

---

ROBOTICS

# Application manual

## WireSense for Fronius TPS/i



Trace back information:  
Workspace Main version a472  
Checked in 2022-09-20  
Skribenta version 5.5.019

**Application manual**  
**WireSense for Fronius TPS/i**

RobotWare 6.14

Document ID: 3HAC082151-001

Revision: B

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

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

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

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

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

© Copyright 2022 ABB. All rights reserved.  
Specifications subject to change without notice.

# Table of contents

Overview of this manual .....	7
<b>1 Introduction</b> .....	<b>9</b>
1.1 Product overview .....	9
1.2 Operation overview .....	10
1.3 Prerequisites .....	11
1.3.1 System prerequisites .....	11
1.3.2 User prerequisites .....	14
<b>2 Installation</b> .....	<b>15</b>
2.1 Safety instructions .....	15
2.2 Hardware installation .....	16
2.3 Software installation .....	17
2.3.1 About WireSense function package .....	17
<b>3 System parameters</b> .....	<b>19</b>
3.1 WireSense Configuration .....	19
3.2 WireSense ErrorHandler IO .....	21
3.3 WireSense Settings .....	22
3.4 WireSense Speeds .....	23
3.5 WireSense Standard Signals .....	24
<b>4 Using SearchEdge with WireSense</b> .....	<b>25</b>
4.1 Basic example .....	25
4.2 Program displacement .....	29
4.3 Using height information .....	32
4.4 Influence of the work angle of the welding torch .....	33
<b>5 Using SenseL with WireSense</b> .....	<b>37</b>
5.1 Basic example .....	37
5.2 Search distance with SenseL .....	41
<b>6 Search Error Recovery I/O Interface</b> .....	<b>43</b>
6.1 Introduction to Search Error Recovery I/O Interface .....	43
6.2 Configure search error recovery I/O interface .....	49
6.3 User defined error handling .....	51
<b>7 Production Monitoring</b> .....	<b>53</b>
<b>8 WireSense with Additional Arc Systems</b> .....	<b>55</b>
<b>9 RAPID references</b> .....	<b>57</b>
9.1 Instructions .....	57
9.1.1 SearchEdge - One-dimensional search .....	57
9.1.2 SenseL - Contour sensing .....	62
9.1.3 SwitchWireSenseSettings - Switch WireSense signals and search speed .....	67
<b>Index</b> .....	<b>69</b>

**This page is intentionally left blank**

# Overview of this manual

## About this manual

This manual explains the basics of when and how to use the WireSense software function package.

- Product overview
- Operation overview
- Requirements overview
- Software set-up
- Software reference, RAPID



### Note

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

## Usage

This manual can be used either as a reference to find out if an option is the right choice for solving a problem, or as a description of how to use an option. Detailed information regarding syntax for RAPID routines, and similar, is not described here, but can be found in the respective reference manual.

This manual is intended for:

- installation personnel
- robot programmers

## Prerequisites

The reader should:

- be familiar with industrial robots and their terminology
- be familiar with the RAPID programming language
- be familiar with system parameters and how to configure them



### Note

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

## References

References	Document ID
<i>Technical reference manual - RAPID Overview</i>	3HAC050947-001
<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>	3HAC050917-001
<i>Operating manual - IRC5 with FlexPendant</i>	3HAC050798-001
<i>Technical reference manual - System parameters</i>	3HAC050948-001
<i>Operating manual - RobotStudio</i>	3HAC032104-001

Continues on next page

## Overview of this manual

---

*Continued*

---

### Revisions

Revision	Description
A	Published with RobotWare 6.13.
B	Published with RobotWare 6.14. <ul style="list-style-type: none"><li>• Added section <a href="#">WireSense with Additional Arc Systems on page 55</a>.</li><li>• Added section <a href="#">SwitchWireSenseSettings - Switch WireSense signals and search speed on page 67</a>.</li></ul>



# 1 Introduction

## 1.1 Product overview

---

### General

The WireSense function package can be used as a tactile sensor as it uses the wire electrode as a sensor to find the location of inconsistent weld joints and offset the programmed points in a weld program.

In addition to the search and offset functionality the function package provides support for the option *659-1 Production Monitoring*.

# 1 Introduction

---

## 1.2 Operation overview

## 1.2 Operation overview

---

### General

With the WireSense function package, a part feature may be "searched" using the welding wire as the sensing portion of the torch. Searches are programmed into a weld sequence. Each search consists of two robtargets; one for the start location and one for the expected location of the part feature. While searching the wire is pulsed in a 100Hz frequency (reversing wire movement). The power source sends the height information and the edge position to the robot. For example, if a lap joint is being welded, the edge position can be precisely defined, and the system can react to any deviations. The robot adjusts the weld seam process based on an application-specific program.

It is also possible to determine the exact gap between the sheets. Edges are detected from a height of 0.5 mm. WireSense can be used with steel, stainless steel, and aluminium as well as with other alloys.

---

### Search instructions

The search instructions included in the WireSense function package software are designed to return offset information. In other words, the result of a search is the distance between where the original search location was programmed and where the robot has now found the part.

## 1.3 Prerequisites

### 1.3.1 System prerequisites

---

#### Introduction

This version of the WireSense software function package is intended for use in arc welding systems incorporating IRB 1600/1660 etc. robots.

- Add-In *FroniusTPSi-WireSense*
- Robot controller: IRC5
- RobotWare requirements: 6.11 or above
- [657-1] – *SmarTac – I/O Version*
- [633-4] – *Arc 6*

The Fronius welding equipment must fulfil the following requirements and must be CMT Ready.

- *WF60i RobactaDrive CMT*
- Wire buffer CMT or SB 60i
- WF25i REEL or WF30i REEL
- *4,067,020 – OPT/I WireSense*

---

#### WireSense function package

The WireSense function package includes software that is loaded into all arc welding motion tasks, when the option is installed. Process configuration parameters are used to connect real I/O signals and to modify the default settings.

---

#### Compatibility

The WireSense software function package can be used together with the SmarTac™ option.

WireSense works best with the official Fronius TPS/I welder interface which is available free of charge and can be downloaded from the RobotStudio Add-In page. In addition, the StdIoWelder interface is supported as well. Additional configuration might be needed.

---

#### Limitations

- The WireSense function package can only be used with the first arc system if option [651-2] – Two Additional Arc Systems is installed in combination with the Fronius TPS/I Add-In .  
See [WireSense with Additional Arc Systems on page 55](#) for more information.
- It is not recommended to restore a backup if you update an existing system with the WireSense function package since new system parameters are introduced and the process configuration database (PROC) is updated. These settings will be removed when restoring a backup that was taken prior adding the WireSense function package. If a backup is restored the default WireSense settings can be added afterwards. See chapter 3.1 – WireSense configuration

*Continues on next page*

# 1 Introduction

---

## 1.3.1 System prerequisites

*Continued*

- The WF60i RobactaDrive CMT has a hard button “F1” which will activate the Teach Mode functionality. By pressing the F1 button Teach Mode is activated internally in the TPS/I and no feedback is provided via the fieldbus interface. This will prevent WireSense from being activated from the Robot. F1 must be pressed again to switch off Teach Mode. Active TeachMode is shown with a capital “T” in the display
- This limitation might be corrected in a future firmware release for the TPS/I power source
- The following image show active Teach mode in the CMT Drive display.



xx2200001316

---

### Technical details Edge detection Fronius WireSense

- Sample rate ca. 100 Hz
- Min. sheet size 0.5 mm
- Max. sheet size 20.00 mm
- Accuracy/precision for height measurement 0.3 mm
- Max. repeatability +/- 0.2 mm
- Suggested Search speed < 25 mm/s

*Continues on next page*

- Height information present at edge detection

---

### Technical details Sensing Fronius WireSense

- Sample rate ca. 100 Hz
- Measurement range +/- 24 mm
- Accuracy/precision for height measurement 0.3 mm
- Max. repeatability +/- 0.2 mm
- Suggested Sense time max. 30 seconds
- Height information present along the path

# 1 Introduction

---

## 1.3.2 User prerequisites

### 1.3.2 User prerequisites

---

#### Robot programmer

Any competent robot programmer (RAPID language) may be self-taught to program and use basic WireSense searches.

---

#### Robot system operator

For the robotic system operator, the addition of searches is largely transparent and requires no further training.



#### Note

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

## 2 Installation

### 2.1 Safety instructions



#### **DANGER**

Before doing any work inside the cabinet, disconnect the mains power.



#### **ELECTROSTATIC DISCHARGE (ESD)**

The components are sensitive to ESD. Always use ESD protection when handling them. Use the wrist strap located on the controller.



#### **DANGER**

All personnel working with the robot system must be very familiar with the safety regulations. Incorrect operation can damage the robot or injure someone.

## 2 Installation

---

### 2.2 Hardware installation

### 2.2 Hardware installation

---

#### Component list

The WireSense function package does not consist of any additional hardware needed for the IRC5 controller. It utilizes the Fronius TPS/I WireSense functionality. The TPS/I welder must be CMT Ready. See [System prerequisites on page 11](#).



## 2.3 Software installation

### 2.3.1 About WireSense function package

#### Installation

The WireSense function package is provided as an Add-In, that needs to be installed in the robot controller using Robotstudio Installation Manager.

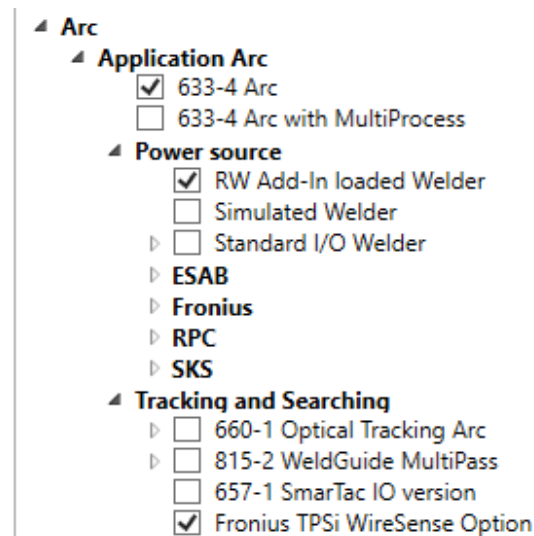
- 1 Add the option to the system.

Added Product(s)

Name	Version	Publisher	Type	Status
RobotWare	6.11.01.00	ABB	RobotWare	Installed
FroniusTPSi-WireSense	1.00.0000.04 Internal Use	ABB Robotics	AddIn	Installed
FroniusTPSi	1.04.00.00	ABB	AddIn	Installed

xx2200000241

- 2 In the Drive modules section, browse to Arc / Application Arc / Tracking and Searching and select Fronius TPS/i WireSense Option. This will install all modules and configuration files needed for WireSense.



xx2200000242

- 3 A user log entry is generated upon installation of the WireSense software package.

