



ROBOTICS

# **Application manual**

## ArcWare for OmniCore



Trace back information:  
Workspace Main version a654  
Checked in 2025-03-10  
Skribenta version 5.6.018

# **Application manual ArcWare for OmniCore**

**Document ID: 3HAC084370-001**

**Revision: D**

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

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

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

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

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

# Table of contents

Overview of this manual .....	7
<b>1 Introduction</b>	<b>9</b>
<b>2 Installation</b>	<b>11</b>
<b>3 System parameters</b>	<b>15</b>
3.1 Introduction .....	15
3.2 The group Arc System .....	17
3.2.1 The type Arc System settings .....	17
3.2.2 The type Arc System Properties .....	18
3.2.3 The type Arc Units .....	22
3.2.4 The type Arc Equipment .....	23
3.2.5 The type Arc Equipment Class .....	24
3.3 The group Generic Equipment Class .....	25
3.3.1 The type Arc Equipment Properties .....	25
3.3.2 The type Arc Equipment Digital Inputs .....	28
3.3.3 The type Arc Equipment Digital Outputs .....	31
3.3.4 The type Arc Equipment Analog Inputs .....	33
3.3.5 The type Arc Equipment Analog Outputs .....	34
3.3.6 The type Arc Equipment Group Outputs .....	35
3.4 The group Optical Sensor .....	36
3.4.1 The type Optical Sensor .....	36
3.4.2 The type Optical Sensor Properties .....	37
3.5 Configurable error handling .....	39
3.6 Welder Ready Supervision for StdIoWelder interface .....	41
<b>4 Programming</b>	<b>45</b>
4.1 Programming for arc welding .....	45
4.2 FlexPendant WebApp Interface .....	51
<b>5 Programming RobotWare Arc systems with MultiMove</b>	<b>55</b>
5.1 RobotWare Arc with MultiMove .....	55
5.2 Functions for arc welding during program execution .....	56
5.3 Configuration .....	57
5.4 Limitations .....	61
<b>6 Weld Error Recovery</b>	<b>63</b>
6.1 Weld Error Recovery and error handling .....	63
6.2 Programming Weld Error Recovery .....	65
6.3 Weld Error Recovery flowchart .....	74
6.4 Configuring Weld Error Recovery .....	75
6.5 Configure the recovery menu .....	77
6.6 Weld Error Recovery I/O interface .....	79
6.7 Configure weld error recovery I/O Interface .....	88
6.8 Configure User defined error handling .....	90
6.9 User defined error handling .....	92
<b>7 Function package for collaborative robots</b>	<b>95</b>
7.1 Introduction .....	95
7.2 FlexPendant application for ArcWare for Collaborative Robots .....	96
7.3 Wizard Easy Programming .....	99

<b>8</b>	<b>RAPID reference</b>	<b>105</b>
8.1	Standard	105
8.1.1	Instructions	105
8.1.1.1	ArcLStart - Arc welding start with linear motion	105
8.1.1.2	ArcL - Arc welding with linear motion	115
8.1.1.3	ArcLEnd - Arc welding end with linear motion	124
8.1.1.4	ArcC - Arc welding with circular motion	133
8.1.1.5	ArcCEnd - Arc welding end with circular motion	142
8.1.1.6	ArcMoveL, ArcMoveC, ArcMoveJ, ArcMoveAbsJ, ArcMoveExtJ - Wrapped movement instructions for non-welding robot or additional axis in Arc <i>MultiMove</i> systems	151
8.1.1.7	ArcRefresh - Refresh arc weld data	153
8.1.1.8	RecoveryMenu - Display the recovery menu	155
8.1.1.9	RecoveryPosSet - Set the recovery position	157
8.1.1.10	RecoveryPosReset - Reset the recovery position	160
8.1.2	Data types	162
8.1.2.1	arcdata - Arc data	162
8.1.2.2	flystartdata - Flying start data	165
8.1.2.3	seamdata - Seam data	166
8.1.2.4	weavedata - Weave data	173
8.1.2.5	welddata - Weld data	180
8.2	Collaborative robots (CRB 15000)	184
8.2.1	Instructions	184
8.2.1.1	ALS - Arcwelding Linear Start	184
8.2.1.2	AL - Arcwelding Linear	186
8.2.1.3	ALE - Arcwelding Linear End	188
8.2.1.4	AC - Arcwelding Circular	189
8.2.1.5	ACE - Arcwelding Circular End	191
	<b>Index</b>	<b>193</b>

# Overview of this manual

## About this manual

This manual contains instructions for installing and programming an *ArcWare Base* system.

This manual contains instructions for operation of OmniCore controller based robots.



### Note

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

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

## 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.



### Note

Before any work on or with the robot is performed, the safety information in the product manual for the controller and manipulator shall be read.

## References

References	Document ID
<i>Operating manual - OmniCore</i>	3HAC065036-001
<i>Operating manual - RobotStudio</i>	3HAC032104-001
<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>	3HAC065038-001
<i>Technical reference manual - RAPID Overview</i>	3HAC065040-001
<i>Technical reference manual - System parameters</i>	3HAC065041-001
<i>Application manual - Controller software OmniCore</i>	3HAC066554-001
<i>Application manual - MultiMove</i>	3HAC089689-001

## Revisions

Revision	Description
A	Published with RobotWare 7.13 and ArcWare 1.0.
B	Published with ArcWare 1.1.1. <ul style="list-style-type: none"> <li>Added information about MultiMove.</li> <li>Minor corrections.</li> </ul>
C	Published with ArcWare 1.1.3. <ul style="list-style-type: none"> <li>Minor corrections.</li> </ul>

*Continues on next page*

Revision	Description
D	Published with ArcWare 1.1.3. <ul style="list-style-type: none"><li>• Updates for system parameters, group <i>Optical Sensor</i>.</li></ul>



# 1 Introduction

---

## Arc welding options and features

ArcWare for OmniCore is a software package for arc welding. It is distributed as a RobotWare Add-In and can be downloaded by means of RobotStudio. The package also contains a special function package *ArcWare for Collaborative Robot CRB15000 (GoFa)*.

ArcWare for OmniCore has support for *MultiMove*, *Standard I/O welder interface*, *External RW Add-In loaded welder* and *Simulated welder*. Support of some specific power source brands is available as separate Add-Ins that can be downloaded by means of RobotStudio.

The supported RAPID interface is described in section [RAPID reference on page 105](#).

[[

---

## Licensing

ArcWare for OmniCore implements the following licensing model. There are three different licensing levels. Within each licensing level you can freely select any of the options provided by arc welding products and covered by the license:

- *Arc Welding Standard*

With this license you get access to

- Basic arc welding functionality
- *ArcWare for Collaborative Robot CRB15000 (GoFa)*

This function package can only be used with the collaborative robot CRB 15000.

- *Standard I/O welder interface*

*Standard I/O welder* support without any I/O signals or process configuration.

- *Simulated Welder*

*Standard I/O welder* with some pre-configured I/O so that welding can be activated in RobotStudio.

- *Add-in loaded welder*

This option makes it possible to install and use an Add-In for a specific power source, e.g. *Fronius TPSi*.

- *Arc Welding Premium*

With this license you get access to the following functionality:

- *BullsEye*
- *Torch Service Center*
- *SmarTac*
- Trough-the-Arc tracking functionality for power sources that support this type of tracking, e.g. *Fronius TPSi Tracking*

- *Arc Welding Premium+*

This license level will in coming releases give you access to the most advanced functionality within the area of Arc welding, e.g. *Optical Tracking*.

*Continues on next page*

The license level *Premium* automatically includes the *Standard* level and *Premium+* includes both *Standard* and *Premium* level.

---

## Prerequisites

The following software is required as a minimum:

- RobotWare version 7.13.2 or higher
- ArcWare for OmniCore version 1.0.1 or higher
- At least the license *Arc Welding Standard*

In addition to this basic Arc welding software you may use licenses *Arc Welding Premium* or *Arc Welding Premium+* and add further functionality by adding other products, e.g. *BullsEye*, *SmarTac*.

## 2 Installation

### Distribution

ArcWare for OmniCore is distributed as a RobotWare Add-In that can be downloaded from the RobotStudio Add-Ins gallery, which installs the Add-In on the PC. Specific support for different power source brands and models is distributed in the same way.

### Licenses

You need at least one of the following Arc Welding licenses to be able to install ArcWare for OmniCore on a real controller:

- 3416-1 Arc Welding Standard
- 3416-2 Arc Welding Premium
- 3416-3 Arc Welding Premium+

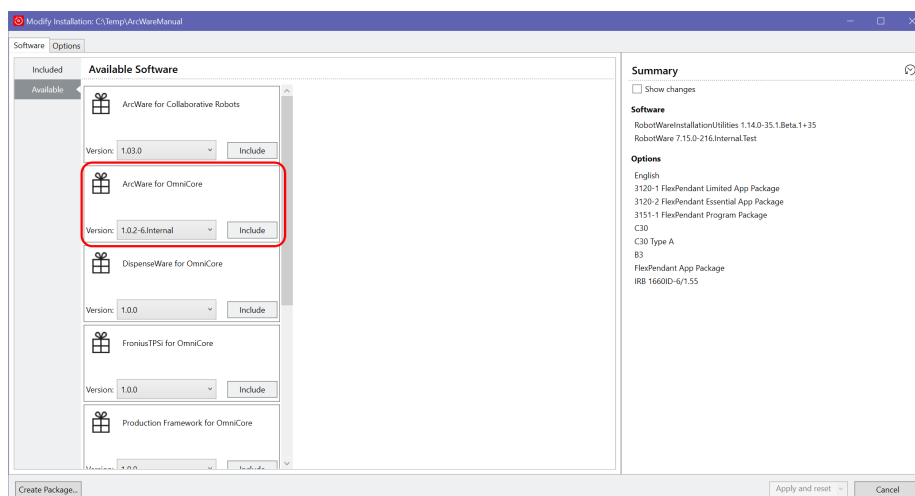
For virtual systems in RobotStudio no licenses are required.

### Software installation

Due to the license structure for ArcWare for OmniCore (see *Licenses*), the only information ABB has, is the licenses, but not the options in those licenses. Thus, for systems with ArcWare for OmniCore, ABB will install default options with power source type *Simulated welder*.

To update your robot system with all the options you want to use, follow these steps:

- 1 Start RobotStudio
- 2 Download the products you need (e.g. some power source package, BullsEye, Torch Service Center, etc.) from the RobotStudio Add-Ins tab and install it on you PC.
- 3 Connect to the robot system
- 4 Include ArcWare for OmniCore from the list of available packages



xx2300001946

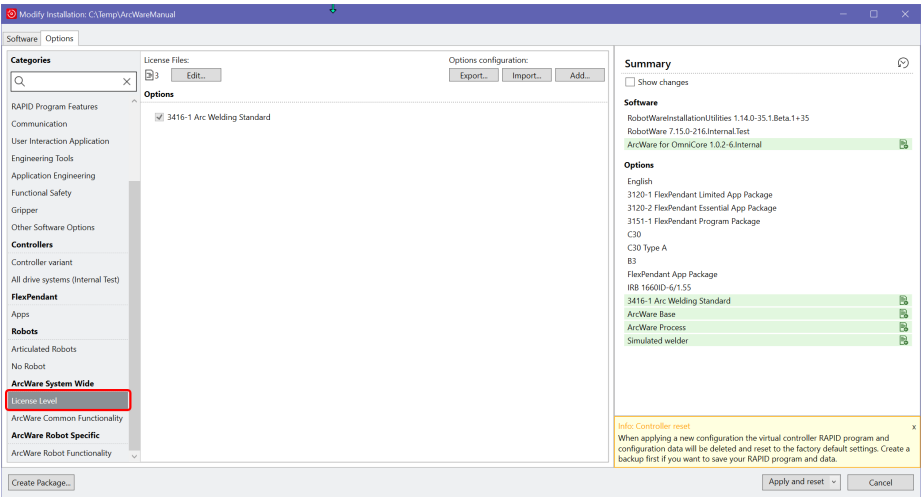
- 5 Option selection:

*Continues on next page*

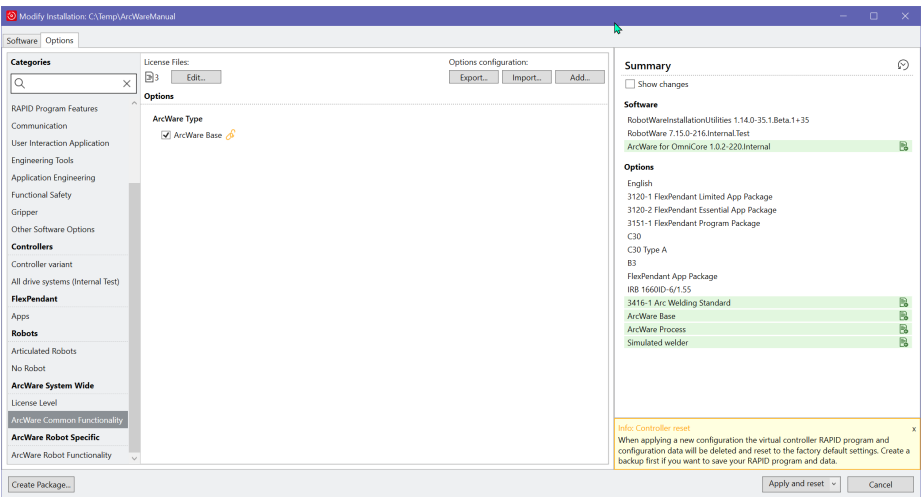
## 2 Installation

*Continued*

A number of options are selected as a default when the Add-In is added to the system. The default selections serve most needs, but check that the option selection fits your system.



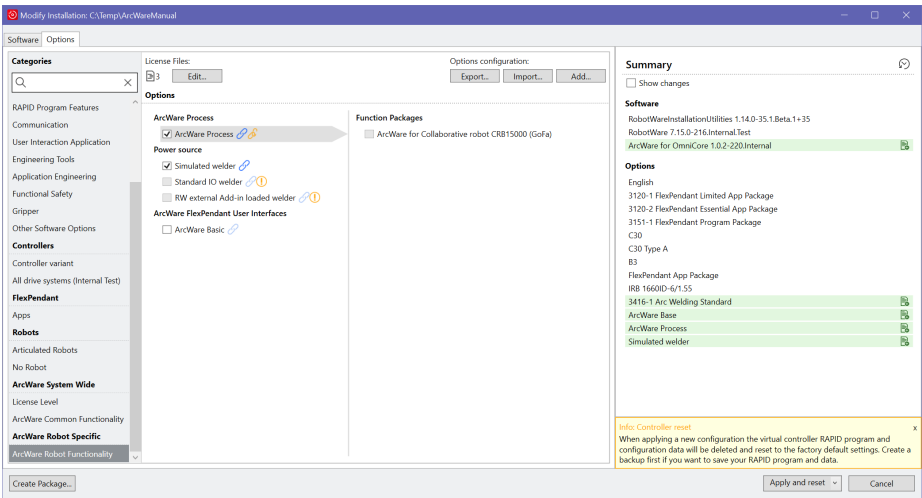
xx2300001947



xx2300001948

- 6 To change the *Power source*, remove the current selection, which will unlock all *Power source* options, and select the option of your choice.

*Continues on next page*



xx2300001949

7 Apply the changes.

**This page is intentionally left blank**

## 3 System parameters

### 3.1 Introduction

#### Groups of parameter types

The system parameters for ArcWare for OmniCore are divided into the following groups:

- Arc System
- Arc Equipment

See [Software installation on page 11](#) for more information about available options and how to install ArcWare for OmniCore.

#### 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 ArcWare for OmniCore.
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 <a href="#">Software installation on page 11</a> 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.

#### The Generic Equipment Class

The *Generic Equipment Class* and settings are activated if one of the following power source types is selected.

- Standard I/O Welder
- Simulated Welder

They all load the standard I/O equipment class that supports basic analog and schedule based welding.

*Continues on next page*

## 3 System parameters

---

### 3.1 Introduction

*Continued*

---

#### 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.

---

#### 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.



## 3.2 The group Arc System

### 3.2.1 The type Arc System settings

#### Description

The top level of configuration parameters for ArcWare for OmniCore is *Arc System*. The settings of *Arc System* is valid for the whole robot system. In a *MultiMove* setup, all robots with ArcWare for OmniCore installed will have these settings.

#### 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 <a href="#">Configuring Weld Error Recovery on page 75</a> .

## 3 System parameters

### 3.2.2 The type Arc System Properties

### 3.2.2 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 <i>Arc System Properties</i> .
Units	string	The arc units used by the arc system. These settings are used by the ArcWare for OmniCore operator interface.
Restart On	bool	<p>Specifies whether the weld is to be restarted in the event of a welding error. This restart can be done in three different ways:</p> <ul style="list-style-type: none"><li>• <b>Automatically:</b> as many times as specified in the parameter <i>Number of Retries</i></li><li>• <b>Program controlled:</b> using the error handler for the routine</li><li>• <b>Manually:</b> when the error has been remedied, the program can be started in the normal way</li></ul> <p>If <i>Restart On</i> is set, the robot automatically reverses to a position as specified in the parameter <i>Restart Distance</i>.</p> <p>Default value: FALSE</p>
Restart Distance	num	<p>The distance that the robot reverses on the current seam relative to the position where it was interrupted.</p> <p>Default value: 0</p>
Number Of Retries	num	<p>The number of automatic restart attempts per seam at welding interrupt.</p> <p>Default value: 0</p>
Scrape On	bool	<p>Specifies if the robot is to weave at the actual weld start (scrape start). The scrape types are specified in seamdata.</p> <p>This weaving is automatically interrupted when the arc is ignited.</p> <p>This parameter does not influence the behavior at restart.</p> <p>Default value: FALSE</p>
Scrape Optional On	bool	<p>Specifies if the robot is to weave at weld restart. The scrape types are specified in seamdata.</p> <p>If the parameter is reset (FALSE), there will be 'Weave scrape' at restart after stop during welding.</p> <p>Default value: TRUE</p>
Scrape Width	num	<p>The width of the weave pattern for a scrape start.</p> <p>Default value: 10</p>
Scrape Direction	num	<p>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.</p> <p>Default value: 0</p>

*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 time-out time for no motion. When all conditions are fulfilled for starting the motion, this timer starts.  This is useful in <i>MultiMove</i> systems when for example one of two robots is ready to start the weld and the other one is trying to ignite. The motion time-out 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 time-out Default value: 1
Weave Sync On	bool	Specifies whether synchronization pulses are to be sent at the end positions of the weave. Default value: FALSE
Block Tuning in Auto	bool	Specifies if tuning is enabled in Auto mode. If the parameter is set TRUE, tuning is enabled. Default value: FALSE
Block Tuning in Manual	bool	Specifies if tuning is enabled in Manual mode. If the parameter is set TRUE, tuning is enabled. Default value: FALSE



#### 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 ArcWare for OmniCore 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 ArcWare for OmniCore User Interface.

The `wirefeed` component of `welddata` is always converted from mm/s to m/s before sending the value to the Analog Output.



#### 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.

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.

*Continues on next page*

### 3 System parameters

---

#### 3.2.2 The type Arc System Properties

*Continued*

##### WELD\_UNITS

-MaxLog	
% (100 max)	-MaxLog = $100 \cdot 1000 / 60 \cdot 1000 = 1.67$
Max value in <b>Program Data</b>	100
Max value in RAPID code	$100 \cdot 1000 / 60 = 1666.67$
wfspeed (22m/min max)	-MaxLog = $22 / 60 = 0.367$
Max value in <b>Program Data</b>	22 (m/min)
Max value in RAPID code	$22 \cdot 1000 / 60 = 367$ (mm/s)

##### US\_UNITS

-MaxLog	
% (100 max)	-MaxLog = $100 \cdot 25.4 / 60 \cdot 1000 = 0.0423$
Max value in <b>Program Data</b>	100
Max value in RAPID code	$100 \cdot 25.4 / 60 = 42.3$
wfspeed (22m/min max)	-MaxLog = $22 / 60 = 0.367$
Max value in <b>Program Data</b>	$22 \cdot 1000 / 25.4 = 866.141$ (ipm)
Max value in RAPID code	$22 \cdot 1000 / 60 = 367$ (mm/s)

##### SI\_UNITS

-MaxLog	
% (100 max)	-MaxLog = $100 / 1000 = 0.1$
Max value in <b>Program Data</b>	100
Max value in RAPID code	100
wfspeed (22m/min max)	-MaxLog = $22 / 60 = 0.367$
Max value in <b>Program Data</b>	$22 \cdot 1000 / 60 = 367$ (mm/s)
Max value in RAPID code	$22 \cdot 1000 / 60 = 367$ (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 (\*)

Value on the wirefeed Analog Output: 1.67 V

*Continues on next page*

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.

## 3 System parameters

### 3.2.3 The type Arc Units

### 3.2.3 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: <ul style="list-style-type: none"><li>• mm</li><li>• inch</li></ul>
Arc Velocity unit	string	The available velocity units are: <ul style="list-style-type: none"><li>• mm/s</li><li>• m/min</li><li>• ipm</li><li>• cm/min</li></ul>
Arc Feed unit	string	The available feed units are: <ul style="list-style-type: none"><li>• mm/s</li><li>• m/min</li><li>• ipm</li></ul>

ArcWare for OmniCore 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

### 3.2.4 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 <i>Arc Equipment</i> . 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
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.

### 3 System parameters

---

#### 3.2.5 The type Arc Equipment Class

#### 3.2.5 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 <i>Arc Equipment Class</i> .
Equipment Class File Name	awEquipSTD	string	The name of the EquipmentClass module to load.
Path	RELEASE:/options/arc/WeldEquipment	string	The path to the equipment class.



### 3.3 The group Generic Equipment Class

#### 3.3.1 The type Arc Equipment Properties

##### Description

The *Arc Equipment Properties* holds parameters for the equipment class.

##### Parameters

The following parameters can be defined in ArcWare for OmniCore.

Parameter	Data type	Note
Name	string	The name of the <i>Arc Equipment Properties</i> .
Use Arc Equipment IO DI	string	The Arc Equipment IO DI used by the <i>Arc Equipment Properties</i> .
Use Arc Equipment IO DO	string	The Arc Equipment IO DO used by the <i>Arc Equipment Properties</i> .
Use Arc Equipment IO AO	string	The Arc Equipment IO AO used by the <i>Arc Equipment Properties</i> .
Use Arc Equipment IO AI	string	The Arc Equipment IO AI used by the <i>Arc Equipment Properties</i> .
Use Arc Equipment IO GO	string	The Arc Equipment IO GO used by the <i>Arc Equipment Properties</i> .
Preconditions On	bool	Specifies whether preconditions is to be used. If precondition 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, <i>Component group: Ignition</i> in seamdata ( <a href="#">seamdata - Seam data on page 166</a> ) 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, <i>Component group: Heat</i> in seamdata ( <a href="#">seamdata - Seam data on page 166</a> ) 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

*Continues on next page*

### 3 System parameters

#### 3.3.1 The type Arc Equipment Properties

*Continued*

Parameter	Data type	Note
Fill On	bool	<p>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 <a href="#">seamdata - Seam data on page 166</a>. If the Crater fill parameter is changed, the contents of seamdata will also change.</p> <p>Default value: FALSE</p>
Burnback On	bool	<p>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 <a href="#">seamdata - Seam data on page 166</a>.</p> <p>If the Burnback parameter is changed, the contents of seamdata will also change. If burnback is set, <code>bback_time</code> in seamdata (<a href="#">seamdata - Seam data on page 166</a>) is available.</p> <p>If both, burnback and burnback voltage, are set, <code>bback_voltage</code> in seamdata (<a href="#">seamdata - Seam data on page 166</a>) is available.</p> <p>Default value: FALSE</p>
Burnback Voltage On	bool	<p>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.</p> <p>If the Burnback voltage parameter is changed, the contents of seamdata will also change.</p> <p>If both, burnback and burnback voltage, are set, <code>bback_voltage</code> in seamdata (<a href="#">seamdata - Seam data on page 166</a>) is available.</p> <p>Default value: FALSE</p>
Rollback On	bool	<p>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.</p> <p>If the Rollback parameter is changed, the contents of seamdata will also change.</p> <p>If rollback is set, <code>rback_time</code> in seamdata (<a href="#">seamdata - Seam data on page 166</a>) is available.</p> <p>If both, rollback and rollback wirefeed, are set, <code>rback_wirefeed</code> in seamdata (<a href="#">seamdata - Seam data on page 166</a>) is available.</p> <p>Default value: FALSE</p>

*Continues on next page*

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, <code>rback_wirefeed</code> in seamdata ( <a href="#">seamdata - Seam data on page 166</a> ) 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
Heat as time	bool	Specifies if the heat phase should use the seamdata parameters <code>heat_time</code> or <code>heat_distance</code> . TRUE means that <code>heat_time</code> is used and visible in the seamdata. FALSE means that <code>heat_distance</code> and <code>heat_speed</code> 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 <i>welder ready</i> active (high), supervision in the main welding phase can be used. Set to TRUE to activate. Default value: FALSE
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
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 ArcWare for OmniCore 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.

### 3 System parameters

#### 3.3.2 The type Arc Equipment Digital Inputs

#### 3.3.2 The type Arc Equipment Digital Inputs

##### Parameters

The following digital inputs can be defined in ArcWare for OmniCore.

Parameter	Data type	Note
Name	string	The name of the <i>Arc Equipment Digital Inputs</i> .
ManFeedInput	signal di	Digital input signal for manual wire feed. A high signal means that the welding equipment has manual wire feed enabled.
WeldInhib	signal di	Digital input signal for program execution without welding. A high signal means that welding is inhibited.
WeaveInhib	signal di	Digital input signal for program execution without weaving. A high signal means that weaving is inhibited.
TrackInhib	signal di	Digital input signal to inhibit tracking (not seen on FlexPendant). A high signal means that the tracking is inhibited.
SupervInhib	signal di	Digital input signal to inhibit supervision (not seen on FlexPendant). A high signal means that supervision is inhibited.
StopProc	signal di	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	signal di	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 <a href="#">Configurable error handling on page 39</a> .
ArcEst2	signal di	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 <a href="#">Configurable error handling on page 39</a> .
WelderReady	signal di	Digital input signal for supervision of the <i>WelderReady</i> signal.
WelderReadyLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See <a href="#">Configurable error handling on page 39</a> .
WeldOk	signal di	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 <a href="#">Configurable error handling on page 39</a> .
VoltageOk	signal di	Digital input signal for supervision of the voltage. A high signal means that the voltage is OK.

*Continues on next page*

Parameter	Data type	Note
VoltageOkLabel	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See <a href="#">Configurable error handling on page 39</a> .
CurrentOk	signal	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 <a href="#">Configurable error handling on page 39</a> .
WaterOk	signal	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 <a href="#">Configurable error handling on page 39</a> .
WirefeedOk	signal	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 <a href="#">Configurable error handling on page 39</a> .
GasOk	signal	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 <a href="#">Configurable error handling on page 39</a> .
GunOk	signal	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 <a href="#">Configurable error handling on page 39</a> .
WirestickErr	signal	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 <a href="#">Configurable error handling on page 39</a> .
USERIO1	signal	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 <a href="#">Configurable error handling on page 39</a> .
USERIO2	signal	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*

### 3 System parameters

#### 3.3.2 The type Arc Equipment Digital Inputs

*Continued*

Parameter	Data type	Note
USERIO2Label	string	Label describing the error level of the signal. There are three available levels, MAJOR, MINOR and INFO. See <a href="#">Configurable error handling on page 39</a> .
USERIO3	signal	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 <a href="#">Configurable error handling on page 39</a> .
USERIO4	signal	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 <a href="#">Configurable error handling on page 39</a> .
USERIO5	signal	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 <a href="#">Configurable error handling on page 39</a> .



#### 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*

## 3.3.3 The type Arc Equipment Digital Outputs

## Parameters

The following digital outputs can be defined in ArcWare for OmniCore.

Parameter	Data type	Note
Name	string	The name of the <i>Arc Equipment Digital Outputs</i> .
AWEError	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 <i>RobotReady</i> 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.

*Continues on next page*

### 3 System parameters

---

#### 3.3.3 The type Arc Equipment Digital Outputs

*Continued*

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.



#### 3.3.4 The type Arc Equipment Analog Inputs

##### Parameters

The following analog inputs can be defined in ArcWare for OmniCore.

Parameter	Data type	Note
Name	string	The name of the <i>Arc Equipment Analog Inputs</i> .
VoltageMeas	signalai	Analog input signal for voltage measurement.
CurrentMeas	signalai	Analog input signal for current measurement.

## 3 System parameters

### 3.3.5 The type Arc Equipment Analog Outputs

### 3.3.5 The type Arc Equipment Analog Outputs

#### Parameters

The following analog outputs can be defined in ArcWare for OmniCore.

Parameter	Data type	Note
Name	string	The name of the <i>Arc Equipment Analog Outputs</i> .
VoltReference	signalao	Analog output signal for analog voltage reference. If weld voltage is defined, the component <code>weld_voltage</code> in <code>welddata</code> ( <a href="#">welddata - Weld data on page 180</a> ) 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 <code>weld_wirefeed</code> in <code>welddata</code> ( <a href="#">welddata - Weld data on page 180</a> ) is available.
CurrentReference	signalao	Analog output signal for analog current reference. If current is defined, the component <code>weld_current</code> in <code>welddata</code> ( <a href="#">welddata - Weld data on page 180</a> ) 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 <code>weld_voltage</code> in <code>welddata</code> ( <a href="#">welddata - Weld data on page 180</a> ) 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 <code>weld_wirefeed</code> in <code>welddata</code> ( <a href="#">welddata - Weld data on page 180</a> ) is available.
ControlPort2	signalao	Tunable analog output2 for certain welders. Used in TwinWire Systems.

