ABB

ROBOTICS

Application manual
Arc and Arc Sensor

Application manual

Arc and Arc Sensor

RobotWare 6.15.06

Document ID: 3HAC050988-001

Revision: K

Table of contents

Table of contents

Overview of this manual

About this manual

This manual contains instructions for installing and programming a RobotWare *Arc* and Arc sensor system.

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

Prerequisites

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.

> **ℹ Note**
>
> Before any work on or with the robot is performed, the safety information in the product manual for the controller and manipulator must be read.

References

References	Document ID
Operating manual - IRC5 with FlexPendant	3HAC050941-001
Operating manual - RobotStudio	3HAC032104-001
Operating manual - Seam tracking with Weldguide III and MultiPass	3HEA802921-001
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
Application manual - MultiMove	3HAC050961-001

Revisions

Revision	Description
-	Published with RobotWare 6.0.
A	Published with RobotWare 6.01.
B	Published with RobotWare 6.02. • Updated options in section *Power source type on page 9*. • Updated the examples for the RAPID instructions and restructured the section, see *Instructions on page 107*.

Continued

Revision	Description
C	Published with RobotWare 6.04. • The manual is partly restructured. • Added information about flying start. • Minor corrections.
D	Published with RobotWare 6.05. • Added information about *Pre Process Tracking*. • Minor corrections.
E	Published with RobotWare 6.07. • Protocol LTPROTOBUF added to sensor interface. • Added track mode 13, 14 and 15 to section *trackdata - Seam tracking data on page 184*. • Minor corrections.

F	Published with RobotWare 6.09.
	• Updated information for `ArcMoveExtJ`.
	• Limitation information updated for instructions:
	    `ArcC, ArcC1, ArcC2`
	    `ArcCEnd, ArcC1End, ArcC2End`
	    `ArcCStart, ArcC1Start, ArcC2Start`
	    `ArcL, ArcL1, ArcL2`
	    `ArcLEnd, ArcL1End, ArcL2End`
	    `ArcLStart, ArcL1Start, ArcL2Start`
	    `ArcMoveExtJ`
	    `ArcRefresh`
	• Updated information for WeldRepair with FlexPositioner.
	• Added information about Add-Ins, see *Power source type on page 9*.
	• Added *Stop Mode*, see *The type Arc System Properties on page 206* and *The type Arc Robot Properties on page 210*.

G	Published with RobotWare 6.11. • Added limitation for `seamdata`.
H	Published with RobotWare 6.13. • Added *WelderReady* supervision for standard I/O welder.
J	Published with RobotWare 6.14. • The system parameter *SupervInhib* is removed from the type *The type Arc Equipment Digital Inputs on page 221*.
K	Published with RobotWare 6.15.06. • Minor corrections.

1 Installation and setup

Installation options

The installation of RobotWare Arc can be customized to fit various application demands, such as a different power source types and MultiMove support.

The following options can be selected in RobotStudio when creating the system, and then customized according to application demands:

- Power source type
- 651-1 Additional Arc Systems
- 660-1 Optical Tracking Arc

💡 **Tip**

How to create systems in RobotStudio is described in *Operating manual - RobotStudio*.

Power source type

The following power sources can be selected from Installation Manager in RobotStudio during installation.

Power source interface	Description
Standard I/O Welder	Standard I/O Welder
Simulated Welder	Simulated Welder
Fronius • Integrated version • DeviceNet • EtherNet/IP	Fronius Welder • Integrated version • DeviceNet configuration • EtherNet/IP configuration
ESAB AristoMig integrated	ESAB AristoMig Welder
Lincoln ArcLink XT	Lincoln ArcLink XT Welder
SKS SynchroWeld • DeviceNet • ProfiBus • ProfiNet	SKS SynchroWeld Welder • DeviceNet configuration • ProfiBus configuration • ProfiNet configuration
RW Add-in loaded Welder	The Add-ins are found here: RobotStudio/Add-ins/Gallery/Common tags: RobotWare-Addin

If no power source is selected, the *Standard I/O Welder* will be loaded.

If *RW Add-in loaded Welder* is selected, only the basic functionality of RobotWare Arc will be loaded without any power source specific functionality. The welder must be loaded from a RobotWare Add-in.

RW Add-in loaded Welder

The following Add-ins are available:

- Fronius TPSi
- Fronius TPSi Seamtracking

Continues on next page

- Miller EIP Welder (Miller Ethernet IP welder)
- Lincoln ArcLink/XT

651-1 Additional Arc Systems

The option *Additional Arc Systems* includes support for additional Arc Systems.

The following additional arc systems can be selected.

Option	Description
651-1 Additional Arc Systems	Additional arc systems

If *Additional Arc Systems* is selected, the instructions and data types for additional systems are installed, where X can be either L for linear motion or C for circular motion.

- `ArcX1Start`
- `ArcX1`
- `ArcX1End`
- `seamdata1`
- `welddata1`
- `trackdata1`
- `ArcX2Start`
- `ArcX2`
- `ArcX2End`
- `seamdata2`
- `welddata2`
- `trackdata2`

660-1 Optical Tracking Arc

The option *Optical Tracking Arc* includes advanced laser tracking features together with the ServoRobot M-Spot-90 and the Scout Sensors.

The following optical tracking options can be selected.

Option	Description
660-1 Optical Tracking Arc	Optical tracking with RobotWare Arc.

2 RobotWare - Arc Adaptive process control

2.1 Adaptive Process Control

Description

The options *Optical tracking Arc [660-1]* and *WeldGuide [815-1]* are for arc welding applications where welding data or path must be dynamically changed during the welding to adapt to changes in geometry or material. In addition to the basic RobotWare Arc package, the options also include functions for *Adaptive Process Control* and *Statistical Process Control*.

Adaptive Process Control provides the following functionality:

- **Seam tracking:** This is used when sensor signals are used, while welding a seam, to correct the path of the robot, thus tracking the real seam. This is useful, for example, if parts are not placed in exactly the same position each time or if the seam geometry can vary. See *Seam tracking on page 12*.

- **Sensor controlled tuning:** This is used when sensor signals are used to update the process data used while welding. This is useful, for example, if the seam features vary while the robot is welding. See *Sensor controlled tuning on page 18*.

- **Program controlled tuning:** This is used when welding data is changed automatically and is related to the path or position. See *Program controlled tuning on page 19*.

2 RobotWare - Arc Adaptive process control

2.2 Seam tracking

2.2.1 Seam tracking systems

Description

RobotWare Arc sensor is prepared to be used in combination with two specific seam tracking sensor systems, which are:

- WeldGuide Tracker systems
- Laser Tracker systems

WeldGuide Tracker systems

These systems are based on measuring the current and voltage of the arc, while performing weaving around the expected path. Variations in current and voltage are measured and used to calculate current offset from the wanted path. Then path correction values are sent from the sensor system to the robot controller, where the corrections will be added to the ordered position values.

WeldGuide Tracker systems are connected to the controller via a serial link.

Laser Tracker systems

These systems are based on using a separate sensor device mounted on the robot arm. The sensor is based on a laser emitter sending a light ray on the part. The reflected ray is received by a photo sensitive array and by triangulation the distance from the sensor to the reflecting surface can be calculated.

Laser Tracker systems are connected to the controller via a serial link or Ethernet..

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

2.2.2 Seam tracking in different instructions

Seam tracking in arc welding instructions

The `ArcX` instructions can be used for seam tracking in the following ways.

If the system is configured for the use of a WeldGuide Tracker or Laser Tracker, then the optional argument `\Track` shall be used to control the tracking function. With this argument it is possible to specify the track data to be used for the specific `ArcX` instruction.

The communication between WeldGuide Tracker system or Laser Tracker system and the controller is via a serial link or via Ethernet (TCP/IP) using a specific link protocol (RTP1) and a specific application protocol (LTAPP / LTPROTOBUF). The option *Optical Tracking Arc* or *WeldGuide* is needed.

If the system is not configured for a WeldGuide Tracker or Laser Tracker seam tracking, that is, none of these parameters are set, then the `ArcX` instructions will work for path corrections using `CorrXXX` instructions. Then the optional argument `\Corr` must be used in these instructions instead of `\Track`. The option *Path Offset* is needed.

Seam tracking in other movement instructions

For ordinary movement instructions like `MoveL` or `MoveC` also path corrections can be done. Then the optional argument `\Corr` must be used in these movement instructions. The path corrections will then be programmed using `CorrXXX` instructions, see below. These instructions are only available if the option *Path Offset* or the option *RobotWare-Arc sensor* are installed.

If the correction values are fetched from an external sensor, then the communication between sensor and robot controller can be via a serial link. Also in this case *Sensor Interface* or *RobotWare-Arc sensor* options should be used, which will include instructions for the serial communication using a specific link protocol (RTP1) and a specific application protocol (LTAPP / LTPROTOBUF) (see short description below and *Application manual - Controller software IRC5*, section *Sensor Interface*).

Path correction instructions

These instructions, included in the option *Path offset*, describe the path correction. The following instructions and data types are available:

- CorrClear
- CorrCon
- CorrDiscon
- CorrRead
- CorrWrite
- Data type: corrdescr

These instructions and the data type will make it possible to add certain offsets to a programmed path, while the robot is moving. The offsets to add can be values given from a sensor connected to the system via e.g. serial link or via analog input.

Continues on next page

Sensor interface

This is the same as the separate option *Sensor Interface*. The option, included in *Optical Tracking* or *WeldGuide*, will make serial communication possible with an external sensor or other unit. The communication will use the link protocol RTP1. With this function it is possible to read data from or write data to the sensor using the instructions listed below. Thus it will be possible to use sensor data for path corrections or for process tuning. The following instructions will be included for the data communication:

- IVarValue
- ReadBlock
- ReadVar
- WriteBlock
- WriteVar

> **Note**
>
> Path corrections for `ArcX` instructions can be done in two ways as indicated above.
>
> - In system which are not configured for *WeldGuide* or *Laser Tracker*, the instructions `Corrxxx` (see above) must be used to write the correction values to a correction generator
> - In system configured for serial *WeldGuide* or *Laser Tracker*, path corrections will be automatically active if the `\Track` argument is used, where the track data to be used is included. This requires that the application protocol LTAPP or LTPROTOBUF is used for the communication with the sensor.

2.2.3 Optical tracking

Prerequisites

If the system is configured for the use of an optical tracker, then the optional argument `\Track` shall be used to control the tracking function. With this argument it is possible to specify the track data to be used for the specific `ArcX` instruction. If the `\Track` argument is not included, no seam tracking will be active.

As an optical tracker is mounted in front of the weld gun and measures the actual seam position in advance, the first part of the seam has no tracking data available. Therefore the distance between the first sensor measurement position and the start of the seam will not be tracked. To avoid this problem you can use the optional argument `\PreProcessTracking`, which activates *Pre Process Tracking*

For more information see *Operating manual - Tracking and searching with optical sensors*.

```
MoveL p10,v1000,fine,tWeldGun;
ArcLStart p20,v1000,sm1,wdHD\Weave:=wv4,fine,tWeldGun\Track:=tr1;
ArcLEnd p30,v1000,sm1,wdHD\Weave:=wv4,fine,tWeldGun\Track:=tr1;
```

The laser sensor needs to be calibrated to have optimal performance. That is done via the calibration programs delivered with the *Sensor* option. The result from that calibration ends up in a RAPID variable of type `pose`. The name of this variable should be specified in the configuration parameters for the optical tracker.

2.2.4 WeldGuide

Tracking methods

A WeldGuide tracking system uses the arc as a sensor to adapt 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.

Adaptive welding. A further enhancement is to use the same data to adapt the robot weave width and travel speed in order to fill a groove that can vary in size.

More information about WeldGuide tracking can be found in *Operating manual - Seam tracking with Weldguide IV and MultiPass*.

Path correction instructions

These instructions, included in the option *Path offset*, describe the path correction. The following instructions and data types are available:

- CorrClear
- CorrCon
- CorrDiscon
- CorrRead
- CorrWrite
- **Data type:** corrdescr

These instructions and the data type will make it possible to add certain offsets to a programmed path, while the robot is moving. The offsets to add can be values given from a sensor connected to the system via e.g. serial link or via analog input.

Sensor interface

This is the same as the separate option *Sensor Interface*. The option, included in *RobotWare-Arc sensor*, will make serial communication possible with an external sensor or other unit. The communication will use the link protocol RTP1. With this function it is possible to read data from or write data to the sensor using the instructions listed below. Thus it will be possible to use sensor data for path corrections or for process tuning.

The following instructions will be included for the data communication:

- IVarValue
- ReadBlock
- ReadVar
- WriteBlock
- WriteVar

Continues on next page

> **ⓘ Note**
>
> Path corrections for `ArcL`/`ArcC` instructions can be done in two ways as indicated above.
>
> - In system which are not configured for WeldGuide or Laser Tracker, the instructions `Corrxxx` (see above) must be used to write the correction values to a correction generator.
> - In system configured for WeldGuide or Laser Tracker, path corrections will be automatically active if the `\Track` argument is used, where the track data to be used is included. This requires that the application protocol LTAPP or LTPROTOBUF is used for the communication with the sensor.

Related information

Information	Described in
Installation parameters for welding equipment and functions	*System parameters on page 203*
`trackdata` - seam tracking data	*trackdata - Seam tracking data on page 184*
Installation and setup	*660-1 Optical Tracking Arc on page 10*
Sensor Interface	*Application manual - Controller software IRC5*
Arc welding instructions	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*
	ArcC, ArcC1, ArcC2 - Arc welding with circular motion on page 107

2.3 Sensor controlled tuning

Description

Sensor controlled tuning provides a powerful tool for changing/tuning a process during the execution of a weld due to the input signals from a sensor.

Example of application are change process data like voltage, wire feed, speed based on current sensor values for seam volume or gap detected by a sensor.

The function is generally used in connection with trap routines and interrupts where the instruction `ArcRefresh` can be used to update the weld data. The communication with the external sensor, which provides the feedback data, can for instance be done using the option *Sensor Interface*, which is included in *RobotWare-Arc sensor*.

2.4 Program controlled tuning

Description

Program controlled tuning means that the weld data can be changed during welding related to specific positions on the path or other known geometry changes.

Example of application are:

- Change process data with reference to a time or distance before or after a defined position
- Change the wire feed speed with reference to the volume of the seam
- Set up a heat pulse variation along a seam
- Set up high penetration on one side of a seam and low penetration on the other side
- Initiate a ramp-down towards the end of a weld

The function is generally used in connection with trap routines and interrupts where the instruction `ArcRefresh` can be used to update the `welddata`.

This page is intentionally left blank

3 Programming

3.1 Programming for arc welding

Prerequisites

Before creating an arc welding program, the arc welding system or systems (see *Defining arc welding systems on page 204*) and additional axes, if any, must be configured. This configuration is described in *System parameters on page 203*.

Program structure

When there are several seams to be welded on an object, the welding sequence may be of critical importance for the quality of the object. The risk of deformation due to thermal stress can be reduced by choosing a suitable seam welding sequence. It is often best to make a specific routine, object routine, for this with all the seams specified in the correct order. When the object is placed in a positioner, its orientation can also be specified in the object routine. The object routine can call a welding routine for each seam to be welded.

Arc welding instructions

An arc welding instruction contains the same information as a positioning instruction (e.g. `MoveL`), plus all information about the welding process, which is given through the arguments `seamdata`, `welddata`, and `weavedata`.

Data for the start and end sequences of the weld

Data for the ongoing weld

Data for weaving

```
ArcL p1, v100, sm1, wd1\Weave:=wv1, fine, torch;
```

Arguments as in the instruction MoveL

L = Linear
C = Circular
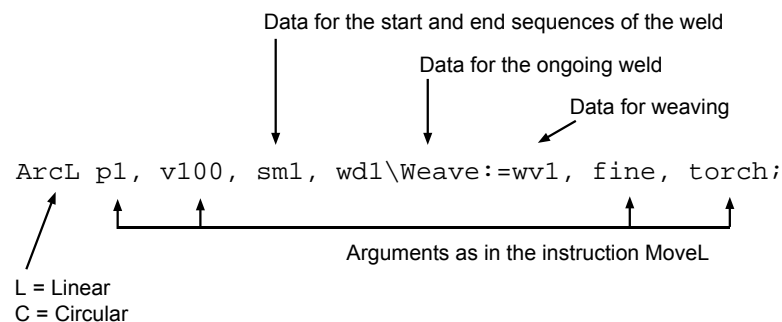
xx1200000641

The speed argument, `v100`, in the instruction is only valid during step-wise execution (forward or backward) and the welding process will in this case automatically be inhibited.

During normal execution, the process speed in different phases of the process is included as components of seam and weld data.

For more information on programming arc welding instructions, see *Programming arc welding instructions on page 23*.

Defining arc welding data

Before starting to program arc welding instructions, arc welding data must be defined. This data is divided into three types:

seamdata	Describes how the seam is to be started and ended.

Continues on next page

3 Programming

welddata	Describes the actual welding phase.
weavedata	Describes how any weaving is to be carried out.

Number and type of the data components depend on the configuration of the robot. Normally, data is stored as a part of the program. However, when data is to remain in memory regardless of which program is loaded, it is stored in a system module.

1. Open the **Program Data** window from the ABB menu.
2.
3. Select the type `seamdata`, `welddata`, **or** `weavedata`.
4. Tap **New**.

   The data properties are displayed.

en1200000642

5. A default name is suggested. If the name needs to be changed, tap the **Name** button and specify a new name.
6. If the data needs to be saved in another module, tap the **Module** drop-down menu and select the desired module.
7. Tap **OK**.

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

8 The data will appear in the list with `welddata` variables. To change the declaration, tap the data.

en1200000643

9 Select a component in the data and specify the desired value. More information on the individual components can be found in *seamdata - Seam data on page 178*, *welddata - Weld data on page 197*, and *weavedata - Weave data on page 190*.

> ℹ️ **Note**
>
> Some of the components of `welddata` depend on the configuration of the robot. If a given feature is omitted, the corresponding component is not present in the `welddata`. See *System parameters on page 203*.

> 💡 **Tip**
>
> In some cases it is easier to create new data by copying and modifying existing data.

Programming arc welding instructions

1 Jog the robot to the desired position.

2 Open the instruction pick list in the **Program Editor**, select picklist **Motion & Process**.

3 Select the instruction `ArcL` or `ArcC`.

The instruction will be added to the program. The arguments are set in relation to the last arc welding instruction that was programmed.

The instruction is now ready for use.

3 Programming

If an argument needs to be changed, the data can be replaced by another.

1 Select the argument you wish to change (`seam1` in this example).

2 When the argument is selected, tap **Change Selected** on the **Edit** menu. The window used to change instruction arguments appears. The selected argument is highlighted see figure below. The lower part of the window displays all available `seamdata` that can be selected.
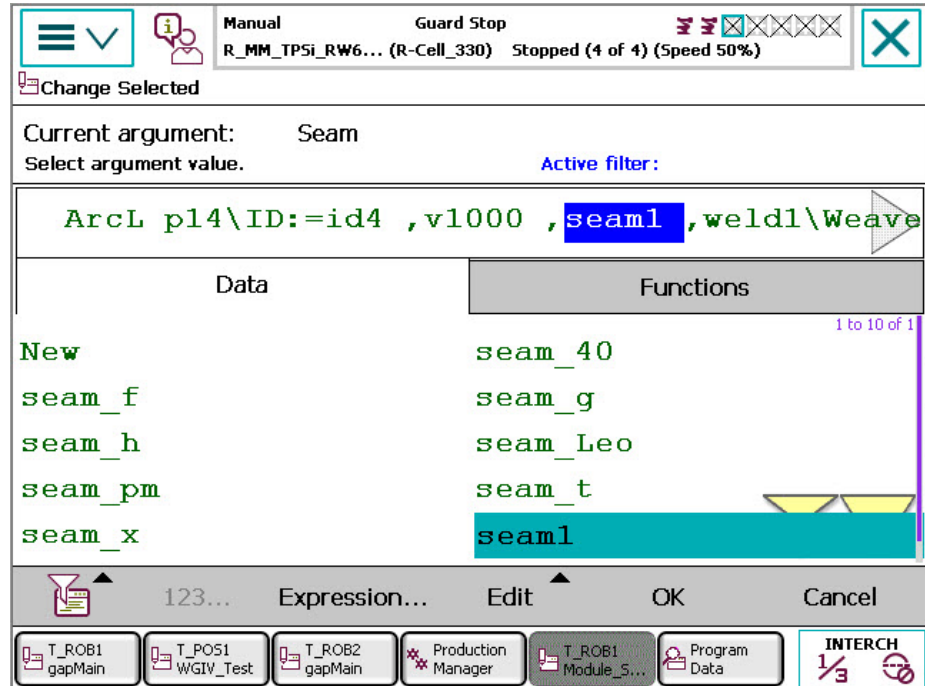
en1200000644

3 Select the desired `seamdata`.

4 Change another argument by tapping on the argument in the instruction.

5 Repeat this for all arguments that needs to be changed.

6 Tap **OK** to confirm the changes.

Example of an arc welding instruction

The seam illustrated in the figure below is to be welded. The seam line is represented by the thick line in the figure.

Preparations for welding (such as gas preflowing) are carried out between points `p10` and `p20`, on the way to the starting-point, `p20`. The weld is terminated at point `p80`.

The `welddata`, **wd1**, applies until position p50 is reached, where a transition to **wd2** takes place.

MoveJ

ArcLStart

MoveJ

p10

p90

p20

p30

p40

ArcC

ArcLEnd

wd1

ArcL

p80

wd2

p50

p60

Delay distance

p70

——— Movement without welding

XXXXX Start preparations

━━━ Movement with welding
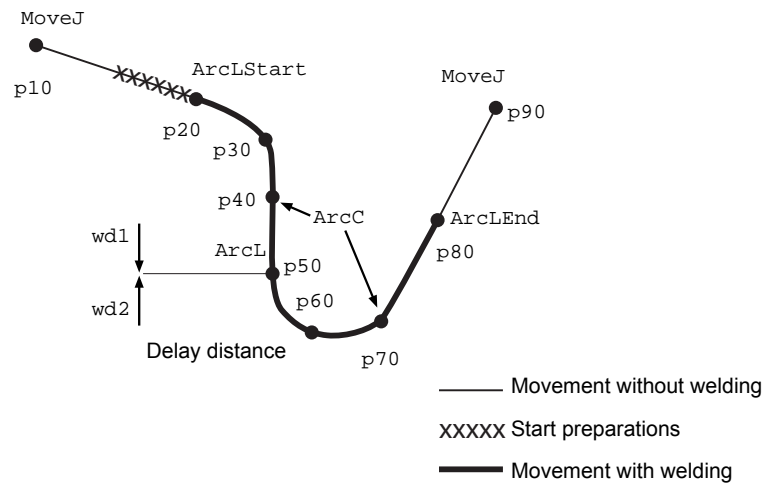
xx1200000645

The programming sequence for this seam could be written as follows:

```
MoveJ p10,v100,z10,torch;
ArcLStart p20,v100,sm1,wd1,wv1,fine,torch;
ArcC p30, p40, v100, sm1, wd1, wv1, z10, torch;
ArcL p50,v100,sm1,wd1,wv1,z10,torch;
ArcC p60,p70,v100,sm1,wd2,wv1,z10,torch;
ArcLEnd p80,v100,sm1,wd2,wv1,fine,torch;
MoveJ p90,v100,z10,torch;
```

If the seam is to be coordinated with an additional axis, an argument of the type work object has to be included in all arc welding instructions except for the start instruction. For more information, see *ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*, and *ArcC, ArcC1, ArcC2 - Arc welding with circular motion on page 107*.

3.2 Functions for arc welding when program execution has been stopped

Manual mode

Arc welding functions (program execution has been stopped) in manual mode:

- Weld data tuning
- Weave data tuning
- Communicate with seam tracker sensor
- Process blocking
- Manual wire feed
- Manual gas on/off
- Select arc welding system
- Changing tuning increments

> **ℹ Note**
>
> If a window is open in Manual mode and the functionality is disabled in Auto mode, switching from Manual to Auto mode will close the window.

Auto mode

Arc welding functions (program execution has been stopped) in Auto mode:

- Manual wire feed
- Manual gas on/off
- Select arc welding system
- Changing tuning increments

> **ℹ Note**
>
> If a window is open in Manual mode and the functionality is disabled in Auto mode, switching from Manual to Auto mode will close the window.

Continues on next page

　　　　Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

RobotWare Arc on FlexPendant

To start RobotWare Arc, tap the ABB menu and then tap RobotWare Arc. When RobotWare Arc desktop is loaded, all arc welding functions can be accessed.
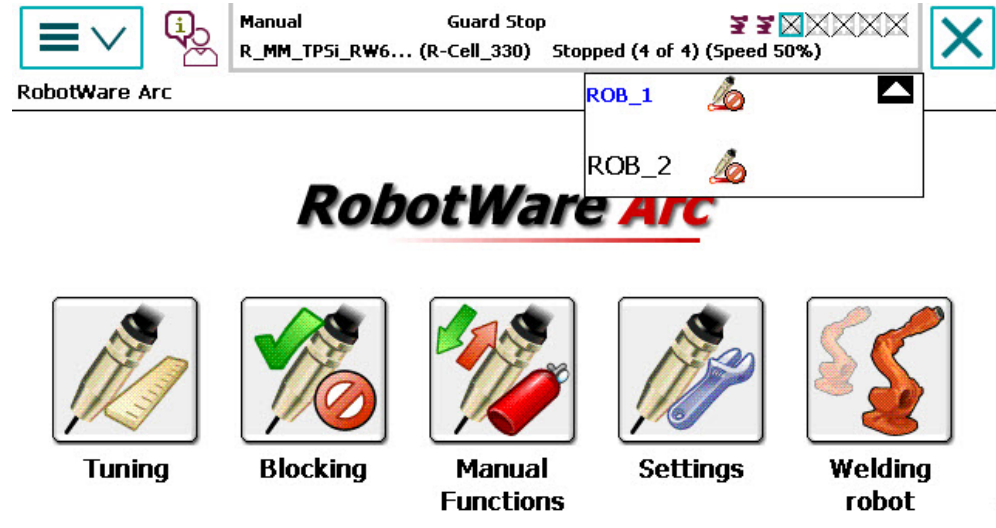
en1200000516

Weld data tuning

The `welddata` **components** `weld_speed`, `weld_wirefeed`, **and** `weld_voltage` **can be tuned using the** `welddata` **tuning function.**

There are two stored values for the tunable data. They are:

1 present value (`weld_speed`, `weld_wirefeed`, **and** `weld_voltage`)

2 original value (`org_weld_speed`, `org_weld_wirefeed`, **and** `org_weld_voltage`)

This allows you to see how much the original value was changed and also to revert to the original value.

During tuning, it is always the present value that is changed. The original value can also be changed by setting it to the same value as the present value.

These changes can also be made from the **Program Data** window.

Tuning weld data

1 Tap **Tuning**.

A window will appear containing functionality for tuning variables of type `welddata`.

2 Select `welddata` to be tuned by tapping the drop-down menu and selecting the desired welding data.

3 If more than 20 `welddata` variables are defined in this task the drop-down menu will be replaced by a text box and a button. Pressing the button opens up a dialog from where it is possible to select other `welddata` to tune.

4 Select the appropriate component in the `welddata` to be tuned by tapping on it.

5   Tap **-** or **+** to decrease or increase the value. Each tap will decrease/increase the value in increments. The tuning increment is preset. For adjustment of the increment see *Data tuning on page 32*.

6   To reset the tuning value, tap **Revert**. The present value will be reset to the original value.

   To reset the original value to the present value, tap **Update Origin**.

7   Tap **OK**.

Tuning weave data

1   Tap **Tuning**.

2   Tap the **Weave tuning** tab to access weaving data.

en1200000646

The `weavedata` tuning dialogs have exactly the same functions as the `welddata` tuning dialogs.

The tunable components are: `weave_width`, `weave_height`, and `weave_bias`.
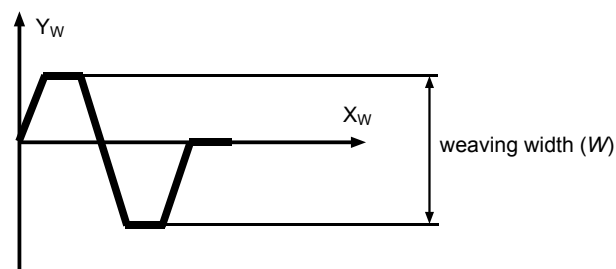
xx1200000647

Continues on next page
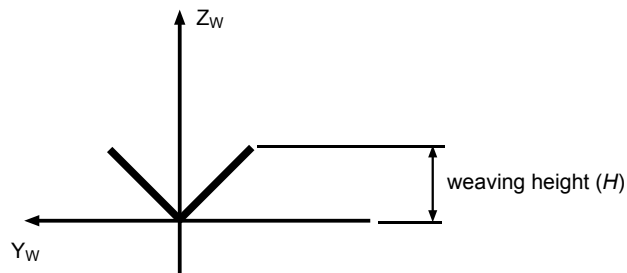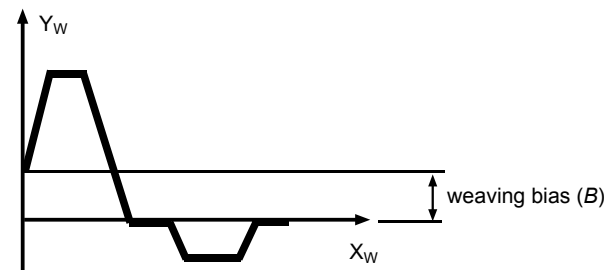
xx1200000648

xx1200000649

Process blocking

Using this display, the operator has the possibility to block welding, weaving, tracking and/ or all. This may be useful during programming or testing phase.

1 Tap **Blocking**.

2 Tap the desired process icon to switch between active and blocked state.

Block All blocks welding, weaving, and tracking and forces the robot to use programmed speed (that is, the `speed` argument).

3 Tap **OK** to confirm or **Cancel** to discard changes.

Blocking can also be activated by setting the digital process blocking inputs.

The parts of the process that have been blocked will be shown on the top border in all RobotWare Arc windows. The blocking status indication is valid in both Manual and Auto mode.
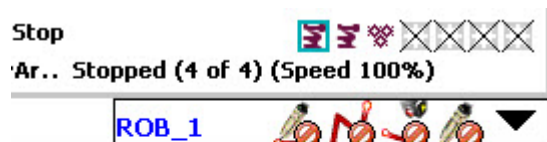
xx1200000650

Blocking that is activated in the above dialog, is active only in the Manual operating mode. It is, however, possible to allow blocking in Auto mode if the arc welding system parameter *auto inhib* is *On*.

> ### ℹ Note
>
> If more than one system is configured in the robot, blocking from the dialog will affect all systems. The digital process blocking inputs will only affect the corresponding system.

Communicate with seam tracker sensor

This display shows the corrections to the seam path, generated by the sensor.

> ### ℹ Note
>
> The seam tracker sensor communication functions can only be used if the robot is configured for use of a seam tracker sensor.

1   Tap **Manual** functions.

    This dialog has two functions:

    - Switch the sensor on/off by tapping the sensor icon.
    - Get the current sensor data by selecting a joint number in the combo box.
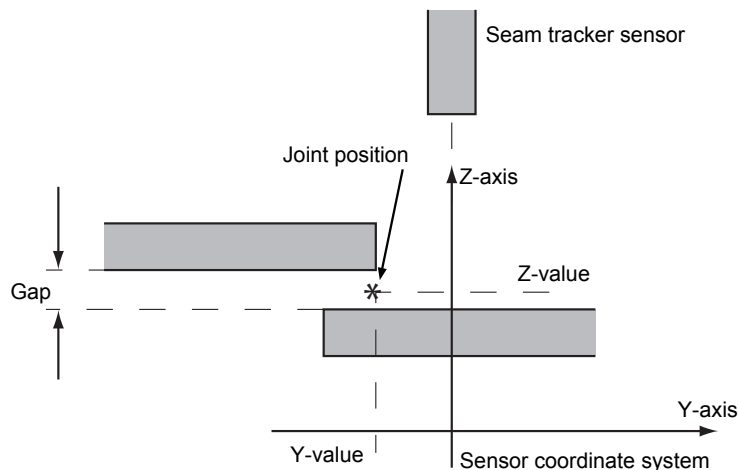
xx1200000651

2   Tap **Close** to close the window.

Manual wirefeed

1   Tap **Manual Functions**.

2   Tap and hold the forward or backward icons to feed the wire. The wire will be fed forward or backward at 50 mm/s, as long as the icon is pressed.

3   Tap the stickout icon to feed 15 mm wire (for each tap).

4   Tap **Close** to close the window.

Continues on next page

> ℹ️ **Note**
>
> If more than one system is configured in the robot, the dialog for selection of arc welding systems can be used to select the corresponding wire feed equipment.

Manual gas purge

1 Tap **Manual Functions**.

2 Tap and hold the gas icon to purge gas.

   The gas valve will be open as long as the icon is pressed.

3 Tap **Close** to close the window.

> ℹ️ **Note**
>
> If more than one system is configured in the robot, the dialog for selection of arc welding systems can be used to select the corresponding gas valve.

Select arc welding system

Up to three arc welding systems can exist at the same time in the robot.

1 Tap **Settings**.

2 Tap a system in the section **System settings** to select a system.

3 Tap **OK** to confirm.

   If **Cancel** is tapped, the original arc welding system is retained as the current system. When a system has been selected as the current system, all other manual functions will operate on this system.

The selection of the arc welding system determines which equipment is active when manual operations, that is, *Gas On*, *Manual Wirefeed* are executed.

Changing tuning increments

1 Tap **Settings**.

2 In the section **Tuning increments**, tap a line to change the increment values.

3 Change the value using the numerical keys.

4 Tap **OK** to close the window and activate the chosen values.

   Tapping **Cancel** discards the changes and closes the window.

3.3 Functions for arc welding during program execution

General

Arc welding functions during program execution:

- Weld data tuning
- Weave data tuning
- Measured value display

Data tuning

During program execution only the present values can be tuned. The original values can only be tuned when the program is stopped.

> **Note**
>
> This chapter refers only to Weld Tuning. The functionality is exactly the same for Weave Tuning.

1 Press **Start** to start the program.

The tuning window is displayed with a list of tunable data if an arc welding instruction is executing.

en1200000652

If no arc welding instruction is executing, then the tuning window is blocked and a yellow label is displayed.
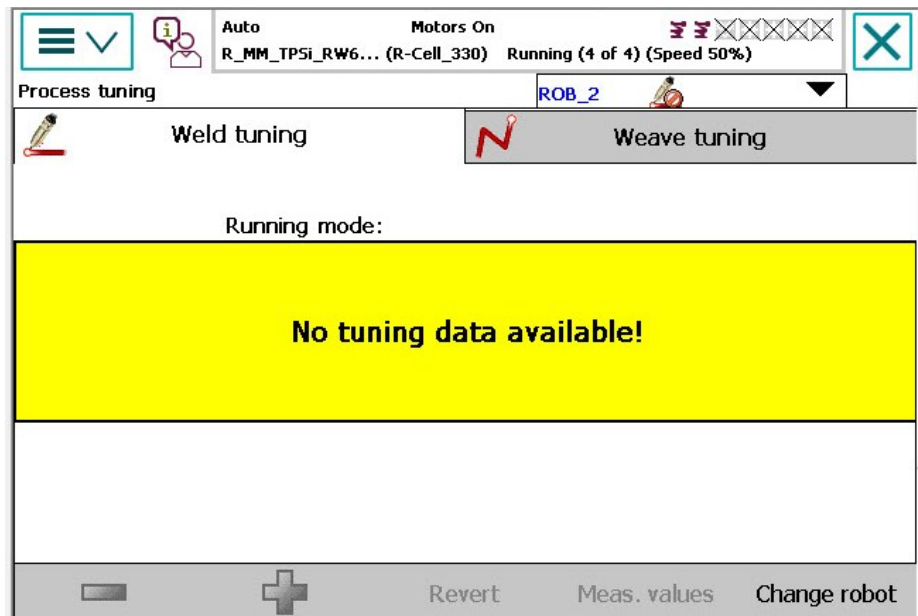
en1200000653

2   Select the data type to be tuned - **Weld Tuning** or **Weave Tuning** - by using the tabs in the tuning window.

3   Select the appropriate component in the `welddata` to be tuned, by tapping on it.

4   Tap **-** or **+** to decrease or increase the value.

Each time these buttons are tapped, the value will decrease/increase in increments. The tuning increment is preset. For adjustment of the increment see *Changing tuning increments on page 31*.

To reset the tuning value, tap **Revert**. The present value is reset to the original value.

This page is intentionally left blank

4 Programming RobotWare Arc systems with MultiMove

4.1 RobotWare Arc with MultiMove

Introduction

The RobotWare Arc functionality for MultiMove systems is similar to the functionality in single arc welding systems. Two or more welding robots are programmed in separate tasks running independent or synchronized.

The user interface provides the possibility to select an active welding robot for the functions to operate on.

This chapter also describes how to install and configure an arc welding MultiMove system.

4.2 Functions for arc welding when program execution has been stopped

Functions

The arc welding functions on the FlexPendant for MultiMove arc welding systems are similar to those for single arc welding systems, except for the possibility to select an active welding robot for the functions to operate on.

- Arc welding functions (program execution has been stopped) in Manual mode:
- Weld data tuning
- Weave data tuning
- Communicate with seamtracker sensor
- Process blocking
- Manual wire feed
- Manual gas on/off
- Select arc welding system
- Changing tuning increments
- Select active arc welding robot

Arc welding functions (program execution has been stopped) in Auto mode:

- Manual wire feed
- Manual gas on/off
- Select arc welding system
- Changing tuning increments
- Select active arc welding robot

> **Note**
>
> If a window is open in Manual mode and corresponding functionality is inhibited in Auto mode, switching from Manual to Auto mode will close the window.

RobotWare Arc on FlexPendant

To start RobotWare Arc tap the ABB menu and then tap RobotWare Arc. The RobotWare Arc desktop contains all arc welding functions. In a MultiMove arc welding system an extra icon, **Welding robot**, will be available. The name of the currently selected welding robot is always visible in the upper right corner, together with the process blocking status. It is possible to click the drop-down menu on the upper right corner to see the blocking status for all welding robots in the system.

> **Note**
>
> The definition of a MultiMove arc welding system is a MultiMove system that contains two or more welding robots. A MultiMove system with only one welding robot will appear and behave as a single arc welding system.

Continues on next page
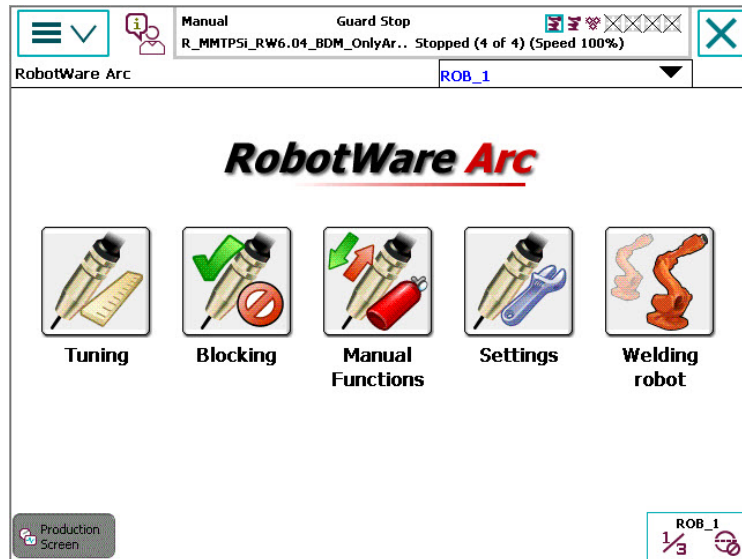
en1200000687

en1200000688

Selecting active welding robot

By selecting an active welding robot, all functions like gas purge, wirefeed, sensor communication, blocking, tuning and system settings will to operate on data or signals belonging to the active robot.

1 Tap the welding robot icon.

2 Select the welding robot to activate.

Continues on next page

3   Tap **OK** to confirm or **Cancel** to discard changes.

en1200000689
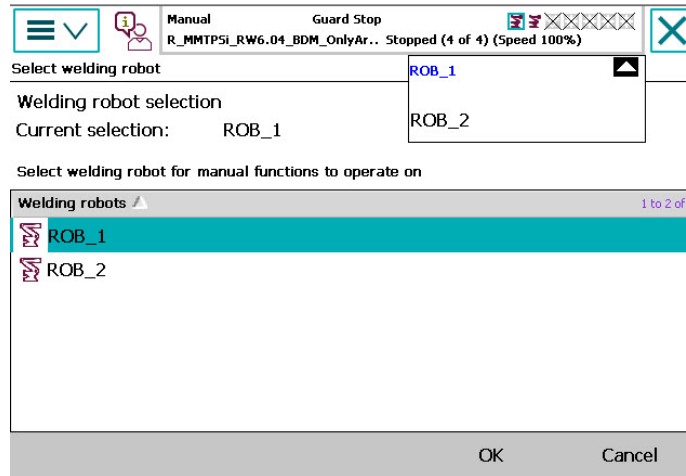
It is also possible to change the active welding robot in the drop-down menu. Tap the line that corresponds to the welding robot to be selected, see figure in *Process blocking on page 39*.

![Note icon]   **Note**

Selecting active welding robot does not affect the jogging settings. The active welding robot is only valid in the scope of RobotWare Arc.

Weld and weave data tuning

The weld and weave tuning functionality is similar to the functionality described in *Programming for arc welding on page 21*, except that the tuning operates on data belonging to the active welding robot task.