#### 111850: Fronius TPSi WireSense initialized

##### Description

WireSense Version: 1.0.0 successfully installed in Task T\_ROB1

xx2200000243

**This page is intentionally left blank**

## 3 System parameters

### 3.1 WireSense Configuration

#### I/O Mapping

WireSense I/O connections are configured in the process configuration database (PROC). Actual I/O assignments to real I/O boards are made by the WireSense installation if the Fronius TPS/I welder Add-In is used. In case the Standard IoWelder interface is used, these definitions must be added to the EIO configuration database by the user or system designer. All related configuration files are loaded by the WireSense installation into the appropriate motion tasks.

#### procWSenseSet\_X.cfg file

The `procWSenseSet_X.cfg` files load default references to WireSense speed and signal configuration groupings and `procWSenseSpd.cfg` files for up to 4 motion tasks, where the 'X' represents task numbers 1-4.

#### procWSenseSpd.cfg file

The `procWSenseSpd.cfg` file loads default search speeds into all applicable motion tasks.

#### Override defaults

The user may override the defaults by replacing the entries with new entries. Below is the default file loaded by WireSense in combination with the Fronius TPS/I welder interface:

```
PROC:CFG_1.0::
#
WIRESENSE_SETTINGS:
-name "T_ROB1" \
-uses_signals "WireSenseSig" \
-uses_speeds "WireSenseSpeed"
#
WIRESENSE_SIGNALS:
-name "WireSenseSig" \
-ws_detect_input "diFrlTouchSense" \
-ws_on_output "doFrlWireSenseStart" \
-ws_active_input "diFrlArcStable" \
-ws_height_input "aiFrlWireSensePosition" \
-ws_height_output "aoFrlWireSenseEdgeHight" \
-ws_break_output "doFrlWireSenseBreak" \
-ws_retract_wire "doFrlFeedRetract"
#
WIRESENSE_ERR_HNDL_IO:
-ws_go_Dialog "" \
-ws_do_Dialog "" \
-ws_di_Ack "" \
-ws_gi_Response "" \
-ws_go_ErrType ""\
```

*Continues on next page*

### 3 System parameters

---

#### 3.1 WireSense Configuration

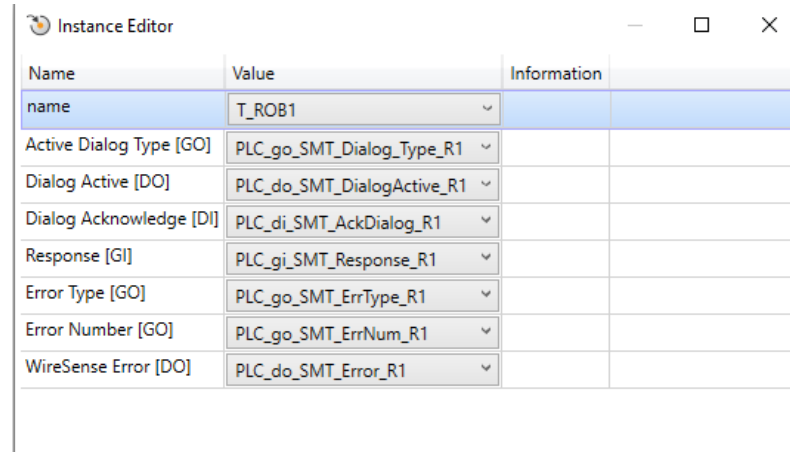
*Continued*

```
-ws_go_ErrNum "" \  
-ws_do_Error ""
```

### 3.2 WireSense ErrorHandler IO

#### Type WIRESENSE\_ERR\_HNDL\_IO

Here you can configure the EIO Signals that communicate with a PLC to indicate an error during search with the WireSense instructions.



Name	Value	Information
name	T_ROB1	
Active Dialog Type [GO]	PLC_go_SMT_Dialog_Type_R1	
Dialog Active [DO]	PLC_do_SMT_DialogActive_R1	
Dialog Acknowledge [DI]	PLC_di_SMT_AckDialog_R1	
Response [GI]	PLC_gi_SMT_Response_R1	
Error Type [GO]	PLC_go_SMT_ErrType_R1	
Error Number [GO]	PLC_go_SMT_ErrNum_R1	
WireSense Error [DO]	PLC_do_SMT_Error_R1	

xx220000245

For more information, see [Search Error Recovery I/O Interface on page 43](#).

## 3 System parameters

---

### 3.3 WireSense Settings

### 3.3 WireSense Settings

---

#### Type WIRESENSE\_SETTINGS

The following parameters are available:

Parameter name	Type	Description
<i>Uses Signals</i>	string	Reference to <i>WireSense Standard Signals</i> type. The configured signals will be used.
Uses Speeds	string	Reference to <i>WireSense Speeds</i> type. The configured speed values will be used.
Use EIO Interface	boolean	Parameter will enable (TRUE) or disable (FALSE) the EIO Interface used to communicate with the PLC. Default: FALSE
Default Sense Height	num	Value in mm at which height the WireSense function will react on the edge detection. Fronius parameter (edge detection wiresense). Default value: 1.0 Min value: 0.5 Max value 20.0
Retract Wire	num	Value in seconds. Time to feed wire backwards after the search instruction ended. Any value >0 will enable the function. Default value: 0 Min value: 0 Max value 3

### 3.4 WireSense Speeds

#### Type WIRESENSE\_speeds

The following parameters are available:

Parameter name	Type	Description
Name	string	Name of the speed instance. It is recommended to use the default name.
Main Search Speed	num	Default search speed used with the <code>SearchEdge</code> instruction (in mm/s). Default value: 25 Min value: 1.0 Max value: 80.0
Contour Search Speed	num	Default search speed used with the <code>SenseL</code> instruction (in mm/s). Default value: 10 Min value: 1.0 Max value: 25.0

### 3 System parameters

---

#### 3.5 WireSense Standard Signals

#### 3.5 WireSense Standard Signals

---

##### Parameter WIRESENSE\_SIGNALS

The following parameters are available:

Parameter name	Type	Description
Name	string	Name of the signals instance. It is recommended to use the default name.
WireSense Detection Input	Digital input	Fronius Touch Signal (Bit 7)
WireSense On Output	Digital output	Fronius WireSense Start (Bit 29)
WireSense Active Input	Digital input	Fronius Arc Stable (Bit 5)
WireSense Height Input	Analog input	Fronius Wire position (Bit 256 – 271)
WireSense Height Output	Analog output	Fronius WireSense Edge Detection (Bit 256 – 271)
WireSense Break Output	Digital output	Fronius WireSense Break (Bit 30)
Wire Retract Output	Digital output	Fronius Wire Retract (Bit 10)



## 4 Using SearchEdge with WireSense

### 4.1 Basic example

---

#### General

WireSense is a function implemented in the Fronius TPS/I welder. The wire electrode is turned into a sensor that checks the component position before each weld.

SearchEdge can compensate inconsistent weld joints by detecting actual sheet heights and positions and offset the programmed points in a weld program.

The welding torch scans the component with a reversing wire movement and the welding system sends the height information and the edge position to the robot.

For example, if a lap joint is being welded, the edge position can be precisely defined, and the system can react to any deviations. The robot adjusts the weld seam process based on an application-specific program.

By evaluating the height information, the robot can determine both the course of the edge and the actual edge height. It is also possible to determine the exact air gap between the sheets.

*Continues on next page*

## 4 Using SearchEdge with WireSense

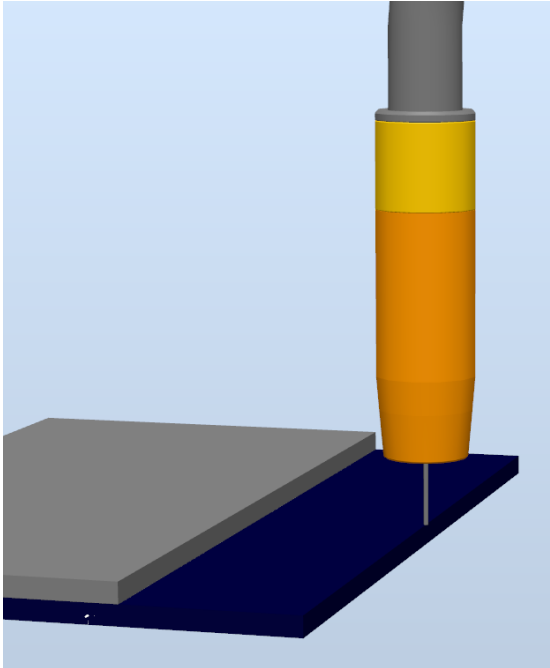
### 4.1 Basic example

*Continued*

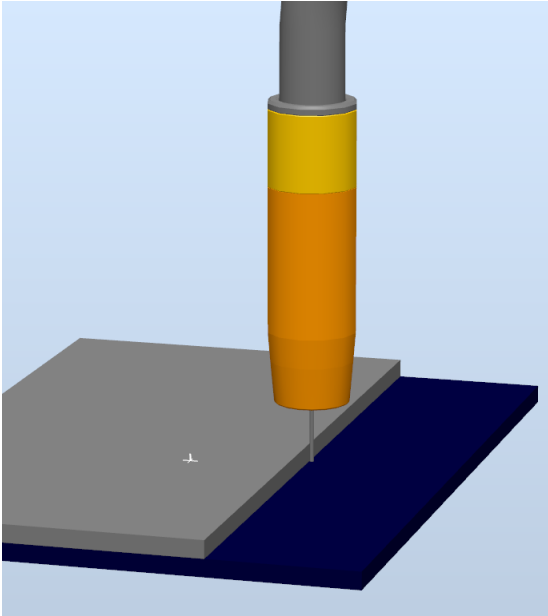
#### Basic example

In this example a simple overlap joint with 5 mm sheets is used to detect the edge. Teach the start point and search point as shown in the table below. It is recommended (but not mandatory) to align the welding gun perpendicular to the surface. The `SearchEdge` instruction can compensate the torch angle if it is not possible to align the welding torch perpendicular to the surface. Teach the search point on the edge of the upper sheet so that the wire touches the edge. Make sure the wire is not on the upper edge but on the lower edge.

When teaching the positions make sure to maintain the same height (Z) for the robotarget, otherwise wrong height information will be calculated.

