correction vector information. This may include having to set ramp-up and ramp-down conditions in the weld process. If the welding conditions are not under control the system will respond to the adverse conditions produced by an unstable welding process rather than to the actual conditions required for tracking and torch height control.

If there are drastic changes in the weld process (instability) the system will react in a drastic manor (i.e. the torch dives into the part or the torch loses the seam and wanders all over the welding surface).

Continues on next page

3.2.1 Tracking parameters

Gain_Y - Horizontal Gain

The recommended starting value is 15. This gain is used to increase or decrease the response of the cross-seam (horizontal) tracking. The lower the number the slower the system will respond to a change of seam direction. This variable impacts the stability of weld bead center. If the weld bead center position is oscillating (snake shape weld bead) decrease this parameter. If the center position is slow to respond to a change in the center position, increase this parameter. This value normally increases with a larger wire diameter.

Gain_Z - Vertical Gain

The recommended starting value is 30. This gain is used to increase or decrease the response of the torch height (vertical) tracking. The lower the number the slower the system will respond to changes to the work surface or geometry. This variable impacts the stability of torch height. If the torch position is oscillating (moving up and down constantly) decrease this parameter. If the torch position is slow to respond to a change in position, increase this parameter. This value normally increases with a larger wire diameter.

Depth of Penetration

This is only used with `trackmode` 1, 2, and 3 (adaptive and single side tracking). This variable sets the percent change from the weld bead center that the AWC will use to detect arc movement into a sidewall position. The percent change from center will determine the new extreme weave position for each weave cycle. Increasing this value will cause the arc to move harder into the sidewall. Decreasing this value will move the arc away from the sidewall.

3.3 Examples for seam tracking

Introduction

The following examples are simple guides on how to use a seam tracking welding system. Most of the tests can be done on a flat plate of steel.

Continues on next page

3.3.1 Checking the sensor

Example procedure

This is an example of how to make a simple test of the sensor.

1. Create a weld program with the weld gun straight up and down.

   The start point and end point should have a correct stick-out, for example 15 mm.
2. Weld and read the current from WGView. This is value is used as the reference in your `welddata` (`main_arc.current`).
3. Modify the start point to a short stick-out. Modify the end point to a long stick-out. See *Illustrations on page 43*.
4. Set the `trackmode` to 0 (centerline tracking).
5. Define a fairly fast weave with small width.
6. Set Y gain and Z gain to 0.
7. Weld and monitor the current meter in WGView. The current should change from high to low. Remember to add the current value in the `weldata` from your first weld.

Example parameters

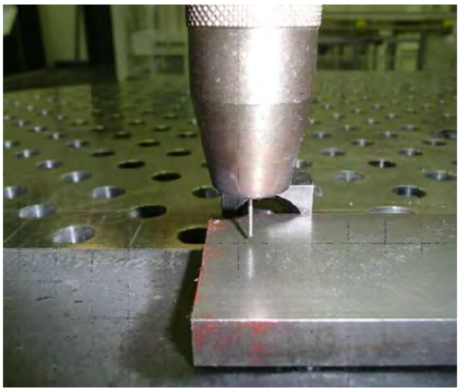
The following parameters have been developed using an ESAB Mig5000i welder and used for this test, they can used as start values.

Parameter	Value
Wirefeed speed	12 m/min
Process	Short/SprayArc
Material	Mild steel
Gas	Ar 18% CO2
Wire size	1 mm
Weld speed	10 mm/s
Current	220 A
Weave_shape	1
Weave_length	1
Weave_width	2

Continues on next page

Illustrations

10 mm stick-out at start point



xx1300000896

Example program

```
PROC Weldguide()
  !
  MoveJ pApproachPos,z10,tWeldGun;
  ArcLStart p10,v1000,sm1,wd1\Weave:=wv1,fine, tWeldGun\Track:=tr1;
  ArcLEnd p20,v1000, sm1,wd1\Weave:=wv1,fine, tWeldGun\Track:=tr1;
  MoveJ pDepartPos, v1000, fine, tWeldGun;
  !
ENDPROC
```

3.3.2 Tracking the height

Example procedure

This is an example of how to make a simple test of to check height tracking.

1. Create a weld program with the weld gun straight up and down. The start point and end point should have a long stick-out.
2. Set the `trackmode` to 0 (centerline tracking).
3. Define a fairly fast weave with small width.
4. Set Y gain and Z gain to 100.
5. Set target current to amperage from the start of the test.
6. Weld and see if the system moves down to the short stick-out. To high gain will cause oscillation. You should see corrections down to a shorter stick-out. Remember to add the current value in your `weldata` as a reference.

Example parameters

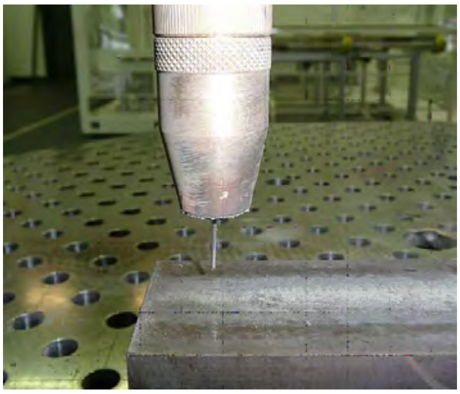
The following parameters have been developed using an ESAB Mig5000i welder and used for this test, they can used as start values.

Parameter	Value
Wirefeed speed	12 m/min
Process	Short/SprayArc
Material	Mild steel
Gas	Ar 18% CO2
Wire size	1 mm
Weld speed	10 mm/s
Current	220 A
Weave_shape	1
Weave_length	2
Weave_width	1
Y gain	50
Z gain	50

Continues on next page

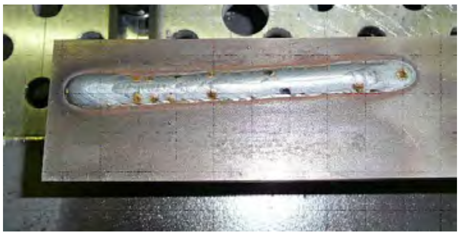
Illustrations

30 mm stick-out at start point



xx1300000900

Weld bead with Z correction



xx1300000902

Example program

```
PROC Weldguide()
  !
  MoveJ pApproachPos,z10,tWeldGun;
  ArcLStart p10,v1000,sm1,wd1\Weave:=wv1,fine, tWeldGun\Track:=tr1;
  ArcLEnd p20,v1000, sm1,wd1\Weave:=wv1,fine, tWeldGun\Track:=tr1;
  Stop;
  MoveJ pDepartPos, v1000, fine, tWeldGun;
  !
ENDPROC
```

Note

You can add a `Stop` instruction before moving away from the weld. If you do so you can move the program pointer to the `ArcLEnd` instruction (after you have welded) and execute step wise. The robot should move upwards to the taught position. This is to verify that the height corrections have been done.

3.3.3 Checking dwell bits

Example procedure

This is an example of how to make a simple test of to check dwell bits.

1. Create a weld program with the weld gun in an angle of approximately 45 degrees. The start point and the end point should have a short stick-out.
2. Set the `trackmode` to 0 (centerline tracking).
3. Change `weavedata` to get a slow weave that is really wide.
4. Set Y gain and Z gain to 50.
5. Weld and verify if the corrections are done (down and to one side with the longer stick-out).

Example parameters

The following parameters have been developed using an ESAB Mig5000i welder and used for this test, they can used as a start value.

Parameter	Value
Wirefeed speed	12 m/min
Process	Short/SprayArc
Material	Mild steel
Gas	Ar 18% CO2
Wire size	1 mm
Weld speed	10 mm/s
Current	220 A
Weave_shape	1
Weave_length	2
Weave_width	2.5
Y gain	50
Z gain	50

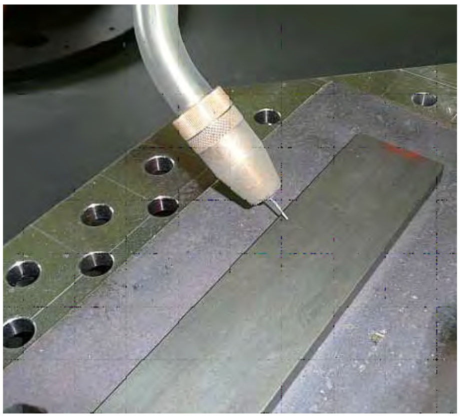
Illustrations

Start position for dwell bit check

xx1300000903

Continues on next page

End position for dwell bit check



xx1300000904

Corrections are done to the side with longer stick-out (A: programmed path)

xx1300000905

Example program

```
PROC Weldguide()
  !
  MoveJ pApproachPos,z10,tWeldGun;
  ArcLStart p10,v1000,sm1,wd1\Weave:=wv1,fine, tWeldGun\Track:=tr1;
  ArcLEnd p20,v1000, sm1,wd1\Weave:=wv1,fine, tWeldGun\Track:=tr1;
  MoveJ pDepartPos, v1000, fine, tWeldGun;
  !
ENDPROC
```

3.3.4 Creating a simple T-joint

Example procedure

This is an example of how to make a simple T-joint.

1. Program the path with proper stick-out using regular `ArcL` instructions and a simple T-joint configuration weld piece. Modify the start point having a good stick-out and a torch angle of 45 degrees. Use a displacement of about 5 mm in Y direction. The robot should move into the weld and then follow the weld without any overshoot. Modify the end point having a good stick-out and a torch angle of 45 degrees. The position has a displacement of about 5 mm in Z direction.
2. Define starting values for voltage and current in the active `welddata`.

Note

If a schedule based welder is used, set the desired schedule in `welddata` and set the current and voltage value to zero. Set centerline tracking (`track_type 0`) and set the gain values to zero in the active `trackdata`. For example:

```
MoveL p10,v1000,fine,tWeldGun;
ArcLStart p20,v1000,sm1,wd1\Weave:=wv1,fine,
      tWeldGun\Track:=tr1;
ArcLEnd p30,v1000,sm1,wd1\Weave:=wv1,fine,tWeldGun\Track:=tr1;
MoveL p10,v1000,fine,tWeldGun;
```

Remember to set the Weldguide sensor parameter *Pattern Sync Threshold* to at least 90. Also define `trackdata` (`tr1, max_corr=n`, where n is the maximum distance in mm from the programmed path).