#### 3.3.6 The type Arc Equipment Group Outputs

---

##### Parameters

The following group outputs can be defined in ArcWare for OmniCore.

Parameter	Data type	Note
Name	string	The name of the <i>Arc Equipment Group Outputs</i> .
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.

## 3 System parameters

---

### 3.4.1 The type Optical Sensor

## 3.4 The group Optical Sensor

### 3.4.1 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 <i>Optical Sensor</i> .
Use Optical Sensor Class	string	The <i>Optical Sensor Class</i> used by the <i>Arc Sensor Class</i> .
Use Optical Sensor Properties	string	The optical sensor properties used by the <i>Optical Sensor</i> . Two sensor properties are available: MSPOT90 and SCOUT.
Connected to Robot	string	The robot to which the sensor is connected.

## 3.4.2 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 <i>Arc Sensor Class</i> .
Physical Sensor	string	The name of the physical sensor used with ArcWare.
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 Incremental Correction	num	Max incremental correction for the arc tracking system. If the incremental TCP correction is bigger than <i>Max Incremental Correction</i> and <i>Max Correction Warning</i> was set, the robot will continue its path but the applied incremental correction will not exceed <i>Max Incremental Correction</i> . If <i>Max Correction Warning</i> 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.
Logfile Size	num	Defines the size of the log file created during tracking. Default value is 1000.
Correction Filter	num	Defines filtering of the correction calculated, using mean value over corr filter values. Use default value: 1
Error Ramp In	num	Defines during how many sensor readings ramp in is done after an error caused by sensor reading.
Error Ramp Out	num	Defines during how many sensor readings ramp out is done after an error caused by sensor reading.
Calib Variable Name (pose)	string	The name of the persistent calibration variable holding the result of the calibration. RAPID data type: pose
Calib Variable Name (pos)	num	The name of the persistent calibration variable holding the result of the calibration. RAPID data type: pos

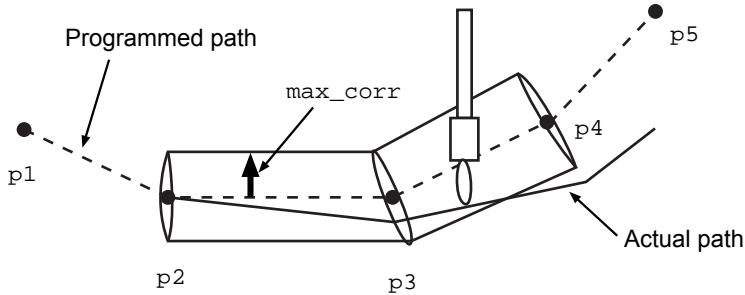
*Continues on next page*

### 3 System parameters

#### 3.4.2 The type Optical Sensor Properties

*Continued*

Parameter	Data type	Note
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
LeftSync	string	Digital output signal for left syncpulse.
RightSync	string	Digital output signal for right syncpulse.



xx1200000686

**Max correction**

### 3.5 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

The following table shows the system behavior.

Label	SYS_STOP	In error hand- ler	Process shut- down	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.

#### 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:=idl,v300,sm1,wd2\Weave:=wv2,z5,
    Rob1_tool\WObj:=wobj_STN1;
```

*Continues on next page*

### 3 System parameters

---

#### 3.5 Configurable error handling

*Continued*

```
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
```



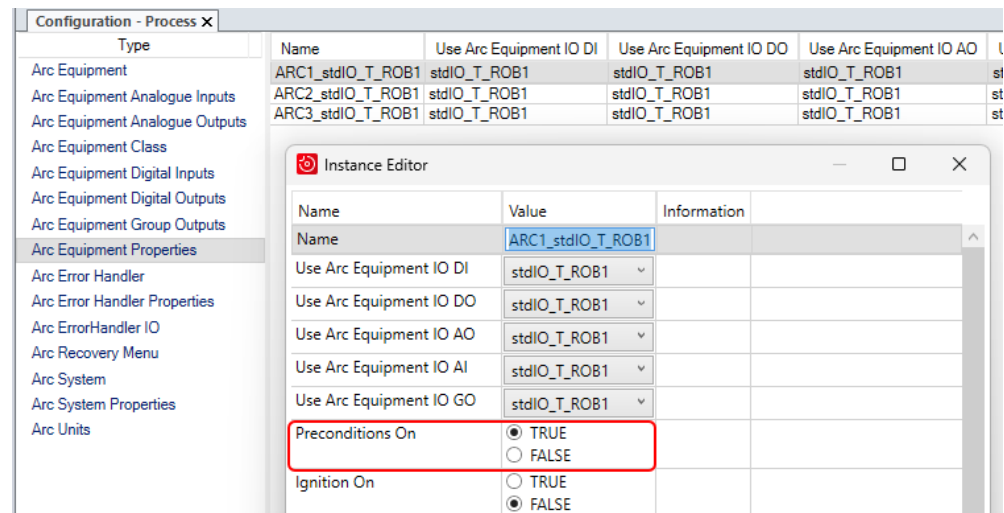
### 3.6 Welder Ready Supervision for StdIoWelder interface

#### 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.



xx2400000123

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. This output can be configured in the system parameters, topic *Process*, type *Arc Equipment Digital Outputs*, for the *WelderReady* instance.

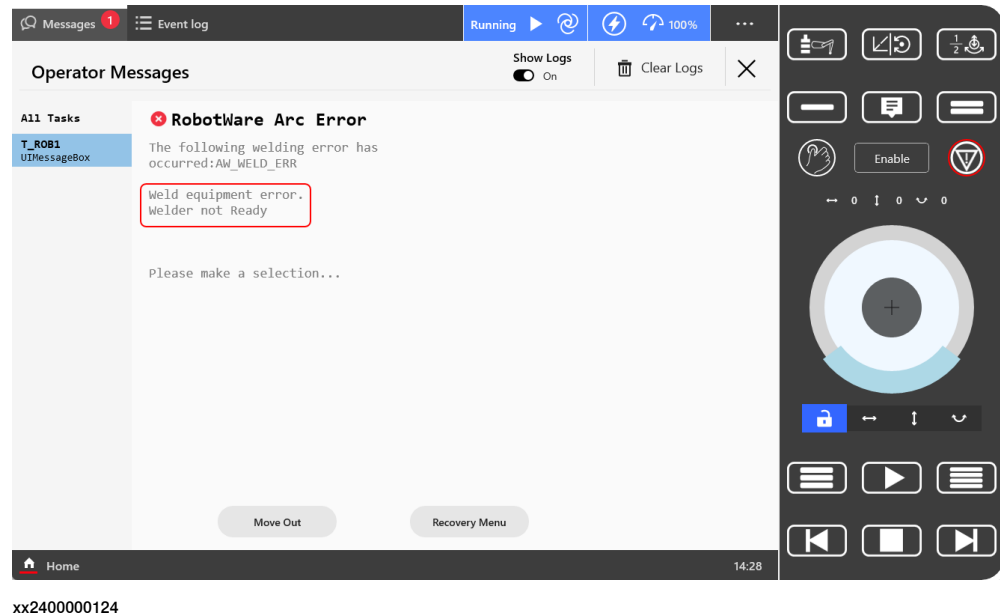
*Continues on next page*

### 3 System parameters

#### 3.6 Welder Ready Supervision for StdIoWelder interface

*Continued*

##### Error message on FlexPendant

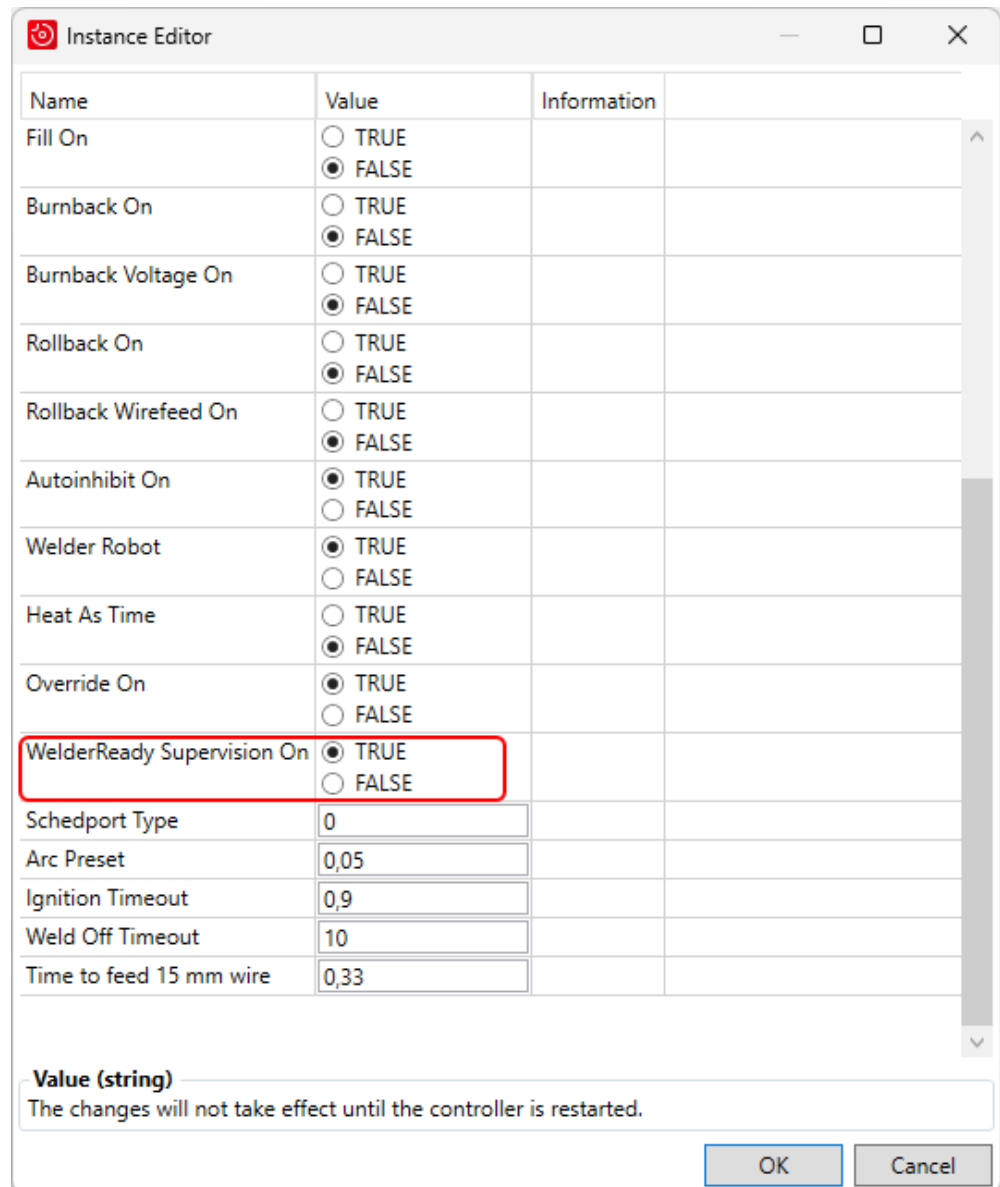


##### Welder Ready supervision while welding is active

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.

*Continues on next page*



The screenshot shows the 'Instance Editor' window with a table of parameters. The 'WelderReady Supervision On' parameter is highlighted with a red box. Below the table, there is a 'Value (string)' section with a message: 'The changes will not take effect until the controller is restarted.' At the bottom right are 'OK' and 'Cancel' buttons.

Name	Value	Information
Fill On	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Burnback On	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Burnback Voltage On	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Rollback On	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Rollback Wirefeed On	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Autoinhibit On	<input checked="" type="radio"/> TRUE <input type="radio"/> FALSE	
Welder Robot	<input checked="" type="radio"/> TRUE <input type="radio"/> FALSE	
Heat As Time	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Override On	<input checked="" type="radio"/> TRUE <input type="radio"/> FALSE	
WelderReady Supervision On	<input checked="" type="radio"/> TRUE <input type="radio"/> FALSE	
Schedport Type	0	
Arc Preset	0,05	
Ignition Timeout	0,9	
Weld Off Timeout	10	
Time to feed 15 mm wire	0,33	

**Value (string)**  
The changes will not take effect until the controller is restarted.

OK Cancel

xx2400000125

**This page is intentionally left blank**

## 4 Programming

### 4.1 Programming for arc welding

#### Prerequisites

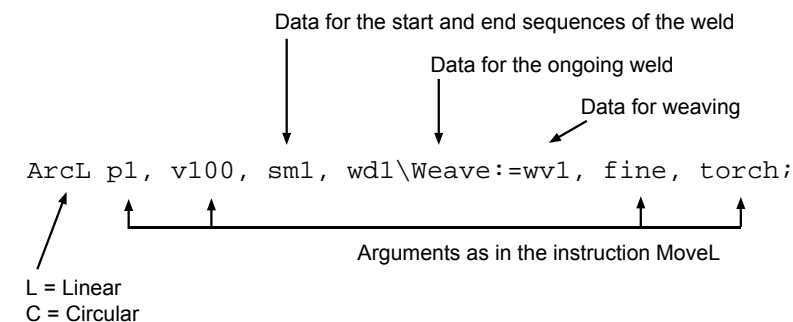
Before creating an arc welding program, the arc welding system or systems (see [Defining arc welding systems on page 16](#)) and additional axes, if any, must be configured. This configuration is described in [System parameters on page 15](#).

#### 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`.



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 47](#).

#### 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*

## 4 Programming

### 4.1 Programming for arc welding

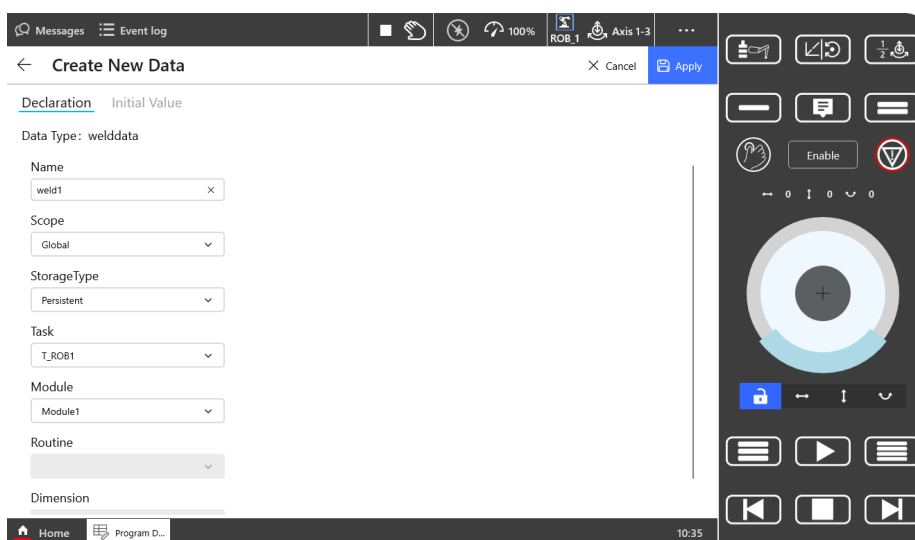
*Continued*

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** application from the ABB Robotics main page on the OmniCore FlexPendant.
- 2
- 3 Select the type `seamdata`, `welddata`, or `weavedata`.
- 4 Select **Create New Data**.

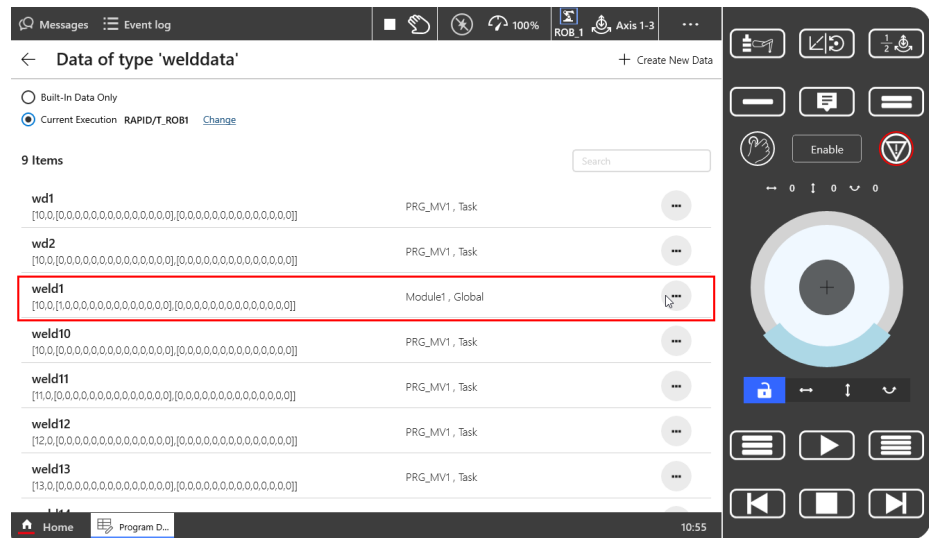
The data properties are displayed.



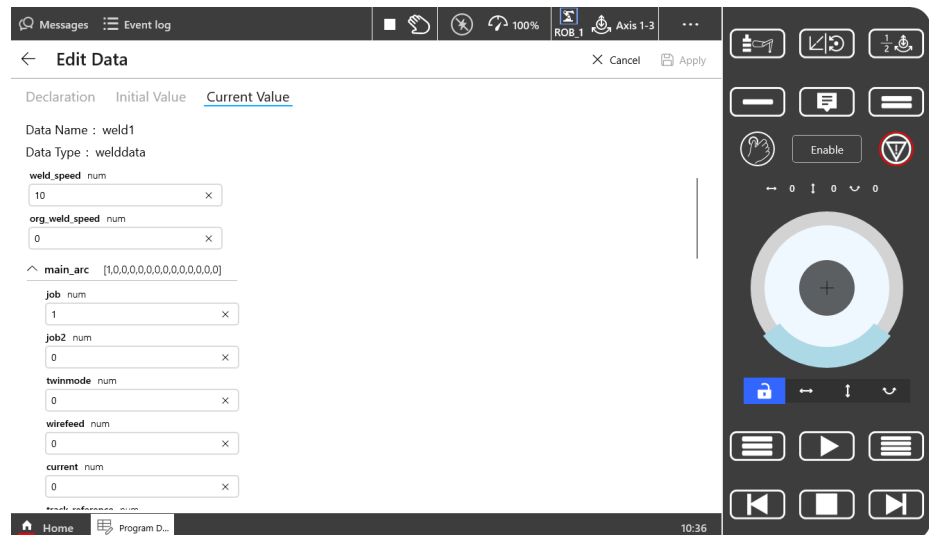
xx2300002018

- 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 **Apply**.
- 8 The data will appear in the list with `welddata` variables. To change the Current value, tap the data.

*Continues on next page*



xx2300002019



xx2300002020



#### 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 **Code Editor**, select **Groups** and picklist **Arc**.
- 3 Select the instruction **ArcLStart**, **ArcL**, **ArcLEnd**, **ArcC** or **ArcCEnd**.  
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.

*Continues on next page*

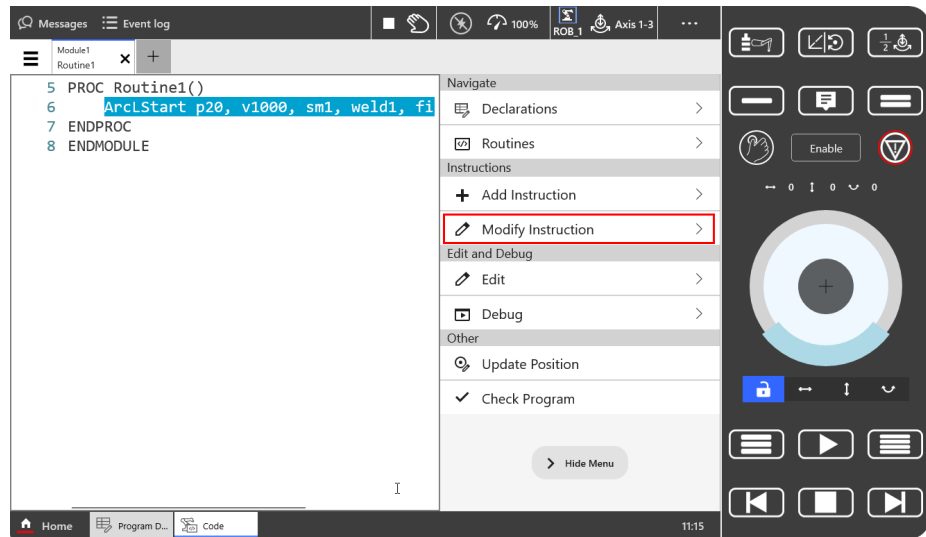
## 4 Programming

### 4.1 Programming for arc welding

*Continued*

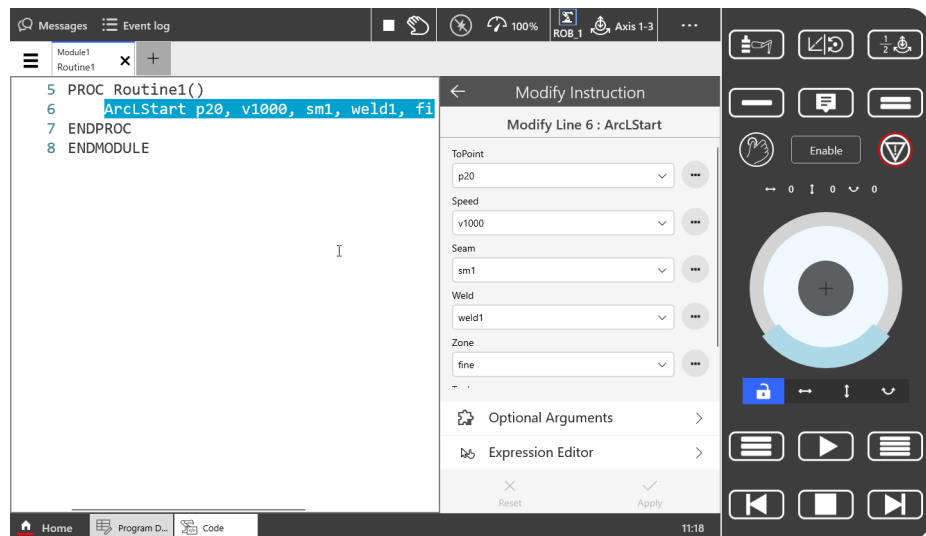
If an argument needs to be changed, the data can be replaced by another.

- 1 Select the instruction you wish to change (`ArcLStart` in this example).



xx2300002021

- 2 When the instruction line is selected, tap **Modify Instruction** on the **Navigate** menu. The window used to change instruction arguments appears.

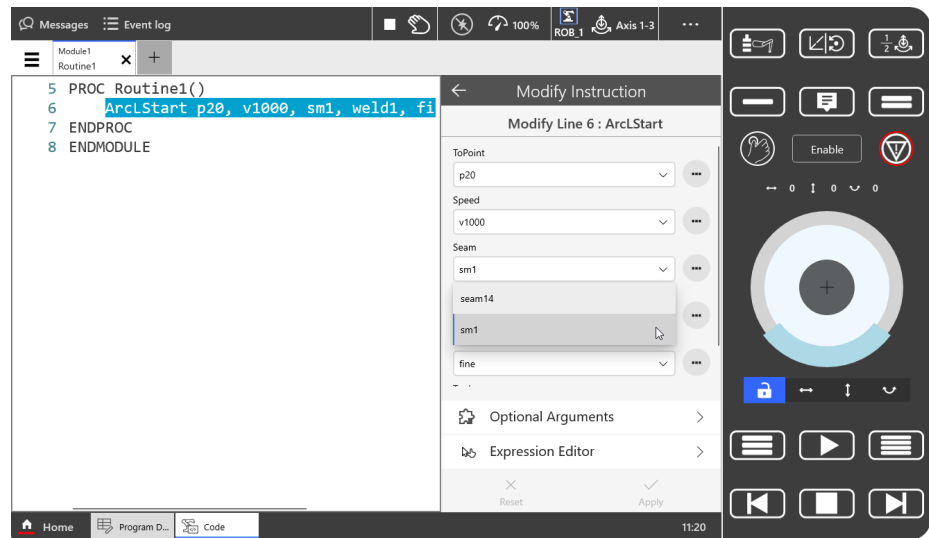


xx2300002022

The selected argument is highlighted see figure below. The drop down displays all available seamdata that can be selected

*Continues on next page*





xx2300002023

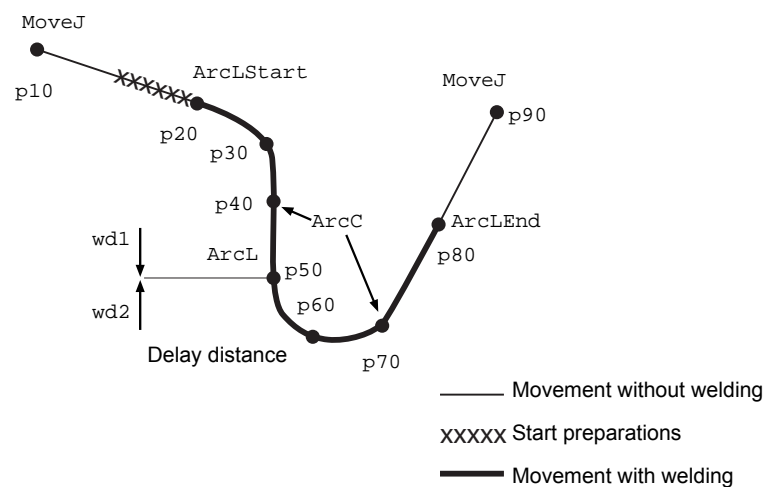
- 3 Select the desired seamdata.
- 4 Change another argument by tapping on the argument in the instruction.
- 5 Repeat this for all arguments that needs to be changed.
- 6 Tap **Apply** to confirm the changes.

#### 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.



xx1200000645

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

*Continues on next page*

## 4 Programming

---

### 4.1 Programming for arc welding

*Continued*

```
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 - Arc welding with linear motion on page 115](#), and [ArcC - Arc welding with circular motion on page 133](#).

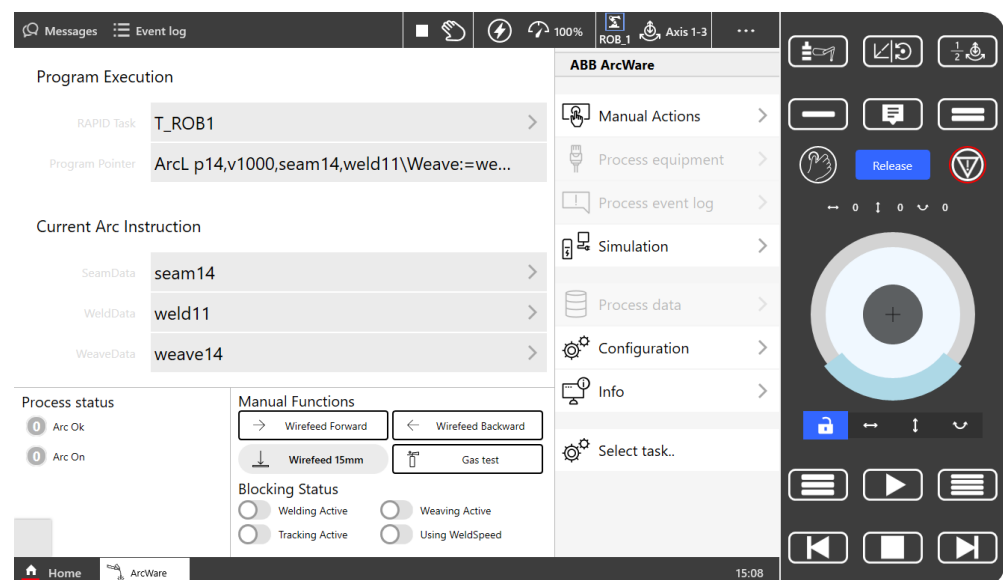
## 4.2 FlexPendant WebApp Interface

### Starting the WebApp

To start the ArcWare for OmniCore WebApp, tap the Arc Application on the main page. As soon as it is loaded, all arc welding functions can be accessed.



xx2300002024



xx2300002025

The application consists of four areas, with sub menus.