Description	Position
<p>StartPoint Teach the position somewhere close to the edge to keep the search movement short.</p>	 <p>xx2200000246</p>

*Continues on next page*

Description	Position
<p>SearchPoint Teach the search position on the edge. Make sure the wire is not on the upper edge. Position must have the same stick-out as the start point.</p>	 <p>xx2200000247</p>

#### RAPID example

```

PROC rDoEdgeSearch()
  MoveL pApproach, v1000, z50, tWeldGun\WObj:=wobj0;
  SearchEdge peResult, Height, pSearchStart, pSearchEnd, v1000,
    tWeldGun\WObj:=wobj0;
  MoveL pApproach, v1000, z50, tWeldGun\WObj:=wobj0;
ENDPROC

```

#### What happens?

When `SearchEdge` is executed the `WireSense` function is activated and the wire is pulsed with a 100Hz frequency. Since the optional argument [`\SenseHeight`] is not used a default sense height of 1 mm (as configured in the configuration database) is used.

The robot will now do a search move towards the `SearchPoint`. With the default sense height all changes in the stick-out more than 1 mm are detected by `WireSense`.

The search distance (search vector) is twice the distance between the start point and search point. The robot will not stop with the edge detection trigger from the welder as the height measurement need to be done on the upper plate. This is helpful when the gun is not aligned perpendicular to the surface.

Use the optional argument [`\SearchEnd`] to stop the search move at a defined position or to avoid any collision, for example, with a part feature or tooling.

In this example, `Height` will have a value of 5 (mm).

If there would be, for example, a 2 mm gap between the lower and upper sheet, the height value would be 7 (5mm sheet size + 2 mm gap).

*Continues on next page*

## 4 Using SearchEdge with WireSense

---

### 4.1 Basic example

*Continued*

The displacement value ( $p_{ose}$ ) for the search point is returned in the variable `peResult`. If the upper sheet is not moved away from the search point, then the value should always be within the specified tolerances, see [Technical details Edge detection Fronius WireSense on page 12](#).

---

#### RAPID example with optional argument [`\SearchEnd`]

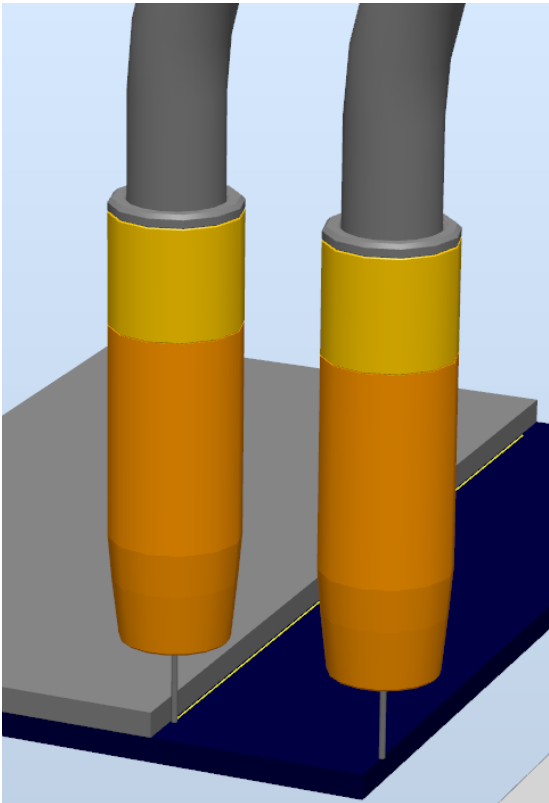
```
PROC rDoEdgeSearch()  
  MoveL pApproach, v1000, z50, tWeldGun;  
  SearchEdge peResult,  
    Height,pSearchStart,pEdge\SearchEnd:=pSearchEnd,v1000,tWeldGun;  
  MoveL pApproach, v1000, z50,tWeldGun\WObj:=wobj0;  
ENDPROC
```

4.2 Program displacement

Program displacement

In this example a simple overlap joint with 5 mm sheets is used to detect the edge but now two search movements are used to search at the beginning and end of the plate. Teach the start point and search point as shown in the table below. It is recommended (but not mandatory) to align the welding torch perpendicular to the surface. Teach the search point on the edge of the upper sheet so that the wire touches the edge. Make sure the wire is not on the upper edge but on the lower edge.

When teaching the positions make sure you keep the wire stick-out on a constant level otherwise wrong height information will be calculated.

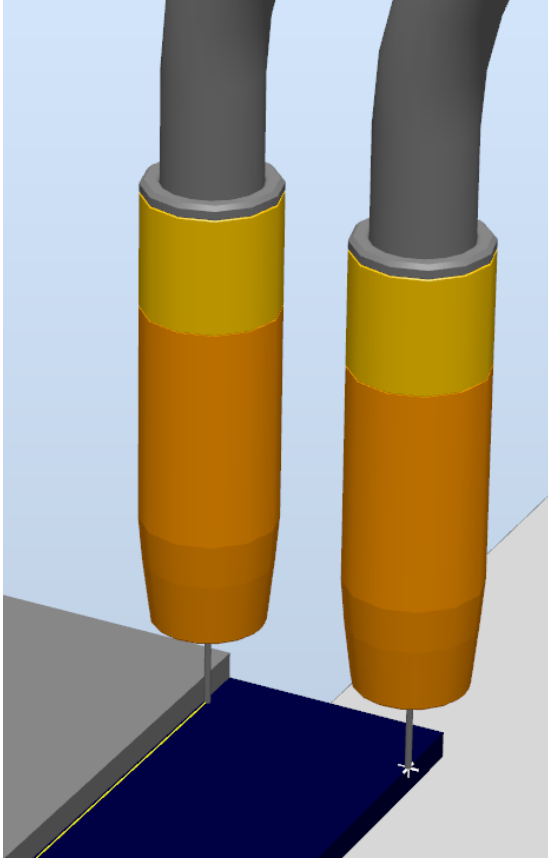
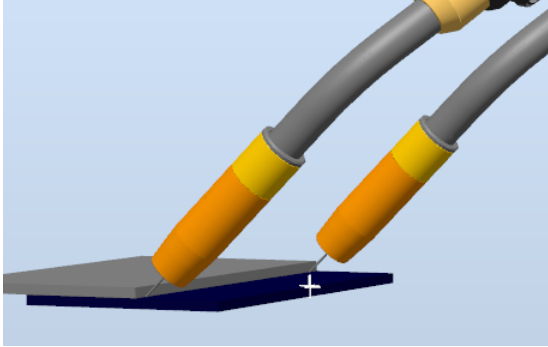
Description	Positions
<p>Startpoint and Endpoint for SearchEdge at the beginning of the plate Position must have the same stick-out</p>	 <p>xx2200000248</p>

*Continues on next page*

## 4 Using SearchEdge with WireSense

### 4.2 Program displacement

Continued

Description	Positions
<p>Startpoint and Endpoint for SearchEdge at the end of the plate Position must have the same stick-out</p>	 <p>xx2200000249</p>
<p>Create a path along the edge. This could be either MoveL instructions, or in this case Arc instructions</p>	 <p>xx2200000250</p>

#### RAPID example

```
PROC rDoEdgeSearch()  
  MoveL pApproach_1,v1000,z100,tWeldGun\WObj:=wobj0;  
  ! Search the first position  
  SearchEdge peResult_Start, Height, pStart_1, pSearch_1, v1000,  
    tWeldGun\SchSpeed:=10;  
  MoveL pApproach_2, v1000, z100, tWeldGun\WObj:=wobj0;  
  ! Search the second position  
  SearchEdge peResult_End, Height, pStart_2, pSearch_2, v1000,  
    tWeldGun\SchSpeed:=10;
```

Continues on next page

```
! Activate program displacement based on search result for start
  position
PDispSet peResult_Start;
ArcLStart pArcStart,v1000,sm1,wdl,fine,tWeldGun\WObj:=wobj0;
! Activate program displacement based on search result for end
  position
PDispSet peResult_End;
ArcLEnd pArcEnd,v1000,sm1,wdl,fine,tWeldGun\WObj:=wobj0;
PDispOff;
ENDPROC
```

---

#### What happens?

SearchEdge will perform a search move to find the edge at the beginning and at the end of the plates. If the upper plate is not moved the weld positions will be in its original location. If the upper plate is moved, for example, shifted 10 mm (in search direction) then the poses `peResult_Start` and `peResult_End` will be updated with the displacement. The displacement will be applied to the robot motion with the execution of `PDispSet` prior to the Arc instructions. The weld should be there again in the right location on the lower edge.

## 4 Using SearchEdge with WireSense

---

### 4.3 Using height information

### 4.3 Using height information

---

#### Height information

SearchEdge will return the height information based on feedback from the welder. The height value is the *sheet size + gap* (if there is a gap between upper and lower sheet). The height information can be used to change the welding parameters and/or travel speed prior to the welding.

---

#### RAPID example

```
PROC rDoEdgeSearch()
  VAR num sheetSize := 5;
  VAR num Gap;
  MoveL pApproach_1,v1000,z100,tWeldGun\WObj:=wobj0;
  ! Search the first position
  SearchEdge peResult_Start, Height, pStart_1, pSearch_1, v1000,
    tWeldGun\SchSpeed:=10;
  MoveL pApproach_2,v1000,z100,tWeldGun\WObj:=wobj0;
  ! Search the second position
  SearchEdge peResult_End, Height, pStart_2, pSearch_2, v1000,
    tWeldGun\SchSpeed:=10;
  Gap := Height - sheetSize;
  IF (Gap > 0.1 ) AND (Gap < 1.0) THEN
    wdl.main_arc.sched := 1;
    wdl.weld_speed := 10;
  ELSEIF (Gap >= 1.0 ) AND (Gap < 2.0) THEN
    wdl.main_arc.sched := 2;
    wdl.weld_speed := 9;
  ELSEIF (Gap >= 2.0 ) AND (Gap < 3.0) THEN
    wdl.main_arc.sched := 3;
    wdl.weld_speed := 8;
  ELSE
    ! Gap to large , check parts
    ! Stop;
  ENDIF
  ! Activate program displacement based on search result for start
  position
  PDispSet peResult_Start;
  ArcLStart pArcStart,v1000,sm1,wdl,fine,tWeldGun\WObj:=wobj0;
  ! Activate program displacement based on search result for end
  position
  PDispSet peResult_End;
  ArcLEnd pArcEnd,v1000,sm1,wdl,fine,tWeldGun\WObj:=wobj0;
  PDispOff;
ENDPROC
```

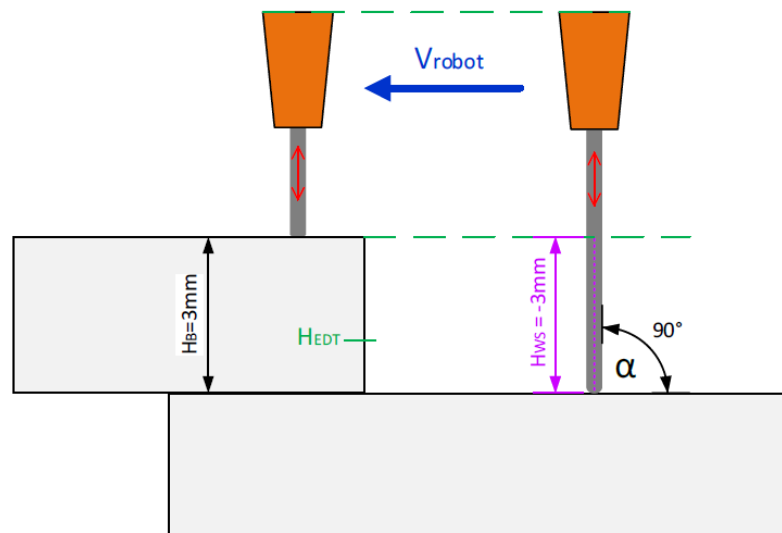


#### 4.4 Influence of the work angle of the welding torch

##### SearchEdge with a torch work angle of $90^\circ$

As the WireSense measuring processes are only carried out based on a power source internal measurement and evaluation of the wire length as it touches the base material, the output measurement result indicates the retracted or extended wire length. This length is reset at the start of the WireSense process the first time the wire touches the base material (first zero adjustment).