3. Develop the weld data that gives the required weld size, voltage, wire feed speed, and robot travel speed. Add some weave to see how wide the weave can be and still get a good weld.

Tip

The wider the robot weaves the better the tracking. The rule of thumb is that the weave width should be minimum two times the wire diameter.

4. Weld and monitor the arc and the corrections shown in WGView. Update the weld current and arc voltage in the active `welddata`.

Note

If a schedule based welder is used, update only the current value. Remember to first ensure that proper path and stick-out was used. Also, make sure that tracking is not active when the real time values are checked. Tracking can be blocked via FlexPendant or I/O or by setting the gain values to zero.

Continues on next page

5 Make a weld program were both the start and the end points are outside the joint. Make a weld and look for the robot response. The robot should move into the weld and then follow the weld without any overshoot. The gain parameters are stored in the active `trackdata`. Make sure that tracking is not blocked via the FlexPendant or any I/O. For example:

```
MoveL p10,v1000,fine,tWeldGun;
ArcLStart
     p20,v1000,sm1,wd1\Weave:=wv1,fine,tWeldGun\Track:=tr1;
ArcLEnd p30,v1000,sm1,wd1\Weave:=wv1,fine,tWeldGun\Track:=tr1;
MoveL p10,v1000,fine,tWeldGun;
```

Example parameters

The following parameters have been developed using an ESAB Mig5000i welder and used for this test, they can used as a start values.

Parameter	Value
Wirefeed speed	12 m/min
Process	Short/SprayArc
Material	Mild steel
Gas	Ar 18% CO2
Wire size	1 mm
Weld speed	8 mm/s
Current	244 A
Weave_shape	1
Weave_length	3
Weave_width	3
Y gain	40
Z gain	40

Illustrations

Start position for the T-joint.

xx1300000906

Continues on next page

End position for the T-joint.

xx1300000907

Weld for the T-joint.

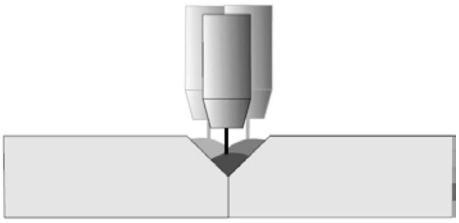
xx1300000908

3.4 Configuring MultiPass

Introduction

Sometimes multiple weld passes are required due to the required weld size and thickness of the material being joined. Weldguide makes this easy by tracking the first pass and storing the actual tracked path so it can offset for subsequent passes.

When building up a weld bead with multi layer welding, the first layer should provide good penetration at the bottom of the V groove. Additional layers must fuse this layer with the filler material and the sidewalls of the joint. The final layer seals the joint and should be crowned slightly above the base metal.



xx1300000879

Limitation for MultiPass

Path recovery does not work together with MultiPass because of a high risk for collisions.

The number of targets that can be saved is 1000. See *Example of calculating MinPointInc value on page 58*.

Storing a path

The first weld pass is recorded by making a weld with normal arc instructions. The following criteria must be fulfilled to record a path.

- The `store_path` flag must be defined in the active `trackdata`.
- There must be a weave pattern active when recording the first pass, that is, that the weave argument must be used in the Arc instructions.
- The same `SeamName` must be used in the Arc instructions when recording the path as when replaying with `ArcRepL`.
- The interval between stored path points is dictated by the weave length. MultiPass welding can be used in conjunction with seam tracking.

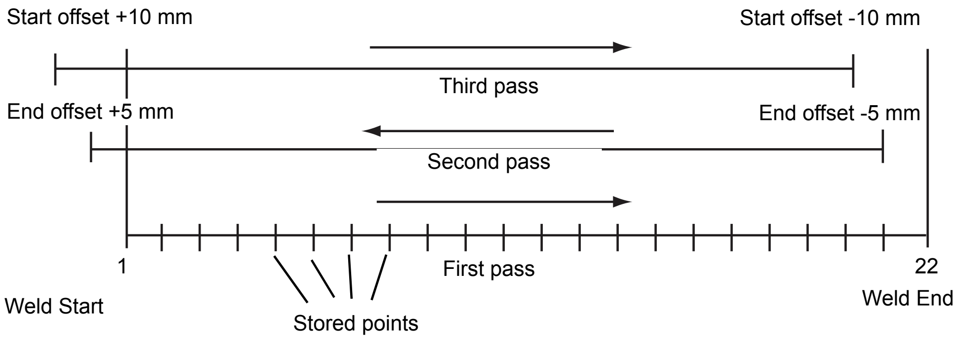
Replaying a stored path

The replayed path can be offset in either the plus or minus Y and Z seam coordinates and rotated plus or minus X and Y in seam coordinates. Replayed paths can also be executed in forward or reverse direction.

The start and end path points can be lengthened or shortened by a specified distance in millimeters. If the path is lengthened, the new end point is projected outward by using the last two points that were stored in the path. Lengthening and

Continues on next page

shortening the path allows for the weld to be tied into previous welds or the parent material itself.



xx1300000909

Continues on next page

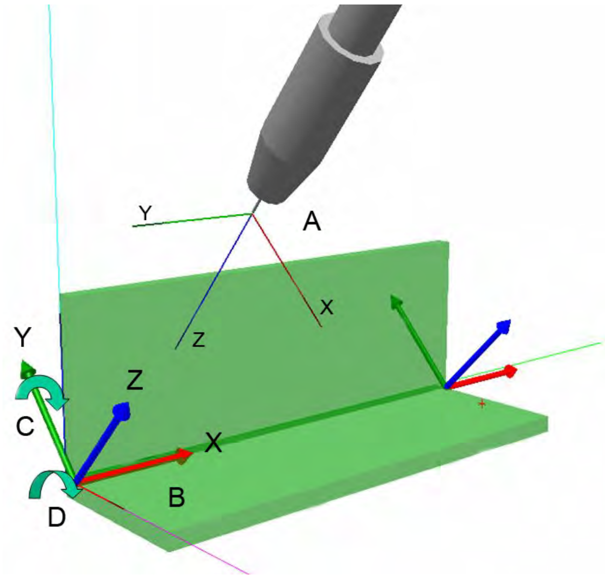
3.4.1 Example of storing and replaying a single weld

Storing and replaying a weld

1. Create a simple weld program using tracking.
2. Active storage by setting the `store_path` flag to `TRUE` in the active `trackdata`. The `trackdata` parameters are sent to Weldguide.
3. Use a weave pattern. It can be really small if weaving is not needed but it must be used to be able to store the path. The recorded path is connected to the `SeamName`, so this seam name must be used in the `ArcRepL` instruction to replay this path.
4. Replay the path using an `ArcRepL` instruction. This instruction replays the stored pass specified by the information contained in the multidata shown as `Layer_2`. `Layer_2` is the second weld pass with reversed direction, a new torch angle (-11 degrees push angle) and position offset with a start offset of -5 mm and an end offset of +5 mm.

Illustration of seam coordinate system

The offsets are calculated in seam coordinates.



xx1300000910

A Tool coordinates
B Seam coordinates
C Rotation Y
D Rotation X

Example program

```
CONST multidata Layer_2:=[-1,15,15,-5,5,2,2,5,-11];
PROC WeldguideMultiPath1()
  !
  MoveToHome;
  MoveJ pApproach, v1000, z10, PKI_500\WObj:=wobj0;
```

Continues on next page

```
  !
  ArcLStart p20, v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_1";
  ArcL p30,v100,sm1,wd1\Weave:=wvd,z1,
       PKI_500\WObj:=wobj0\Track:=trd1;
  ArcL p40,v100,sm1,wd1\Weave:=wvd,z1,
       PKI_500\WObj:=wobj0\Track:=trd1;
  ArcL p50,v100,sm1,wd1\Weave:=wvd,z1,
       PKI_500\WObj:=wobj0\Track:=trd1;
  ArcLEnd p60,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  !
  MoveL pDepart, v1000, z10, PKI_500\WObj:=wobj0;
  MoveToHome;
  !
  ArcRepL\Start\End,Layer_2,v100,sm1,wd1,wvd,z10,
       PKI_500\SeamName:=" Weld_1";
  !
  MoveToHome;
ENDPROC
```

Example program with additional layers

Additional layers can be welded by adding another `ArcRepL` instruction.

```
CONST multidata Layer_2:=[-1,15,15,-5,5,2,2,5,-11];
CONST multidata Layer_3:=[1,15,15,5,-5,2,2,5,11];
PROC WeldguideMultiPath1()
  !
  MoveToHome;
  MoveJ pApproach, v1000, z10, PKI_500\WObj:=wobj0;
  !
  ArcLStart p20, v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_1";
  ArcL p30,v100,sm1,wd1\Weave:=wvd,z1,
       PKI_500\WObj:=wobj0\Track:=trd1;
  ArcL p40,v100,sm1,wd1\Weave:=wvd,z1,
       PKI_500\WObj:=wobj0\Track:=trd1;
  ArcL p50,v100,sm1,wd1\Weave:=wvd,z1,
       PKI_500\WObj:=wobj0\Track:=trd1;
  ArcLEnd p60,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  !
  MoveL pDepart, v1000, z10, PKI_500\WObj:=wobj0;
  MoveToHome;
  !
  ArcRepL\Start\End,Layer_2,v100,sm1,wd1,wvd,z10,
       PKI_500\SeamName:=" Weld_1";
  ArcRepL\Start\End,Layer_3,v100,sm1,wd1,wvd,z10,
       PKI_500\SeamName:=" Weld_1";
  !
  MoveToHome;
ENDPROC
```

Continues on next page

Illustration of additional layers

One additional layer:

xx1300000911

Two additional layers:

xx1300000912

3.4.2 Example of storing and replaying multiple welds

Introduction

If more then one weld needs to be recorded then the path has to be saved before proceeding with the next seam. Each path must be loaded in the memory before it can be replayed with the `ArcRepL` instruction. This can be done with the instructions `MpSavePath` and `MpLoadPath`. The technique is shown in this example.