- runtime display area
- process status area
- manual functions area
- ABB ArcWare area

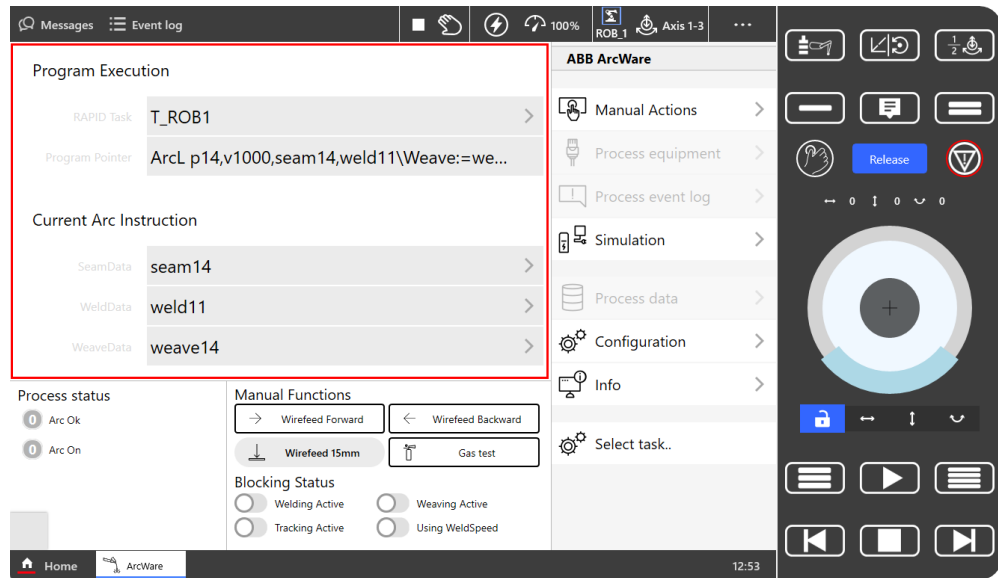
*Continues on next page*

## 4 Programming

### 4.2 FlexPendant WebApp Interface

*Continued*

#### Runtime data area



xx2300002026

In this area you find information about

#### Program execution

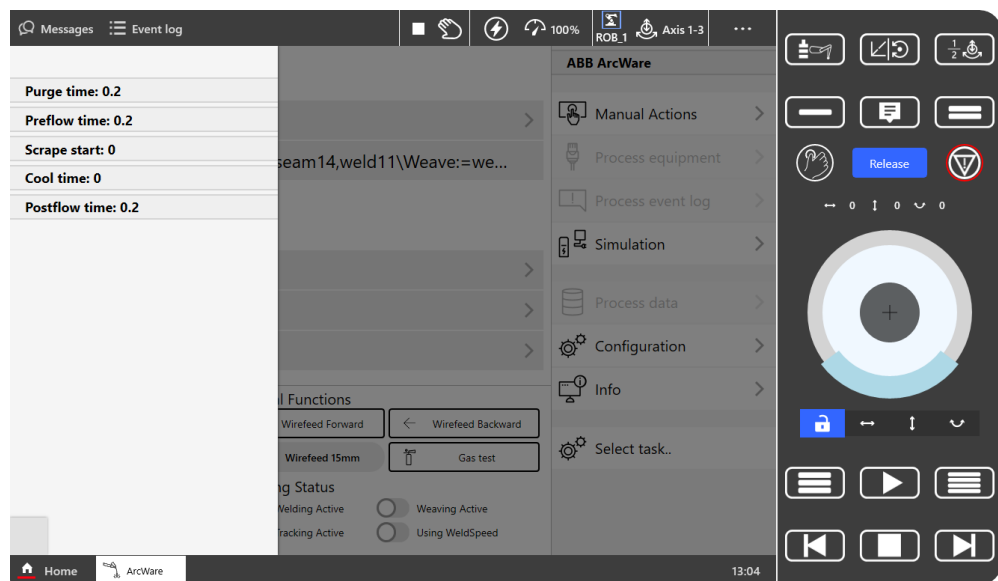
- 1 Selected RAPID task
- 2 Program pointer

#### Current Arc instruction

Active data in the current instruction

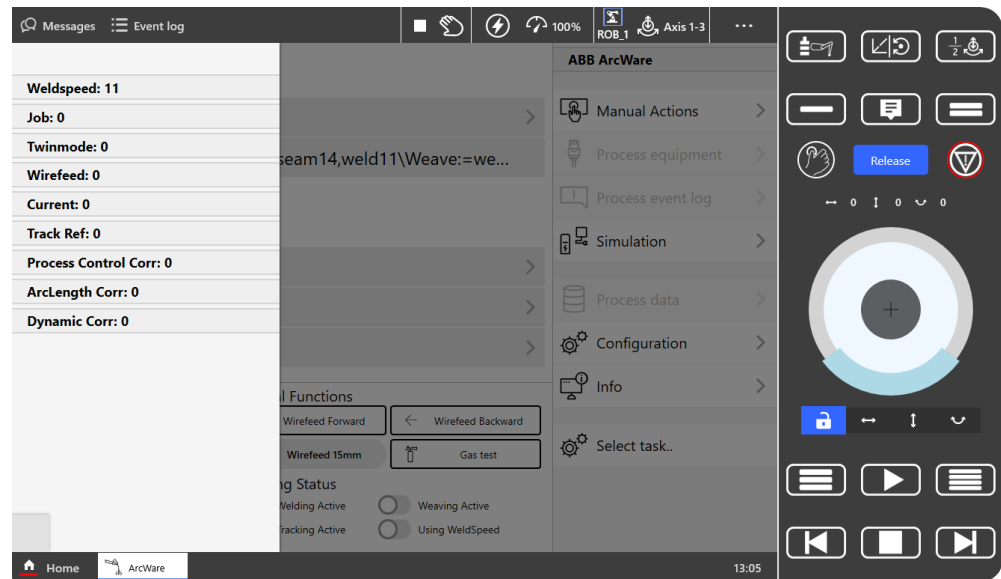
- seamdata
- welddata
- weavedata

The active data can be expanded to view all its components.

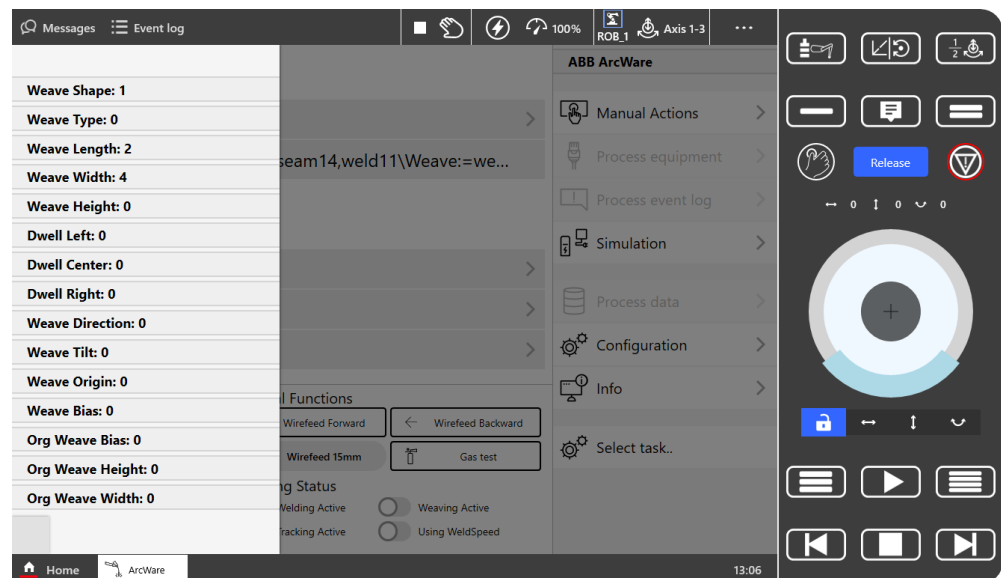


xx2300002027

*Continues on next page*



xx2300002028



xx2300002029

#### Process status area

- 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.



#### 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.

*Continues on next page*

## 4 Programming

---

### 4.2 FlexPendant WebApp Interface

*Continued*

---

#### 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.



#### 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.

## 5 Programming RobotWare Arc systems with MultiMove

### 5.1 RobotWare Arc with MultiMove

---

#### Introduction

The ArcWare for OmniCore 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 coordinated.

The user interface provides the possibility to select which welding robot the functions should operate on.

### 5.2 Functions for arc welding during program execution

#### Functions

Arc welding functions during program execution:

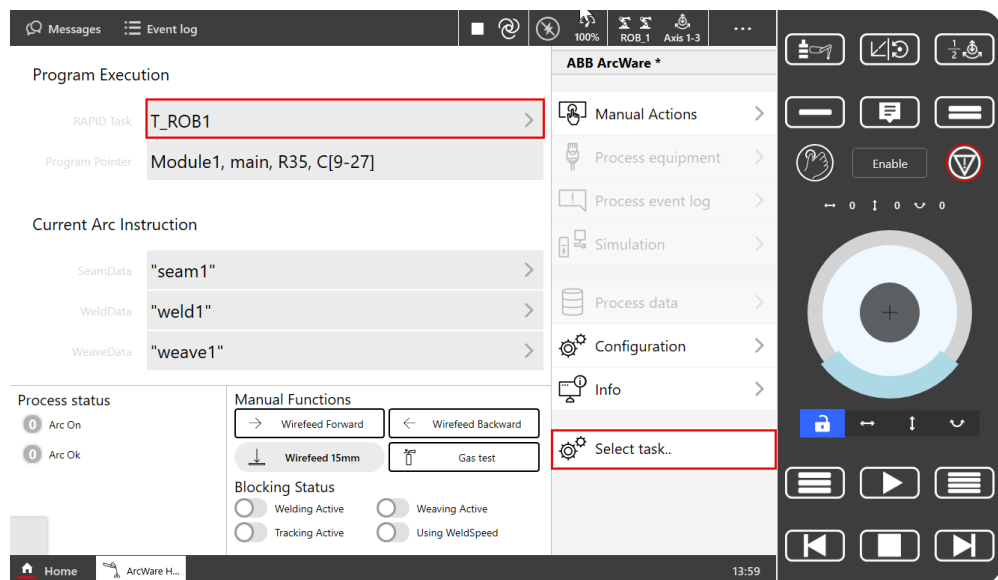
- Selecting active welding robot

#### Selecting active welding robot

It is possible to change active welding robot.

- 1 Open the menu for **RAPID Task** under **Program Execution**  
or  
Tap **Select task...**
- 2 Select the welding robot from the task list.

Manual functions and process blocking will now operate on data belonging to the active welding robot.



xx240000903



## 5.3 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, wdl_ind\Weave:=wv0, fine,
  Rob2_tool\Wobj:=wobj_STN1;
ArcLEnd p6, v1000, sm1, wdl_ind\Weave:=wv0, fine,
  Rob2_tool\Wobj:=wobj_STN1;
ERROR
StorePath;
```

*Continues on next page*

### 5.3 Configuration

#### Continued

```
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 robtargt errPos1;  
VAR tooldata tErr;  
VAR wobjdata obErr;  
MoveJ p1, v1000, fine, Rob2_tool\WObj:=wobj_STN1;  
ArcLStart p2, v1000, sml, wdl_ind\Weave:=wv0, fine,  
    Rob2_tool\WObj:=wobj_STN1;  
ArcLEnd p6, v1000, sml, wdl_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 robtargt errPos1;  
VAR tooldata tErr;  
VAR wobjdata obErr;  
MoveJ p1, v1000, fine, Rob2_tool\WObj:=wobj_STN1;  
ArcLStart p2, v1000, sml, wdl_ind\Weave:=wv0, fine,  
    Rob2_tool\WObj:=wobj_STN1;  
ArcLEnd p6, v1000, sml, wdl_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;
```

*Continues on next page*

```
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;
```

*Continues on next page*

## 5 Programming RobotWare Arc systems with MultiMove

---

### 5.3 Configuration

*Continued*

---

#### Move instructions

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

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 [Configurable error handling on page 39](#).

### 5.4 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*

### 5.4 Limitations

*Continued*

```
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 with ArcWare for OmniCore and one or several additional axis [or positioners, the following instructions must be used to have proper error handling.

- `ArcMoveExtJ`
- `ArcMoveAbsJ`
- `ArcMoveL`
- `ArcMoveC`
- `ArcMoveJ`

For the tasks `T_POS1` and `T_POS2`, if they exist, these instructions are installed automatically during installation of ArcWare for OmniCore. For other tasks the RAPID module `awBase.sysx` needs to be installed by adding the following in the controller configuration (*taskname* represents the name of the additional axis RAPID task name):

```
CAB_TASK_MODULES:
#
-File "ARC:/ArcBase/Code/awBase.sysx" \
-ModName "awBase" -Install -Task "taskname"
```

## 6 Weld Error Recovery

### 6.1 Weld Error Recovery and error handling

#### Weld Error Recovery

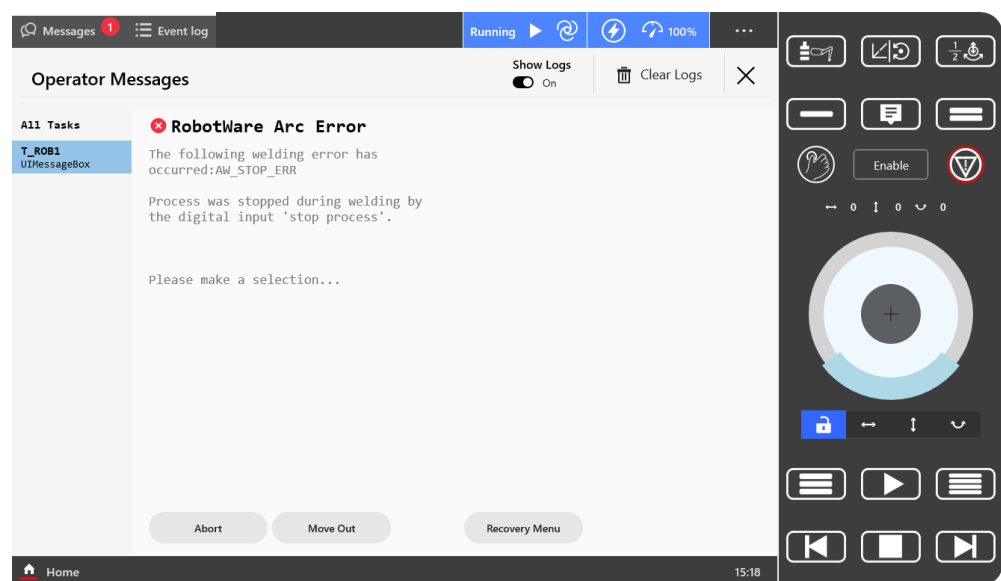
During robot production process errors sometimes stop the robot. The Weld Error Recovery feature provides several different solutions for process error recovery, which allows operators to automatically move the robot out from the error position to get a better overview of the torch. After the process 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 ArcWare for OmniCore, 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.



xx2400000105

*Continues on next page*

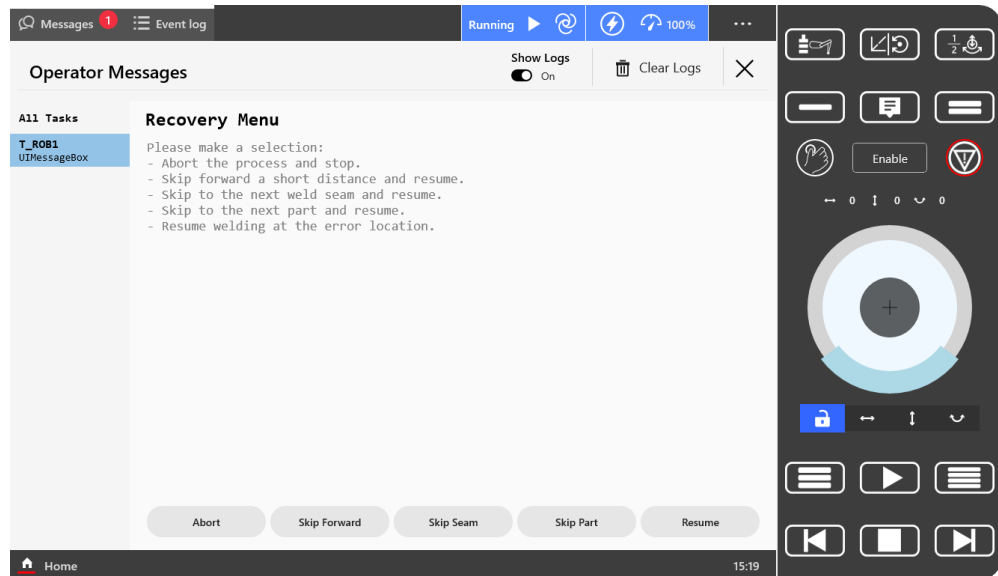
## 6 Weld Error Recovery

### 6.1 Weld Error Recovery and error handling

*Continued*

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

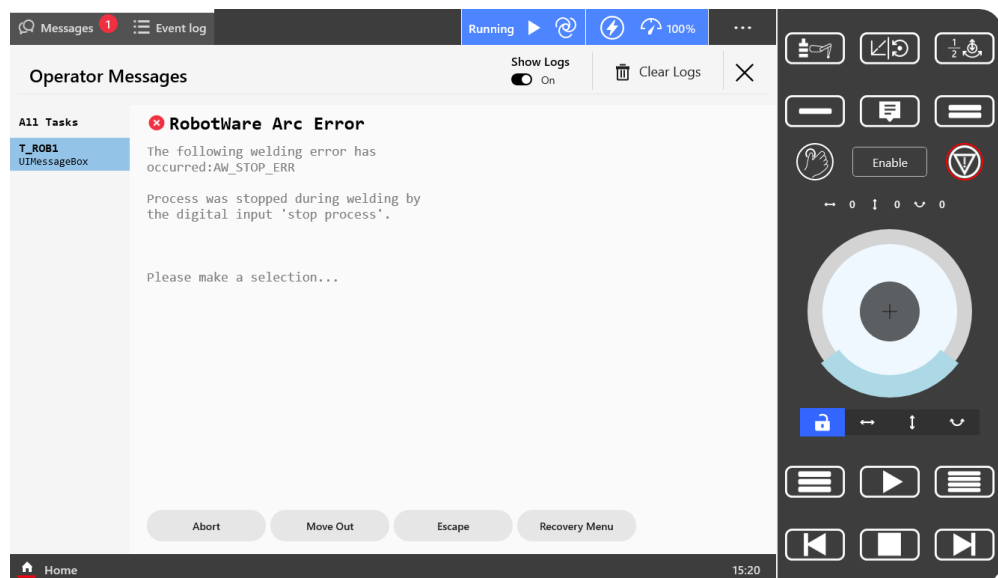
If **Move Out** is tapped the robot will attempt to move out a small distance along the tool center line. The **Error** menu will be shown again. **Move Out** can be tapped repeatedly. If **Recovery** menu is tapped, the user is presented with the **Recovery** menu.



xx2400000106

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.



xx2400000107



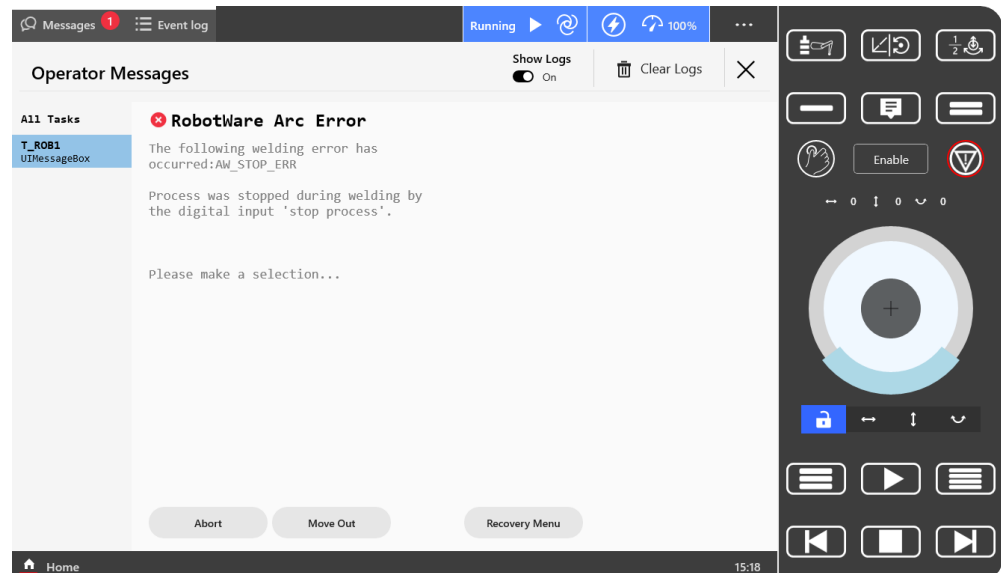
## 6.2 Programming Weld Error Recovery

### 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:



xx2400000105

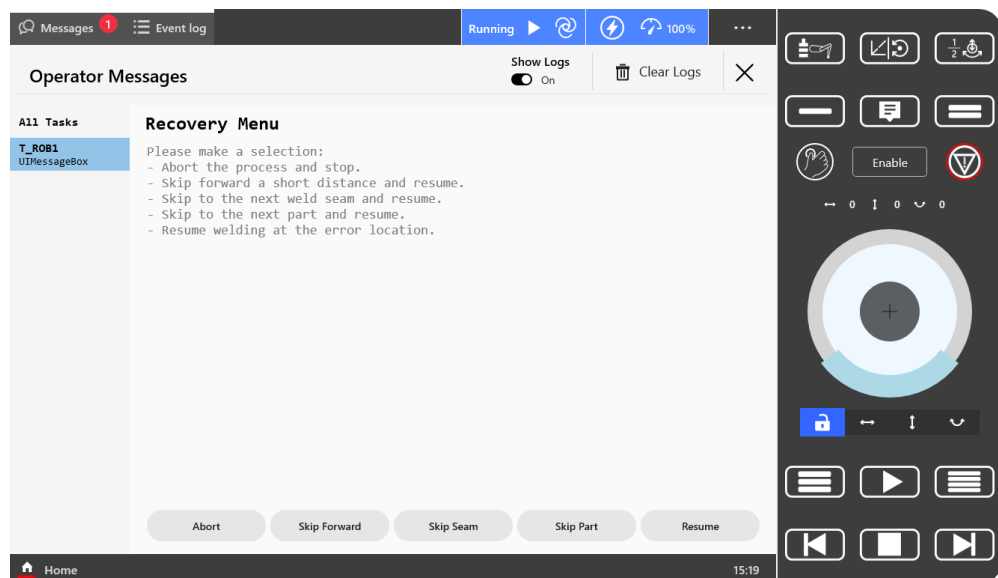
The user can tap **Move Out** to extract the tool from the partially welded part in increments. Tapping **Abort**, stops execution. Tapping **Recovery** menu will bring up the **Recovery** menu.

*Continues on next page*

## 6 Weld Error Recovery

### 6.2 Programming Weld Error Recovery

*Continued*



xx2400000106

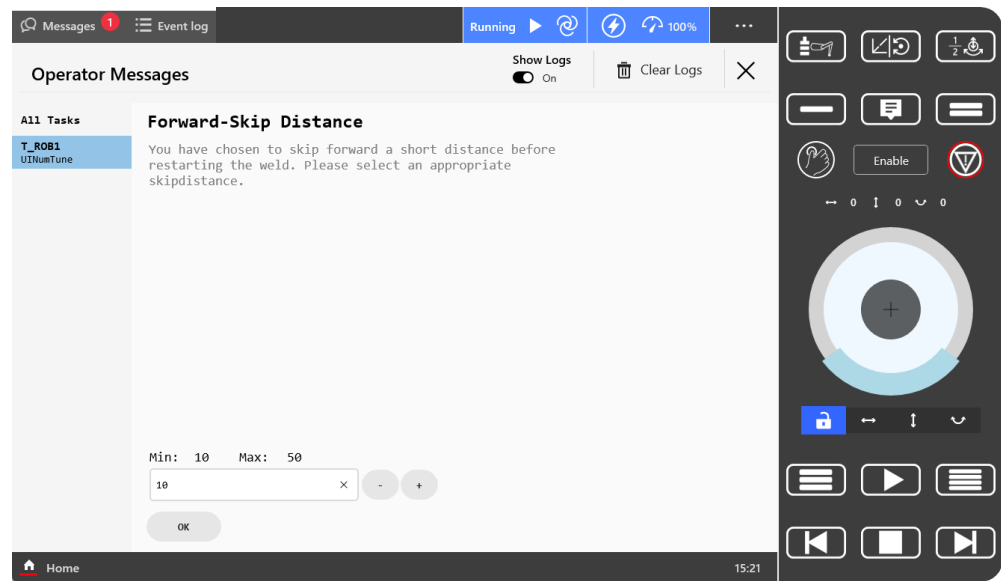
When **Resume** is selected the robot executes a standard retry with the configured restart distance.

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.

*Continues on next page*



xx2400000109

#### 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
```

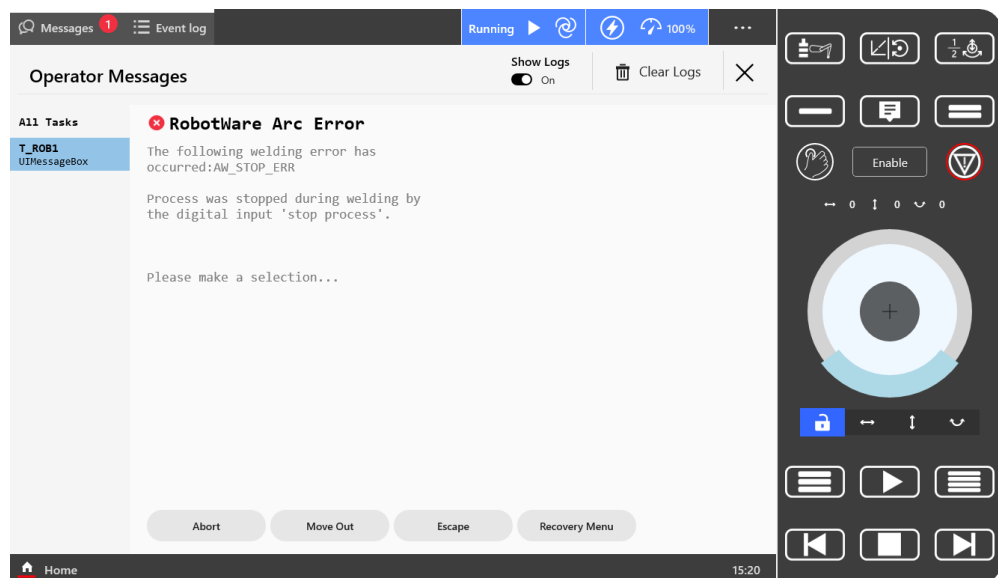
The instruction `RecoveryPosSet` is used to set the recovery set point. If an error occurs during the weld seam, the error menu will display an **Escape** button:

*Continues on next page*

## 6 Weld Error Recovery

### 6.2 Programming Weld Error Recovery

*Continued*



xx2400000107

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;  
  ENDTST  
  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;
```

*Continues on next page*

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,sml,wdl\Weave:=wv1,fine,tWeldGun;  
  ArcL *,v500,sml,wdl\Weave:=wv1,z10,tWeldGun;  
  SetDO doClamp,high;  
  RecoveryPosSet;  
  ArcL *,v500,sml,wdl\Weave:=wv1,z10,tWeldGun;  
  ArcLEnd *,v500,sml,wdl\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,sml,wdl\Weave:=wv1,fine,tWeldGun;  
  ArcL *,v500,sml,wdl\Weave:=wv1,z10,tWeldGun;  
  ArcL *,v500,sml,wdl\Weave:=wv1,z10,tWeldGun;  
  ArcLEnd *,v500,sml,wdl\Weave:=wv1,fine,tWeldGun;  
  MoveJ *,vmax,z10,tWeldGun;
```

*Continues on next page*

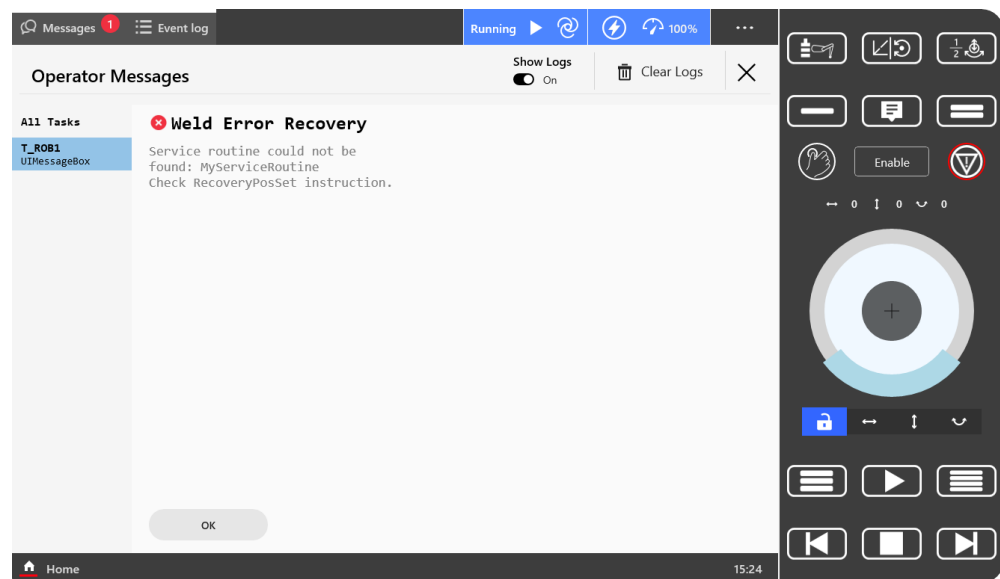
## 6 Weld Error Recovery

### 6.2 Programming Weld Error Recovery

*Continued*

```
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 `ServiceRoutine` 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.