Using the SearchEdge instruction, the height of the edge of the sheet is measured automatically by the power source. In this case, the measurement is not taken from the first zero position but rather a positional difference is calculated based on the height value immediately before the edge and the value shortly after the edge, which is then output as the height. As a result, it is also possible to measure the edges of sheets on uneven component surfaces.



xx220000251

As shown in the previous figure, the retracted path of the wire (HWS) at a work angle of  $90^\circ$  also corresponds exactly to the height of the sheet edge to be measured. No additional factors need to be subsequently considered and the value that is output on the robot interface for the "Wire Position" variable can be used directly as a valid measured value.

*Continues on next page*

## 4 Using SearchEdge with WireSense

### 4.4 Influence of the work angle of the welding torch

Continued

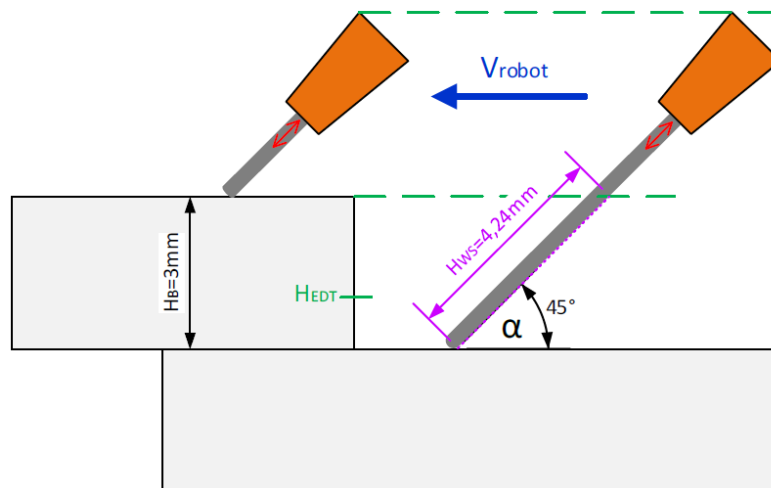
#### SearchEdge with a torch work angle of $45^\circ$

The following figure shows a measurement process with a work angle of  $45^\circ$ . The robot movement towards the edge of the sheet and the inclined welding torch position does not cause the wire to be retracted vertically upwards but along an imaginary line at an angle of  $45^\circ$ . As a result, the power source internally measured wire length (HWS), which is also output on the robot interface for the "Wire Position" variable, no longer corresponds to the real edge height and must be mathematically corrected.



#### Note

These corrections are already done using the provided SearchEdge instruction. No additional calculation is needed.

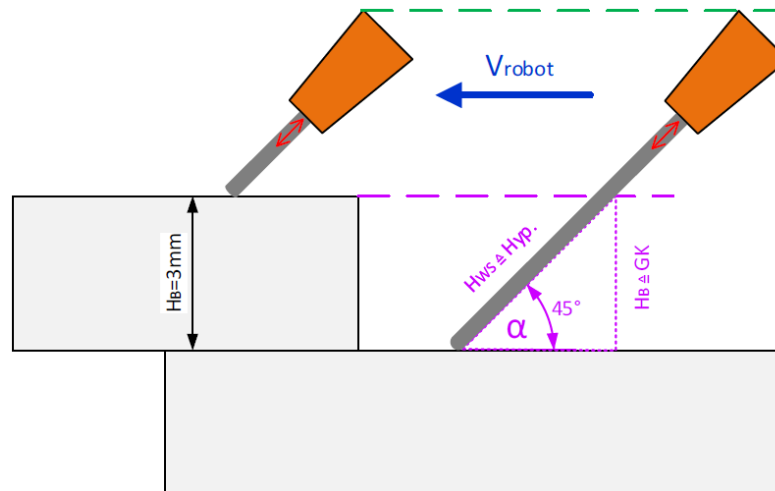


xx2200000252

As shown in the following figure, the real height of the sheet edge (HB) can be determined by means of a simple trigonometric function. Here the measured value HWS output by WireSense for the hypotenuse and the height of the sheet edge

Continues on next page

HB for the opposite side correspond to an imaginary triangle. The relationship can then be calculated based on the work angle  $\alpha$  of the welding torch.



xx2200000253

These corrections are already done using the provided `SearchEdge` instruction. No additional calculation is needed. The sheet height is automatically calculated and updated by the `SearchEdge` instruction. In addition, an updated "Edge detection value" is sent to the power source.

If for example `SearchEdge` is used on a 5 mm sheet and the edge detection value in the instruction is set to 4.5 mm a value of roughly 6.4 mm is carried out to the welder due to the  $45^\circ$  work angle.

**This page is intentionally left blank**

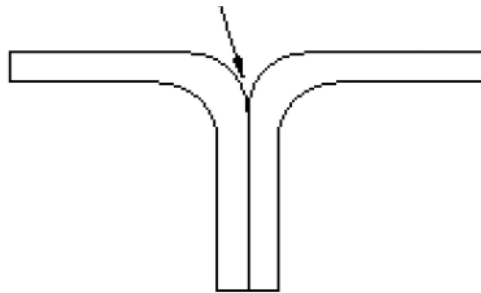
## 5 Using SenseL with WireSense

### 5.1 Basic example

#### General information

`SenseL` is an instruction used for contour sensing. Its main purpose is to scan the component and return a position (or an offset) with the longest wire stick-out which is normally the center of the groove. The instruction can, for example, be used on a flare-V groove (commonly used to join two rounded or curved parts) or v groove.

Example of a double flare-V groove:



xx2200000254

The robot path and stick-out length is internally stored when the `SenseL` is executed. The position is stored every 0.25-0.3 mm by default (this can be changed within 0.25-1.5 mm using the optional argument [`\Resolution`]).

It is recommended to not exceed a maximum search time for 30 seconds and to not use a fast search speed as this will have an impact on precision. The default search speed is 10 mm/s. To avoid bending the wire once the search is ended, when moving to the next scan or `ArcStart`, a retract function can be activated by adding a time in the configuration (topic *Process*, type *WireSense Settings*, parameter *Retract Wire*). Make sure the digital output is connected in WireSense Standard Signals.

*Continues on next page*

## 5 Using SenseL with WireSense

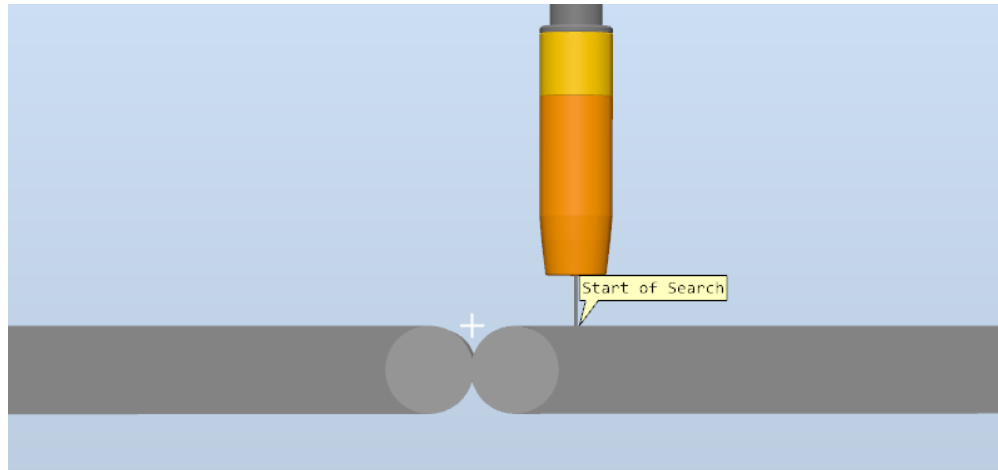
### 5.1 Basic example

*Continued*

#### Basic example

In this example a simple double flare v joint is used to detect the center of the groove. Teach the start point and search point as shown in the table below. It is mandatory to align the welding gun perpendicular to the surface. The Stick-Out length will not be compensated if the gun is not aligned. Teach the search point at the center of the groove where we expect the longest wire stick out.