Advanced users can use the `MpReadInPath` instruction to modify the path data in the memory before storing such as:

- Adding an overlap.
- Adding external axis offsets.
- Spin angle (around Z-axis).
- Normalize a path (can be used to normalize a path if it appears unsteady).

See *MPReadInPath on page 72*.

Example program

```
PROC Weldguide_Pth_1()
  !
  MovetoHome;
  !
  MoveJ p22,v1000,z10,PKI_500\WObj:=wobj0;
  ArcLStart p23,v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_1";
  ArcLEnd p27,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  MoveL p28,v1000,z10,PKI_500\WObj:=wobj0;
  !
  MpSavePath "Part1_Weld_1"\SeamName:="Weld_1";
  !
  MoveJ p29,v1000,z10,PKI_500\WObj:=wobj0;
  ArcLStart p30,v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_2";
  ArcLEnd p34,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  MoveL p35,v1000,z10,PKI_500\WObj:=wobj0;
  !
  MpSavePath "Part1_Weld_2"\SeamName:="Weld_2";
  !
  MoveJ p29,v1000,z10,PKI_500\WObj:=wobj0;
  ArcLStart p30,v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_3";
  ArcLEnd p34,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  MoveL p35,v1000,z10,PKI_500\WObj:=wobj0;
  !
  MpSavePath "Part1_Weld_3"\SeamName:="Weld_3";
  !
  MovetoHome;
```

Continues on next page

```
    !
    MpLoadPath "Part1_Weld_1";
    ArcRepL\Start\End,Layer_1,v100,sm1,wd1,wvd,z10,
         PKI_500\SeamName:="Weld_1";
    !
    MpLoadPath "Part2_Weld_2";
    ArcRepL\Start\End, Layer_1,v100,sm1,wd1,wvd,z10,
         PKI_500\SeamName:="Weld_2";
    MoveAbsJ jtHome,v1000,z100,PKI_500\WObj:=wobj0;
    !
    MpLoadPath "Part1_Weld_3";
    ArcRepL\Start\End, Layer_1,v100,sm1,wd1,wvd,z10,
         PKI_500\SeamName:="Weld_3";
    !
    MoveToHome;
  ENDPROC
```

Program description

In this example three seams are welded in a row. Each seam is saved with the instruction `MpSavePath`. A filename and a seam name are specified in the instruction. This information is later used with the instructions `MpLoadPath` and `ArcRepL`.

A module is programmatically created with the instruction `MpSavePath` instruction. The module name is specified in the instruction and holds all the stored positions (`robtarget`). In this example `Part1_Weld_1`, `Part1_Weld_2`, and Part1_Weld_3.

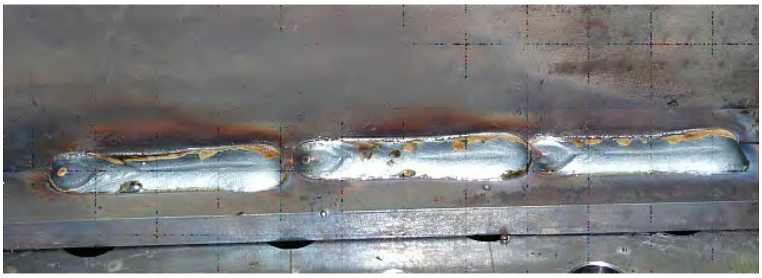
These modules are stored in the temp folder for the system.

CAUTION

A maximum of 1000 `robtargets` can be stored in the array for every seam. Do not modify these modules. This can cause unexpected robot movements that can damage the robot or the welding equipment.

Illustration of additional layers

One additional layer for each weld.



xx1300000913

3.4.3 Example of calculating MinPointInc value

Introduction

When using MultiPass, the limit for number of targets that can be read from the list of saved targets is 1000. If the number of targets exceeds this then no targets will be read and a message is displayed in the event log, **Stored path not complete**.

To solve this, the reading of targets can be done incrementally by using the option argument `\MinPointInc` in the instruction `MPReadInPath`. This means that the instruction only reads every x target, where x is the value of `MinPointInc`. See *MPReadInPath on page 72*.

Example message

Stored path not complete

Path in memory: ws21

Stored path: ws21

Distance between points to large

Index: 1850

Example solution

When the error message **Stored path not complete** is displayed, it also shows an index number. This is the number of targets that are saved for the path.

1. Calculate `MinPointInc` according to the following formula: *Index* (number of targets from the error message) / 1000.

   In this example: 1850/1000=1.85
2. Round up to the next integer.

   In this example: 2
3. Add the the value of `MinPointInc` in the instruction `MpReadInStoredPath`.

   In this example: `MinPointInc` is 2

4 ArcWelding PowerPac

Introduction

With RobotStudio and ArcWelding PowerPac, welding programs can be created offline. ArcWelding PowerPac supports all of the MultiPass and adaptive fill instructions.

This chapter gives an introduction to using MultiPass and adaptive filling in ArcWelding PowerPac.

Importing MultiPass instructions and data types

Use this procedure to import the MultiPass instructions automatically.

1. Create an arc welding station in RobotStudio.
2. Start ArcWelding PowerPac.
3. Acknowledge the message.

   The MultiPass instructions and data types are now imported automatically.
4. Use the MultiPass instructions from the instruction picklist.

Installing adaptive filling instructions

Use this procedure to install the adaptive filling manually.

1. Start ArcWelding PowerPac.
2. In the **Templates** tree view, right click on **Processes** and select **Import**.
3. Click to expand **RobotStudio**, then **ProcessPac**, then **ProcessTemplates**, then **Arc**.
4. Right-click to import the files ArcCalcDefault.xml and ArcAdaptDefault.xml.

   This will import all the `ArcCalc` and `ArcAdapt` instructions.
5. Use the adaptive fill instructions from the instructions picklist.

Note

Remember to change the process template when creating a weld.

Related information

Operating Manual - ArcWelding PowerPac

This page is intentionally left blank

5 Running in production

The FlexPendant application

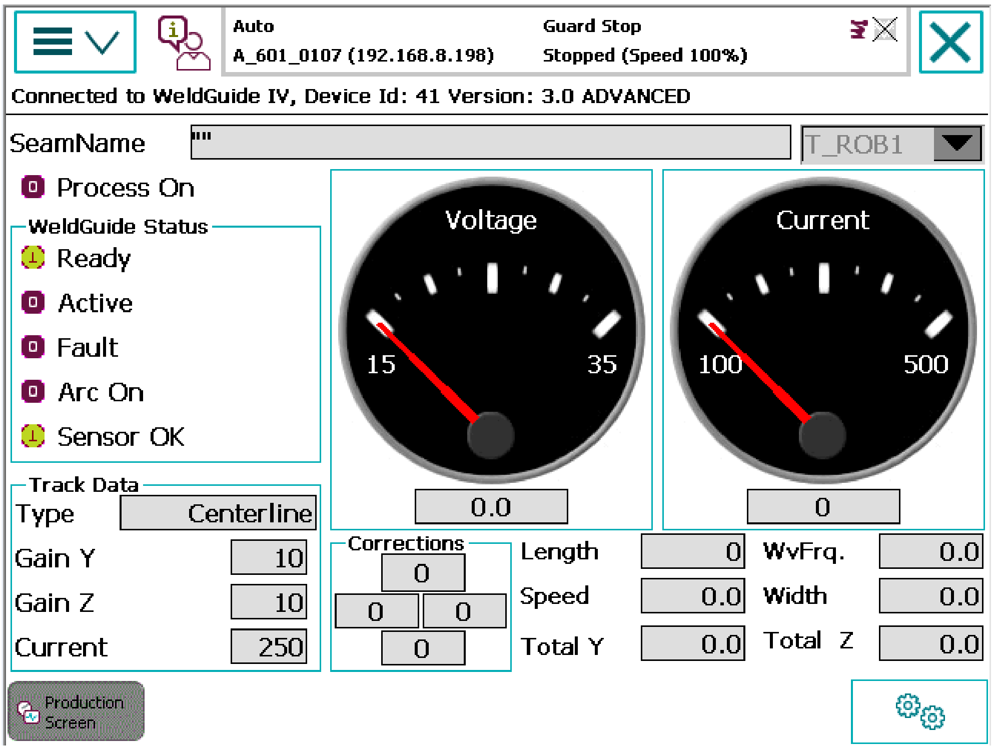
The Weldguide graphical user interface on the FlexPendant is called WGView. It is included in the option *815-2 Weldguide MultiPass*.

WGView shows valuable process information, such as:

- Real-time voltage and current.
- Real-time corrections for y and z-direction.
- Real-time `trackdata` values.
- Actual weld length.
- Actual weld speed.
- Accumulated corrections in Y and Z direction.
- Weave frequency.
- Weave width.
- Weldguide status signals.
- Actual seam name.

The minimum and maximum value for the analog meters can be configured in the system parameters, in the topic *Process*.

Illustration WGView



xx1500000544

Function	Description
SeamName	The active seam name.

Continues on next page

Function	Description
Process on	Active when a process instruction is executed.
Weldguide status	• **Ready** - Indicates that Weldguide is operational. • **Active** - Tracking is active. • **Fault** - A tracking fault is detected. • **Arc On** - Current is detected. • **Sensor OK** - Current sensor is working correctly. For information about the status LEDs on the Weldguide controller, see *Status LEDs on page 24*.
Track data	The `trackdata` used in the current instruction. • **Type** - Selected type of tracking in `trackdata`. • **Gain Y** - Gain used for Y-corrections in `trackdata`. • **Gain Z** - Gain used for Y-corrections in `trackdata`. • **Current** - Specified target current in the weld data.
Corrections	Correction in Y and Z directions. Max correction is indicated by red colored text.
Active data	The following parameters are read from active data. Data is updated if adaptive tracking mode is used. • **Length** - Weave length • **WvFrq.** - Weave frequency • **Speed** - Weld speed • **Width** - Weave width • **Total Y/Z** - Total correction in Y and Z.