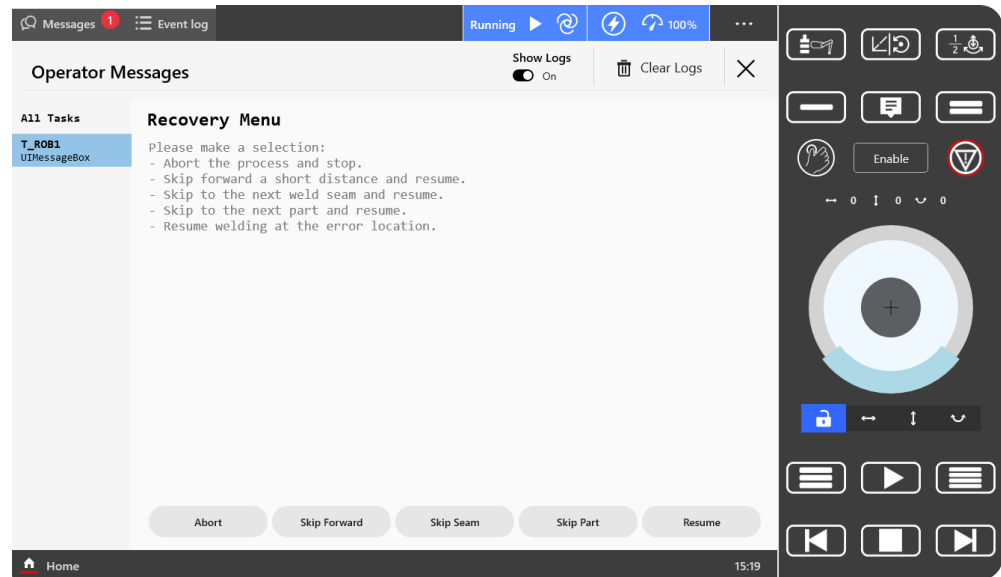


xx2400000110

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

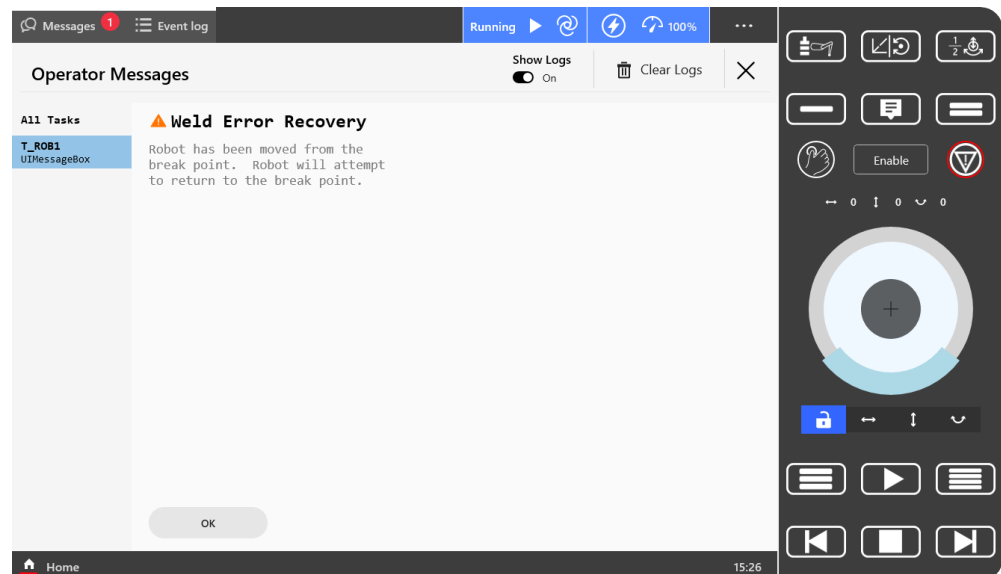
The *ServiceRoutine* example above is an example of a service routine that can be created by a RAPID programmer. The Weld Error Recovery feature does not provide the *ServiceRoutine* procedure. In this example the service routine contains move instructions that move the robot from the safe position,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

*Continues on next page*



xx2400000106

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.



xx2400000111

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.

#### 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.

*Continues on next page*

## 6 Weld Error Recovery

---

### 6.2 Programming Weld Error Recovery

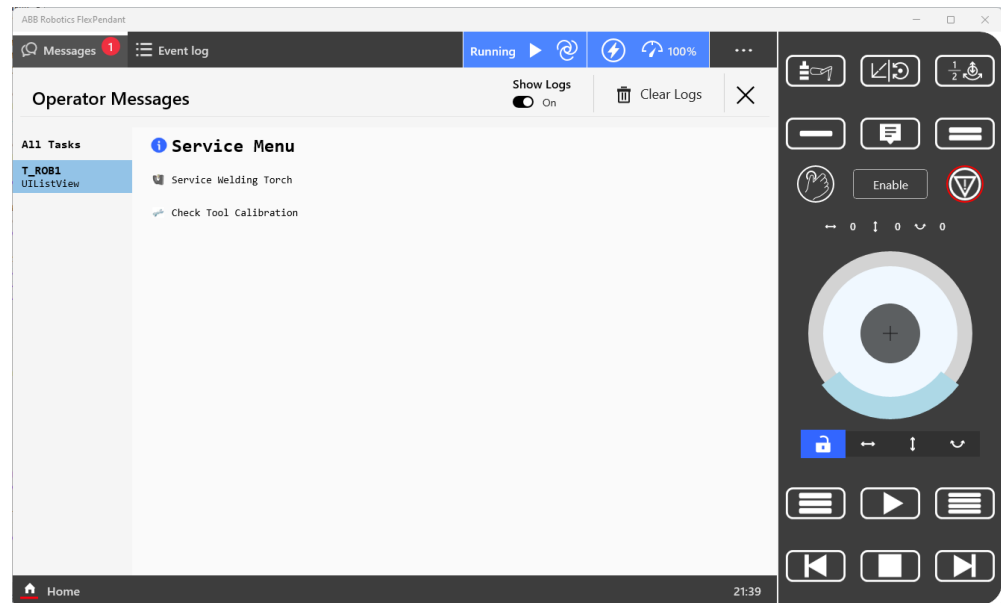
*Continued*

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

*Continues on next page*





xx2400000112

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.



## 6.4 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.

The screenshot shows the 'Instance Editor' dialog box with the following configuration:

Name	Value	Information
Name	default	
Use Arc Recovery Menu	default	
Enabled	<input checked="" type="radio"/> TRUE <input type="radio"/> FALSE	
Default Action	None	
Default Resume Type	None	
Moveout Distance	10	
Pathrecorder Speed	800	
Pathrecorder Tool Offset	10	

Value (string)  
The changes will not take effect until the controller is restarted.

OK Cancel

xx2400000108

### Parameters

Parameter	Description	Data type
Name	The name of the instance ARC_ERR_HNDL	typeStringNormal

*Continues on next page*

## 6 Weld Error Recovery

### 6.4 Configuring Weld Error Recovery

*Continued*

Parameter	Description	Data type
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
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	<a href="#">RecoveryPosSet - Set the recovery position on page 157</a>
Reset the recovery position	<a href="#">RecoveryPosReset - Reset the recovery position on page 160</a>
Display the recovery menu	<a href="#">RecoveryMenu - Display the recovery menu on page 155</a>
Configure the recovery menu	<a href="#">Configure the recovery menu on page 77</a>

## 6.5 Configure the recovery menu

### Recovery menu

The Arc recovery menu allows the user to choose 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.

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

### Examples

The default configuration has the following definition.

*Continues on next page*

## 6 Weld Error Recovery

### 6.5 Configure the recovery menu

*Continued*

Name	Value	Information
Name	default	
Hide Resume At Error	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Hide Skip Forward	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Hide Skip Seam	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Hide Skip Part	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Hide Abort	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	

**Value (string)**  
The changes will not take effect until the controller is restarted.

OK Cancel

xx2400000113

### Parameters

Parameter	Description	Data type
Name	The name of the instance ARC_RECOVERY_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

## 6.6 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 75](#).

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*

## 6 Weld Error Recovery

### 6.6 Weld Error Recovery I/O interface

*Continued*

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.

*Continues on next page*



#### Dialog types

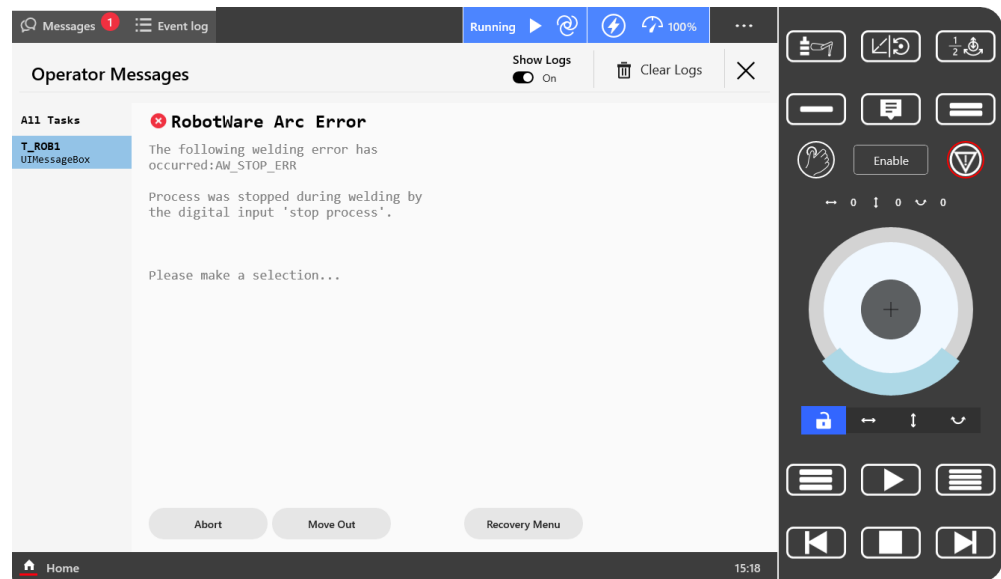
There are six possible dialog prompts from Weld Error Recovery. These are:

- 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.



xx2400000105

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

## 6 Weld Error Recovery

### 6.6 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

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
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
WeldOK supervision error	110431	31
Arc ignition supervision error	110432	32
Arc Fill supervision error	110433	33
User sig1 supervision error	110435	35

*Continues on next page*

Description	Arc ElogNumber	ErrNum
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*

## 6 Weld Error Recovery

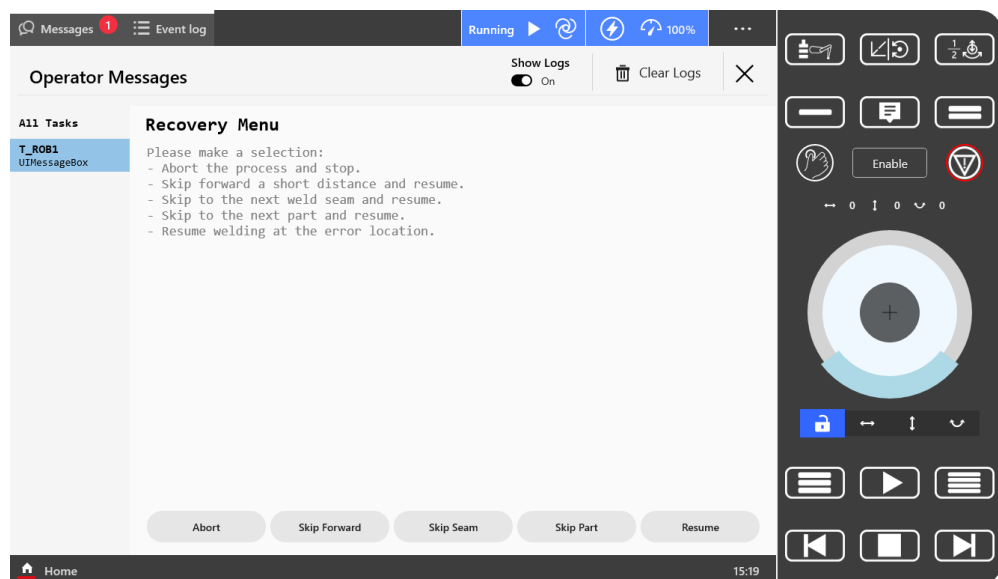
### 6.6 Weld Error Recovery I/O interface

*Continued*

#### 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.



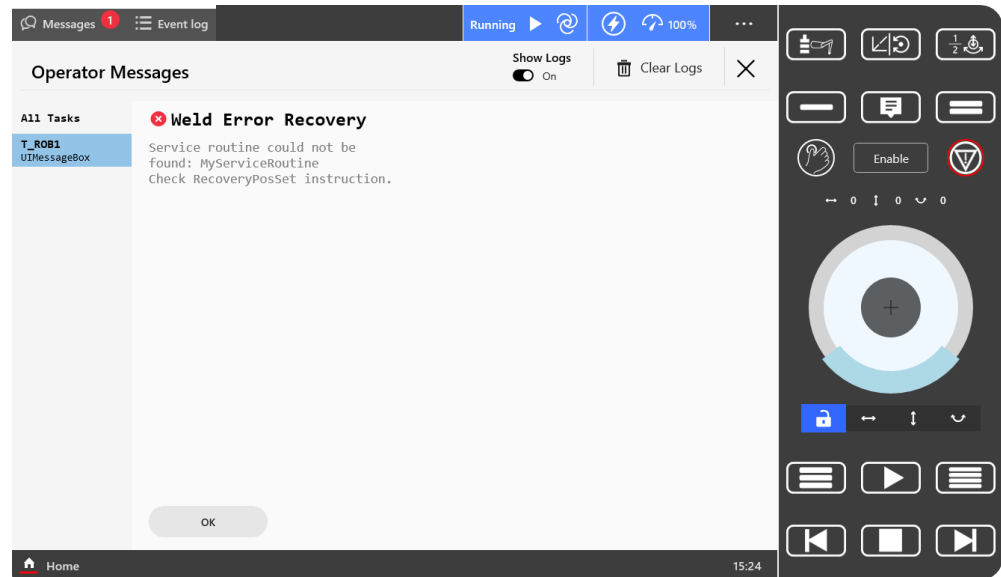
xx2400000106

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.

*Continues on next page*

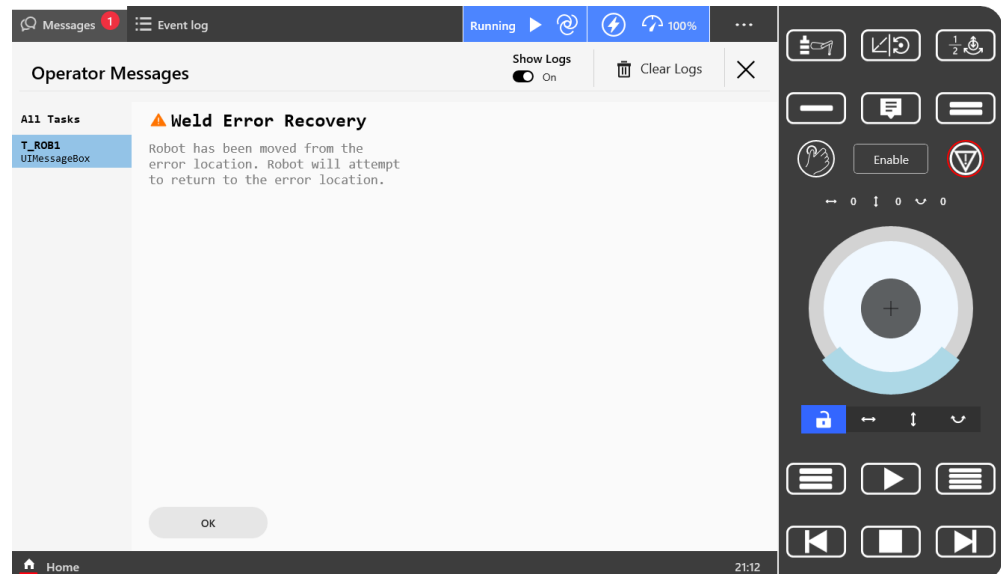


xx2400000110

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.



xx2400000114

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.

*Continues on next page*

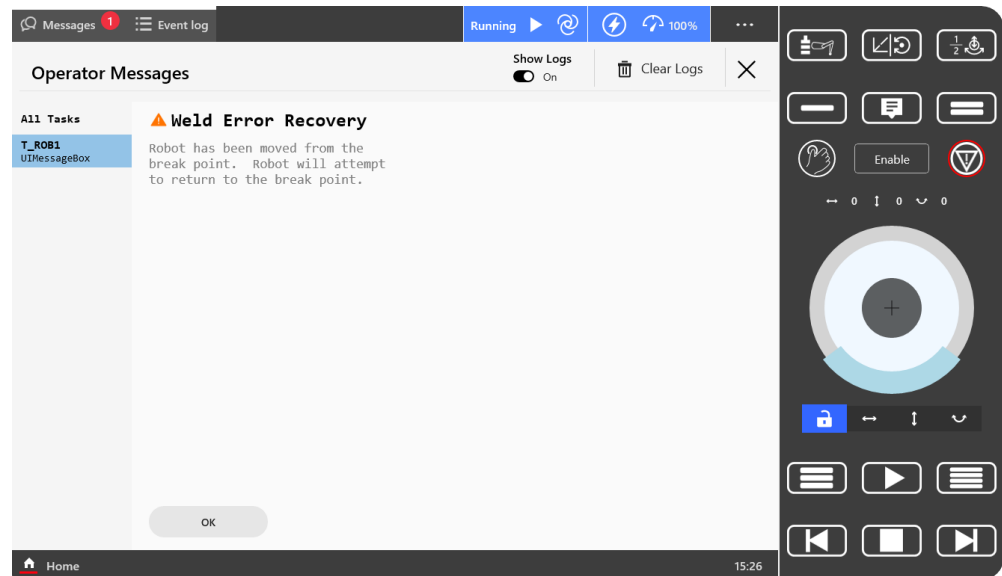
## 6 Weld Error Recovery

### 6.6 Weld Error Recovery I/O interface

*Continued*

#### Dialog type - Moved from Breakpoint

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

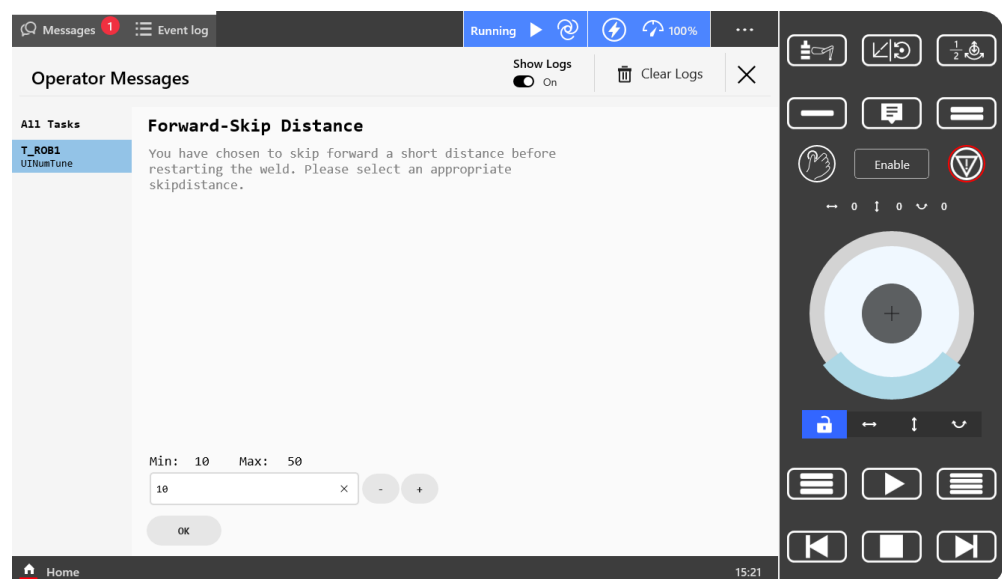


xx2400000111

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.



xx2400000109

*Continues on next page*

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, 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 [Weld Error Recovery flowchart on page 74](#).

## 6 Weld Error Recovery

### 6.7 Configure weld error recovery I/O Interface

### 6.7 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 ArcWare for OmniCore.

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.

The screenshot shows the 'Instance Editor' window with a table of configuration parameters. The table has three columns: Name, Value, and Information. The first row is for 'Name' with the value 'T\_ROB1'. The subsequent rows are for 'Active Dialog Type [GO]', 'Dialog Active [DO]', 'Escape Possible [DO]', 'Dialog Acknowledge [DI]', 'Response [GI]', 'Error Type [GO]', and 'Error Number [GO]', all with dropdown menus. Below the table, there is a section titled 'Value (string)' with a text box containing the message: 'The changes will not take effect until the controller is restarted.' At the bottom right, there are 'OK' and 'Cancel' buttons.

Name	Value	Information
Name	T_ROB1	
Active Dialog Type [GO]		
Dialog Active [DO]		
Escape Possible [DO]		
Dialog Acknowledge [DI]		
Response [GI]		
Error Type [GO]		
Error Number [GO]		

**Value (string)**  
The changes will not take effect until the controller is restarted.

OK Cancel

xx2400000115

Continues on next page



**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
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	<a href="#">RecoveryPosSet - Set the recovery position on page 157</a>
Reset the recovery position	<a href="#">RecoveryPosReset - Reset the recovery position on page 160</a>
Display the recovery menu	<a href="#">RecoveryMenu - Display the recovery menu on page 155</a>
Configure the recovery menu	<a href="#">Configure the recovery menu on page 77</a>

## 6 Weld Error Recovery

### 6.8 Configure User defined error handling

### 6.8 Configure User defined error handling

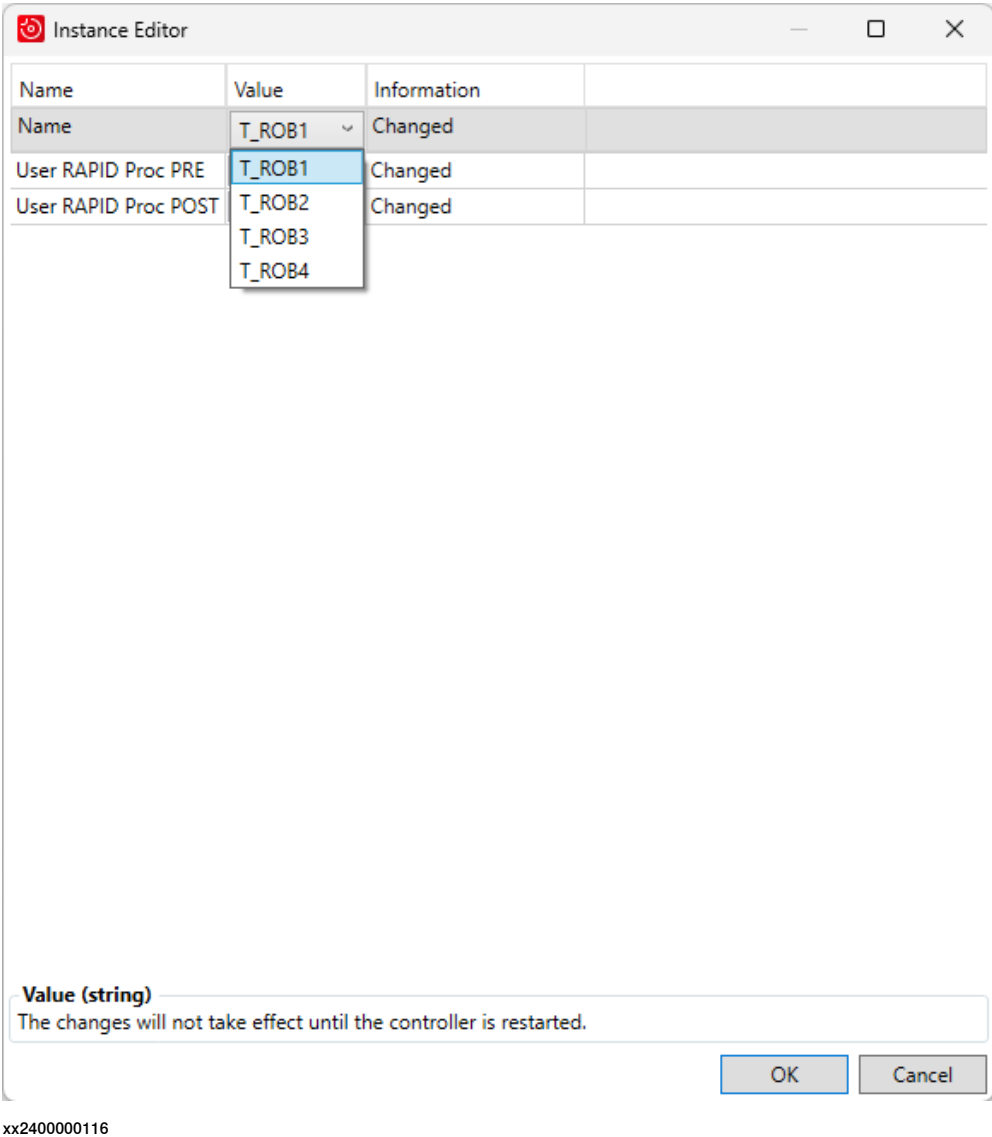
#### Description

*Arc Error Handler Properties* configures the Weld Error Recovery, user defined part of Weld Error Recovery feature in ArcWare for OmniCore.

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.



#### Parameters

Parameter	Description	Data Type
Name	The name of the instance ARC_ERR_HNDL_PROP. Must be (T_ROB1-T_ROB4)	typeStringNormal

*Continues on next page*

Parameter	Description	Data Type
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

#### Related information

Information	Described in
Set the recovery position	<a href="#">RecoveryPosSet - Set the recovery position on page 157</a>
Reset the recovery position	<a href="#">RecoveryPosReset - Reset the recovery position on page 160</a>
Display the recovery menu	<a href="#">RecoveryMenu - Display the recovery menu on page 155</a>
Configure the recovery menu	<a href="#">Configure the recovery menu on page 77</a>

## 6 Weld Error Recovery

### 6.9 User defined error handling

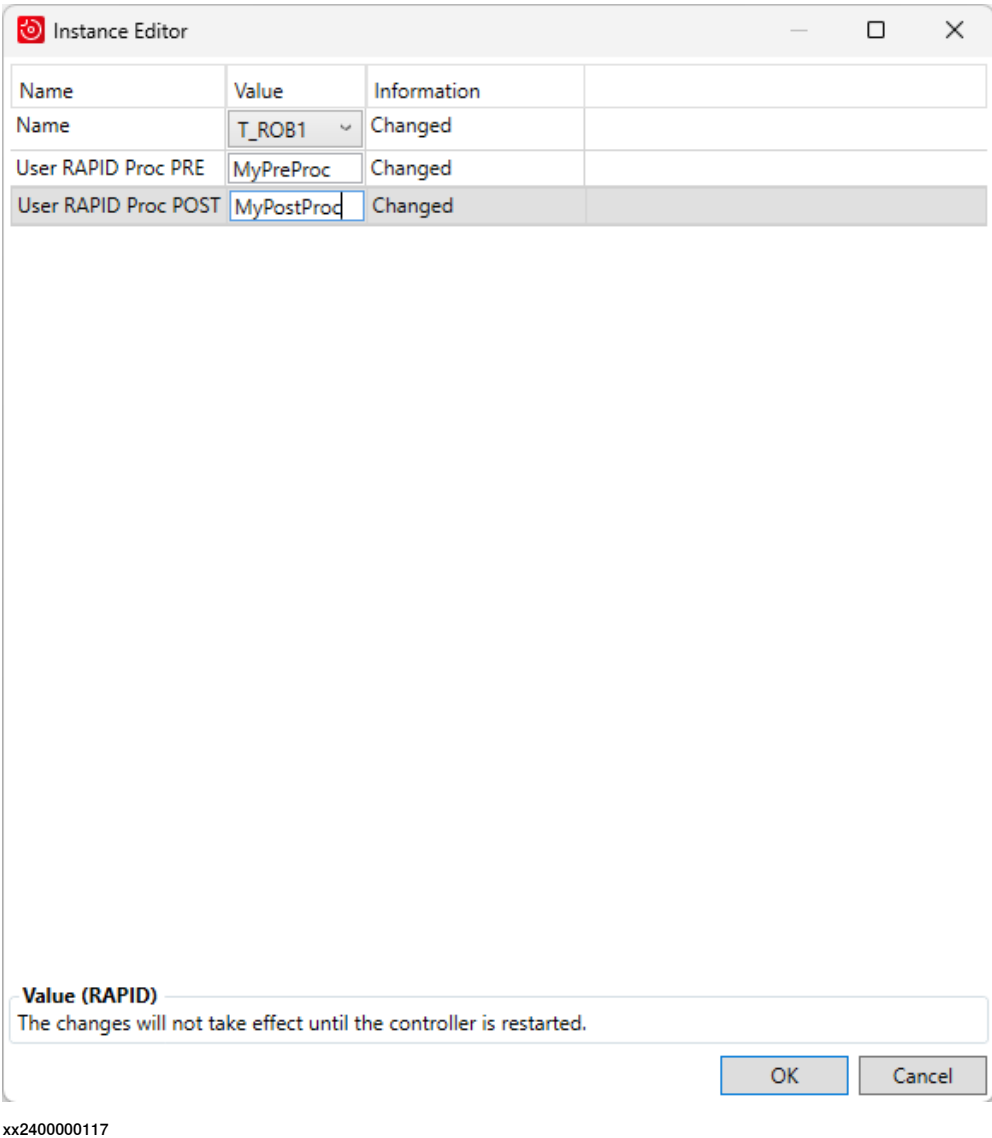
### 6.9 User defined error handling

#### 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`.



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`.

*Continues on next page*

The name of the current robtarget is retrieved via the RAPID string variable `stArcToPoint`.