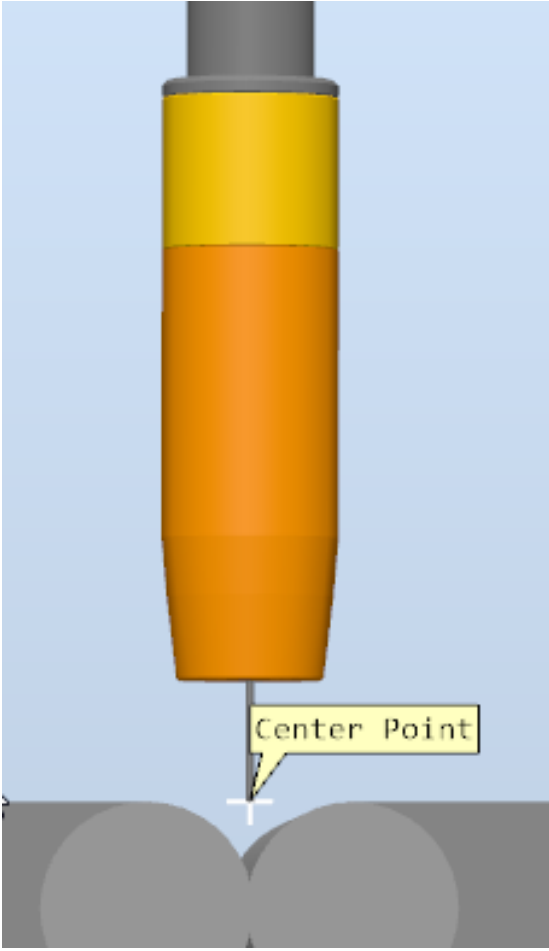
When teaching the positions make sure you keep the wire stick-out on a constant level otherwise wrong height information will be calculated.



xx2200000255

Description	Position
StartPoint Teach the position somewhere close to the edge to keep the search movement short.	 xx2200000256

*Continues on next page*

Description	Position
<p>SearchPoint (Center Point) Teach the search position at the center of the groove. Position must have the same height as the start point.</p>	 <p>xx2200000257</p>

**RAPID example**

```

PROC rContourSense()
  MoveL pApproach_1,v1000,z100,tWeldGun\WObj:=wobj0;
  ! Search the first position
  SenseL peResult_Start, pStart_1, pGroove_1, v1000,
    tWeldGun,"Seam1_Start";
  MoveL pApproach_2,v1000,z100,tWeldGun\WObj:=wobj0;
  ! Search the second position
  SenseL peResult_End, pStart_2, pGroove_2, v1000,
    tWeldGun,"Seam1_End";
  MoveL pApproach_2,v1000,z100,tWeldGun\WObj:=wobj0;
  PDispSet peResult_Start ;
  ArcLStart pArcStart,v1000,sm1,wd1,fine,tWeldGun\WObj:=wobj0;
  PDispSet peResult_End ;
  ArcLEnd pArcEnd,v1000,sm1,wd1,fine,tWeldGun\WObj:=wobj0;
  PDispOff;
ENDPROC

```

*Continues on next page*

## 5 Using SenseL with WireSense

---

### 5.1 Basic example

*Continued*

---

#### What happens?

SenseL will perform a search move to find the position with the longest stick out. The search vector is double the distance between the start point of the search and the search point. While searching is active the positions and stick-out length (feedback from welder) is stored internally. The displacement pose `Result` is then calculated from the position with the longest stick out and the applied search point. The displacement will be applied to the robot motion with the execution of `PDispSet` prior to the arc instructions. The weld should be then again in the right location in the center of the groove.

SenseL additionally provides a `robtarg` that can be used after searching in any `Arc` or `Move` instructions instead of activating a program displacement. This `robtarg` will have the same orientation [`rot`] as the search start point.

The optional argument [`\SearchResult`] must be used. Since this position will be updated by SenseL, it must be defined as `PERS` (persistent).

---

#### RAPID example

```
PROC rContourSense()  
  MoveL pApproach_1, v1000, z100, tWeldGun\WObj:=wobj0;  
  ! Search the first position  
  SenseL peResult_Start, pStart_1,  
        pGroove_1\SearchResult:=pArcStart, v1000,  
        tWeldGun,"Seam1_Start";  
  MoveL pApproach_2, v1000, z100, tWeldGun\WObj:=wobj0;  
  ! Search the second position  
  SenseL peResult_End, pStart_2, pGroove_2\SearchResult:=pArcEnd,  
        v1000, tWeldGun,"Seam1_End";  
  MoveL pApproach_2, v1000, z100, tWeldGun\WObj:=wobj0;  
  ArcLStart pArcStart, v1000, sm1, wd1, fine, tWeldGun\WObj:=wobj0;  
  ArcLEnd pArcEnd, v1000, sm1, wd1, fine, tWeldGun\WObj:=wobj0;  
ENDPROC
```



## 5.2 Search distance with SenseL

### General information

The search vector is calculated between `StartPoint` and `SearchPoint`. The robot will continue past the search point for a total search distance described by twice the distance between `StartPoint` and `SearchPoint`.

If for some reason this calculated distance is too long, for example, there is a risk of a collision with the tooling/jig or the part, then the optional argument `[\SearchEnd]` can be used to define the end of the search.

### RAPID example

```
PROC rContourSense()
  MoveL pApproach_1, v1000, z100, tWeldGun\WObj:=wobj0;
  ! Search the first position
  SenseL peResult_Start, pStart_1,
    pGroove_1\SearchEnd:=pSearchEnd_1, v1000,
    tWeld-Gun, "Seam1_Start";
  MoveL pApproach_2, v1000, z100, tWeldGun\WObj:=wobj0;
  ! Search the second position
  SenseL peResult_End, pStart_2, pGroove\SearchEnd:=pSearchEnd_2,
    v1000, tWeldGun, "Seam1_End";
  MoveL pApproach_2, v1000, z100, tWeldGun\WObj:=wobj0;
  ArcLStart pArcStart, v1000, sm1, wd1, fine, tWeldGun\WObj:=wobj0;
  ArcLEnd pArcEnd, v1000, sm1, wd1, fine, tWeldGun\WObj:=wobj0;
  PDispSet peResult_Start ;
  ArcLStart pArcStart, v1000, sm1, wd1, fine, tWeldGun\WObj:=wobj0;
  PDispSet peResult_End ;
  ArcLEnd pArcEnd, v1000, sm1, wd1, fine, tWeldGun\WObj:=wobj0;
  PDispOff;
ENDPROC
```

**This page is intentionally left blank**

## 6 Search Error Recovery I/O Interface

### 6.1 Introduction to Search Error Recovery I/O Interface

#### General information

The function package provides an I/O based interface to communicate with an external device, mainly a PLC, to indicate an active user dialog on the FlexPendant that need attention and remote control that user dialog. The I/O interface supports all search instructions provided by the function package.

The Error Recovery I/O interface behave like RobotWare Arc *Weld Error Recovery I/O interface* as it follows the same concept. If RobotWare Arc *Weld Error Recovery I/O interface* is configured, the same signals can then be configured to remote control the user interface for the search instructions.



#### Note

The Bit Mapping (length) for the group outputs/inputs might be changed if RobotWare Arc *Weld Error Recovery I/O interface* and *Search Error Recovery I/O interface* are used.

Additional information for the *Weld Error Recovery I/O Interface* can be found in *Application manual - Arc and Arc Sensor*, section *Weld Error Recovery interface*.

The internal error handling can be switched off if needed. All search related errors must be handled on user level by adding an error handler.



#### Note

If the internal error handler is switched off, the Search Error I/O interface cannot be used.

#### Usage

The Search 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.

*Continues on next page*

## 6 Search Error Recovery I/O Interface

### 6.1 Introduction to Search Error Recovery I/O Interface

*Continued*

#### Architecture

All I/O signals used with the Search Error Recovery I/O interface must be configured. In a MultiMove system, each welding robot will have its own Search 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: *wi\_di\_Ack\_X*, where x specifies the number of the welding robot. 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.

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

Signal common name	Signal definition name	Description
<i>Application Error</i>	<i>ws_do_Error_X</i>	Indicates a general WireSense error. This output can be used if the internal error handler is switched off. It will work independent from the I/O interface. Type: Digital Output
<i>Prompt Acknowledge</i>	<i>ws_di_Ack_X</i>	Allows the remote device to acknowledge a Weld Error Recovery prompt. Type: Digital Input
<i>Dialog Active</i>	<i>ws_do_Dialog_X</i>	Indicates to a remote device that a dialog is active and awaiting a response. Type: Digital Output
<i>Active Dialog Type</i>	<i>ws_go_Dialog_X</i>	Indicates to the remote device that the Dialog Type prompt is active (Type 13 to Type 14). Type 1 to 8 are reserved/used by RobotWare Arc Errorhandler IO interface. Type 9 to 12 are reserved/used by the SmarTac Errorhandler IO interface. Type: Group Output
<i>Response</i>	<i>ws_gi_Response_X</i>	Allows the remote device to communicate a response. The context of the response is dictated by the active dialog type. <ul style="list-style-type: none"> <li>• Active Dialog 13 <ul style="list-style-type: none"> <li>1 Retry</li> <li>2 Return</li> <li>3 Abort (Raise)</li> </ul> </li> <li>• Active Dialog 14 <ul style="list-style-type: none"> <li>1 Return</li> <li>2 Abort (Raise)</li> </ul> </li> </ul> Type: Group Input
<i>Error Type</i>	<i>ws_go_ErrType_X</i>	Indicates to the remote device the WireSense error type. Valid output data range: 40-43 <ul style="list-style-type: none"> <li>• 0 = No active error type</li> <li>• (40) WSENSE_ACT_ERR</li> <li>• (41) WSENSE_SENON_ERR</li> <li>• (42) WSENSE_LIMIT</li> <li>• (43) ERR_WHLSEARCH</li> </ul> Type: Group Output 6 bit

*Continues on next page*

## 6 Search Error Recovery I/O Interface

### 6.1 Introduction to Search Error Recovery I/O Interface

*Continued*

Signal common name	Signal definition name	Description
<i>Error Number</i>	<i>ws_go_ErrNum_X</i>	Not yet used since WireSense does not create specific error codes. Only a <i>WireSense Search Override warning</i> is logged. Type: Group Output

#### Sequence

The I/O sequence is as follows:

- 1 A search error occurs triggering a Search Error Recovery prompt to be displayed. Search Error Recovery will set *ws\_do\_Dialog\_X* high to indicate an active prompt. Search Error Recovery will also set *ws\_go\_Dialog\_X* to indicate the type of prompt. If the prompt is an error type, an error type will be supplied on group outputs *ws\_go\_ErrType\_X*.
- 2 The remote device interprets the information. If the dialog prompt type requires a numeric response, the remote device supplies the value on *ws\_gi\_Response\_X*.
- 3 The remote device acknowledges the prompt by pulsing the *ws\_di\_Ack\_X* signal. Search Error Recovery responds by closing the prompt on the FlexPendant.
- 4 The Weld Error Recovery I/O interface will be inoperable until the *ws\_di\_Ack\_X* signal is reset.

A warning will be written in the user log if *ws\_diAck\_X* was active before the User Dialog was active. In such a case the group outputs *ws\_go\_ErrType\_X* and *ws\_go\_Dialog\_X* remain 0. The output *ws\_do\_Dialog\_X* is still set to indicate a necessary user action on the FlexPendant.

#### Active Dialog Type

There are currently 2 possible dialog prompts from Search Error Recovery. When one of the 2 prompts are active, the digital output *ws\_do\_Dialog\_X* will be high. The prompts require a numeric response from *ws\_gi\_Response\_X* followed by an acknowledgment from *ws\_di\_Ack\_X*.