6 RAPID reference

6.1 MultiPass instructions

6.1.1 ArcRepL

Usage

ArcRepL is used for replaying a stored path and can be used for one complete layer or a section of a layer. The path is stored by activating the `store_path` flag in the `trackdata` for normal arc instructions.

The `ArcRepL` instruction is used in MultiPass welding to replay a stored weld path without teaching each subsequent pass. The replayed path direction, start and end offset, Y and Z path offset, and Y and X torch rotation information are set in the `multidata`.

Note

It is recommended to use a zone z5 in this instruction. If a fine point is used in the `ArcRepL` instruction the weave will stop and restart at every path point recorded with the `store_path` flag in `trackdata`.

Basic examples

In the following program example, the `multidata` is named *Layer2* and can be noticed that `\Start` and `\End` is used in the same instruction therefore the entire weld process will be initiated and terminated in this single instruction. Transition welding can be accomplished by using separate `ArcRepL\Start` and `ArcRepL\End` instructions with unique multi, seam, weld, and weave data.

```
MoveL ...
ArcLStart *,v100,sm1,wdFL104m\Weave:=wvAdapt,fine,
      tWeldGun\Track:=tr1;
ArcLEnd *,v100,sm1,wdFL104m\Weave:=wvAdapt,fine,
      tWeldGun\Track:=tr1;
MoveL ...
ArcRepL\Start\End,Layer2,v100,sm1,wdFL10m,wv2,z5,tWeldGun;
MoveL ...
```

Arguments

```
ArcRepL [\Start] [\End] [\NoProcess] Offset [\StartInd]
[\EndInd] Speed, Seam, Weld, Weave, Zone, Tool, [\Wobj] [\Track]
[\SeamName] [\ServRoutine] [\TLoad]
```

[\Start]

Data type: `switch`

`\Start` is used at the start of a replay sequence. Regardless of what is specified in the `Zone` argument, the destination position will be a stop point.

Continues on next page

[\End]

Data type: `switch`

If `\End` is used, welding ends when the robot reaches the destination position (end of the stored path). Regardless of what is specified in the `Zone` argument, the destination position will be a stop point.

> **Note**
>
> For the `ArcRepL` instruction both the `Start` and `End` switch can be activated.

[\NoProcess]

Data type: `switch`

The `\NoProcess` argument is used if the instruction should be executed without the welding process active.

Offset

Data type: `multidata`

The `Offset` data contains the offset information for the path.

[\StartInd]

Data type: `num`

The optional argument `\StartInd` is used if the path should be replayed from a specific index instead of from the beginning of the stored path.

> **Note**
>
> First index in a path is always 1.

[\EndInd]

Data type: `num`

The optional argument `\EndInd` is used if the path should be replayed to a specific index not the end of the stored path. If a negative value is entered, the end index will be used as reference, for example -2 is index 2 from the end.

> **Note**
>
> First index in a path is always 1.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `Speed` data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

Seam

Data type: `seamdata`

Continues on next page

`Seam` data describes the start and end phases of a welding process. The argument `Seam` is included in all arc welding instructions so that, regardless of the position of the robot when the process is interrupted, a proper weld end and restart is achieved.

Weld

Data type: `welddata`

`Weld` data describes the weld phase of the welding process.

Weave

Data type: `weavedata`

`Weave` data describes the weaving that is to take place during the heat and weld phases. Welding without weaving is obtained by specifying, for example, the weave data `noweave` (no weaving if the `weave_shape` component value is zero).

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 case of a fly-by point, a corner path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

[\Track]

Data type: `trackdata`

`Track` data describes the parameters used for tracking.

[\SeamName]

Data type: `string`

`SeamName` defines the name used in error logs if an error occurs during the welding sequence. `\SeamName` can only be used in the first instruction of a sequence of weld instructions, that is, together with the `\Start` argument. The `SeamName` in

Continues on next page

the `ArcRepL` instruction specifies which path to replay, so the `SeamName` must be the same as the `SeamName` used to record the path.

[\ServRoutine]

Data type: `string`

A service routine can be specified and used together with the `Escape` selection in the *Weld Error Recovery* menu. If `Escape` is selected, the robot will reverse back along the recorded path to the first recorded point on the path with the speed and offset specified by the settings in Arc Error Handler in the process configuration.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Error handling

The process is supervised by a number of signal inputs. If anything abnormal is detected, program execution will stop. For more information about error handling, see *Application manual - Arc and Arc Sensor*.

Program execution

The process equipment is controlled by the robot so that the entire process and each of its phases are coordinated with the robot's movement.

Syntax

```
ArcRepL
  [ '\' Start ',' ] < expression (IN) of switch >
  [ '\' End ',' ] < expression (IN) of switch >
  [ '\' NoProcess ',' ] < expression (IN) of switch >'
  [ Offset ':=' ] < expression (IN) of multidata >','
  [ '\' StartInd ':=' < expression (IN) of num > ';'
  [ '\' EndInd ':=' < expression (IN) of num > ';'
  [ Speed ':=' ] < expression (IN) of speeddata >','
  [ Seam ':=' ] < persistent (PERS) of seamdata > ','
  [ Weld ':=' ] < persistent (PERS) of welddata > ','
  [ Weave ':=' ] < persistent (PERS) of weavedata > ','
  [ Zone ':=' ] < expression (IN) of zonedata >','
  [ Tool ':=' ] < persistent (PERS) of tooldata >','
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ';'
  [ '\' Track ':=' ] < persistent (PERS) of trackdata >','
  [ '\' SeamName ':=' < expression (IN) of string > ]
  [ '\' ServRoutine ':=' < expression (IN) of string > ]
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Continues on next page

Related information

trackdata on page 105

multidata on page 108

Technical reference manual - RAPID Instructions, Functions and Data types

Technical reference manual - RAPID overview

6.1.2 MPSavePath

Usage

`MpSavePath` is used to save a path in memory to a RAPID file. This file can then later be loaded in to memory with the `MpLoadPath` instruction. This feature can be used if the replay operation has to be done later. It is only necessary to save the path to a file, if another path is to be stored before replaying this path with the replay instructions. The default path for saving the rapid file is `TEMP: /`. A separate subfolder is created for each robot.

Basic example

```
PROC Path_1()
  MoveAbsJ jtHome,v1000,fine,PKI_500\WObj:=wobj0;
  ArcLStart p10,v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_1";
  ArcLEnd p20,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  !
  MpSavePath "Part1_Weld_1"\SeamName:="Weld_1";
  !
  ArcLStart p30,v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_2";
  ArcLEnd p40,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  !
  MpSavePath "Part1_Weld_2"\SeamName:="Weld_2";
  !
ENDPROC
```

Arguments

`MpSavePath, FileName, [\SeamName] [CreateLogFile]`

FileName

Data type: `string`

When the path is stored it is associated with a `Filename` in the event that multiple paths are stored. Therefore when reading a path in the `Filename` must be specified.

[\SeamName]

Data type: `string`

`SeamName` is used to identify the seam in the stored file. The argument must be used when storing the path.

[\CreateLogFile]

Data type: `switch`

Different log files will be created. The log files affected are called StoredPath.csv and ReadInPathLogFile.csv. Also a log with the name specified as `FileName` (used to save the module) are available in the `TEMP` directory of the robot.

Continues on next page

Error handling

If anything abnormal is detected a message is written to the elog file.

Program execution

A weld path is executed and recorded. The instruction `MpSavePath` is executed and the recorded weld path is saved with a specific `FileName`. The saved path can be loaded with the `MpReadInPath` instruction and can be replayed using the replay instructions.

Syntax

```
MpSavePath
  [ FileName:='] < expression (IN) of string >','
  [ \SeamName:='] < expression (IN) of string >','
  [ '\' CreateLogFile ','] < expression (IN) of switch > ","
```

Related information

MPLoadPath on page 70

6.1.3 MPLoadPath

Usage

`MpLoadPath` is used to load a path in to memory which was stored earlier with the `MpSavePath` instruction.

Basic example

```
PROC Weldguide_Pth_1()
  !
  MovetoHome;
  !
  MoveJ p22,v1000,z10,PKI_500\WObj:=wobj0;
  ArcLStart p23,v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_1";
  ArcLEnd p27,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  MoveL p28,v1000,z10,PKI_500\WObj:=wobj0;
  !
  MpSavePath "Part1_Weld_1"\SeamName:="Weld_1";
  !
  MoveJ p29,v1000,z10,PKI_500\WObj:=wobj0;
  ArcLStart p30,v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_2";
  ArcLEnd p34,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  MoveL p35,v1000,z10,PKI_500\WObj:=wobj0;
  !
  MpSavePath "Part1_Weld_2"\SeamName:="Weld_2";
  !
  MoveJ p29,v1000,z10,PKI_500\WObj:=wobj0;
  ArcLStart p30,v1000,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1\SeamName:="Weld_3";
  ArcLEnd p34,v100,sm1,wd1\Weave:=wvd,fine,
       PKI_500\WObj:=wobj0\Track:=trd1;
  MoveL p35,v1000,z10,PKI_500\WObj:=wobj0;
  !
  MpSavePath "Part1_Weld_3"\SeamName:="Weld_3";
  !
  MovetoHome;
  !
  MpLoadPath "Part1_Weld_1";
  ArcRepL\Start\End,Layer_1,v100,sm1,wd1,wvd,z10,
       PKI_500\SeamName:="Weld_1";
  !
  MpLoadPath "Part2_Weld_2";
  ArcRepL\Start\End, Layer_1,v100,sm1,wd1,wvd,z10,
       PKI_500\SeamName:="Weld_2";
  MoveAbsJ jtHome,v1000,z100,PKI_500\WObj:=wobj0;
  !
  MpLoadPath "Part1_Weld_3";
```

Continues on next page