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

Process blocking

The process blocking functionality is similar to the functionality described in *Functions for arc welding when program execution has been stopped on page 26*, except that the blocking is valid per welding robot. The blocked processes are visible at the upper right corner of all RobotWare Arc screens. Tap the dropdown menu to see the blocking status for all welding robots in the system. Tap on the dropdown menu again to close it.
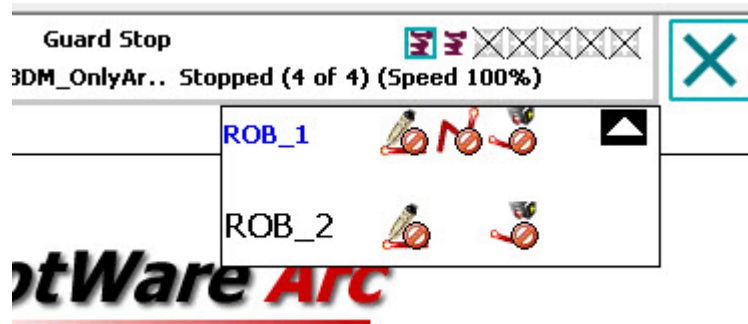
en1200000690

Communicate with seamtracker sensor

The sensor functionality is similar to the functionality described in *Functions for arc welding when program execution has been stopped on page 26*, except that the sensor functions operate on the active welding robot sensor.

> ### ℹ Note
>
> This functionality is available only if the active welding robot is configured with a seamtracker sensor.

Manual wire feed

The wire feed functionality is similar to the functionality described in *Functions for arc welding when program execution has been stopped on page 26*, except that the wire feed operates on the active welding robot.

Manual gas purge

The gas purge functionality is similar to the functionality described in *Functions for arc welding when program execution has been stopped on page 26*, except that the gas purge operates on the active welding robot.

Select welding system

The welding system functionality is similar to the functionality described in *Functions for arc welding when program execution has been stopped on page 26*, except that the welding system is valid for the active welding robot.

Changing tuning increments

The tuning increment functionality is similar to the functionality described in , except that the increments are valid for the active welding robot.

4.3 Functions for arc welding during program execution

Functions

Arc welding functions during program execution:
- Weld data tuning
- Weave data tuning
- Measured value display
- Selecting active welding robot

Data tuning

The data tuning functionality is similar to the functionality described in *Functions for arc welding during program execution on page 32*, except that the tuning operates on data belonging to the active welding robot task.

Measurement values

The measurement values functionality is similar to the functionality described in *Functions for arc welding during program execution on page 32*, except that the values displayed belongs to the active welding robot.

Selecting active welding robot

It is possible to change active welding robot during program execution.

1 Tap **Change robot**.

2 Select active welding robot.

3 Tap **OK** to confirm or **Cancel** to discard changes.

The weld and weave data tuning will now operate on data belonging to the active welding robot.

4.4 Configuration

Introduction

In a MultiMove system, the configuration parameter *MotionTimeout* is of great importance, especially when running in synchronized mode. The parameter should have a non-zero value to be able to shut down process equipment when one of the robots does not start the intended motion after a certain time frame.

Example

The robots ROB1 and ROB2 are both welding in synchronized mode. The *MotionTimeout* parameter is set to 1s. Since they are running in synchronized mode, both of the robots TCP should arrive at the starting position of the weld at the same time. Both robots strike the arc at the same time. ROB1 gets the *Arc OK* signal, the robot is ready to start the motion, the motion timer starts to tick. ROB2 has a problem to ignite properly. That means that during this period ROB1 is standing still with the arc on. The motion timeout will cause an error after 1 second in ROB1. Then the error ERR_PATH_STOP will be distributed to the other motion tasks to react on. This parameter is used to avoid that one of the robots is standing still with the arc ignited and burning through the material.

> **Note**
>
> When running in synchronized mode, the motion timeout must not be lower than the ignition timeout value. There is otherwise a risk that the motion timer will expire before the ignition timer. Since the motion timeout error is non recoverable, it will hide the real error, arc ignition timeout. The recommendation is to set the ignition timeout value some milliseconds shorter than the movement timeout value, 0.05 seconds is sufficient.

Error handling

Error handling in a MultiMove setup (running synchronized) requires that the error handlers are the same in all robot tasks. That is due to the fact that if there is an error in one robot, the other robots will also end up in their local error handler.

Example 1

Automatic retries directly after an error.

If *no_of_retries* is set to a value other than 0, automatic retries will be performed by RobotWare Arc until *no_of_retries* has expired. Then the user error handler will be executed.

If the error handler has the following contents, it will be executed until the error is fixed or the SYS domain parameter *-NoOfRetry* has expired.

```
MoveJ p1, v1000, fine, Rob2_tool\WObj:=wobj_STN1;
ArcLStart p2, v1000, sm1, wd1_ind\Weave:=wv0, fine,
    Rob2_tool\WObj:=wobj_STN1;
ArcLEnd p6, v1000, sm1, wd1_ind\Weave:=wv0, fine,
    Rob2_tool\WObj:=wobj_STN1;
ERROR
StorePath;
```

```
RestoPath;
StartMoveRetry;
```

Example 2

Automatic retry after cleaning the welding torch.

The following is an example of an error handler with the possibility to move to a service position in the failing robot, clean the welding gun, go back to the error location and start welding again. The other robots will wait for the failing robot to get ready and they will all restart the synchronized motion again when the failing robot executes `StartMoveRetry`.

```
VAR robtarget errPos1;
VAR tooldata tErr;
VAR wobjdata obErr;
MoveJ p1, v1000, fine, Rob2_tool\WObj:=wobj_STN1;
ArcLStart p2, v1000, sm1, wd1_ind\Weave:=wv0, fine,
    Rob2_tool\WObj:=wobj_STN1;
ArcLEnd p6, v1000, sm1, wd1_ind\Weave:=wv0, fine,
    Rob2_tool\WObj:=wobj_STN1;
ERROR
IF ERRNO=AW_WELD_ERR THEN
  StorePath;
  errPos1:=CRobT(\Tool:=tErr\WObj:=obErr);
  MoveL RelTool(errPos1,0,0,-20),v100,fine,tErr\WObj:=obErr;
  TPWrite "Cleaning...";
  WaitTime 1;
  MoveL errPos1,v100,fine,tErr\WObj:=obErr;
  RestoPath;
ELSE
  StorePath;
  RestoPath;
ENDIF
StartMove;
RETRY;
```

Example 3

The following example shows error handling with the possibility to jog away from the path at an error, press start and the welding will resume. Here this is done only at a wire stick error, otherwise automatic cleaning is performed.

```
VAR robtarget errPos1;
VAR tooldata tErr;
VAR wobjdata obErr;
MoveJ p1, v1000, fine, Rob2_tool\WObj:=wobj_STN1;
ArcLStart p2, v1000, sm1, wd1_ind\Weave:=wv0, fine,
    Rob2_tool\WObj:=wobj_STN1;
ArcLEnd p6, v1000, sm1, wd1_ind\Weave:=wv0, fine,
    Rob2_tool\WObj:=wobj_STN1;
ERROR
IF ERRNO=AW_WIRE_ERR THEN StorePath;
  TPWrite "This error is caused by wire stuck";
  TPWrite "Cut the wire and press start !";
  Stop;
```

```
                    RestoPath;
                    StartMove;
                    RETRY;
             ENDIF
             IF ERRNO=AW_WELD_ERR THEN
                ! Automatic move to cleaning position
                ! Move back to error position and start welding again.
                StorePath;
                errPos1:=CRobT(\Tool:=tErr);
                MoveL RelTool(errPos1,0,0,-50),v10,fine,tErr;
                TPWrite "Cleaning...";
                WaitTime 1;
                MoveL errPos1,v10,fine,tErr;
                RestoPath;
                StartMove;
                RETRY;
             ENDIF
```

Instructions in non-welding robot

Programming RobotWare Arc in synchronized mode with instruction id's requires some special considerations for the error handling to work correctly. In the non-welding robot or additional axis, some new instructions must be used when there are corresponding weld instructions in the welding robots.

The instructions should be used to ensure that the automatic retry functionality works correctly and that the error levels are the same in all motion tasks.

Example 1

FlexPositioner (`ArcMoveJ` instead of `MoveJ`)

```
T_ROB1 (non-welding robot):
ArcMoveJ p2 \ID:=101, v1000, z1, tSvetsbord;
T_ROB2:
ArcL p2 \ID:=101, v1000, sm1, wd2, wv1,z1,wGun_ROB2\WObj:=WOBJ_ROB1;
T_ROB3:
ArcL p2 \ID:=101, v1000, sm1, wd2, wv1, z1,
    wGun_ROB3\WObj:=WOBJ_ROB1;
```

Example 2

TwinArc (`ArcMoveExtJ` instead of `MoveExtJ`)

```
STN1 (additional axis):
ArcMoveExtJ p2 \ID:=101, v1000, z1;
T_ROB1:
ArcL p2 \ID:=101, v1000, sm1, wd2, wv1, z1,
    wGun_ROB1\WObj:=WOBJ_STN1;
T_ROB2:
ArcL p2 \ID:=101, v1000, sm1, wd2, wv1, z1,
    wGun_ROB2\WObj:=WOBJ_STN1;
```

Move instructions

The following list shows the move instructions and the corresponding instruction to use in the non-welding motiontask.

Move instructions	Arc instructions
MoveJ	ArcMoveJ
MoveL	ArcMoveL
MoveC	ArcMoveC
MoveAbsJ	ArcMoveAbsJ
MoveExtJ	ArcMoveExtJ

Configure error handling

The error handling in terms of severity levels of the error, can be configured in detail. See .

4.5 Limitations

Restart distance

It is not possible to have different restart distances if running synchronized motions. Since it is not possible to determine which robot that controls the restart distance in this case, the recommendation is to have the same parameter values in each robot.

Use of finepoint

Finepoint must be used in the arc welding instruction before:

- SyncMoveOn
- SyncMoveOff
- WaitSyncTask

Error handling

If an error handler is present, but it does not handle the error, that is none of the instructions `RETRY`, `TRYNEXT`, `RETURN`, or `RAISE` are present in the error handler, then the active motion path is cleared. That means, that neither *regain to path* nor *backing on the path* is possible. The robot movement starts from the current position of the TCP, which might result in a *path shortcut*.

RaiseToUser problem

If `ArcL/ArcC` instructions are encapsulated by `NOSTEPIN`/`NOVIEW` routines, the ERROR handler of this `NOSTEPIN`/`NOVIEW` routine is ignored for the following recoverable errors:

- AW_START_ERR
- AW_IGNI_ERR
- AW_WELD_ERR
- AW_EQIP_ERR
- AW_WIRE_ERR
- AW_STOP_ERR
- AW_TRACK_ERR
- AW_TRACKSTA_ERR
- AW_TRACKCORR_ERR
- AW_USERSIG_ERR
- ERR_PATH_STOP

The system looks for error handlers to be run, starting with the first STEPIN routine found in the RAPID call chain.

Example

```
MODULE MY_PROG
  PROC main ()
    MyArcL;
    ERROR
    TPWrite "main error handler";
```

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

```
      ENDPROC
   ENDMODULE
   MODULE MY_ARC (SYSMODULE, NOVIEW)
     PROC MyArcL ()
       ArcL;
       ERROR
       TPWrite "MyArcL error handler!";
     ENDPROC
   ENDMODULE
```

If an error occurs in `ArcL`, the error handler of `MyArcL` is NOT executed (because `MyArcL` is part of a `NOSTEPIN/NOVIEW` module), but the error handler of main is executed.

Missing instructions in additional axis

If MultiMove cells are configured and setup with an additional axis or positioner, without any external option from ABB of the type ABB ATRM/AW System Disk, then the awBase.sys RAPID module must be installed in that task.

If this is not done, it will not be possible to select any of the following non-welding arc instructions from the pick list on the FlexPendant.

- **ArcMoveExtJ**
- **ArcMoveAbsJ**
- **ArcMoveL**
- **ArcMoveC**
- **ArcMoveJ**

These must be used to have proper error handling when using RobotWare Arc in MultiMove. The installation of the module is done by adding the following to SYS.cfg, where task name represents the name of the additional axis RAPID task name.

```
CAB_TASK_MODULES:
#
-File RELEASE:/options/arc/ArcBase/code/awBase.sys
-ModName "awBase" -Install -Task "taskname"
```

This page is intentionally left blank

5 Weld Error Recovery

5.1 Weld Error Recovery and error handling

Weld Error Recovery

During robot production process errors sometimes stop the robot. The Weld Error Recovery feature provides several different solutions for process error recovery, which allows operators to automatically move the robot out from the error position to get a better overview of the torch. After the process error is corrected the robot automatically returns back to the error location and continues production. This will help minimizing production downtime.

Since the creation of safe collision free escape paths for error handling often is more time consuming than the creation of the actual production program, error handling under program control is rarely utilized. That is why the Weld Error Recovery feature is always included with RobotWare Arc, and the basic error recovery features are available without any additional programming. This includes FlexPendant screens to provide standard error recovery support for the welding process.

Advanced features such as the ability to escape to a service location, require additional programming on the part of the user. The Weld Error Recovery feature will store position information during execution of the production program, utilizing a built-in Path Recorder. When an error occurs the stored sequence of position data is traversed backwards extracting the robot from the work piece. Thus, the path recorder eliminates any need for additional programming of escape paths.

Basic weld error handling

In its simplest form, when a welding error occurs, a simple prompt will be presented to the user on the FlexPendant.

en1200000691

If **Abort** is tapped the program execution will stop and the weld routine in the program editor window will be shown.

If **Move Out** is tapped the robot will attempt to move out a small distance along the tool center line. The **Error** menu will be shown again. **Move Out** can be tapped repeatedly. If **Recovery** menu is tapped, the user is presented with the **Recovery** menu.
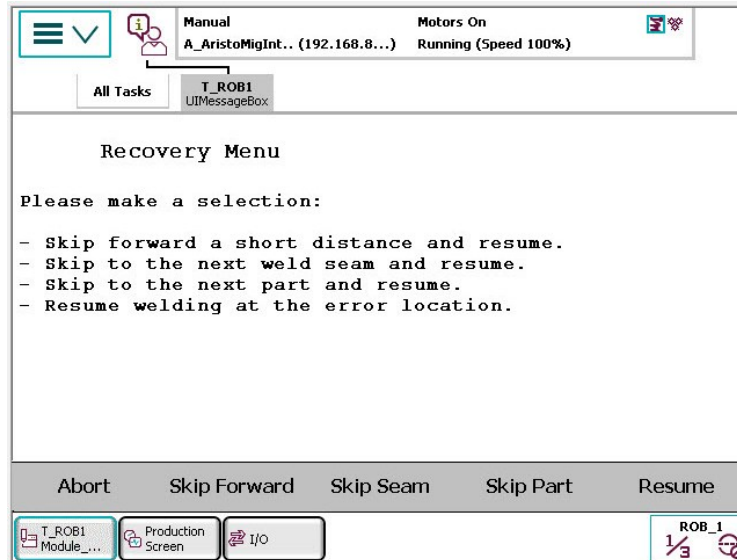
en1200000692

The **Recovery** menu is possible to configure to allow for the user to block some of the available resume features. For example, the user may choose to disable the "Skip Seam" option. This is described in the **Recovery** menu configuration section.

The user can add escape functionality to the **Error** menu by introducing recovery set points in the program. This allows the user to access the **Escape** button of the Weld Error Recovery feature, which is reflected in the **Error** menu.
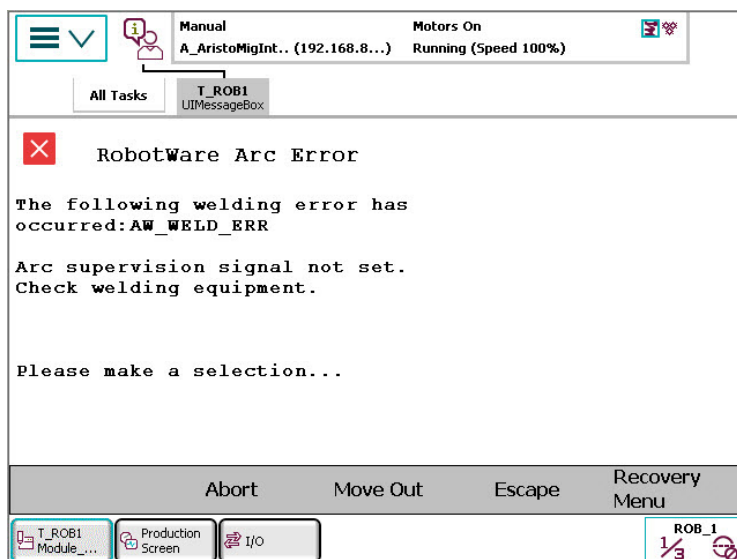
en1200000693

5.2 Programming Weld Error Recovery

Basic usage - Example

The user programs a simple weld routine without adding any of the advanced tools provided by Weld Error Recovery.

```
PROC WeldMyTruck ()
MoveJ *,vmax,z10,tWeldGun; MoveJ *,vmax,z10,tWeldGun;
ArcLStart *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
ArcL *,v500, sm1,wd1\Weave:=wv1,z10,tWeldGun;
ArcL *,v500, sm1,wd1\Weave:=wv1,z10,tWeldGun;
ArcLEnd *,v500, sm1,wd1\Weave:=wv1,fine,tWeldGun;
MoveJ *,vmax,z10,tWeldGun;
MoveJ *,vmax,z10,tWeldGun;
ENDPROC
```

If an error occurs during the weld seam, the Error Menu will be presented without the **Escape** button:

en1200000691

The user can tap **Move Out** to extract the tool from the partially welded part in increments. Tapping **Abort**, stops execution. Tapping **Recovery** menu will bring up the **Recovery** menu.
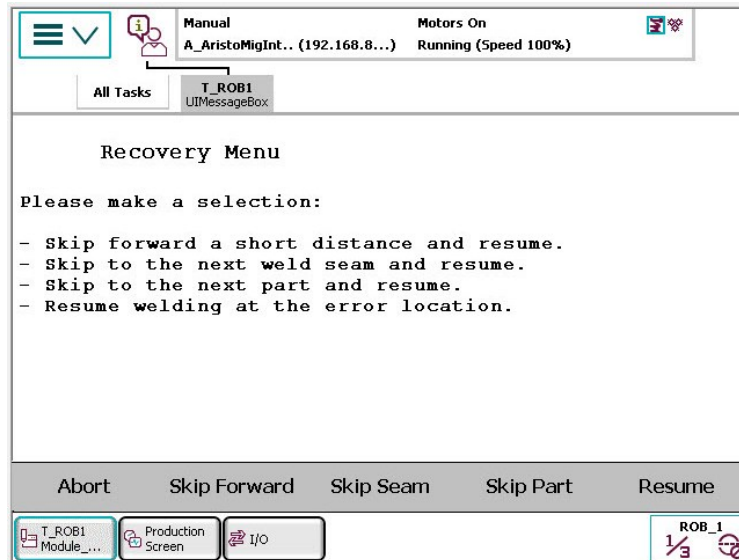
en1200000692

When **Resume** is selected the robot executes a standard retry with the configured restart distance.

If **Skip Seam** is selected, the robot will finish the seam without welding. The specified welding speed will be used for the remaining part of the segment, the next segment within the same seam will use the speed specified in the `Speed` argument of the `ArcX` instruction. Welding will resume at the next `ArcLStart` instruction.

If **Skip Part** is selected, the robot will run without welding until the next part is executed (*Production Manager*) or until the `RecoveryPosReset` instruction is executed. The specified welding speed will be used for the remaining part of the segment, the next segments will use the speed specified in the `Speed` argument of the `ArcX` instruction. Welding will resume at the next `ArcXStart` instruction.

If **Skip forward** is selected, the robot will skip forward a selectable distance on the programmed path without process, and then make a normal weld retry with process

activation at that position. The forward skip distance is entered via the following user dialog.

en1200000695

Advanced usage - Example 1

By adding a recovery set point, escape is made possible. Consider this example:

```
PROC WeldMyTruck()
RecoveryPosSet;
MoveJ *,vmax,z10,tWeldGun;
MoveJ *,vmax,z10,tWeldGun;
ArcLStart *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
ArcL *,v500, sm1,wd1\Weave:=wv1,z10,tWeldGun;
ArcLEnd *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
MoveJ *,vmax,z10,tWeldGun; MoveJ *,vmax,z10,tWeldGun;
RecoveryPosReset;
ENDPROC
```

The instruction `RecoveryPosSet` is used to set the recovery set point. If an error occurs during the weld seam, the error menu will display an **Escape** button:
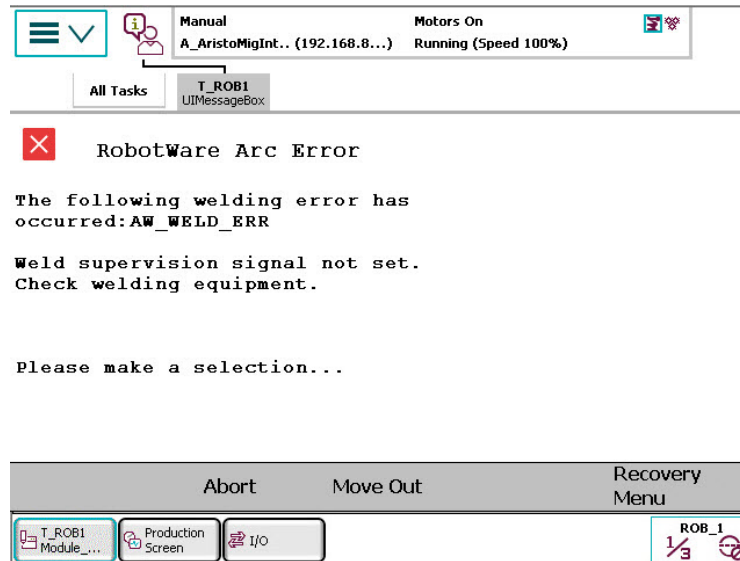
en1200000691

Tapping **Escape** causes the robot to retrace its path to the recovery position set by the `RecoveryPosSet` instruction. At that location the recovery menu is displayed. This simple implementation is useful when the user would like the robot to move back to a position that is clear of the part and accessible for service.

The path recorder is stopped and the service routine cleared using the RAPID instruction, `RecoveryPosReset`. The instruction takes no arguments. This instruction should be used at the end of the weld sequence to ensure that the path recorder is stopped and cleared before starting a new weld sequence. A failure to do so could result in undesirable results, as an old recovery set point could remain active during a new weld sequence. This type of implementation is typically done in the following way:

```
PROC main()
  MoveJ pSafe,vmax,fine,tool0;
  RecoveryPosSet;
  TEST nSelection
  CASE 1:
    WeldMyTruck;
  CASE 2:
    WeldMyCar;
  ENDTEST
  RecoveryPosReset;
ENDPROC
PROC WeldMyTruck()
  MoveJ *,vmax,z10,tWeldGun;
  MoveJ *,vmax,z10,tWeldGun;
  ArcLStart *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
  ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
  ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
  ArcLEnd *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
```

```
      MoveJ *,vmax,z10,tWeldGun;
      MoveJ *,vmax,z10,tWeldGun;
   ENDPROC
```

This type of implementation also provides escape behavior for multiple part procedures shown in the test case logic above.

Advanced usage - Example 2

Recovery positions may be set at any point in a weld sequence. In some cases it may be necessary to have an alternate recovery position that is set mid-weld. This is perfectly ok.

```
PROC WeldMyCar()
   RecoveryPosSet;
   MoveJ *,vmax,z10,tWeldGun;
   MoveJ *,vmax,z10,tWeldGun;
   ArcLStart *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
   ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
   SetDO doClamp,high;
   RecoveryPosSet;
   ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
   ArcLEnd *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
   MoveJ *,vmax,z10,tWeldGun;
   MoveJ *,vmax,z10,tWeldGun;
   RecoveryPosReset;
ENDPROC
```

An implementation like this is useful if the robot is not permitted to move backward past the `SetDO` instruction.

Advanced usage - Example 3

Using the service routine feature will extend the Weld Error Recovery escape functionality. The service routine is a user-defined procedure that is launched after the robot retraces a recorded path back to a recovery position. The routine may be used to move the robot from the recovery position to a service location, or any other behavior that can be implemented in RAPID. Consider the following example:

```
PROC main()MoveJ pSafe,vmax,fine,tool0;
   RecoveryPosSet\ServRoutine:="ServiceRoutine";
   TEST nSelect
     CASE 1:
       WeldMyTruck;
     CASE 2:
       WeldMyCar;
   ENDTEST
   RecoveryPosReset;
ENDPROC
PROC WeldMyTruck ()
   MoveJ *,vmax,z10,tWeldGun;
   MoveJ *,vmax,z10,tWeldGun;
   ArcLStart *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
   ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
   ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
```

```
            ArcLEnd *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
            MoveJ *,vmax,z10,tWeldGun;
            MoveJ *,vmax,z10,tWeldGun;
        ENDPROC
        PROC ServiceRoutine()
            MoveJ *,vmax,z10,tool0;
            MoveJ *,vmax,z10,tool0;
            MoveL pService,vmax,z10,tool0;
            RecoveryMenu;
            MoveL *,vmax,z10,tool0;
            MoveJ *,vmax,z10,tool0;
            MoveJ pSafe,vmax,z10,tool0;
        ENDPROC
```

In this example, the optional argument `ServRoutine` is applied to `RecoveryPosSet`. The procedure name *ServiceRoutine* has been applied as the name of the service routine. If an error occurs during the weld seam and the user selects Escape from the Error Menu, the robot will retrace its path back to the `RecoveryPosSet` location. Then the *ServiceRoutine* procedure will be executed. If the specified *ServiceRoutine* cannot be located in the RAPID program, the following user menu will be displayed.
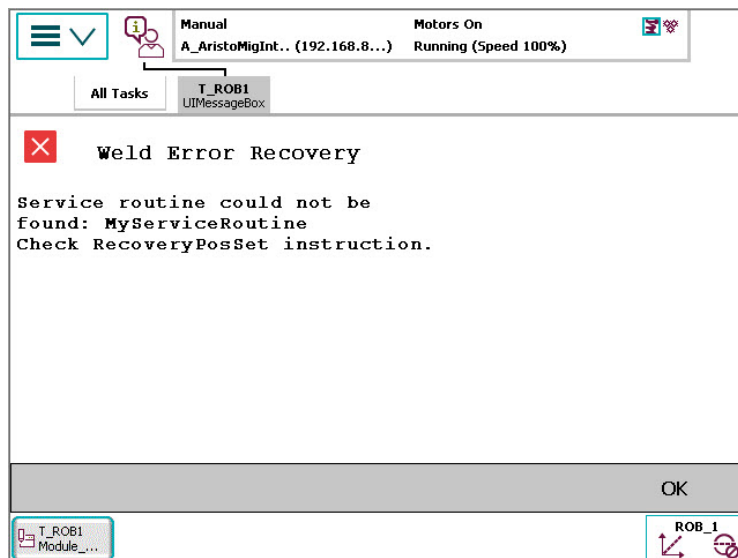
en1200000696

Tapping OK continues program execution as if no *Serviceroutine* has been specified.

The *ServiceRoutine* example above is an example of a service routine that can be created by a RAPID programmer. The Weld Error Recovery feature does not provide the *ServiceRoutine* procedure. In this example the service routine contains move instructions that move the robot from the safe position, pSafe, to a special service position called `pService`. Once this position is reached, an instruction called

`RecoveryMenu` is called. The Weld Error Recovery feature provides this instruction. `RecoveryMenu` is a RAPID instruction that launches the standard Recovery Menu.
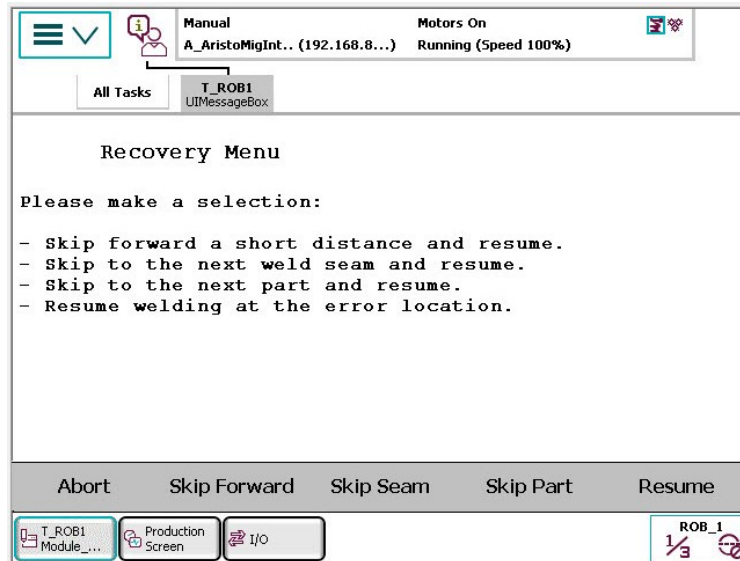
en1200000692

After the operator makes the recovery choice, the robot executes the programmed moves back to the recovery set point location, in this case `pSafe`. This completes the user-defined *ServiceRoutine* procedure. If the robot is not moved back to the recovery set point location in the *ServiceRoutine*, the following user dialog will be displayed.
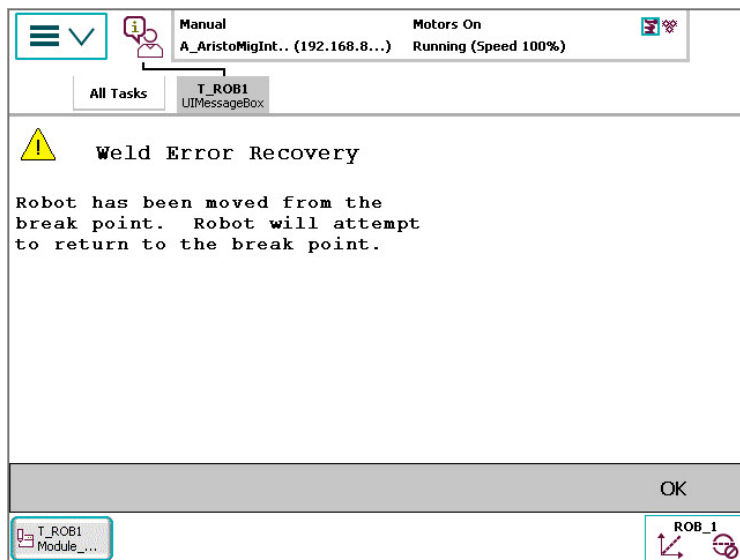
en1200000697

Tapping OK moves the robot to the recovery set point.

At this point, the Weld Error Recovery feature takes over and executes the path recorder to the error location and the selected recovery behavior is executed.

5 Weld Error Recovery

Advanced usage - Example 4

It is possible to create RAPID driven user menus. These menus enable interaction so that an operator can respond by making choices from a menu list.

```
PROC ServiceRoutine()
  MoveJ *,vmax,z10,tool0;
  MoveJ *,vmax,z10,tool0;
  MoveL pService,vmax,z10,tool0;
  ServiceMenu;
  RecoveryMenu;
  MoveL *,vmax,z10,tool0;
  MoveJ *,vmax,z10,tool0;
  MoveJ pSafe,v1000,z10,tool0;
ENDPROC
PROC ServiceMenu()
  VAR num nListIndex;
  VAR listitem liMyItems{2};
  VAR btnres button_answer;
  liMyItems{1}.text:="Service Welding Torch";
  liMyItems{2}.text:="Check Tool Calibration";
  liMyItems{1}.image:="TorchService48.bmp";
  liMyItems{2}.image:="ToolCalibration48.bmp";
  nListIndex:=UIListView(\Result:=button_answer,\Header:="Service
      Menu",liMyItems\Icon:=iconInfo);
  IF nListIndex = 1 THEN
    TorchService;
  ELSEIF nListIndex = 2 THEN
    ToolCalibration;
  ENDIF
ENDPROC
PROC TorchService()
  MoveJ RelTool(pToolClean,0,0,-200),v1000,z1,tool0;
  MoveL pToolClean,v1000,fine,tool0;
  ! Run torch cleaner here
  MoveL RelTool(pToolClean,0,0,-200),v1000,z1,tool0;
  MoveJ pService,v1000,z10,tool0;
ENDPROC
PROC ToolCalibration()
  MoveJ RelTool(pToolCalib,0,0,-200),v1000,z1,toll0;
  MoveL pToolCalib,v1000,fine,tool0;
  ! Run BullsEye TCP calibration here
  MoveL RelTool(pToolCalib,0,0,-200),v1000,z1,toll0;
  MoveJ pService,v1000,z10,tool0;
ENDPROC
```

In this example we have extended the service routine with a call to a user defined service menu, called *ServiceMenu*. The service menu will present two choices for

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

the operator, *Service Welding Torch* and *Check Tool Calibration*. This is what the service menu in this example would look as follows.
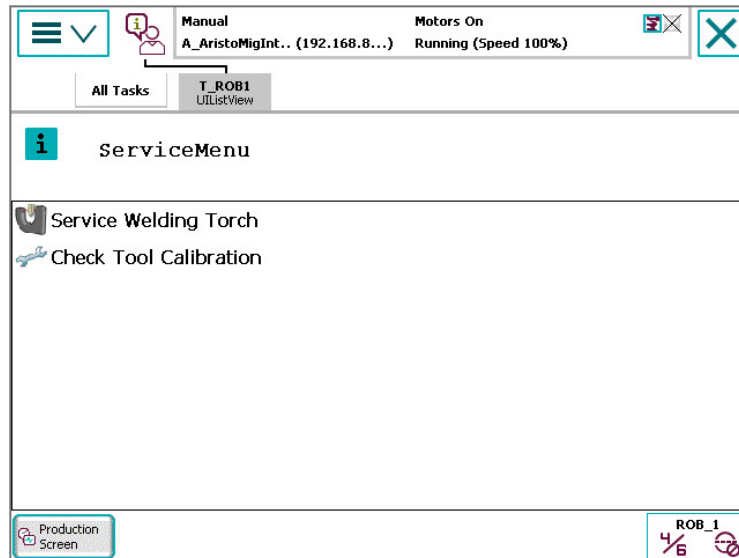
en1200000698

If the operator selects *Service Welding Torch*, the routine `TorchService` will be executed. In this example the torch service routine contains move instructions that move the robot from the service position, pService, to the torch service position, pToolClean. Once this position is reached, instructions for running the torch cleaner device may be added to this routine. After the torch has been serviced the robot executes the programmed moves back to the service location, in this case pService. This completes the user-defined *ServiceRoutine* procedure. The tool calibration routine is implemented in a similar fashion.

5.3 Weld Error Recovery flowchart

Description

The following chart shows the basic program flow of the Weld Error Recovery feature. The symbols marked with the letters A to E represents the available user dialog windows on the FlexPendant.
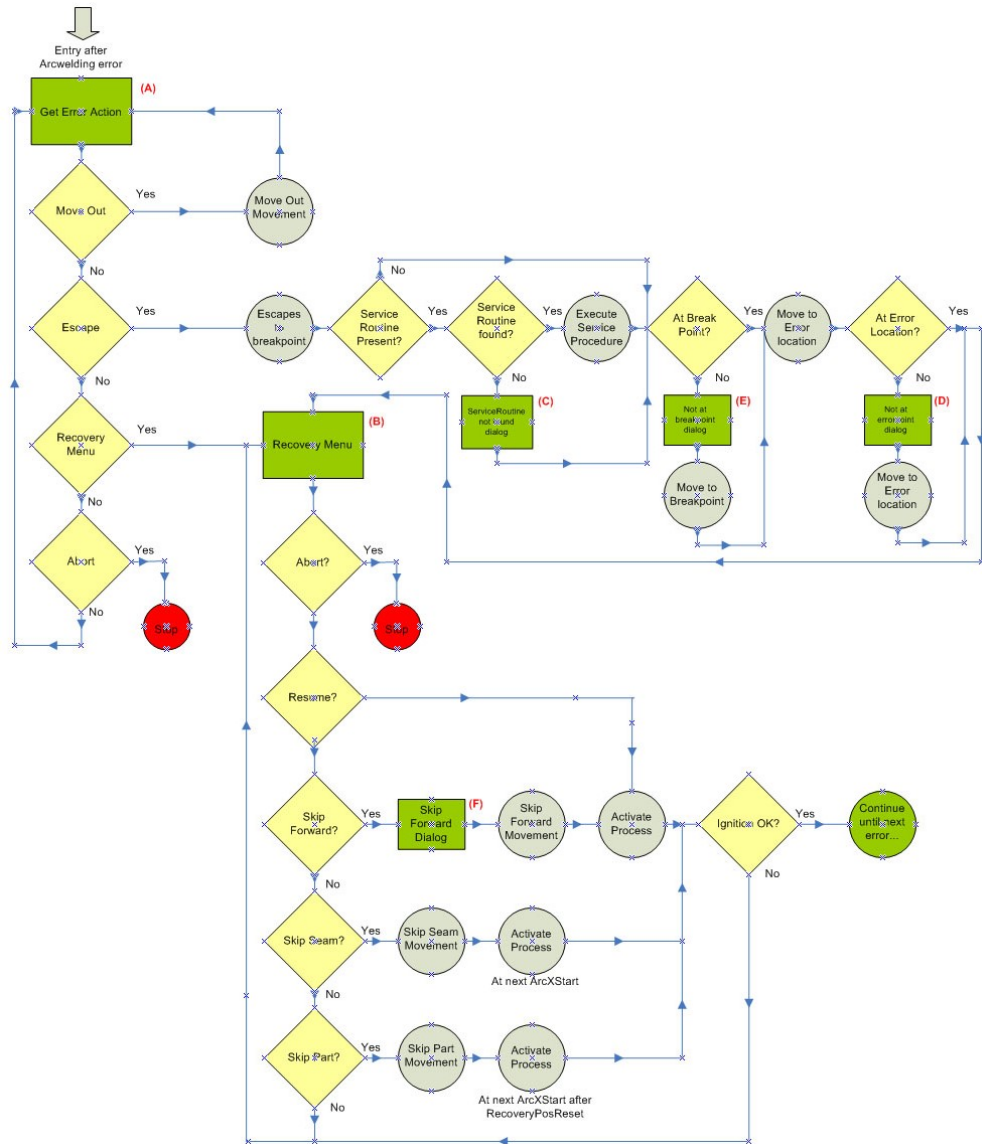
en1200000699

5.4 Configuring Weld Error Recovery

Description

Weld Error Recovery is configured in the system parameters, topic *Process*, type *Arc Error Handler*.

Default values

The default configuration has the following definition.

en1200000694

Parameters

Parameter	Description	Data type
Name	The name of the instance ARC_ERR_HNDL	typeStringNormal
Use Arc Recovery Menu	The reference to instance ARC_RECOVERY_MENU	typeStringNormal
Enabled	If True, the Weld Error Recovery will be used.	typeBoolean
Default Action	Sets the default action that will be executed at process error.	typeFloat

5.4 Configuring Weld Error Recovery
Continued

Parameter	Description	Data type
Default Resume Type	Sets the default resume type that will be automatically returned from recovery menu.	typeFloat
Moveout Distance	Sets the distance for the `MoveOut` **function.**	typeFloat
Pathrecorder Speed	Sets the default path recovery speed.	typeFloat
Pathrecorder Tool Offset	Sets the tool offset that is used during the recovery motions.	typeFloat

Related information

Information	Described in
Set the recovery position	*RecoveryPosSet - Set the recovery position on page 166*
Reset the recovery position	*RecoveryPosReset - Reset the recovery position on page 169*
Display the recovery menu	*RecoveryMenu - Display the recovery menu on page 162*
Configure the recovery menu	*Configure the recovery menu on page 63*

5.5 Configure the recovery menu

Recovery menu

The Arc recovery menu allows the user to chose a suitable recovery method. The recovery menu is configured in the system parameters, topic *Process*, type *Arc Recovery Menu*.

The following selections can be hidden in the recovery menu.

Abort	Tapping **Abort** stops execution and aborts the process.
Skip Forward	If **Skip Forward** is selected the robot will skip forward a short distance from the error location and then execute a standard retry to resume the welding.
Skip Seam	If **Skip Seam** is selected, the robot will finish the seam without welding. The specified welding speed will be used for the remaining part of the seam.
Skip Part	If **Skip Part** is selected, the robot will run without welding until the next part is executed or until the `RecoveryPosReset` instruction is executed. The specified welding speed will be used for the remaining part of the segment, the next segments will use the speed specified in the `Speed` argument of the `ArcX` instruction. Welding will resume at the next `ArcXStart` instruction.
Resume	When **Resume** is selected the robot executes a standard retry at the error location. The robot will move backwards the configured `Restart Distance` before restart.

Examples

The default configuration has the following definition.

en1200000700

Parameters

Parameter	Description	Data type
Name	The name of the instance ARC_RECOV-ERY_MENU	typeStringNormal
HideResumeAtErr	If true, the Resume option will be hidden.	typeBoolean
HideSkipFwd	If true, the Skip Forward option will be hidden.	typeBoolean
HideSkipSeam	If true, the Skip Seam option will be hidden.	typeBoolean
HideSkipPart	If true, the Skip Part option will be hidden.	typeBoolean
HideAbort	If true, the Abort option will be hidden.	typeBoolean

5.6 Weld Error Recovery I/O interface

Usage

The Weld Error Recovery dialogs presented on the FlexPendant may be acknowledged from a remote source through an optional I/O interface. This is necessary if a PLC or other remote computer is used for the primary operator interface while running production.

Architecture

All I/O signals used with the Weld Error Recovery I/O interface must be configured. In a MultiMove system, each welding robot will have its own Weld Error recovery I/O interface with separate I/O signals. The end user can specify his own signal names for each welding robot in the system parameters (topic *Process*). To simplify this document, the signal names will here be described as signalname_x.

For example: diWER_Ack_X, where x specifies the welding robot number. The I/O interface will be activated if all the signals for each welding robot are defined in the system, otherwise the I/O interface will be disabled. See *Configuring Weld Error Recovery on page 61*.

Weld Error Recovery I/O Interface signal definition (X represents robot number 1-4).