Active Dialog Type	User dialog
1-8	Used with RobotWare Arc Weld Error Recovery
9-12	Used with SmarTac Recovery Interface
13-14	Used with WireSense Recovery Interface

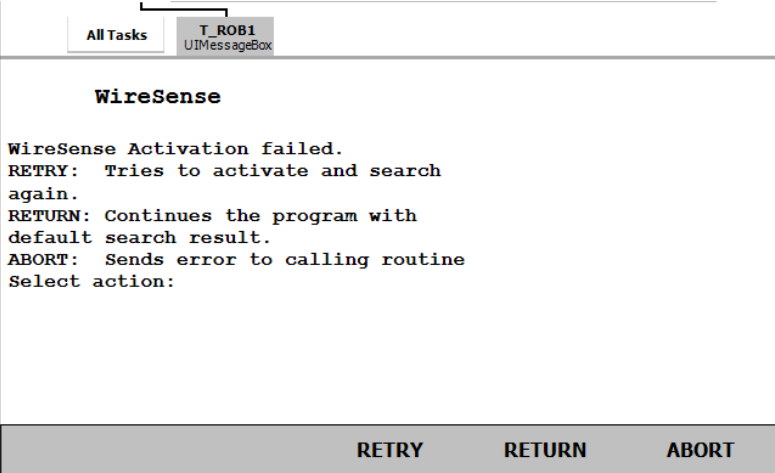
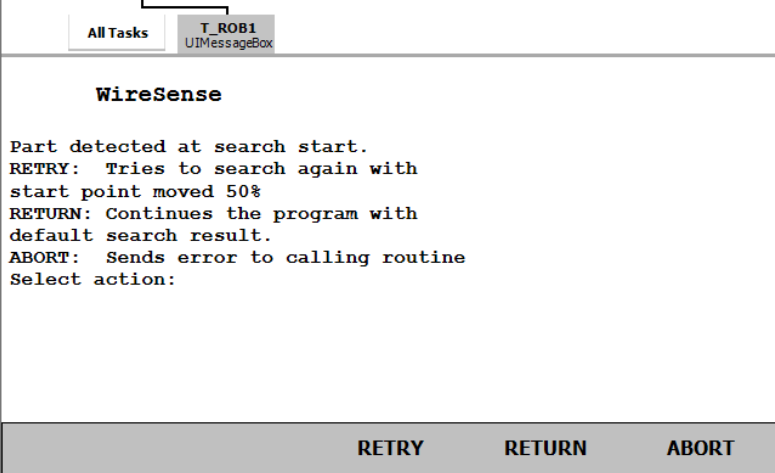
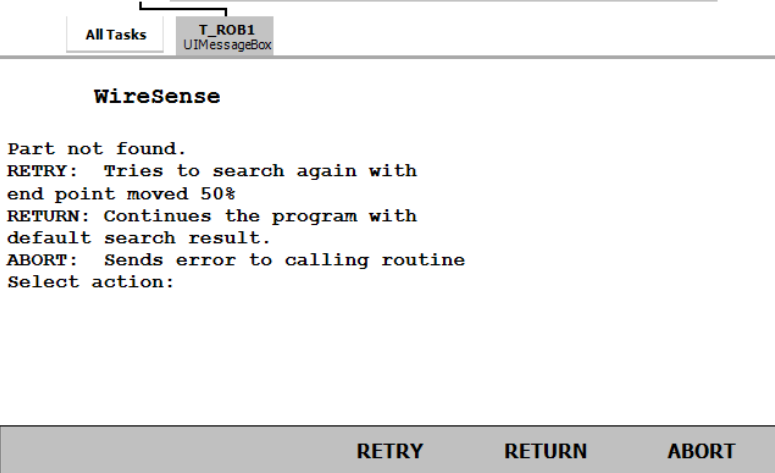
*Continues on next page*

## 6 Search Error Recovery I/O Interface

### 6.1 Introduction to Search Error Recovery I/O Interface

*Continued*

#### Dialog Type 13

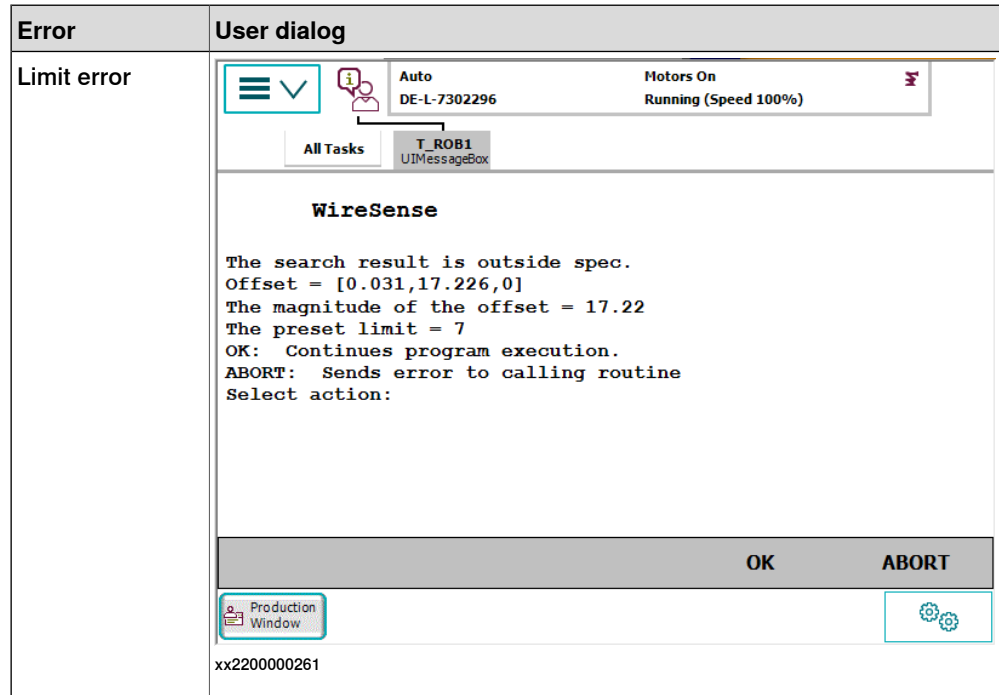
Error	User dialog
Activation failed	 <p>xx2200000258</p>
Part found prior Search	 <p>xx2200000259</p>
Error while Search	 <p>xx2200000260</p>

*Continues on next page*

When one of the above shown dialogs is active, the signal *ws\_do\_Dialog\_X* will be high and *ws\_go\_Dialog\_X* will be set to 13. The remote device may respond to the dialog by setting *ws\_gi\_Response\_X* to a value from the list below, followed by pulsing *ws\_di\_Ack\_X*.

- 1 Retry
- 2 Return
- 3 Abort (Raise)

#### Dialog Type 14



When the above shown dialog is active, the signal *ws\_do\_Dialog\_X* will be high and *ws\_go\_Dialog\_X* will be set to 14. The remote device may respond to the dialog by setting *ws\_gi\_Response\_X* to a value from the list below, followed by pulsing *wi\_di\_Ack\_X*.

- 1 OK
- 2 Abort (Raise)

#### Error Type

The Error Type will be sent on *ws\_go\_ErrType\_X*. The following is a list of possible error types from WireSense.

ERRNO	Description	Error Type
WSENSE_ACT_ERR	WireSense activation error	40
WSENSE_SEN-ON_ERR	Part detected prior search	41
WSENSE_LIMIT	Limit Error (Seach_1D)	42
ERR_WHLSEARCH	Error during search	43
WSENSE_PATHPOS	Error while storing path	44

*Continues on next page*

## 6 Search Error Recovery I/O Interface

---

### 6.1 Introduction to Search Error Recovery I/O Interface

*Continued*

---

#### Error Number

The group output *ws\_go\_ErrNum\_X* is not yet used and reserved for future use. In the current version, WireSense does not provide any error codes related to search errors. The output will be set to 0.



## 6.2 Configure search error recovery I/O interface

### Description

Search Error Handler I/O configures the Search Error Recovery I/O part of Search Error Recovery feature in the WireSense function package. The system parameters can be viewed in RobotStudio, in the **Configuration Editor**, in the topic *Process*, type *Arc Error Handler I/O*.

In order to use the Search Error Handler I/O interface, the parameter *Use EIO Interface* in the *WireSense Settings* must be set to active.

### Examples

The default configuration has the following definition and can be found in the topic *Process*, type *WireSense Errorhandler IO*.

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

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

OK Cancel

xx220000262

### Parameters

Parameter	Description	Data Type
<i>Name</i>	The name of the instance WIRE-SENSE_ERR_HNDL_IO. Must be (T_ROB1-T_ROB4).	typeStringNormal
<i>ws_go_Dialog</i>	The signal name for <i>Active Dialog Type</i> .	go
<i>ws_do_Dialog</i>	The signal name for <i>Dialog Active</i> .	do
<i>ws_di_Ack</i>	The signal name for <i>Prompt Acknowledge</i> .	di

*Continues on next page*

## 6 Search Error Recovery I/O Interface

### 6.2 Configure search error recovery I/O interface

*Continued*

Parameter	Description	Data Type
ws_gi_Response	The signal name for <i>Response</i> .	gi
ws_go_ErrType	The signal name for <i>Error Type</i> .	go
ws_go_ErrNum	The signal name for <i>Error Number</i> .	go
ws_do_Error	The signal name for general error.	do

To activate the Search Error handler IO interface, set *Use EIO Interface* to *TRUE*. The parameter belongs to the type *WireSense Settings*, in the topic *Motion*.

Name	Value	Information
Name	T_ROB1	
Uses Signals	WireSenseSig	
Uses Speeds	WireSenseSpeed	
Use EIO Interface	<input checked="" type="radio"/> TRUE <input type="radio"/> FALSE	
Disable ErrorHandler	<input type="radio"/> TRUE <input checked="" type="radio"/> FALSE	
Default Sense Height	1	

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

OK Cancel

xx2200000263

## 6.3 User defined error handling

### Description

The internal error handler in WireSense can be switched off to handle all possible errors on user level. The error handler can be switched off in the process configuration database.

The parameter *Disable ErrorHandler* belongs to the type *WireSense Settings*, in the topic *Process*.

The EIO Interface will remain inactive if the internal error handler in WireSense is disabled, even if the EIO interface is configured.

Name	Value	Information
Name	T_ROB1	
Uses Signals	WireSenseSig	
Uses Speeds	WireSenseSpeed	
Use EIO Interface	<input checked="" type="radio"/> TRUE <input type="radio"/> FALSE	
Disable ErrorHandler	<input checked="" type="radio"/> TRUE <input type="radio"/> FALSE	
Default Sense Height	0,5	

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

OK Cancel

xx220000264

The output *ws\_do\_Error\_X* is set if a search error or activation error occurs. The signal is set to 0 with the next execution of a WireSense search instruction.