```
    ArcRepL\Start\End, Layer_1,v100,sm1,wd1,wvd,z10,
        PKI_500\SeamName:="Weld_3";
    !
    MoveToHome;
  ENDPROC
```

Arguments

`MpLoadPath FileName`

FileName

Data type: `string`

When the path is stored it is associated with a `Filename` in the event that multiple paths are stored. Therefore when reading a path in, the `Filename` must be specified.

Error handling

If the file cannot be loaded from the `Temp` folder or if there is not enough program memory available to load the file, then a message is written to the log file.

Program execution

The instruction `MpLoadPath` is executed and the recorded weld path is loaded with a specific `FileName`.

Syntax

```
  MpLoadPath
    [ FileName:='] < expression (IN) of string >','
```

Related information

MPSavePath on page 68

6.1.4 MPReadInPath

Usage

`MPReadInPath` is used to read in a stored path to memory. The path and adaptive data are stored in an internal file during execution. This file is normally read automatically when a replay instruction is executed. If any special operations are required during the read phase, the `MpReadInPath` instruction can be executed before the replay instruction.

Arguments

```
MpReadInPath [\Overlap] [\SeamName] [\OffsEax_a] [\OffsEax_b]
[\OffsEax_c] [\OffsEax_b] [\OffsEax_e] [\OffsEax_f] [\SpinAngle]
[\NormalizePath] [\MinPointInc] [\MaxPointInc]
[\MaxPathDeviation] [\SavePathFileName] [\CreateLogFile]
[\PointInc ]
```

[\Overlap]

Data type: `num`

[\SeamName]

Data type: `string`

`SeamName` is used to identify the seam in the stored file. The argument must be used when storing the path.

[\OffsEax_a]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_b]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_c]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_d]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_e]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_f]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

Continues on next page

[\SpinAngle]

Data type: `num`

Spins all targets on the path in the memory around its Z-axis.

[\NormalizePath]

Data type: `switch`

[\MinPointInc]

Data type: `num`

Defines an increment (ratio) of how many targets are read. If `MinPointInc` is defined as 1, then every saved target is read. If `MinPointInc` is defined as 2, then every second saved target is read.

Note

When using MultiPass, the limit for number of targets that can be read from the list of saved targets is 1000. If the number of targets exceeds this then no targets will be read and a message is displayed in the event log, **Stored path not complete.**

If this error message is displayed, see *Example of calculating MinPointInc value on page 58*.

[\MaxPointInc]

Data type: `num`

[\MaxPathDeviation]

Data type: `num`

[\SavePathFileName]

Data type: `string`

When the path is stored it is associated with a `Filename` in the event that multiple paths are stored. Therefore while reading a path the `Filename` must be specified. This optional argument can be used instead of the `MpSavePath` instruction.

[\CreateLogFile]

Data type: `switch`

[\PointInc]

Data type: `num`

Syntax

```
MpReadInPath
  [ '\' Overlap ':='] < Expression (IN) of num >
  [ '\' SeamName ':=']< Expression (IN) of string >
  [ '\' OffsEax_a ':='] < Expression (IN) of num >
  [ '\' OffsEax_b ':='] < Expression (IN) of num >
  [ '\' OffsEax_c ':='] < Expression (IN) of num >
  [ '\' OffsEax_d':='] < Expression (IN) of num >
  [ '\' OffsEax_e ':='] < Expression (IN) of num >
  [ '\' OffsEax_f ':='] < Expression (IN) of num >
```

Continues on next page

6.1.4 MPReadInPath

Continued

```
[ '\' SpinAngle ':='] < Expression (IN) of num >
[ '\' NormalizePath ','] < Expression (IN) of switch >
[ '\' MinPointInc ':='] < Expression (IN) of num >
[ '\' MaxPointInc ':='] < Expression (IN) of num >
[ '\' MaxPathDeviation ':='] < Expression (IN) of num >
[ '\' SavePathFileName ':=']< Expression (IN) of string >
[ '\' CreateLogFile','] < Expression (IN) of switch >
[ '\' PointInc ':='] < Expression (IN) of num >
```

Related information

MPSavePath on page 68

6.1.5 MPOffsEaxOnPath

Usage

`MpOffsEaxOnPath` is used to add on offset to the additional axes values on the path in memory. When replaying a coordinated path it is sometimes an advantage to execute the replayed path with the part in a slightly different orientation. To do this, add an offset to all the points in the path in memory.

Arguments

`MpOffsEaxOnPath [\OffsEax_a] [\OffsEax_b] [\OffsEax_c] [\OffsEax_d] [\OffsEax_e] [\OffsEax_f] [\SpinAngle]`

[\OffsEax_a]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_b]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_c]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_d]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_e]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\OffsEax_f]

Data type: `num`

Add an offset to the additional axes values on the path in memory.

[\SpinAngle]

Data type: `num`

Spins all positions (`robtarget`) on the path in the memory around its z-axis.

Syntax

```
MpOffsEaxOnPath
  [ '\' OffsEax_a ':='] < Expression (IN) of num >
  [ '\' OffsEax_b ':='] < Expression (IN) of num >
  [ '\' OffsEax_c ':='] < Expression (IN) of num >
  [ '\' OffsEax_d':='] < Expression (IN) of num >
  [ '\' OffsEax_e ':='] < Expression (IN) of num >
  [ '\' OffsEax_f ':='] < Expression (IN) of num >
  [ '\' SpinAngle ':='] < Expression (IN) of num >
```

6.2 Adaptive fill instructions

6.2.1 ArcAdaptLStart

Usage

`ArcAdaptLStart` is used for adaptive tracking. Weave width and weld speed are updated based on data from the tracking system.

Example

```
CONST adaptdata adCalc1:=[20,1,0,0,2,100,1,30];
PROC Adaptive_1_ViaPoint_Pth_1()
  !
  MoveJ p10,v1000,z10,tWeldGun\WObj:=wobj0;
  !
  ArcCalcLStartp20,v1000,10,adCalc1,sm1,wdWg1,wvWg2,fine,
       tWeldGun\WObj:=wobj0,trWg1\SeamName:="Weld_1";
  !
  ArcCalcL p30,v100,60,adCalc1,z5,tWeldGun\WObj:=wobj0;
  !
  ArcCalcLEnd p40,v100,10,adCalc1,fine, tWeldGun\WObj:=wobj0;
  !
  MoveL p50,v1000,z10,tWeldGun\WObj:=wobj0;
ENDPROC
```

Arguments

`ArcAdaptLStart ToPoint [\ID] Speed, GrooveWidth, Adapt, Seam, Weld, Weave, Zone, Tool, [\WObj] Track [\SeamName] [\TLoad]`

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `Speed` data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

Continues on next page

GrooveWidth

Datatype: `num`

The `GrooveWidth` is used to calculate the initial weave width and weld speed. The groove width is normally the result of a groove search.

Adapt

Data type: `adaptdata`

Data structure with parameters for calculating the initial settings.

Seam

Data type: `seamdata`

`Seam` data describes the start and end phases of a welding process. The argument `Seam` is included in all arc welding instructions so that, regardless of the position of the robot when the process is interrupted, a proper weld end and restart is achieved.

Weld

Data type: `welddata`

`Weld` data describes the weld phase of the welding process.

Weave

Data type: `weavedata`

`Weave` data describes the weaving that is to take place during the heat and weld phases. Welding without weaving is obtained by specifying, for example, the weave data `noweave` (no weaving if the `weave_shape` component value is zero).

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 case of a fly-by point, a corner path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to

Continues on next page

the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

Track

Data type: `trackdata`

`Track` data describes the parameters used for tracking.

[\SeamName]

Data type: `string`

`SeamName` defines the name used in error logs if an error occurs during the welding sequence. `\SeamName` can only be used in the first instruction of a sequence of weld instructions, that is, together with the `\Start` argument. The `SeamName` in the `ArcRepL` instruction specifies which path to replay, so the `SeamName` must be the same as the `SeamName` used to record the path.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Description

The `ArcAdaptLStart` instruction replaces the traditional `ArcLStart` instruction when welding a path where using adaptive process parameters is required.

Nominal weld and weave data are used in the `ArcAdaptL` instruction. The same seam and weld data are used in all welding instructions thereafter. These two data are continuously changing depending on varying joint conditions.

If storage of the path is required then activate the `store_path` flag in the used `trackdata`.

Syntax

```
ArcAdaptLStart
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
  [ GrooveWidth ','] < Expression (IN) of num>'
  [ Adapt ':='] < Expression (IN) of adaptdata >','
  [ Seam ':='] < persistent (PERS) of seamdata > ','
  [ Weld ':='] < persistent (PERS) of welddata > ','
  [ Weave ':='] < persistent (PERS) of weavedata > ','
  [ Zone ':='] < Expression (IN) of zonedata >','
  [ Tool ':='] < persistent (PERS) of tooldata >','
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ','
  [ Track ':='] < persistent (PERS) of trackdata >','
```

Continues on next page

```
[ '\' SeamName ':=' < expression (IN) of string >]
[ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

trackdata on page 105

6.2.2 ArcAdaptL

Usage

`ArcAdaptL` is used for adaptive tracking. Weave width and weld speed are updated based on data from the tracking system. `ArcAdaptL` is a via instruction, that inherits seam, weld, weave, and track data from the `ArcAdaptLStart` instruction.

Arguments

```
ArcAdaptL ToPoint [\ID], Speed, Zone, Tool [\WObj] [\TLoad]
```

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `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 case of a fly-by point, a corner path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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.