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

#### Program example

```
MODULE ErrorHandler
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**

## 7 Function package for collaborative robots

### 7.1 Introduction

#### Introduction

*ArcWare for Collaborative robot CRB15000 (GoFa)* is a function package for arc welding with the CRB 15000 (GoFa) robot. It is distributed together with ArcWare for OmniCore, that is, in the same Add-In. The intent of this package is to lower the threshold to use a CRB 15000 for arc welding. As a consequence the functionality is limited compared to programming without this package. The RAPID interface, which is simplified and the *Wizard Easy Programming* interface is used for programming.

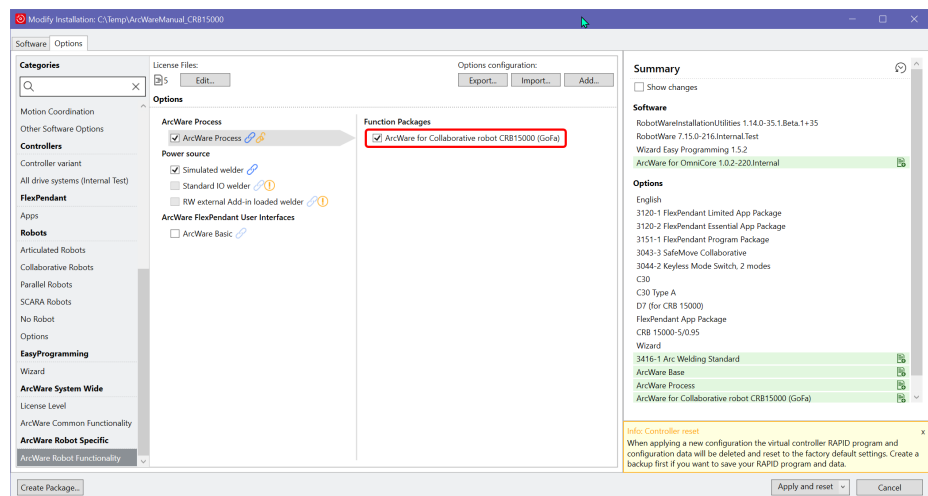
The supported RAPID interface is described in the RAPID reference for collaborative robots, see [Collaborative robots \(CRB 15000\) on page 184](#).

#### Prerequisites

The following software is required.

- *Arc for Collaborative robot CRB15000*

Has to be selected when the robot system is created.



xx2300002041

- *Wizard Easy Programming* version 1.3 or higher
- *ASI HMI* web app 1.0.5 or higher

#### Limitations

*ArcWare for Collaborative robot CRB15000 (GoFa)* can only be used with the collaborative robot CRB 15000.

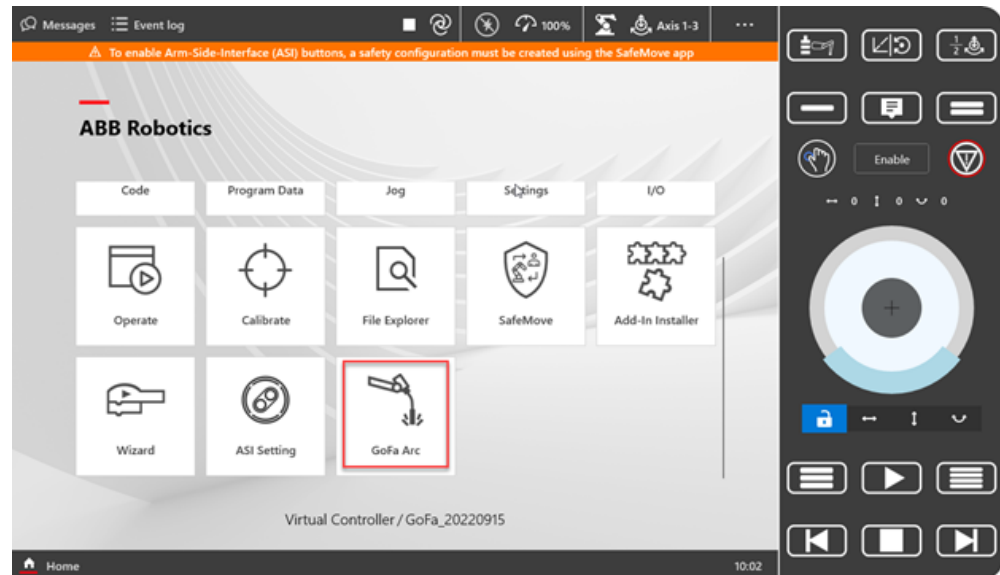
## 7 Function package for collaborative robots

### 7.2 FlexPendant application for ArcWare for Collaborative Robots

### 7.2 FlexPendant application for ArcWare for Collaborative Robots

#### FlexPendant application for ArcWare for Collaborative Robots

The ArcWare for Collaborative Robots add-in has its own application on the FlexPendant, **GoFa Arc**. It is available on the start page.



xx2200001361

#### Main page

The main page shows relevant information about program and process related topics. The following information is available:

- Active RAPID task
- Location of program pointer
- Active weld speed in selected units
- Active job number active in the power source
- **Manual Functions**
- **Manual Actions**
- **Configuration**
- **Info**

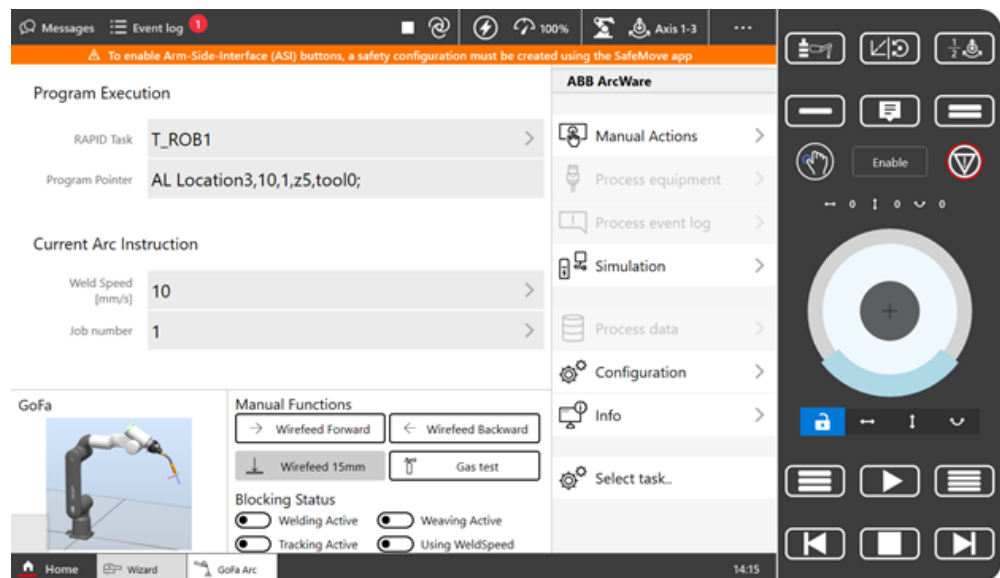
*Continues on next page*



## 7 Function package for collaborative robots

### 7.2 FlexPendant application for ArcWare for Collaborative Robots

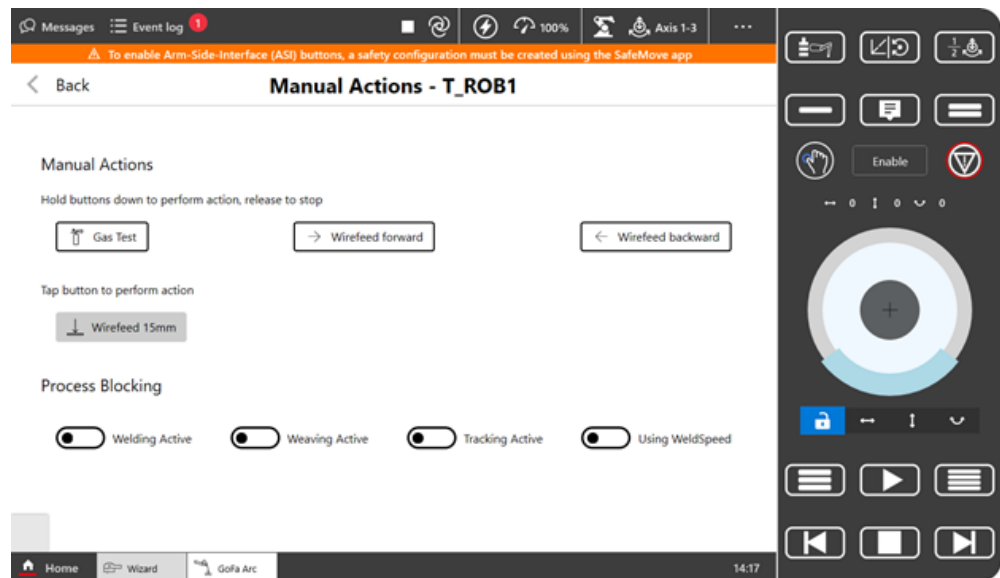
*Continued*



xx2200001362

#### Manual Actions

The **Manual Actions** view contains manual actions and process blocking. This functionality is also available on the main page.



xx2200001363

#### Configuration

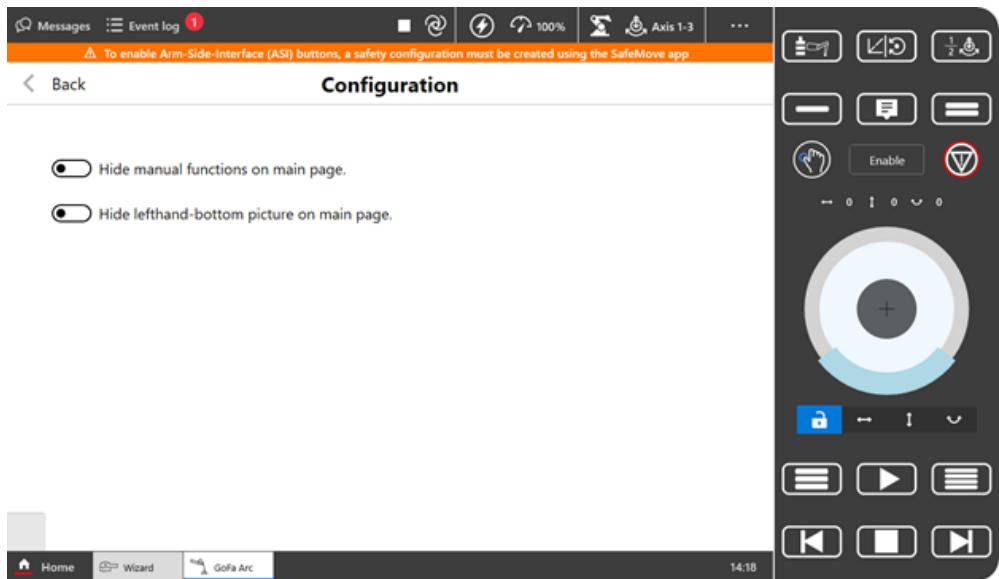
In the **Configuration** view, it is possible to hide manual functions on the main page, and also to hide the graphic shown bottom left on the main page.

*Continues on next page*

# 7 Function package for collaborative robots

## 7.2 FlexPendant application for ArcWare for Collaborative Robots

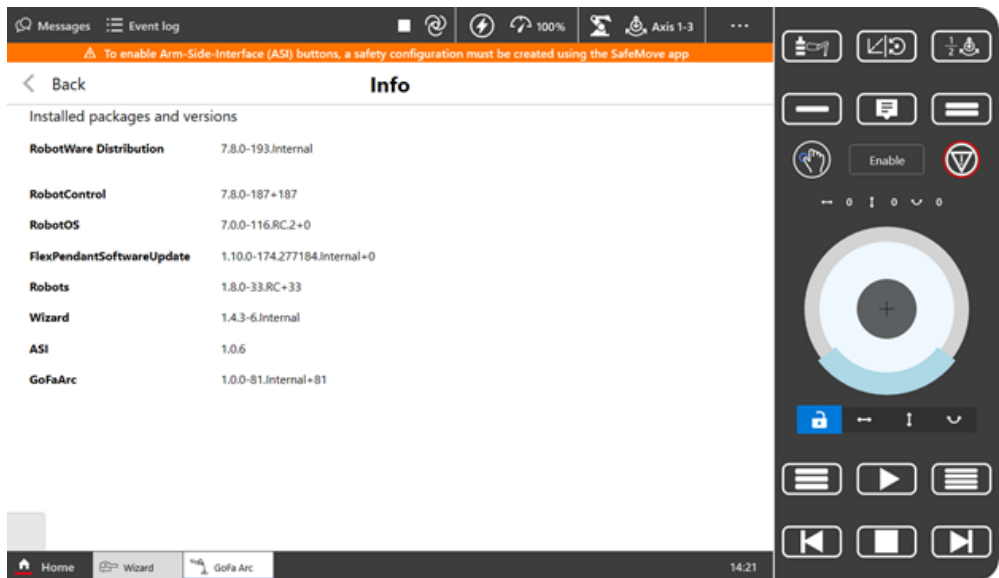
Continued



xx2200001364

### Info

The Info view shows the installed packages and versions.



xx2200001365

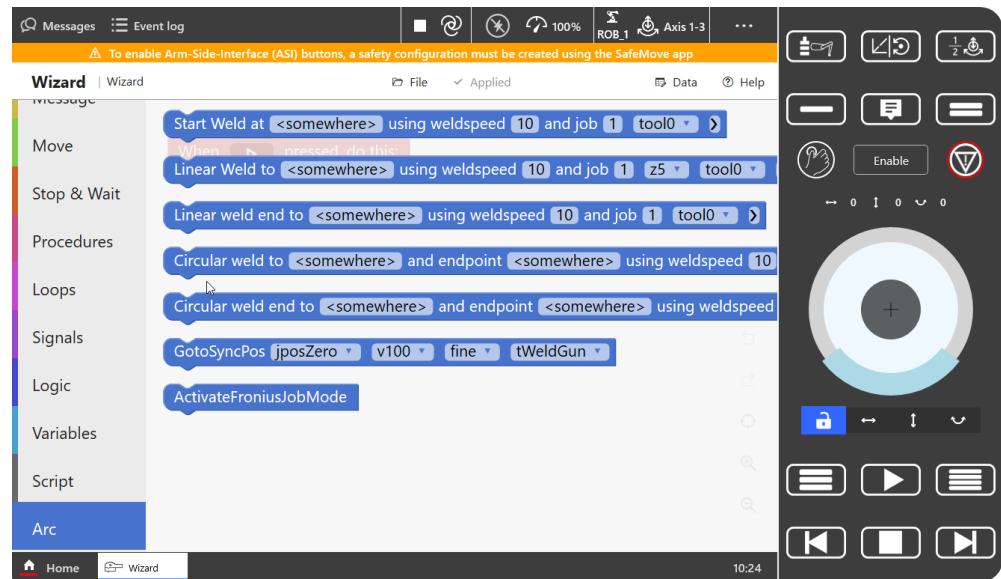
## 7.3 Wizard Easy Programming

### Introduction

The *Wizard Easy Programming* add-in is a graphical programming tool designed to get users up and running quickly. Simply drag and drop instruction blocks on the FlexPendant to create and modify programs. The Wizard add-in contains an integrated user manual.

The add-in contains instruction blocks for arc welding with CRB 15000. New categories and blocks can easily be created with the *ABB Skill Creator*.

The following blocks related to arc welding is available in the category **Arc**.



xx2200001366

There are two methods available for programming wizard blocks. Drag & drop, or programming via I/O signals.

When using the drag & drop method, the position, in this case named *<somewhere>* must be updated manually by adding a new location. If the I/O bind block method is used, the location is automatically added and updated. Weld speed and job can be edited directly in the block.

For more information about the available RAPID objects, see [RAPID reference on page 105](#).

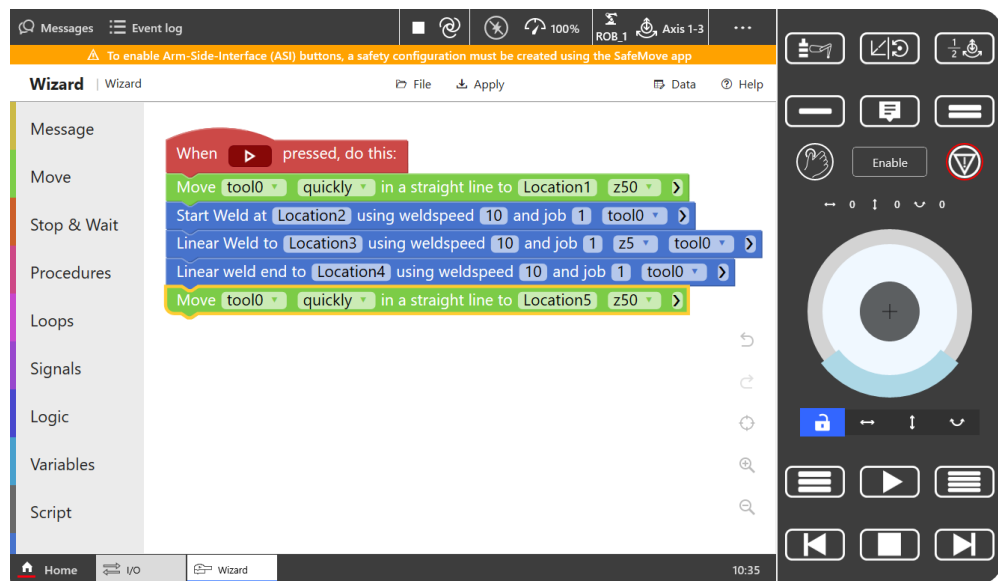
A finished program can look something like this.

*Continues on next page*

## 7 Function package for collaborative robots

### 7.3 Wizard Easy Programming


*Continued*



xx2200001367

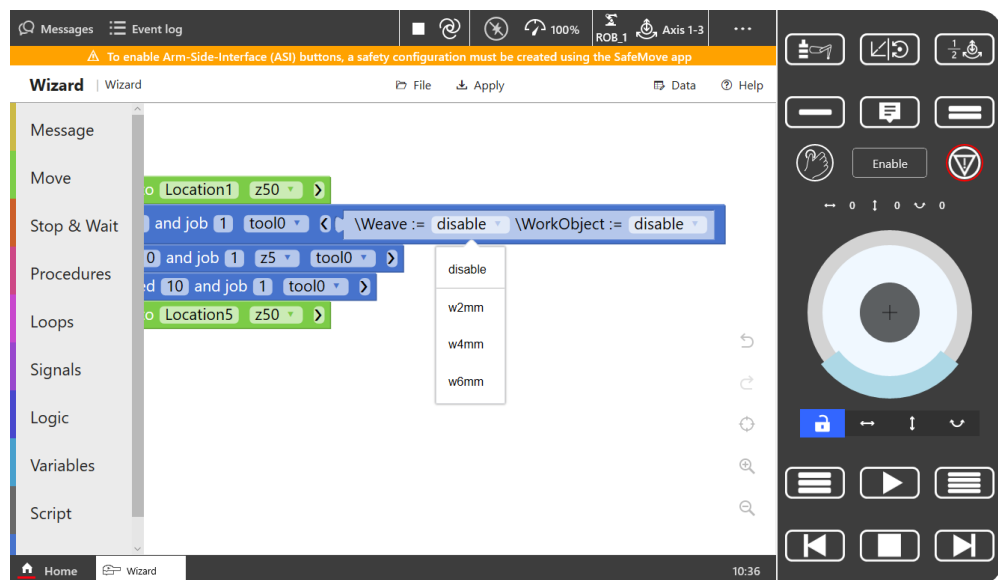
#### Optional parameters

There are optional parameters available in the Wizard blocks.

This is indicated with an arrow icon on the block .

#### Weave

A weaving pattern can be added to the RAPID instruction by using a pre-defined weave template in the drop-down menu, **w2mm**, **w4mm**, or **w6mm**. The option **disable** will remove the weave pattern from the RAPID instruction. Collapsing the arrow icon will also remove the weave pattern.

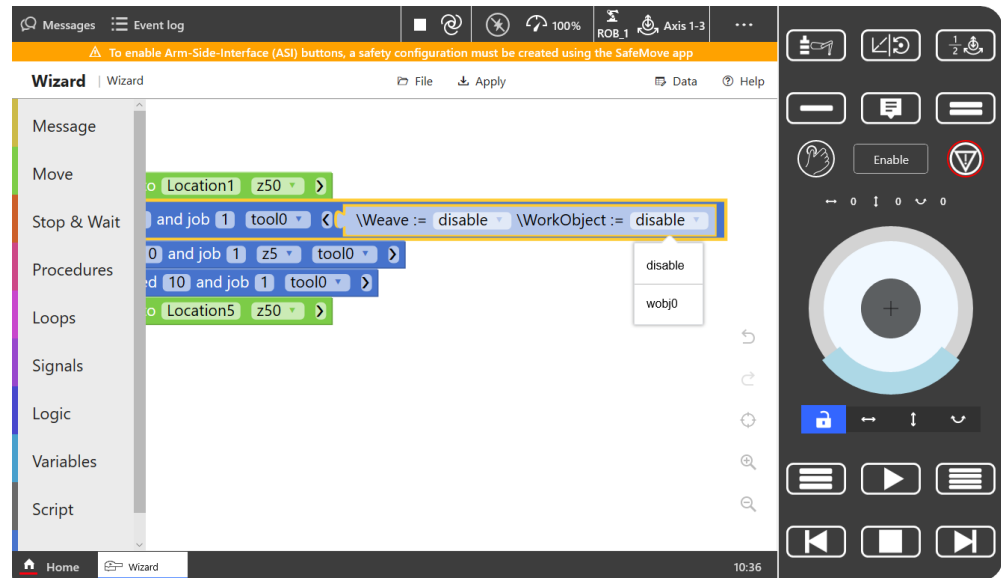


xx2200001428

*Continues on next page*

#### WorkObject

A workobject can be added using the drop-down menu, listing all workobjects that are available in the controller. The option **disable** will remove the workobject from the RAPID instruction. Collapsing the arrow icon will also remove the workobject.

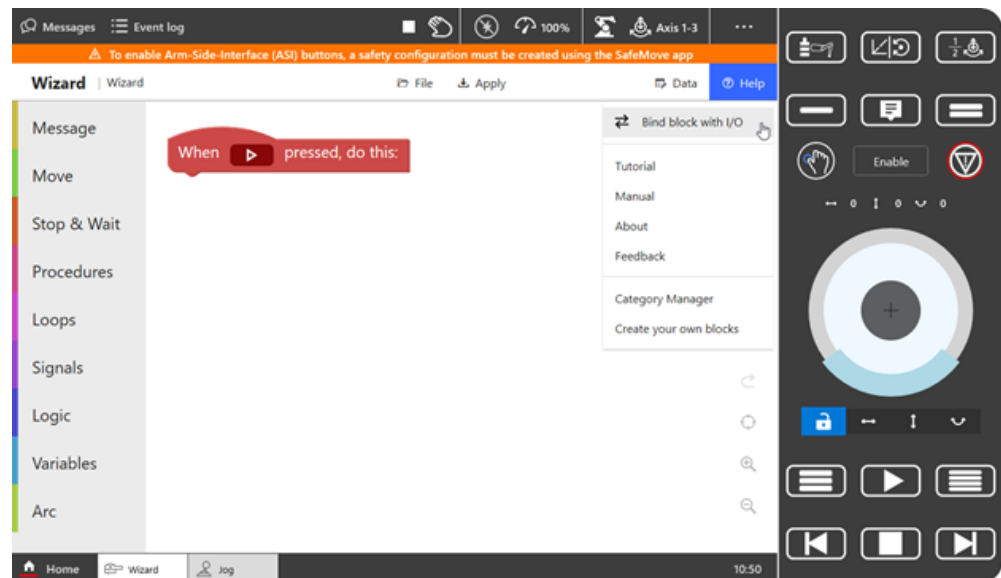


xx2200001429

#### Bind block with I/O

GoFa Arc comes pre-configured with block bind to I/O signals.

The configuration can be viewed in **Help** and then **Bind block with I/O**.



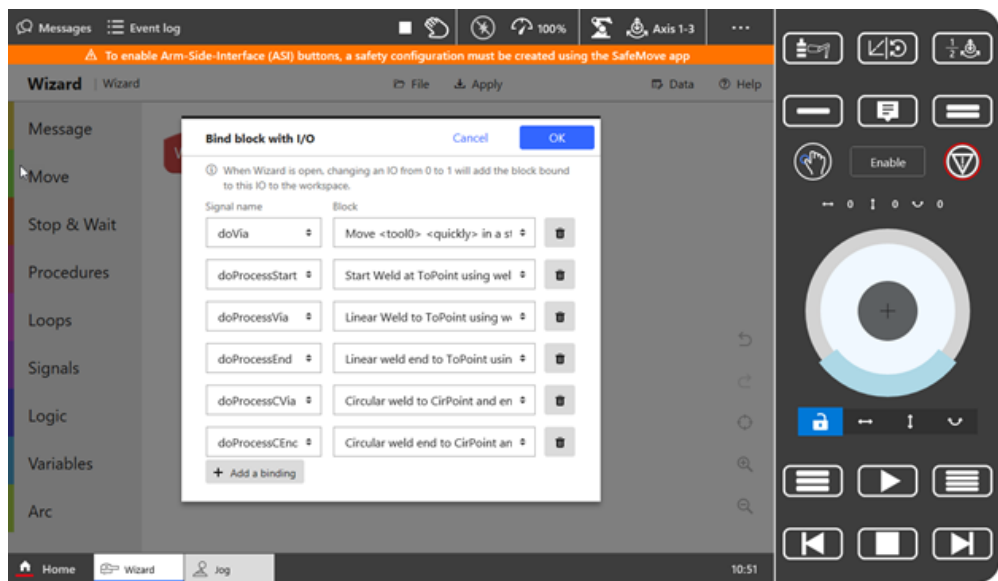
xx2200001368

*Continues on next page*

# 7 Function package for collaborative robots

## 7.3 Wizard Easy Programming

Continued



xx2200001369

The pre-configured I/O can be viewed in RobotStudio.

I/O System X   T_ROB1/Wizard										
Name	Type	Value	Min Value	Max Value	Simulated	Network	Device	Device Mapping	Category	Label
ACOK	DI	0	0	1	No	IntBus	DrvSys	3		
Auto	DO	1	0	1	No	IntBus	IoPanel	9		
AutomaticMode	DI	1	0	1	No	SC_Feedback_Net	SC_Feedback_Dev	0	SC_Feedback	
AutoModeStatus	DO	1	0	1	No	IntBus	EPanel	0		
AutoReqExt	DI	0	0	1	No	IntBus	IoPanel	2		
AutoReqTPU	DI	0	0	1	No	IntBus	IoPanel	5		
AXDCOK	DI	0	0	1	No	IntBus	DrvSys	5		
BrakeEn	DO	0	0	1	No	IntBus	DrvSys	6		
BrakeFb	DI	0	0	1	No	IntBus	DrvSys	0		
BrakeOk	DI	0	0	1	No	IntBus	DrvSys	1		
BrakeSupply	DI	0	0	1	No	IntBus	DrvSys	4		
doProcessCEnd	DO	0			Yes	<none>	<none>			
doProcessCVia	DO	0			Yes	<none>	<none>			
doProcessEnd	DO	0			Yes	<none>	<none>			
doProcessStart	DO	0			Yes	<none>	<none>			
doProcessVia	DO	0			Yes	<none>	<none>			
doVia	DO	0			Yes	<none>	<none>			
doWZ1	DO	0	0	1	Yes	<none>	<none>	104	internal	
doWZ2	DO	0	0	1	Yes	<none>	<none>	105	internal	
doWZ3	DO	0	0	1	Yes	<none>	<none>	106	internal	
doWZ4	DO	0	0	1	Yes	<none>	<none>	107	internal	

xx2200001370

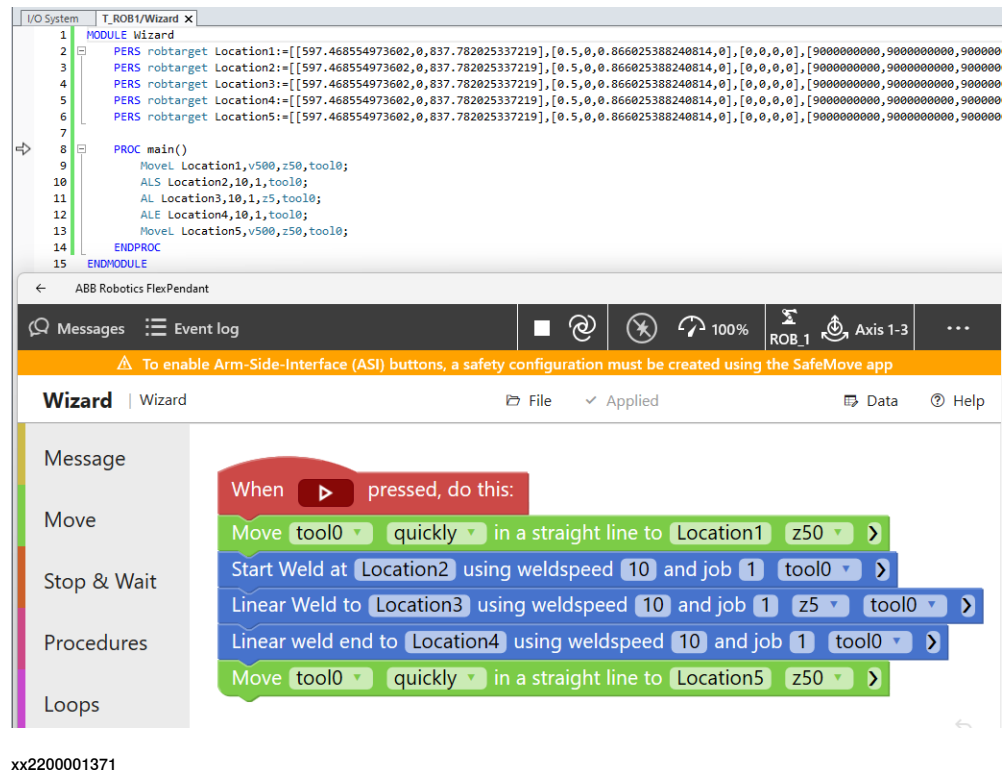
Toggling the I/O signals will create new entries in Wizard. Programming via I/O signals can be used with external devices, for example, setting the I/O signals via buttons for easy programming of welds.



### Note

The circular instructions require two toggles of the I/O signal. The first toggle will add the instruction and update the first position. The second toggle will update the second position.

Continues on next page



#### Advanced settings

If the predefined Wizard blocks are not suitable for the application, new Wizard blocks, and categories can be created with the ABB Skill Creator.

**This page is intentionally left blank**



## 8 RAPID reference

### 8.1 Standard

#### 8.1.1 Instructions

##### 8.1.1.1 ArcLStart - 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.

If a weld seam is programmed without an `ArcLStart` instruction, 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.

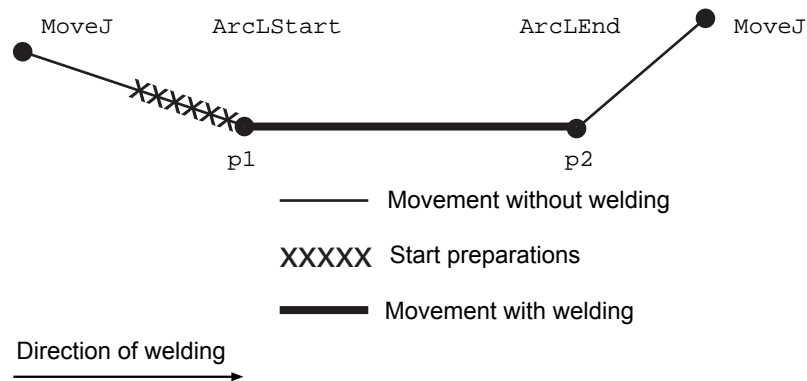
*Continues on next page*

## 8 RAPID reference

### 8.1.1.1 ArcLStart - Arc welding start with linear motion

ArcWare

Continued



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 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]
[\ArcSystem]
```

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:

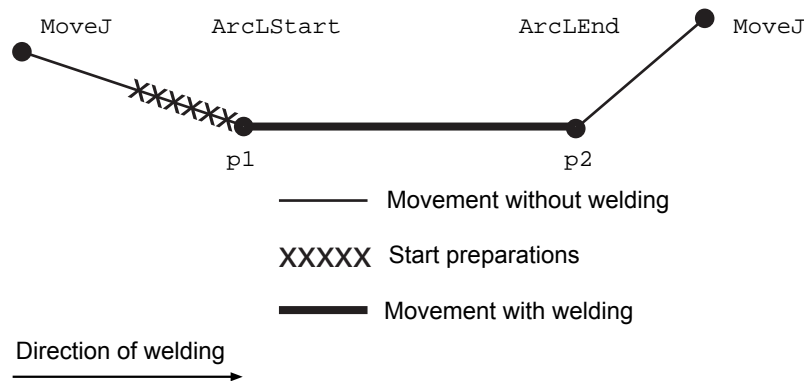
- When the ArcLStart instruction is used.
- When the program is run instruction-by-instruction (no welding).