Signal common name	Signal definition name	Description
Dialog Acknowledge	diWER_Ack_X	Makes it possible to acknowledge a weld error using the here specified digital input signal. Digital Input

Active Dialog Type	goWER_Dialog_X	Indicates to a remote device which Weld Error Recovery prompt is active.
		Valid output data range: 0-6
		Recommended Group Output size: 7 bits
		0 No Active Dialog
		  1 Get Error Action
		  2 Recovery Menu
		  3 ServiceRoutine not found
		  4 Moved from error point
		  5 Moved from break point
		  6 Skip forward

Dialog Active	doWER_Dialog_X	Indicates to a remote device that a dialog is active and awaiting a response. Digital Output
Escape Possible	doWER_EscapeOK_X	Indicates to a remote device that a valid Escape path is available. Digital Output

Continues on next page

Signal common name	Signal definition name	Description
Response	giWER_Response_X	Allows the remote device to communicate a response. The context of the response is dictated by the active dialog type.
		Valid input data range: 1-5
		Group Input 3 bits
		Active dialog type 1:
		1 Abort
		2 Move Out
		3 Escape
		4 Recovery Menu
		Active dialog type 2:
		1 Abort
		2 Skip Forward
		3 Skip Seam
		4 Skip Part
		5 Resume
		Active dialog type 3:
		• Skip forward distance
Error Type	goWER_ErrType_X	Indicates to the remote device the arc error type.
		Valid output data range:
		0-12
		0 = No active error type
		Group Output 4 bit
Error Number	goWER_ErrNum_X	Indicates to the remote device the specific arc error number.
		Valid output data range:
		0-102
		0 = No active error type
		Group Output 7 bit

Sequence

The I/O sequence is as follows:

1 An arc error occurs triggering a Weld Error Recovery prompt to be displayed. Weld Error Recovery will set *doWER_Dialog_X* high to indicate an active prompt. Weld Error Recovery will also set *goWER_Dialog_X* to indicate the type of prompt. If the prompt is an error type, an error type and number will be supplied on group outputs *goWER_ErrType_X* and *goWER_ErrNum_X*.

2 The remote device interprets the information. If the dialog prompt type requires a numeric response, the remote device supplies the value on *giWER_Response_X*.

3 The remote device acknowledges the prompt by pulsing the *diWER_Ack_X* signal. Weld Error Recovery responds by closing the prompt on the FlexPendant. Weld Error Recovery allows the *diWER_Ack_X* signal to stay high for up to 3 seconds. If the signal is left on, a warning will be issued.

The Weld Error Recovery I/O interface will be inoperable until the *diWER_Ack_X* signal is reset.

Dialog types

There are six possible dialog prompts from Weld Error Recovery. These are:

1 Get Error Action - Choose Abort, MoveOut, Escape, or Resume.

2 RecoveryMenu - Resume type.

3 ServiceRoutine Not Found - Service routine specified in RecoveryPosSet can't be located.

4 Moved from Error Location - Warning that robot will move slowly back to the error location.

5 Moved from Breakpoint - Warning that robot will move slowly back to the Breakpoint.

6 Skip Forward Distance – Number entry screen prompt for distance value.

When one of the six prompts is active, the digital output *doWER_Dialog_X* will be high. Some of the prompts require a numeric response from *giWER_Response_X* followed by an acknowledgment from *diWER_Ack_X*. Others simply require an acknowledgment from *diWER_Ack_X*.

Dialog Type - Get Error Action

If *goWER_Dialog_X* is set to 1, the **Get Error Action** dialog is active. This is the first dialog that appears after an arc error occurs.

en1200000691
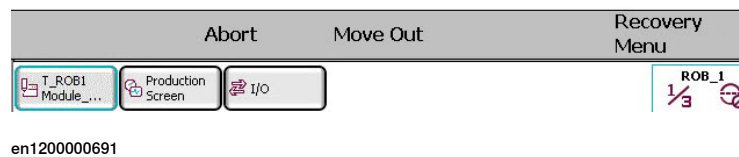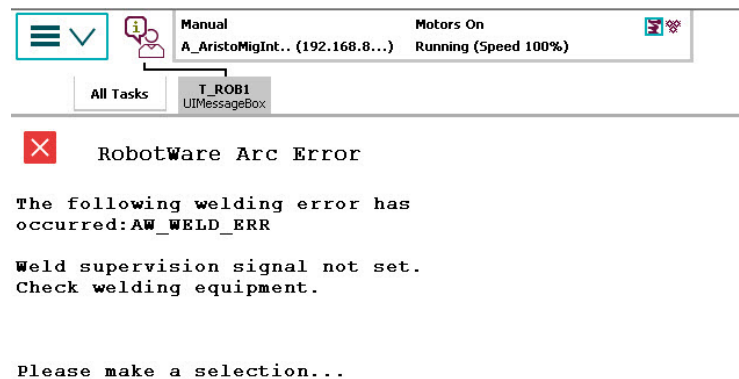
The Error Type will be sent on *goWER_ErrType_X*. The following is a list of possible error types from arc.

Arc ERRNO	Description	ErrType
AW_START_ERR	Error during the start of the process	1
AW_IGNI_ERR	Error during the ignition phase	2
AW_WELD_ERR	Error during the main weld phase	3
AW_EQIP_ERR	Equipment error	4
AW_WIRE_ERR	Wire error	5

Continues on next page

5.6 Weld Error Recovery I/O interface
Continued

Arc ERRNO	Description	ErrType
AW_STOP_ERR	Process stop was commanded.	6
AW_TRACK_ERR	Tracking error	7
AW_TRACKSTA_ERR	Tracking error	8
AW_TRACKSTA_ERR	Tracking correction error	9
AW_USERSIG_ERR	User error	10
AW_WDM_STABSTOP	WDM stability out-out-of-range	11
AW_WDM_SIGNSTOP	WDM signature out-out-of-range	12

The information could be used to provide an appropriate description of the problem. The Error Number will be sent on *goWER_ErrNum_X*. The following is the list of possible specific errors.

Description	Arc ElogNumber	ErrNum
Gas supervision error	110401	1
Water supervision error	110402	2
ArcOK supervision error	110403	3
Voltage supervision error	110404	4
Current supervision error	110405	5
Wirefeed supervision error	110406	6
Wirestick supervision error at the start of the weld	110407	7
Arc Ignition supervision error	110408	8
SchedStrobe supervision error	110409	9
SchedBusy supervision error	110410	10
Process Stop supervision error	110411	11
Arc Fill supervision error	110412	12
Torch supervision error	110413	13
WeldOK supervision error	110414	14
Arc timeout supervision error	110415	15
WeldOk supervision error	110416	16
Gas exec supervision error	110421	21
Water exec supervision error	110422	22
Arc exec supervision error	110423	23
Voltage exec supervision error	110424	24
Current exec supervision error	110425	25
Wirefeed exec supervision error	110426	26
Process Stop supervision error	110427	27
Torch exec supervision error	110428	28
Arc ignition supervision error	110429	29
Arc Fill supervision error	110430	30

Continues on next page

Description	Arc ElogNumber	ErrNum
WeldOK supervision error	110431	31
Arc ignition supervision error	110432	32
Arc Fill supervision error	110433	33
User sig1 supervision error	110435	35
User sig2 supervision error	110436	36
User sig3 supervision error	110437	37
User sig4 supervision error	110438	38
User sig5 supervision error	110439	39
User sig1 supervision info	110440	40
User sig2 supervision info	110441	41
User sig3 supervision info	110442	42
User sig4 supervision info	110443	43
User sig5 supervision info	110444	44
Gas supervision info	110445	45
Water supervision info	110446	46
Arc supervision info	110447	47
Voltage supervision info	110448	48
Current supervision info	110449	49
Wirefeed supervision info	110450	50
Torch supervision info	110451	51
Track supervision error	110500	100
Track start error	110501	101
Track correction error	110502	102
Wirestick supervision error at the end of the weld	110508	108

The information could be used to provide an appropriate description of the problem. The **Get Error Action** dialog prompt has several possible responses that the remote device may issue through *giWER_Response_X*, such as the following:

1 - Abort

2 - Move Out

3 - Escape (if valid)

4 - **Resume** or **Recovery Menu** depending on default behavior setting

For example, if the remote device wants the system to perform a *Move Out* action, it should supply 2 to *giWER_Response_X*, followed by pulsing *diWER_Ack_X*.

Escape behavior

The escape feature is not always available. It is only available when a path has been stored using `RecoveryPosSet`. The signal *doWER_EscapeOK_X* will be high if a valid escape path is available. Otherwise it will be low. If an escape command is issued by the remote device while *doWER_EscapeOK_X* is low, an error message will appear and the request will be Move Out instead of Escape.

Continues on next page

Dialog type - RecoveryMenu

The RecoveryMenu dialog ordinarily appears after the **Get Error Action** dialog. It provides the user a set of choices for restarting production. These areas follows:

1 Abort – Kills the process allows error to propagate leading to an execution stop.

2 Skip Forward – Allows the user to skip forward a short distance and resume welding.

3 Skip Seam – Jump ahead to the next seam and resume welding.

4 Skip Part – Do not start welding until the beginning of the next part cycle.

5 Resume – Resume welding at the error location.
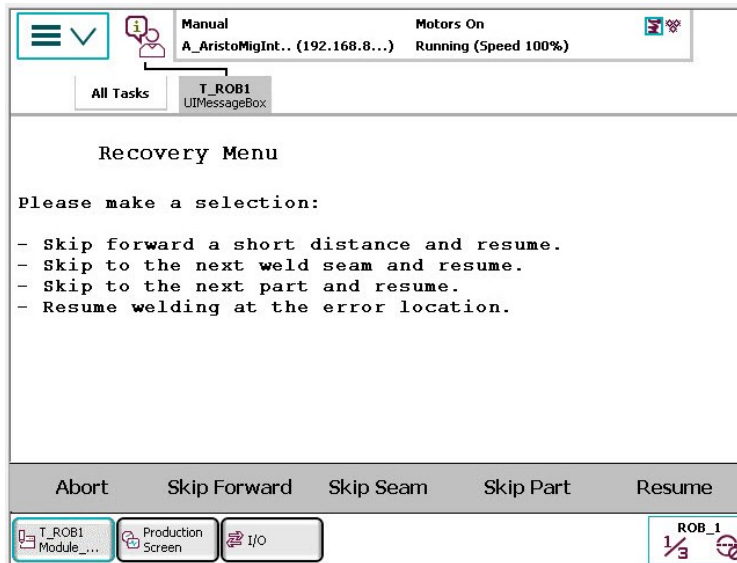
en1200000692

When the RecoveryMenu is active, the signal *doWER_Dialog_X* will be high and *goWER_Dialog_X* will be set to 2. The remote device may respond to the dialog by setting *giWER_Response_X* to a value from the list above, followed by pulsing *diWER_Break_X*.

Dialog type - Service Routine Not Found

This is an error message resulting from an invalid ServiceRoutine specified in a `RecoveryPosSet` instruction.
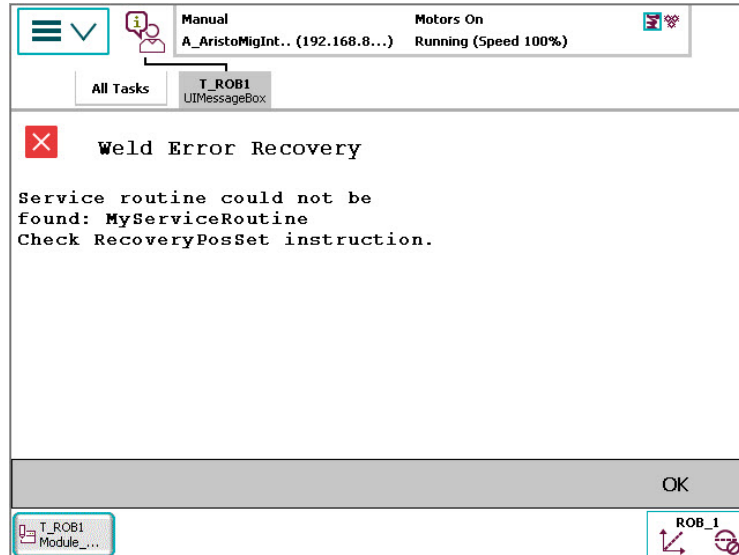
en1200000696

When this dialog is active, the signal *doWER_Dialog_X* will be high and *goWER_Dialog_X* will be set to 3. It needs only to be acknowledged by pulsing *diWER_Break_X*.

Dialog type - Moved from Error Location

This is an error message resulting from jogging the robot away from the error location.
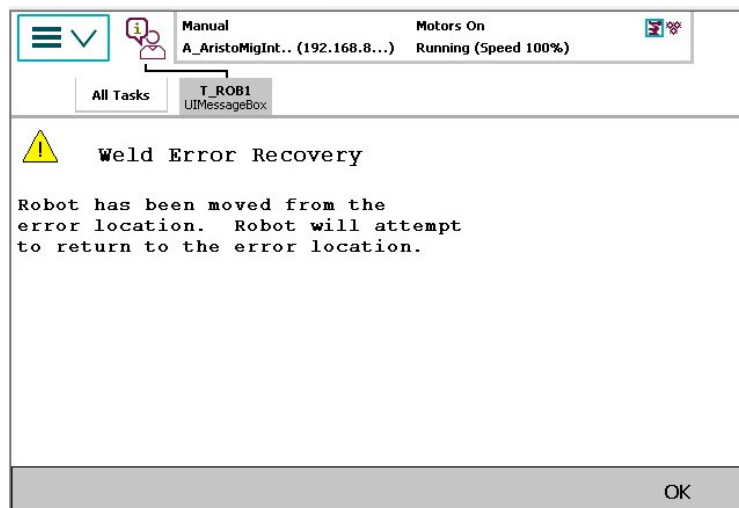
en1200000701

When this dialog is active, the signal *doWER_Dialog_X* will be high and *goWER_Dialog_X* will be set to 4. It needs only to be acknowledged by pulsing *diWER_Break_X*. The system will attempt to return to the error location by making a slow move towards the target.

Dialog type - Moved from Breakpoint

This is an error message resulting from a ServiceRoutine failing to return the robot to the breakpoint.
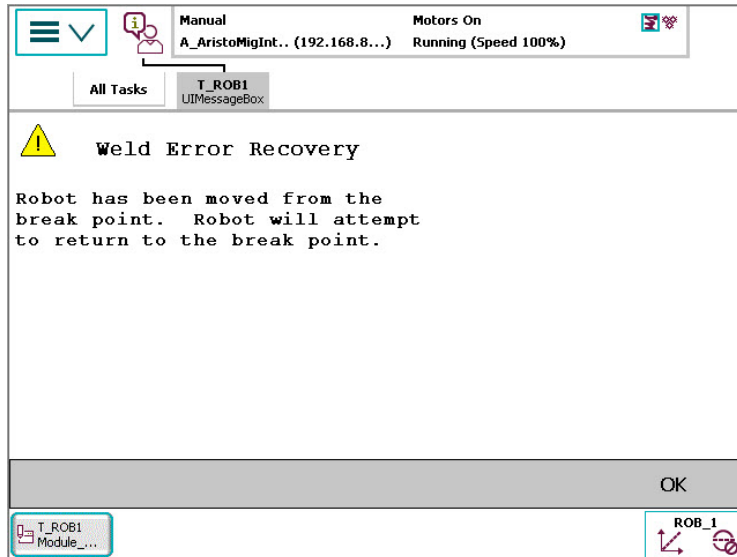
en1200000697

When this dialog is active, the signal *doWER_Dialog_X* will be high and *goWER_Dialog_X* will be set to 5. It needs only to be acknowledged by pulsing *diWER_Ack_X*. The system will attempt to return to the breakpoint location by making a slow move towards the target.

Dialog type - Skip Forward Distance

When Skip-Forward is selected from the RecoveryMenu, the user is prompted to enter a skip-forward distance.
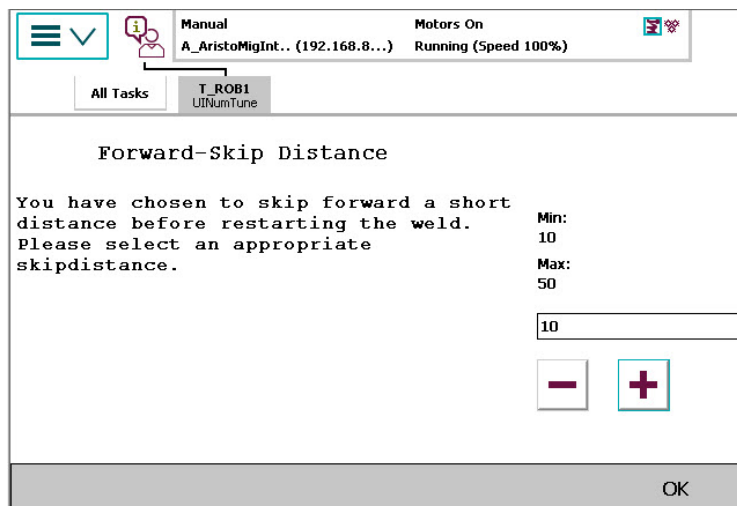
en1200000695

When this dialog is active, the signal *doWER_Dialog_X* will be high and *goWER_Dialog_X* will be set to 6. The remote device must supply a distance using the *giWER_Response_X* signal. The value should be supplied in centimeters. So,

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

to skip 2 cm, supply 2 to the group. Decimal values are not supported. Pulse *diWER_Ack_X* to send the command.

Dialog Selection Masking

The selections available in the *Get Error Action* and *RecoveryMenu* dialog prompts presented on the FlexPendant are configurable in the system parameters (topic *Process*). The remote device will be unaware of these settings. Selections provided in the remote device will not be affected by the configuration specified in the system parameters.

MultiMove considerations

No special provisions are necessary for MultiMove implementations. These considerations are already handled by Weld Error Recovery.

See .

5.7 Configure weld error recovery I/O Interface

Description

Arc Error Handler I/O configures the Weld Error Recovery I/O part of Weld Error Recovery feature in RobotWare Arc.

The Configuration parameters can be found in RobotStudio in the **Configuration Editor**, topic *Process*, type *Arc Error Handler I/O*.

Examples

The default configuration has the following definition.

en1200000702

Parameters

Parameter	Description	Data Type
Name	The name of the instance ARC_ERR_HNDL_IO. Must be (T_ROB1-T_ROB4)	typeStringNormal
goWER_Dialog	The signal name for Active Dialog Type.	go
doWER_Dialog	The signal name for Dialog Active.	do
doWER_EscapeOK	The signal name for Escape Possible.	do
diWER_Ack	The signal name for Prompt Acknowledge.	di

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

Parameter	Description	Data Type
giWER_Response	The signal name for Response.	gi
goWER_ErrType	The signal name for Error Type.	go
goWER_ErrNum	The signal name for Error Number.	go

Related information

Information	Described in
Set the recovery position	*RecoveryPosSet - Set the recovery position on page 166*
Reset the recovery position	*RecoveryPosReset - Reset the recovery position on page 169*
Display the recovery menu	*RecoveryMenu - Display the recovery menu on page 162*
Configure the recovery menu	*Configure the recovery menu on page 63*

5.8 Configure User defined error handling

Description

Arc Error Handler Properties configures the Weld Error Recovery, user defined part of Weld Error Recovery feature in RobotWare Arc.

The configuration parameters can be found in RobotStudio in the **Configuration Editor**, topic *Process*, type *Arc Error Handler Properties*.

Examples

The default configuration has the following definition.
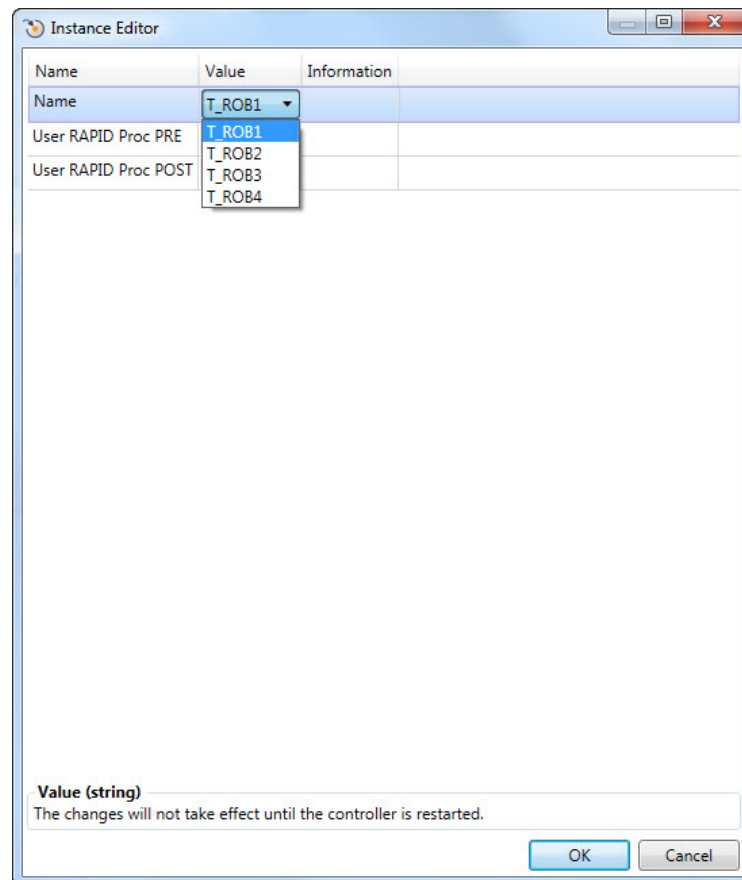
en1200000703

Parameters

Parameter	Description	Data Type
Name	The name of the instance ARC_ERR_HNDL_PROP. Must be (T_ROB1-T_ROB4)	typeStringNormal
Userproc_pre	The name of the RAPID procedure to be executed before the Weld Error Recovery menu appears.	typeStringRAPID
Userproc_post	The name of the RAPID procedure to be executed after the Weld Error Recovery menu has appeared, when the selection has been made in the menu.	typeStringRAPID

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

Related information

Information	Described in
Set the recovery position	*RecoveryPosSet - Set the recovery position on page 166*
Reset the recovery position	*RecoveryPosReset - Reset the recovery position on page 169*
Display the recovery menu	*RecoveryMenu - Display the recovery menu on page 162*
Configure the recovery menu	*Configure the recovery menu on page 63*

5.9 User defined error handling

Description

There is a possibility for the user to configure and run user defined RAPID procedures before and after the Weld Error Recovery dialogs are presented on the FlexPendant. This can for example be used to set specific I/O signals to an external PLC. The procedures are executed on StorePath level, so any motion instructions executed in the procedure will not destroy the original path.

Examples

In the following example, the system is configured to run the RAPID procedures `MyPreProc` and `MyPostProc`.
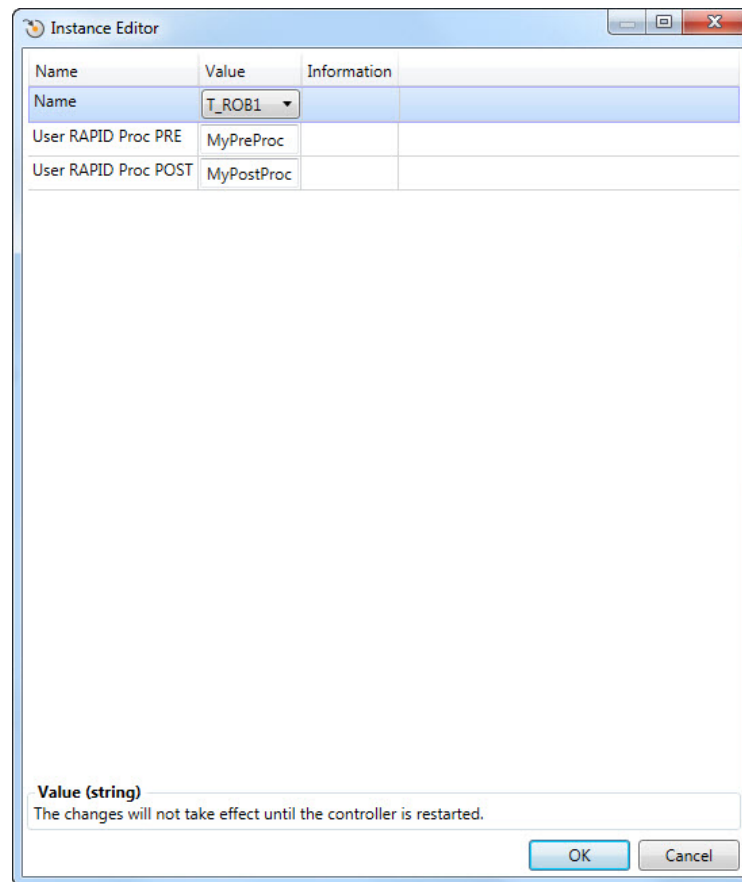
en1200000704

In the procedure `MyPreProc`, the elog number for the process error is retrieved via the Weld Error Recovery I/O interface group output signal *goWER_ErrNum_1*. The name of the current robtarget is retrieved via the RAPID string variable `stArcToPoint`.

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

In the procedure `MyPostProc`, the selected resume type in the Recovery Menu is retrieved via the RAPID variable `nAEResumeType`.

Program example

```
MODULE ErrorHandling
  PROC MyPreProc()
    VAR num nErrNo;
    VAR string stJointName;

    ! Get Errornumber
    nErrNo:=GetArcErrNo();
    ! Get Jointnumber
    stJointName:=GetJointNumber();

    UIMsgBox \Header:="MyPreProc T_ROB1", "Failing robtarget name:
          "+stJointName\MsgLine2:="Arc Elog error:
          "+ValToStr(nErrNo);
  ENDPROC

  PROC MyPostProc()
    VAR string stBtnSelected;

    TEST nAEResumeType
    CASE RESUME_KILL:
      stBtnSelected:="Abort";
    CASE RESUME_SKIP_FWD:
      stBtnSelected:="Skip FWD";
    CASE RESUME_SKIP_SEAM:
      stBtnSelected:="Skip Seam";
    CASE RESUME_SKIP_PART:
      stBtnSelected:="Skip Part";
    CASE RESUME_SKIP_OFF:
      stBtnSelected:="Resume";
    ENDTEST

    UIMsgBox \Header:="MyPostProc T_ROB1", "Selected action:
          "+stBtnSelected;
  ENDPROC

  FUNC num GetArcErrNo()
    RETURN GOutput(goWER_ErrNum_1);
  ENDFUNC

  FUNC string GetJointNumber()
    RETURN stArcToPoint;
  ENDFUNC

ENDMODULE
```

This page is intentionally left blank

6 Weld Repair

6.1 Introduction

Background

When welding synchronized with two or more welding robots and a positioner, weld errors can sometimes occur in one of the welding robots. Since the motion and process is synchronized, the welding robot without weld error will also stop and perform error handling. Using the functionality *Weld Repair* will shut down the arc only in the failing robot while the others are continuing its weld. The faulty seam is automatically re-welded at the error location.

Functional description

If a part is welded in synchronized coordinated mode and a weld error is detected (for example, the *ArcEst* signal goes low for robot 1), the robot will continue its movement to the end of the weld and is not forced to stop by RobotWare Arc. The other robots are also continuing its weld. The weld error is detected internally and a fully automatic weld repair can be done after the robots reached the instruction `ArcLEnd`.

The robots are moving backwards on path (with the help of the path recorder) into a service position and do an automatic re-weld of the seam (move with welding blocked until the error position is reached). All robots can move at the same time as only one weld error has been detected. The number of automatic repair retries is configured in the system parameters (PROC.cfg). If an ignition error or weld error is detected during the repair phase a retry can be done. If all weld repairs are done the robots can continue with the next seam.

If multiple errors are detected (for example, *ArcEst* goes low for more then one robot on different locations within a seam) then the behavior is the same as described but with the exception that now only one robot at the time can do the re-weld. The other welding robot(s) are moving with blocked welding.

If for some reason the re-weld failed again than the robots are moving backwards on path into the service position. User code can be executed in service position, for example, to communicate with a PLC and inform the operator that the re-weld failed. It is possible to continue the program from here on RAPID level to have the possibility to move to the next seam.

It is also possible to display a message on the FlexPendant to interact with the operator to continue production.

Weld Repair limitations

- Each weld has to be defined in its own procedure.
- Only synchronized motions can be used in the procedure. No independent movements after a `SynchMoveOff` instruction.
- The correct procedure name must be provided in the instruction `SetWRProcName`.

- In a FlexPositioner setup, the optional argument `\FlexPositioner` must be used in the instruction `SetWRProcName` in the non-welding task/robot.
- `ArcMoveExtJ\Start` must be used to indicate start to corresponding `ArcLStart` instruction (synchronized).
- Program displacement with the instruction `PDispSet` is not supported.

Addition to RobotWare Arc

- Two new instructions are introduced:
-   - `RecoveryMenuWR`
    - `SetWRProcName`
- A new data type `advSeamData` is introduced
- A new optional argument `\Start` is introduced to all `ArcMoveX` instructions
- A new optional argument `\advData` is introduced to the `ArcXStart` instructions
- New PLC codes (active dialog types) for the weld error recovery I/O interface
-   - Active Dialog Type Value 7 – RecoveryMenuWR prompted
    - Active Dialog Type Value 8 – Weld Repair Menu prompted

6.2 Configuring Weld Repair

Basic procedure

All configuration of the weld repair functionality is done in the system parameters, in the topic *Process* (Proc.cfg).

Use this procedure to activate the weld repair functionality:

1 Define the parameter *Enabled* in *Arc Error Handler* as *TRUE*.

en1300000292

2 Define the parameter *Arc Repair Enabled* in *Arc Error Handler* as *TRUE*.

3 Define the signal *ArcEst* in *Arc Equipment Digital Inputs* as *MINOR*[i].

[i] For the *StdIOWelder* this can be done in *Arc Equipment Digital Inputs*. It might be in a different place for other *EquipmentClasses*.

The EIO interface can be configured but this is not mandatory. However the current status can be sent to a PLC to indicate the current system status.

Continues on next page

Arc Repair Properties

The behavior of the weld repair function is configured in the type *Arc Repair Properties*.

Parameter	Description
Number of repair retries	The number of repair retires that are done before the robots are moving into their service position. This parameter is only available for *Full Automatic Mode*.
Mode	The following modes can be selected: • *Full Automatic Mode* • *Semi Automatic Mode*

en1300000295

Arc Repair IO

The EIO interface that indicates the current weld repair status is configured in the type *Arc Repair IO*. The signals can be independently configured for each robot.
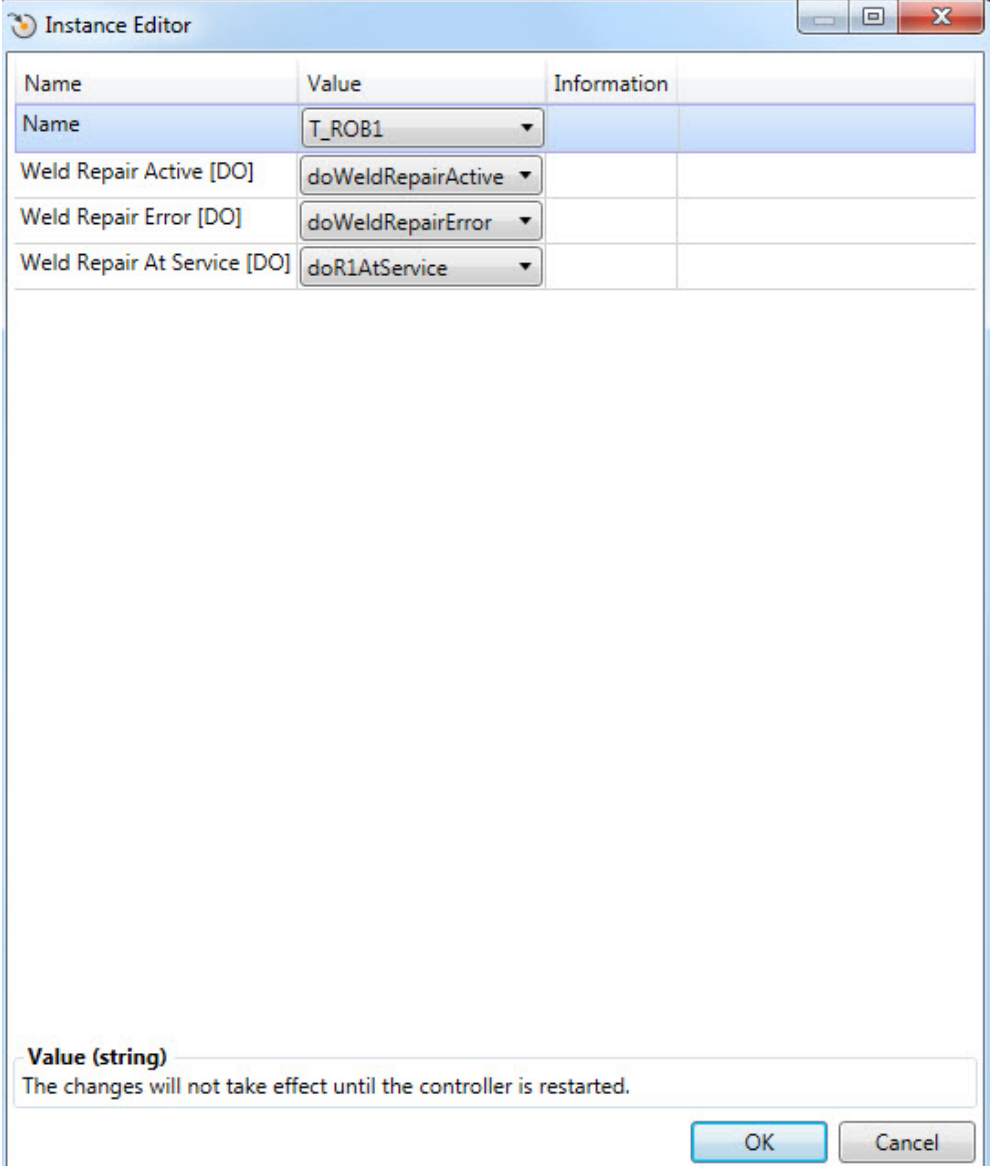
Parameter	Data type	Description
Weld Repair Active	Digital output	Indicates that the weld repair function is active. The signal is set/reset automatically from RobotWare Arc.
Weld Repair Error	Digital output	Indicates that a weld error occurred while the weld repair function was active. The signal is set/reset automatically from RobotWare Arc.
Weld Repair at Service	Digital output	Indicates that the service procedure connected to the instruction `RecoveryPosSet` is executed. The signal is set/reset automatically from RobotWare Arc.

en1300000293

Arc Equipment Digital Inputs

en1300000291

6.3 Best practice

Avoiding risk of collisions

When programming, place the arc welding instructions in a separate procedure to avoid the risk of collisions.

The following examples describe an arc welding application set up in two different ways, where one has a high risk of collisions for example between robots and fixtures, and the other has no risk of collision.

Example with high risk of collision

The following example program has a high risk of collision. The execution is described as follows.

1   The execution starts in the procedure `Part_1`.

2   A weld error occurs after that the welding has started.

3   The robot(s) move backwards on path to the instruction `ArcLStart`.

4   A recursive program call of `Part_1` is done as the PP is still in the instruction `ArcLEnd`.

    This is a high risk of a collision as the robots will move from `pArc10` to `p10`.

5   Then the weld routine continues execution to the end, the robots move to `p50`.

6   Now the PP moves back to the instruction `ArcLEnd` (end of recursive call).

    This is also a high risk for a collision as the robot moves from `p50` back to `p40`.

Regular execution | Recursive call within ArcLEnd

```
PROC Part_1()
  SetWRProcName "Part_1";
  ...
  ArcLStart pArc10 ...;
  ...
  ArcLEnd pArc50 ... ;
  ...
  MoveJ p50 ... ;
  ...
ENDPROC
```

```
PROC Part_1()
  SetWRProcName "Part_1";
  ...
  ArcLStart pArc10 ...;
  ...
  ArcLEnd pArc50 ... ;
  ...
  MoveJ p50 ... ;
  ...
ENDPROC
```

Error
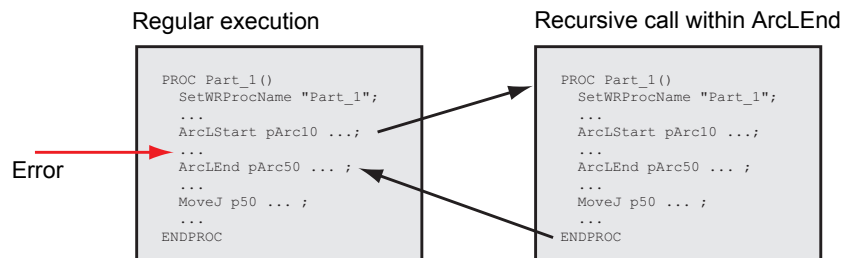
xx1300000328

Program example

```
PROC Part_1()
  SetWRProcName "Part_1";
  IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;
  MoveJ p10\ID:=10, vmax, z50, tWeldGun\WObj:=wobjStn1;
  MoveJ p20\ID:=20, vmax, z50, tWeldGun\WObj:=wobjStn1;
  MoveJ p30\ID:=30, vmax, z50, tWeldGun\WObj:=wobjStn1;
  RecoveryPosSet\ServRoutine:="mvToService";
  ArcLStart pArc10\ID:=40, vmax, sm1, wd1, fine,
      tWeldGun\WObj:=wobjStn1\SeamName:="Part_1";
  ArcL pArc20\ID:=50, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
  ArcL pArc30\ID:=60, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
```

Continues on next page

```
ArcLEnd pArc50\ID:=80, v100, sm1, wd1, fine,
     tWeldGun\WObj:=wobjStn1;
RecoveryPosReset;
MoveJ p40\ID:=90, vmax, z50, tWeldGun\WObj:=wobjStn1;
MoveJ p50\ID:=100, vmax, z50, tWeldGun\WObj:=wobjStn1;
SyncMoveOff sync002;
ENDPROC
```

Example with no risk of collision

The following example program has no risk of collisions because the program is divided into to modules. The execution is described as follows.

1 The execution starts in the procedure `Part_1`, which calls the procedure `Part_1_Seam1`.

2 A weld error occurs after that the welding has started, so the robot(s) move backwards on path to the instruction `ArcLStart` (position `pArc10`).

3 A recursive program call of `Part_1_Seam1` is done as the PP is still in the instruction `ArcLEnd`.

   There is no risk of a collision because the robot is already in the position `pArc10`.

4 The procedure `Part_1_Seam1` is executed and the PP then jumps back to the instruction `ArcLEnd`.

5 The PP leaves `Part_1_Seam1` and jumps back to the procedure `Part_1`.

   There is no risk of a collision as the departure positions have not been executed before.

Regular execution

```
PROC Part_1()
   ...
   MoveJ p10... ;
   ...
   Part_1_Seam1;
   ...
   MoveJ p50 ... ;
   ...
ENDPROC
```

Recursive call within ArcLEnd

```
PROC Part_1_Seam1()
   SetWRProcName "Part_1_Seam1";
   ...
   ArcLStart pArc10 ...;
   ...
   ArcLEnd pArc50 ... ;
   ...
ENDPROC
```

```
PROC Part_1_Seam1()
   SetWRProcName "Part_1_Seam1";
   ...
   ArcLStart pArc10 ...;
   ...
   ArcLEnd pArc50 ... ;
   ...
ENDPROC
```

Error

xx1300000329

Program example

```
PROC Part_1()
   IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;
```

Continues on next page

```
          MoveJ p10\ID:=10, vmax, z50, tWeldGun\WObj:=wobjStn1;
          MoveJ p20\ID:=20, vmax, z50, tWeldGun\WObj:=wobjStn1;
          MoveJ p30\ID:=30, vmax, z50, tWeldGun\WObj:=wobjStn1;
          Part_1_Seam1;
          MoveJ p40\ID:=90, vmax, z50, tWeldGun\WObj:=wobjStn1;
          MoveJ p50\ID:=100, vmax, z50, tWeldGun\WObj:=wobjStn1;
          SyncMoveOff sync002;
    ENDPROC
    PROC Part_1_Seam1()
      SetWRProcName "Part_1_Seam1";
      IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;
      RecoveryPosSet\ServRoutine:="mvToService";
      ArcLStart pArc10\ID:=40, vmax, sm1, wd1, fine,
            tWeldGun\WObj:=wobjStn1\SeamName:="Part_1_Seam1";
      ArcL pArc20\ID:=50, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
      ArcL pArc30\ID:=60, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
      ArcLEnd pArc50\ID:=80, v100, sm1, wd1, fine,
            tWeldGun\WObj:=wobjStn1;
      RecoveryPosReset;
    ENDPROC
```

6.4 Full Automatic Mode

Introduction

If a weld error occurs in full automatic mode (mode defined as *Full Automatic Mode*), all mechanical units are moving backwards on path with the help of the path recorder and the weld procedure is automatically re-executed in such a way that all robots and mechanical units are moving in synchronized mode and try to re-weld the faulty seam. Only the failing robot that had the weld error will re-strike the arc. All other robots are in blocked mode. If there was a weld error in more then one robot then the procedure is re-executed again until all weld errors are fixed for each robot. If another weld error occurs during the weld repair phase that cannot be handled with the configured number of retries then the robots will move backwards on path into a service position. Here a recovery menu can be displayed and an operator or a PLC can decide how to continue.

If no recovery menu is used in the service position the default restart behavior at the error position will be to resume (**Resume**). A recovery menu will be displayed at the error position if the configured number of repair retries is reached.

en1300000301

Program examples for *Full Automatic Mode*

The following examples describe scenarios of how the weld repair functionality reacts on ignition errors and weld errors in full automatic mode. All scenarios are based on synchronized welds and related to the program examples below.

> **ℹ Note**
>
> Make sure to use the correct name (`SetWRProcName`) in the welding sequence to inform *Robotware Arc* which weld procedure should be re-executed with the weld repair function. There might be a risk that the welding equipment, for example the welding gun, gets damaged if the wrong procedure name is specified.
> `ArcMoveExtJ\Start` must be used in the positioner's task to indicate start to corresponding `ArcLStart` instruction.