The following errors can be handled:

ERRNO	Description
WSENSE_ACT_ERR	WireSense activation error
WSENSE_SENON_ERR	Part detected prior search
WSENSE_LIMIT	Limit Error ( <i>SeachEdge</i> )
ERR_WHLSEARCH	Error during search

*Continues on next page*

## 6 Search Error Recovery I/O Interface

---

### 6.3 User defined error handling

*Continued*

---

#### RAPID example

```
PROC rDoEdgeSearch()  
  MoveL pApproach_1,v1000,z100,tWeldGun\WObj:=wobj0;  
  SearchEdge peResult_Start, Height, pStart_1, pSearch_1, v1000,  
    tWeldGun\SchSpeed:=20;  
  ERROR  
    IF ERRNO = WSENSE_ACT_ERR THEN  
      ! WireSense not active , handle activation error  
    ELSEIF ERRNO = WSENSE_SENON_ERR THEN  
      ! Feature detected at activation , handle error here  
    ELSEIF ERRNO = WSENSE_LIMIT THEN  
      ! Limit error while searching , handle limit error here  
    ELSEIF ERRNO = ERR_WHLSEARCH THEN  
      ! Edge not found , handle error here  
    ENDIF  
  ENDPROC
```

## 7 Production Monitoring

### Introduction

The WireSense instructions have support for the controller option *659-1 Production Monitoring*. A CSV file, `SearchEdge.csv` and respectively `SenseL.csv`, is created as specified in *Production Monitor Settings*. The default path is the temp folder in the robot system. For more information on how to use and set up Production Monitoring, see *Application manual - Production Monitor*.

### SearchEdge.csv

The following data is logged for `SearchEdge` in the file:

Column name	Data type	Description
EventID	Num (Dnum)	A number supplied by GAP Execution Engine or <i>ProcCycleStart</i> ,
CycleID	Num (Dnum)	A number supplied by GAP Execution Engine or <i>ProcCycleStart</i> .
SearchName	String	Name derived from the optional argument in the RAPID instruction <code>SearchEdge</code> .
DispMag	Num	Magnitude of displacement.
Displimit	Num	Maximum allowed displacement.
Stick-Out	Num	Stick-Out information from welder.
Search Angle	Num	Calculated search angle.
Height	Num	Calculated sheet height based on search angle and stick-out length.
SenseHeight	Num	Value sent to the power source to determine the minimum edge height that triggers the edge detection signal.
Result X	Num	The search result value after completion represented as a frame.
Result Y	Num	The search result value after completion represented as a frame.
Result Z	Num	The search result value after completion represented as a frame.
ErrorType	Num	Number indicating possible errors during action.
Duration	Num	Time in seconds to complete search.
Speed	Num	The used search speed.
UserID	String	Logged in UAS User.
RobotID	String	Task name.
ControllerID	String	The controller ID.
Time	Date Time	Current Date / Time.

*Continues on next page*

## 7 Production Monitoring

Continued

### SenseL.csv

The following data is logged for SenseL in the file:

Column name	Data type	Description
EventID	Num (Dnum)	A number supplied by GAP Execution Engine or ProcCycleStart.
CycleID	Num (Dnum)	A number supplied by GAP Execution Engine or ProcCycleStart.
SearchName	String	Name derived from the optional argument in the RAPID instruction SearchEdge.
DispMag	Num	Magnitude of displacement.
DispLimit	Num	Maximum allowed displacement.
Stick-Out	Num	Stick-Out information from welder.
Result X	Num	The search result value after completion represented as a frame.
Result Y	Num	The search result value after completion represented as a frame.
Result Z	Num	The search result value after completion represented as a frame.
ErrorType	Num	Number indicating possible errors during action.
Duration	Num	Time in seconds to complete search.
Speed	Num	The used search speed.
UserID	String	Logged in UAS User.
RobotID	String	Task name.
ControllerID	String	The controller ID.
Time	Date Time	Current Date / Time.

---

# 8 WireSense with Additional Arc Systems

### Additional arc systems

The WireSense software package can only be used with the first arc system if the robot system is configured with the option *[651-2] Two Additional Arc Systems* together with the Fronius TPS/I add-in.

If WireSense should be used with an additional arc system, the Standard I/O Welder can be used. Configure the parameter *WireSense Standard Signals* with the signals used with the additional arc system.



#### Note

WireSense can only be used and configured for one of the three possible arc systems in combination with the Standard I/O Welder. If, for some reason, more than one TPS/I welders are used the WireSense settings can be switched with the RAPID instruction `Switch-WireSenseSettings`.



#### Note

It is the responsibility of the integrator to make sure the right welding equipment and welding gun is mounted on the robot prior executing a `SearchEdge/SenseL` instruction to avoid any damage to the welding equipment or robot system.

**This page is intentionally left blank**



## 9 RAPID references

### 9.1 Instructions

#### 9.1.1 SearchEdge - One-dimensional search

##### Usage

`SearchEdge` is an instruction used for tactile searching with the *Fronius TPS/i WireSense* option. The search path is described by two required robtargets. The search result is stored as pose data in the required argument `Result`. All *WireSense* activation and deactivation is automatically handled.

##### Basic examples

The following example illustrates the instruction `SearchEdge`.

##### Example 1

```
SearchEdge peOffset,Height, p1, p2, v200, tWeldGun;
```

The robot moves on a path from `p1` through `p2`. When contact is made with the upper sheet, the difference between the contact location on the upper sheet and `p2` is stored in `peOffset`.

##### Arguments

```
SearchEdge [\StopAtEndPoint] Result Height [\SearchStop]
           [\SenseHeight] StartPoint SearchPoint [\SearchEnd] Speed Tool
           [\WObj] [\PrePDisp] [\Limit] [SchSpeed] [\WireSize]
           [\SearchName] [\TLoad]
```

`[\StopAtEndPoint]`

**Data type:** switch

The robot will stop at the end of the search move and not move back to the `StartPoint`.

`Result`

**Data type:** pose

The displacement frame that will be updated.

`Height`

**Data type:** num

Height value from the power source for the edge height measurement (in millimeter).

`[\SearchStop]`

**Data type:** robtarget

If selected this robtarget will be updated as the point where the robot detects the part feature.

`[\SenseHeight]`

**Data type:** num

*Continues on next page*

## 9 RAPID references

---

### 9.1.1 SearchEdge - One-dimensional search

*Continued*

Value in millimeter sent to the power source to determine the minimum edge height that triggers the edge detection. If not used the value configured in process configuration is used. Default is 1 mm.

StartPoint

**Data type:** `robtarget`

The starting point for the search motion.

SearchPoint

**Data type:** `robtarget`

The point where the robot expects to touch the part. This `robtarget` is programmed so that the torch is touching the surface of the part feature.

[`\SearchEnd`]

**Data type:** `robtarget`

Use the optional argument `SearchEnd` to stop the search move at a defined position, or to avoid any collision, for example, with a part feature or tooling.

Speed

**Data type:** `speeddata`

The speed data used when moving to the `StartPoint`. The velocity of the search motion is unaffected.

Tool

**Data type:** `tooldata`

The tool used during the search.

[`\Wobj`]

**Data type:** `wobjdata`

The work object used during the search. `Wobj` determines what frame `Result` will be related to. If not selected, `wobj0` is used.

[`\PrePDisp`]

**Data type:** `pose`

If selected, the search will be conducted with this displacement frame active, effectively adding the two displacement frames. This may or may not be the same as the pose data selected for `Result`.

[`\Limit`]

**Data type:** `num`

If selected, an error will be flagged if the magnitude of the search result, `Result`, is larger than the value entered for the limit (in mm).

[`SchSpeed`]

**Data type:** `num`

Search speed during execution of the search movements. If not used the default speed configured in the process configuration is used.

*Continues on next page*

`[\WireSize]`

**Data type:** num

Not yet implemented. (Half of the wire size will be subtracted from the found edge as the search movement stops on the upper sheet.)

`SearchName`

**Data type:** string

The search will be assigned this identifying name. The name will accompany any error messages that are written to the event log.

`[\TLoad]`

**Data type:** loaddata

The argument `\TLoad` 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 argument `\TLoad` is used, then the `loaddata` in the current `tooldata` is not considered.

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

---

### Program execution

When executed, the robot makes a linear movement to the start point, `StartPoint`. `WireSense` is activated and motion starts towards the search point, `SearchPoint`. The robot will continue past the search point for a total search distance described by twice the distance between `StartPoint` and `SearchPoint`. Once the part feature is sensed the displacement data, `Result`, is stored. This program displacement can later be used to shift programmed points using the RAPID instruction `PDispSet`. In addition, the height information is stored in `height`

---

### Error handling

Fault	Menu message
Fault 1	Activation of WireSense failed
Fault 2	Search Failed
Fault 3	Part found prior search

#### Fault 1

If an error occurs when activating `WireSense`, a menu will appear with the following prompts:

#### Activation of WireSense failed

RETRY	Tries to search again with start point moved 50%
RETURN	Continues the program with default search result
ABORT	Sends error to calling routine

When `RETRY` is selected the start point of the search is shifted further from the part feature. This may give a good search result in cases where the part feature is

*Continues on next page*

## 9 RAPID references

---

### 9.1.1 SearchEdge - One-dimensional search

*Continued*

unusually close, and the wire is touching the part feature at the beginning of a normal search. When RETURN is selected a default search result is used which will include any pre-offset included in the search instruction. A message will be logged in the event log.

#### Fault 2

If an error occurs during the search process, a menu will appear with the following prompts:

##### Search failed

RETRY	Tries to search again with start point moved 50%
RETURN	Continues the program with default search result
ABORT	Sends error to calling routine

When RETRY is selected the start point of the search is shifted further from the part feature. This may give a good search result in cases where the part feature is unusually close, and the wire is touching the part feature at the beginning of a normal search. When RETURN is selected a default search result is used which will include any pre-offset included in the search instruction. A message will be logged in the event log.

#### Fault 3

If for some reason the welder feedback that the edge is found before search begins, the following menu appears:

##### Part detected at Search Start

RETRY	Tries to search again with start point moved 50%
RETURN	Continues the program with default search result
ABORT	Sends error to calling routine

When RETRY is selected the start point of the search is shifted further from the part feature. This may give a good search result in cases where the part feature is unusually close, and the wire is touching the part feature at the beginning of a normal search. When RETURN is selected a default search result is used which will include any pre-offset included in the search instruction. A message will be logged in the event log.