Continues on next page

[\WObj]

Data type: `wobjdata`

The work object (coordinate system) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Syntax

```
ArcAdaptL
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
  [ Zone ':='] < Expression (IN) of zonedata >','
  [ Tool ':='] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata >
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcAdaptLStart on page 76

6.2.3 ArcAdaptC

Usage

`ArcAdaptC` is used for adaptive tracking. Weave width and weld speed are updated based on data from the tracking system. `ArcAdaptC` is a via instruction, that inherits seam, weld, weave, and track data from the `ArcAdaptLStart` instruction.

Arguments

```
ArcAdaptC CirPoint, ToPoint [\ID] Speed, Zone, Tool [\WObj]
[\TLoad]
```

CirPoint

Data type: `robtarget`

The circle point of the robot. The circle point is a position on the circle between the start point and the destination point. To obtain the best accuracy, it should be placed about halfway between the start and destination points. If it is placed too close to the start or destination point, the robot may give a warning. The circle point is defined as a named position or stored directly in the instruction (marked with an * in the instruction).

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `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 case of a fly-by point, a corner path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld (even in the case of

Continues on next page

a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Syntax

```
ArcAdaptC
  [ CirPoint ':='] < Expression (IN) of robtarget > ','
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
  [ Zone ':='] < Expression (IN) of zonedata >','
  [ Tool ':='] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata >
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcAdaptLStart on page 76

6.2.4 ArcAdaptLEnd

Usage

`ArcAdaptLEnd` is used for adaptive tracking. Weave width and weld speed are updated based on data from the tracking system. `ArcAdaptLEnd` is the process end instruction, that inherits seam, weld, weave, and track data from the `ArcAdaptLStart` instruction.

Arguments

```
ArcAdaptLEnd ToPoint [\ID] Speed, Zone, Tool [\WObj] [\TLoad]
```

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `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 case of a fly-by point, a corner path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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.

Continues on next page

[\WObj]

Data type: `wobjdata`

The work object (coordinate system) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Syntax

```
ArcAdaptLEnd
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
  [ Zone ':='] < Expression (IN) of zonedata >','
  [ Tool ':='] < persistent (PERS) of tooldata >','
  [ '\' WObj ':=' < persistent (PERS) of wobjdata >
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcAdaptLStart on page 76

6.2.5 ArcAdaptCEnd

Usage

`ArcAdaptCEnd` is used for adaptive tracking. Weave width and weld speed are updated based on data from the tracking system. `ArcAdaptCEnd` is the process end instruction, that inherits seam, weld, weave and track data from the `ArcAdaptLStart` instruction.

Arguments

```
ArcAdaptCEnd CirPoint, ToPoint [\ID] Speed, Zone, Tool [\WObj]
[\TLoad]
```

CirPoint

Data type: `robtarget`

The circle point of the robot. The circle point is a position on the circle between the start point and the destination point. To obtain the best accuracy, it should be placed about halfway between the start and destination points. If it is placed too close to the start or destination point, the robot may give a warning. The circle point is defined as a named position or stored directly in the instruction (marked with an * in the instruction).

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `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 case of a fly-by point, a corner path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is

Continues on next page

always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Syntax

```
ArcAdaptCEnd
  [ CirPoint ':='] < Expression (IN) of robtarget > ','
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
  [ Zone ':='] < Expression (IN) of zonedata >','
  [ Tool ':='] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata >
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcAdaptLStart on page 76

6.2.6 ArcCalcLStart

Usage

ArcCalcLStart is used for adaptive tracking and welding adapted to the measured groove width. Weave width and weld speed are updated based on measured grove width.

Arguments

```
ArcCalcLStart ToPoint [\ID] Speed, GrooveWidth, Adapt
[SpeedGain] [AdaptToMinMax] Seam, Weld, Weave, Zone, Tool,
[\WObj] Track, [\SeamName] [\TLoad]
```

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `Speed` data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

GrooveWidth

Datatype: `num`

The `GrooveWidth` is used to calculate the initial weave width and weld speed. The groove width is normally the result of a groove search.

Adapt

Data type: `adaptdata`

Data structure with parameters for calculating the initial settings.

[SpeedGain]

Data type: `num`

Minimum value 0.5, maximum value 1.5.

[AdaptToMinMax]

Data type: `switch`

Continues on next page

Seam

Data type: `seamdata`

`Seam` data describes the start and end phases of a welding process. The argument `Seam` is included in all arc welding instructions so that, regardless of the position of the robot when the process is interrupted, a proper weld end and restart is achieved.

Weld

Data type: `welddata`

`Weld` data describes the weld phase of the welding process.

Weave

Data type: `weavedata`

`Weave` data describes the weaving that is to take place during the heat and weld phases. Welding without weaving is obtained by specifying, for example, the weave data `noweave` (no weaving if the `weave_shape` component value is zero).

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 case of a fly-by point, a corner path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

Track

Data type: `trackdata`

`Track` data describes the parameters used for tracking.

[\SeamName]

Data type: `string`

Continues on next page

`SeamName` defines the name used in error logs if an error occurs during the welding sequence. `\SeamName` can only be used in the first instruction of a sequence of weld instructions, that is, together with the `\Start` argument. The `SeamName` in the `ArcRepL` instruction specifies which path to replay, so the `SeamName` must be the same as the `SeamName` used to record the path.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Syntax

```
ArcCalcLStart
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
  [ GrooveWidth ','] < Expression (IN) of num>'
  [ Adapt ':='] < Expression (IN) of adaptdata >','
  [ '\' SpeedGain':='] < Expression (IN) of num>','
  [ '| AdaptToMinMax: '='] < switch>','
  [ Seam ':='] < persistent (PERS) of seamdata > ','
  [ Weld ':='] < persistent (PERS) of welddata > ','
  [ Weave ':='] < persistent (PERS) of weavedata > ','
  [ Zone ':='] < Expression (IN) of zonedata >','
  [ Tool ':='] < persistent (PERS) of tooldata >','
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ','
  [ Track ':='] < persistent (PERS) of trackdata >','
  [ '\' SeamName ':=' < expression (IN) of string >]
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcCalcLStart on page 88

6.2.7 ArcCalcL

Usage

`ArcCalcL` is used for adaptive tracking and welding adapted to the measured groove width. Weave width and weld speed are updated based on measured grove width. `ArcCalcL` is a via instructions, that inherits seam, weld, weave, and track data from the `ArcCalcLStart` instruction.

Arguments

```
ArcCalcL ToPoint [\ID] Speed, GrooveWidth, Adapt, Zone, Tool
[\WObj] [\TLoad]
```

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `Speed` data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

GrooveWidth

Datatype: `num`

The `GrooveWidth` is used to calculate the initial weave width and weld speed. The groove width is normally the result of a groove search.

Adapt

Data type: `adaptdata`

Data structure with parameters for calculating the initial settings.

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

Continues on next page

always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

Track

Data type: `trackdata`

`Track` data describes the parameters used for tracking.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Syntax

```
ArcCalcL
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
  [ GrooveWidth ','] < Expression (IN) of num>'
  [ Adapt ':='] < Expression (IN) of adaptdata >','
  [ Zone ':='] < Expression (IN) of zonedata >','
  [ Tool ':='] < persistent (PERS) of tooldata >','
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ','
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcCalcLStart on page 88

6.2.8 ArcCalcC

Usage

`ArcCalcC` is used for adaptive tracking, based and adapted to the measured groove width. Weave width and weld speed are updated based on measured grove width. `ArcCalcC` is a via instruction that inherits seam, weld, weave, and track data from the `ArcCalcLStart` instruction.

Arguments

```
ArcCalcC CirPoint, ToPoint [\ID] Speed, GrooveWidth, Adapt,
Zone, Tool [\WObj] [\TLoad]
```

CirPoint

Data type: `robtarget`

The circle point of the robot. The circle point is a position on the circle between the start point and the destination point. To obtain the best accuracy, it should be placed about halfway between the start and destination points. If it is placed too close to the start or destination point, the robot may give a warning. The circle point is defined as a named position or stored directly in the instruction (marked with an * in the instruction).

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `Speed` data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

GrooveWidth

Datatype: `num`

The `GrooveWidth` is used to calculate the initial weave width and weld speed. The groove width is normally the result of a groove search.

Continues on next page

Adapt

Data type: `adaptdata`

Data structure with parameters for calculating the initial settings.

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 case of a fly-by point, a corner path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

Track

Data type: `trackdata`

`Track` data describes the parameters used for tracking.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Syntax

```
ArcCalcC
  [ CirPoint ':='] < Expression (IN) of robtarget > ','
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
```

Continues on next page

```
[ GrooveWidth ','] < Expression (IN) of num>'
[ Adapt ':='] < Expression (IN) of adaptdata >','
[ Zone ':='] < Expression (IN) of zonedata >','
[ Tool ':='] < persistent (PERS) of tooldata >','
[ '\' WObj ':=' < persistent (PERS) of wobjdata > ','
[ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcCalcLStart on page 88

6.2.9 ArcCalcLEnd

Usage

`ArcCalcLEnd` is used for adaptive tracking based and adapted to the measured groove width. Weave width and weld speed are updated based on measured grove width. `ArcCalcLEnd` is the process end instruction, that inherits seam, weld, weave, and track data from the `ArcCalcLStart` instruction.

Arguments

```
ArcCalcLEnd ToPoint [\ID] Speed, GrooveWidth, Adapt, Zone, Tool,
[\WObj] [\TLoad]
```

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `Speed` data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

GrooveWidth

Datatype: `num`

The `GrooveWidth` is used to calculate the initial weave width and weld speed. The groove width is normally the result of a groove search.

Adapt

Data type: `adaptdata`

Data structure with parameters for calculating the initial settings.

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

Continues on next page

always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Syntax

```
ArcCalcLEnd
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
  [ GrooveWidth ','] < Expression (IN) of num>'
  [ Adapt ':='] < Expression (IN) of adaptdata >','
  [ Zone ':='] < Expression (IN) of zonedata >','
  [ Tool ':='] < persistent (PERS) of tooldata >','
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ','
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcCalcLStart on page 88

6.2.10 ArcCalcCEnd

Usage

`ArcCalcCEnd` is used for adaptive tracking based and adapted to the measured groove width. Weave width and weld speed are updated based on measured grove width. `ArcCalcCEnd` is the process end instruction, that inherits seam, weld, weave, and track data from the `ArcCalcLStart` instruction.