Example program for robot 1

Each weld is in a separate procedure to avoid collisions if the Weld Repair sequence is active.

```
MODULE R1_Part_A
  PROC PART_1_R1()
    IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;

    ! Approach positions towards the weld
    MoveJ p10\ID:=10, vmax, z50, tWeldGun\WObj:=wobjStn1;
    MoveJ p20\ID:=20, vmax, z50, tWeldGun\WObj:=wobjStn1;
    MoveJ p30\ID:=30, vmax, z50, tWeldGun\WObj:=wobjStn1;

    R1_Part1_Seam1;

    ! Depart positions
    MoveJ p40\ID:=90, vmax, z50, tWeldGun\WObj:=wobjStn1;
    MoveJ p50\ID:=100, vmax, z50, tWeldGun\WObj:=wobjStn1;
    MoveJ p60\ID:=110, vmax, z50, tWeldGun\WObj:=wobjStn1;

    SyncMoveOff sync002;
  ENDPROC
  PROC R1_Part1_Seam1()
    ! Inform RobotWare Arc which procedure shall be re-executed
    SetWRProcName "R1_Part1_Seam1";

    IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;

    ! Set Recovery Position, here it will be "p30" from example
        above
    ! Turn on PathRecorder
    RecoveryPosSet\ServRoutine:="mvToService";

    ArcLStart pArc10\ID:=40, vmax, sm1, wd1, fine,
        tWeldGun\WObj:=wobjStn1\SeamName:="R1_Part1_Seam1";
    ArcL pArc20\ID:=50, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
    ArcL pArc30\ID:=60, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
```

```
            ArcC pArc40, pArc50\ID:=70, v100, sm1, wd1, z1,
                tWeldGun\WObj:=wobjStn1;
            ArcLEnd pArc50\ID:=80, v100, sm1, wd1, fine,
                tWeldGun\WObj:=wobjStn1;


            ! Turn off PathRecorder
            RecoveryPosReset;
        ENDPROC
    ENDMODULE
```

Example program for robot 2

Each weld is in a separate procedure to avoid collisions if the Weld Repair sequence is active.

```
    MODULE R1_Part_A
      PROC PART_1_R2()
        IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;

        ! Approach positions towards the weld
        MoveJ p10\ID:=10, vmax, z50, tWeldGun\WObj:=wobjStn1;
        MoveJ p20\ID:=20, vmax, z50, tWeldGun\WObj:=wobjStn1;
        MoveJ p30\ID:=30, vmax, z50, tWeldGun\WObj:=wobjStn1;

        R2_Part1_Seam1;

        ! Departure positions
        MoveJ p40\ID:=90, vmax, z50, tWeldGun\WObj:=wobjStn1;
        MoveJ p50\ID:=100, vmax, z50,tWeldGun\WObj:=wobjStn1;
        MoveJ p60\ID:=110, vmax, z50,tWeldGun\WObj:=wobjStn1;

        SyncMoveOff sync002;
      ENDPROC
      PROC R2_Part1_Seam1()
        ! Define which procedure shall be re-executed
        SetWRProcName "R2_Part1_Seam1";

        IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;

        ! Set Recovery Position, here it will be "p30" from example
            above
        ! Turn on PathRecorder
        RecoveryPosSet\ServRoutine:="mvToService";

        ArcLStart pArc10\ID:=40, vmax, sm1, wd1, fine,
            tWeldGun\WObj:=wobjStn1\SeamName:="R2_Part1_Seam1";
        ArcL pArc20\ID:=50, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
        ArcL pArc30\ID:=60, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
        ArcC pArc40, pArc50\ID:=70, v100, sm1, wd1, z1,
            tWeldGun\WObj:=wobjStn1;
        ArcLEnd pArc60\ID:=80, v100, sm1, wd1, fine,
            tWeldGun\WObj:=wobjStn1;
```
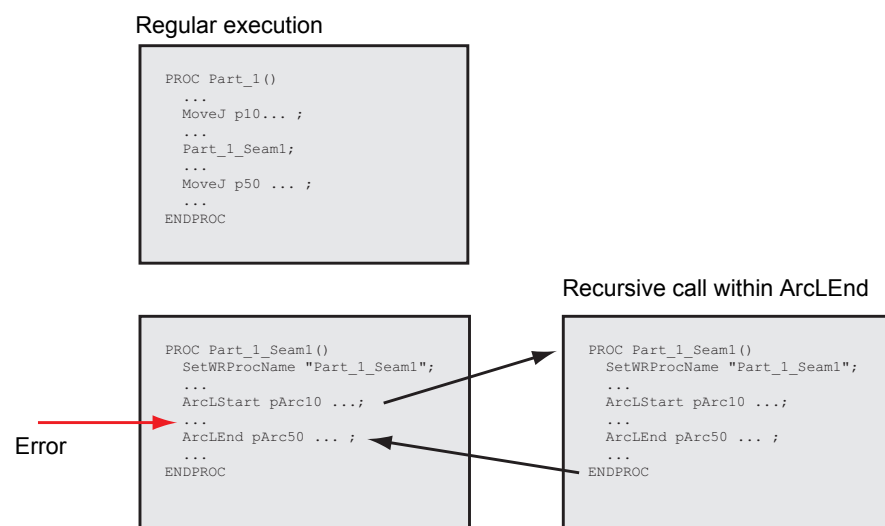
Continues on next page

```
                     ! Turn off PathRecorder
              RecoveryPosReset;
          ENDPROC
       ENDMODULE
```

Example program for STN1

Each weld is in a separate procedure to avoid collisions if the Weld Repair sequence is active.

```
MODULE STN1_Part_A
  PROC STN1_Part1()
    IF NOT IsMechUnitActive(STN1) ActUnit STN1;
    IF NOT IsSyncMoveOn() SyncMoveOn sync001, Alle_STN1;

    MoveExtJ p10\ID:=10,vmax,z50;
    MoveExtJ p20\ID:=20,vmax,z50;
    MoveExtJ p30\ID:=30,vmax,z50;

    STN1_Part1_Seam1;

    MoveExtJ p40\ID:=90,vmax,z50;
    MoveExtJ p50\ID:=100,vmax,z50;
    MoveExtJ p60\ID:=110,vmax,z50;

    SyncMoveOff sync002;
  ENDPROC
  PROC STN1_Part1_Seam1()
    ! Define which procedure shall be re-executed
    SetWRProcName "STN1_Part1_Seam1";

    IF NOT IsMechUnitActive(STN1) ActUnit STN1;
    IF NOT IsSyncMoveOn() SyncMoveOn sync001, Alle_STN1;

    ! Set Recovery Position, here it will be "p30" from example
         above
    ! Turn on PathRecorder
    RecoveryPosSet\ServRoutine:="mvToService";

    ! Optional argument "\Start" set to the corresponding ArcLStart
         instruction
    ArcMoveExtJ pArc10\ID:=40,vmax,fine\Start;
    ArcMoveExtJ pArc20\ID:=50,vrot50,z1;
    ArcMoveExtJ pArc30\ID:=60,vrot50,z1;
    ArcMoveExtJ pArc40\ID:=70,vrot50,z1;
    ArcMoveExtJ pArc50\ID:=80,vrot50,fine;

    ! Turn off PathRecorder
    RecoveryPosReset;
  ENDPROC
ENDMODULE
```

Example program for a service routine

```
PROC mvToService()
  VAR bool bTaskInSync:=FALSE;
  ! Check if in synchronized mode
  IF IsSyncMoveOn() bTaskInSync:=TRUE;

  GetCurrentPosition pEnter\Tool:= tWeldGun \Wobj:=wobj0;

  ! Suspend synchronization, robots are now in independent mode
  IF bTaskInSync SyncMoveSuspend;
  IF GetProcErr() THEN
    MoveJ pService, v500, fine, tWeldGun\WObj:=wobj0;
    RecoveryMenuWR;
  ENDIF
  ! Get synchronized again
  IF bTaskInSync SyncMoveResume;
  ! Move to saved position
  IF bTaskInSync THEN
    MoveJ pEnter\ID:=200, v500, fine, tWeldGun \WObj:=wobj0;
  ELSE
    MoveJ pEnter, v500, fine, tWeldGun \WObj:=wobj0;
  ENDIF
ENDPROC
```

Scenario 1: Ignition error at arc start instruction for any robot

The robots are moving to the instruction `ArcLStart` and the path recorder is turned on with the instruction `RecoveryPosSet`. A service routine is specified with the optional argument `\ServiceRoutine` that is used with the instruction `RecoveryPosSet`. Assuming that an ignition error occurs, for example for robot 1 and the configured number of retires is exceeded, then the robots will move backwards on path into their defined recovery position and the specified service routine (in this example `mvToService`) is executed from here.

The synchronization between the robots and positioner can be suspended so that individual movements, for example for gun cleaning can be done.

The function `GetProcError()` can be used to check which robot had a process error so that only the failing robot will move into the service position.

A recovery menu can be used so that an operator or a PLC can decide how to continue. A limited recovery menu (instruction `RecoveryMenuWR`) can also be used, but the buttons for that menu cannot be configured.

If the seam is not skipped by the operator then the Weld repair function will loop until it gets a stable arc at the instruction `ArcLStart`.
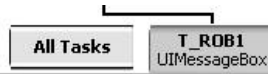
en1300000301

If no recovery menu is used in the service position the default restart behavior at the error position will be to resume (**Resume**). The limited recovery menu will be automatically presented at the error position if the configured number of repair retries failed.

Scenario 2: Weld error along the path for any robot

The robots are moving to the instruction `ArcLStart` and the path recorder is turned on with the instruction `RecoveryPosSet`. A service routine is specified with the optional argument `\ServiceRoutine` that is used with the instruction `RecoveryPosSet`. If a weld error is detected (for example the *ArcEst* signal goes low for robot 1) the robot will continue its movement to the end of the weld and is not forced to stop by *RobotWare Arc*. The other robot(s) are also continuing their welds. The weld error is detected internally and a fully automatic weld repair is done after the robots reached the instruction `ArcLEnd`. The robots move backwards on path to the instruction `ArcLStart` and the weld program is re-executed. All robots are moving in blocked mode to the error position of robot 1. Only robot 1 re-strikes the arc. (The restart distance is taken into account).

If multiple errors are detected (for example *ArcEst* goes low for more than one robot on different locations within a seam) then the behavior is the same as described but with the exception that now only one robot at the time can do the re-weld. The other welding robots are moving with blocked welding. The repair order cannot be changed and it is always done in the following order: first robot 1, second robot 2, third robot 3, and robot 4 will be last. For example if a weld error

is detected in robot 2 and later along the path for robot 1 then the weld repair for robot 1 will be done first.

If for some reason the re-weld fails again then the robots move backwards on path into the service position.

6.5 Semi Automatic Mode

Introduction

If a part is welded in synchronized coordinated mode (mode defined as *Semi Automatic Mode*) and a weld error is detected (for example the signal *ArcEst* goes low for robot 1), the robot will continue its movement to the end of the weld and is not forced to stop by *RobotWare Arc*. The other robots are also continuing their weld.

The weld error is detected internally and the robots will move backwards on path with help of path recorder into their defined service position once they reached the instruction `ArcLEnd`. No automatic weld repair is done in this mode, instead the recovery menu has to be used and the operator or the PLC has to decide how to continue. If for example *resume* is selected the robots are moving to the error position and here only the failing robots re-strikes the arc. The other robots are moving in blocked mode. If there was a weld error in more than one robot then the procedure is re-executed again until all weld errors are fixed for each robot. If another weld error occurs during the weld repair phase that cannot be handled within he configured number of retries, then the robots will again move backwards on path into a service position.

> **Note**
>
> If no recovery menu is used in the service position then the default restart behavior at the error position will be to resume. A recovery menu will be presented automatically at the error position if the configured number of repair retries is reached.

> **Note**
>
> The data type `AdvSeamData` cannot be declared as `TASK PERS` (*MultiMove* regulations), use `PERS` instead. See *advSeamData - Advanced seam data on page 172*.

> **Note**
>
> In synchronized mode, the optional argument `\Start` has to be used with the instruction `ArcMoveExtJ` to indicate the start for the corresponding `ArcLStart` instruction.

Example program

```
PERS AdvSeamData AdvSeamData1:=[[FALSE,0,2,1,10,10],[TRUE,0,0.5,5]];

PROC Weld_1()
  SetWRProcName "Weld_1";

  SyncMoveOn sync001, allTasks;
  MoveL p10\ID:=10, vmax, z10, tWeldGun;
```

```
RecoveryPosSet\ServRoutine:="mvToService";
ArcLStart p20\ID:=20, v100, sm1\AdvData:=
    AdvSeamData1,wd1,fine,tWeldGun;
ArcL p30\ID:=30, v100, sm1, wd1, z10, tWeldGun;
ArcLEnd p40\ID:=40, v100, sm1, wd1, fine, tWeldGun;
RecoveryPosReset;

MoveJ p50\ID:=50, vmax, z10, tWeldGun;
SyncMoveOff sync_testblech_2;
ENDPROC
```


en1300000296

en1300000297

Program examples for *Semi Automatic Mode*

The following scenarios describes how the weld repair functionality reacts on ignition errors and weld errors in semi automatic mode. All scenarios are based on synchronized welds.

Example program for robot 1

Each weld is in a separate procedure to avoid collisions if the Weld Repair sequence is active.

```
MODULE R1_Part_A
PERS AdvSeamData AdvSeamData1:=[[TRUE,0.5,1,2,10,10],[FALSE,0,1,5]];
  PROC PART_1_R1()
    IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;

    ! Approach positions towards the weld
    MoveJ p10\ID:=10, vmax, z50, tWeldGun\WObj:=wobjStn1;
    MoveJ p20\ID:=20, vmax, z50, tWeldGun\WObj:=wobjStn1;
    MoveJ p30\ID:=30, vmax, z50, tWeldGun\WObj:=wobjStn1;

    R1_Part1_Seam1;

    ! Departure positions
    MoveJ p40\ID:=90, vmax, z50, tWeldGun\WObj:=wobjStn1;
    MoveJ p50\ID:=100, vmax, z50, tWeldGun\WObj:=wobjStn1;
    MoveJ p60\ID:=110, vmax, z50, tWeldGun\WObj:=wobjStn1;

    SyncMoveOff sync002;
```

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

```
        ENDPROC
        PROC R1_Part1_Seam1()
          ! Define which procedure shall be re-executed
          SetWRProcName "R1_Part1_Seam1";

          IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;

          ! Set Recovery Position, here it will be "p30" from example
                above
          ! Turn on PathRecorder
          RecoveryPosSet\ServRoutine:="mvToService";

          ArcLStart pArc10\ID:=40, vmax, sm1\AdvData:=AdvSeamData1, wd1,
                fine, tWeldGun\WObj:=wobjStn1\SeamName:="R1_Part1_Seam1";
          ArcL pArc20\ID:=50, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
          ArcL pArc30\ID:=60, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
          ArcC pArc40, pArc50\ID:=70, v100, sm1, wd1, z1,
                tWeldGun\WObj:=wobjStn1;
          ArcLEnd pArc50\ID:=80, v100, sm1, wd1, fine,
                tWeldGun\WObj:=wobjStn1;

          ! Turn off PathRecorder
          RecoveryPosReset;
        ENDPROC
      ENDMODULE
```

Example program for robot 2

Each weld is in a separate procedure to avoid collisions if the Weld Repair sequence is active.

```
      MODULE R1_Part_A
      PERS AdvSeamData AdvSeamData1:=[[TRUE,0.5,1,2,10,10],[FALSE,0,1,5]];
        PROC R2_PART1()
          IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;

          ! Approach positions towards the weld
          MoveJ p10\ID:=10, vmax, z50, tWeldGun\WObj:=wobjStn1;
          MoveJ p20\ID:=20, vmax, z50, tWeldGun\WObj:=wobjStn1;
          MoveJ p30\ID:=30, vmax, z50, tWeldGun\WObj:=wobjStn1;

          R2_Part1_Seam1;

          ! Departure positions
          MoveJ p40\ID:=90, vmax, z50, tWeldGun\WObj:=wobjStn1;
          MoveJ p50\ID:=100, vmax, z50,tWeldGun\WObj:=wobjStn1;
          MoveJ p60\ID:=110, vmax, z50,tWeldGun\WObj:=wobjStn1;

          SyncMoveOff sync002;
        ENDPROC
        PROC R2_Part1_Seam1()
          ! Define which procedure shall be re-executed
          SetWRProcName "R2_Part1_Seam1";
```

Continues on next page

6 Weld Repair

```
                    IF NOT IsSyncMoveOn() SyncMoveOn sync001, syncR1R2STN1;

                    ! Set Recovery Position, here it will be "p30" from example
                          above
                    ! Turn on PathRecorder
                    RecoveryPosSet\ServRoutine:="mvToService";

                    ArcLStart pArc10\ID:=40, vmax, sm1\AdvData:=AdvSeamData1, wd1,
                          fine, tWeldGun\WObj:=wobjStn1\SeamName:="R2_Part1_Seam1";
                    ArcL pArc20\ID:=50, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
                    ArcL pArc30\ID:=60, v100, sm1, wd1, z1, tWeldGun\WObj:=wobjStn1;
                    ArcC pArc40, pArc50\ID:=70, v100, sm1, wd1, z1,
                          tWeldGun\WObj:=wobjStn1;
                    ArcLEnd pArc60\ID:=80, v100, sm1, wd1, fine,
                          tWeldGun\WObj:=wobjStn1;

                    ! Turn off PathRecorder
                    RecoveryPosReset;
                ENDPROC
            ENDMODULE
```

Example program for STN1

Each weld is in a separate procedure to avoid collisions if the Weld Repair sequence is active.

```
        MODULE STN1_Part_A
          PROC STN1_Part1()
            IF NOT IsMechUnitActive(STN1) ActUnit STN1;
            IF NOT IsSyncMoveOn() SyncMoveOn sync001, Alle_STN1;

            MoveExtJ p10\ID:=10,vmax,z50;
            MoveExtJ p20\ID:=20,vmax,z50;
            MoveExtJ p30\ID:=30,vmax,z50;

            STN1_Part1_Seam1;

            MoveExtJ p40\ID:=90,vmax,z50;
            MoveExtJ p50\ID:=100,vmax,z50;
            MoveExtJ p60\ID:=110,vmax,z50;

            SyncMoveOff sync002;
          ENDPROC
          PROC STN1_Part1_Seam1()
            ! Define which procedure shall be re-executed
            SetWRProcName "STN1_Part1_Seam1";

            IF NOT IsMechUnitActive(STN1) ActUnit STN1;
            IF NOT IsSyncMoveOn() SyncMoveOn sync001, Alle_STN1;

            ! Set Recovery Position, here it will be "p30" from example
                  above
```

```
        ! Turn on PathRecorder
        RecoveryPosSet\ServRoutine:="mvToService";

        ! Optional argument "\Start" set to the corresponding ArcLStart
            instruction
        ArcMoveExtJ pArc10\ID:=40,vmax,fine\Start;
        ArcMoveExtJ pArc20\ID:=50,vrot50,z1;
        ArcMoveExtJ pArc30\ID:=60,vrot50,z1;
        ArcMoveExtJ pArc40\ID:=70,vrot50,z1;
        ArcMoveExtJ pArc50\ID:=80,vrot50,fine;

        ! Turn off PathRecorder
        RecoveryPosReset;
    ENDPROC
ENDMODULE
```

Example program for a service routine

```
PROC mvToService()
  VAR bool bTaskInSync:=FALSE;
  ! Check if in synchronized mode
  IF IsSyncMoveOn() bTaskInSync:=TRUE;

  GetCurrentPosition pEnter\Tool:= tWeldGun \Wobj:=wobj0;

  ! Suspend synchronization, robots are now in independent mode
  IF bTaskInSync SyncMoveSuspend;
  IF GetProcErr() THEN
    MoveJ pService, v500, fine, tWeldGun\WObj:=wobj0;
    RecoveryMenuWR;
  ENDIF
  ! Get synchronized again
  IF bTaskInSync SyncMoveResume;
  ! Move to saved position
  IF bTaskInSync THEN
    MoveJ pEnter\ID:=200, v500, fine, tWeldGun \WObj:=wobj0;
  ELSE
    MoveJ pEnter, v500, fine, tWeldGun \WObj:=wobj0;
  ENDIF
ENDPROC
```

Scenario 1: Ignition error at arc start instruction for any robot using `advSeamData`

The robots are moving to the instruction `ArcLStart` and the path recorder is turned on with the instruction `RecoveryPosSet`. A service routine is specified with the optional argument `\ServiceRoutine` that is used with the instruction `RecoveryPosSet`. Assuming that an ignition error occurs, for example for robot 1 and the configured number of retires is exceeded, then the robots will move backwards on path into their defined recovery position and the specified service routine (in this example `mvToService`) is executed from here.

The synchronization between the robots and positioner can be suspended so that individual movements, for example for gun cleaning can be done.

Continues on next page

The function `GetProcError()` can be used to check which robot had a process error so that only the failing robot will move into the service position.

A recovery menu can be used so that an operator or a PLC can decide how to continue.

en1300000300

If no recovery menu is used in the service position the default restart behavior at the error position will be to resume.

For more information about the instructions see *RecoveryPosSet - Set the recovery position on page 166*, *RecoveryPosReset - Reset the recovery position on page 169*, and *RecoveryMenu - Display the recovery menu on page 162*.

Since the regular ignition failed (one strike and two re-strikes) scrape start will now be active when the robots start from the service position. At `ArcLStart` they try to get a stable arc with the defined scrape parameters in `advSeamData`. If it fails again, then the robots will move back to the service position. They loop until the arc gets stable. Scrape start is now always active until the arc is stable. The system behaves in the same way if wire retract is active. (Wire will be retracted the specified time in the instruction `ArcLStart`.)

Scenario 2: Weld error along the path for any robot using `advSeamData`

The robots are moving to the instruction `ArcLStart` and the path recorder is turned on with the instruction `RecoveryPosSet`. A service routine is specified with the optional argument `\ServiceRoutine` that is used with the instruction `RecoveryPosSet`. If a weld error is detected (for example the *ArcEst* signal goes low for robot 1) the robot will continue its movement to the end of the weld and is not forced to stop by *RobotWare Arc*. The other robot(s) are also continuing their welds. The weld error is detected internally and a fully automatic weld repair is done after the robots reached the instruction `ArcLEnd`.

A recovery menu is displayed and the operator or a PLC can select how to continue. If for example **Resume** is selected, then the robots move to the error position and

here only the failing robot re-strikes the arc. The other robots are moving in blocked mode. If there was a weld error for more than one robot then the procedure is re-executed again until all weld errors are fixed for each robot. If another weld error or ignition error occurs during the weld repair phase that cannot be handled within the configured number of retries, then the robots will try to strike the arc again with the different behavior as configured in `advSeamData` (wire retract and scrape start). If this additional strikes failing as well then the robots are move backwards on path into their service position.

If the configured number of allowed weld errors exceeded then the robots will stop at the error position and the following menu is presented automatically.

```
┌──────┐
│      │
All Tasks    T_ROB1
             UIMessageBox
─────────────────────────────────────────

    Weld Repair Menu

The maximum number of weld errors (1)
was reached on the seam Naht_1

Please make a selection:
- Skip to the next weld seam and resume.
- Resume welding at the error location.




                              Skip Seam      Resume
─────────────────────────────────────────
```

en1300000302

This page is intentionally left blank

7 RAPID reference

7.1 Instructions

7.1.1 ArcC, ArcC1, ArcC2 - Arc welding with circular motion

Usage

`ArcC` is used to weld along a circular path. The instruction controls and monitors the entire welding process as follows:

- The tool center point (TCP) is moved in a circle to the specified destination position.
- All phases of the welding process, such as the start and end phases, are controlled.
- The welding process is monitored continuously.

The only difference between `ArcC`, `ArcC1` and `ArcC2` is that they are connected to different Arc Weld systems configured in the system parameters. Although `ArcC` is used in the examples, `ArcC1` or `ArcC2` could equally well be used.

If a weld seam is programmed without an `ArcXStart` instruction, `ArcLStart` or `ArcCStart`, there is an error message. When the error message has been acknowledged, the program execution in that task is stopped.

Example

```
MoveJ ...
ArcLStart p1, v100, seam1, weld5, fine, gun1;
ArcC p2, p3, v100, seam1, weld5, fine, gun1;
ArcCEnd p4, p5, v100, seam1, weld5, fine, gun1;
MoveJ ...
```

This welds a circular seam between points p1 and p3 (via point p2), and a circular seam between points p3 and p5 (via point p4), as illustrated in the following figure.
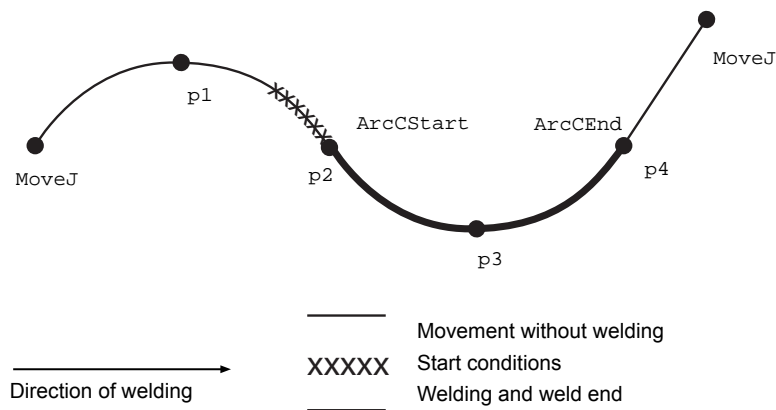
xx1500001031

On the way to `p1`, preparations for the weld start, such as gas preflowing, are carried out. The process and the actual weld movement then start at position `p1`

Continues on next page

7.1.1 ArcC, ArcC1, ArcC2 - Arc welding with circular motion
Continued

and end at `p5`. The start and end processes are determined by `seam1` and the welding process by `weld5`.

Arguments

```
ArcC CirPoint ToPoint [\ID] Speed Seam Weld [\Weave] Zone Tool
     [\WObj] [\Corr] [\Track] [\Time] [\T1] [\T2] [\T3] [\T4] [\T5]
     [\T6] [\T7] [\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). The position of the external axes are not used.

ToPoint

Data type: `robtarget`

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

[\ID]

Synchronization id

Data type: `identno`

The argument [\ID] is mandatory in 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` in the following cases:

- When the `ArcLStart` instruction is used.
- When the program is run instruction-by-instruction (no welding).

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. In the figure below, the speed is defined by the `Speed` argument in the respective instructions.

Direction of welding

——— Movement without welding

XXXXX Start preparations

▬▬▬ Movement with welding

xx1200000710

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. Normally the same seam data is used in all instructions of a seam.

Weld

Data type: `welddata`

Weld data describes the weld phase of the welding process.

Weld data is often changed from one instruction to the next along a seam.

[\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 not specifying any `weavedata` in the instruction.

Zone

Data type: `zonedata`

Zone data defines how close the axes must be to the programmed position before they can start moving towards the next position. In the case of a fly-by point, a corner path is generated past that position. In the case of a stop point (fine), the

movement is interrupted until all axes have reached the programmed point. A stop point is always generated automatically at the start position of a weld and at a controlled weld end position. Fly-by points, such as **z10**, should be used for all other weld positions. Weld data changes over to the next arc welding instruction at the center point of the corner path.

`Tool`

Data type: `tooldata`

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

`[ \WObj ]`

Work Object

Data type: `wobjdata`

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

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

`[ \Corr ]`

Correction

Data type: `switch`

Correction data written to a corrections entry by the instruction `CorrWrite` will be added to the path and destination position if this argument is present.

The RobotWare option *Path Offset* is required when using this argument.

`[\Track]`

Data type: `trackdata`

Trackdata is used and is only applicable when the system is configured for seam tracking with a serial weld guide system or with a Laser Tracker system. Seam tracking is activated when this argument is included in the `ArcC` instruction, but deactivated if it is omitted. The optional `trackdata` argument must be used during the whole weldseam, that is, from the `ArcXStart` to the `ArcXEnd` instruction.

> **ℹ️ Note**
>
> Seam tracking requires one of the following options: or options.
> - Tracking Interface
> - *WeldGuide*
> - *Optical tracking*

`[\Time]`

Data type: `num`

This argument is used to specify the total time in seconds during which the robot and additional axes move. It is then substituted for the corresponding speed data.

[\T1] [\T2] [\T3] [\T4] [\T5] [\T6] [\T7]

        Data type: triggdata

        Variables that refer to trigger conditions and trigger activity, defined earlier in the program using the instructions TriggRampAO, TriggIO, TriggEquip or TriggInt.

[\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 - Moves the robot linearly*.

Program execution

Controlling process equipment

        The process equipment is controlled by the robot in such a way that the entire process and each of its phases are coordinated with the robot's movements.

Motion

        Robot and additional axes are moved to the destination position as follows:

- The TCP of the tool is moved circularly at a constant programmed speed. When coordinated axes are used, the robot and the coordinated axes are moved simultaneously, resulting in the programmed path and speed for the TCP relative to the work object.
- The tool is reorientated at even intervals throughout the entire course.
- Uncoordinated additional axes are executed at a constant speed which means that they reach their destination at the same time as the robot axes.

        If the programmed speed of reorientation or of the additional axes is exceeded, these speeds will be limited, thereby reducing the speed of the TCP.

        The destination position is referenced to the:

- Specified object coordinate system if the argument \WObj is used;
- World coordinate system if the argument \WObj is not used.

Limitations

        When weaving, the distance between the programmed positions should be longer than the periodic time of weaving. If the distance is shorter and if there is a significant change of angle in the path, the weaving pattern will be distorted.

        The instruction ArcC should never be restarted after the circle point has been passed. Otherwise the robot will not take the programmed path (positioning around the circular path in another direction compared with that programmed).

7.1.1 ArcC, ArcC1, ArcC2 - Arc welding with circular motion
Continued

Error handling

The process is supervised by a number of signal inputs. If anything abnormal is detected, program execution will stop. If, however, an error handler is programmed, the errors defined below can be remedied without stopping production. See the example in the instruction `RestoPath`.

Error constant (ERRNO value)	Description
AW_START_ERR	Start condition error; torch, gas or water supervision
AW_IGNI_ERR	Ignition error; arc supervision
AW_WELD_ERR	Weld error; arc supervision
AW_EQIP_ERR	Weld equipment error; voltage, current, water or gas supervision during welding
AW_WIRE_ERR	Wire stick error; wire stick supervision
AW_STOP_ERR	Welding interrupted using the stop process input

The process supervision is determined by a part of the process equipment configuration.

At the start of the process the robot checks that the following preconditions have been met, that is, the following signals are set as follows (see *Defining arc welding systems on page 204*):

- Stop process: low
- Water supervision: high
- Gas supervision: high
- Torch supervision: high

If, after the start command is given, no approved start profile is indicated on the digital input, arc_OK, within a predetermined time period, the process start will be interrupted. When the process is started, all supervision inputs selected - such as stop process, water supervision, gas supervision, arc supervision, volt supervision, current supervision, wire supervision - are monitored continuously.

The wirestick status is checked at the start and end of the weld. Wirestick errors are non-recoverable. That is, the welding process and motion can not be resumed until the wirestick error is corrected.

> **ℹ Note**
>
> Only supervision errors resulting in the error constants `AW_IGNI_ERR` and `AW_WELD_ERR` will have automatic retries (if configured). The other error constants are considered non-recoverable. On `AW_WIRE_ERR` there will be no automatic MoveOut movement (if configured). In a multimove system, when running synchronized welding, there will be no automatic MoveOut movement (if configured) in any of the synchronized robots, if there is an active wirestick error in any of the synchronized robots.

Example

```
MoveL ...
ArcLStart *,v100, seam1, weld5 \Weave:=weave1,
     fine,gun1\Wobj:=wobj1;
```

Continues on next page

```
ArcC *, *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcC *, *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcCEnd *, *, v100, seam1,weld3\Weave:=weave3,
        fine,gun1\Wobj:=wobj1;
MoveL ...
```

In this example, a weld is performed in which weld data and weave data are changed in the final part of the weld, which is illustrated in the following figure.

xx1200000712

It is assumed, in this example, that a coordinated additional axis is used in the movement. In this case, the wobj1 work object must be specified in the instruction.

Limitations

ArcC, ArcC1, ArcC2 cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

Syntax

```
ArcC
  [CirPoint ':='] <expression (IN) of robtarget>
  [ToPoint ':='] <expression (IN) of robtarget>
  [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>]
  ['\' Corr ',']
  |['\' Track ':=' <persistent (PERS) of trackdata>]
  ['\' Time ':=' <expression (IN) of num>]
  ['\' T1 ':=' <variable (VAR) of triggdata>]
  ['\' T2 ':=' <variable (VAR) of triggdata>]
  ['\' T3 ':=' <variable (VAR) of triggdata>]
  ['\' T4 ':=' <variable (VAR) of triggdata>]
```

Continues on next page

7 RAPID reference

7.1.1 ArcC, ArcC1, ArcC2 - Arc welding with circular motion
Continued

```
['\' T5 ':=' <variable (VAR) of triggdata>]
['\' T6 ':=' <variable (VAR) of triggdata>]
['\' T7 ':=' <variable (VAR) of triggdata>]
['\' TLoad':='] <persistent (PERS) of loaddata>]';'
```

Related information

Information	Described in
Performing a linear weld	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*
Other positioning instructions	*Technical reference manual - RAPID Overview*
Definition of speed, speeddata	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of zone data, zonedata	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of tools, tooldata	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of work objects, wobjdata	*Technical reference manual - RAPID Instructions, Functions and Data types*
MoveL	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of loads, loaddata	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of seam data	*seamdata - Seam data on page 178*
Definition of weld data	*welddata - Weld data on page 197*
Definition of weave data	*weavedata - Weave data on page 190*
Installation parameters for welding equipment and welding functions	*System parameters on page 203*
Movements in general	*Technical reference manual - RAPID Overview*
Coordinate systems	*Technical reference manual - RAPID Overview*
Process phases and sub-activities	*Programming on page 21*
Path Offset	*Application manual - Controller software IRC5*
Using optical sensors for tracking or searching.	*Operating manual - Tracking and searching with optical sensors*

7.1.2 ArcCEnd, ArcC1End, ArcC2End - Arc welding end with circular motion

Usage

`ArcCEnd` is used to weld along a circular path. The instruction controls and monitors the entire welding process as follows:

- The tool center point (TCP) is moved in a circle to the specified destination position.
- All phases of the welding process, such as the start and end phases, are controlled.
- The welding process is monitored continuously.

The only difference between `ArcCEnd`, `ArcC1End` and `ArcC2End` is that they are connected to different Arc Weld systems configured in the system parameters. Although `ArcCEnd` is used in the examples, `ArcC1End` or `ArcC2End` could equally well be used.

When the instruction `ArcCEnd` is used, welding ends when the robot reaches the destination position. Regardless of what is specified in the `Zone` argument, the destination position will be a stop point (fine).

If a weld seam is programmed without an `ArcXStart` instruction, `ArcLStart` or `ArcCStart`, there is an error message. When the error message has been acknowledged, the program execution in that task is stopped.

Example

```
MoveL ...
ArcLStart p1, v100, seam1, weld5, fine, gun1;
ArcCEnd p2, p3, v100, seam1, weld5, fine, gun1;
MoveL ...
```

Continues on next page

This welds a circular seam between points p1 and p3 (via point p2) as illustrated in the following figure.
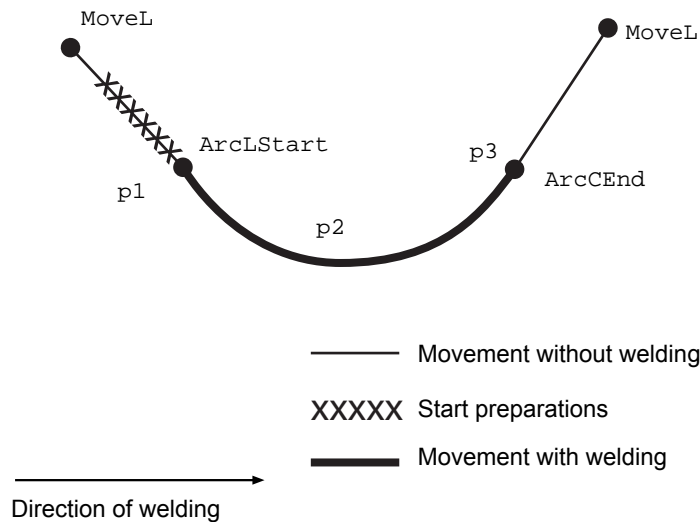
xx1200000710

On the way to `p1`, preparations for the weld start, such as gas preflowing, are carried out. The process and the actual weld movement then start at position `p1` and end at `p3`. The start and end processes are determined by `seam1` and the welding process by `weld5`.

Arguments

```
ArcCEnd CirPoint ToPoint [\ID] Speed Seam Weld [\Weave] Zone Tool
      [\WObj] [\Corr] [\Track] [\Time] [\T1] [\T2] [\T3] [\T4] [\T5]
      [\T6] [\T7] [\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. This is either defined as a named position or stored directly in the instruction (indicated by an * in the instruction).

[\ID]

Data type: `identno`

Continues on next page

The argument `[ \ID ]` is mandatory in 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` in the following cases:

- When the `ArcCStart` instruction is used.
- When the program is run instruction-by-instruction (no welding).

The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. In the figure below, the speed is defined by the `Speed` argument in the respective instructions.

xx1200000710

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. Normally the same seam data is used in all instructions of a seam.

Weld

Data type: `welddata`

Weld data describes the weld phase of the welding process.

Weld data is often changed from one instruction to the next along a seam.

[\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 not specifying any weavedata in the instruction.

Zone

Data type: zonedata

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

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

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

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

Tool

Data type: tooldata

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 omitted, the robot position is referenced to the world coordinate system. It must, however, be specified if a stationary TCP or coordinated additional axes are used.

\WObj can be used if a coordinate system is defined for either the object in question or the weld seam.

[\Corr]

Data type: switch

Correction data written to a corrections entry by the instruction CorrWrite will be added to the path and destination position, if this argument is present.

The RobotWare option *Path Offset* is required when using this argument.

[\Track]

Data type: trackdata

Trackdata is used and is only applicable when the system is configured for seam tracking with a serial weld guide system or with a Laser Tracker system. Seam tracking is activated when this argument is included in the ArcC instruction, but

deactivated if it is omitted. The optional `trackdata` argument must be used during the whole weldseam, that is, from the `ArcXStart` to the `ArcXEnd` instruction.

> **ℹ Note**
>
> Seam tracking requires one of the following options: or options.
> - Tracking Interface
> - *WeldGuide*
> - *Optical tracking*

`[\Time]`

Data type: `num`

This argument is used to specify the total time in seconds during which the robot and additional axes move. It is then substituted for the corresponding speed data.

`[\T1] [\T2] [\T3] [\T4] [\T5] [\T6] [\T7]`

Data type: `triggdata`

Variables that refer to trigger conditions and trigger activity, defined earlier in the program using the instructions `TriggRampAO`, `TriggIO`, `TriggEquip` or `TriggInt`.

`[\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 - Moves the robot linearly*.

Program execution

Controlling process equipment

The process equipment is controlled by the robot in such a way that the entire process and each of its phases are coordinated with the robot's movements.

Motion

Robot and additional axes are moved to the destination position as follows:
- The TCP of the tool is moved circularly at a constant programmed speed. When coordinated axes are used, the robot and the coordinated axes are moved simultaneously, resulting in the programmed path and speed for the TCP relative to the work object.
- The tool is reorientated at even intervals throughout the entire course.
- Uncoordinated additional axes are executed at a constant speed which means that they reach their destination at the same time as the robot axes.

If the programmed speed of reorientation or of the additional axes is exceeded, these speeds will be limited, thereby reducing the speed of the TCP.

Continues on next page

The destination position is referenced to the:

- Specified object coordinate system if the argument `\WObj` is used;
- World coordinate system if the argument `\WObj` is not used.

Limitations

When weaving, the distance between the programmed positions should be longer than the periodic time of weaving. If the distance is shorter and if there is a significant change of angle in the path, the weaving pattern will be distorted.

The instruction `ArcC` should never be restarted after the circle point has been passed. Otherwise the robot will not take the programmed path (positioning around the circular path in another direction compared with that programmed).

Error handling

The process is supervised by a number of signal inputs. If anything abnormal is detected, program execution will stop. If, however, an error handler is programmed, the errors defined below can be remedied without stopping production. See the example in the instruction `RestoPath`.

Error constant (ERRNO value)	Description
AW_START_ERR	Start condition error; torch, gas or water supervision
AW_IGNI_ERR	Ignition error; arc supervision
AW_WELD_ERR	Weld error; arc supervision
AW_EQIP_ERR	Weld equipment error; voltage, current, water or gas supervision during welding
AW_WIRE_ERR	Wire stick error; wire stick supervision
AW_STOP_ERR	Welding interrupted using the stop process input

The process supervision is determined by a part of the process equipment configuration.

At the start of the process the robot checks that the following preconditions have been met, that is, the following signals are set as follows (see *Defining arc welding systems on page 204*):

- Stop process: low
- Water supervision: high
- Gas supervision: high
- Torch supervision: high

If, after the start command is given, no approved start profile is indicated on the digital input, arc_OK, within a predetermined time period, the process start will be interrupted. When the process is started, all supervision inputs selected - such as stop process, water supervision, gas supervision, arc supervision, volt supervision, current supervision, wire supervision - are monitored continuously.

The wirestick status is checked at the start and end of the weld.

Wirestick errors are non-recoverable. That is, the welding process and motion can not be resumed until the wirestick error is corrected.

> **ℹ️ Note**
>
> Only supervision errors resulting in the error constants `AW_IGNI_ERR` and `AW_WELD_ERR` will have automatic retries (if configured). The other error constants are considered non-recoverable. On `AW_WIRE_ERR` there will be no automatic MoveOut movement (if configured). In a multimove system, when running synchronized welding, there will be no automatic MoveOut movement (if configured) in any of the synchronized robots, if there is an active wirestick error in any of the synchronized robots.

Example

```
MoveL ...
ArcLStart *,v100, seam1, weld5 \Weave:=weave1,
     fine,gun1\Wobj:=wobj1;
ArcC *, *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcC *, *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcCEnd *, *, v100, seam1,weld3\Weave:=weave3,
     fine,gun1\Wobj:=wobj1;
MoveL ...
```

In this example, a weld is performed in which weld data and weave data are changed in the final part of the weld, which is illustrated in the following figure.

xx1200000712

It is assumed, in this example, that a coordinated additional axis is used in the movement. In this case, the `wobj1` work object must be specified in the instruction.