If the optional argument `Limit` is selected and the magnitude of `peResult` is larger than the value entered for the limit, the following message appears:

##### The search result is outside spec.

Offset:=	[12.012,3.002,-5.013]
The magnitude of the offset :=	13.34
The present limit :=	10
OK	Continue with program execution
RAISE	Sends the error to calling routine

When OK is selected the search result is accepted regardless of magnitude. A message will be logged in the event log.

*Continues on next page*

**Syntax**

SearchEdge

```
[ '\ ' StopAtEndPoint ', ' ]
[ Result ':=' ] < expression (INOUT) of pose > ', '
[ Height ':=' < expression (INOUT) of num > ]
[ '\ ' SearchStop ':=' < expression (INOUT) of robtarg > ', '
[ '\ ' SenseHeight ':=' < expression (IN) of num > ', '
[ StartPoint ':=' ] < expression (IN) of robtarg > ', '
[ SearchPoint ':=' ] < expression (IN) of robtarg > ', '
[ '\ ' SearchEnd ':=' < expression (INOUT) of robtarg > ', '
[ Speed ':=' ] < expression (IN) of speeddata > ', '
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\ ' WObj ':=' < persistent (PERS) of wobjdata > ]
[ '\ ' PrePDisp ':=' < expression (IN) of pose > ]
[ '\ ' Limit ':=' < expression (IN) of num > ]
[ '\ ' SchSpeed ':=' < expression (IN) of num > ]
[ '\ ' WireSize ':=' < expression (IN) of num > ]
[ '\ ' SearchName ':=' < expression (IN) of string > ]
[ '\ ' TLoad ':=' ] < persistent (PERS) of loaddata > ] ';'

```

**Related information**

For information about	See
PDispSet	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>

## 9 RAPID references

---

### 9.1.2 SenseL - Contour sensing

#### 9.1.2 SenseL - Contour sensing

---

##### Usage

`SenseL` is an instruction used for tactile searching with the *Fronius TPS/i WireSense* option. The search path is described by two required robtargets. The search result is stored as pose data in the required argument `Result`. All *WireSense* activation and deactivation is automatically handled. The instruction can be used to search for a position (robtarget) along a path that has either the longest stick-out (usually at the center of the V-joint) or the shortest stick-out (outer edge of an inverted V-joint).

---

##### Basic examples

The following example illustrates the instruction `SenseL`.

##### Example 1

```
SenseL peOffset, p1, p2, v200, tWeldGun, "Seam1"
```

The robot moves on a path from `p1` through `p2`. When contact is made with the upper sheet, the difference between the contact location on the upper sheet and `p2` is stored in `peOffset`.

---

##### Arguments

```
SenseL Result SearchStart SearchPoint [\SearchEnd] [\SearchResult]  
      Speed Tool [\WObj] [\PrePDisp] [\Limit] [\Resolution]  
      [SchSpeed] [\ShortStickOut] SearchName [\TLoad]
```

##### Result

**Data type:** pose

The displacement frame that will be updated.

##### SearchStart

**Data type:** robtarget

The starting point for the search motion.

##### SearchPoint

**Data type:** robtarget

The point where the robot expects the position with the longest stick out. This robtarget is programmed in the center of the groove.

##### [\SearchEnd]

**Data type:** robtarget

If selected the robot movement will stop at the `SearchEnd` position. Otherwise the movement is two times the distance between the `SearchStart` and `SearchPoint`.

##### [\SearchResult]

**Data type:** robtarget

If selected this robtarget will be updated as the point where the robot detects the longest stick out (or the shortest if the optional argument `[\ShortStickOut]` is used).

*Continues on next page*

Speed

**Data type:** speeddata

The speed data used when moving to the SearchStart. The velocity of the search motion is unaffected.

Tool

**Data type:** tooldata

The tool used during the search.

[\WObj]

**Data type:** wobjdata

The work object used during the search. WObj determines what frame Result will be related to. If not selected, wobj0 is used.

[\PrePDisp]

**Data type:** pose

Not implemented.

[\Limit]

**Data type:** num

If selected, an error will be flagged if the magnitude of the search result, Result, is larger than the value entered for the limit (in mm).

[\Resolution]

**Data type:** num

The resolution in mm. Default value is 0.5 mm. Maximum value is 1.5 mm.

[SchSpeed]

**Data type:** num

Search speed during execution of the search movements. If not used the default speed configured in the process configuration is used.

[\ShortStickOut]

**Data type:** switch

If used the position with the shortest stick-out is returned.

SearchName

**Data type:** string

The search will be assigned this identifying name. The name will accompany any error messages that are written to the event log.

[\TLoad]

**Data type:** loaddata

The argument \TLoad 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 argument \TLoad is used, then the loaddata in the current tooldata is not considered.

*Continues on next page*

## 9 RAPID references

---

### 9.1.2 SenseL - Contour sensing

*Continued*

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

---

#### Program execution

When executed, the robot makes a linear movement to the search start point, `SearchStart`. `WireSense` is activated and motion starts towards the search point, `SearchPoint`. The robot will continue past the search point for a total search distance described by twice the distance between `SearchStart` and `SearchPoint`. Once motion stops the displacement data, `Result`, is stored. This program displacement can later be used to shift programmed points using the RAPID instruction `PDispSet`.

---

#### Error handling

Fault	Menu message
Fault 1	Activation of WireSense failed
Fault 2	Search Failed
Fault 3	Part found prior search

##### Fault 1

If an error occurs when activating `WireSense`, a menu will appear with the following prompts:

##### Activation of WireSense failed

RETRY	Tries to search again with start point moved 50%
RETURN	Continues the program with default search result
ABORT	Sends error to calling routine

When `RETRY` is selected the start point of the search is shifted further from the part feature. This may give a good search result in cases where the part feature is unusually close, and the wire is touching the part feature at the beginning of a normal search. When `RETURN` is selected a default search result is used which will include any pre-offset included in the search instruction. A message will be logged in the event log.

##### Fault 2

If an error occurs during the search process, a menu will appear with the following prompts:

##### Search failed

RETRY	Tries to search again with start point moved 50%
RETURN	Continues the program with default search result
ABORT	Sends error to calling routine

When `RETRY` is selected the start point of the search is shifted further from the part feature. This may give a good search result in cases where the part feature is unusually close, and the wire is touching the part feature at the beginning of a

*Continues on next page*



normal search. When RETURN is selected a default search result is used which will include any pre-offset included in the search instruction. A message will be logged in the event log.

**Fault 3**

If for some reason the welder feedback that the edge is found before search begins, the following menu appears:

**Part detected at Search Start**

RETRY	Tries to search again with start point moved 50%
RETURN	Continues the program with default search result
ABORT	Sends error to calling routine

When RETRY is selected the start point of the search is shifted further from the part feature. This may give a good search result in cases where the part feature is unusually close, and the wire is touching the part feature at the beginning of a normal search. When RETURN is selected a default search result is used which will include any pre-offset included in the search instruction. A message will be logged in the event log.

If the optional argument `Limit` is selected and the magnitude of `peResult` is larger than the value entered for the limit, the following message appears:

**The search result is outside spec.**

Offset:=	[12.012,3.002,-5.013]
The magnitude of the offset :=	13.34
The present limit :=	10
OK	Continue with program execution
RAISE	Sends the error to calling routine

When OK is selected the search result is accepted regardless of magnitude. A message will be logged in the event log.

**Syntax**

SenseL

```
[ Result ':=' ] < expression (INOUT) of pose > ','
[ SearchStart ':=' ] < expression (IN) of robtarg > ','
[ SearchPoint ':=' ] < expression (IN) of robtarg >
[ '\ SearchEnd ':=' ] < expression (IN) of robtarg >
[ '\ SearchResult ':=' ] < expression (INOUT) of robtarg >
[ Speed ':=' ] < expression (IN) of speeddata > ','
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\ WObj ':=' ] < persistent (PERS) of wobjdata > ]
[ '\ PrePDisp ':=' ] < expression (IN) of pose > ]
[ '\ Resolution ':=' ] < expression (IN) of num > ]
[ '\ SchSpeed ':=' ] < expression (IN) of num > ]
[ '\ ShortStickOut ]
[ '\ SearchName ':=' ] < expression (IN) of string > ]
[ '\ TLoad ':=' ] < persistent (PERS) of loaddata > ] ';'

```

*Continues on next page*

## 9 RAPID references

---

### 9.1.2 SenseL - Contour sensing

*Continued*

---

#### Related information

For information about	See
PDispSet	<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>

### 9.1.3 SwitchWireSenseSettings - Switch WireSense signals and search speed

#### Usage

`SwitchWireSenseSettings` is an instruction used to switch the configuration of *WireSense - Standard Signals* and *WireSense Speeds* to be used for searching with *WireSense*. This can for example be used to change between multiple arc systems.

#### Basic examples

The following example illustrates the instruction `SwitchWireSenseSettings`.

#### Example 1

```
SwitchWireSenseSettings "WireSenseSig_R1", "WireSenseSpeed_R1";
```

#### Arguments

```
SwitchWireSenseSettings ( sWireSenseSignals sWireSenseSpeeds
                          [\WaitInpos] )
```

`sWireSenseSignals`

**Data type:** string

This argument specifies the *WireSense - Standard Signals* configuration instance that will be activated.

`sWireSenseSpeeds`

**Data type:** string

This argument specifies the *WireSense Speeds* configuration instance that will be activated.

`\WaitInpos`

**Data type:** num

If this argument is used, RAPID execution will wait the specified number of seconds for robot and external axes to come to a standstill.

#### Syntax

```
SwitchWireSenseSettings
  [sWireSenseSignals ':=' ] <expression (IN) of string>
  [sWireSenseSpeeds ':=' ] <expression (IN) of string>
  ['\ ' WaitInpos ':=' ] <expression (IN) of num> ;'
```

**This page is intentionally left blank**

# Index

## S

SearchEdge, 57

SenseL, 62

SwitchWireSenseSettings, 67







**ABB AB**

**Robotics & Discrete Automation**

S-721 68 VÄSTERÅS, Sweden

Telephone +46 (0) 21 344 400

**ABB AS**

**Robotics & Discrete Automation**

Nordlysvegen 7, N-4340 BRYNE, Norway

Box 265, N-4349 BRYNE, Norway

Telephone: +47 22 87 2000

**ABB Engineering (Shanghai) Ltd.**

Robotics & Discrete Automation

No. 4528 Kangxin Highway

PuDong New District

SHANGHAI 201319, China

Telephone: +86 21 6105 6666

**ABB Inc.**

**Robotics & Discrete Automation**

1250 Brown Road

Auburn Hills, MI 48326

USA

Telephone: +1 248 391 9000

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