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.

Continues on next page

## 8.1.1.1 ArcLStart - Arc welding start with linear motion

*ArcWare  
Continued*



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 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.

*Continues on next page*

## 8 RAPID reference

---

### 8.1.1.1 ArcLStart - Arc welding start with linear motion

ArcWare

Continued

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.



#### 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.

*Continues on next page*

For more information see *Application manual - Tracking 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

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.

[ \ArcSystem ]

**Data type:** num

This parameter has allowed values 1, 2, and 3. It defines, which Arc Weld system the instruction is connected to. All ArcL and ArcC instructions between ArcLStart and including ArcLEnd or ArcCEnd will be connected to that Arc Weld system . Up to three different Arc Weld systems can be configured in the system parameters. If this parameter is not specified, the arc welding system 1 is connected. For more information see [Defining arc welding systems on page 16](#).

---

## 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.

*Continues on next page*

## 8 RAPID reference

---

### 8.1.1.1 ArcLStart - Arc welding start with linear motion

ArcWare

Continued

#### 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.

#### 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 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.

Continues on next page

## 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 16](#)):

- 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.

**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 `AW_WIRE_ERR` error in any of the synchronized robots.

## Example

```
MoveL ...
```

*Continues on next page*

## 8 RAPID reference

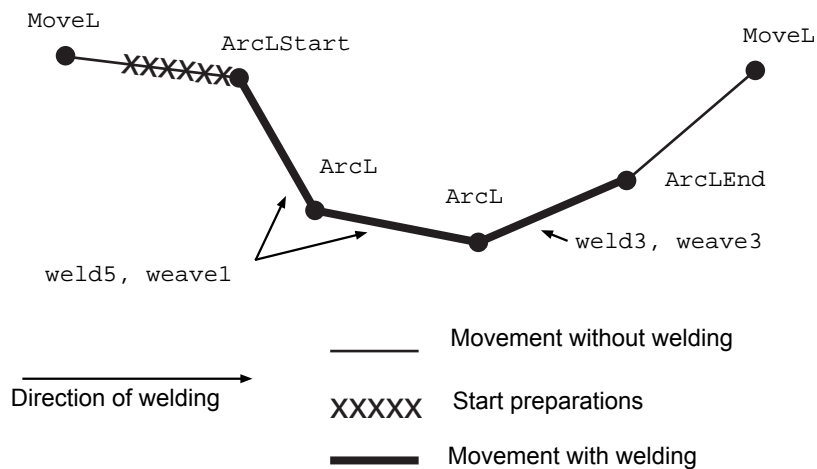
### 8.1.1.1 ArcLStart - Arc welding start with linear motion

ArcWare

Continued

```
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 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.



#### Note

The optional argument \PreProcessTracking in ArcLStart cannot be used for robots with coordinated movement.

### Syntax

```
ArcLStart  
[ToPoint ':=' ] <expression (IN) of robtarg>  
[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>','
```

Continues on next page



## 8.1.1.1 ArcLStart - Arc welding start with linear motion

ArcWare

Continued

```

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

## Related information

Information	Described in
Performing a linear weld	<a href="#">ArcL - Arc welding with linear motion on page 115</a>
Other positioning instructions	<i>Technical reference manual - RAPID Overview</i>
Definition of speed, speeddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of zone data, zonedata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of tools, tooldata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of work objects, wobjdata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
MoveL	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of loads, loaddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of flying start data	<a href="#">flystartdata - Flying start data on page 165</a>
Definition of seam data	<a href="#">seamdata - Seam data on page 166</a>
Definition of weld data	<a href="#">welddata - Weld data on page 180</a>
Definition of weave data	<a href="#">weavedata - Weave data on page 173</a>
Installation parameters for welding equipment and welding functions	<a href="#">System parameters on page 15</a>
Movements in general	<i>Technical reference manual - RAPID Overview</i>
Coordinate systems	<i>Technical reference manual - RAPID Overview</i>
Process phases and sub-activities	<a href="#">Programming on page 45</a>

Continues on next page

# 8 RAPID reference

---

## 8.1.1.1 ArcLStart - Arc welding start with linear motion

ArcWare

Continued

Information	Described in
<i>Path Corrections</i>	<i>Application manual - Controller software OmniCore</i>

### 8.1.1.2 ArcL - 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:

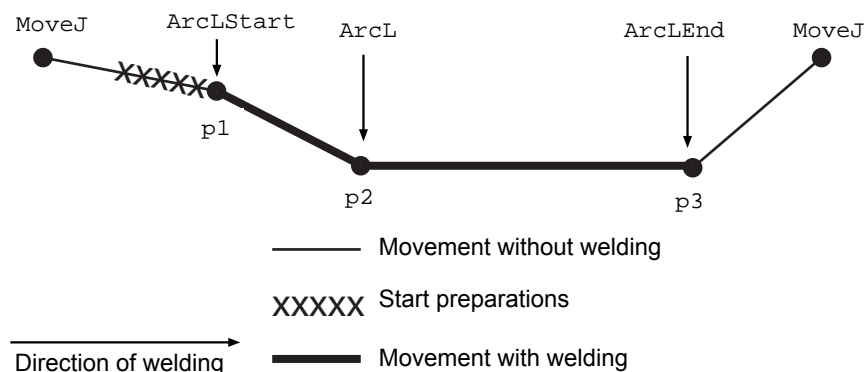
- The tool center point (TCP) is moved linearly to the specified destination position.
- All phases of the welding process, such as the start and end phases, are controlled.
- The welding process is monitored continuously.

If a weld seam is programmed without an ArcLStart instruction, 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.

#### 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

*Continues on next page*

## 8 RAPID reference

### 8.1.1.2 ArcL - Arc welding with linear motion

ArcWare

Continued

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.

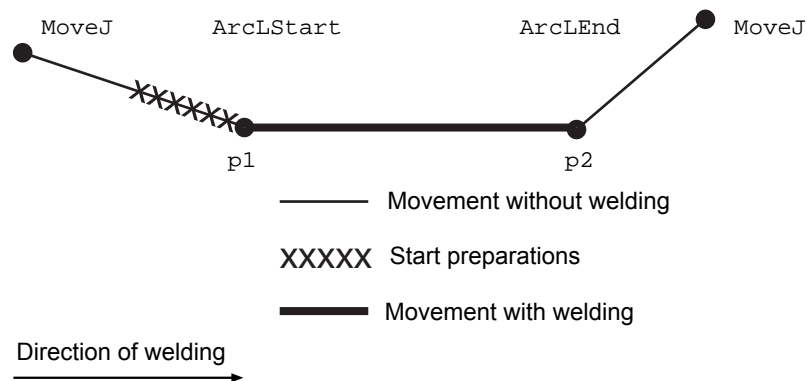
Speed

**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.



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.

Continues on next page

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 Corrections* 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

*Continues on next page*

## 8 RAPID reference

---

### 8.1.1.2 ArcL - Arc welding with linear motion

ArcWare

Continued

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.



#### Note

Seam tracking requires the *Optical tracking* or *WeldGuide* options.

[`\TrackOffsetFrame`]

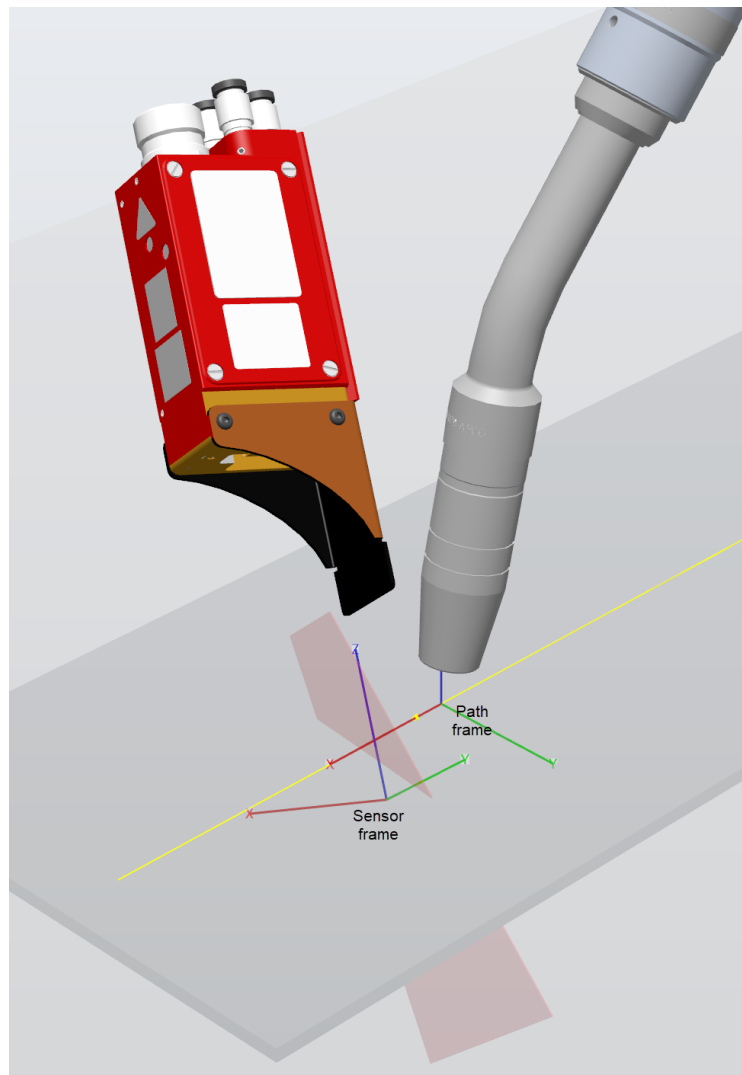
Data type: `captrackoffsframe`

This optional argument is used to select the frame in which the optical tracking offset (`seamoffs_y` and `seamoffs_z`) is applied.

The following predefined values are available:

Value	Description
<code>CAP_OFFSET_FRAME_SENSOR</code>	The sensor measurement coordinate system. This is the default value, if this optional argument is not present.
<code>CAP_OFFSET_FRAME_PATH</code>	The path coordinate system.

Continues on next page



xx2400000789

[ \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:** trigdata

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.

*Continues on next page*

## 8 RAPID reference

---

### 8.1.1.2 ArcL - Arc welding with linear motion

ArcWare

Continued

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

*Continues on next page*



Error constant (ERRNO value)	Description
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.



#### 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 `AW_WIRE_ERR` 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.

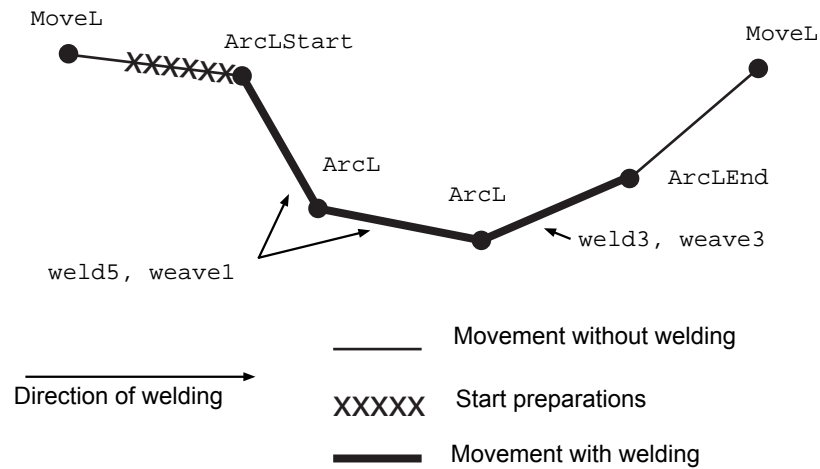
*Continues on next page*

## 8 RAPID reference

### 8.1.1.2 ArcL - Arc welding with linear motion

ArcWare

Continued



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

ArcL 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 robtarg>
[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>]
['\ ' TrackOffsetFrame ':=' <expression (IN) of captrackoffsframe
> ]
['\ ' 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>]';'
```

Continues on next page

## Related information

Information	Described in
Performing a circular motion weld	<a href="#">ArcC - Arc welding with circular motion on page 133</a>
Other positioning instructions	<i>Technical reference manual - RAPID Overview</i>
Definition of speed, speeddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of zone data, zonedata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of tools, tooldata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of work objects, wobjdata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
MoveL	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of loads, loaddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of seam data	<a href="#">seamdata - Seam data on page 166</a>
Definition of weld data	<a href="#">welddata - Weld data on page 180</a>
Definition of weave data	<a href="#">weavedata - Weave data on page 173</a>
Installation parameters for welding equipment and welding functions	<a href="#">System parameters on page 15</a>
Movements in general	<i>Technical reference manual - RAPID Overview</i>
Coordinate systems	<i>Technical reference manual - RAPID Overview</i>
Process phases and sub-activities	<a href="#">Programming on page 45</a>
<i>Path Corrections</i>	<i>Application manual - Controller software OmniCore</i>

## 8 RAPID reference

### 8.1.1.3 ArcLEnd - Arc welding end with linear motion

ArcWare

### 8.1.1.3 ArcLEnd - 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.

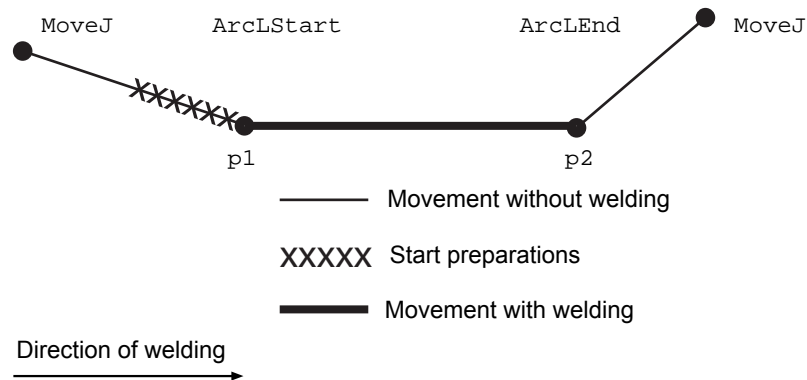
When the instruction `ArcLEnd` is used, welding ends when the robot reaches the destination position. Regardless of what is specified in the `Zone` argument, the destination position will be a stop point (fine).

If a weld seam is programmed without an `ArcLStart` instruction, 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 and end at p2. The start and end processes are determined by `seam1` and the welding process by `weld5`.

*Continues on next page*

**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.

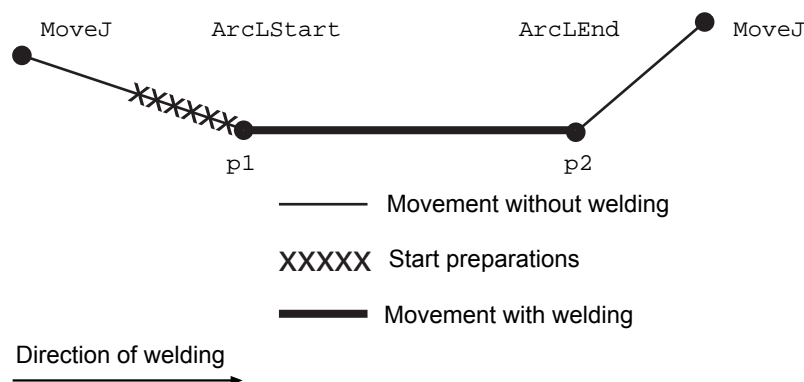
Speed

**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*Continues on next page*

## 8 RAPID reference

---

### 8.1.1.3 ArcLEnd - Arc welding end with linear motion

ArcWare

Continued

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`

*Continues on next page*

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.



#### Note

Seam tracking requires the *Optical tracking* or *WeldGuide* options.

[`\TrackOffsetFrame`]

**Data type:** `captrackoffsframe`

This optional argument is used to select the frame in which the optical tracking offset (`seamoffs_y` and `seamoffs_z`) is applied.

The following predefined values are available:

Value	Description
<code>CAP_OFFSET_FRAME_SENSOR</code>	The sensor measurement coordinate system. This is the default value, if this optional argument is not present.
<code>CAP_OFFSET_FRAME_PATH</code>	The path coordinate system.

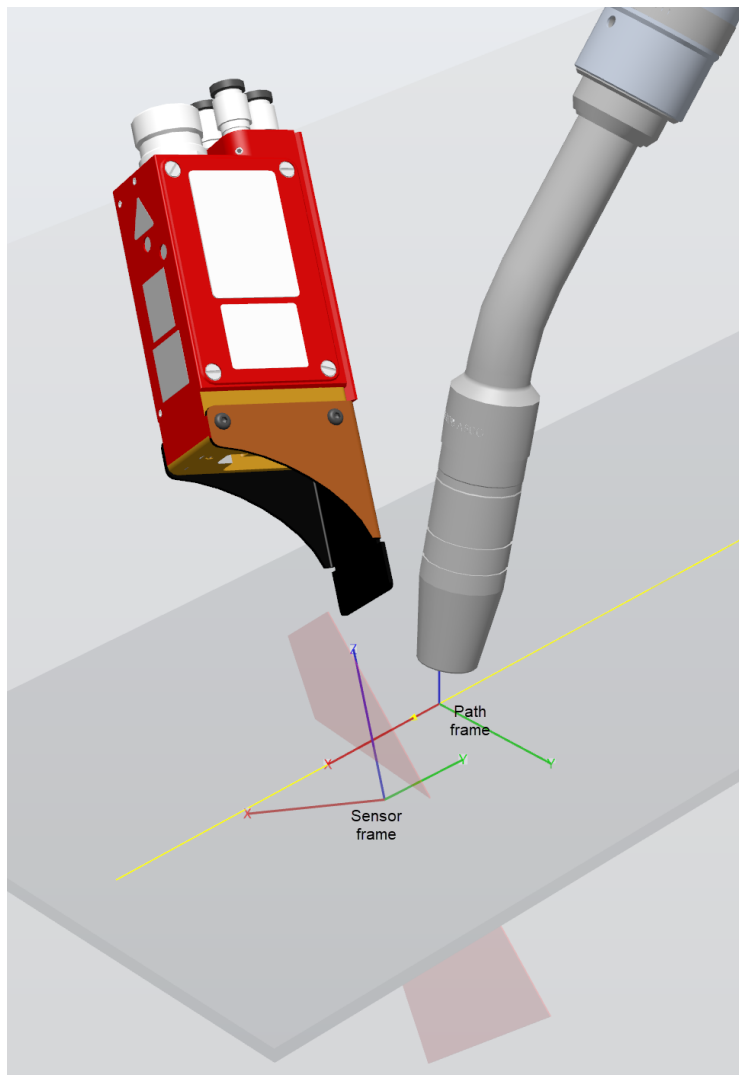
*Continues on next page*

## 8 RAPID reference

### 8.1.1.3 ArcLEnd - Arc welding end with linear motion

ArcWare

Continued



xx2400000789

[ \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:** trigdata

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.

*Continues on next page*



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

*Continues on next page*

## 8 RAPID reference

### 8.1.1.3 ArcLEnd - Arc welding end with linear motion

ArcWare

Continued

Error constant (ERRNO value)	Description
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 16](#)):

- 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.



#### 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 AW\_WIRE\_ERR error in any of the synchronized robots.

#### Example

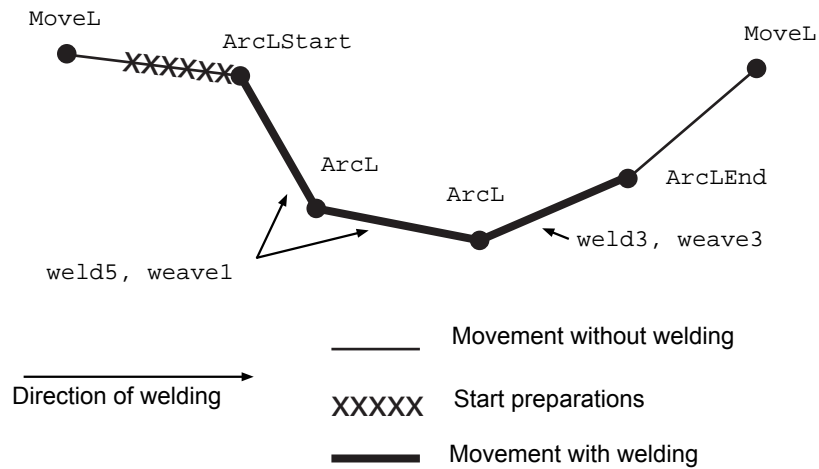
```
MoveL ...
ArcLStart *, v100, seam1, weld5 \Weave:=weave1, fine,
gun1\Wobj:=wobj1;
ArcL *, v100, seam1, weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcL *, v100, seam1, weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcLEnd *, v100, seam1, weld3 \Weave:=weave3, fine,
gun1\Wobj:=wobj1;
MoveL ...
```

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.

Continues on next page

## 8.1.1.3 ArcLEnd - Arc welding end with linear motion

ArcWare  
Continued



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.

## Syntax

```
ArcLEnd
[ToPoint ':='] <expression (IN) of robtarg>
[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>]
['\ ' TrackOffsetFrame ':=' <expression (IN) of captrackoffsframe
> ]
['\ ' Time ':=' <expression (IN) of num>]
['\ ' T1 ':=' <variable (VAR) of trigdata>]
['\ ' T2 ':=' <variable (VAR) of trigdata>]
['\ ' T3 ':=' <variable (VAR) of trigdata>]
['\ ' T4 ':=' <variable (VAR) of trigdata>]
['\ ' T5 ':=' <variable (VAR) of trigdata>]
['\ ' T6 ':=' <variable (VAR) of trigdata>]
['\ ' T7 ':=' <variable (VAR) of trigdata>]
['\ ' TLoad ':=' <persistent (PERS) of loaddata>]';'
```

## Related information

Information	Described in
Performing a linear weld	<a href="#">ArcL - Arc welding with linear motion on page 115</a>
Other positioning instructions	<i>Technical reference manual - RAPID Overview</i>

Continues on next page

## 8 RAPID reference

### 8.1.1.3 ArcLEnd - Arc welding end with linear motion

ArcWare

Continued

Information	Described in
Definition of speed, speeddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of zone data, zonedata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of tools, tooldata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of work objects, wobjdata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
MoveL	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of loads, loaddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of seam data	<a href="#">seamdata - Seam data on page 166</a>
Definition of weld data	<a href="#">welddata - Weld data on page 180</a>
Definition of weave data	<a href="#">weavedata - Weave data on page 173</a>
Installation parameters for welding equipment and welding functions	<a href="#">System parameters on page 15</a>
Movements in general	<i>Technical reference manual - RAPID Overview</i>
Coordinate systems	<i>Technical reference manual - RAPID Overview</i>
Process phases and sub-activities	<a href="#">Programming on page 45</a>
<i>Path Corrections</i>	<i>Application manual - Controller software OmniCore</i>

### 8.1.1.4 ArcC - 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:

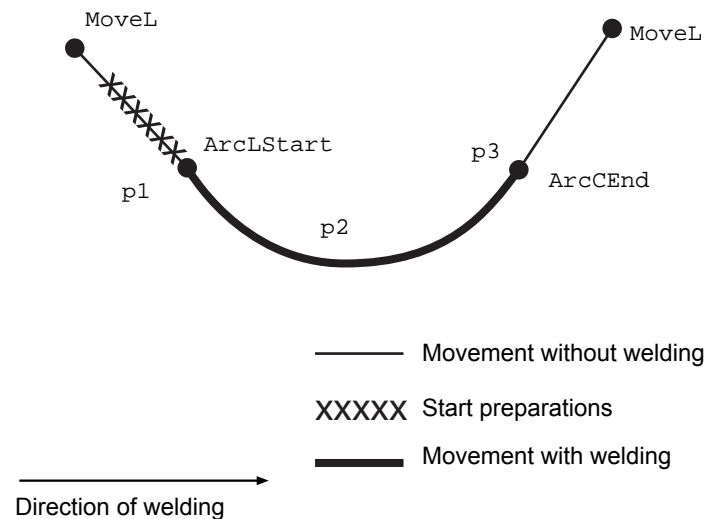
- The tool center point (TCP) is moved in a circle to the specified destination position.
- All phases of the welding process, such as the start and end phases, are controlled.
- The welding process is monitored continuously.

If a weld seam is programmed without an `ArcLStart` instruction, 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.



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

*Continues on next page*

## 8 RAPID reference

---

### 8.1.1.4 ArcC - Arc welding with circular motion

ArcWare

Continued

---

#### 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

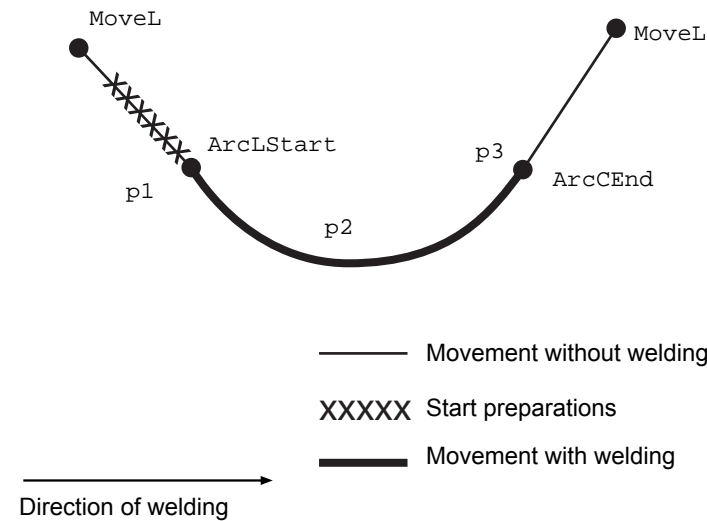
**Data type:** speeddata

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).

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.

*Continues on next page*



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

Continues on next page

## 8 RAPID reference

---

### 8.1.1.4 ArcC - Arc welding with circular motion

*ArcWare*

*Continued*

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 Corrections* 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.



#### Note

Seam tracking requires one of the following options: or options.

- *Tracking Interface*
- *WeldGuide*

[ `\TrackOffsetFrame` ]

**Data type:** `captrackoffsframe`

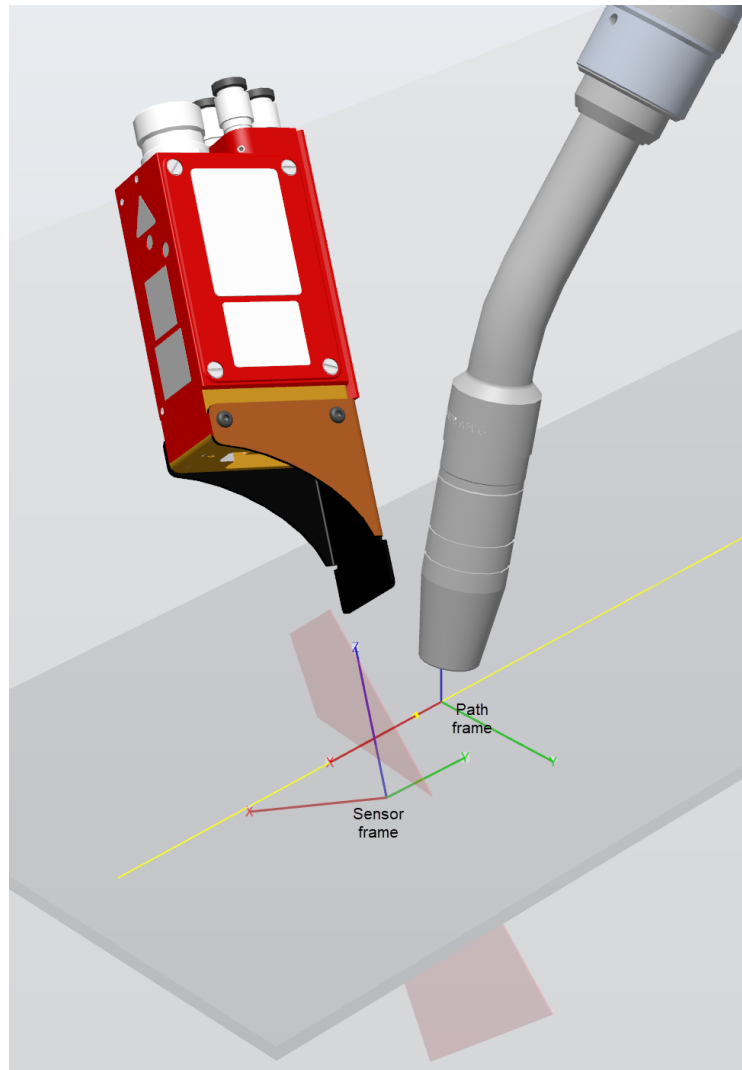
This optional argument is used to select the frame in which the optical tracking offset (`seamoffs_y` and `seamoffs_z`) is applied.

*Continues on next page*



The following predefined values are available:

Value	Description
CAP_OFFSET_FRAME_SENSOR	The sensor measurement coordinate system. This is the default value, if this optional argument is not present.
CAP_OFFSET_FRAME_PATH	The path coordinate system.



xx2400000789

[ \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

*Continues on next page*

## 8 RAPID reference

---

### 8.1.1.4 ArcC - Arc welding with circular motion

ArcWare

Continued

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).

*Continues on next page*

## 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 16](#)):

- 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.

**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 `AW_WIRE_ERR` error in any of the synchronized robots.

## Example