Arguments

```
ArcCalcLEnd CirPoint, ToPoint [\ID] Speed, GrooveWidth, Adapt,
Zone, Tool, [\WObj] [\TLoad]
```

CirPoint

Data type: `robtarget`

The circle point of the robot. The circle point is a position on the circle between the start point and the destination point. To obtain the best accuracy, it should be placed about halfway between the start and destination points. If it is placed too close to the start or destination point, the robot may give a warning. The circle point is defined as a named position or stored directly in the instruction (marked with an * in the instruction).

ToPoint

Data type: `robtarget`

The destination position of the robot and additional axes. The position is defined as a named position or stored in the instruction.

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in the MultiMove systems, if the movement is synchronized or coordinated synchronized. This argument is not allowed in any other case. The specified id number must be the same in all the cooperating program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `Speed` data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

GrooveWidth

Datatype: `num`

The `GrooveWidth` is used to calculate the initial weave width and weld speed. The groove width is normally the result of a groove search.

Continues on next page

Adapt

Data type: `adaptdata`

Data structure with parameters for calculating the initial settings.

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 case of a fly-by point, a corner path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Syntax

```
ArcCalcCEnd
  [ CirPoint ':='] < Expression (IN) of robtarget > ','
  [ ToPoint ':='] < Expression (IN) of robtarget > ','
  [ '\' ID ','] < Expression (IN) of identno > ','
  [ Speed ':='] < Expression (IN) of speeddata >','
  [ GrooveWidth ','] < Expression (IN) of num>'
  [ Adapt ':='] < Expression (IN) of adaptdata >','
  [ Zone ':='] < Expression (IN) of zonedata >','
  [ Tool ':='] < persistent (PERS) of tooldata >','
```

Continues on next page

```
[ '\' WObj ':=' < persistent (PERS) of wobjdata > ','
[ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcCalcLStart on page 88

6.2.11 ArcAdaptRepL

Usage

`ArcAdaptRepL` works like the `ArcRepL` instruction but uses the adaptive data stored in addition to the stored points. If the path was stored using an adaptive instruction (like `ArcAdapt` or `ArcCalc`), the weave width and weld speed will be modified based on the stored information.

Arguments

```
ArcAdaptRepL [\Start] [\End] [\NoProcess] Offset, Adapt
[\SpeedGain] [| AdaptToMinMax] [\StartInd] [\EndInd] Speed,
Seam, Weld, Weave, Zone, Tool [\Wobj] [\Track] [\SeamName]
[\ServRoutine] [\TLoad]
```

[\Start]

Data type: `switch`

`\Start` is used at the start of a replay sequence. Regardless of what is specified in the `Zone` argument, the destination position will be a stop point.

[\End]

Data type: `switch`

If `\End` is used, welding ends when the robot reaches the destination position (end of the stored path). Regardless of what is specified in the `Zone` argument, the destination position will be a stop point.

Note

For the `ArcRepL` instruction both the `Start` and `End` switch can be activated.

[\NoProcess]

Data type: `switch`

The `\NoProcess` argument is used if the instruction should be executed without the welding process active.

Offset

Data type: `multidata`

The `Offset` data contains the offset information for the path.

Adapt

Data type: `adaptdata`

Data structure with parameters for calculating the initial settings.

[SpeedGain]

Data type: `num`

Minimum value 0.5, maximum value 1.5.

[| AdaptToMinMax]

Data type: `switch`

Continues on next page

[\StartInd]

Data type: `num`

The optional argument `\StartInd` is used if the path should be replayed from a specific index instead of from the beginning of the stored path.

> **Note**
>
> First index in a path is always 1.

[\EndInd]

Data type: `num`

The optional argument `\EndInd` is used if the path should be replayed to a specific index not the end of the stored path. If a negative value is entered, the end index will be used as reference, for example -2 is index 2 from the end.

> **Note**
>
> First index in a path is always 1.

Speed

Data type: `speeddata`

The speed of the TCP is controlled by the argument `Speed` during the movement towards the start of the replay sequence. The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. `Speed` data also describes the speed of the tool's reorientation and the speed of any uncoordinated additional axes.

Seam

Data type: `seamdata`

`Seam` data describes the start and end phases of a welding process. The argument `Seam` is included in all arc welding instructions so that, regardless of the position of the robot when the process is interrupted, a proper weld end and restart is achieved.

Weld

Data type: `welddata`

`Weld` data describes the weld phase of the welding process.

Weave

Data type: `weavedata`

`Weave` data describes the weaving that is to take place during the heat and weld phases. Welding without weaving is obtained by specifying, for example, the weave data `noweave` (no weaving if the `weave_shape` component value is zero).

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 case of a fly-by point, a corner

Continues on next page

path is generated past that position. In case of a stop point (fine), the movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld (even in the case of a flying start) and at a controlled welds end position. Fly-by points, such as z10, should be used for all other weld positions. It is recommended to use a z5 data for the replay instruction.

Tool

Data type: `tooldata`

`Tool` defines the tool used in the 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) to which the instruction's robot position is referenced. When this argument is not used, the robot position is referenced to the world coordinate system. It must be specified if a stationary TCP or coordinated additional axes are used.

[\Track]

Data type: `trackdata`

`Track` data describes the parameters used for tracking.

[\SeamName]

Data type: `string`

`SeamName` defines the name used in error logs if an error occurs during the welding sequence. `\SeamName` can only be used in the first instruction of a sequence of weld instructions, that is, together with the `\Start` argument. The `SeamName` in the `ArcRepL` instruction specifies which path to replay, so the `SeamName` must be the same as the `SeamName` used to record the path.

[\ServRoutine]

Data type: `string`

A service routine can be specified and used together with the `Escape` selection in the *Weld Error Recovery* menu. If `Escape` is selected, the robot will reverse back along the recorded path to the first recorded point on the path with the speed and offset specified by the settings in Arc Error Handler in the process configuration.

[\TLoad]

Data type: `loaddata`

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

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Continues on next page

Syntax

```
ArcRepL
  [ '\' Start ',' ] < expression (IN) of switch >
  [ '\' End ',' ] < expression (IN) of switch >
  [ '\' NoProcess ',' ] < expression (IN) of switch >'
  [ Offset ':=' ] < expression (IN) of multidata >','
  [ Adapt ':='] < Expression (IN) of adaptdata >','
  [ '\' SpeedGain':='] < Expression (IN) of num>','
  [ '| AdaptToMinMax: '='] < switch>','
  [ '\' StartInd ':=' < expression (IN) of num > ';'
  [ '\' EndInd ':=' < expression (IN) of num > ';'
  [ Speed ':=' ] < expression (IN) of speeddata >','
  [ Seam ':=' ] < persistent (PERS) of seamdata > ','
  [ Weld ':=' ] < persistent (PERS) of welddata > ','
  [ Weave ':=' ] < persistent (PERS) of weavedata > ','
  [ Zone ':=' ] < expression (IN) of zonedata >','
  [ Tool ':=' ] < persistent (PERS) of tooldata >','
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ';'
  [ '\' Track ':=' ] < persistent (PERS) of trackdata >','
  [ '\' SeamName ':=' < expression (IN) of string > ]
  [ '\' ServRoutine ':=' < expression (IN) of string > ]
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

ArcRepL on page 63

6.3 Data types

6.3.1 trackdata

Description

`trackdata` is used to control path corrections during the weld phase. `trackdata` used in a given instruction along a path affects the path correction until the specified position is reached. By using instructions with different `trackdata` it is possible to achieve optimum position control along an entire seam. If the optional `trackdata` argument is left out, tracking is suspended.

The process path should be programmed accurately with respect to the nominal geometry and orientation of the work piece. The tracking function activated by the optional `trackdata` argument will compensate for deviations from the nominal path.

Using `trackdata` works best for welding applications with long strait seams with speed lower than 20 mm/s and orientation errors smaller than 10 degrees.

Components

track_system

Data type: `num`

This parameter defines which tracking system that is used, Optical or Weldguide. It is also used for data masking of the `trackdata`. The `track_device` is configured in the equipment configuration parameters.

store_path

Data type: `bool`

Parameter used when the path should be stored.

max_corr

Data type: `num`

`max_corr` defines the maximum path correction allowed. If the TCP is offset more than `max_corr` by path corrections a track error is reported and program execution is stopped.

track_type

Data type: `num`

`track_type` defines the type of tracking. The optional argument `\Track` must be added to each weld instruction in the program.

Value	Description
0	Centerline tracking
1	Adaptive tracking
2	Single side tracking (right side)
3	Single side tracking (left side)
4	Not used for Weldguide

Continues on next page

Value	Description
5	Height only tracking. Constant stick-out length is kept. Current is specified, voltage is floating.
10 to 13	Same as 0 to 3 but both voltage and current are specified
20	Inverted centerline
30	Inverted centerline, both voltage and current are specified **Note** For inverted centerline tracking, make sure to use v-shaped weaving and negative height. See `weavedata` and `trackdata` in *Application manual - Continuous Application Platform.*

gain_y

Data type: `num`

The `gain_y` parameter defines the size of the correction sent to the robot. The higher the number the faster the system corrects.

Allowed values: 1-100. Initial starting values depends on weave size. Start with 30 for most weave widths and 5 for very small weave widths.

gain_z

Data type: `num`

The `gain_z` parameter defines the size of the correction sent to the robot. The higher the number the faster the system corrects.

Allowed values: 1-100. Initial starting values depends on weave size. Start with 30 for most weave widths and 5 for very small weave widths.

weld_penetration

Data type: `num`

Defines how hard the system should bite to the sidewall of the parent material in percentage of penetration. Although always present, Weldguide uses this parameter only during adaptive, right, and left side tracking. Normal value: 1-10.

track_bias

Data type: `num`

`track_bias` is used to move the TCP in the seam y direction to bias one side of the joint or the other.

Allowed value: -30 to +30, where +30 is the highest amount of bias possible in the plus Y direction of the seam coordinates. Only used in centerline tracking.

min_weave

Data type: `num`

The minimum weave width setting that system is allowed to change during adaptive tracking. Must be larger than 2 mm.

max_weave

Data type: `num`

Continues on next page

The maximum weave width setting that system is allowed to change during adaptive tracking.

max_speed

Data type: `num`

The minimum travel speed setting that system is allowed to change during adaptive tracking.

min_speed

Data type: `num`

The maximum travel speed setting that system is allowed to change during adaptive tracking. Must be larger than 2 mm/s.