Limitations

`ArcCEnd`, `ArcC1End`, `ArcC2End` cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

Continues on next page

Syntax

```
ArcCEnd
    [CirPoint ':='] <expression (IN) of robtarget>
    [ToPoint ':='] <expression (IN) of robtarget>
    [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>]
    ['\' Corr ',']
    |['\' Track ':=' <persistent (PERS) of trackdata>]
    ['\' Time ':=' <expression (IN) of num>]
    ['\' T1 ':=' <variable (VAR) of triggdata>]
    ['\' T2 ':=' <variable (VAR) of triggdata>]
    ['\' T3 ':=' <variable (VAR) of triggdata>]
    ['\' T4 ':=' <variable (VAR) of triggdata>]
    ['\' T5 ':=' <variable (VAR) of triggdata>]
    ['\' T6 ':=' <variable (VAR) of triggdata>]
    ['\' T7 ':=' <variable (VAR) of triggdata>]
    ['\' TLoad ':='] <persistent (PERS) of loaddata>]';'
```

Related information

Information	Described in
Performing a linear weld	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*
Other positioning instructions	*Technical reference manual - RAPID Overview*
Definition of speed, `speeddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of zone data, `zonedata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of tools, `tooldata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of work objects, `wobjdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
`MoveL`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of loads, `loaddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of seam data	*seamdata - Seam data on page 178*
Definition of weld data	*welddata - Weld data on page 197*
Definition of weave data	*weavedata - Weave data on page 190*
Installation parameters for welding equipment and welding functions	*System parameters on page 203*
Movements in general	*Technical reference manual - RAPID Overview*

7.1.2 ArcCEnd, ArcC1End, ArcC2End - Arc welding end with circular motion
Continued

Information	Described in
Coordinate systems	*Technical reference manual - RAPID Overview*
Process phases and sub-activities	*Programming on page 21*
Path Offset	*Application manual - Controller software IRC5*
Using optical sensors for tracking or searching.	*Operating manual - Tracking and searching with optical sensors*

7.1.3 ArcCStart, ArcC1Start, ArcC2Start - Arc welding start with circular motion

Usage

`ArcCStart` is used to weld along a circular path. The instruction controls and monitors the entire welding process as follows:

- The tool center point (TCP) is moved in a circle to the specified destination position.
- All phases of the welding process, such as the start and end phases, are controlled.
- The welding process is monitored continuously.

The only difference between `ArcCStart`, `ArcC1Start` and `ArcC2Start` is that they are connected to different Arc Weld systems configured in the system parameters. Although `ArcCStart` is used in the examples, `ArcC1Start` or `ArcC2Start` could equally well be used.

If a weld seam is programmed without an `ArcXStart` instruction, `ArcLStart` or `ArcCStart`, there is an error message. When the error message has been acknowledged, the program execution in that task is stopped.

A weld can start either with a non moving TCP or with a moving TCP (flying start). In both cases the weld will start to ignite as close as possible to the start point, but in the flying start case the TCP will have moved away from the point due to speed and ignition time before the actual weld begins.

Example

```
MoveJ ...
ArcCStart p1, p2, v100, seam1, weld5, fine, gun1;
ArcCEnd p3, p4, v100, seam1, weld5, fine, gun1;
MoveJ ...
```

This welds a circular seam between points `p2` and `p4` (via point `p3`) as illustrated in the following figure.
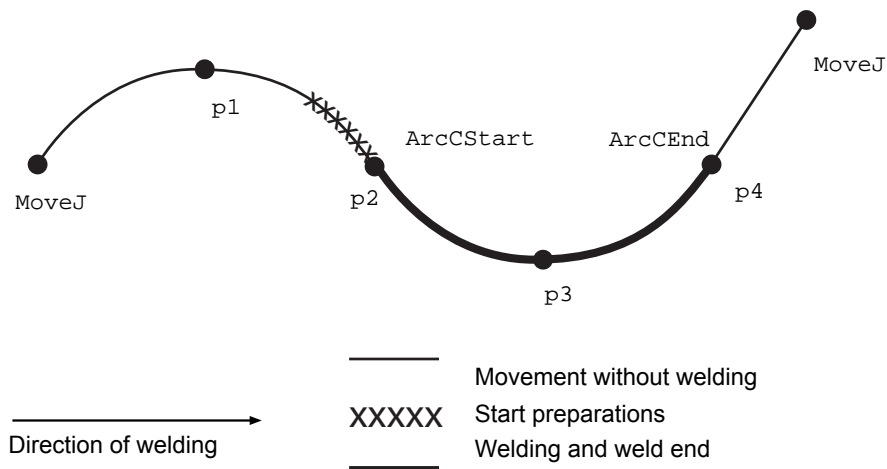
xx1200000711

Continues on next page

7.1.3 ArcCStart, ArcC1Start, ArcC2Start - Arc welding start with circular motion
Continued

On the way to `p2`, preparations for the weld start, such as gas preflowing, are carried out. The process and the actual weld movement then start at position `p2` and end at `p4`. The start and end processes are determined by `seam1` and the welding process by `weld5`.

Arguments

```
ArcCStart CirPoint ToPoint [\ID] Speed Seam Weld [\Weave] Zone Tool
        [\WObj] [\Corr] [\Track] [\PreProcessTracking] [\Seamname]
        [\T1] [\T2] [\T3] [\T4] [\T5] [\T6] [\T7] [\TLoad] [\FlyStart]
```

`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. This is either defined as a named position or stored directly in the instruction (indicated by an * in the instruction).

`[\ID]`

Data type: `identno`

The argument [`\ID`] is mandatory in 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` in the following cases:

- When the `ArcCStart` instruction is used.
- When the program is run instruction-by-instruction (no welding).

Continues on next page

The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. In the figure below, the speed is defined by the `Speed` argument in the respective instructions.

xx1200000710

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. Normally the same seam data is used in all instructions of a seam.

Weld

Data type: `welddata`

Weld data describes the weld phase of the welding process.

Weld data is often changed from one instruction to the next along a seam.

[\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 not specifying any `weavedata` in the instruction.

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.

7.1.3 ArcCStart, ArcC1Start, ArcC2Start - Arc welding start with circular motion
Continued

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

A stop point (`fine`) is always generated automatically at the start position of a weld if the parameter `\flyStart` is not used, and at a controlled weld end position. Fly-by points, such as `z10`, should be used for all other weld positions.

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

If flying start is activated, the start point must be a zone.

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

`Tool`

Data type: `tooldata`

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 omitted, the robot position is referenced to the world coordinate system. It must, however, be specified if a stationary TCP or coordinated additional axes are used.

`\WObj` can be used if a coordinate system is defined for either the object in question or the weld seam.

`[\Corr]`

Data type: `switch`

Correction data written to a corrections entry by the instruction `CorrWrite` will be added to the path and destination position, if this argument is present.

The RobotWare option *Path Offset* is required when using this argument.

`[\Track]`

Data type: `trackdata`

`Trackdata` is used and is only applicable when the system is configured for seam tracking with a serial weld guide system or with a Laser Tracker system. Seam tracking is activated when this argument is included in the `ArcL` instruction, but deactivated if it is omitted. The optional `trackdata` argument must be used during the whole weldseam, that is, from the `ArcXStart` to the `ArcXEnd` instruction.

> **ℹ Note**
>
> Seam tracking requires the *Optical tracking* or *WeldGuide* options.

`[\PreProcessTracking]`

Data type: switch

This argument is effective only if `first_instruction` is set to `TRUE` and the `\Track` argument is present.

This argument activates *Pre Process Tracking*, which means that the robot will be tracking only, without process, during that CapX instruction. Thereby sensor data are available for successful tracking right off the start of the path with process, e.g. welding.

For more information see *Operating manual - Tracking and searching with optical sensors*.

`[\SeamName]`

Data type: `string`

The seam name is a string which will be added to error logs if an error occurs during the welding sequence. `\SeamName` is only applicable together with the `ArcCStart` instruction.

`[\T1] [\T2] [\T3] [\T4] [\T5] [\T6] [\T7]`

Data type: `triggdata`

Variables that refer to trigger conditions and trigger activity, defined earlier in the program using the instructions `TriggRampAO`, `TriggIO`, `TriggEquip` or `TriggInt`.

`[\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 - Moves the robot linearly*.

`[\FlyStart]`

Data type: `flystartdata`

If the weld shall start with a moving TCP it has to be activated via the parameter `active` within the `flystartdata`. The supervision for the ignition is different than for a standing still start. If no ignition has occurred within `superv_distance` from the starting zone a supervision error will occur.

When using flying start the start point must be a zone.

Program execution

Controlling process equipment

The process equipment is controlled by the robot in such a way that the entire process and each of its phases are coordinated with the robot's movements.

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

Motion

Robot and additional axes are moved to the destination position as follows:

- The TCP of the tool is moved circularly at a constant programmed speed. When coordinated axes are used, the robot and the coordinated axes are moved simultaneously, resulting in the programmed path and speed for the TCP relative to the work object.
- The tool is reorientated at even intervals throughout the entire course.
- Uncoordinated additional axes are executed at a constant speed which means that they reach their destination at the same time as the robot axes.

If the programmed speed of reorientation or of the additional axes is exceeded, these speeds will be limited, thereby reducing the speed of the TCP.

The destination position is referenced to the:

- Specified object coordinate system if the argument `\WObj` is used;
- World coordinate system if the argument `\WObj` is not used.

Limitations

When weaving, the distance between the programmed positions should be longer than the periodic time of weaving. If the distance is shorter and if there is a significant change of angle in the path, the weaving pattern will be distorted.

The instruction `ArcC` should never be restarted after the circle point has been passed. Otherwise the robot will not take the programmed path (positioning around the circular path in another direction compared with that programmed).

Flying start

When using flying start the system will trigger the ignition when the TCP passes the starting point. The TCP will be moving and it will change to welding speed as close as possible to the zone centre. Due to the movement the actual position for the start point of the weld will be some distance away from the starting point. That distance is a result of the welding speed and the ignition time of the actual welder.

Flying start cannot be used it in combination with *Ignition Movement Delay* or a *Scrape Start*. The starting point must be a zone.

Flying start ignores the PRE supervision phase. Instead there is a ignition supervision distance that is given with the parameter `superv_distance`. If no ignition has occurred within that distance an ignition error will raise.

Flying start can be deactivated by setting the parameter `active` to false. By doing so the start will be treated as a normal weld start with a stopping TCP. The zone point will be automatic changed to a stop point (`fine`).

Flying start will not be used when restarting after an ignition error or any other weld error.

Error handling

The process is supervised by a number of signal inputs. If anything abnormal is detected, program execution will stop. If, however, an error handler is programmed, the errors defined below can be remedied without stopping production. See the example in the instruction `RestoPath`.

Error constant (ERRNO value)	Description
AW_START_ERR	Start condition error; torch, gas or water supervision
AW_IGNI_ERR	Ignition error; arc supervision
AW_WELD_ERR	Weld error; arc supervision
AW_EQIP_ERR	Weld equipment error; voltage, current, water or gas supervision during welding
AW_WIRE_ERR	Wire stick error; wire stick supervision
AW_STOP_ERR	Welding interrupted using the stop process input

The process supervision is determined by a part of the process equipment configuration.

At the start of the process the robot checks that the following preconditions have been met, that is, the following signals are set as follows (see *Defining arc welding systems on page 204*):

- Stop process: low
- Water supervision: high
- Gas supervision: high
- Torch supervision: high

If, after the start command is given, no approved start profile is indicated on the digital input, arc_OK, within a predetermined time period, the process start will be interrupted. When the process is started, all supervision inputs selected - such as stop process, water supervision, gas supervision, arc supervision, volt supervision, current supervision, wire supervision - are monitored continuously.

Wirestick errors are non-recoverable. That is, the welding process and motion can not be resumed until the wirestick error is corrected.

> **Note**
>
> Only supervision errors resulting in the error constants `AW_IGNI_ERR` and `AW_WELD_ERR` will have automatic retries (if configured). The other error constants are considered non-recoverable. On `AW_WIRE_ERR` there will be no automatic MoveOut movement (if configured). In a multimove system, when running synchronized welding, there will be no automatic MoveOut movement (if configured) in any of the synchronized robots, if there is an active wirestick error in any of the synchronized robots.

Example

```
MoveL ...
ArcLStart *,v100, seam1, weld5 \Weave:=weave1,
     fine,gun1\Wobj:=wobj1;
ArcC *, *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
```

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

7.1.3 ArcCStart, ArcC1Start, ArcC2Start - Arc welding start with circular motion
Continued

```
ArcC *, *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcCEnd *, *, v100, seam1,weld3\Weave:=weave3,
     fine,gun1\Wobj:=wobj1;
MoveL ...
```

In this example, a weld is performed in which weld data and weave data are changed in the final part of the weld, which is illustrated in the following figure.
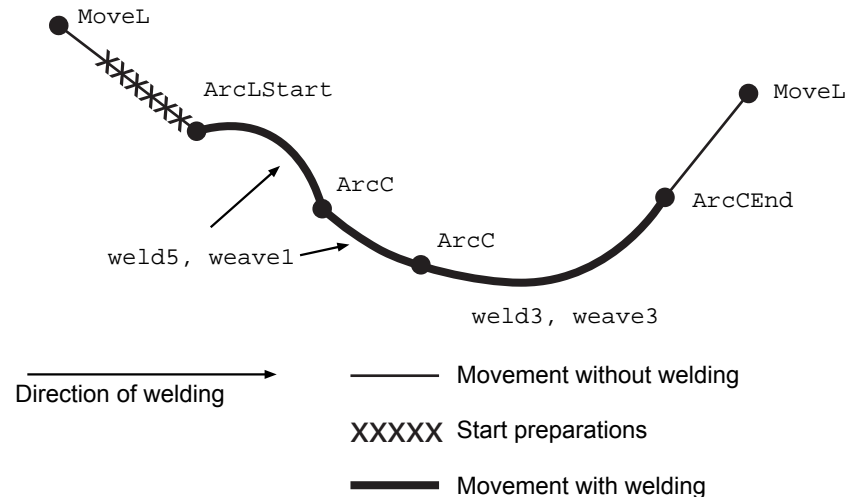
xx1200000712

It is assumed, in this example, that a coordinated additional axis is used in the movement. In this case, the `wobj1` work object must be specified in the instruction.

Limitations

`ArcCStart`, `ArcC1Start`, `ArcC2Start` cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

Syntax

```
ArcCStart
  [CirPoint ':='] <expression (IN) of robtarget>','
  [ToPoint ':='] <expression (IN) of robtarget>','
  [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>]
  ['\' Corr]
  |['\' Track ':=' <persistent (PERS) of trackdata>]
  |['\' PreProcessTracking]
  ['\' SeamName ':=' <expression (IN) of string>]
  ['\' T1 ':=' <variable (VAR) of triggdata>]
  ['\' T2 ':=' <variable (VAR) of triggdata>]
  ['\' T3 ':=' <variable (VAR) of triggdata>]
  ['\' T4 ':=' <variable (VAR) of triggdata>]
```

Continues on next page

7 RAPID reference

7.1.3 ArcCStart, ArcC1Start, ArcC2Start - Arc welding start with circular motion
Continued

```
['\' T5 ':=' <variable (VAR) of triggdata>]
['\' T6 ':=' <variable (VAR) of triggdata>]
['\' T7 ':=' <variable (VAR) of triggdata>]
['\' TLoad ':='] <persistent (PERS) of loaddata>]
['\' FlyStart ':='] <persistent (PERS) of flystartdata>]';'
```

Related information

Information	Described in
Performing a linear weld	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*
Other positioning instructions	*Technical reference manual - RAPID Overview*
Definition of speed, `speeddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of zone data, `zonedata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of tools, `tooldata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of work objects, `wobjdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
`MoveL`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of loads, `loaddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of flying start data	*flystartdata - Flying start data on page 177*
Definition of seam data	*seamdata - Seam data on page 178*
Definition of weld data	*welddata - Weld data on page 197*
Definition of weave data	*weavedata - Weave data on page 190*
Installation parameters for welding equipment and welding functions	*System parameters on page 203*
Movements in general	*Technical reference manual - RAPID Overview*
Coordinate systems	*Technical reference manual - RAPID Overview*
Process phases and sub-activities	*Programming on page 21*
Path Offset	*Application manual - Controller software IRC5*
Using optical sensors for tracking or searching.	*Operating manual - Tracking and searching with optical sensors*

7.1.4 ArcL, ArcL1, ArcL2 - Arc welding with linear motion

Usage

`ArcL` is used to weld along a straight seam. The instruction controls and monitors the entire welding process as follows:

- The tool center point (TCP) is moved linearly to the specified destination position.
- All phases of the welding process, such as the start and end phases, are controlled.
- The welding process is monitored continuously.

The only difference between `ArcL`, `ArcL1` and `ArcL2` is that they are connected to different Arc Weld systems configured in the system parameters. Although `ArcL` is used in the examples, `ArcL1` or `ArcL2` could equally well be used.

If a weld seam is programmed without an `ArcXStart` instruction, `ArcLStart` or `ArcCStart`, there is an error message. When the error message has been acknowledged, the program execution in that task is stopped.

Example

```
MoveJ ...
ArcLStart p1, v100, seam1, weld5, fine, gun1;
ArcL p2, v100, seam1, weld5, fine, gun1;
ArcLEnd p3, v100, seam1, weld5, fine, gun1;
MoveJ ...
```

This welds a seam between points `p1` and `p3`, as illustrated in the following figure.
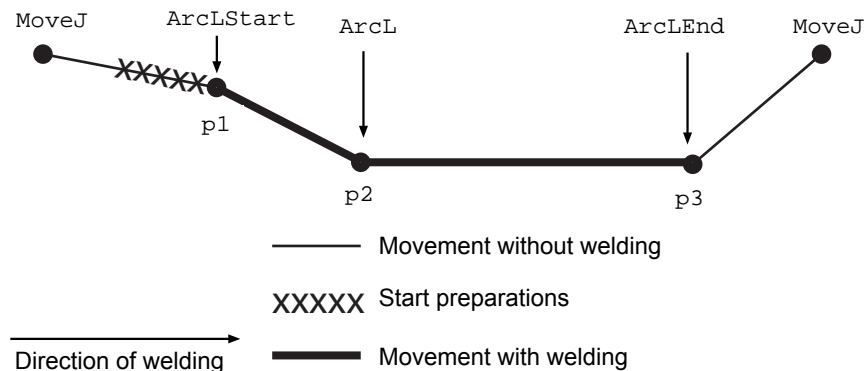
xx1200000706

On the way to `p1`, preparations for the weld start, such as gas preflowing, are carried out. The process and the actual weld movement then start at position `p1` and end at `p3`. The start and end processes are determined by `seam1` and the welding process by `weld5`.

Continues on next page

Arguments

```
ArcL ToPoint [\ID] Speed Seam Weld [\Weave] Zone Tool [\WObj]
     [\Corr] [\Track] [\Time] [\T1] [\T2] [\T3] [\T4] [\T5] [\T6]
     [\T7] [\TLoad]
```

ToPoint

Data type: robtarget

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

[\ID]

Data type: identno

The argument [\ID] is mandatory in 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 in the following cases:

*   When the ArcLStart instruction is used.
*   When the program is run instruction-by-instruction (no welding).

The speed of the TCP during welding is the same as for the arguments Seam and Weld. In the figure below, the speed is defined by the Speed argument in the respective instructions.

xx1200000705

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. Normally the same seam data is used in all instructions of a seam.

Weld

Data type: `welddata`

Weld data describes the weld phase of the welding process.

Weld data is often changed from one instruction to the next along a seam.

[\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 not specifying any weavedata in the instruction.

Zone

Data type: `zonedata`

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

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

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

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

Tool

Data type: `tooldata`

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 omitted, the robot position is referenced to the world coordinate system. It must, however, be specified if a stationary TCP or coordinated additional axes are used.

`\WObj` can be used if a coordinate system is defined for either the object in question or the weld seam.

[\Corr]

Data type: `switch`

Correction data written to a corrections entry by the instruction `CorrWrite` will be added to the path and destination position, if this argument is present.

The RobotWare option *Path Offset* is required when using this argument.

[\Track]

Data type: `trackdata`

`Trackdata` is used and is only applicable when the system is configured for seam tracking with a serial weld guide system or with a Laser Tracker system. Seam tracking is activated when this argument is included in the `ArcL` instruction, but deactivated if it is omitted. The optional `trackdata` argument must be used during the whole weldseam, that is, from the `ArcXStart` to the `ArcXEnd` instruction.

> **ℹ️ Note**
>
> Seam tracking requires the *Optical tracking* or *WeldGuide* options.

[\Time]

Data type: `num`

This argument is used to specify the total time in seconds during which the robot and additional axes move. It is then substituted for the corresponding speed data.

[\T1] [\T2] [\T3] [\T4] [\T5] [\T6] [\T7]

Data type: `triggdata`

Variables that refer to trigger conditions and trigger activity, defined earlier in the program using the instructions `TriggRampAO`, `TriggIO`, `TriggEquip` or `TriggInt`.

[\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 - Moves the robot linearly*.

Program execution

Controlling process equipment

The process equipment is controlled by the robot in such a way that the entire process and each of its phases are coordinated with the robot's movements.

Motion

Robot and additional axes are moved to the destination position as follows:

- The TCP of the tool is moved linearly at a constant programmed speed. When coordinated axes are used, the robot and the coordinated axes are moved simultaneously, resulting in the programmed path and speed for the TCP relative to the work object.
- The tool is reorientated at even intervals throughout the entire course.

- Uncoordinated additional axes are executed at a constant speed which means that they reach their destination at the same time as the robot axes.

If the programmed speed of reorientation or of the additional axes is exceeded, these speeds will be limited, thereby reducing the speed of the TCP.

The destination position is referenced to the:

- Specified object coordinate system if the argument \WObj is used;
- World coordinate system if the argument \WObj is not used.

Limitations

When weaving, the distance between the programmed positions should be longer than the periodic time of weaving. If the distance is shorter and if there is a significant change of angle in the path, the weaving pattern will be distorted.

Do not use double points, i.e. two consecutive robtargets with the same coordinates (x,y,z) in the same weld. It will result in a short weld process stop with possible weld defects. When the error occurs, the error message *110003 Arc Supervision* is reported.

Error handling

The process is supervised by a number of signal inputs. If anything abnormal is detected, program execution will stop. If, however, an error handler is programmed, the errors defined below can be remedied without stopping production. See the example in the instruction RestoPath.

Error constant (ERRNO value)	Description
AW_START_ERR	Start condition error; torch, gas or water supervision
AW_IGNI_ERR	Ignition error; arc supervision
AW_WELD_ERR	Weld error; arc supervision
AW_EQIP_ERR	Weld equipment error; voltage, current, water or gas supervision during welding
AW_WIRE_ERR	Wire stick error; wire stick supervision
AW_STOP_ERR	Welding interrupted using the stop process input

The process supervision is determined by a part of the process equipment configuration.

At the start of the process the robot checks that the following preconditions have been met, that is, the following signals are set as follows (see Defining arc welding systems on page 53):

- Stop process: low
- Water supervision: high
- Gas supervision: high
- Torch supervision: high

If, after the start command is given, no approved start profile is indicated on the digital input, arc supervision, within a predetermined time period, the process start will be interrupted. When the process is started, all supervision inputs selected - such as stop process, water supervision, gas supervision, arc supervision, volt supervision, current supervision, wire supervision - are monitored continuously.

The wirestick status is checked at the start and end of the weld. Wirestick errors are non-recoverable. That is, the welding process and motion can not be resumed until the wirestick error is corrected.

> **ℹ️ Note**
>
> Only supervision errors resulting in the error constants `AW_IGNI_ERR` and `AW_WELD_ERR` will have automatic retries (if configured). The other error constants are considered non-recoverable. On `AW_WIRE_ERR` there will be no automatic MoveOut movement (if configured). In a multimove system, when running synchronized welding, there will be no automatic MoveOut movement (if configured) in any of the synchronized robots, if there is an active wirestick error in any of the synchronized robots.

Example

```
MoveL ...
ArcLStart *, v100, seam1, weld5 \Weave:=weave1, fine,
    gun1\Wobj:=wobj1;
ArcL *, v100, seam1, weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcL *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcLEnd *, v100, seam1, weld3 \Weave:=weave3, fine,
    gun1\Wobj:=wobj1;
MoveL ...
```

In this example, a weld is performed in which weld data and weave data are changed in the final part of the weld, which is illustrated in the following figure. Note that an arc welding instruction must be used to change the direction of the path despite the fact that no weld data is changed.

xx1200000707

It is assumed, in this example, that a coordinated additional axis is used in the movement. In this case, the wobj1 work object must be specified in the instruction.

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

Limitations

`ArcL`, `ArcL1`, `ArcL2` **cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.**

Syntax

```
ArcL
  [ToPoint ':='] <expression (IN) of robtarget>
  [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>]
  ['\'Corr ',']
  |['\' Track ':=' <persistent (PERS) of trackdata>]
  ['\' Time ':=' <expression (IN) of num>]
  ['\' T1 ':=' <variable (VAR) of triggdata>]
  ['\' T2 ':=' <variable (VAR) of triggdata>]
  ['\' T3 ':=' <variable (VAR) of triggdata>]
  ['\' T4 ':=' <variable (VAR) of triggdata>]
  ['\' T5 ':=' <variable (VAR) of triggdata>]
  ['\' T6 ':=' <variable (VAR) of triggdata>]
  ['\' T7 ':=' <variable (VAR) of triggdata>]
  ['\' TLoad ':='] <persistent (PERS) of loaddata>]';'
```

Related information

Information	Described in
Performing a circular motion weld	*ArcC, ArcC1, ArcC2 - Arc welding with circular motion on page 107*
Other positioning instructions	*Technical reference manual - RAPID Overview*
Definition of speed, `speeddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of zone data, `zonedata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of tools, `tooldata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of work objects, `wobjdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
`MoveL`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of loads, `loaddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of seam data	*seamdata - Seam data on page 178*
Definition of weld data	*welddata - Weld data on page 197*
Definition of weave data	*weavedata - Weave data on page 190*

Information	Described in
Installation parameters for welding equipment and welding functions	*System parameters on page 203*
Movements in general	*Technical reference manual - RAPID Overview*
Coordinate systems	*Technical reference manual - RAPID Overview*
Process phases and sub-activities	*Programming on page 21*
Path Offset	*Application manual - Controller software IRC5*
Using optical sensors for tracking or searching.	*Operating manual - Tracking and searching with optical sensors*

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

7.1.5 ArcLEnd, ArcL1End, ArcL2End - Arc welding end with linear motion

Usage

`ArcLEnd` is used to weld along a straight seam. The instruction controls and monitors the entire welding process as follows:

- The tool center point (TCP) is moved linearly to the specified destination position.
- All phases of the welding process, such as the start and end phases, are controlled.
- The welding process is monitored continuously.

The only difference between `ArcLEnd`, `ArcL1End` and `ArcL2End` is that they are connected to different Arc Weld systems configured in the system parameters. Although `ArcLEnd` is used in the examples, `ArcL1End` or `ArcL2End` could equally well be used.

When the instruction `ArcLEnd` is used, welding ends when the robot reaches the destination position. Regardless of what is specified in the `Zone` argument, the destination position will be a stop point (fine).

If a weld seam is programmed without an `ArcXStart` instruction, `ArcLStart` or `ArcCStart`, there is an error message. When the error message has been acknowledged, the program execution in that task is stopped.

Example

```
MoveJ ...
ArcLStart p1, v100, seam1, weld5, fine, gun1;
ArcLEnd p2, v100, seam1, weld5, fine, gun1;
MoveJ ...
```

This welds a straight seam between points p1 and p2, as illustrated in the following figure.
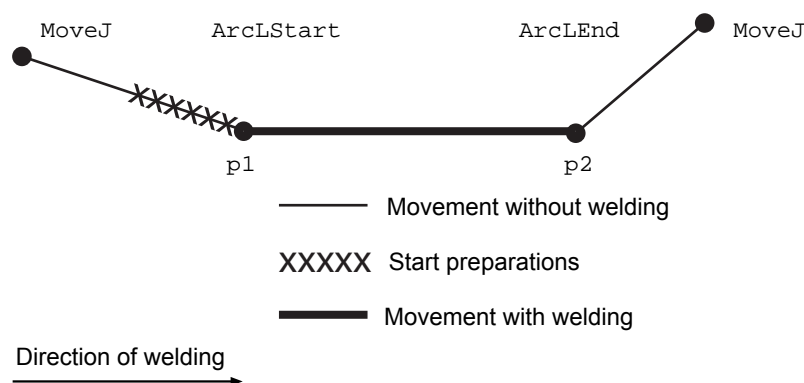
xx1200000705

On the way to `p1`, preparations for the weld start, such as gas preflowing, are carried out. The process and the actual weld movement then start at position `p1`

Continues on next page

and end at `p2`. The start and end processes are determined by `seam1` and the welding process by `weld5`.

Arguments

```
ArcLEnd ToPoint [\ID] Speed Seam Weld [\Weave] Zone Tool [\WObj]
        [\Corr] [\Track] [\Time] [\T1] [\T2] [\T3] [\T4] [\T5] [\T6]
        [\T7] [\TLoad]
```

ToPoint

Data type: `robtarget`

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

[\ID]

Data type: `identno`

The argument [\ID] is mandatory in 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` in the following cases:

- When the `ArcLStart` instruction is used.
- When the program is run instruction-by-instruction (no welding).

The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. In the figure below, the speed is defined by the `Speed` argument in the respective instructions.

xx1200000705

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. Normally the same seam data is used in all instructions of a seam.

Weld

Data type: `welddata`

Weld data describes the weld phase of the welding process.

Weld data is often changed from one instruction to the next along a seam.

[\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 not specifying any `weavedata` in the instruction.

Zone

Data type: `zonedata`

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

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

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

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

Tool

Data type: `tooldata`

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 omitted, the robot position is referenced to the world coordinate system. It must, however, be specified if a stationary TCP or coordinated additional axes are used.

`\WObj` can be used if a coordinate system is defined for either the object in question or the weld seam.

[\Corr]

Data type: `switch`

Correction data written to a corrections entry by the instruction `CorrWrite` will be added to the path and destination position, if this argument is present.

The RobotWare option *Path Offset* is required when using this argument.

[\Track]

Data type: `trackdata`

`Trackdata` is used and is only applicable when the system is configured for seam tracking with a serial weld guide system or with a Laser Tracker system. Seam tracking is activated when this argument is included in the `ArcL` instruction, but deactivated if it is omitted. The optional `trackdata` argument must be used during the whole weldseam, that is, from the `ArcXStart` to the `ArcXEnd` instruction.

> ℹ️ **Note**
>
> Seam tracking requires the *Optical tracking* or *WeldGuide* options.

[\Time]

Data type: `num`

This argument is used to specify the total time in seconds during which the robot and additional axes move. It is then substituted for the corresponding speed data.

[\T1] [\T2] [\T3] [\T4] [\T5] [\T6] [\T7]

Data type: `triggdata`

Variables that refer to trigger conditions and trigger activity, defined earlier in the program using the instructions `TriggRampAO`, `TriggIO`, `TriggEquip` or `TriggInt`.

[\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 - Moves the robot linearly*.

Program execution

Controlling process equipment

The process equipment is controlled by the robot in such a way that the entire process and each of its phases are coordinated with the robot's movements.

Motion

Robot and additional axes are moved to the destination position as follows:

- The TCP of the tool is moved linearly at a constant programmed speed. When coordinated axes are used, the robot and the coordinated axes are moved

simultaneously, resulting in the programmed path and speed for the TCP relative to the work object.

- The tool is reorientated at even intervals throughout the entire course.
- Uncoordinated additional axes are executed at a constant speed which means that they reach their destination at the same time as the robot axes.

If the programmed speed of reorientation or of the additional axes is exceeded, these speeds will be limited, thereby reducing the speed of the TCP.

The destination position is referenced to the:

- Specified object coordinate system if the argument \WObj is used;
- World coordinate system if the argument \WObj is not used.

Limitations

When weaving, the distance between the programmed positions should be longer than the periodic time of weaving. If the distance is shorter and if there is a significant change of angle in the path, the weaving pattern will be distorted.

Do not use double points, i.e. two consecutive robtargets with the same coordinates (x,y,z) in the same weld. It will result in a short weld process stop with possible weld defects. When the error occurs, the error message *110003 Arc Supervision* is reported.

Error handling

The process is supervised by a number of signal inputs. If anything abnormal is detected, program execution will stop. If, however, an error handler is programmed, the errors defined below can be remedied without stopping production. See the example in the instruction RestoPath.

Error constant (ERRNO value)	Description
AW_START_ERR	Start condition error; torch, gas or water supervision
AW_IGNI_ERR	Ignition error; arc supervision
AW_WELD_ERR	Weld error; arc supervision
AW_EQIP_ERR	Weld equipment error; voltage, current, water or gas supervision during welding
AW_WIRE_ERR	Wire stick error; wire stick supervision
AW_STOP_ERR	Welding interrupted using the stop process input

The process supervision is determined by a part of the process equipment configuration.

At the start of the process the robot checks that the following preconditions have been met, that is, the following signals are set as follows (see *Defining arc welding systems on page 204*):

- Stop process: low
- Water supervision: high
- Gas supervision: high
- Torch supervision: high

If, after the start command is given, no approved start profile is indicated on the digital input, arc supervision, within a predetermined time period, the process start

will be interrupted. When the process is started, all supervision inputs selected - such as stop process, water supervision, gas supervision, arc supervision, volt supervision, current supervision, wire supervision - are monitored continuously.

The wirestick status is checked at the start and end of the weld. Wirestick errors are non-recoverable. That is, the welding process and motion can not be resumed until the wirestick error is corrected.

> **ℹ️ Note**
>
> Only supervision errors resulting in the error constants `AW_IGNI_ERR` and `AW_WELD_ERR` will have automatic retries (if configured). The other error constants are considered non-recoverable. On `AW_WIRE_ERR` there will be no automatic MoveOut movement (if configured). In a multimove system, when running synchronized welding, there will be no automatic MoveOut movement (if configured) in any of the synchronized robots, if there is an active wirestick error in any of the synchronized robots.

Example

```
MoveL ...
ArcLStart *, v100, seam1, weld5 \Weave:=weave1, fine,
    gun1\Wobj:=wobj1;
ArcL *, v100, seam1, weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcL *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcLEnd *, v100, seam1, weld3 \Weave:=weave3, fine,
    gun1\Wobj:=wobj1;
MoveL ...
```

In this example, a weld is performed in which weld data and weave data are changed in the final part of the weld, which is illustrated in the following figure. Note that an arc welding instruction must be used to change the direction of the path despite the fact that no weld data is changed.

xx1200000707

It is assumed, in this example, that a coordinated additional axis is used in the movement. In this case, the wobj1 work object must be specified in the instruction.

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

Limitations

`ArcLEnd`, `ArcL1End`, `ArcL2End` cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

`ArcLEnd` cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

Syntax

```
ArcLEnd
  [ToPoint ':='] <expression (IN) of robtarget>
  [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>]
  ['\'Corr ',']
  |['\' Track ':=' <persistent (PERS) of trackdata>]
  ['\' Time ':=' <expression (IN) of num>]
  ['\' T1 ':=' <variable (VAR) of triggdata>]
  ['\' T2 ':=' <variable (VAR) of triggdata>]
  ['\' T3 ':=' <variable (VAR) of triggdata>]
  ['\' T4 ':=' <variable (VAR) of triggdata>]
  ['\' T5 ':=' <variable (VAR) of triggdata>]
  ['\' T6 ':=' <variable (VAR) of triggdata>]
  ['\' T7 ':=' <variable (VAR) of triggdata>]
  ['\' TLoad ':='] <persistent (PERS) of loaddata>]';'
```

Related information

Information	Described in
Performing a linear weld	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*
Other positioning instructions	*Technical reference manual - RAPID Overview*
Definition of speed, `speeddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of zone data, `zonedata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of tools, `tooldata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of work objects, `wobjdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
`MoveL`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of loads, `loaddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*

7 RAPID reference

Information	Described in
Definition of seam data	*seamdata - Seam data on page 178*
Definition of weld data	*welddata - Weld data on page 197*
Definition of weave data	*weavedata - Weave data on page 190*
Installation parameters for welding equipment and welding functions	*System parameters on page 203*
Movements in general	*Technical reference manual - RAPID Overview*
Coordinate systems	*Technical reference manual - RAPID Overview*
Process phases and sub-activities	*Programming on page 21*
Path Offset	*Application manual - Controller software IRC5*
Using optical sensors for tracking or searching.	*Operating manual - Tracking and searching with optical sensors*

7.1.6 ArcLStart, ArcL1Start, ArcL2Start - Arc welding start with linear motion

Usage

ArcLStart is used to weld along a straight seam. The instruction controls and monitors the entire welding process as follows:

- The tool center point (TCP) is moved linearly to the specified destination position.
- All phases of the welding process, such as the start and end phases, are controlled.
- The welding process is monitored continuously.

The instruction ArcLStart is used for start preparations, for example gas purging, that are carried out on the way to the weld start position.

The only difference between ArcLStart, ArcL1Start and ArcL2Startis is that they are connected to different arc weld systems configured in the system parameters. Although ArcLStart is used in the examples, ArcL1Start or ArcL2Start could equally well be used.

If a weld seam is programmed without an ArcXStart instruction, ArcLStart or ArcCStart, there is an error message. When the error message has been acknowledged, the program execution in that task is stopped.

A weld can start either with a non moving TCP or with a moving TCP (flying start). In both cases the weld will start to ignite as close as possible to the start point, but in the flying start case the TCP will have moved away from the point due to speed and ignition time before the actual weld begins.

Example

```
MoveJ ...
ArcLStart p1, v100, seam1, weld5, fine, gun1;
ArcLEnd p2, v100, seam1, weld5, fine, gun1;
MoveJ ...
```

This welds a straight seam between points p1 and p2, as illustrated in the following figure.

xx1200000705

Continues on next page

On the way to `p1`, preparations for the weld start, such as gas preflowing, are carried out. The process and the actual weld movement then start at position `p1` and end at `p2`. The start and end processes are determined by `seam1` and the welding process by `weld5`.

Arguments

```
ArcLStart ToPoint [\ID] Speed Seam Weld [\Weave] Zone Tool [\WObj]
       [\Corr] [\Track] [\PreProcessTracking] [\SeamName] [\T1] [\T2]
       [\T3] [\T4] [\T5] [\T6] [\T7] [\TLoad] [\FlyStart]
```

ToPoint

Data type: `robtarget`

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

[\ID]

Data type: `identno`

The argument `[ \ID ]` is mandatory in 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` in the following cases:

- When the `ArcLStart` instruction is used.
- When the program is run instruction-by-instruction (no welding).

The speed of the TCP during welding is the same as for the arguments `Seam` and `Weld`. In the figure below, the speed is defined by the `Speed` argument in the respective instructions.

xx1200000705

7.1.6 ArcLStart, ArcL1Start, ArcL2Start - Arc welding start with linear motion
Continued

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

Seam

Data type: `seamdata`

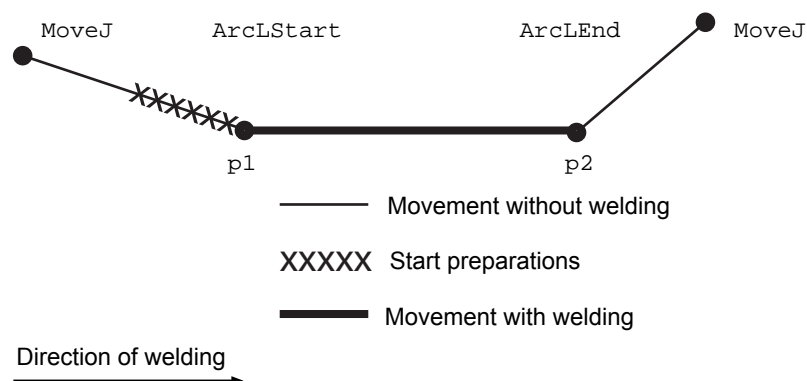Seam data describes the start and end phases of a welding process. The argument `Seam` is included in all arc welding instructions so that, regardless of the position of the robot when the process is interrupted, a proper weld end and restart is achieved. Normally the same seam data is used in all instructions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

Weld

Data type: `welddata`

Weld data describes the weld phase of the welding process.

Weld data is often changed from one instruction to the next along a seam.

[\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 not specifying any `weavedata` in the instruction.

Zone

Data type: `zonedata`

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

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

A stop point (`fine`) is always generated automatically at the start position of a weld if the parameter `\flyStart` is not used, and at a controlled weld end position. Fly-by points, such as `z10`, should be used for all other weld positions.

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

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

Tool

Data type: `tooldata`

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 omitted, the robot position is referenced to the

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

`\WObj` can be used if a coordinate system is defined for either the object in question or the weld seam.

[\Corr]

Data type: `switch`

Correction data written to a corrections entry by the instruction `CorrWrite` will be added to the path and destination position, if this argument is present.

The RobotWare option *Path Offset* is required when using this argument.

[\Track]

Data type: `trackdata`

`Trackdata` is used and is only applicable when the system is configured for seam tracking with a serial weld guide system or with a Laser Tracker system. Seam tracking is activated when this argument is included in the `ArcL` instruction, but deactivated if it is omitted. The optional `trackdata` argument must be used during the whole weldseam, that is, from the `ArcXStart` to the `ArcXEnd` instruction.

> ℹ️ **Note**
>
> Seam tracking requires the Optical tracking or WeldGuide options.

[\PreProcessTracking]

Data type: switch

This argument is effective only if `first_instruction` is set to `TRUE` and the `\Track` argument is present.

This argument activates *Pre Process Tracking*, which means that the robot will be tracking only, without process, during that CapX instruction. Thereby sensor data are available for successful tracking right off the start of the path with process, e.g. welding.

For more information see *Operating manual - Tracking and searching with optical sensors*.

[\SeamName]

Data type: `string`

The seam name is a string which will be added to error logs if an error occurs during the welding sequence. `\SeamName` is only applicable together with the `ArcLStart` instruction.