```
MoveL ...
```

*Continues on next page*

## 8 RAPID reference

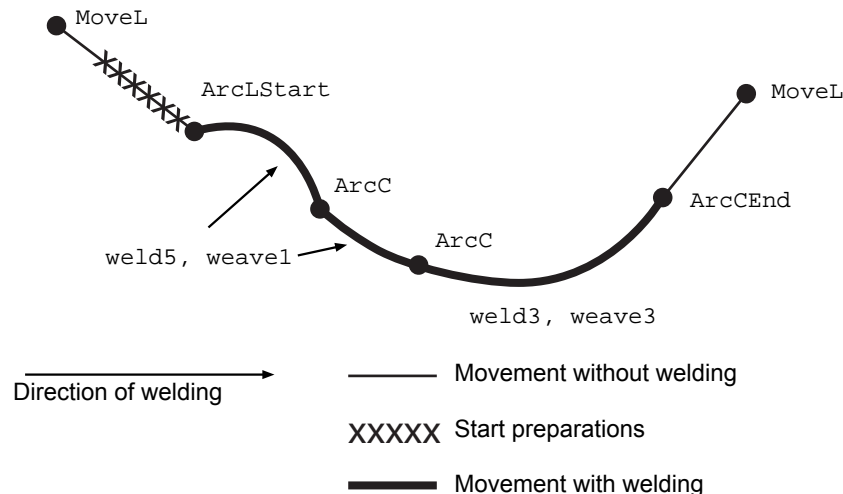
### 8.1.1.4 ArcC - Arc welding with circular motion

ArcWare

Continued

```
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

**ArcC 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 robtarg>
[ToPoint ':='] <expression (IN) of robtarg>
[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>]
['\ ' TrackOffsetFrame ':=' < expression (IN) of captrackoffsframe
> ]
['\ ' Time ':=' <expression (IN) of num>]
```

Continues on next page

```
[ '\ ' 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	<a href="#">ArcL - Arc welding with linear motion on page 115</a>
Other positioning instructions	<i>Technical reference manual - RAPID Overview</i>
Definition of speed, speeddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of zone data, zonedata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of tools, tooldata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of work objects, wobjdata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
MoveL	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of loads, loaddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of seam data	<a href="#">seamdata - Seam data on page 166</a>
Definition of weld data	<a href="#">welddata - Weld data on page 180</a>
Definition of weave data	<a href="#">weavedata - Weave data on page 173</a>
Installation parameters for welding equipment and welding functions	<a href="#">System parameters on page 15</a>
Movements in general	<i>Technical reference manual - RAPID Overview</i>
Coordinate systems	<i>Technical reference manual - RAPID Overview</i>
Process phases and sub-activities	<a href="#">Programming on page 45</a>
<i>Path Corrections</i>	<i>Application manual - Controller software OmniCore</i>

## 8 RAPID reference

### 8.1.1.5 ArcCEnd - Arc welding end with circular motion

ArcWare

#### 8.1.1.5 ArcCEnd - 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.

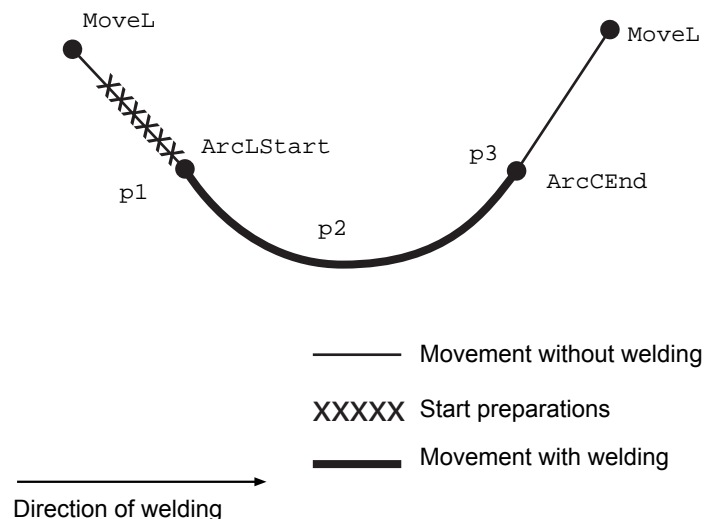
When the instruction `ArcCEnd` is used, welding ends when the robot reaches the destination position. Regardless of what is specified in the `Zone` argument, the destination position will be a stop point (fine).

If a weld seam is programmed without an `ArcLStart` instruction, 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 ...
```

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`.

*Continues on next page*

## 8.1.1.5 ArcCEnd - Arc welding end with circular motion

ArcWare

Continued

**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

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 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.

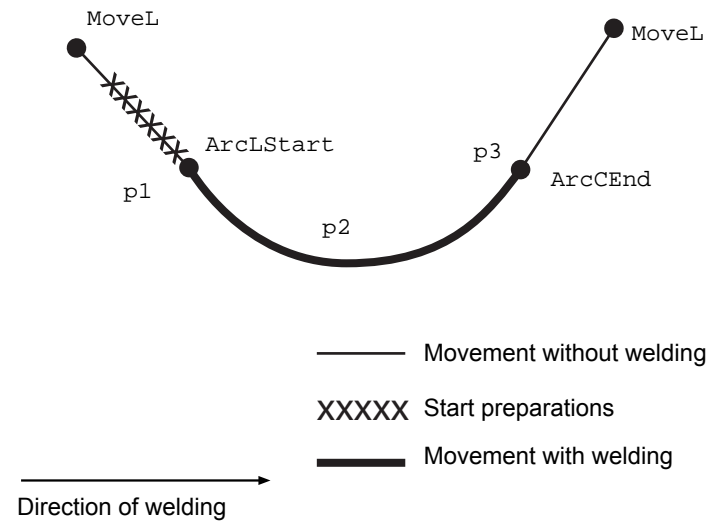
*Continues on next page*

## 8 RAPID reference

### 8.1.1.5 ArcCEnd - Arc welding end with circular motion

*ArcWare*

*Continued*



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.

*Continues on next page*



## 8.1.1.5 ArcCEnd - Arc welding end with circular motion

ArcWare

Continued

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.

**Note**

Seam tracking requires one of the following options: or options.

- Tracking Interface
- WeldGuide

[\TrackOffsetFrame]

**Data type:** captrackoffsetsframe

This optional argument is used to select the frame in which the optical tracking offset (`seamoffs_y` and `seamoffs_z`) is applied.

Continues on next page

8 RAPID reference

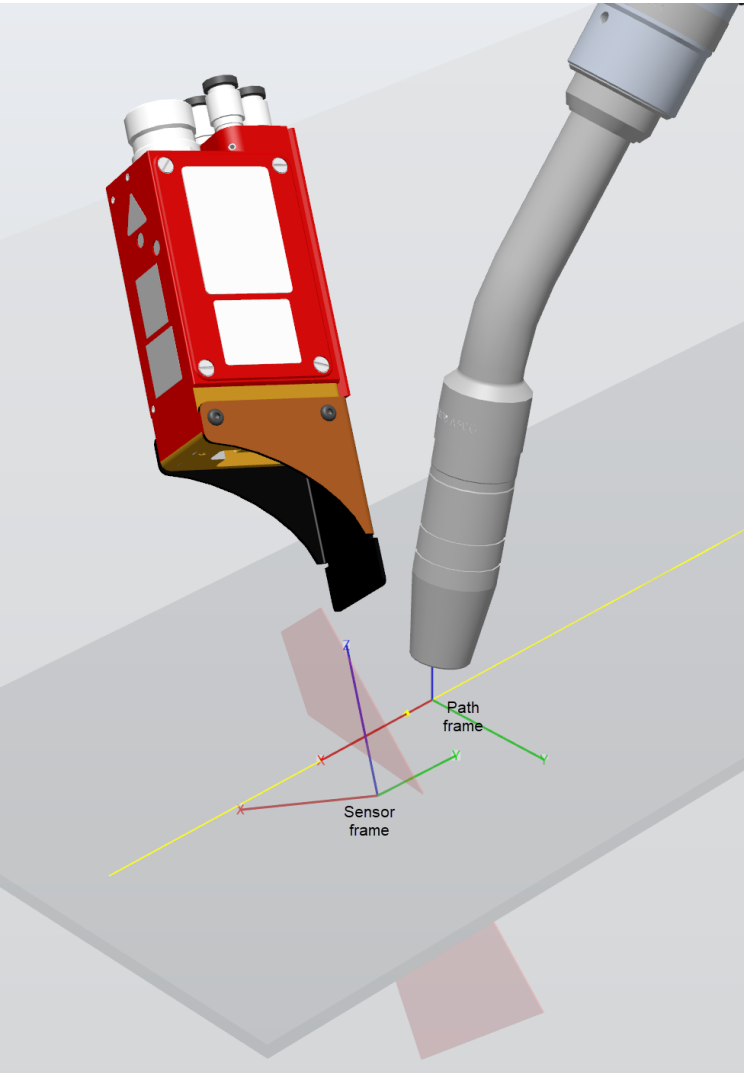
8.1.1.5 ArcCEnd - Arc welding end with circular motion

ArcWare

Continued

The following predefined values are available:

Value	Description
CAP_OFFSET_FRAME_SENSOR	The sensor measurement coordinate system. This is the default value, if this optional argument is not present.
CAP_OFFSET_FRAME_PATH	The path coordinate system.



xx2400000789

[ \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.

Continues on next page

[\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).

### 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

*Continues on next page*

## 8 RAPID reference

### 8.1.1.5 ArcCEnd - Arc welding end with circular motion

ArcWare

Continued

Error constant (ERRNO value)	Description
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 16](#)):

- 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.



#### 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 AW\_WIRE\_ERR error in any of the synchronized robots.

#### Example

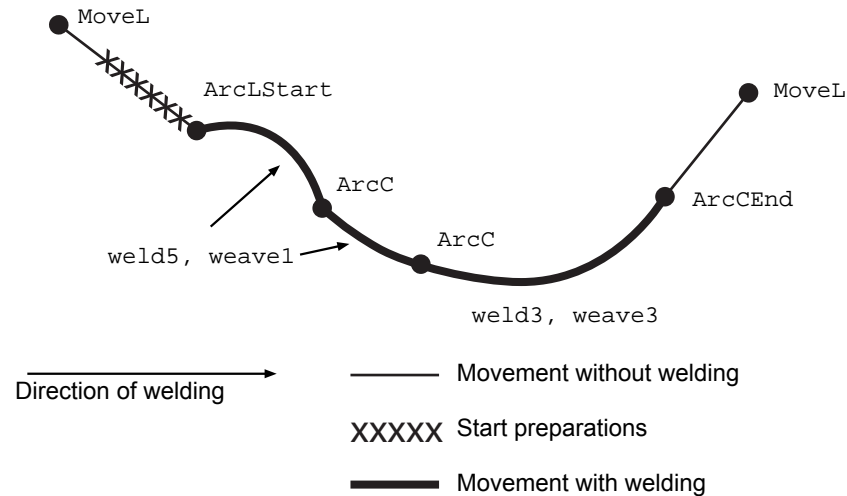
```
MoveL ...
ArcLStart *,v100, seam1, weld5 \Weave:=weave1,
    fine,gun1\Wobj:=wobj1;
ArcC *, *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcC *, *, v100, seam1,weld5 \Weave:=weave1, z10, gun1\Wobj:=wobj1;
ArcCEnd *, *, v100, seam1,weld3\Weave:=weave3,
    fine,gun1\Wobj:=wobj1;
MoveL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

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.

Continues on next page

## 8.1.1.5 ArcCEnd - Arc welding end with circular motion

*ArcWare*  
Continued



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` 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

```
ArcCEnd
[CirPoint ':='] <expression (IN) of robtarg>
[ToPoint ':='] <expression (IN) of robtarg>
[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>]
['\ ' TrackOffsetFrame ':=' < expression (IN) of captrackoffsetsframe
> ]
['\ ' 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>]';'
```

*Continues on next page*

## 8 RAPID reference

### 8.1.1.5 ArcCEnd - Arc welding end with circular motion

ArcWare

Continued

#### Related information

Information	Described in
Performing a circular motion weld	<a href="#">ArcC - Arc welding with circular motion on page 133</a>
Other positioning instructions	<i>Technical reference manual - RAPID Overview</i>
Definition of speed, speeddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of zone data, zonedata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of tools, tooldata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of work objects, wobjdata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
MoveL	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of loads, loaddata	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of seam data	<a href="#">seamdata - Seam data on page 166</a>
Definition of weld data	<a href="#">welddata - Weld data on page 180</a>
Definition of weave data	<a href="#">weavedata - Weave data on page 173</a>
Installation parameters for welding equipment and welding functions	<a href="#">System parameters on page 15</a>
Movements in general	<i>Technical reference manual - RAPID Overview</i>
Coordinate systems	<i>Technical reference manual - RAPID Overview</i>
Process phases and sub-activities	<a href="#">Programming on page 45</a>
<i>Path Corrections</i>	<i>Application manual - Controller software OmniCore</i>

### 8.1.1.6 ArcMoveL, ArcMoveC, ArcMoveJ, ArcMoveAbsJ, ArcMoveExtJ - Wrapped movement instructions for non-welding robot or additional axis in Arc *MultiMove* systems

#### 8.1.1.6 ArcMoveL, ArcMoveC, ArcMoveJ, ArcMoveAbsJ, ArcMoveExtJ - Wrapped movement instructions for non-welding robot or additional axis in Arc *MultiMove* systems

##### Usage

In a *MultiMove* system the following instructions are necessary to get the Arc error handling working correctly between welding robots, and with non-welding robots or additional axes.

- ArcMoveL - Wrapper for MoveL
- ArcMoveC - Wrapper for MoveC
- ArcMoveJ - Wrapper for MoveJ
- ArcMoveAbsJ - Wrapper for MoveAbsJ
- ArcMoveExtJ - Wrapper for MoveExtJ

These wrapped instructions have the same arguments and functionality as their originals have, plus one additional argument (`\Start`) and an integrated error handler for ArcWare for OmniCore. Whenever an `ArcX` instruction is synchronized with a non-welding robot or additional axis, these instructions must be used.

##### Basic examples

###### 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;
```

##### Additional arguments

ArcMoveXX ... (arguments from the original instruction) ... [`\Start`]

[`\Start`]

**Data type:** switch

This argument is placed after the last argument of the original instruction. It must be used to indicate start to the corresponding `ArcLStart` instruction in a *MultiMove*

*Continues on next page*

## 8 RAPID reference

---

### 8.1.1.6 ArcMoveL, ArcMoveC, ArcMoveJ, ArcMoveAbsJ, ArcMoveExtJ - Wrapped movement instructions for non-welding robot or additional axis in Arc *MultiMove* systems

*Continued*

system (synchronized), that is, the synchronized welding robot has a [`\Start`] argument.

---

#### Limitations

These instructions must not be used in an UNDO handler or RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset or Step.

---

#### Syntax

```
ArcMoveL, ArcMoveC, ArcMoveJ, ArcMoveAbsJ, ArcMoveExtJ
...
(arguments from the original instruction)
...
['\ ' Start]';'
```



### 8.1.1.7 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 intnol WITH TuneTrp;
  ITimer,0.5 ,intnol;
  ! Weld the seam
  ArcLStart p1, v100, seam1, weld5 \Weave:=noweave, fine, gun1;
  ArcLEnd p2, v100, seam1, weld5 \Weave:=noweave, fine, gun1;
  IDelete intnol;
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*

## 8 RAPID reference

### 8.1.1.7 ArcRefresh - Refresh arc weld data

ArcWare

Continued

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 >]';'
```

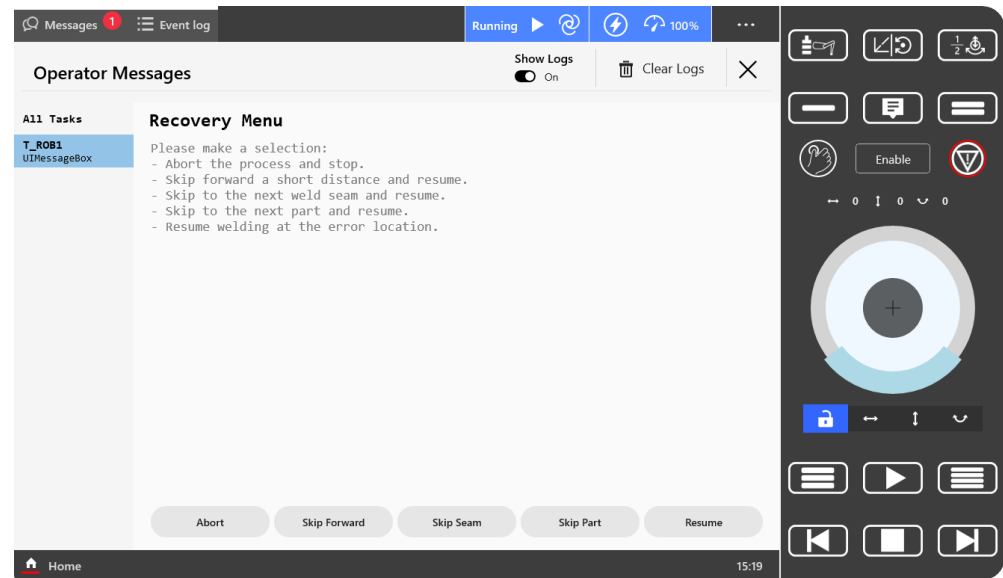
#### Related information

Information	Described in
Performing a circular weld	<a href="#">ArcC - Arc welding with circular motion on page 133</a>
Performing a linear weld	<a href="#">ArcL - Arc welding with linear motion on page 115</a>
Other positioning instructions	<i>Technical reference manual - RAPID Overview</i>
Definition of speed, <code>speeddata</code>	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of zone data, <code>zonedata</code>	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of tools, <code>tooldata</code>	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of work objects, <code>wobjdata</code>	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
<code>MoveL</code>	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of loads, <code>loaddata</code>	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>
Definition of seam data	<a href="#">seamdata - Seam data on page 166</a>
Definition of weld data	<a href="#">welddata - Weld data on page 180</a>
Definition of weave data	<a href="#">weavedata - Weave data on page 173</a>
Installation parameters for welding equipment and welding functions	<a href="#">System parameters on page 15</a>
Movements in general	<i>Technical reference manual - RAPID Overview</i>
Coordinate systems	<i>Technical reference manual - RAPID Overview</i>
Process phases and sub-activities	<a href="#">Programming on page 45</a>

### 8.1.1.8 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 ArcWare for OmniCore attempts to re-ignite the arc.



xx2400000106

#### 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	<a href="#">RecoveryPosSet - Set the recovery position on page 157</a>
Reset the recovery position	<a href="#">RecoveryPosReset - Reset the recovery position on page 160</a>

*Continues on next page*

## 8 RAPID reference

---

### 8.1.1.8 RecoveryMenu - Display the recovery menu

*ArcWare*

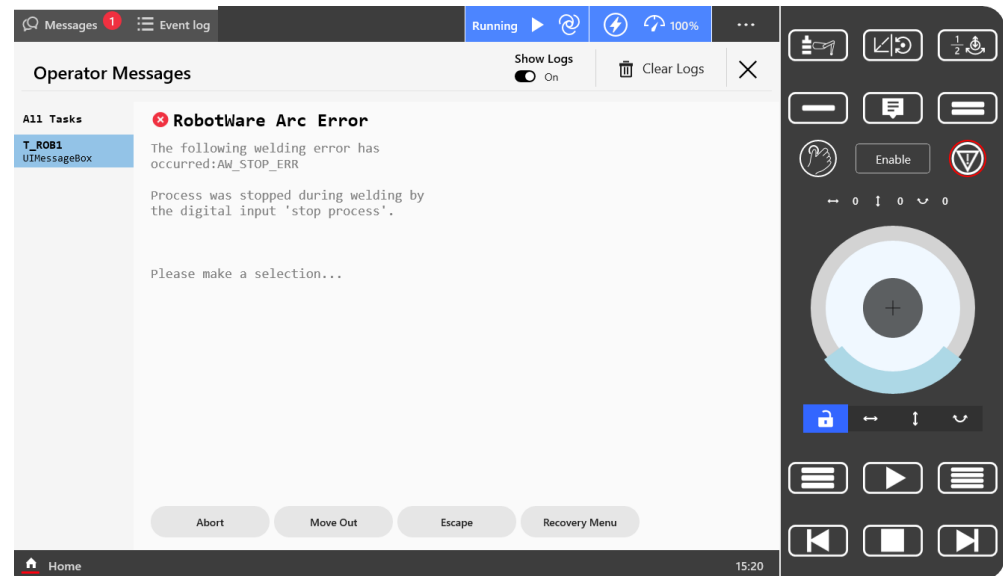
*Continued*

Information	Described in
Configure the recovery menu	<a href="#">Configure the recovery menu on page 77</a>
Configuring Weld Error Recovery	<a href="#">Configuring Weld Error Recovery on page 75</a>

## 8.1.1.9 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.



xx2400000107

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 are 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

## 8 RAPID reference

### 8.1.1.9 RecoveryPosSet - Set the recovery position

ArcWare

Continued

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	<a href="#">RecoveryPosReset - Reset the recovery position on page 160</a>
Display the recovery menu	<a href="#">RecoveryMenu - Display the recovery menu on page 155</a>

Continues on next page

### 8.1.1.9 RecoveryPosSet - Set the recovery position

*ArcWare*

*Continued*

Information	Described in
Configure the recovery menu	<a href="#">Configure the recovery menu on page 77</a>
Configure weld error recovery	<a href="#">Configuring Weld Error Recovery on page 75</a>

## 8 RAPID reference

### 8.1.1.10 RecoveryPosReset - Reset the recovery position

ArcWare

### 8.1.1.10 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,w1\Weave:=wv1,fine,tWeldGun;  
  ArcL *,v500,sm1,w1\Weave:=wv1,z10,tWeldGun;  
  SetDO doClamp,high;  
  RecoveryPosSet;  
  ArcL *,v500,sm1,w1\Weave:=wv1,z10,tWeldGun;  
  ArcLEnd *,v500,sm1,w1\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	<a href="#">RecoveryPosSet - Set the recovery position on page 157</a>
Display the recovery menu	<a href="#">RecoveryMenu - Display the recovery menu on page 155</a>

*Continues on next page*



### 8.1.1.10 RecoveryPosReset - Reset the recovery position

*ArcWare*

*Continued*

Information	Described in
Configure the recovery menu	<a href="#">Configure the recovery menu on page 77</a>
Configure weld error recovery	<a href="#">Configuring Weld Error Recovery on page 75</a>

## 8 RAPID reference

---

### 8.1.2.1 arcddata - Arc data

ArcWare

## 8.1.2 Data types

### 8.1.2.1 arcddata - Arc data

---

#### Usage and description

`arcddata` 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 15](#)) 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 15](#)) 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 16](#). This parameter is only available if weld voltage ([System parameters on page 15](#)) is defined.

`wirefeed`

**Data type:** `num`

This parameter is only available, if wirefeed ([System parameters on page 15](#)) is defined. The feed speed of the weld electrode during the weld phase. The unit for `arcddata` components that specify a velocity, is defined by the parameter *Units*, see [Units and values on page 19](#).

The value specified is scaled and sent to the corresponding analog output, in accordance with the setting in [Defining arc welding systems on page 16](#).

`control`

**Data type:** `num`

Analog tuning value sent to certain welders.

`current`

**Data type:** `num`

The welding current (in Ampere) during the weld phase.

*Continues on next page*

The value specified is scaled and sent to the corresponding analog output, in accordance with the setting in [Defining arc welding systems on page 16](#). This parameter is only available if current ([System parameters on page 15](#)) 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 16](#). This parameter is only available if weld voltage ([System parameters on page 15](#)) is defined.

wirefeed2

**Data type:** num

The wire feed speed during the weld phase. Used in a TwinWire setup.

This parameter is only available, if wirefeed ([System parameters on page 15](#)) 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 [Units and values on page 19](#).

The value specified is scaled and sent to the corresponding analog output, in accordance with the setting in [Defining arc welding systems on page 16](#).

control2

**Data type:** num

Analog tuning value sent to certain welders. Used in a TwinWire setup.

track\_reference

**Data type:** num

The reference value for height corrections (Z-direction).

## 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	<a href="#">seamdata - Seam data on page 166</a>
Installation parameters for welding	<a href="#">System parameters on page 15</a>

*Continues on next page*

## 8 RAPID reference

---

### 8.1.2.1 arcddata - Arc data

*ArcWare*

*Continued*

Information	Described in
Process phases and time diagrams	<a href="#">Programming on page 45</a>
Circular arc welding instructions	<a href="#">ArcC - Arc welding with circular motion on page 133</a>
Linear arc welding instructions	<a href="#">ArcL - Arc welding with linear motion on page 115</a>

### 8.1.2.2 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 linear motion	<a href="#">ArcLStart - Arc welding start with linear motion on page 105</a>

## 8 RAPID reference

---

### 8.1.2.3 seamdata - Seam data

ArcWare

### 8.1.2.3 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 180](#).

`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.

The data structure described here is valid for *Standard I/O Welder*. If a power source specific Add-In is installed, the `seamdata` structure might be slightly different. You find its description in the user manual for that Add-In.

All voltages can be expressed in two ways (determined by the welding equipment):

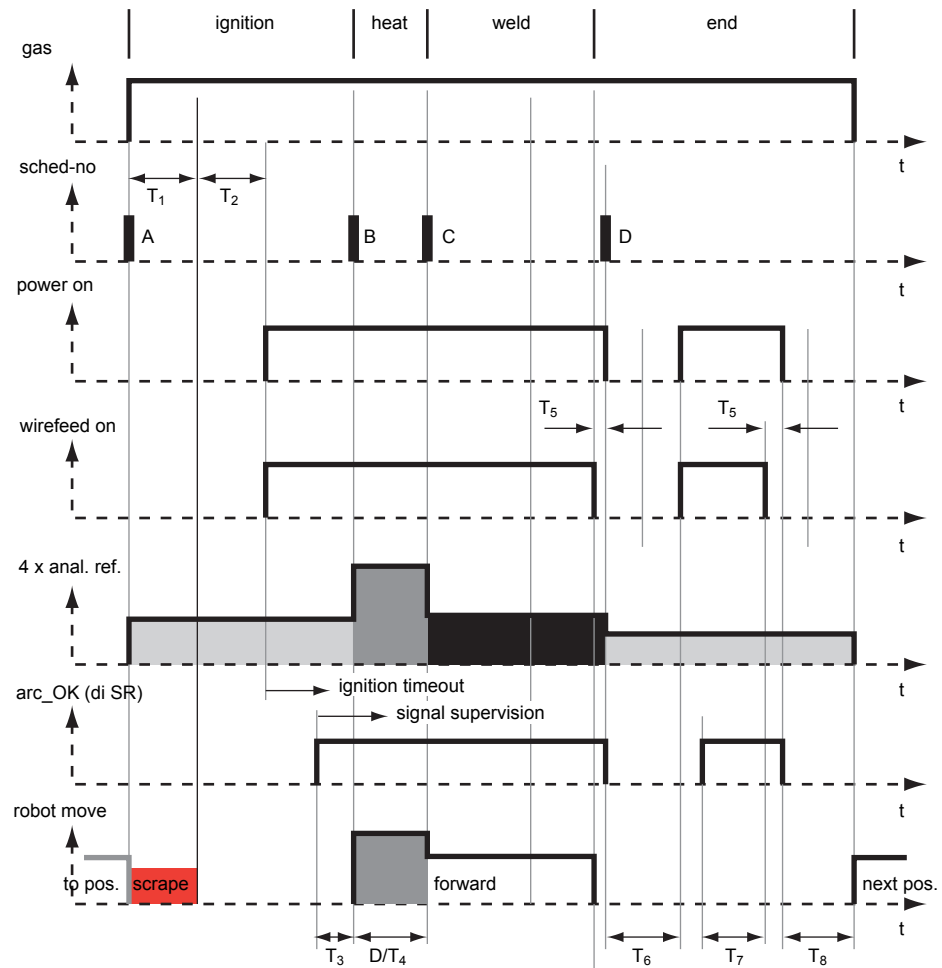
- As absolute values (only positive values are used in this case).
- As corrections of values set in the process equipment (both positive and negative values can be used in this case).