6.3.2 multidata

Description

`multidata` is used to define the path offset for a replayed path. The data type contains the information on how the robot should position the tool relative to a stored path.

Components

Direction

Data type: `num`

Direction of travel for the replayed path. Can be set to 1 or -1. 1 defines that the path will be replayed in the same direction as it was stored. -1 will replay the path in the opposite direction.

ApproachDistance

Data type: `num`

Offset in tool coordinate system -z-axis in mm for the first stored point. An approach point is created here.

DepartDistance

Data type: `num`

Offset in tool coordinate system -z-axis in mm for the last stored point. A depart point is created here.

StartOffset

Data type: `num`

Offset in mm for the start of the path relative to the first or last stored point depending on direction. A negative number shortens the weld path.

EndOffset

Data type: `num`

Offset in mm for the end of the path relative to the first or last stored point depending on direction. A negative number shortens the weld path.

SeamOffs_y

Data type: `num`

Fixed path offset in millimeters for the seam y-direction.

SeamOffs_z

Data type: `num`

Fixed path offset in millimeters for the seam z-direction.

SeamRot_x

Data type: num

Torch rotation in degrees around seam x-axis. Rotation is relative to the stored path.

Continues on next page

SeamRot_y

Data type: `num`

Torch rotation in degrees around seam y-axis. Rotation is relative to the stored path.

6.3.3 adaptdata

Description

`adaptdata` is used to define the adaptive parameters used for adaptive welding. This data type is used with the `ArcAdaptX` and `ArcCalcX` instructions.

The `ArcAdaptX` instructions are used for Weldguide based adaptive tracking. Weave width and weld speed are updated based on data from the tracking system. The `ArcAdaptX` instructions must be used for all instructions in the weld.

The `ArcCalcX` instructions are used for adaptive tracking based on the measured groove width (often with *SmarTac*). Weave width and weld speed are updated based on measured groove width.

Components

NominalWidth

Data type: `num`

Nominal width of the groove. Normally the initial result of the groove search.

WeaveAdapt

Data type: `num`

Weave change factor, normally set to 1.

AdaptOffs_y

Data type: `num`

Not used when using `ArcAdaptX` instructions.

AdaptOffs_z

Data type: `num`

Not used when using `ArcAdaptX` instructions.

min_weave

Data type: `num`

The minimum weave width. Only used with the `ArcCalcX` instructions.

max_weave

Data type: `num`

The maximum weave width. Only used with the `ArcCalcX` instructions.

min_speed

Data type: `num`

The minimum speed. Only used with the `ArcCalcX` instructions.

max_speed

Data type: `num`

The maximum speed. Only used with the `ArcCalcX` instructions.

7 System parameters

7.1 Topic Process

Introduction

The system parameters for the Weldguide sensor can be modified using the FlexPendant or RobotStudio.

The parameters belong to the type *WG Sensor Properties* in the topic *Process*. The instance is called *WG_T_ROB1* (or *WG_T_ROB2* in a multi robot setup). For more information about system parameters, see *Technical reference manual - System parameters*.

Type WG Sensor Properties

Parameter	Data type	Description
Name	`string`	The name of the Weldguide sensor
Device	`string`	The device name used for the tracker. *Device* must match the transmission protocol name configured in the topic *Communication* (SIO.cfg."swg:") for Weldguide.
Max Incremental Correction	`num`	Maximum correction allowed per weave. Default value: 0.5 mm
Adapt Start Delay	`num`	Number of weaves before adaptive tracking starts. Default value: 10 (5 weave cycles)
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. Default value: 95%
LeftWeaveSyncDO	`string`	Digital output signal for left sync pulse.
RightWeaveSyncDO	`string`	Digital output signal for right sync pulse.
Logfile	`string`	Filename for the log file (normally not used).
Voltage Offset	`num`	Voltage offset added to the measured real time value. If the value shown in the FlexPendant application differs from the value measured with an external current clamp, then an offset can be added here.
Current Offset	`num`	Current offset added to the measured real time value. If the value shown in the FlexPendant application differs from the value measured with an external current clamp, then an offset can be added here.
Voltage Analog Meter Min	`num`	Minimum value for the voltage analog meter used for the Weldguide FlexPendant application. Default value: 15 (Volt)

Continues on next page

Parameter	Data type	Description
Voltage Analog Meter Max	`num`	Maximum value for the voltage analog meter used for the Weldguide FlexPendant application. Default value: 35 (Volt)
Current Analog Meter Min	`num`	Minimum value for the current analog meter used for the Weldguide FlexPendant application. Default value: 100 (Ampere)
Current Analog Meter Max	`num`	Maximum value for the current analog meter used for the Weldguide FlexPendant application. Default value: 500 (Ampere)
Disable startup check	`bool`	This flag disables the start-up check (sensor connected, dwell bits, alive ping which is sent every 10 seconds). Only necessary if MultiPass with an optical sensor is used or if MultiPass without tracking is needed. Default value: FALSE
Sensor check interval	`num`	Sets the interval for the heartbeat check for the sensor. The system will verify the status of the Weldguide sensor periodically. The verification will detect loss of communication or if the current sensor has been disconnected. The default verification frequency is 30 s.
Enable WeldData voltage field	`bool`	This flag is used to unmask the voltage field for the weld data editor. In situations when track type 10-13 is to be used but the voltage for the welding power supply is not specified in the weld data, the voltage field can be enabled by this parameter.
Sensor 1 Calibrated	`bool`	Internal parameter that is set by the system when the first current/voltage sensor has been calibrated. Set this parameter to *False*, and the parameters *Sensor 1 OffsetA* and *Sensor 1 OffsetB* to zero (0) if the current/voltage sensors need to be recalibrated and then restart the controller. For more information, see *Calibrating the sensor on page 29*.
Sensor 1 OffsetA	`num`	Internal parameter that is set by the system during voltage sensor calibration.
Sensor 1 OffsetB	`num`	Internal parameter that is set by the system during current sensor calibration.

Continues on next page

Parameter	Data type	Description
Sensor 2 Calibrated	`bool`	Internal parameter that is set by the system when the first current/voltage sensor has been calibrated. Set this parameter to *False*, and the parameters *Sensor 2 OffsetA* and *Sensor 2 OffsetB* to zero (0) if the current/voltage sensors need to be recalibrated and then restart the controller. Sensor 2 is only used for systems with twin wire welding. For more information, see *Calibrating the sensor on page 29*.
Sensor 2 OffsetA	`num`	Internal parameter that is set by the system during voltage sensor calibration.
Sensor 2 OffsetB	`num`	Internal parameter that is set by the system during current sensor calibration.

This page is intentionally left blank

8 Trouble shooting

Trouble shooting check list

	Action	Note
1	Is the weld return separated from system ground?	Weld return should be connected directly from the positioner/work-piece to power source.
2	Are there bakelite insulators mounted under the welding robot?	
3	Are the power cables routed close together with signal cables?	Try to separate them as much as possible.
4	Are there cables wound up like coils?	This could cause an electrical field disturbing the signals
5	Is the welding gun isolated from the robot?	The welding gun should be isolated from the flange on the robot with an insulating plate.
6	Is the wire feeder isolated from the robot?	The wire feeder should be isolated from the robot with rubber bush-ings.
7	Are there any current collectors mounted?	The current collectors should be cleaned and greased.
8	Is the shielding of the cables correct?	Shielding slipped out of position or bad connection. Also paint in screw holes could cause bad connection to ground. Check SMB-box.
9	Are the weld connectors warm or hot?	Could indicate bad contact, can make the current taking another route. Check temperature in PIB and all connectors involved (espe-cially the power source connector).
10	Are there any newly exchanged parts in the sta-tion?	Has the station worked ok before?
11	Has the Robot controller correct ground connec-tion.	Should be grounded to incoming PE.
12	Has the power source the correct ground connec-tion.	Should be grounded to incoming PE.
13	Are the power sources (inverters) placed close to cables or the controller.	Could be tested by deactivating the external devices and test.
14	Measure all cables for continuity.	
15	Is both ends of the voltage cable correctly connec-ted?	See *Example of voltage cable con-nections on page 30*.
16	Is the sensor correctly connected?	See *Installing the sensor on page 27*.
17	Are the dwell bits correctly connected and con-figured?	See *Mounting and connecting the board on page 25*.
18	Check that the communication with the Weldguide board is set up correctly.	See *Verifying communication on page 37*.
19	Check that the sensor is calibrated.	See *Calibrating the sensor on page 29*.

This page is intentionally left blank

9 Spare parts

Spare parts for Weldguide IV

Article number	Spare part	Note
3HAC052650-001	WG IV Board-Basic	
3HAC052823-001	WG IV Board-Advanced	
3HAC052824-001	WG IV solidcore sensor kit	Current Sensor.1000A AMP Sensor cable solid core WG IV Volt sensor cable
3HAC052869-001	WGIV splitcore sensor kit	Current sensor open core AMP sensor cable split core WG IV Volt sensor cable
3HAC052649-001	WG IV Volt sensor cable	
3HAC055476-001	Weldguide Voltage Adaptor	
3HAC055475-001	Wiring Set Internal WG IV	Ethernet Cable if switch WG IV Ethernet cable Bulkhead cable sensors WG IV Power cable 24 VDC WG IV Ext I/O harness

This page is intentionally left blank

Index

Contact us

ABB AB
Discrete Automation and Motion
Robotics
S-721 68 VÄSTERÅS, Sweden
Telephone +46 (0) 21 344 400

ABB AS, Robotics
Discrete Automation and Motion
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 51489000

ABB Engineering (Shanghai) Ltd.
No. 4528 Kangxin Hingway
PuDong District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

www.abb.com/robotics

3HAC054886-001, Rev A, en