[\T1] [\T2] [\T3] [\T4] [\T5] [\T6] [\T7]

Data type: `triggdata`

Variables that refer to trigger conditions and trigger activity, defined earlier in the program using the instructions `TriggRampAO`, `TriggIO`, `TriggEquip` or `TriggInt`.

[\TLoad]

Data type: `loaddata`

Continues on next page

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

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

[\FlyStart]

Data type: `flystartdata`

If the weld shall start with a moving TCP it has to be activated via the parameter `active` within the `flystartdata`. The supervision for the ignition is different than for a standing still start. If no ignition has occurred within `superv_distance` from the starting zone a supervision error will occur.

When using flying start the start point must be a zone.

Program execution

Controlling process equipment

The process equipment is controlled by the robot in such a way that the entire process and each of its phases are coordinated with the robot's movements.

Motion

Robot and additional axes are moved to the destination position as follows:

- The TCP of the tool is moved linearly at a constant programmed speed. When coordinated axes are used, the robot and the coordinated axes are moved simultaneously, resulting in the programmed path and speed for the TCP relative to the work object.
- The tool is reoriented at even intervals throughout the entire course.
- Uncoordinated additional axes are executed at a constant speed which means that they reach their destination at the same time as the robot axes.

If the programmed speed of reorientation or of the additional axes is exceeded, these speeds will be limited, thereby reducing the speed of the TCP.

The destination position is referenced to the:

- Specified object coordinate system if the argument `\WObj` is used;
- World coordinate system if the argument `\WObj` is not used.

Limitations

When weaving, the distance between the programmed positions should be longer than the periodic time of weaving. If the distance is shorter and if there is a significant change of angle in the path, the weaving pattern will be distorted.

Do not use double points, i.e. two consecutive robtargets with the same coordinates (x,y,z) in the same weld. It will result in a short weld process stop with possible weld defects. When the error occurs, the error message *110003 Arc Supervision* is reported.

7.1.6 ArcLStart, ArcL1Start, ArcL2Start - Arc welding start with linear motion
Continued

Flying start

When using flying start the system will trigger the ignition when the TCP passes the starting point. The TCP will be moving and it will change to welding speed as close as possible to the zone centre. Due to the movement the actual position for the start point of the weld will be some distance away from the starting point. That distance is a result of the welding speed and the ignition time of the actual welder.

Flying start cannot be used it in combination with *Ignition Movement Delay* or a *Scrape Start*. The starting point must be a zone.

Flying start ignores the PRE supervision phase. Instead there is a ignition supervision distance that is given with the parameter `superv_distance`. If no ignition has occurred within that distance an ignition error will raise.

Flying start can be deactivated by setting the parameter `active` to false. By doing so the start will be treated as a normal weld start with a stopping TCP. The zone point will be automatic changed to a stop point (`fine`).

Flying start will not be used when restarting after an ignition error or any other weld error.

Error handling

The process is supervised by a number of signal inputs. If anything abnormal is detected, program execution will stop. If, however, an error handler is programmed, the errors defined below can be remedied without stopping production. See the example in the instruction `RestoPath`.

Error constant (ERRNO value)	Description
AW_START_ERR	Start condition error; torch, gas or water supervision
AW_IGNI_ERR	Ignition error; arc supervision
AW_WELD_ERR	Weld error; arc supervision
AW_EQIP_ERR	Weld equipment error; voltage, current, water or gas supervision during welding
AW_WIRE_ERR	Wire stick error; wire stick supervision
AW_STOP_ERR	Welding interrupted using the stop process input

The process supervision is determined by a part of the process equipment configuration.

At the start of the process the robot checks that the following preconditions have been met, that is, the following signals are set as follows (see *Defining arc welding systems on page 204*):

- Stop process: low
- Water supervision: high
- Gas supervision: high
- Torch supervision: high

If, after the start command is given, no approved start profile is indicated on the digital input, arc supervision, within a predetermined time period, the process start will be interrupted. When the process is started, all supervision inputs selected - such as stop process, water supervision, gas supervision, arc supervision, volt supervision, current supervision, wire supervision - are monitored continuously.

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

The wirestick status is checked at the start and end of the weld. Wirestick errors are non-recoverable. That is, the welding process and motion can not be resumed until the wirestick error is corrected.

> **ℹ Note**
>
> Only supervision errors resulting in the error constants `AW_IGNI_ERR` and `AW_WELD_ERR` will have automatic retries (if configured). The other error constants are considered non-recoverable. On `AW_WIRE_ERR` there will be no automatic MoveOut movement (if configured). In a multimove system, when running synchronized welding, there will be no automatic MoveOut movement (if configured) in any of the synchronized robots, if there is an active wirestick error in any of the synchronized robots.

Example

```
MoveL ...
ArcLStart *, v100, seam1, weld5 \Weave:=weave1, fine,
    gun1\Wobj:=wobj1;
ArcL *, v100, seam1, weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcL *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcLEnd *, v100, seam1, weld3 \Weave:=weave3, fine,
    gun1\Wobj:=wobj1;
MoveL ...
```

In this example, a weld is performed in which weld data and weave data are changed in the final part of the weld, which is illustrated in the following figure. Note that an arc welding instruction must be used to change the direction of the path despite the fact that no weld data is changed.

xx1200000707

It is assumed, in this example, that a coordinated additional axis is used in the movement. In this case, the wobj1 work object must be specified in the instruction.

Limitations

`ArcLStart`, `ArcL1Start`, `ArcL2Start` **cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.**

Syntax

```
ArcLStart
   [ToPoint ':='] <expression (IN) of robtarget>
   [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>]
   ['\' Corr]
   |['\' Track ':=' <persistent (PERS) of trackdata>]
   |['\' PreProcessTracking]
   ['\' SeamName ':=' <expression (IN) of string>]
   ['\' T1 ':=' <variable (VAR) of triggdata>]
   ['\' T2 ':=' <variable (VAR) of triggdata>]
   ['\' T3 ':=' <variable (VAR) of triggdata>]
   ['\' T4 ':=' <variable (VAR) of triggdata>]
   ['\' T5 ':=' <variable (VAR) of triggdata>]
   ['\' T6 ':=' <variable (VAR) of triggdata>]
   ['\' T7 ':=' <variable (VAR) of triggdata>]
   ['\' TLoad ':='] <persistent (PERS) of loaddata>]
   ['\' FlyStart ':='] <persistent (PERS) of flystartdata>]';'
```

Related information

Information	Described in
Performing a linear weld	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*
Other positioning instructions	*Technical reference manual - RAPID Overview*
Definition of speed, `speeddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of zone data, `zonedata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of tools, `tooldata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of work objects, `wobjdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
`MoveL`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of loads, `loaddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of flying start data	*flystartdata - Flying start data on page 177*

Continues on next page

Information	Described in
Definition of seam data	*seamdata - Seam data on page 178*
Definition of weld data	*welddata - Weld data on page 197*
Definition of weave data	*weavedata - Weave data on page 190*
Installation parameters for welding equipment and welding functions	*System parameters on page 203*
Movements in general	*Technical reference manual - RAPID Overview*
Coordinate systems	*Technical reference manual - RAPID Overview*
Process phases and sub-activities	*Programming on page 21*
Path Offset	*Application manual - Controller software IRC5*
Using optical sensors for tracking or searching.	*Operating manual - Tracking and searching with optical sensors*

7.1.7 ArcMoveExtJ - Move one or several mechanical units without TCP

Usage

ArcMoveExtJ (*Move External Joints*) is used to move linear or rotating additional axes in a MultiMove system, if the RAPID program for the additional axis is coordinated with an arc instruction. The additional axes can belong to one or several mechanical units without TCP. This instruction can only be used with an actual program task defined as a motion task and if the task controls one or several mechanical units without TCP.

ArcMoveExtJ is almost a copy of the MoveExtJ instruction but has integrated error handling for *RobotWare Arc*.

ArcMoveExtJ must be used together with ArcX instructions in the robot task.

Basic examples

The following example illustrates the instruction ArcMoveExtJ.

Example 1

```
ActUnit STN1;
SyncMoveOn sync001, allTasks;
MoveExtJ p10\ID:=10,vrot_max, z10;
MoveExtJ p20\ID:=20,vrot_max, z10;
ArcMoveExtJ p30\ID:=30, vrot50, fine\Start;
ArcMoveExtJ p40\ID:=40, vrot50, z10;
```

Arguments

```
ArcMoveExtJ [\Conc] ToJointPos [\ID] Speed [\T] Zone [\InPos]
        [\Start]
```

[\Conc]

Data type: switch

Subsequent instructions are executed while the external axis is moving. The argument is usually not used but can be used to avoid unwanted stops caused by overloaded CPU when using fly-by points. This is useful when the programmed points are very close together at high speeds. The argument is also useful when, for example, communicating with external equipment and synchronization between the external equipment and robot movement is not required.

ToJointPos

Data type: robtarget

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

[\ID]

Data type: identno

The argument [\ID] is mandatory in 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

Continues on next page

program tasks. By using the id number the movements are not mixed up at the runtime.

Speed

Data type: speeddata

The speed data that applies to movements. Speed data defines the velocity of the linear or rotating additional axis.

[\T]

Data type: num

This argument is used to specify the total time in seconds during which the additional axis moves. It is then substituted for the corresponding speed data.

Zone

Data type: zonedata

Zone data for the movement. Zone data defines stop point or fly-by point. If it is a fly-by point then the zone size describes the deceleration and acceleration for the linear or rotational additional axis.

[\InPos]

Data type: stoppointdata

This argument is used to specify the convergence criteria for the position of the external axis in the stop point. The stop point data substitutes the zone specified in the Zone parameter.

[\Start]

Data type: switch

This argument must be used to indicate start to the corresponding ArcLStart instruction in a *MultiMove* system (synchronized).

Limitations

ArcMoveExtJ cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

Syntax

```
ArcMoveExtJ
  ['\' Conc ',']
  [ToJointPos ':='] <expression (IN) of jointtarget>
  ['\' ID ':=' <expression (IN) of identno>]','
  [Speed ':='] <expression (IN) of speeddata>
  ['\' T ':=' <expression (IN) of num>]','
  [Zone ':='] <expression (IN) of zonedata>
  ['\' Inpos ':=' <expression (IN) of stoppointdata>]','
  ['\' Start]';'
```

7.1.8 ArcRefresh - Refresh arc weld data

Usage

ArcRefresh is used to tune arc welding process parameters during program execution.

Example

```
PROC PulseWeld()
  ! Setup a two Hz timer interrupt
  CONNECT intno1 WITH TuneTrp;
  ITimer,0.5 ,intno1;
  ! Weld the seam
  ArcLStart p1, v100, seam1, weld5 \Weave:=noweave, fine, gun1;
  ArcLEnd p2, v100, seam1, weld5 \Weave:=noweave, fine, gun1;
  IDelete intno1;
ENDPROC
TRAP TuneTrp
  ! Modify the weld_voltage component of active welddata
  IF HighValueFlag = TRUE THEN
    weld5.main_arc.voltage := 10;
    HighValueFlag := FALSE;
    ELSE
    weld5.main_arc.voltage := 15;
    HighValueFlag := TRUE;
  ENDIF
  ! Order the process control to refresh process parameters
  ArcRefresh;
ENDTRAP
```

The weld voltage will be switched between 10 and 15 volts by the trap routine at a 2 Hz rate.

Arguments

ArcRefresh [\UpdateCalib] [\WeldSpeed] [\WeaveWidth]

[\UpdateCalib]

Data type: switch

This optional switch is used to update calibration data to the optical tracking sensor. Calibration data is transferred to the optical tracking sensor at controller warmstart or by using ArcRefresh with this optional switch.

[\WeldSpeed]

Data type: num

This optional parameter is used to update the weld speed in welddata for the currently executing ArcX instruction. The weld speed is the only data that is updated with this optional parameter. The weldspeed should be expressed in mm/s.

[\WeaveWidth]

Data type: num

Continues on next page

This optional parameter is used to update the `weavewidth` in `weavedata` for the currently executing `ArcX` instruction. The `weavewidth` is the only data that is updated with this optional parameter. The weave width should be expressed in mm.

Limitations

`ArcRefresh` cannot be executed in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

Syntax

```
ArcRefresh
  ['\' UpdateCalib]
  ['\' WeldSpeed ':=' <variable (VAR) of num>]
  ['\' WeaveWidth ':=' <variable (VAR) of num >]';'
```

Related information

Information	Described in
Performing a circular weld	*ArcC, ArcC1, ArcC2 - Arc welding with circular motion on page 107*
Performing a linear weld	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*
Other positioning instructions	*Technical reference manual - RAPID Overview*
Definition of speed, `speeddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of zone data, `zonedata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of tools, `tooldata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of work objects, `wobjdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
`MoveL`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of loads, `loaddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of seam data	*seamdata - Seam data on page 178*
Definition of weld data	*welddata - Weld data on page 197*
Definition of weave data	*weavedata - Weave data on page 190*
Installation parameters for welding equipment and welding functions	*System parameters on page 203*
Movements in general	*Technical reference manual - RAPID Overview*
Coordinate systems	*Technical reference manual - RAPID Overview*
Process phases and sub-activities	*Programming on page 21*

7.1.9 RecoveryMenu - Display the recovery menu

Usage

`RecoveryMenu` is used by custom welding error handlers to display the recovery menu user interface. The selection made by the user will be stored internally, and will be referenced by the Weld Error Recovery feature when RobotWare Arc attempts to re-ignite the arc.
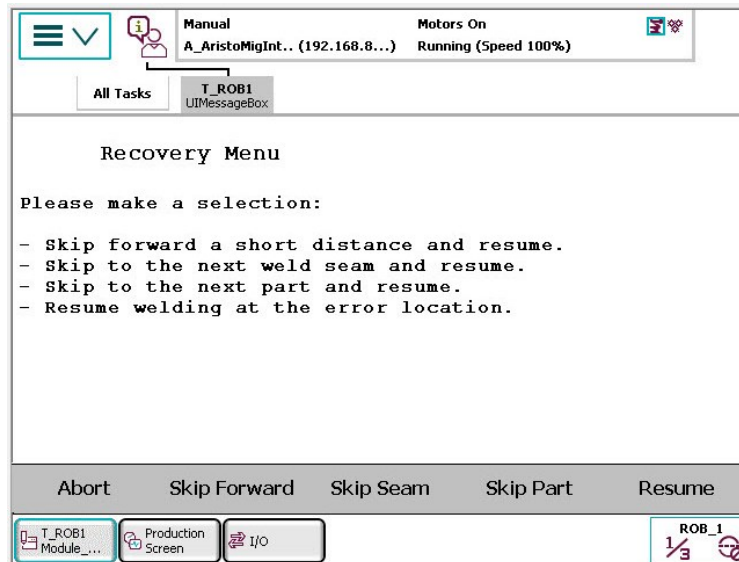
en1200000692

Example

```
RecoveryMenu;
```

The recovery menu is launched and waits for the user's response before allowing execution to resume.

Program execution

`RecoveryMenu` displays a modal dialog that requires user input before the executing thread will be allowed to continue.

Limitations

Backward step mode is not supported.

Syntax

```
RecoveryMenu ';'
```

Related information

Information	Described in
Set the recovery position	*RecoveryPosSet - Set the recovery position on page 166*
Reset the recovery position	*RecoveryPosReset - Reset the recovery position on page 169*

Continues on next page

Information	Described in
Configure the recovery menu	*Configure the recovery menu on page 63*
Configuring Weld Error Recovery	*Configuring Weld Error Recovery on page 61*

7.1.10 RecoveryMenuWR - Display the recovery menu

Usage

`RecoveryMenuWR` is used by custom welding error handlers to display the *Weld Repair* recovery menu. The selection made by the user will be stored internally, and will be referenced by *Weld Error Recovery* when *RobotWare Arc* attempts to re-ignite the arc.
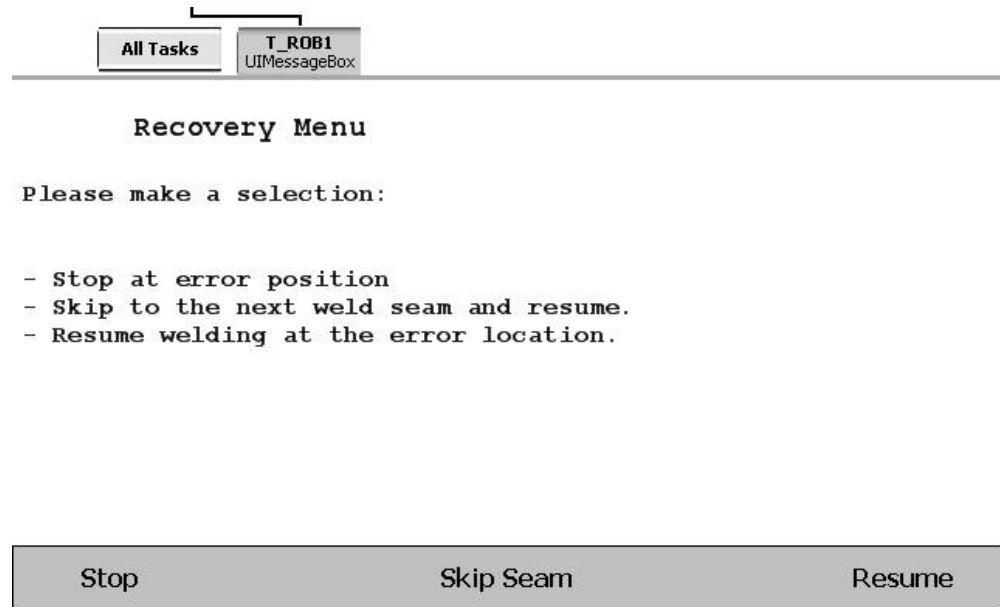
en1300000301

Basic examples

The following example illustrates the instruction `RecoveryMenuWR`.

Example 1

```
RecoveryMenuWR;
```

The recovery menu is displayed and the system waits for response from the user before allowing execution to resume.

Program execution

The recovery menu displays a modal dialog that requires user input before the execution will be allowed to continue.

Limitations

Backward step mode is not supported.

Syntax

```
RecoveryMenuWR ';'
```

Continues on next page

Related information

For information about	See
The instruction `RecoveryMenu`	*RecoveryMenu - Display the recovery menu on page 162*

7.1.11 RecoveryPosSet - Set the recovery position

Usage

`RecoveryPosSet` sets the recovery position, starts recording the robot path and enables the Escape function in the Error Menu. The internal path recorder will store path information during execution of the RAPID program. If an error occurs during the weld seam, the Error Menu will display an Escape option.
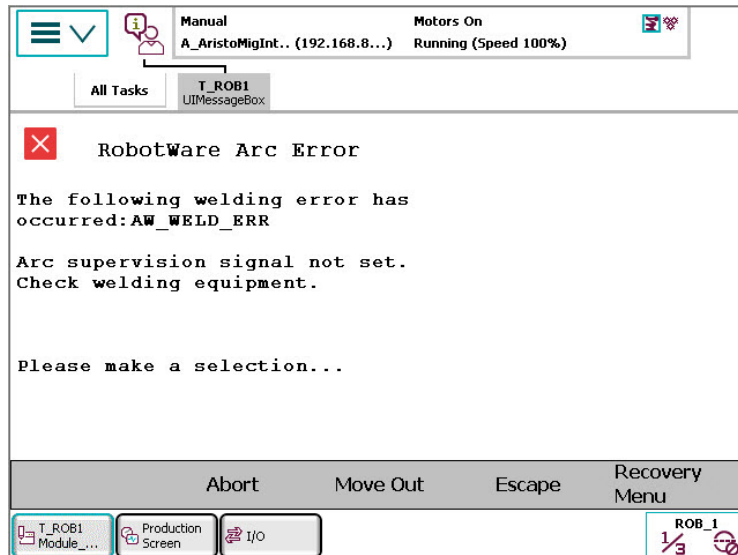
en1200000693

Pressing Escape causes the robot to retrace its path to the recovery position set by the `RecoveryPosSet` instruction. An optional service routine can be executed after the recovery position has been reached.

Example

```
RecoveryPosSet\ServRoutine:="ServiceRoutine";
```

The path recorder is started and the recovery point (the instruction's position in the RAPID program) is set. After backing up to the recovery position, the service routine *ServiceRoutine* is executed.

Limitations

There is a limitations to the use of `RecoveryPosSet`. The Pathrecorder can not be turned on with `RecoveryPosSet` before a `WaitSyncTask` instruction, that is the robot can never escape past a `WaitSyncTask` instruction. Therefore, make sure that `RecoveryPosSet` is always used after the `WaitSyncTask` instruction in the RAPID program.

Arguments

```
RecoveryPosSet [\ServRoutine]
```

`ServRoutine`

Data type: `string`

Continues on next page

Using the `ServRoutine` argument will extend the Weld Error Recovery escape functionality. The Service Routine is a user-defined procedure that is launched after the robot retraces a recorded path back to a recovery position. The routine may be used to move the robot from the recovery position to a service location, or any other behavior that can be implemented in RAPID.

Program execution

When the path recorder is ordered to start, the robot path will be recorded internally in the robot controller. At welding error the recorded sequence of program positions can be traversed backwards by selecting the Escape option from the Error Menu, causing the robot to move backwards along its executed path to the recovery position.

Recovery positions may be set at any point in a weld sequence. In some cases it may be necessary to have an alternate recovery position that is set mid-weld. This is perfectly ok.

Example

```
PROC MyWeld()
  MoveJ pSafe,vmax,z10,tWeldGun;
  RecoveryPosSet\ServRoutine:="ServiceRoutine";
  MoveJ *,vmax,z10,tWeldGun;
  ArcLStart *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
  ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
  SetDO doClamp,high;
  RecoveryPosSet;
  ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
  ArcLEnd *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
  MoveJ *,vmax,z10,tWeldGun;
  MoveJ *,vmax,z10,tWeldGun;
  RecoveryPosReset;
ENDPROC
PROC ServiceRoutine()
  MoveJ *,vmax,z10,tool0;
  MoveL pService,vmax,z10,tool0;
  RecoveryMenu;
  MoveL *,vmax,z10,tool0;
ENDPROC
```

Syntax

```
RecoveryPosSet
  ['\' ServRoutine ':=' <expression (IN) of string>]';'
```

Related information

Information	Described in
Reset the recovery position	*RecoveryPosReset - Reset the recovery position on page 169*
Display the recovery menu	*RecoveryMenu - Display the recovery menu on page 162*

Continues on next page

7 RAPID reference

Information	Described in
Configure the recovery menu	*Configure the recovery menu on page 63*
Configure weld error recovery	*Configuring Weld Error Recovery on page 61*

7.1.12 RecoveryPosReset - Reset the recovery position

Usage

`RecoveryPosReset` resets the recovery position, stops recording the robot path and the service routine is cleared.

Example

```
RecoveryPosReset;
```

The path recorder is stopped and the recovery position is reset. If a service routine was active it is cleared.

Program execution

This instruction should be used at the end of the weld sequence to ensure that the path recorder is stopped and cleared before starting a new weld sequence. A failure to do so could result in undesirable results, as an old recovery set point could remain active during a new weld sequence.

Example

```
PROC MyWeld()
  MoveJ pSafe,vmax,z10,tWeldGun;
  RecoveryPosSet\ServRoutine:="ServiceRoutine";
  MoveJ *,vmax,z10,tWeldGun;
  ArcLStart *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
  ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
  SetDO doClamp,high;
  RecoveryPosSet;
  ArcL *,v500,sm1,wd1\Weave:=wv1,z10,tWeldGun;
  ArcLEnd *,v500,sm1,wd1\Weave:=wv1,fine,tWeldGun;
  MoveJ *,vmax,z10,tWeldGun;
  MoveJ *,vmax,z10,tWeldGun;
  RecoveryPosReset;
ENDPROC
PROC ServiceRoutine()
  MoveJ *,vmax,z10,tool0;
  MoveL pService,vmax,z10,tool0;
  RecoveryMenu;
  MoveL *,vmax,z10,tool0;
ENDPROC
```

Syntax

```
RecoveryPosReset ';'
```

Related information

Information	Described in
Set the recovery position	*RecoveryPosSet - Set the recovery position on page 166*
Display the recovery menu	*RecoveryMenu - Display the recovery menu on page 162*

Continues on next page

7.1.12 RecoveryPosReset - Reset the recovery position
RobotWare Arc
Continued

Information	Described in
Configure the recovery menu	*Configure the recovery menu on page 63*
Configure weld error recovery	*Configuring Weld Error Recovery on page 61*

7.1.13 SetWRProcName - Set name of process to re-execute

Usage

`SetWRProcName` is used to inform *RobotWare Arc* which welding procedure needs to be re-executed by the weld repair function. The optional switch `\Flexpositioner` should be used in a FlexPositioner setup in the non-welding robot to inform WeldRepair that the task is not welding.

Basic examples

The following example illustrates the instruction `SetWRProcName`.

Example 1

```
SetWRProcName "Weldseam_1";
```

The RAPID procedure `Weldseam_1` is re-executed if the weld repair function is active.

Arguments

```
SetWRProcName st1 [\FlexPositioner]
```

`st1`

Data type: `string`

The name of the welding procedure that should be re-executed.

`[\FlexPositioner]`

Data type: `switch`

The robot/task is considered to be a non-welding robot if this argument is present. Should be used in a FlexPositioner setup.

Syntax

```
SetWRProcName
  [st1 <expression (IN) of string>]
  ['\' FlexPositioner ] ';'
```

7.2 Data types

7.2.1 advSeamData - Advanced seam data

Usage

`advSeamData` is used to configure the restart behavior *Seam local*. Normally the restart behavior such as number of retries, restart distance, skip forward distance, and scrape start behavior is defined globally in the system parameters (PROC.cfg). These settings are used on all welds in the system. With `advSeamData` the behavior can be defined for each seam.

> **ℹ Note**
>
> `AdvSeamData` can only be used if the system is configured in *Semi Automatic Mode 2*.

Basic examples

The following example illustrates the data type `advSeamData`.

Example 1

```
PROC Weld_1()
  SetWRProcName "Weld_1";

  SyncMoveOn sync001, allTasks;
  MoveL p10\ID:=10, vmax, z10, tWeldGun;

  RecoveryPosSet\ServRoutine:="mvToService";
  ArcLStart p20\ID:=20, v100, sm1\AdvData:=adv1, wd1, fine,
      tWeldGun;
  ArcL p30\ID:=30, v100, sm1, wd1, z10, tWeldGun;
  ArcLEnd p40\ID:=40, v100, sm1, wd1, fine, tWeldGun;
  RecoveryPosReset;

  MoveJ p50\ID:=50, vmax, z10, tWeldGun;
  SyncMoveOff sync_testblech_2;
ENDPROC
```

Component group: ErrorFunc

`RetractWire`

Data type: `bool`

Specifies if the welding wire should be retracted after the configured number of retries is exceeded.

Default value: FALSE

`RetractTime`

Data type: `num`

Time in seconds for wire retract.

Continues on next page

Default value: FALSE

The maximum allowed value is 2.25

NuOfRetries

Data type: `num`

The number of automatic restart attempts per seam at welding interrupts.

Default value: 0

NuOfWeldErrors

Data type: `num`

Numbers of allowed weld errors per seam.

Default value: 0

SkipForwardDist

Data type: `num`

The distance in mm that the robot moves forward on the current seam relative to the position where it was interrupted.

Default value: 0

The maximum allowed value is 50

RestartDist

Data type: `num`

The distance that the robot reverses on the current seam relative to the position where it was interrupted.

Default value: 0

Component group: ScrapeFunc

ScrapeStart

Data type: `bool`

Specifies if the robot is to weave at the actual weld start (scrape start). This weaving is automatically interrupted when the arc is ignited.

Default value: FALSE

ScrapeDir

Data type: `num`

The angle of direction of the weave for a scrape start. It is specified in degrees, where 0 implies a weave that is carried out at a 90 degrees angle to the direction of the weld.

Default value: 0

ScrapeTime

Data type: `num`

The time (in seconds) it takes for a complete weave cycle for a scrape start.

Default value: 0.2

`ScrapeWidth`

>
> **Data type:** `num`
>
> **The width of the weave pattern for a scrape start.**
>
> **Default value: 0**

Limitations

> `AdvSeamData` **can only be used if the system is configured in** *Semi Automatic Mode*.

7.2.2 arcdata - Arc data

Usage and description

arcdata is a data structure which is a subdata component of seamdata and welddata. It contains components that are commonly used in both data types.

Components

sched (**schedule**)

> **Data type:** num
>
> The identity (expressed as a number) of weld programs to send to the welding equipment. This parameter is only available if schedule port type (see *System parameters on page 203*) is defined as 1 (Binary), or 2 (Pulse), or 3 (CAN).

mode

> **Data type:** num
>
> The identity (expressed as a number) of weld mode to send to the welding equipment.
>
> This parameter is only available if schedule port type (see *System parameters on page 203*) is defined as 2 (Pulse) or 3 (CAN).

voltage

> **Data type:** num
>
> The welding voltage (in Volt) during the weld phase.
>
> The value specified is scaled and sent to the corresponding analog output, in accordance with the setting in *Defining arc welding systems on page 204*. This parameter is only available if weld voltage (*System parameters on page 203*) is defined.

wirefeed

> **Data type:** num
>
> This parameter is only available, if wirefeed (*System parameters on page 203*) is defined. The feed speed of the weld electrode during the weld phase. The unit for arcdata components that specify a velocity, is defined by the parameter *Units*, see *The type Arc Robot Properties on page 210*.
>
> The value specified is scaled and sent to the corresponding analog output, in accordance with the setting in *Defining arc welding systems on page 204*.

control

> **Data type:** num
>
> Analog tuning value sent to certain welders.

current

> **Data type:** num
>
> The welding current (in Ampere) during the weld phase.

7.2.2 arcdata - Arc data
RobotWare Arc
Continued

The value specified is scaled and sent to the corresponding analog output, in accordance with the setting in *Defining arc welding systems on page 204*. This parameter is only available if current (*System parameters on page 203*) is defined.

voltage2

Data type: `num`

The welding voltage (in Volt) during the weld phase. Used in a TwinWire setup.

The value specified is scaled and sent to the corresponding analog output, in accordance with the setting in *Defining arc welding systems on page 204*. This parameter is only available if weld voltage (*System parameters on page 203*) is defined.

control2

Data type: `num`

Analog tuning value sent to certain welders. Used in a TwinWire setup.

Structure

```
<data object of arcdata>
  <sched of num>
  <mode of num>
  <voltage of num>
  <wirefeed of num>
  <control of num>
  <current of num>
  <voltage2 of num>
  <wirefeed2 of num>
  <control2 of num>
  <track_reference of num>
```

Related information

Information	Described in
Seam data	*seamdata - Seam data on page 178*
Installation parameters for welding	*System parameters on page 203*
Process phases and time diagrams	*Programming on page 21*
Circular arc welding instructions	*ArcC, ArcC1, ArcC2 - Arc welding with circular motion on page 107*
Linear arc welding instructions	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*

7.2.3 flystartdata - Flying start data

Usage and description

`flystartdata` is used to setup the needed parameters for a weld with a flying start.

Components

`active`

Data type: `bool`

When `active` is `TRUE`, the ignition of the weld is done with a moving TCP (flying start).

When `active` is `FALSE`, the ignition is done with a non moving TCP and the zone is executed as a fine point. This makes it easy to test flying start without having to change the zone parameter.

Flying start cannot be used it in combination with *Ignition Movement Delay* or a *Scrape Start*

`superv_distance`

Data type: `num`

`superv_distance` sets the distance from the zone center to where an ignition must occur. If no ignition has occurred when the robot reaches this distance, the system stops and enter its error handler. The required action is then based on the setup of the error handler.

Structure

```
<data object of flystartdata>
  <active of bool>
  <superv_distance of num>
```

Related information

Information	Described in
Arc welding start with circular motion	*ArcCStart, ArcC1Start, ArcC2Start - Arc welding start with circular motion on page 124*
Arc welding start with linear motion	*ArcLStart, ArcL1Start, ArcL2Start - Arc welding start with linear motion on page 149*

7.2.4 seamdata - Seam data

Usage and description

seamdata is used to control the start and end of the weld. seamdata is also used if the process is restarted after a welding operation has been interrupted.

The actual weld phase is controlled using welddata, see *welddata - Weld data on page 197*.

seamdata describes data, which, as a rule, can be maintained unaltered during a whole seam and often also during welding several seams. seamdata is used during the start phase of a welding operation (ignition, heating after ignition) and during the final phase of the weld. seamdata is included in all arc welding instructions to facilitate controlled start and end phases independent of where interrupts or restarts might occur.

All voltages can be expressed in two ways (determined by the welding equipment):

- As absolute values (only positive values are used in this case).
- As corrections of values set in the process equipment (both positive and negative values can be used in this case).

Feeding the weld electrode in this section refers to MIG/MAG welding. For TIG welding the following applies:

- A cold wire is supplied to the wire feed.
- The necessary welding current reference value can be connected to any of the three analog outputs that are not used. The Welding voltage reference is not used.

Continues on next page

Welding sequence
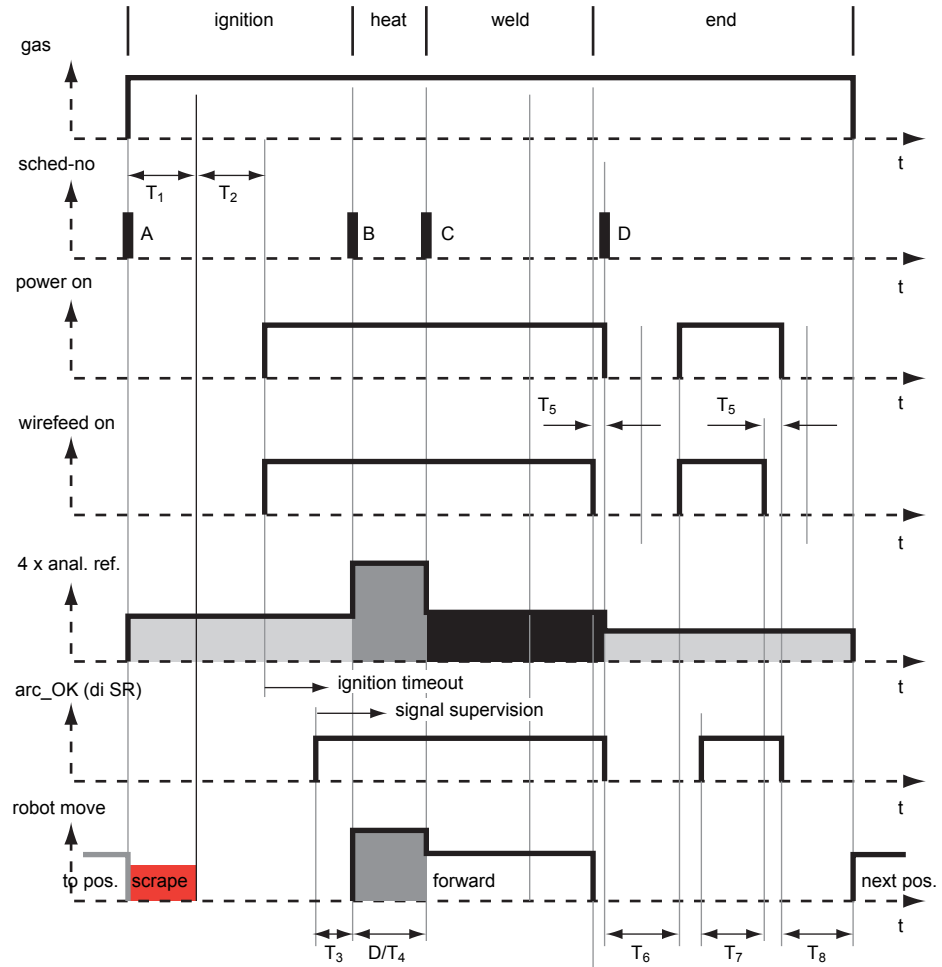
xx1200000713

T_1	**maximum** `gas_purge/arc_preset` **time**
T_2	`gas_preflow` **time**
T_3	`ignition_movement_delay` **time**
D/T_4	**heating distance/time**
T_5	**burnback time**
T_6	**maximum cooling/**`arc_preset` **time**
T_7	**filling time**
T_8	**maximum cooling/**`gas_postflow` **time**
A	`ign_sched`
B	`heat_sched`
C	`weld_sched`
D	`fill_sched`

Component group: Ignition

`purge_time`

> **Data type:** `num`
>
> The time (in seconds) it takes to fill gas lines and the welding gun with protective gas, so called "gas purging". The first weld instruction is an `ArcLStart` or `ArcLCStart`, the gas flow is activated at the specified gas purge time before the programmed position is reached.
>
> If the positioning time to the start position of the weld is shorter than the gas purge time, or if the `ArcLStart` or `ArcCStart` instruction is not used, the robot waits in the weld start position until the gas purge time has expired.

`preflow_time`

> **Data type:** `num`
>
> The time (in seconds) it takes to preflow the weld object with protective gas, so called "gas preflowing".
>
> The robot is stationary in position during this time before the arc is ignited.
>
> If a schedule based welder is used, the ignition schedule is sent to the welder at the same time as the arc is ignited. This is in most cases too late for the welder. Setting the `preflow_time` to for example 0.2 seconds, will give the welder some time to react on the schedule sent to it.

`ign_arc`

> **Data type:** `arcdata`
>
> Weld parameters during the ignition phase. See definition of `arcdata`, *arcdata - Arc data on page 175*.

`ign_move_delay` (ignition movement delay)

> **Data type:** `num`
>
> The delay (in seconds) from the time the arc is considered stable at ignition until the heating phase is started. The ignition references remain valid during the ignition movement delay.

`scrape_start` (scrape start type)

> **Data type:** `num`
>
> Type of scrape at weld start. Scrape type at restart will not be affected. It will always be *weaving scrape*.
>
> Scrape types:
>
> - 0 - No scrape. No scrape will occur at weld start.
> - 1 - Weaving scrape.

Component group: Heat

`heat_speed`

> **Data type:** `num`
>
> The welding speed during heating at the start of the weld phase.

Continues on next page

The unit for `seamdata` components that specify a velocity, is defined by parameter *Units*, see *The type Arc Robot Properties on page 210*.

heat_time

Data type: `num`

The heating time (in seconds) at the start of the weld phase.

`Heat_time` is only used during timed positioning and when `heat_distance` or `heat_speed` equal zero.

heat_distance

Data type: `num`

The distance along which heat data must be active at the start of the weld.

The unit for `seamdata` components that specify a distance, is defined by parameter *Units*, see *The type Arc Robot Properties on page 210*.

heat_arc

Data type: `arcdata`

Weld parameters during the heat phase. See definition of *arcdata - Arc data on page 175*.

Component group: End

cool_time (cooling time)

Data type: `num`

The time (in seconds) during which the process is closed before other terminating activities (filling) take place.

fill_time

Data type: `num`

The crater-filling time (in seconds) at the end phase of the weld.

This component needs *crater fill* to be set for the Arc Welding function.

fill_arc

Data type: `arcdata`

Weld parameters during the filling phase. See definition of `arcdata`, *arcdata - Arc data on page 175*.

bback_time (burnback time)

Data type: `num`

The time (in seconds) during which the weld electrode is burnt back when electrode feeding has stopped. This to prevent the electrode getting stuck to the hardening weld when a MIG/ MAG process is switched off. Burnback time is used twice in the end phase; first when the weld phase is being finished, the second time after crater-filling. This component needs *burnback* to be set for the Arc Welding function.

rback_time (rollback time)

Data type: `num`

Continues on next page

The time (in seconds) during which a cold wire is rolled back after the power source has been switched off. This to prevent the wire getting stuck to the hardening weld when a TIG process is switched off. This component needs *rollback* to be set for the Arc Welding function.

`bback_arc`

Data type: `arcdata`

Weld parameters during the burnback and rollback phase. See definition of `arcdata`, *arcdata - Arc data on page 175*.

`postflow_time`

Data type: `num`

The time (in seconds) required for purging with protective gas after the end of a process. The purpose of gas postflow is to prevent the weld electrode and the seam from oxidizing during cooling.

Limitation

There is no component for rollback wire feed in `seamdata`.

However, the functionality can be achieved by using the wire feed component in the burnback part of `seamdata` (`bback_arc`).

To activate rollback functionality with rollback wire feed, the following needs to be fulfilled:

- *Rollback On* needs to be activated in the topic *Process* (configuration).
- *Rollback Wirefeed On* needs to be activated in the topic *Process* (configuration).
- *Burnback On* needs to be activated in the topic *Process* (configuration).
- *Burnback Voltage On* needs to be activated in the topic *Process* (configuration).

If this is done, rollback time (`rback_time`) in `seamdata` and wirefeed component in `bback_arc` will be visible.

Structure

```
<data object of seamdata>
  <purge_time of num>
  <preflow_time of num>
  <startcurrent_time of num>
  <startcurrent_slope of num>
  <ign_arc of arcdata>
  <ign_move_delay of num>
  <scrape_start of num>
  <heat_speed of num>
  <heat_time of num>
  <heat_distance of num>
  <heat_arc of arcdata>
  <endcurrent_time of num>
  <endcurrent_slope of num>
  <cool_time of num>
```

Continues on next page

```
<fill_time of num>
<fill_arc of arcdata>
<bback_time of num>
<rback_time of num>
<bback_arc of arcdata>
<postflow_time of num>
```

Related information

Information	Described in
Weld data	*welddata - Weld data on page 197*
Arc data	*arcdata - Arc data on page 175*
Installation parameters for welding	*System parameters on page 203*
Process phases and time diagrams	*Programming on page 21*
Circular arc welding instruction	*ArcC, ArcC1, ArcC2 - Arc welding with circular motion on page 107*
Linear arc welding instruction	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*

7.2.5 trackdata - Seam tracking data

Usage and 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. Using instructions with different track data, it is thus possible to achieve optimum position control along an entire seam. The optional `trackdata` argument must be used during the whole weldseam, that is, from the `ArcXStart` to the `ArcXEnd` instruction.

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.