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

Continues on next page

## 8 RAPID reference

---

### 8.1.2.3 seamdata - Seam data

ArcWare

Continued

---

#### 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`, 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 `ArcLStart` 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.

startcurrent\_time

Data type: num

The start current value is active `startcurrent_time` (in seconds) until the current is ramped up/down to the main current value. Only the analogue output *Feed Reference* (wirefeed speed) can be used to ramp up/down the current. System parameter *Ignition On* must be *TRUE* to use the start current time.

Allowed values: 0 - 10 seconds

startcurrent\_slope

Data type: num

`startcurrent_slope` (in seconds) is the time to ramp up/down from the start current value to the main current value. A practical value for the slope time is any value < 1.0 seconds. Only the analogue output *Feed Reference* (wirefeed speed) can be used to ramp up/down the current. System parameter *Ignition On* must be *TRUE* to use the start current slope time.

Allowed values: 0 - 10 seconds

ign\_arc

Data type: arcdata

Weld parameters during the ignition phase. See definition of `arcdata`, [arcdata - Arc data on page 162](#).

ign\_move\_delay (ignition movement delay)

Data type: num

Continues on next page



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.

The unit for `seamdata` components that specify a velocity, is defined by parameter *Units*, see [Units and values on page 19](#).

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 [Units and values on page 19](#).

heat\_arc

Data type: arcdata

Weld parameters during the heat phase. See definition of [arcdata - Arc data on page 162](#).

---

### Component group: End

endcurrent\_time

Data type: num

The start current value is active `endcurrent_time` (in seconds) the current value is active. Only the analogue output *Feed Reference* (wirefeed speed) can be used to ramp up/down the current. System parameter *Fill On* must be *TRUE* to use the end current time. The parameter `fill_time` must be set to 0, otherwise `fill_time` is used as `endcurrent_time`.

Allowed values: 0 - 10 seconds

*Continues on next page*

## 8 RAPID reference

---

### 8.1.2.3 seamdata - Seam data

ArcWare

Continued

endcurrent\_slope

**Data type:** num

endcurrent\_slope (in seconds) is the time to ramp up/down from the main current value to the end current value. A practical value for the slope time is any value < 1.0 seconds. Ramping starts the configured `endcurrent_time` before the arc end position is reached. Only the analogue output *Feed Reference* (wirefeed speed) can be used to ramp up/down the current. System parameter *Fill On* must be *TRUE* to use the end current slope time.

Allowed values: 0 - 10 seconds

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 162](#).

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

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 162](#).

postflow\_time

**Data type:** num

*Continues on next page*

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>
  <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	<a href="#">welddata - Weld data on page 180</a>
Arc data	<a href="#">arcdata - Arc data on page 162</a>

*Continues on next page*

## 8 RAPID reference

---

### 8.1.2.3 seamdata - Seam data

*ArcWare*

*Continued*

Information	Described in
Installation parameters for welding	<a href="#">System parameters on page 15</a>
Process phases and time diagrams	<a href="#">Programming on page 45</a>
Circular arc welding instruction	<a href="#">ArcC - Arc welding with circular motion on page 133</a>
Linear arc welding instruction	<a href="#">ArcL - Arc welding with linear motion on page 115</a>

## 8.1.2.4 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 173](#).

- 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 [Units and values on page 19](#).

**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 15](#), and components of [weavedata - Weave data on page 173](#).

## 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.

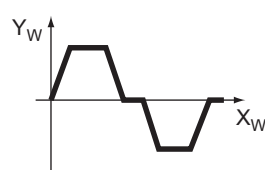
**Note**

The path coordinate system is shown with x-axis in path direction.

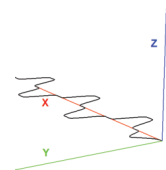
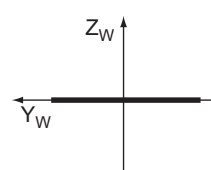
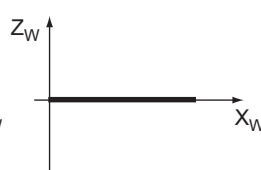
## Types of weave shape

0 - No weaving.

1 - Zigzag weaving results in a weaving horizontal to the seam.



xx1200000714



*Continues on next page*

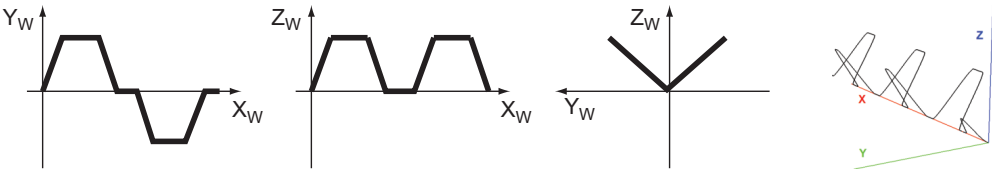
8 RAPID reference

8.1.2.4 weavedata - Weave data

ArcWare

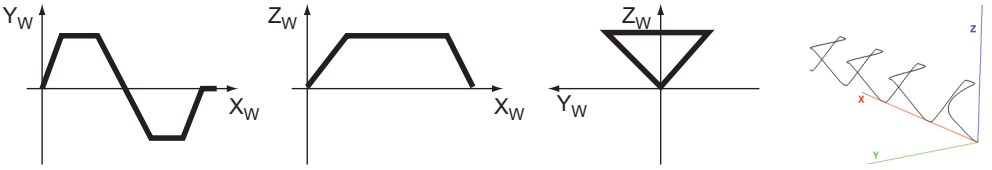
Continued

2 - V-shaped weaving results in weaving in the shape of a "V", vertical to the seam.



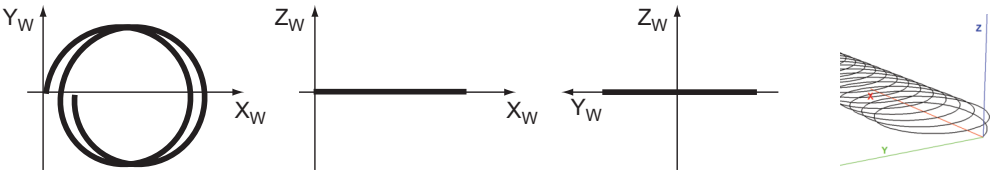
xx1200000715

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.



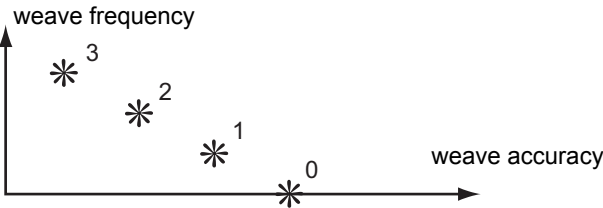
xx1200000717

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

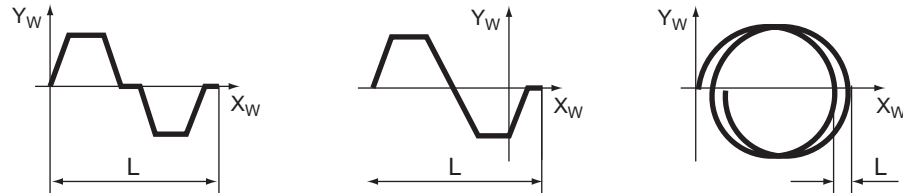
Data type: num

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.

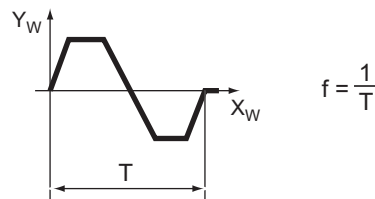
For circular weaving the length attribute defines the rotation frequency of the TCP. The TCP rotates left with a positive length value, and right with a negative length value. L is calculated as  $L = \text{weld\_speed} / \text{weave\_length}$ .



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.

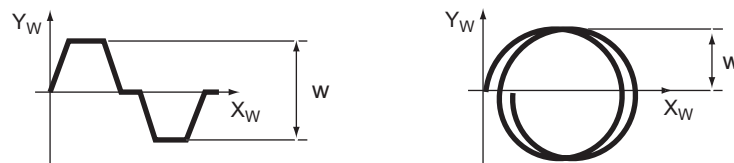


xx1200000720

`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

`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.

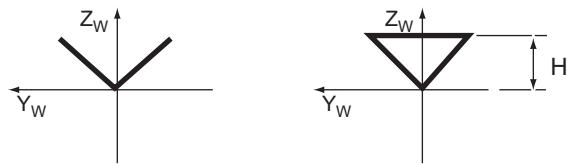
*Continues on next page*

8 RAPID reference

8.1.2.4 weavedata - Weave data

ArcWare

Continued

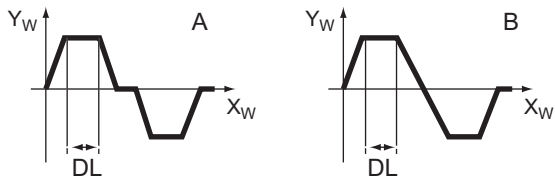


xx1200000722

dwel\_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.



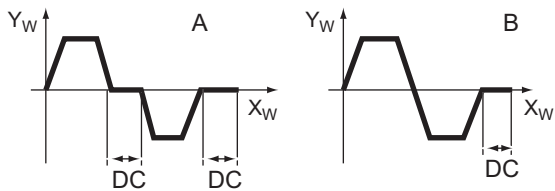
xx1200000723

A	Zigzag and V-shaped weaving
B	Triangular weaving

dwel\_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.



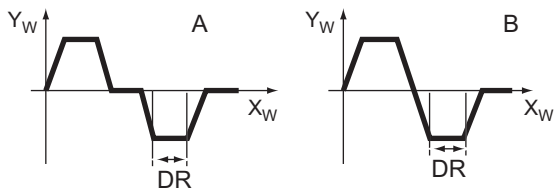
xx1200000724

A	Zigzag and V-shaped weaving
B	Triangular weaving

dwel\_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

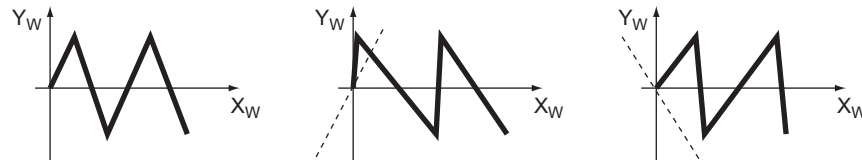
A	Zigzag and V-shaped weaving
---	-----------------------------

Continues on next page



**B Triangular weaving****weave\_dir** (weave direction angle)**Data type:** num

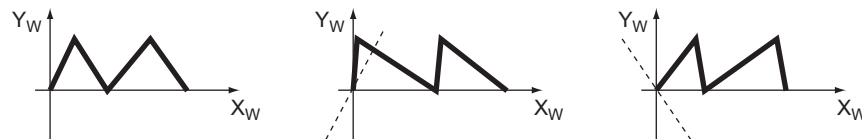
The weave direction angle horizontal to the seam. An angle of zero degrees results in a weave vertical to the seam.



xx1200000726

**weave\_tilt** (weave tilt angle)**Data type:** num

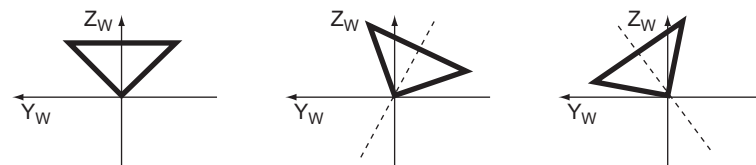
The weave tilt angle, vertical to the seam. An angle of zero degrees results in a weave which is vertical to the seam.



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

**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).

*Continues on next page*

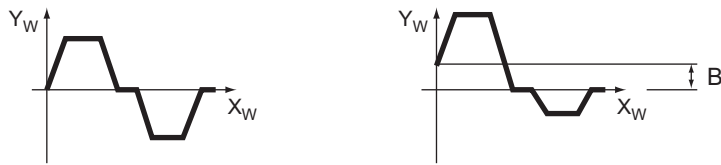
## 8 RAPID reference

---

### 8.1.2.4 weavedata - Weave data

ArcWare

Continued



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

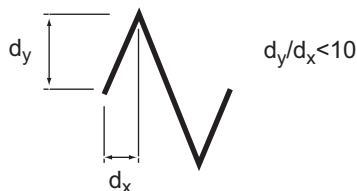
This component holds the last saved original value for the tuning function. It can be used for a quick restore of a changed value to the original value.

---

### 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.



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.
- 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.

*Continues on next page*

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>
  <dwelleft of num>
  <dwelcenter of num>
  <dwelright 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	<a href="#">System parameters on page 15</a>
Process phases and timing schedules	<a href="#">Programming on page 45</a>
Arc welding instructions	<a href="#">ArcL - Arc welding with linear motion on page 115</a> <a href="#">ArcC - Arc welding with circular motion on page 133</a>

## 8 RAPID reference

### 8.1.2.5 welddata - Weld data

ArcWare

### 8.1.2.5 welddata - 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 166](#).

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 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.

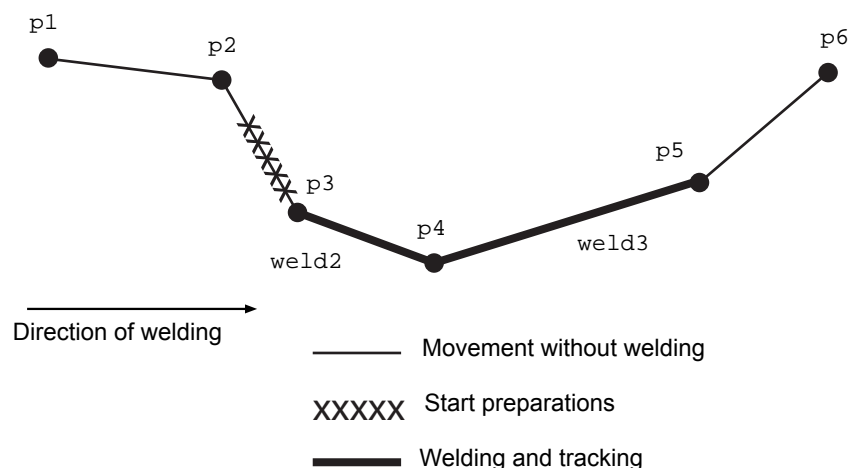
All voltages can be expressed in two ways (determined by the welding equipment):

- 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;
```

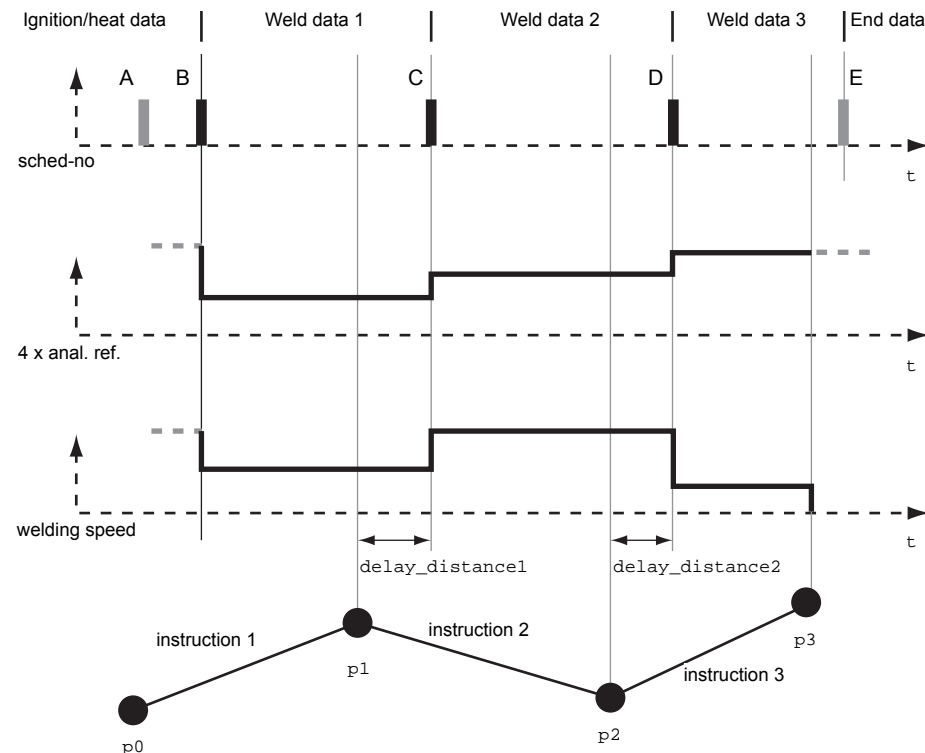


xx1200000733

Every welding instruction has different welddata. As the instruction ArcLStart is used in the first instruction, the first welddata is actually never used.

Continues on next page

## Welding sequence



xx1200000734

## Components

weld\_speed

**Data type:** num

The desired welding speed.

The unit for welddata components that specify a velocity, is defined by the parameter *Units*, see [Units and values on page 19](#).

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

Continues on next page

## 8 RAPID reference

---

### 8.1.2.5 welddata - Weld data

ArcWare

Continued

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`

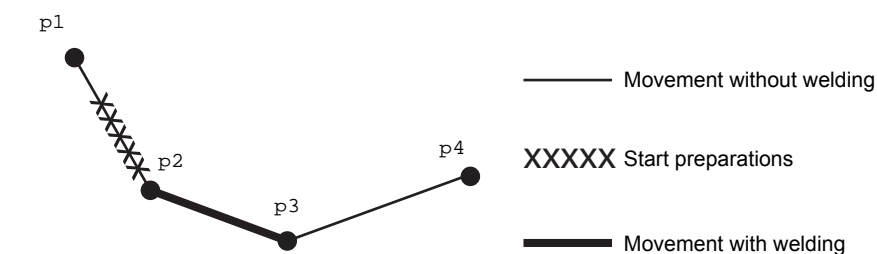
The original weld wirefeed speed during the weld phase. It is used internally by tuning functions.

This parameter is only available, if wirefeed is defined. See [System parameters on page 15](#).

---

#### 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;
```

Continues on next page

```
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	<a href="#">seamdata - Seam data on page 166</a>
Arc data	<a href="#">arcdata - Arc data on page 162</a>
Installation parameters for welding	<a href="#">System parameters on page 15</a>
Process phases and time diagrams	<a href="#">Programming on page 45</a>
Circular arc welding instructions	<a href="#">ArcC - Arc welding with circular motion on page 133</a>
Linear arc welding instructions	<a href="#">ArcL - Arc welding with linear motion on page 115</a>

## 8 RAPID reference

---

### 8.2.1.1 ALS - Arcwelding Linear Start

*ArcWare*

## 8.2 Collaborative robots (CRB 15000)

### 8.2.1 Instructions

#### 8.2.1.1 ALS - Arcwelding Linear Start

---

##### Usage

ALS is an instruction used for arc welding with CRB 15000.

---

##### Basic examples

The following example illustrates the instruction ALS.

##### Example 1

```
ALS Location1, 10, 1, tWeldGun;
```

The robot moves linearly to position `Location1` and prepares gas preflow and sets the job number in advance.

Default values for `gas_preflow` is 0.2 s and `gas_purge` is 0.05 s. The default value can be changed as follows:

- `sm_default.purge_time := 'new value';`
- `sm_default.preflow_time := 'new value';`

---

##### Arguments

```
ALS ToPoint, WeldSpeed, Job, Tool [\Weave] [\Wobj]
```

`ToPoint`

**Data type:** `robtarget`

Start position of the weld.

`WeldSpeed`

**Data type:** `num`

Weld speed in mm/s.

`Job`

**Data type:** `num`

Job number sent to the welder.

`Tool`

**Data type:** `tooldata`

The tool used during the movement to `ToPoint`.

`[\Weave]`

**Data type:** `weavedata`

Weave data used for the movement.

`[\Wobj]`

**Data type:** `wobjdata`

The work object used during the movement.

*Continues on next page*



**Syntax**

ALS

```
[ ToPoint ':=' ] < expression (IN) of robtargt > ','  
[ WeldSpeed ':=' ] < expression (IN) of num >  
[ Job ':=' ] < expression (IN) of num >  
[ Tool ':=' ] < persistent (PERS) of tooldata >  
[ '\ ' Weave ':=' < expression (IN) of weavedata > ]  
[ '\ ' WObj ':=' < persistent (PERS) of wobjdata > ] ';' 
```

## 8 RAPID reference

---

### 8.2.1.2 AL - Arcwelding Linear

ArcWare

### 8.2.1.2 AL - Arcwelding Linear

---

#### Usage

AL is an instruction used for arc welding with CRB 15000.

---

#### Basic examples

The following example illustrates the instruction AL.

##### Example 1

```
AL Location1, 10, 1, z5, tWeldGun;
```

The robot moves linearly to position Location1 with welding process active.

---

#### Arguments

```
AL ToPoint, WeldSpeed, Job, Zone, Tool [\Weave] [\Wobj]
```

ToPoint

**Data type:** robtarget

Start position of the weld.

WeldSpeed

**Data type:** num

Weld speed in mm/s.

Job

**Data type:** num

Job number sent to the welder.

Zone

**Data type:** zonedata

Tool

**Data type:** tooldata

The tool used during the movement to ToPoint.

[\Weave]

**Data type:** weavedata

Weave data used for the movement.

[\Wobj]

**Data type:** wobjdata

The work object used during the movement.

---

#### Syntax

```
AL
[ ToPoint ':' '=' ] < expression (IN) of robtarget > ','
[ WeldSpeed ':' '=' ] < expression (IN) of num >
[ Job ':' '=' ] < expression (IN) of num >
[ Zone ':' '=' ] < expression (IN) of zonedata >
[ Tool ':' '=' ] < persistent (PERS) of tooldata >
```

*Continues on next page*

```
[ '\ ' Weave ' := ' < expression (IN) of weavedata > ]  
[ '\ ' WObj ' := ' < persistent (PERS) of wobjdata > ] ';' 
```

## 8 RAPID reference

---

### 8.2.1.3 ALE - Arcwelding Linear End

*ArcWare*

### 8.2.1.3 ALE - Arcwelding Linear End

---

#### Usage

ALE is an instruction used for arc welding with CRB 15000.

---

#### Basic examples

The following example illustrates the instruction ALE.

##### Example 1

```
ALE Location1, 10, 1, tWeldGun;
```

The robot moves linearly to position Location1 with welding process active.

When the robot has reached Location1, the welding process is ended.

---

#### Arguments

```
ALE ToPoint, WeldSpeed, Job, Tool [\Weave] [\Wobj]
```

ToPoint

**Data type:** robtarget

Move linearly to ToPoint.

WeldSpeed

**Data type:** num

Weld speed in mm/s.

Job

**Data type:** num

Set the configured group output to this value.

Tool

**Data type:** tooldata

The tool used during the movement to ToPoint.

[\Weave]

**Data type:** weavedata

Weave data used for the movement.

[\Wobj]

**Data type:** wobjdata

The work object used during the search.

---

#### Syntax

```
ALE
[ ToPoint ':' ] < expression (IN) of robtarget > ','
[ WeldSpeed ':' ] < expression (IN) of num >
[ Job ':' ] < expression (IN) of num >
[ Tool ':' ] < persistent (PERS) of tooldata >
[ '\ Weave ':' < expression (IN) of weavedata > ]
[ '\ WObj ':' < persistent (PERS) of wobjdata > ] ';'

```

### 8.2.1.4 AC - Arcwelding Circular

---

#### Usage

AC is an instruction used for arc welding with CRB 15000.

---

#### Basic examples

The following example illustrates the instruction AC.

##### Example 1

```
AC Location1, Location2, 10, 1, z5, tWeldGun;
```

The robot moves circularly to position `Location1` and `Location2` with welding process active.

---

#### Arguments

```
AC CirPoint, ToPoint, WeldSpeed, Job, Zone, Tool [\Weave] [\Wobj]
```

`CirPoint`

**Data type:** `robtarget`

Circle position of the weld.

`ToPoint`

**Data type:** `robtarget`

`ToPoint` position of the weld.

`WeldSpeed`

**Data type:** `num`

Weld speed in mm/s.

`Job`

**Data type:** `num`

Job number sent to the welder.

`Zone`

**Data type:** `zonedata`

`Tool`

**Data type:** `tooldata`

The tool used during the movement to `ToPoint`.

`[\Weave]`

**Data type:** `weavedata`

Weave data used for the movement.

`[\Wobj]`

**Data type:** `wobjdata`

The work object used during the movement.

---

#### Syntax

AC

*Continues on next page*

## 8 RAPID reference

---

### 8.2.1.4 AC - Arcwelding Circular

*ArcWare*

*Continued*

```
[ CirPoint ':=' ] < expression (IN) of robtarget > ','  
[ ToPoint ':=' ] < expression (IN) of robtarget > ','  
[ WeldSpeed ':=' ] < expression (IN) of num >  
[ Job ':=' ] < expression (IN) of num >  
[ Zone ':=' ] < expression (IN) of zonedata >  
[ Tool ':=' ] < persistent (PERS) of tooldata >  
[ '\ Weave ':=' < expression (IN) of weavedata > ]  
[ '\ WObj ':=' < persistent (PERS) of wobjdata > ] ';' 
```

## 8.2.1.5 ACE - Arcwelding Circular End

### Usage

ACE is an instruction used for arc welding with CRB 15000.

### Basic examples

The following example illustrates the instruction ACE.

#### Example 1

```
ACE Location1, Location2, 10, 1, tWeldGun;
```

The robot moves circularly to position Location1 and Location2 with welding process active. When the robot has reached Location2, the welding process is ended.

### Arguments

```
ACE CirPoint, ToPoint, WeldSpeed, Job, Tool [\Weave] [\Wobj]
```

CirPoint

**Data type:** robtarget

Circle position of the weld.

ToPoint

**Data type:** robtarget

ToPoint position of the weld.

WeldSpeed

**Data type:** num

Weld speed in mm/s.

Job

**Data type:** num

Job number sent to the welder.

Tool

**Data type:** tooldata

The tool used during the movement to ToPoint.

[\Weave]

**Data type:** weavedata

Weave data used for the movement.

[\Wobj]

**Data type:** wobjdata

The work object used during the movement.

### Syntax

ACE

```
[ CirPoint ':' = ] < expression (IN) of robtarget > ','
[ ToPoint ':' = ] < expression (IN) of robtarget > ','
```

*Continues on next page*

## 8 RAPID reference

---

### 8.2.1.5 ACE - Arcwelding Circular End

*ArcWare*

*Continued*

```
[ WeldSpeed ':=' ] < expression (IN) of num >  
[ Job ':=' ] < expression (IN) of num >  
[ Tool ':=' ] < persistent (PERS) of tooldata >  
[ '\ ' Weave ':=' < expression (IN) of weavedata > ]  
[ '\ ' WObj ':=' < persistent (PERS) of wobjdata > ] ';' 
```



# Index

## A

AC, 189  
 ACE, 191  
 AL, 186  
 ALE, 188  
 ALS, 184  
 ArcC, 133  
 ArcCEnd, 142  
 arcdata, 162  
 Arc Equipment, 23  
 Arc Equipment Class, 24  
 ArcL, 115  
 ArcLEnd, 124  
 ArcLStart, 105  
 ArcMoveAbsJ, 151  
 ArcMoveC, 151  
 ArcMoveExtJ, 151  
 ArcMoveJ, 151  
 ArcMoveL, 151  
 ArcRefresh, 153  
 Arc System Properties, 18  
 Arc System settings, 17  
 Arc Units, 22  
 arc welding data, 45  
 arc welding instructions, 45

## D

data types  
   arcdata, 162  
   flystartdata, 165  
   seamdata, 166  
   weavedata, 173  
   welddata, 180

## E

error levels, 39

## F

flystartdata, 165

## G

gas purge, 54  
 Generic Equipment Class, 15, 25

## I

instructions  
   ArcC, 133  
   ArcCEnd, 142  
   ArcL, 115

ArcLEnd, 124  
 ArcLStart, 105  
 ArcMoveAbsJ, 151  
 ArcMoveC, 151  
 ArcMoveExtJ, 151  
 ArcMoveJ, 151  
 ArcMoveL, 151  
 ArcRefresh, 153  
 RecoveryMenu, 155  
 RecoveryPosReset, 160  
 RecoveryPosSet, 157

## L

limitations  
   MultiMove, 61

## M

manual gas purge, 54  
 manual wirefeed, 53  
 MultiMove  
   description, 55  
   programming, 57

## O

Optical Sensor, 36

## P

programming, 47  
   MultiMove, 57  
   Weld Error Recovery, 65

## R

RecoveryMenu, 155  
 RecoveryPosReset, 160  
 RecoveryPosSet, 157

## S

seamdata, 166

## U

unit settings, 19

## W

weavedata, 173  
 welddata, 180  
 weld error handling, 63  
 Weld Error Recovery  
   description, 63  
   programming, 65  
 weld errors, 63  
 wirefeed, 53





**ABB AB****Robotics & Discrete Automation**

S-721 68 VÄSTERÅS, Sweden

Telephone +46 10-732 50 00

**ABB AS****Robotics & Discrete Automation**

Nordlysvegen 7, N-4340 BRYNE, Norway

Box 265, N-4349 BRYNE, Norway

Telephone: +47 22 87 2000

**ABB Engineering (Shanghai) Ltd.**

Robotics & Discrete Automation

No. 4528 Kangxin Highway

PuDong New District

SHANGHAI 201315, China

Telephone: +86 21 6105 6666

**ABB Inc.****Robotics & Discrete Automation**

1250 Brown Road

Auburn Hills, MI 48326

USA

Telephone: +1 248 391 9000

**[abb.com/robotics](http://abb.com/robotics)**