The function best suited for welding applications with long strait seams with speeds lower than 20 mm/s and orientation errors less than 10 deg.

> **ℹ️ Note**
>
> Some of the components of `trackdata` depend on the configuration of the robot. `trackdata` will only include component appropriate for the selected sensor type.

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`

For Optical:

- If the TCP offset due to path corrections is more than `max_corr` and `Max Correction Warning` was set in the `Optical Sensor Properties`, the robot will continue its path but the applied path correction will not exceed `max_corr`. If `Max Correction Warning` was not set, a track error is reported and program execution is stopped.

For WeldGuide:

- The `max_corr` component 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.

Continues on next page

arctrack

> **Data type:** `arctrackdata`
>
> Track data with parameters for non-optical trackers (AWC).

opttrack

> **Data type:** `opttrackdata`
>
> Track data with parameters for optical trackers (laser trackers).

Structure of trackdata

```
<data object of trackdata>
<track_system of num>
<store_path of bool>
<max_corr of num>
<arctrack of arctrackdata>
<opttrack of opttrackdata>
```

Components of arctrackdata

track_type

> **Data type:** `num`
>
> The parameter defines what type of tracking to be performed. The tracking types available are: Center line, Adaptive, Right side, Left side and Height only. In order for the robot to track, the optional argument \Track must be added to each weld instruction in the program.

Value	Description
0	Center line tracking
1	Adaptive tracking
2	Single side tracking (Right)
3	Single side tracking (Left)
4	Height only tracking (Constant stick-out length is kept) (a value of 4 is reserved for TIG welding only)
5	Height only tracking (Constant stick-out length is kept)

gain_y

> **Data type:** `num`
>
> The `gain_y` parameters define how big of a correction is sent to the robot. The higher the number the faster the system corrects. The range of this parameter is from 1 to 100. Initial starting values for this parameter depend 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` parameters define how big of a correction is sent to the robot. The higher the number the faster the system corrects. The range of this parameter is from 1 to 100. Initial starting values for this parameter depend on weave size. Start with 30 for most weave widths and 5 for very small weave widths.

7 RAPID reference

7.2.5 trackdata - Seam tracking data
RobotWare Arc
Continued

weld_penetration

Data type: num

Defines how hard the system should bite in to the sidewall of the parent material in percentage of penetration. Although always present, the WG uses this parameter only during adaptive, right and left side tracking. Range, about 1-4.

track_bias

Data type: num

The bias parameter is used to move the TCP in the seam y direction to bias one side of the joint or the other. The range for this parameter is from -30 to +30 where +30 is the highest amount of bias achievable in the plus Y direction of the seam coordinates. Used in center line-tracking only.

min_weave

Data type: num

This is the minimum weave width setting that system is allowed to change during adaptive tracking. Minimum value need to be > 2mm.

max_weave

Data type: num

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

max_speed

Data type: num

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

min_speed

Data type: num

This is the maximum travel speed setting that system is allowed to change during adaptive tracking. Minimum value need to be > 2mm/s.

Structure of arctrackdata

```
<data object of arctrackdata>
<track_type of num>
<gain_y of num>
<gain_z of num>
<weld_penetration of num>
<track_bias of num>
<min_weave of num>
<max_weave of num>
<min_speed of num>
<max_speed of num>
```

Components of opttrackdata

joint_no **(joint number)**

Data type: num

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

The identity (expressed as a number) of path correction programs to send to the sensor equipment.

filter **(path correction filter)**

Data type: num

The filter component defines the time constant of a low pass filter applied to path corrections. The component may be set to values from 1 to 10 where 1 gives the fastest response to path errors detected by the sensor.

seamoffs_y, seamoffs_z **(seam offset y and z)**

Data type: num

The seam offset components are used to add offsets to the path generated by the sensor input. If for instance the sensor considers the upper edge of a lap joint to be the correct seam position, as indicated in the figure below, the seam offsets may be used to correct the path.

Path generated by sensor

Corrected path
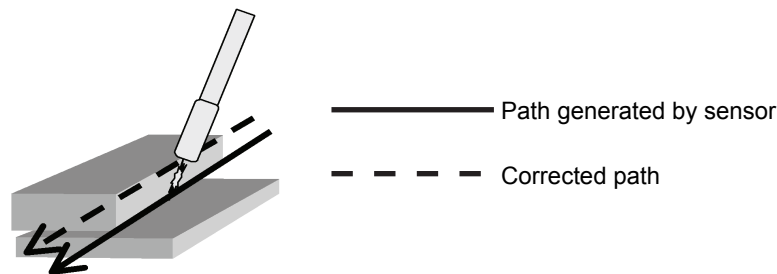
xx1200000731

The correction is defined in a seam related right hand coordinate system with the following definition:
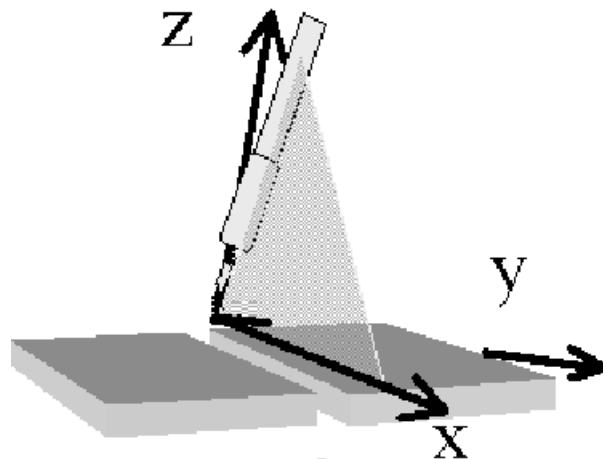
xx1200000732

- The x vector is parallel to the path tangent.
- The y vector is perpendicular to a plane through the x vector and the tool z-vector.
- The z vector is perpendicular to a plane through the x and y vectors.

Continues on next page

7.2.5 trackdata - Seam tracking data
RobotWare Arc
Continued

seamadapt_y, seamadapt_z **(adaptive seam offset y and z)**

> **Data type:** num

> The seamadapt components are similar to the seam offset components. The magnitudes of the offsets are however not given as fixed values. The offsets are calculated as the measured seam gap multiplied by the seamadapt values.

> The components are used to adaptively offset the torch with respect to the seam to optimize the welding process for different gap sizes.

> The components are supported for lap joints.

track_mode **(tracking mode for Laser Tracker)**

> **Data type:** num

> With the track_mode component it is possible to selectively influence the tracking behavior of a laser tracker.

Value	Track Mode
0	Normal tracking. y- and z-corrections are both taken into account
1	Tracking as if y-corrections sent by the Laser Tracker were zero. z-corrections are taken into account. [i]
2	Tracking as if z-corrections sent by the Laser Tracker were zero. y-corrections are taken into account. [i]
3	Tracking as if y- and z-corrections sent by the Laser Tracker were zero. [i]
4	y-correction switched off totally, that is, the correction of the y component is set to zero before it is sent to the robot. z-correction is taken into account. [ii]
5	z-correction switched off totally, that is, the correction of the z component is set to zero before it is sent to the robot. y-correction is taken into account. [ii]
6	y- and z-corrections are switched off totally, that is, the correction of the y and the z component is set to zero before it is sent to the robot. [ii]
7	y-correction is faded out, that is, the TCP returns ramped to the programmed y component of the path. z-correction is active.
8	z-correction is faded out, that is, the TCP returns ramped to the programmed z component of the path. y-correction is active.
9	y- and z-corrections are faded out, that is, the TCP returns ramped to the programmed path.
10	y-correction is faded in, that is, the TCP returns ramped to the programmed y component of the path. z-correction is active.
11	z-correction is faded in, that is, the TCP returns ramped to the programmed z component of the path. y-correction is active.
12	y- and z-corrections are faded in, that is, the TCP returns ramped to the programmed path.
13	Tracking as if y-corrections sent by the Laser Tracker were zero. z-corrections are taken into account. The difference to track_mode 1 is, that the mode starts at the robot TCP position and not at the sensor TCP position. [i]

Continues on next page

Value	Track Mode
14	Tracking as if **z-corrections** sent by the Laser Tracker were zero. y-corrections are taken into account. The difference to `track_mode` 2 is that the mode starts at the robot TCP position and not at the sensor TCP position. [i]
15	Tracking as if **y- and z-corrections** sent by the Laser Tracker were zero. The difference to `track_mode` 3 is that the mode starts at the robot TCP position and not at the sensor TCP position. [i]

[i] For `track_mode` 1, 2, or 3, the accumulated correction from the previous arc welding instruction in the same seam will be preserved for y and/or z and passed on to the next arc welding instruction in the same seam.

[ii] For `track_mode` 4, 5, or 6, the sensor readings are accumulated even though y- and/or z-correction is set to zero before sending to the robot. That means, a 'dip' might occur in the beginning and in the end of the arc weld instruction.
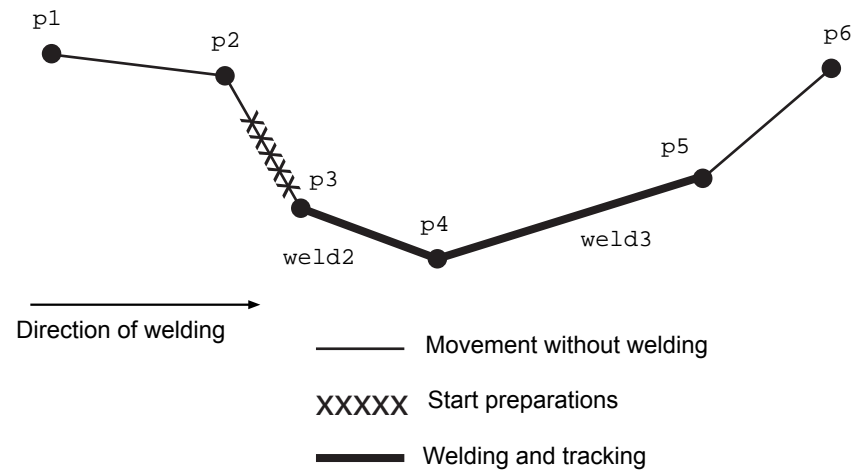
Structure of opttrackdata

```
<data object of opttrackdata>
<joint_no of num>
<filter of num>
<seamoffs_y of num>
<seamoffs_z of num>
<seamadapt_y of num>
<seamadapt_z of num>
<track_mode of num>
```

Example

xx1200000733

```
MoveJ p1, v100, z10, gun1;
MoveJ p2, v100, fine, gun1;
ArcLStart p3, v100, seam1, weld1, weave1, fine, gun1\Track:=track1;
ArcL p4, v100, seam1, weld2, weave1, z10, gun1\Track:=track2;
ArcLEnd p5, v100, seam1,weld3, weave3, fine, gun1\Track:=track3;
MoveJ p6, v100, z10, gun1;
```

Continued

7.2.6 weavedata - Weave data

Usage and description

weavedata is used to define any weaving carried out during arc welding. Weaving can be used during the heat and weld phases of a seam.

Weaving is a movement, superimposed on the basic path of the process. That means, the weld speed is kept as defined in welddata and the TCP speed is increased unless the physical robot limitations are reached. There are four types of weaving patterns, see *Types of weave shape on page 190*.

- zigzag
- V-shaped
- triangular weaving
- circular weaving

All weave data components apply to both the heat phase and the weld phase.

The unit for weave data components that specify a distance, is defined by the parameter *Units*, see *The type Arc Robot Properties on page 210*.

> **Note**
>
> Some of the components of weavedata depend on the configuration of the robot. If a given feature is omitted, the corresponding component is left out from the weavedata. The conditions that must be met for components to exist are described in *System parameters on page 203*, and components of *weavedata - Weave data on page 190*.

Components

weave_shape (weld weave shape)

Data type: num

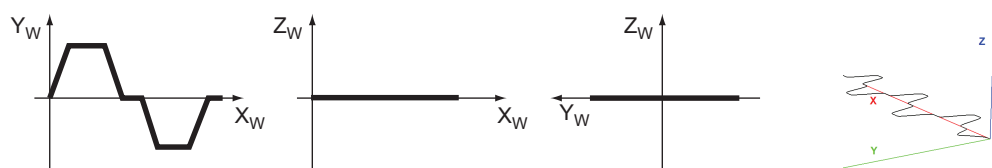The shape of the weaving pattern in the weld phase as illustrated in the following figures.

> **Note**
>
> The path coordinate system is shown with x-axis in path direction.
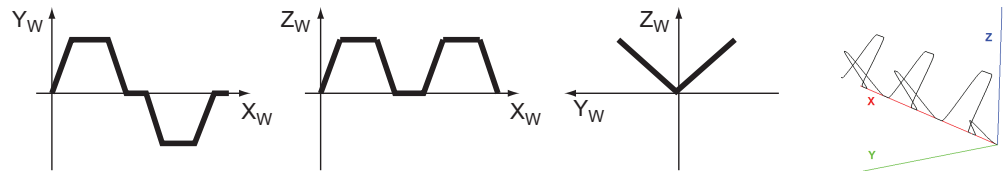
Types of weave shape

0 - No weaving.

1 - Zigzag weaving results in a weaving horizontal to the seam.
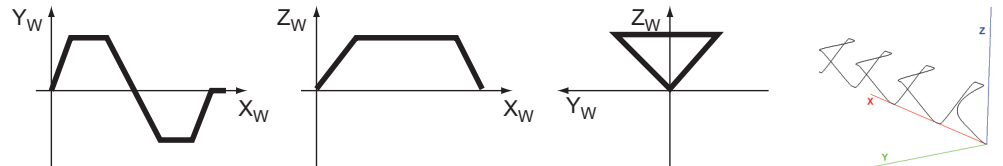

xx1200000714

Continues on next page

2 - V-shaped weaving results in weaving in the shape of a "V", vertical to the seam.
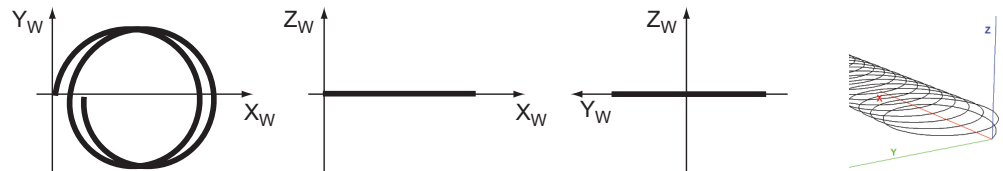

xx1200000715

3 - Triangular weaving results in a triangular shape, vertical to the seam.

xx1200000716

4 - Circular weaving results in a circular shape, vertical to the seam.
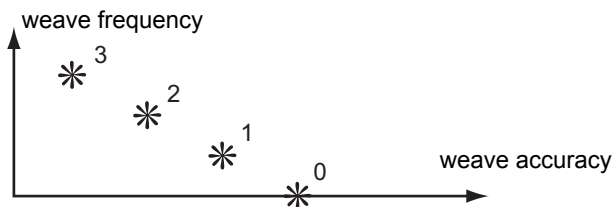

xx1200000717

The type of weaving in the weld phase

weave_type **(weld weave interpolation type)**

Data type: num

Specified value	Weaving type
0	Geometric weaving. All axes are used during weaving.
1	Wrist weaving.
2	Rapid weaving. Axes 1, 2, and 3 used.
3	Rapid weaving. Axes 4, 5, and 6 used.

xx1200000718

weave_length

Data type: num

There are two meanings of the weave_length component: length and frequency.
For length the component weave_length is defined as a length of the weaving
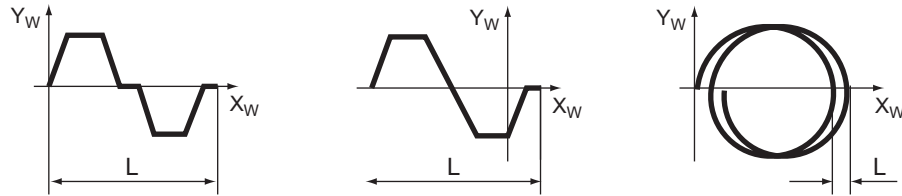
Continues on next page

cycle in the weld phase for weaving types 0 and 1, see the following figure. See the measurement L in the following figure.

For circular weaving the length attribute defines the rotation frequency of the TCP. The TCP rotates left with a positive length value, and right with a negative length value. L is calculated as `L = weld_speed / weave_length`.

xx1200000719

For frequency the component `weave_length` is defined as the frequency of the weaving cycle in the weld phase for weaving types 2 and 3, see the following figure. For circular weaving the `weave_length` argument defines the weaving frequency (in Hz).

The TCP rotates left with a positive `weave_length` value, and right with a negative `weave_length` value.
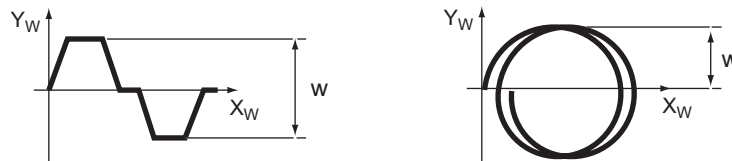

$$f = \frac{1}{T}$$
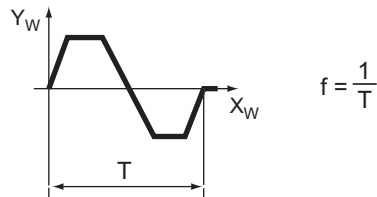
xx1200000720

`weave_width`

Data type: `num`

For circular weaving, width is the radius of the circle. For all other weaving shapes, width is the total amplitude of the weaving pattern. See the measurement W in the following figure.

xx1200000721

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

`weave_height`

Data type: `num`

The height (H) of the weaving pattern during V-shaped and triangular weaving, see the following figure. Not available for circular weaving.
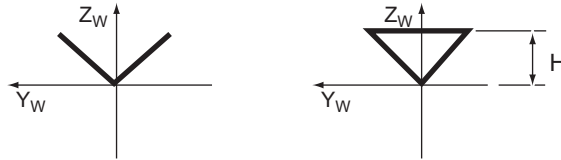
xx1200000722

`dwell_left`

Data type: `num`

The length of the dwell (DL) used to force the TCP to move only in the direction of the seam at the left turning point of the weave. Not available for circular weaving.

xx1200000723

A	Zigzag and V-shaped weaving
B	Triangular weaving

`dwell_center`

Data type: `num`

The length of the dwell (DC) used to force the TCP to move only in the direction of the seam at the center point of the weave. Not available for circular weaving.

xx1200000724

A	Zigzag and V-shaped weaving
B	Triangular weaving

Continues on next page

7.2.6 weavedata - Weave data
RobotWare Arc
Continued

`dwell_right`

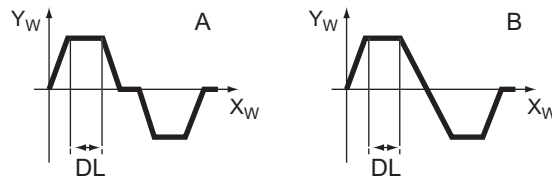> **Data type:** `num`
>
> **The length of the dwell (DR) used to force the TCP to move only in the direction of the seam at the right turning point of the weave. Not available for circular weaving.**
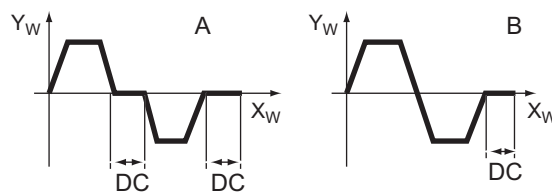>
>
>
> xx1200000725
>
> | A | Zigzag and V-shaped weaving |
> |---|---|
> | B | Triangular weaving |
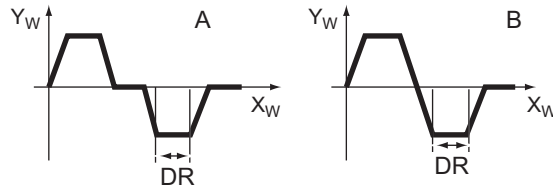
`weave_dir` **(weave direction angle)**

> **Data type:** `num`
>
> **The weave direction angle horizontal to the seam. An angle of zero degrees results in a weave vertical to the seam.**
>
>
>
> xx1200000726
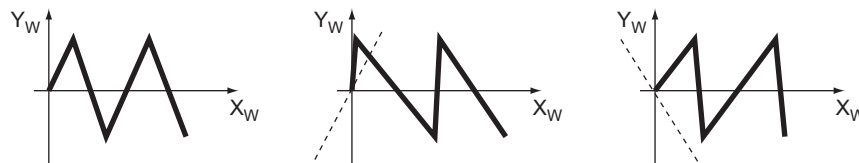
`weave_tilt` **(weave tilt angle)**

> **Data type:** `num`
>
> **The weave tilt angle, vertical to the seam. An angle of zero degrees results in a weave which is vertical to the seam.**
>
>
>
> xx1200000727
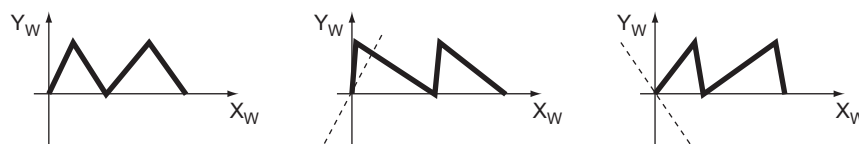
`weave_ori` **(weave orientation angle)**

> **Data type:** `num`
>
> **The weave orientation angle, horizontal-vertical to the seam. An angle of zero degrees results in symmetrical weaving.**
>
>
>
> xx1200000728

Continues on next page

`weave_bias` (**weave center bias**)

Data type: `num`

The bias horizontal to the weaving pattern. The bias can only be specified for zig-zag weaving and may not be greater than half the width of the weave. Not available for circular weaving.

The following figure shows zigzag weaving with and without bias (B).
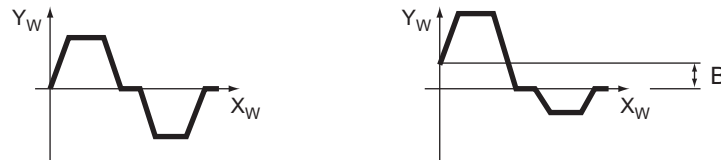
xx1200000729

`org_weave_width`

Data type: `num`

This component holds the last saved original value for the tuning function. It can be used for a quick restore of a changed value to the original value.

`org_weave_height`

Data type: `num`
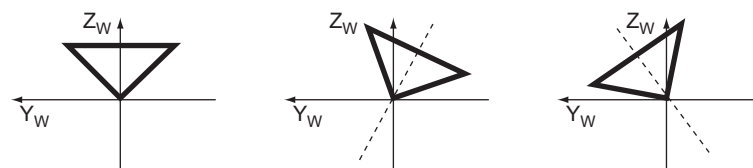
This component holds the last saved original value for the tuning function. It can be used for a quick restore of a changed value to the original value.

`org_weave_bias`

Data type: `num`

This component holds the last saved original value for the tuning function. It can be used for a quick restore of a changed value to the original value.

Limitations

The maximum weaving frequency is 2 Hz.

The inclination of the weaving pattern must not exceed the ratio 1:10 (84 degrees). See the following figure.
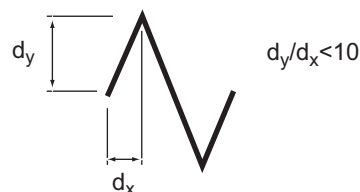
xx1200000730

Change of `weave_type` in `weavedata` is not possible in zone points, only in fine points. This is the behavior for both spline & decbuf interpolator. All robots, that use *TrueMove* or *QuickMove* second generation have the following changed behavior for the different weaving types available in RW Arc, compared to *TrueMove* or *QuickMove* first generation:

- Geometric weaving - There is no change.

Continues on next page

- Wrist weaving - uses mainly the wrist axes (4, 5, and 6) but small corrections can also be added to the main axes to be able to keep the pattern in the desired plane.
- Rapid weaving - In *TrueMove* or *QuickMove* second generation both geometric weaving and wrist weaving have highly improved performance. Therefore Rapid weaving (both types) is not necessary as a special weaving type any more.

  Rapid weaving axis 1, 2, and 3 is the same as geometric weaving.

  Rapid weaving axis 4, 5, and 6 is the same as wrist weaving.

  The weaving types are still available for backward compatibility.

The system uses *TrueMove* or *QuickMove* second generation, if there is a switch `dyn_ipol_type 1` in MOC.cfg in the MOTION_PLANNER data (system parameters).

Structure

```
<data object of weavedata>
  <weave_shape of num>
  <weave_type of num>
  <weave_length of num>
  <weave_width of num>
  <weave_height of num>
  <dwell_left of num>
  <dwell_center of num>
  <dwell_right of num>
  <weave_dir of num>
  <weave_tilt of num>
  <weave_ori of num>
  <weave_bias of num>
  <org_weave_width of num>
  <org_weave_height of num>
  <org_weave_bias of num>
```

Related information

Information	Described in
Installation parameters for welding equipment and functions	*System parameters on page 203*
Process phases and timing schedules	*Programming on page 21*
Arc welding instructions	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133* *ArcC, ArcC1, ArcC2 - Arc welding with circular motion on page 107*

7.2.7 weldata - Weld data

Usage and description

`welddata` controls the weld during the weld phase, i.e. as long as the arc is established. Start, restart and end phases are controlled using *seamdata - Seam data on page 178*.

`welddata` describes data that normally vary along a seam. `welddata` used in a given instruction along a path affects the weld until the specified position is reached. By using instructions with different weld data, it is thus possible to achieve optimum control over the welding equipment along a seam. `welddata` affects the weld when fusion has been established (after heating) at the start of a process.

When using an `ArcLStart` or `ArcCStart` instruction, the arc is not ignited until the destination position is reached, which means that weld data does not have any effect on the weld in this instruction.

When using `ArcLStart`, the arc is not ignited until the destination position is reached, which means that weld data does not have any effect on the weld in this instruction.

When going from one arc welding instruction to another during a weld, the new weld data will be applied starting in the middle of the corner path.

All voltages can be expressed in two ways (determined by the welding equipment):

- As absolute values (only positive values are used in this case).
- As corrections of values set in the process equipment (both positive and negative values are used in this case).

Feeding the weld electrode in this section refers to MIG/MAG welding. For TIG welding, a cold wire is supplied to the wire feed. The necessary welding current reference value can be connected to any of the three analog outputs that are not used. The Welding voltage reference is not used.

Example

```
MoveJ p1, v100, z10, gun1;
MoveJ p2, v100, fine, gun1;
ArcLStart p3, v100, seam1, weld1 \Weave:=weave1, fine, gun1;
ArcL p4, v100, seam1, weld2 \Weave:=weave1, z10, gun1;
ArcLEnd p5, v100, seam1,weld3 \Weave:=weave3, fine, gun1;
MoveJ p6, v100, z10, gun1;
```

7.2.7 welddata - Weld data
RobotWare Arc
Continued
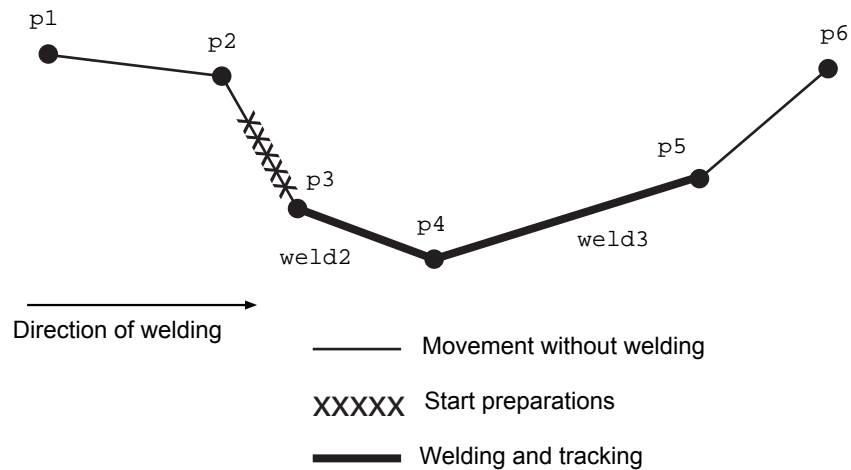
xx1200000733

Every welding instruction has different `welddata`. As the instruction `ArcLStart` is used in the first instruction, the first `welddata` is actually never used.

Welding sequence
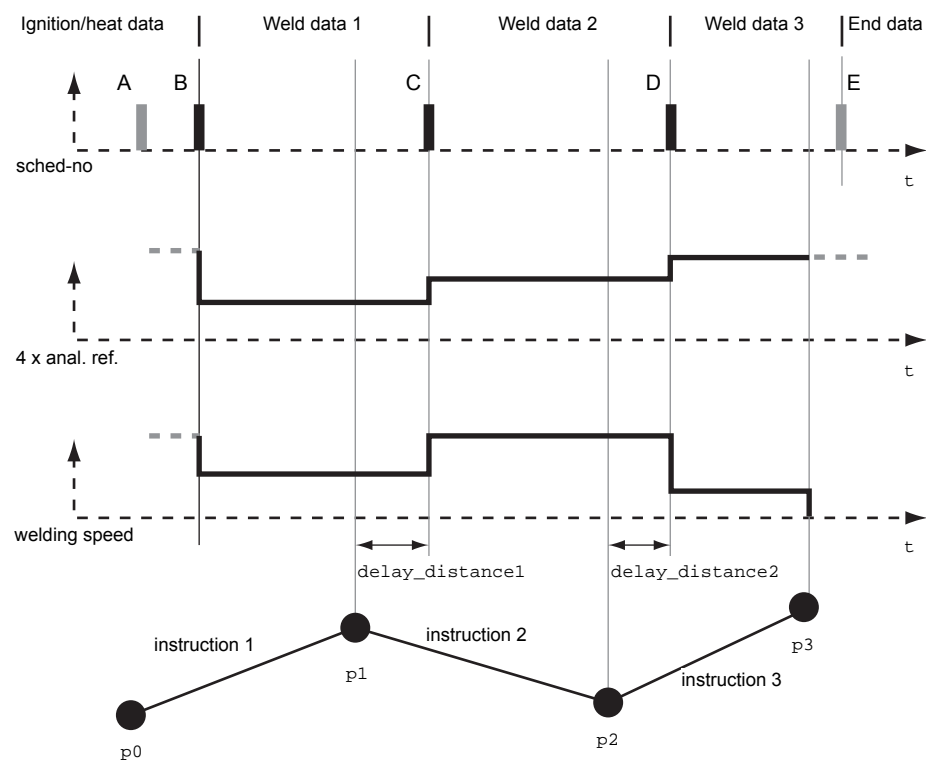
xx1200000734

Components

`weld_speed`

Data type: `num`

The desired welding speed.

Continues on next page

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

The unit for welddata components that specify a velocity, is defined by the parameter *Units*, see *The type Arc Robot Properties on page 210*.

If the movements of additional axes are coordinated, the welding speed is the relative speed between the tool and the object.

If the movements of additional axes are not coordinated, the welding speed is the TCP speed. The speed of the additional axes is then described in the instruction's speed data. The slowest axis determines the speed to enable all axes to reach the destination position at the same time

`org_weld_speed` (original weld speed)

Data type: `num`

The original weld speed during the weld phase. This parameter is visible if *override_on* is activated.

`main_arc`

Data type: `arcdata`

The main arc parameters during the weld phase. See definition of `arcdata` for more information.

`org_arc`

Data type: `arcdata`

The original weld parameters during the weld phase. See definition of `arcdata` for more information.

This parameter is visible if *override_on* is activated.

Component group: Override

This component group needs 'override' to be set in the Arc Welding function definition.

`org_weld_speed` (original weld speed)

Data type: `num`

The original weld speed during the weld phase. It is used internally by tuning functions.

`org_weld_voltage` (original weld voltage)

Data type: `num`

The original weld voltage during the weld phase. It is used internally by tuning functions.

`org_weld_wfeed` (original weld wirefeed speed)

Data type: `num`

The original weld wirefeed speed during the weld phase. It is used internally by tuning functions.

This parameter is only available, if wirefeed is defined. See *System parameters on page 203*.

Continues on next page

More examples

The type of weld shown in the following figure is desired, with a welding voltage of 30 V and a wire feed speed of 15 m/min. The welding speed is 20 mm/s.
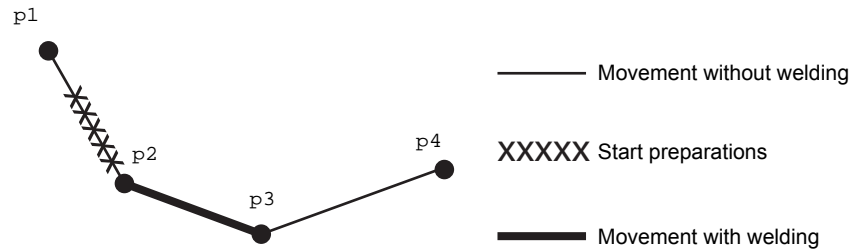
xx1200000735

```
PERS welddata weld1 :=
    [20,0,[0,0,30,250,0,0,0,0,0],[0,0,0,0,0,0,0,0,0]];
MoveJ p1, v100, z20, gun1;
ArcLStart p2, v100, seam1, weld1, fine, gun1;
ArcLEnd p3, v100, seam1, weld1, fine, gun1;
MoveJ p4, v100, z20, gun1;
```

The weld data values for a weld such as the one in the preceding figure are as follows:

Component	weld1	Description
weld_speed	20 mm/s	Speed in relation to the seam
weld_voltage	30 V	Sent to an analog output signal
weld_wirefeed	250 mm/s	Sent to an analog output signal

The weld schedule identity, weld voltage adjustment and weld current adjustment components are not active in this example.

The weld data argument does not have any effect in the `ArcLStart` instruction.

Structure

```
<data object of welddata>
  <weld_speed of num>
  <org_weld_speed of num>
  <main_arc of arcdata>
  <org_arc of arcdata>
```

Related information

Information	Described in
Seam data	*seamdata - Seam data on page 178*
Arc data	*arcdata - Arc data on page 175*
Installation parameters for welding	*System parameters on page 203*
Process phases and time diagrams	*Programming on page 21*

Continues on next page

Information	Described in
Circular arc welding instructions	*ArcC, ArcC1, ArcC2 - Arc welding with circular motion on page 107*
Linear arc welding instructions	*ArcL, ArcL1, ArcL2 - Arc welding with linear motion on page 133*

This page is intentionally left blank

8 System parameters

8.1 Introduction

Groups of parameter types

The system parameters for RobotWare Arc are divided into the following groups:

- Arc System
- Arc Equipment
- Optical Sensor

See *Installation and setup on page 9* for more information about available options and how to install RobotWare Arc.

Parameter types

Parameters	Definitions
Arc System	Defines the top level of the Arc System parameters.
Arc System Properties	Defines the properties for Arc System. Includes the units that will be used when programming Robot-Ware Arc.
Arc Robot Properties	Defines the individual Robot properties for Arc System.
Arc Error Handler	Defines the properties for Arc Error Handler.
Arc Recovery Menu	Defines the properties for Arc Recovery Menu.
Arc Equipment	Defines the Equipment Class. The Equipment Class is a software package that is customized to handle a specific welding Power Source.See *Installation and setup on page 9* for more information about available options and how to select Power Source.
Arc Equipment Properties	Defines the properties for the Equipment Class.
Arc Equipment Digital Inputs	Defines the external Digital Input signals that will be used by the process.
Arc Equipment Digital Outputs	Defines the external Digital Output signals that will be used by the process.
Arc Equipment Analog Inputs	Defines the external Analog Input signals that will be used by the process.
Arc Equipment Analog Outputs	Defines the external Analog Output signals that will be used by the process.
Arc Equipment Group Outputs	Defines the external Group Output signals that will be used by the process.
Optical Sensor	Defines the Optical Sensor.
Optical Sensor Properties	Defines the properties of the Optical Sensor.

The Generic Equipment Class

The *Generic Equipment Class* and settings are activated if one of the following power source types is selected.

- Fronius TPS

- Standard I/O Welder
- Simulated Welder

They all load the standard I/O equipment class that supports basic analog and schedule based welding.

Defining arc welding systems

Up to three arc welding systems can be activated simultaneously in the same robot installation. This may be required in the following cases:

- More than one process equipment is connected
- Two different electrode dimensions are used (different feeding systems must be used for this to happen)
- More than one process is used. For example, TIG and MIG/MAG.

If more than one arc welding system is defined, then a new set of instructions and data types is activated for each system. The first additional system, which is defined by the sequence of defined system in a configuration file, is connected to instructions and data types with the suffix 1, and the second to the suffix 2. That is, `ArcL1` and `ArcC1` would be connected to `seamdata1`, `welddata1`.

Configuration files

> **Note**
>
> Configuration files and backups shall not be loaded into systems running an older RobotWare version than the one they were created in.
>
> Configuration files and backups are not guaranteed to be compatible between major releases of RobotWare and may need to be migrated after a RobotWare upgrade.

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

8.2 The group Arc System

8.2.1 The type Arc System settings

Description

The top level of configuration parameters for RobotWare Arc is *Arc System*. The settings of *Arc System* is valid for the whole robot system. That is, in a MultiMove setup, all robots with RobotWare Arc installed will have these settings. If individual settings for each robot is wanted, then the parameter *Individual Robot Properties Active* should be set to True.

Parameters

Parameter	Default value	Data type	Note
Name	ARC1	string	The name of the system. ARC1 for arc system 1, ARC2 for arc system 2, and ARC3 for arc system 3. Only ARC1 is installed by default. Additional arc systems can be selected and installed, in RobotStudio.
Use Arc System Properties	ARC1	string	The arc system properties used by the arc system.
Use Arc Error Handler	default	string	The arc error handler used by the arc system. See *Configuring Weld Error Recovery on page 61*.
Individual Robot Properties Active	False	boolean	Defines if individual robot properties are active.

8.2.2 The type Arc System Properties

Description

The type *Arc System Properties* holds parameters that specify the behavior of the system. The system includes all robots in the configuration.

Parameters

Parameter	Data type	Note
Name	string	The name of the *Arc System Properties*.
Units	string	The arc units used by the arc system. These settings are used by the RobotWare Arc operator interface.
Restart On	bool	Specifies whether the weld is to be restarted in the event of a welding error. This restart can be done in three different ways: • **Automatically:** as many times as specified in the parameter *Number of Retries* • **Program controlled:** using the error handler for the routine • **Manually:** when the error has been remedied, the program can be started in the normal way If *Restart On* is set, the robot automatically reverses to a position as specified in the parameter *Restart Distance*. Default value: FALSE
Restart Distance	num	The distance that the robot reverses on the current seam relative to the position where it was interrupted. Default value: 0
Number Of Retries	num	The number of automatic restart attempts per seam at welding interrupt. Default value: 0
Scrape On	bool	Specifies if the robot is to weave at the actual weld start (scrape start). The scrape types are specified in seamdata. This weaving is automatically interrupted when the arc is ignited. This parameter does not influence the behavior at restart. Default value: FALSE
Scrape Optional On	bool	Specifies if the robot is to weave at weld restart. The scrape types are specified in seamdata. If parameter is reset (OFF), there will be 'Weave scrape' at restart. Default value: TRUE
Scrape Width	num	The width of the weave pattern for a scrape start. Default value: 10
Scrape Direction	num	The angle of direction of the weave for a scrape start. It is specified in degrees, where 0 implies a weave that is carried out at a 90 degrees angle to the direction of the weld. Default value: 0

Continues on next page

Parameter	Data type	Note
Scrape Cycle Time	num	The time (in seconds) it takes for a complete weave cycle for a scrape start. Default value: 0.2
Ignition Move Delay On	bool	Specifies whether the move delay specified in seamdata is to be used from the time the arc is considered stable at ignition until the heating phase is started. Default value: FALSE
Motion Timeout	num	Specifies the timeout time for no motion. When all conditions are fulfilled for starting the motion, this timer starts. This is useful in MultiMove systems when for example one of two robots is ready to start the weld and the other one is trying to ignite. The motion timeout on the first robot will then cause an error, CAP_MOV_WATCHDOG, that will stop all motion in the system. If the parameter is set to 0 there is no timeout. Default value: 1
Weave Sync On	bool	Specifies whether synchronisation pulses are to be sent at the end positions of the weave. Default value: FALSE
Stop Mode	num	Specifies the stop mode at weld errors. The following stop modes are available: • Smooth Stop On Path (0) • Quick Stop On Path (1) • Emergency Stop (2) The default stop mode is *Smooth Stop On Path (0)*. This is used in all RobotWare releases previous to 6.09. *Quick Stop On Path (1)* stops the robot faster than *Smooth Stop On Path (0)* and should be used in combination with the *Production Monitoring* option to get more accurate seam length calculations. *Emergency Stop (2)* is the fastest stop, but the path might not be followed in this case.

> **Note**
>
> Scaling between a logical and a physical value on an analog output signal, is always expressed in m/s. The units in the RAPID code is always SI_UNITS, the settings above is used only by the RobotWare Arc operator interface for converting to the above units in the user interface.

Units and values

The units in the RAPID code is always mm and mm/s. The conversion to the specified units is made in the RobotWare Arc User Interface.

The unit settings available in the *Arc System Properties* apply to the presentation of data in the **Program Data** window and to the RW Arc tuning window.

The `wirefeed` component of `welddata` is always converted from mm/s to m/s before sending the value to the Analog Output.

The chart below shows how to calculate `-MaxLog` in EIO.cfg when `-MaxPhys 10` and the `wfeed` unit has a maximum speed of 22 m/min. It is valid for all power

8.2.2 The type Arc System Properties
Continued

source options except 650-9 FroniusTPS4000/5000 where the wirefeed speed is not converted to m/s. (For more information, see *Application manual - Fronius TPS 4000/5000 IRC5 Interface.*)

WELD_UNITS

-MaxLog	
% (100 max)	-MaxLog = 100*1000/60*1000 = 1.67
Max value in **Program Data**	100
Max value in RAPID code	100*1000/60 = 1666.67
wfspeed (22m/min max)	-MaxLog = 22/60 = 0.367
Max value in **Program Data**	22 (m/min)
Max value in RAPID code	22*1000/60 = 367 (mm/s)

US_UNITS

-MaxLog	
% (100 max)	-MaxLog = 100*25.4/60*1000 = 0.0423
Max value in **Program Data**	100
Max value in RAPID code	100*25.4/60 = 42.3
wfspeed (22m/min max)	-MaxLog = 22/60 = 0.367
Max value in **Program Data**	22*1000/25.4 = 866.141 (ipm)
Max value in RAPID code	22*1000/60 = 367 (mm/s)

SI_UNITS

-MaxLog	
% (100 max)	-MaxLog = 100/1000 = 0.1
Max value in **Program Data**	100
Max value in RAPID code	100
wfspeed (22m/min max)	-MaxLog = 22/60 = 0.367
Max value in **Program Data**	22*1000/60 = 367 (mm/s)
Max value in RAPID code	22*1000/60 = 367 (mm/s)

Example 1

WELD_UNITS is used.

Max speed for the wirefeeder is 22 (m/min)

Max speed in the **Program Data** window: 22 (m/min)

Value in the RAPID code: 367 (mm/s)

Value on the wirefeed Analog Output: 0.367 V

Example 2

WELD_UNITS is used.

Max speed for the wirefeeder is 22 (m/min)

Max speed in the **Program Data** window: 100 (%)

Value in the RAPID code: 1667 (*)

Continues on next page

Value on the wirefeed Analog Output: 1.67 V

The * indicates that the % value is converted according to WELD_UNITS, which in this case does not give a very useful value. Therefore, if wirefeed is expressed as %, we recommend using SI_UNITS.

8.2.3 The type Arc Robot Properties

Description

The *Arc Robot Properties* holds parameters that specifies the behavior for the individual robots.

The parameter values are active when running in independent motion. In synchronized motion, the parameters in *Arc System Properties* are used.

Parameters

Parameter	Data type	Note
Name	string	The name of the *Arc Robot Properties*.
Units	string	The arc units used by the arc system. These settings are used by the [RobotWare Arc operator interface.
Restart On	bool	Specifies whether the weld is to be restarted in the event of a welding error. This restart can be done in three different ways: • **Automatically:** as many times as specified in the parameter *Number of Retries* • **Program controlled:** using the error handler for the routine • **Manually:** when the error has been remedied, the program can be started in the normal way If *Restart On* is set, the robot automatically reverses to a position as specified in the parameter *Restart Distance*. Default value: FALSE
Restart Distance	num	The distance that the robot reverses on the current seam relative to the position where it was interrupted. Default value: 0
Number Of Retries	num	The number of automatic restart attempts per seam at welding interrupt. Default value: 0
Scrape On	bool	Specifies if the robot is to weave at the actual weld start (scrape start). The scrape types are specified in seamdata. This weaving is automatically interrupted when the arc is ignited. This parameter does not influence the behavior at restart. Default value: FALSE
Scrape Optional On	bool	Specifies scrape type at weld restart. The scrape types are specified in seamdata. If parameter is reset (OFF), there will be 'Weave scrape' at restart. Default value: TRUE
Scrape Width	num	The width of the weave pattern for a scrape start. Default value: 10

Continues on next page

Parameter	Data type	Note
Scrape Direction	num	The angle of direction of the weave for a scrape start. It is specified in degrees, where 0 degrees implies a weave that is carried out at a 90 degrees angle to the direction of the weld. Default value: 0
Scrape Cycle Time	num	The time (in seconds) it takes for a complete weave cycle for a scrape start. Default value: 0.2
Ignition Move Delay On	bool	Specifies whether the move delay specified in seamdata is to be used from the time the arc is considered stable at ignition until the heating phase is started. Default value: FALSE
Motion Timeout	num	Specifies the timeout time for no motion. When all conditions are fulfilled for starting the motion, this timer starts. This is useful in MultiMove systems when for example one of two robots is ready to start the weld and the other one is trying to ignite. The motion timeout on the first robot will then cause an error, CAP_MOV_WATCHDOG, that will stop all motion in the system. If the parameter is set to 0 there is no timeout. Default value: 1
Weave Sync On	bool	Specifies whether synchronisation pulses are to be sent at the end positions of the weave. Default value: FALSE
Stop Mode	num	Specifies the stop mode at weld errors. The following stop modes are available: • Smooth Stop On Path (0) • Quick Stop On Path (1) • Emergency Stop (2) The default stop mode is *Smooth Stop On Path (0)*. This is used in all RobotWare releases previous to 6.09. *Quick Stop On Path (1)* stops the robot faster than *Smooth Stop On Path (0)* and should be used in combination with the *Production Monitoring* option to get more accurate seam length calculations. *Emergency Stop (2)* is the fastest stop, but the path might not be followed in this case.

ℹ️ **Note**

Scaling between a logical and a physical value on an analog output signal, is always expressed in m/s. The units in the RAPID code is always SI_UNITS, the settings above is used only by the RobotWare Arc operator interface for converting to the above units in the user interface.

Units and values

The units in the RAPID code is always mm and mm/s. The conversion to the specified units is made in the RobotWare Arc User Interface.

The unit settings available in the *Arc System Properties* apply to the presentation of data in the **Program Data** window and to the RW Arc tuning window.

8 System parameters

8.2.3 The type Arc Robot Properties
Continued

The Wirefeed component of welddata is always converted from mm/s to m/s before sending the value to the Analog Output.

The chart below shows how to calculate -MaxLog in EIO.cfg when -MaxPhys 10 and the wfeed unit has a maximum speed of 22 m/min. It is valid for all Powersource options except 650-9 FroniusTPS4000/5000 where the wirefeed speed is not converted to m/s. (See Fronius TPS 4000/5000 IRC5 Interface doc for more information.)

WELD_UNITS

-MaxLog	
% (100 max)	-MaxLog = 100*1000/60*1000 = 1.67
Max value in **Program Data**	100
Max value in RAPID code	100*1000/60 = 1666.67
wfspeed (22m/min max)	-MaxLog = 22/60 = 0.367
Max value in **Program Data**	22 (m/min)
Max value in RAPID code	22*1000/60 = 367 (mm/s)

US_UNITS

-MaxLog	
% (100 max)	-MaxLog = 100*25.4/60*1000 = 0.0423
Max value in **Program Data**	100
Max value in RAPID code	100*25.4/60 = 42.3
wfspeed (22m/min max)	-MaxLog = 22/60 = 0.367
Max value in **Program Data**	22*1000/25.4 = 866.141 (ipm)
Max value in RAPID code	22*1000/60 = 367 (mm/s)

SI_UNITS

-MaxLog	
% (100 max)	-MaxLog = 100/1000 = 0.1
Max value in **Program Data**	100
Max value in RAPID code	100
wfspeed (22m/min max)	-MaxLog = 22/60 = 0.367
Max value in **Program Data**	22*1000/60 = 367 (mm/s)
Max value in RAPID code	22*1000/60 = 367 (mm/s)

Example 1

WELD_UNITS is used.

Max speed for the wirefeeder is 22 (m/min)

Max speed in the Program Data window: 22 (m/min)

Value in the RAPID code: 367 (mm/s)

Value on the wirefeed Analog Output: 0.367 V

Continues on next page

Example 2

WELD_UNITS is used.

Max speed for the wirefeeder is 22 (m/min)

Max speed in the Program Data window: 100 (%)

Value in the RAPID code: 1667 (*)

Value on the wirefeed Analog Output: 1.67 V

The * indicates that the % value is converted according to WELD_UNITS, which in this case does not give a very useful value. Therefore, if wirefeed is expressed as %, it is recommended to use SI_UNITS.

8.2.4 The type Arc Units

Description

The *Arc Units* type holds parameters that specifies the units used for the *Arc System Properties*. It is possible to define custom units based on the speed, length and wirefeed attributes shown in the following table.

Parameter	Data type	Note
Arc Length unit	string	The available length units are: • mm • inch
Arc Velocity unit	string	The available velocity units are: • mm/s • m/min • ipm • cm/min
Arc Feed unit	string	The available feed units are: • mm/s • m/min • ipm

RobotWare Arc provides three different predefined units: SI_UNITS, US_UNITS and WELD_UNITS. These units are write protected in the configuration database.

> **Note**
>
> For the standard I/O welder it is possible to use different units for *wire feed speed*. For all other power sources check in the respective user manual.

Parameters

Unit string	Speed	Length	Wirefeed
SI_UNITS	mm/s	mm	mm/s
US_UNITS	ipm	inch	ipm
WELD_UNITS	mm/s	mm	m/min

8.2.5 The type Arc Equipment

Description

The *Arc Equipment* holds parameters for the equipment.

Parameters

Parameter	Default value	Data type	Note
Name	ARC1_EQUIP_T_ROB1	string	The name of the *Arc Equipment*. These names must not be changed. ARC1_EQUIP_T_ROB1 for System1 in T_ROB1 ARC1_EQUIP_T_ROB2 for System1 in T_ROB2 ARC1_EQUIP_T_ROB3 for System1 in T_ROB3 ARC1_EQUIP_T_ROB4 for System1 in T_ROB4
Welder Type	StandardIO	string	The name of the welder type.
Loaded In Robot	T_ROB1	string	In which robot the equipment is loaded.
Use Arc Equipment Class	stdIO_T_ROB1	string	The arc equipment class used by the arc equipment.
Use Arc Equipment Properties	stdIO_T_ROB1	string	The arc equipment properties used by the arc equipment.

8.2.6 The type Arc Equipment Class

Description

The *Arc Equipment Class* holds parameters for the equipment class.

Parameters

Parameter	Default value	Data type	Note
Name	stdIO_T_ROB1	string	The name of the *Arc Equipment Class*.
Equipment Class File Name	awEquipSTD	string	The name of the EquipmentClass module to load.
Path	RELEASE:/op-tions/arc/WeldEquip-ment	string	The path to the equipment class.

Application manual - Arc and Arc Sensor
3HAC050988-001 Revision: K

8.3 The group Generic Equipment Class

8.3.1 The type Arc Equipment Properties

Description

The *Arc Equipment Properties* holds parameters for the equipment class.

Parameters

The following parameters can be defined in RobotWare Arc.

Parameter	Data type	Note
Name	string	The name of the *Arc Equipment Properties*.
Use Arc Equipment IO DI	string	The Arc Equipment IO DI used by the *Arc Equipment Properties*.
Use Arc Equipment IO DO	string	The Arc Equipment IO DO used by the *Arc Equipment Properties*.
Use Arc Equipment IO AO	string	The Arc Equipment IO AO used by the *Arc Equipment Properties*.
Use Arc Equipment IO AI	string	The Arc Equipment IO AI used by the *Arc Equipment Properties*.
Use Arc Equipment IO GO	string	The Arc Equipment IO GO used by the *Arc Equipment Properties*.
Preconditions On	bool	Specifies whether preconditions is to be used. If precond is on, the gas, torch and water supervision signals are verified before welding is started. Default value: FALSE
Ignition On	bool	If ignition data is defined, *Component group*: *Ignition* in `seamdata` (*seamdata - Seam data on page 178*) is available. Specifies if ignition data specified in seamdata is to be used at the start of the weld phase. At the start it is often beneficial to define higher weld data values for a better ignition. If the ignition data parameter is changed, the contents of seamdata will also change. Default value: FALSE
Heat On	bool	If ignition data is defined, *Component group*: *Heat* in `seamdata` (*seamdata - Seam data on page 178*) is available. When the arc is ignited, the seam will generally not have reached the correct temperature. Preheating can thus be used at the start of the weld to define higher weld data values. If the preheating parameter is changed, the contents of seamdata will also change. Default value: FALSE

8 System parameters

Parameter	Data type	Note
Fill On	bool	Specifies whether a crater fill is to be used in the final phase. This means that the end crater that can form in the completed weld will be filled in with extra filler material. Exactly how the crater fill is to be carried out is described in *seamdata - Seam data on page 178*. If the Crater fill parameter is changed, the contents of seamdata will also change. Default value: FALSE
Burnback On	bool	Specifies whether burnback as defined in seamdata is to be used in the final phase. It is used in MIG/MAG welding and means that the power supply switches on for a short while after the electrode feed has been turned off. The end of the weld electrode is then melted and transferred to the molten metal in the weld deposit. In this way, the electrode will separate from the molten metal and not stick to it when it starts to harden. Exactly how the burnback is to be carried out is described in *seamdata - Seam data on page 178*. If the Burnback parameter is changed, the contents of seamdata will also change. If burnback is set, `bback_time` in seamdata (*seamdata - Seam data on page 178*) is available. If both, burnback and burnback voltage, are set, `bback_voltage` in seamdata (*seamdata - Seam data on page 178*) is available. Default value: FALSE
Burnback Voltage On	bool	Specifies whether a specific burnback voltage should be used in the burnback phase. If not specified, burnback will be performed with the voltage used in the previous welding phase. If the Burnback voltage parameter is changed, the contents of seamdata will also change. If both, burnback and burnback voltage, are set, `bback_voltage` in seamdata (*seamdata - Seam data on page 178*) is available. Default value: FALSE
Rollback On	bool	Specifies whether rollback is to be used in the final phase. It is used in TIG welding and means that the cold wire is reversed before the molten metal hardens, to prevent the wire sticking. Exactly how the rollback is to be carried out is described in seamdata. If the Rollback parameter is changed, the contents of seamdata will also change. If rollback is set, `rback_time` in seamdata (*seamdata - Seam data on page 178*) is available. If both, rollback and rollback wirefeed, are set, `rback_wirefeed` in seamdata (*seamdata - Seam data on page 178*) is available. Default value: FALSE

Parameter	Data type	Note
Rollback Wirefeed On	bool	Specifies whether a specific rollback wirefeed speed should be used in the rollback phase. If not specified, a wirefeed speed of 10 mm/s will be used. If the Rollback wirefeed speed parameter is changed, the contents of seamdata will also change. If both, rollback and rollback wirefeed, are set, `rback_wirefeed` in seamdata (*seamdata - Seam data on page 178*) is available. Default value: FALSE
Autoinhibit On	bool	If this flag is set, weld inhibition will be allowed in AUTO-mode. Otherwise it is not allowed. Default value: TRUE
Welder Robot	bool	Specifies whether the IRB is a welding IRB. Used by Arc operator interface. Default value: TRUE
Heat as time	bool	Specifies if the heat phase should use the seamdata parameters `heat_time` or `heat_distance`. TRUE means that `heat_time` is used and visible in the seamdata. FALSE means that `heat_distance` and `heat_speed` is used and visible in the seamdata. Default value: FALSE
Override on	bool	Specifies the visibility of the org value components in welddata. Default value: FALSE
Welder Ready Supervision On	bool	For power sources that keep *welder ready* active (high), supervision in the main welding phase can be used. Set to TRUE to activate. Default value: FALSE
Schedport Type	num	Type of port used to transfer program data to the welding equipment: • Binary (=1): Binary-coded group of digital output signals. • Pulse (=2): Program numbers are sent in the form of a number of pulses on the Weldschedule port signal which should then comprise two digital signals. They are pulsed in tens on one of the outputs and in ones on the other. If Binary, Pulse or CAN is defined, the component weld_sched in welddata (*welddata - Weld data on page 197*) is available. Default value: 0
Arc Preset	num	Delays the power control signal. This allows the analog reference signals enough time (in seconds) to stabilize before the weld is started. Default value: 0.05

Continues on next page

Parameter	Data type	Note
Arc OK Delay	num	The time it takes the welding arc to stabilize at the start of a weld. The arc is only considered ignited after the arc supervision signal has been high for arc ok delay seconds. **Note** The functionality of this parameter is moved to the EIO domain. To get the same behavior, an active filter must be defined for the *ArcEst* digital input signal. The time is expressed in [ms].
Ignition Timeout	num	The maximum time (in seconds) permitted for igniting the welding arc. If the parameter is set to 0 there is no timeout. Default value: 0.9
Weld Off Timeout	num	The maximum time (in seconds) permitted for shutting off the welding arc. If the parameter is set to 0 there is no timeout. Default value: 10
Time to feed 15mm wire	num	The time in seconds to feed wire. Used by the stickout button in the RobotWare Arc operator interface. The default values are adjusted to feed 15 mm wire. If longer or shorter stickout is wanted, the time can be adjusted via this parameter. Default value: 1.1s for AristoMigIntegrated, 0.38s for MigRob, 1s for Fronius and 0.33 for the other welders.

8.3.2 The type Arc Equipment Digital Inputs

Parameters

The following digital inputs can be defined in RobotWare Arc.

Parameter	Data type	Note
Name	string	The name of the *Arc Equipment Digital Inputs*.
ManFeedInput	signaldi	Digital input signal for manual wire feed. A high signal means that the welding equipment has manual wire feed enabled.
WeldInhib	signaldi	Digital input signal for program execution without welding. A high signal means that welding is inhibited.
WeaveInhib	signaldi	Digital input signal for program execution without weaving. A high signal means that weaving is inhibited.
TrackInhib	signaldi	Digital input signal to inhibit tracking (not seen on FlexPendant). A high signal means that the tracking is inhibited.
StopProc	signaldi	Digital input signal for stopping program execution. This signal affects arc welding instructions only. A high signal means that program execution will stop as soon as an arc welding instruction is executed.
ArcEst	signaldi	Digital input signal for supervision of the welding arc. A high signal means that the welding arc is ignited. This parameter must always be defined.
ArcEstLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
ArcEst2	signaldi	Digital input signal for supervision of the welding arc in gun number 2 in a TwinWire setup. A high signal means that the welding arc is ignited.
ArcEst2Label	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
WelderReady	signaldi	Digital input signal for supervision of the *WelderReady* signal.
WelderReadyLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
WeldOk	signaldi	Digital input signal for supervision of the weld process. Same signal flow as ArcEst.
WeldOkLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
VoltageOk	signaldi	Digital input signal for supervision of the voltage. A high signal means that the voltage is OK.

8.3.2 The type Arc Equipment Digital Inputs
Continued

Parameter	Data type	Note
VoltageOkLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
CurrentOk	signaldi	Digital input signal for supervision of the current. A high signal means that the current is OK.
CurrentOkLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
WaterOk	signaldi	Digital input signal for supervision of the water. A high signal means that the water is OK.
WaterOkLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
WirefeedOk	signaldi	Digital input signal for supervision of the wirefeed. A high signal means that the wirefeed is OK.
WirefeedOkLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
GasOk	signaldi	Digital input signal for supervision of the protective gas. A high signal means that the protective gas is OK.
GasOkLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
GunOk	signaldi	Digital input signal for supervision of the torch. A high signal means that the torch is OK.
GunOkLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
WirestickErr	signaldi	Digital input signal for supervision of the wire stick status. A high signal means that an error has occurred.
WirestickErrLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
USERIO1	signaldi	Digital input signal for supervision of the user defined input signal USERIO1 during the weld process. The level is only supervised during the weld, not at start or end of the weld. A high signal means that the signal is OK.
USERIO1Label	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
USERIO2	signaldi	Digital input signal for supervision of the user defined input signal USERIO2 during the weld process. The level is only supervised during the weld, not at start or end of the weld. A high signal means that the signal is OK.

Continues on next page

Parameter	Data type	Note
USERIO2Label	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
USERIO3	signaldi	Digital input signal for supervision of the user defined input signal USERIO3 during the weld process. The level is only supervised during the weld, not at start or end of the weld. A high signal means that the signal is OK.
USERIO3Label	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
USERIO4	signaldi	Digital input signal for supervision of the user defined input signal USERIO4 during the weld process. The level is only supervised during the weld, not at start or end of the weld. A high signal means that the signal is OK.
USERIO4Label	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.
USERIO5	signaldi	Digital input signal for supervision of the user defined input signal USERIO5 during the weld process. The level is only supervised during the weld, not at start or end of the weld. A high signal means that the signal is OK.
USERIO5Label	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See *Configurable error handling on page 233*.

Note

If the signals from the arc process equipment are not stable enough, it might be necessary to filter them. See *Topic I/O System* section *Type Signal* in *Technical reference manual - System parameters*

8.3.3 The type Arc Equipment Digital Outputs

Parameters

The following digital outputs can be defined in RobotWare Arc.

Parameter	Data type	Note
Name	string	The name of the *Arc Equipment Digital Outputs*.
AWError	signaldo	Digital output signal for indication of welding defects. A high signal means that an error has occurred. If a normal program stop occurs in the middle of a weld, no high signal will be generated.
GasOn	signaldo	Digital output signal for control of the gas flow. A high signal means that the gas flow is active.
WeldOn	signaldo	Digital output signal for control of the weld voltage. A high signal means that the weld voltage control is active. This parameter must always be defined
RobotReady	signaldo	Digital output signal for indication of *RobotReady* signal.
FeedOn	signaldo	Digital output signal for activation of the wire feed. A high signal means wirefeed forward.
FeedOnBwd	signaldo	Digital output signal for backward activation of the wire feed. A high signal means wirefeed backward.
SchedStrobe	signaldo	Digital output signal used for handshaking during data transfer from the program to the welding equipment. Used if schedule port type has been defined as Pulse. A high signal means that the schedule strobe signal is used for handshaking during data transfer.
ProcessStopped	signaldo	Digital output signal used to indicate that the weld has been interrupted. A high signal means that the weld has been interrupted either because of a welding defect or because of a normal program stop.
SupervArc	signaldo	Digital output signal for indication of welding arc errors. A high signal means that an error has occurred.
SupervVolt	signaldo	Digital output signal for indication of voltage errors. A high signal means that an error has occurred.
SupervCurrent	signaldo	Digital output signal for indication of current errors. A high signal means that an error has occurred.
SupervWater	signaldo	Digital output signal for indication of cooling water errors. A high signal means that an error has occurred.
SupervGas	signaldo	Digital output signal for indication of protective gas errors. A high signal means that an error has occurred.
SupervFeed	signaldo	Digital output signal for indication of wire feed errors. A high signal means that an error has occurred.

Parameter	Data type	Note
SupervGun	signaldo	Digital output signal for indication of torch errors. A high signal means that an error has occurred.
AWBlock	signaldo	Digital output signal for indication of Blocked process
SupervWelder-Ready	signaldo	Digital output signal for indication of WelderReady signal errors. A high signal means that an error has occurred.

8.3.4 The type Arc Equipment Analog Outputs

Parameters

The following analog outputs can be defined in RobotWare Arc.

Parameter	Data type	Note
Name	string	The name of the *Arc Equipment Analog Outputs*.
VoltReference	signalao	Analog output signal for analog voltage reference. If weld voltage is defined, the component `weld_voltage` in welddata (*welddata - Weld data on page 197*) is available.
FeedReference	signalao	Analog output signal for analog wire feed reference.If wire feed is defined and schedule port type is set to CAN (=3), the component `weld_wirefeed` in welddata (*welddata - Weld data on page 197*) is available.
CurrentReference	signalao	Analog output signal for analog current reference. If current is defined, the component `weld_current` in welddata (*welddata - Weld data on page 197*) is available.
ControlPort	signalao	Tunable analog output for certain welders.
VoltReference2	signalao	Analog output signal for analog voltage reference for gun number 2 in a TwinWire setup. If weld voltage is defined, the component weld_voltage in welddata (*welddata - Weld data on page 197*) is available.
FeedReference2	signalao	Analog output signal for analog wire feed reference for gun number 2 in a TwinWire setup.If wire feed is defined and schedule port type is set to CAN (=3), the component `weld_wirefeed` in welddata (*welddata - Weld data on page 197*) is available.
CurrentReference2	signalao	Analog output signal for analog current reference for gun number 2 in a TwinWire setup. If current is defined, the component `weld_current` in welddata (*welddata - Weld data on page 197*) is available.
ControlPort2	signalao	Tunable analog output2 for certain welders. Used in TwinWire Systems.

8.3.5 The type Arc Equipment Analog Inputs

Parameters

The following analog inputs can be defined in RobotWare Arc.

Parameter	Data type	Note
Name	string	The name of the *Arc Equipment Analog Inputs*.
VoltageMeas	signalai	Analog input signal for voltage measurement.
CurrentMeas	signalai	Analog input signal for current measurement.

8.3.6 The type Arc Equipment Group Outputs

Parameters

The following group outputs can be defined in RobotWare Arc.

Parameter	Data type	Note
Name	string	The name of the *Arc Equipment Group Outputs*.
SchedulePort	signalgo	Group of digital output signals used to transfer schedule data to the welding equipment.
ModePort	signalgo	Group of digital output signals used to transfer mode data to the welding equipment.

8.4 The group Optical Sensor

8.4.1 The type Optical Sensor

Parameters

The type *Optical Sensor* holds parameters for the option *Optical Sensor*.

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

8.4.2 The type Optical Sensor Properties

Parameters

The *Optical Sensor Properties* holds parameters for the optical sensor.

Parameter	Data type	Note
Name	string	The name of the *Arc Sensor Class*.
Sensor Manufacturer	string	The name of the sensor manufacturer.
Track System	num	Defines the TrackSystem type. 1 for LaserTrack
Device	string	The device name used for the tracker. Device must match the transmission protocol name configured in SIO.cfg. "Laser1:" for lasertracker.
Pattern Sync Threshold	num	The coordination position at the extents of the weaving pattern. It is specified as a percentage of the width on either side of the weaving center. When weaving is carried out beyond this point, a digital output signal is automatically set to one. This type of coordination is intended for seam tracking using Through-the-Arc Tracker.
Max Blind	num	The max_blind component defines the maximum distance the robot is allowed to continue moving under the assumption that the last reported position error is still valid. The parameter should be tuned to match the maximum expected tack lengths used, or the length of other features. For example, clamps that may prevent the sensor from accurately detect the actual position and geometry of the seam. If the max_blind distance has been exceeded with no new position data from the sensor an error will be reported and program execution is stopped.
Max Corr	num	Not used. The `max_corr` component in trackdata is used instead.
Adapt Start Delay	num	Not used.
Max Incremental Correction	num	Max incremental correction for the arc tracking system. If the incremental TCP correction is bigger than *Max Incremental Correction* and *Max Correction Warning* was set, the robot will continue its path but the applied incremental correction will not exceed *Max Incremental Correction*. If *Max Correction Warning* was not set, a track error is reported and program execution is stopped. Default value is 3 mm.
Log File	string	The name of the log file created during tracking.
Sensor Frequency	num	Defines the sample frequency of the sensor used. (e.g. M-Spot-90 has 5Hz sampling frequency)
Ipol Servo Delay	num	Defines the robot controller internal time delay between ipol task and servo task. Use default value: 74 ms

Continues on next page

| Parameter	Data type	Note
Ipol Correction Gain	num	Defines the gain factor for the correction imposed on ipol. Use default value: 0
Servo Sensor Factor	num	Defines the number of servo corrections per sensor readings. Use default value: 0
Correction Filter	num	Defines filtering of the correction calculated, using mean value over corr filter values. Use default value: 1
Ipol Correction Filter	num	Defines filtering of the ipol correction, using mean value over path filter values. Use default value: 1
Servo Correction	num	Defines filtering of the servo correction, using mean value Filter over path servo filter values. Use default value: 1
Error Ramp In	num	Defines during how many sensor readings ramp in is done after an error caused by sensor reading.
Error Ramp Out	num	Defines during how many sensor readings ramp out is done after an error caused by sensor reading.
CB Angle	num	Defines the angle between a 3D sensor beam and the sensor z-axis. Use default value: 0 for M-Spot-90 and 25 for SCOUT.
Calib Variable Name	num	The name of the calibration variable name found in the calibration programs. RAPID data type: pose
Calib Variable Offset Name	num	The name of the calibration variable offset name found in the calibration programs. RAPID data type: pos
Max Correction Warning	bool	If this parameter is enabled, program execution is not interrupted, when the limit for maximum correction, specified in the trackdata, is exceeded. Only a warning will be sent. Default value: FALSE
WgLeftSynch	string	Digital output signal for left syncpulse.
WgRightSynch	string	Digital output signal for right syncpulse.
Wg track	string	Not used.

8.4.2 The type Optical Sensor Properties
Continued
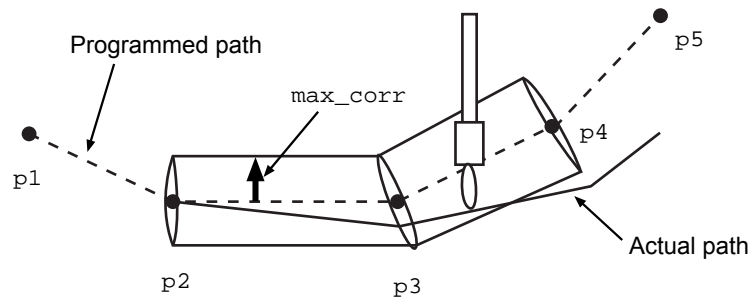
xx1200000686

Max correction

8.5 Configurable error handling

Error levels

Some of the supervision signals have labels, to make it possible to configure the error behavior when this signal is the cause of the error. There are three available error levels. MAJOR, MINOR and INFO.

- MAJOR is the default setting if no level is specified. A signal error results in a process stop. Normal error handling is executed after the stop.

- MINOR label on a signal does not result in normal error handling. A signal error results in process shutdown but without stop of the motion. An error message is displayed like the normal error handling does. After the weld is completed, there is RAPID variable that can be checked to see whether or not an MINOR error has occurred.

- INFO label on a signal does not result in normal error handling. A signal error does not stop the process, just a warning is sent and the welding process continues.

Background

When welding a part in synchronized mode and one of the welders stop to weld (due to a signal supervision error), this will cause all synchronized robots to stop and the program execution will end up in the local error handler routine. This also means that the welder without problems also stops, even though it was welding OK.

The solution to this problem is to shut down the process of the failing robot and continue with the synchronized motion. By doing so, the non-failing robot will continue with the ongoing weld without interruption. This can be achieved by labeling the recoverable system input errors in three levels - MAJOR, MINOR and INFO.

The following table shows the system behavior.

Label	SYS_STOP	In error handler	Process shutdown	Elog error	Elog warning
MAJOR	YES	YES	YES	YES	NO
MINOR	NO	NO	YES	YES	NO
INFO	NO	NO	NO	NO	YES

The MAJOR label works as if no level is specified, that is, normal error handling.

The MINOR label is only active when the weld has started. At the start of the weld the normal signal supervision is active.

User defined I/O signals

It is possible to add 5 user defined signals which will be supervised during the weld process.

These signals can also be labelled with the above mentioned levels.

Continues on next page

Error detection

If an error labelled with MINOR has occurred during the weld, a global variable is set which can be checked after the weld seam is finished. This variable will be reset at the beginning of the next weld seam. The variable name is *bMinorErr*.

The same applies for the INFO labelled errors, the variable name here is *bInfoErr*. The following shows an example of how these variables can be checked.

```
PROC Rob1_UpSide()
   ArcLStart pPrep10,v500,sm1,wd2\Weave:=wv2,fine,
         Rob1_tool\WObj:=wobj_STN1;
   SyncMoveOn sync1, all_task_list;
   ArcC pSync10,pSync20\ID:=id1,v300,sm1,wd2\Weave:=wv2,z5,
         Rob1_tool\WObj:=wobj_STN1;
   ArcC
         pSync30,pSync40\ID:=11,v300,sm1,wd2\Weave:=wv2,z5,Rob1_tool\WObj:=wobj_STN1;
   ArcC pSync50,pSync60\ID:=110,v300,sm1,wd2\Weave:=wv2,z5,
         Rob1_tool\WObj:=wobj_STN1;
   ArcCEnd pSync70,pSync80\ID:=120,v300,sm1,wd2\Weave:=wv2,fine,
         Rob1_tool\WObj:=wobj_STN1;
   !
   CheckError;
   ERROR TPWrite "Error in ROB1";
   StorePath;
   RestoPath;
   StartMoveRetry;
ENDPROC

PROC CheckError()
   ! Global VAR bInfoErr is set if there has been an INFO labelled
         error during the seam.
   IF bInfoErr THEN
     TPWrite "--- An INFO tagged signal error occurred during the
           seam ---";
     TPWrite "--- Check the elog messages for more information. --";
   ENDIF
   ! Global VAR bMinorErr is set if there has been a MINOR labelled
         error during the seam.
   IF bMinorErr THEN
     TPWrite "--- A MINOR tagged signal error occurred during the
           seam ---";
     TPWrite "---Check the elog messages for more information. ---";
   ENDIF

   IF bInfoErr OR bMinorErr THEN
     ! Stop motion and RAPID execution in all robot tasks.
     Stop;
     ! Handle error and continue execution with StartMove or press
           start button.
     !
   ENDIF
ENDPROC
```

8.6 Data masking

Description of masking

The RobotWare Arc data types, `seamdata`, `welddata`, `arcdata`, `weavedata`, and `trackdata` are masked depending on how the system is configured. Masking means that the components in the data types are visible or not visible. The idea behind this is to show only the parameters that are relevant to the user. The data masking is only valid on the FlexPendant and not in RobotStudio.

welddata

Data component	Masking rules	Unit conversion
weld_speed	Always visible	Yes (velocity unit)
org_weld_speed	Visible if parameter `override_on` is activated in PROC.	Yes (velocity unit)
main_arc	Always visible	No
org_arc	Visible if parameter `override_on` is activated in PROC.	No

arcdata

Data component	Masking rules	Unit conversion
sched	Visible if GO SchedulePort is defined in PROC. For Fronius option 650-9 visible if not Job-Mode	No
mode	Visible in Program mode for Fronius option 650-9	No
voltage	Visible if AO VoltReference is defined in PROC. For Fronius option 650-9 visible if not Job-Mode	No
wirefeed	Visible if AO FeedReference is defined in PROC. For Fronius option 650-9 visible if not Job-Mode	Yes (feed unit)
control	Visible if AO ControlPort or ControlPort2 is defined in PROC. For Fronius option 650-9 visible if not Job-Mode	No
current	Visible if AO CurrentReference is defined in PROC, and/or WeldGuide is used.	No
voltage2	Visible if AO VoltReference is defined in PROC. Not valid for Fronius option 650-9.	No
wirefeed2	Visible if AO FeedReference2 is defined in PROC. Not valid for Fronius option 650-9.	Yes (feed unit)

Continues on next page

8 System parameters

Data component	Masking rules	Unit conversion
control2	Visible if AO VoltReference2 or FeedReference2, and if AO ControlPort or ControlPort2 is defined in PROC. For Fronius option 650-9 visible if not Job-Mode	No

seamdata

Data component	Masking rules	Unit conversion
purge_time	Always visible.	No
preflow_time	Always visible.	No
ign_arc	Visible if parameter ignition_on is activated in PROC.	No
ignition_move_delay	Visible if parameter ignition_on and ign_move_delay_on is activated in PROC.	No
scrape_start	Visible if parameter scrape_on and scrape_opt_on is activated in PROC.	No
heat_speed	Visible if parameter heat_on and heat_as_time is activated in PROC.	Yes (velocity unit)
heat_time	Visible if parameter heat_on and heat_as_time is activated in PROC.	No
heat_distance	Visible if parameter heat_on is activated in PROC.	Yes (length unit)
heat_arc	Visible if parameter heat_on is activated in PROC.	No
cool_time	Visible if parameter cool_time_on and fill_on is activated in PROC.	No
fill_time	Visible if parameter fill_on is activated in PROC.	No
fill_arc	Visible if parameter fill_on is activated in PROC.	No
bback_time	Visible if parameter burnback_on is activated in PROC.	No
rback_time	Visible if parameter rollback_on is activated in PROC.	No
bback_arc	Visible if parameter burnb_volt_on and burnback_on is activated in PROC.	No
postflow_time	Always visible.	No

trackdata

Data component	Masking rules	Unit conversion
track_system	Always visible.	No
store_path	Always visible.	No
max_corr	Always visible.	No
arctrack	Visible if track_system = 0	No
opttrack	Visible if track_system = 1	No

Continues on next page

arctrackdata

Data component	Masking rules	Unit conversion
track_type	Always visible.	No
gain_y	Always visible.	No
gain_z	Always visible.	No
weld_penetration	Always visible.	No
track_bias	Always visible.	No
min_weave	Always visible.	Yes (length unit)
max_weave	Always visible.	Yes (length unit)
min_speed	Always visible.	Yes (velocity unit)
max_speed	Always visible.	Yes (velocity unit)

opttrackdata

Data component	Masking rules	Unit conversion
joint_no	Always visible.	No
filter	Always visible.	No
seamoffs_y	Always visible.	Yes (length unit)
seamoffs_z	Always visible.	Yes (length unit)
seamadapt_y	Always visible.	No
seamadapt_z	Always visible.	No

weavedata

Data component	Masking rules	Unit conversion
weave_shape	Always visible.	No
weave_type	Always visible.	No
weave_length	Always visible.	Yes (length unit)
weave_width	Always visible.	Yes (length unit)
weave_height	Always visible.	Yes (length unit)
dwell_left	Always visible.	Yes (length unit)
dwell_center	Always visible.	Yes (length unit)
dwell_right	Always visible.	Yes (length unit)
weave_dir	Always visible.	No
weave_tilt	Always visible.	No
weave_ori	Always visible.	No
weave_bias	Always visible.	Yes (length unit)
org_weave_width	Visible if parameter override_on is activated in PROC.	Yes (length unit)
org_weave_height	Visible if parameter override_on is activated in PROC.	Yes (length unit)

8.6 Data masking
Continued

Data component	Masking rules	Unit conversion
org_weave_bias	Visible if parameter override_on is activated in PROC.	Yes (length unit)

8.7 Welder Ready Supervision for StdIoWelder interface

General

The interface can handle two variants of *Welder Ready*. Some power sources interpret welder ready in such way that the signal is high once the welder is switched on, including a check of electronic components such as main power supply and fieldbus communication. The welder ready signal will stay high if no error occurs. Other power sources will set the welder ready signal low once the arc start output is set, and the signal will remain low until the weld is completed, indicating ready for next weld.

Welder Ready supervision before the ignition phase

Supervision for Welder Ready can be activated in the system parameters, topic *Process*, type *Arc Equipment Properties*, by setting the parameter *Preconditions On* to TRUE.
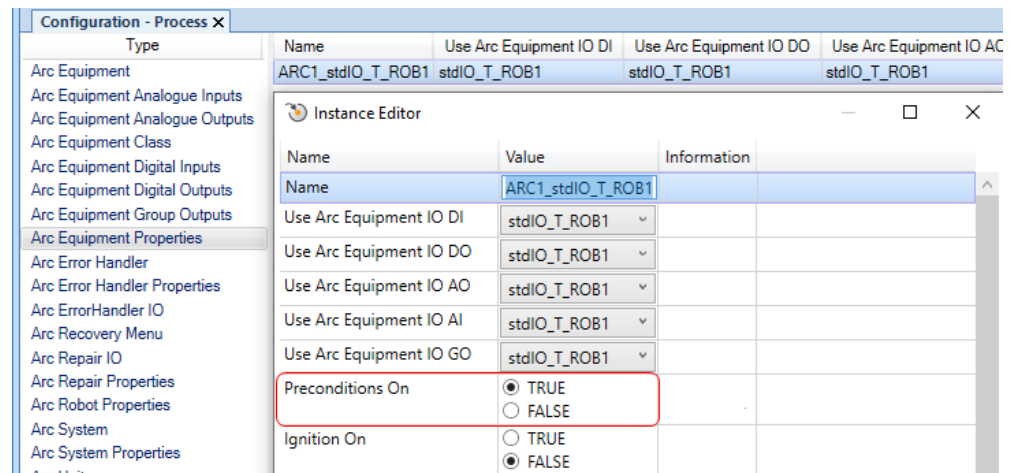
xx2100002522

A digital signal must be configured in *Arc Equipment Digital Inputs* for the *WelderReady* instance.

If *Preconditions On* is set to TRUE, the interface will check welder ready before the weld start. An error message is presented if supervision fails. If configured, a corresponding output will be set indicating that welder ready supervision failed.

Continues on next page

This output can be configured in the system parameters, topic *Process*, type *Arc Equipment Digital Outputs*, for the *WelderReady* instance.

Error message on FlexPendant
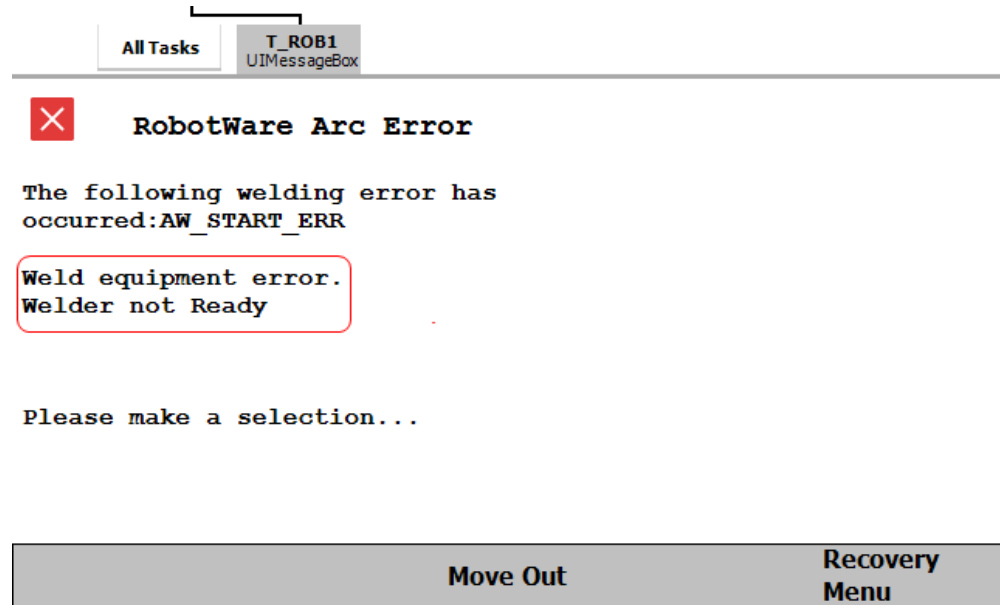
xx2100002523

Welder Ready supervision while welding is active

Supervision in the main welding phase can be only used for power sources that keep *welder ready* active (high).

Supervision can be activated in the system parameters, topic *Process*, type *Arc Equipment Properties*, by setting the parameter *WelderReady Supervision On* to TRUE.

xx2100002524

This page is intentionally left blank

Index

Index

ABB

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

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

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

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

abb.com/robotics

3HAC050988-001, Rev K, en